On Automated Lesson Construction from Electronic Textbooks

Gultekin Ozsoyoglu, *Senior Member*, *IEEE Computer Society*, Nevzat H. Balkir, Z. Meral Ozsoyoglu, *Senior Member*, *IEEE Computer Society*, and Graham Cormode

Abstract—An *electronic book* may be viewed as an application with a multimedia database. We define an *electronic textbook* as an electronic book that is used in conjunction with instructional resources such as lectures. In this paper, we propose an electronic textbook data model with *topics*, *topic sources*, *metalinks* (relationships among topics), and *instructional modules* which are multimedia presentations possibly capturing real-life lectures of instructors. Using the data model, the system provides users a topic-guided multimedia lesson construction. This paper concentrates, in detail, on the use of one metalink type in lesson construction, namely, *prerequisite dependencies*, and provides a sound and complete axiomatization of prerequisite dependencies. We present a simple automated way of constructing lessons for users where the user lists a set of topic names (s)he is interested in, and the system automatically constructs and delivers the "best" user-tailored lesson as a multimedia presentation, where "best" is characterized in terms of both topic closures with respect to prerequisite dependencies and what the user knows about topics. We model and present sample lesson construction requests for users, discuss their complexity, and give algorithms that evaluate such requests. For expensive lesson construction requests, we list heuristics and empirically evaluate their performance. We also discuss the worst-case performance guarantees of lesson request algorithms.

Index Terms—Electronic textbooks, electronic textbook data models, instructional modules, topics, lesson construction, lesson construction heuristics, lesson complexity.

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 a hyperlinked table of contents, provide indexed search facilities on keywords, and occasionally have multimedia data such as images, maps, and audio/video streams. There are now also electronic versions of hardcopy textbooks on CDs or on the Web (e.g., see NetLibrary [22]). In this paper, we are interested in *electronic textbooks*, which we view to be electronic versions of hardcopy textbooks, enhanced by other instructional resources containing, among others, precaptured multimedia presentations about topics in the book. In this paper, our goal is to develop generic and domain-application-independent techniques for automated assembly of *lessons* from electronic textbooks. We

1. formalize concepts about electronic textbooks in particular and educational technology, in general,
2. relate data modeling and instructional design,
3. analyze the complexity of automated lesson construction from topics and instructional modules, and
4. give heuristics when it is infeasible to find the "best" lesson.

We identify three data-centric research issues for electronic textbooks: Defining

1. a very basic data model for electronic textbooks with instructional resources about topics,
2. simple ways of topic-based searching and extracting information from electronic textbooks for computer-naïve users, and
3. a query language for more advanced users to query the information in electronic textbooks.

This paper deals with the first two of the above-listed issues.

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 *instructional module*, and maintain it in the electronic textbook database. As an example, an instructional module can be an enhanced, tagged, annotated, and catalogued version of a course lecture, a tutorial, or a seminar. It may contain instructors' audio/video clips, students' audio/video clips interacting with instructors, whiteboards, animated data, slides, text, etc. [14]. This way, a content-based model of instructional modules is provided. The DBMS maintains users' knowledge levels on topics, and allows automated construction of user-tailored multimedia lessons from instructional modules. We define the notions of *teaching* and *learning* topics, and use them in defining lessons, which are multimedia presentations. In short, the electronic textbook application has the ability to

• *G. Ozsoyoglu and Z.M. Ozsoyoglu are with the Department of Electrical Engineering and Computer Science, Case Western Reserve Univeristy, Cleveland, OH 44106. E-mail: {tekin, ozsoy}@eecs.cwru.edu.*
• *N.H. Balkir is with Experian North America, 1089 Kingsdale, Hoffman Estates, IL 60194-2378.*
• *G. Cormode is with DIMACS Center, CoRE Building, 4th Floor, Rutgers University, 96 Frelinghuysen Road, Piscataway, NJ 08854. E-mail: graham@dimacs.rutgers.edu.*

Manuscript received 1 Aug. 2001; revised 16 July 2002; accepted 11 Dec. 2002.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 114633.

1) model its data and 2) automatically create lessons as multimedia presentations.

Our electronic textbook data model, adapted from the industry standard topic maps data model [34], contains

1. *topic entities* (or, simply *topics*) with topic names, topic types, topic detail levels, topic domains, topic hierarchies, etc. (which constitutes metadata),
2. *metalinks*, which are relationships between topics (also metadata),
3. *topic sources* which are parts of the textbook (e.g., paragraphs, or sections containing text, figures, tables, etc., of the actual textbook and, hence, constitute data. Also note that topic sources constitute the simplest form of instructional modules.),
4. instructional modules, and
5. explicit knowledge levels of users on each topic.

We investigate, in-depth, just one specific metalink type, namely, *prerequisite dependencies* that specify the order of presenting different topics at different knowledge levels.

Section 2 discusses related work. In Section 3, we list the main features of the electronic textbook data model to be used as a basis in the rest of the paper. Section 4 discusses the use of the data model in constructing lessons. In Section 5, we present the axiomatization of prerequisite dependencies. Section 6 discusses automated construction of the best lesson for a specific lesson construction request. Section 7 concludes our article.

2 Related Work

Advantages of using multimedia in learning, in general, have been reported in the literature [26], [6], [31], [7], [18], [27]. The Open eBook Forum (OeBF) [24] is an international trade and standards organization for the development of e-book applications. In [9], an example of a commercial electronic publishing system that provides online books is given. The BookWorks application combines full-text search and navigation tools, has full multimedia support with access to the Web, and allows users to organize their data. Net Library [22] is another online textbook system that allows users to keep personal notes in a database.

The Multibook effort (*iTeach* and *IT-Beankit projects*) [28], [32], [33] builds a Web-based adaptive hypermedia learning system (knowledge networks): It describes the resources by metadata, connects them by relations, and has an ontology containing all relevant concepts. It considers large user groups, user levels and learning strategies, gains information from interactions with users, and uses standardized content relations and metainformation to adaptively compile a selection from the set of available information units. A subset of the metainformation represents a "compiled lesson" at a high level, and is presented to the learner as a dynamically generated table of contents. IT-Beankit describes a three-tiered layered interactive approach for the visualization content according to the user's needs using metadata.

The Instructional Architect [25] project builds a suite of tools to enhance teachers' and students' use of "learning objects" within educational digital libraries. The emphasis is to define learning objects to support inquiry-based online learning, and to mass customize and personalize learning for instructional use.

The Walden's Paths project [35] develops tools for tailoring multimedia materials available on the Web and making them available to students. The two projects of this effort are the Path Authoring Tool and the Path Server. The Path Authoring Tool enables primary or secondary school teachers to create, modify, validate, and reuse paths over the Web and over paths that have been developed previously. The Path Server is the implementation means through which students use the paths.

ARIADNE projects [2] developed tools and methodologies for producing, managing, and reusing computer-based pedagogical elements and training curricula. The *Simulation Authoring Tool* allows users to develop models of generic, goal-oriented simulations in specified fields. Once a model is available, educators specify the scenario of their own exercise or simulation-based courseware. The *Questionnaire Authoring Tool* helps educators create, generate, and analyze computerized questionnaires, which can either be executed locally or remotely. The *Autoevaluation Exercise Authoring Tool* enables the student to solve a problem, with hypermedia access to pertinent information for understanding the learned concepts. There are also the *Pedagogic Hypertext Generator tool* and the *Video Clip Generator tool*.

Our proposal is distinguished from the above works by its data-centric view of electronic textbooks and its use of the topic-oriented data model with metalinks. To the best of our knowledge, our electronic textbook data model and topic closure-based automated lesson construction are both new contributions.

We now summarize the Topic Map data model, as described in [34]. The definition of a topic is very general: *A topic can be anything about which anything can be asserted by any means*. As an example, in the context of the encyclopedia, the country Spain or the city Rome are topics (about subjects). Topics are *typed* (e.g., type of the topic Rome is city) and have names. Topic names are also typed; e.g., base name (required), display name (optional), etc. Topic names have *scopes*, e.g., language, style, domain, etc. Topics have *occurrences (sources)* within addressable information resources. For example, a topic can be *described* in a monograph, *depicted* in a video or a picture, or *mentioned* in the context of something else. Moreover, each occurrence is typed using the notion of *occurrence role*. A *topic association* specifies a relationship between two or more topics. For example, topic Rome *is-in* topic Italy; topic Tom Robbins *was-born-in* topic US, etc. A *topic map* is a structure, perhaps a file or a database or an XML document, which contains a topic data model, together with occurrences, types, contexts, and associations. From the descriptions above, a Topic Map data model is similar to the Entity-Relationship model specialized for the abstract domain of topics and topic-related information.

The Resource Description Framework (RDF) [19] is designed to describe Web information sources by attaching metadata specified in XML. RDF is a graph-based information model, and consists of a set of statements, represented as triples. A triple denotes an edge between two nodes, and has a property name (edge), a resource (node), and a value (node). A resource can be anything from a Web document to an abstract notion. A value can be a resource or a literal. A literal is an atomic type, e.g., real, integer, string. RDF Schema [4] defines a type system for RDF, and is similar to the type systems of object-oriented programming languages such as Java. RDF Schema allows the definition of *classes* for resources and property types. The resource *Class* is used to type resources, and the resource *Type* is used to type properties. Various properties such as *SubClassOf, SubPropertyOf, isDefinedBy, seeAlso*, and *type* are available. Various constraints on resources (e.g., *ConstraintResource*), and on properties (e.g., *range* and *domain* constraints, which restrict the range and the domain of a property, respectively) also are available.

3 Electronic Textbook Data Model

Clearly, one may model very different aspects of electronic textbooks. In what follows, we only model topics, topic sources with detail levels, metalinks, users' knowledge levels, multimedia instructional modules, and lessons. Also, note that, in choosing the features of the data model (and in lesson requests later in Section 6), we have tried to be as generic as possible. Nevertheless, we have made decisions (such as the chosen set of topic attributes, etc.) based on our experiences as educators in computer science, which may not apply to textbooks in other disciplines, or may not match with the experiences of other educators.

3.1 Topic Entities and Topic Sources

The topic entity constitutes metadata, and we assume that the electronic textbook is modeled in terms of topics. We envision that the topics for an electronic book are derived using automated techniques from information retrieval [29] (first stage), and then revised using data mining techniques [17] (second stage). For example, one approach in information retrieval is to use the vector space model, e.g., the TF-IDF (Term Frequency-Inverse Domain Frequency) model, and to create a "word list" [29], [12] which may be viewed as topics. One can then use data mining techniques to further locate "semantically meaningful" topics and to reduce the number of topics obtained in the first stage. In [23], we have used this approach to generate about 43,000 topics in the first stage and 10,000 topics in the second stage from the titles of 221,000 conference and journal papers in the DBLP Bibliography [13]. Alternatively, topics also may be generated manually by experts, e.g., the author of the textbook, or an instructor who uses the textbook.

The notion of topic in hardcopy books (and textbooks) is not new: Topics in their simplest form as keywords appear as part of a book index at the end of each hardcopy book. We expand this notion as a topic entity, where a topic entity has a *unique topic identifier*, a *topic name* (Tname) (arbitrarily specified phrases or words, e.g., "basics of data models" that characterize the data in the electronic book), a *topic type* (Ttype), e.g., "database textbook author," "research article author," "survey," "introductory chapter") specifying the type of the topic, a *topic domain* (Tdomain) (e.g., "introductory database information for CS undergraduates") specifying the domain within which the topic is to be used, and an integer-valued maximum detail level. The attributes (Tname, Ttype, Tdomain) constitute a key for the topic entity. And, the Tid attribute is also a key for topics. A topic is instantiated by a *topic source* in the electronic book, which is part of the text such as a paragraph, multiple paragraphs from different parts of the book, or a section in the book. Each topic source has an integer-valued *topic detail level* describing how advanced the level of the topic source is. 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 $i > j$.

Topics have topic sources, which refer to parts of the electronic textbook. For example, for the topic "relational query languages," Section 3.1 of the electronic textbook may be a topic source at detail level 1 (beginner). Or, Section 3.2 of the textbook may be a topic source at detail level 2 (intermediate) for the topic "SQL query specification," etc. For a given topic, it is possible to have multiple topic sources at the same detail level.

3.2 Metalinks and Topic Closure

Topic Metalinks represent relationships among topics. Any relationship involving topics deemed suitable by an expert in the field can be a topic metalink. For instance, *Prerequisite, RelatedTo*, and *WrittenBy* are possible metalink types. *SubTopicOf* and *SuperTopicOf* metalink types together would represent a topic composition hierarchy. Metalinks represent relationships among topics, not topic sources. Therefore, they are *metarelationships*, hence, our choice of the term "metalink." Metalink types are usually recursive relationships, and may have as attributes types, roles, domains, etc. Each metalink type has a *signature*, e.g., the *RelatedToPapers* metalink type has the signature

$$RelatedToPapers: \ \text{PaperId} \rightarrow \text{PaperIdI}.$$

In this paper, we concentrate solely on one metalink type, namely, the *Prerequisite* metalink type with the signature *Prerequisite*: SetOf Topic $\rightarrow$ SetOf Topic.

Some hardcopy textbooks provide "*dependency diagrams*" in an attempt to help instructors and 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 metalink type of *prerequisite dependencies* among topics, and use it (and the existing instructional modules in the database) for automated lesson (i.e., multimedia presentation) construction. For example, the prerequisite dependency "the topic *relational algebra* (*ra*) should be taught after teaching the topic *relational data model* (*rm*)" implies that the dependency $ra \rightarrow rm$ holds. In a given course, if topic y is a prerequisite to another topic x (i.e., $x \rightarrow y$ 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 topic x, and those instructional modules that correspond to topic y

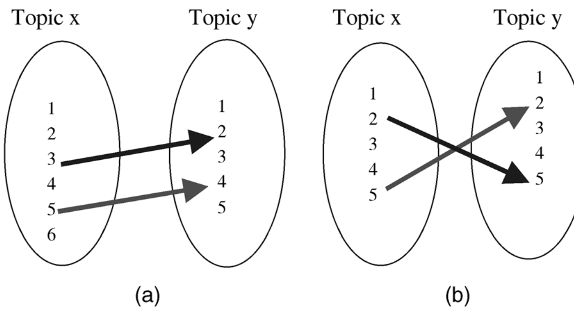

Fig. 1. Prerequisite dependencies.

have not yet been rendered, then the constructed lesson should also have the topic sources and the instructional modules that correspond to topic y. Note that we actually use prerequisite dependencies among topics at different detail levels, e.g., the prerequisite dependency $\mathrm{ra}^4 \rightarrow \mathrm{rm}^1$ 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*." As an illustration, consider the interpretation of the prerequisite dependency shown in Fig. 1: If topic x at level 3 is presented in a lesson, say S, then topic y at level 2 or a higher level must either 1) be in S before x, or 2) the user knows y at level 2 already. That is, the prerequisite dependency $\mathrm{x}^3 \rightarrow \mathrm{y}^2$ holds. Note that, in this prerequisite dependency, the right-hand side specifies "topic y at level 2 or a level higher than 2," as opposed to "topic y at level 2." Also, note that new constraints such as the consistency constraint as illustrated in Fig. 1 (e.g., forbidding the crossing over two dependencies), may be needed [3].

As stated before, metalink types are usually recursive. For example, the metalink type *RelatedTo* is both transitive and reflexive. Metalink type *IsIn* is transitive, but not reflexive; metalink type *SubTopicOf* is transitive. In an electronic textbook application, when a user lists a set X of topics and asks for topic sources of topics in X as well as others that are *prerequisite(s)* of topics in X, we need to take the "topic closure" of the topic set X with respect to the recursive metalink type *Prerequisite*. We emphasize the notion of *Topic Closure* with respect to prerequisite dependencies, in order to return query results that satisfy all prerequisite dependencies. Given a set X of topics, the query response will include the topic closure X^+, which is formed of all topics that are logically implied by the initial set X. Clearly, computing topic closures requires a sound and complete set of prerequisite dependency axioms, and a polynomial-time algorithm that computes the topic closure using the axioms. We discuss the axiomatization of prerequisite dependencies and the computation of topic closures in Section 5.

3.3 Personalization: User Profiles

User profiles contain the knowledge levels of users about topics as well as other information such as users' preferences. Other information kept in user profiles may be sections (chapters, examples, pictures, etc.) of electronic books that are read or viewed by users, the number of times each section is taught, and the time spent on sections. To simplify our presentation, in this paper, we only deal with one piece of user profile information: users' knowledge levels on topics. For a given user and a topic, the knowledge level of the user on the topic (zero, originally) is kept in the user profile.

3.4 Multimedia Instructional Modules

An instructional module is a multimedia presentation that describes and illustrates one or more topics. We assume that the electronic textbook has a number of instructional modules on topics. An instructional module is a synchronized multimedia presentation which contains audio/video segments of instructors, students, teaching assistants, as well as images, text, animation, and whiteboards used by instructors and students, etc. As an example, a specific instructional module M_3 may exist for the topic name "Instructor John Doe explaining SQL Triggers operator." In Fig. 3, using the horizontal x-axis as a timeline, an instructional module is illustrated. Possibly, an instructional module is captured in real-time from a live lecture session in an automated manner (we have built such a tool in an electronic classroom project [14]). 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., and entered into the database. In its simplest form, an instructional module may consist of a topic source. We expect that the process of instructional module creation is a labor-intensive but one-time task.

There may be various relationships between topics and instructional modules as topic sources, e.g., mappings between instructional modules and topics as illustrated in Fig. 2. For the sake of simplicity, we assume that each topic maps to a set of modules. In practice, a topic may only map to parts of a module, e.g., "the first 20 minutes of the instructional module *M* is about the topic *Databases*." Next, we define the notions of teaching and learning.

Definition 1 (*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.*

Thus, in this paper, we assume that users are taught a topic if they view the corresponding lesson once. How do we model as to whether users "learn" a topic? A common measure for learning is the concept of "testing:" One may assume that, 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 predefined 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 textbook users, given a topic, passing the associated test constitutes "learning" the topic at that level. Thus, we will not deal with learning at different taxonomy levels, as specified by Bloom [5].

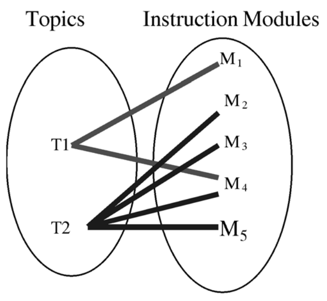

Fig. 2. Mappings between topics and instructional modules.

Definition 2 (*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.*

4 Making Use of the Data Model

4.1 Lesson Construction Based on Prerequisite Topic Closures

Users request multimedia *lessons* from the electronic textbook application where each lesson contains one or more instruction instructional modules. Thus, a lesson is also a multimedia presentation. The electronic textbook application is to construct from instructional modules "semantically coherent" lessons that satisfy the prerequisite dependencies: Given a set X of topics by a user, the sytem finds the closure X^+ of all topics that contains all the prerequisite topics of X (i.e., topic closure with respect to prerequisite dependencies), eliminates from X^+ those topics that are already known by users to obtain the set T of topics, and prepares a lesson L from the instructional modules for the topics in T.

All the topics in the topic set T must be covered by the lesson L. We say that lesson L containing a set of instructional modules *covers* topic t at level i if L contains all the instructional modules in the mapping from the topic t at level i to the set of instructional modules.

For the sake of simplicity, in this paper, we assume that there is a total ordering of all the instructional modules in the database so that, for a given lesson L containing a number of instructional modules to be rendered (i.e., played out), the modules in L are ordered into a sequence. Thus, we assume that any chosen sets of instructional modules are always ordered by this total ordering in order to form a lesson. In reality, a lesson as a multimedia presentation can be formed by selectively choosing or eliminating different media in the chosen instructional modules, and by "merging" instructional modules using height and length constraints [15], [16]. The resulting multimedia presentation in such a case can be viewed as a directed acyclic graph with nodes representing the media and edges representing the presentation ordering of media.

Section 6 deals with lesson construction requests based on prerequisite topic closures.

4.2 Topic Closures with Regular Expressions and Values, and Querying Topics

In this paper, we assume that a dependency (i.e., a metalink) between topics either exists and has to be employed for topic closure, or it does not exist. In reality, dependencies have relative "significance" or "importance" or "validity" values, e.g., topic A is a prerequisite of topic B with the importance value of 0.7, where the values are normalized to the range [0, 1]. Then, users' requests are in the form of "give all the prerequisites of the topic set X with an importance value above a given threshold". In a recent work, we call such requests *threshold-based closure requests* [23]. An alternative request is "give the k highest-scored prerequisites of the topic set X," which we call as *top-k closure requests* [23], [10], [11]).

One can generalize the notion of topic closure with respect to a single dependency type into a topic closure with respect to a regular expression of multiple dependency types. For example, assume that we have 1) the dependency type *ResearchArticlesOf* with the signature *ResearchArticlesOf*: Topic → SetOfPaper, which, given a topic, lists the related research articles in the book, and 2) the dependency type *AuthorOf* with signature *AuthorOf*: Paper → SetOfAuthor, which, given a reference paper, returns the authors of that paper. And, assume that the function *TopicSourcesOf* takes a set of topics and returns the associated topic sources, i.e., parts of the actual textbook. *TopicSourcesOf* has the signature *TopicSourcesOf*: SetOfTopic → SetOfTopicSource. Then, given a set X of topics by users, the topic closure X with respect to the regular expression *TopicSourcesOf* (*AuthorOf* (*ReferencesOf* (*Prerequisites** (X)))) returns the topic sources of the authors of papers referenced by the topics in X and *all* of their prerequisites (here, * denotes the Kleene star). Note that here, papers and authors are also topics. In our recent work [23], we define such a topic closure operator within the

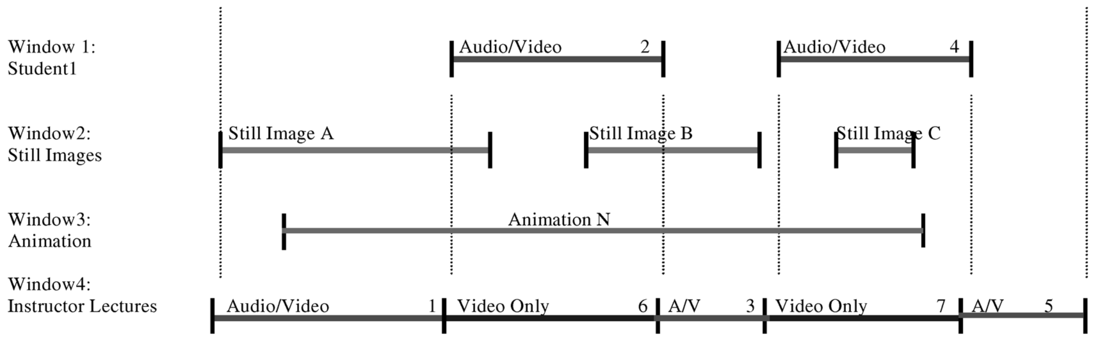

Fig. 3. Multimedia presentation as an instructional module.

context of an object-relational model of topics with importance values, metalinks with importance values, and topic sources, and investigate its evaluation techniques.

In [1], [23], we described an SQL-like language for retrieving topic sources (in our case, parts of electronic textbooks), and discussed its query processing issues. These languages can easily be extended into electronic textbooks, which, for space limitations, we do not discuss here.

5 Prerequisite Dependencies and Topic Closure Computation

This section discusses the axiomatization of prerequisite dependencies, the sole metalink type investigated in this paper, and presents polynomial-time topic closure algorithms. In order to compute topic closures with respect to prerequisite dependencies, we classify prerequisite dependencies into four classes in terms of cyclicity/acyclicity and whether a dependency is left-hand side-decomposable or not. For each class, in Sections 5.1, 5.2, 5.3, and 5.4, we discuss how to compute topic closures. In Section 5.4, for acyclic and nondecomposable prerequisite dependencies, we define two new axioms, prove their soundness and completeness, present the topic closure algorithm, and prove its correctness.

We classify prerequisite dependencies as 1) *cyclic* (e.g., $\mathrm{x} \rightarrow \mathrm{x}$ forms a trivial cycle; $\mathrm{x} \rightarrow \mathrm{y}$ and $\mathrm{y} \rightarrow \mathrm{x}$ form a nontrivial cycle) or *acyclic* (e.g., trivial or nontrivial cycles are not allowed), 2) (left-hand side) *decomposable* (e.g., $\mathrm{xy} \rightarrow \mathrm{z}$ is equivalent to $\mathrm{x} \rightarrow \mathrm{z}$ and $\mathrm{y} \rightarrow \mathrm{z}$) or *nondecomposable* (e.g., $\mathrm{xy} \rightarrow \mathrm{z}$ is not equivalent to $\mathrm{x} \rightarrow \mathrm{z}$ and $\mathrm{y} \rightarrow \mathrm{z}$). Next, we define what it means for a set of dependencies to be acyclic.

Defintion 3. *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* $\mathrm{F} = \{\mathrm{x} \rightarrow \mathrm{y}, \mathrm{y} \rightarrow \mathrm{z}, \mathrm{z} \rightarrow \mathrm{x}\}$ *is strongly cyclic. This still holds if x represents a set of topics.*

Definition 4. *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* $\mathrm{F} = \{\mathrm{wx} \rightarrow \mathrm{y}, \mathrm{yz} \rightarrow \mathrm{V}, \mathrm{V} \rightarrow \mathrm{w}\}$ *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 the absence of strong cycles.

The simplest prerequisite dependency model commonly used in hardcopy textbooks allows only acyclic and decomposable prerequisite dependencies. However, when the semantics of applications require or when dependencies are created in automated ways using data mining or information retrieval techniques, one also can have electronic textbook environments in which prerequisite dependencies are cyclic and/or nondecomposable [3]. Consider the case of a cycle of three prerequisite dependencies, namely, $\mathrm{x} \rightarrow \mathrm{y}$, $\mathrm{y} \rightarrow \mathrm{z}$, $\mathrm{z} \rightarrow \mathrm{x}$, among topics x, y, and z. We interpret this cycle as "in any lesson request having one of topics x, y, or z, the instructional 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 $\mathrm{ab} \rightarrow \mathrm{c}$ 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 $\mathrm{ab} \rightarrow \mathrm{c}$ is nondecomposable if $\mathrm{ab} \rightarrow \mathrm{c}$ does not imply that $\mathrm{a} \rightarrow \mathrm{c}$ and $\mathrm{b} \rightarrow \mathrm{c}$. (Note that the reverse is always true, i.e., the prerequisite dependencies $\mathrm{a} \rightarrow \mathrm{c}$ and $\mathrm{b} \rightarrow \mathrm{c}$ always imply the prerequisite dependency $\mathrm{ab} \rightarrow \mathrm{c}$). Below, we illustrate a case of nondecomposable prerequisite dependencies.

Example 1. Consider the three topics, 1) "Training for high school downhill ski racing," 2) "Training for state slalom downhill ski racing," and 3) "Advanced Strength Training." For learning about either topic 1) (i.e., high school racing) or topic 2) (i.e., state racing) alone, one does not need to learn about 3) (Advanced Strength Training), but for the two together, one needs the advanced strength training—purportedly because there will be too many races in a season so that one needs to be much stronger physically to handle both at the same time. That is, $\mathrm{ab} \rightarrow \mathrm{c}$ is not equivalent to $\mathrm{a} \rightarrow \mathrm{c}$ and $\mathrm{b} \rightarrow \mathrm{c}$. The nondecomposibility requirement in $\mathrm{ab} \rightarrow \mathrm{c}$ states that, to learn a and b, one first needs to learn c while, for learning a or b alone, it is not necessary to learn c first. Here, a and b are not independent in the sense that learning one of the two alone does not need c, but learning both pushes the requirement high enough for c to become a prerequisite.

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

5.1 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 [36]. One can then compute P^{+}, the 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 [36] 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 $\mathrm{xy} \rightarrow \mathrm{z}$. 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 $\mathrm{xy} \rightarrow \mathrm{z}$. One possible solution to this problem is to utilize the user profiles. Since user profiles contain users' knowledge about all instructional modules that are taught to the user, topic z coverage can be added to the second lesson request when topic y is requested (since it is known in the user profile that x was taught to the user before).

```
Algorithm TC:
1. $X^{(0)}$ is set to empty
2. $X^{(i+1)}$ is $X^{(i)} \cup \{y\}$ such that there is a dependency in F of the form $X_i \rightarrow y$, where $X_i \subseteq X \cup X^{(i)}$ and $y \notin X$.
```

Fig. 4. Topic closure algorithm for acyclic and nondecomposable prerequisite dependencies.

5.2 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, the augmentation axiom of functional dependencies does not apply either [3]. 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 $a \rightarrow b$. 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.

5.3 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 5.2 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 directed edge from node a to node b iff P' contains the prerequisite dependency $a \rightarrow b$. 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.

5.4 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 5.2 and 5.3, the transitive closure of such a graph would not capture all the dependencies. Consider, for example, the set of dependencies $\{x \rightarrow a, ab \rightarrow c\}$, 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. We also 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. Next, we give a sound and complete axiomatization for this case, and describe a topic closure algorithm. We first define two axioms.

Definition 5. Pseudotransitivity axiom*: If $x \rightarrow y$ and $wy \rightarrow z$, then $wx \rightarrow z$.*

Definition 6. Split/join axiom*: If $x \rightarrow ab$, then $x \rightarrow a$ and $x \rightarrow b$, and vice-versa.*

Proofs of all lemmas and theorems are given in the Appendix which can be found on the Computer Society Digital Library at http://computer.org/tkde/archives.htm.

Theorem 1. *The pseudotransitivity and split/join axioms are sound and complete.*

Algorithm TC in Fig. 4, which is similar to the algorithm to compute the closure of functional dependencies [36], computes the closure of a set of topics X.

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

Lemma 1. *Algorithm TC correctly computes X^+.*

Algorithm TC is functionally equivalent to that described for cyclic and nondecomposable dependencies of Section 5.1, and runs in time linear in the size of the representation of the dependencies. 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.*

6 AUTOMATED LESSON CONSTRUCTION

What would be the common features of lesson requests? While this question needs to be answered empirically when electronic textbooks with lessons are commonplace, in our opinion, most lesson requests will have the following features.

1. *Lessons about topics*. An example request is "prepare a lesson on topics x at level i and y at level j." Clearly, this corresponds to computing the closure X^+ of the topic set $X = \{x^i, y^j\}$ which, from Section 5 above, we know to be O(N) for all four prerequisite dependency types, where N is the number of topics in the database and the length of a dependency

```
Request 2 Algorithm:
CurrentLevel := 0;
time := 0;
repeat
        begin
        CurrentLevel = CurrentLevel + 1;
        Find the topic closure X+ of topics in X for each topic at the CurrentLevel;
        Eliminate from X+ the topics that are already known by the user to obtain Y;
        time := sum of the time to present the topics in Y ;
        end
until time>tUB;(time exceeds the upper bound tUB)
```

Fig. 5. Algorithm for evaluating Request 2.

encoding is one. We view this request to be the most basic lesson request.

2. *Lessons with the highest detail levels of topics.*
3. *Lessons with as many covered topics as possible.*
4. *Lessons with time constraints (e.g., 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."
5. *Lessons in which users prioritize topics and request an optimization of the total topic priorities in the lesson.*
6. *Lessons tied to a quantifiable increase, say integer k, in the user's knowledge level(s) on a given topic.* An example is "prepare a 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")."
7. *Lessons constructed around tests, assignments, quizzes, chapters, etc.* An example is "Prepare a lesson on (the topics covered in) the current assignment." We assume that there are mappings from tests, quizzes, assignments, chapters, etc., into topics; and these requests reduce to requests of type 1 above.
8. *Lessons constructed around instructional module contents.* An example is "prepare a lesson that contains all the audio/video question streams of student John Smith." One can characterize such lesson requests as requests of the type "prepare a lesson request that contains instructional modules (or their media components) satisfying a Boolean condition C about module contents." Such lesson requests require an extension to our data model in that the content semantics of instructional modules need to be modeled [20], [21]. In this paper, we will not be dealing with such lesson requests.

In the next section, we characterize and classify a number of automated lesson construction requests that represent combinations of lessons of type 2 through 6 above, and discuss how they can be evaluated.

6.1 Automated Lesson Construction Requests

The lesson construction requests described in this section have different solutions for each prerequisite dependency case described in Section 5. The differences between the solutions are in the handling of cycles. Topic closure calculation is included in deciding the complexity of the algorithms: Topic closure can be calculated with the same algorithm 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 1) the user's knowledge levels for topics, 2) the set X of topics, and 3) prerequisite dependencies in the electronic textbook, produce a lesson that covers topics X at the highest levels.

Request 1 can be evaluated by a polynomial-time algorithm as follows: 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 instructional modules that map to the topics that are left in X^+, and to order them (using their total ordering) to obtain a lesson. Step 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 1) the user's knowledge levels for topics, 2) prerequisite dependencies in the electronic textbook, 3) the set X of topics, and 4) an upper bound t_{UB} on the lesson timelength, produce within the time bound t_{UB} a lesson that covers all the topics in X where all the topics are at the same and the highest possible levels.

Request 2 can also be evaluated by a polynomial-time algorithm, which is given in Fig. 5. The complexity of the algorithm is O(LN), where N is the number of topics in the database and L is the maximum number of detail levels.

Lesson Request 3. Given 1) the user's knowledge levels for topics, 2) the set X of topics and priorities attached to topics in X, 3) prerequisite dependencies in the electronic textbook, and 4) 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 an NP-Complete problem.*

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 1) the user's knowledge levels for topics, 2) the set X of topics, 3) prerequisite dependencies in the electronic textbook, and 4) an upper bound t_{UB} on the lesson timelength, produce a lesson of duration t_{UB} or less that covers as many of the topics in X as possible.

Theorem 3. *Request 4 is NP-Complete.*

Section 6.2 describes four heuristics, proposed for evaluating Requests 3 and 4, and Section 6.3 gives the evaluation of the four heuristics for Request 4. A heuristic

```
Request 4 Algorithm:
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 $\cup$ y;
        time := time + time of y;
        end
until time > $t_{UB}$;
```

Fig. 6. Algorithm for evaluating Request 4.

algorithm, which is linear in the number of topics and uses the heuristics described in Section 6.2, is shown in Fig. 6. Lesson construction requests above dealt with constructing learning lessons with no tests. The request below is for constructing a teaching lesson with tests, where the user's knowledge levels about topics are evaluated.

Lesson Request 5. Given 1) the user's knowledge levels for topics, 2) prerequisite dependencies in the electronic textbook, and 3) an upper bound t_{UB} on the lesson time length, produce a lesson with tests and 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.*

An approximate algorithm similar to the one in Fig. 6 can also be used for evaluating Request 5. Other lesson requests and their evaluation algorithms can be found at [3].

6.2 Heuristics for Expensive Lesson Requests

Next, for NP-complete lesson requests such as Requests 3 through 5 of Section 6.1, we propose and empirically evaluate four different heuristics.

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 instructional 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 2. 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 instructional modules of all topics at all levels take the same amount of time, say t, to present (e.g., a^4 takes 4*t 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 $e^3 \rightarrow f^2$. Using Table 1, 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 instructional 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 3. Assume that the knowledge level of the 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

TABLE 1
Prerequisites for Example 2

To include:	*We need to include:*
a^4	b^5
b^5	-
c^4	b^5 and a^4
d^6	b^3
e^3	f^2
f^2	-

TABLE 2
Prerequisites for Example 5

To include:	*We need to include*
a^5	b^6 and f^4
b^6	f^4
c^5	a^5, b^6 and f^4
d^6	b^6 and f^4
e^3	f^4
f^4	-

is not known by the user, and add the corresponding instructional modules into the lesson being constructed.

The motivation for the 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 the LDL, including more prerequisites in a lesson increases the chances of other topics in X to be included into the lesson.

Example 4. Assume that the knowledge level of the 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 the 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 instructional 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 5. 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 5*t 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 duration of at least 22t. Clearly, including f as the first topic into the solution by the heuristic LNP is a good choice.

Finally, we considered the Random Selection (RS) Heuristic that chooses topics randomly from the set of topics unknown to the user, and adds the corresponding instruction modules into the lesson being constructed. The RS heuristic performed poorly; 70 percent (or more) worse with large (greater than 10,000) number of topics, and with a large number of topic details (greater than 15). Hence, to save space, we did not report the performance of RS heuristic in this paper.

TABLE 3
Heuristics

Heuristic Name	Acronym
Best Base Heuristic	BB
Lowest Detail Level Heuristic	LDL
Highest Number of Detail Levels Heuristic	HNDL
Lowest Number of Prerequisites Heuristic	LNP

6.3 Evaluating the Expected Case Behavior of Heuristics

This section describes the experiments conducted to evaluate the expected performances of the four heuristics described above. The heuristics and the acronyms that are used in the rest of this section are given in Table 3.

To evaluate the heuristics, we have simulated an electronic textbook.[1] In this environment, students decide on the length and the content of a presentation about a lecture using various constraints.

6.3.1 Electronic Textbook Environment

We have used four components (users, topics, dependencies, and requests) to model the electronic textbook environment, summarized in Table 4. Table 5 lists the parameters and their respective ranges in our simulations.

Users: This component represents the users of the system, and contains information about users' knowledge of topics. For each topic, information about the highest detail level known by the user is kept.

Topics: The number of detail levels for each topic is controlled by the topic depth parameter. Also, information about the length (in time) of the instructional module corresponding to each topic at a given detail level is stored in this component. In our simulation, the length of an instructional module for a topic at a given detail level varied between 1 minute and 15 minutes, and the number of topics varied between 200 to 2,000.

Dependencies: These components correspond to prerequisite dependencies which have the form $x^k \rightarrow y^r$, where x and y are topics, and k and r are detail levels. The number of dependencies in each simulation run varied depending on the number of topics and number of detail levels within a topic. We have chosen to generate the dependencies randomly, with no structure (e.g., hierarchical) among them (which may or may not hold, in practice, and remains to be empirically verified).

1. Experiments with real electronic textbooks would have been ideal. However, we did not have access to electronic textbooks and could not get permission from publishers. In any case, we think that a simulation that allows changes to its electronic book parameters provides flexibility that experiments from a single electronic book or few books do not provide.

TABLE 4
Components of the Electronic Textbook Simulation

Component	Parameters
user	amount of knowledge about each topic and detail level
topic	topic depth, length of instructional module for a topic at a given detail level in minutes.
dependency	topic name and detail level, specifying where the dependency originates and ends
request	a set of topic and detail level two-tuples that define a user request

TABLE 5
Parameters of the Simulation Environment and Their Corresponding Ranges

Component	Parameters
# of topics	200 – 2000
topic depth (# of detail levels)	2 – 20
length of an instructional module for a topic at a given detail level	1-15 minutes
# of dependencies in a given run	20 – 20000
size of a request	2 – 20 topics
presentation length	10 – 120 minutes

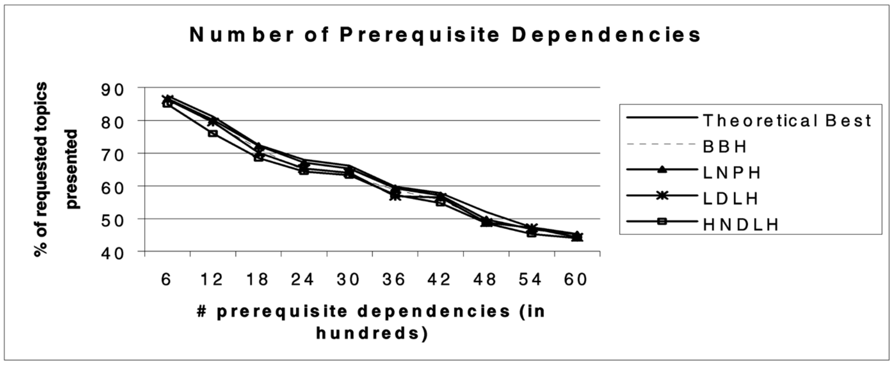

Fig. 7. Effects of changing the number of prerequisite dependencies.

The experimental parameter ranges of Table 5 are in line with our experience: In [23], we generated topics for the titles of journal and conference papers from the DBLP bibliography [13]. Using the titles of 91,000 journal papers and 132,000 conference papers, after eliminating stop-words such as words like "the," "a," "of," etc., from the words in each title and after stemming, we gathered 43,000 topics (as a word list or vocabulary for paper titles). We used the TF-IDF (Term Frequency-Inverse Domain Frequency) vector space model with the cosine similarity function [29], [12], and had, on the average, about 5 to 15 dependencies (metalinks) with "significant" importance values, emanating from each topic. In the next stage, in an effort to reduce topic and metalink counts, we applied rule-based data mining techniques to the topics and dependencies at hand, and, without losing too much of the richness of the data model, we reduced the number of topics in the range of 20 percent to 5 percent of the original number of 43,000 topics. Using 5 percent as the basis, the number of topics is 2,000 for our simulation. Using the ratio of 10 metalinks per topic, one can use the range of 200 (manually generated) to 20,000 (automatically generated) dependencies. These findings confirm our choice of parameter ranges as listed in Table 5.

Requests: Users increase their knowledge levels by lesson requests from the system. Requests contain a set of 2-tuples of topics and detail levels. The size of requests in our simulation varied between two topics and 20 topics.

6.3.2 Experimental Results for Lesson Request 4

The first set of results (Fig. 7) shows the effects of changing the number of prerequisite dependencies. In this experiment, we have kept the following parameters constant: the number of topics: 1,000, topic depth: 12, instructional module length: 10 minutes, presentation length: 60 minutes, and size of requests: 10 topics. There are five curves displayed on Fig. 7; one for each of the four heuristics and one for the theoretical best for the percentage of requested topics presented. For all five curves, 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 percent of the

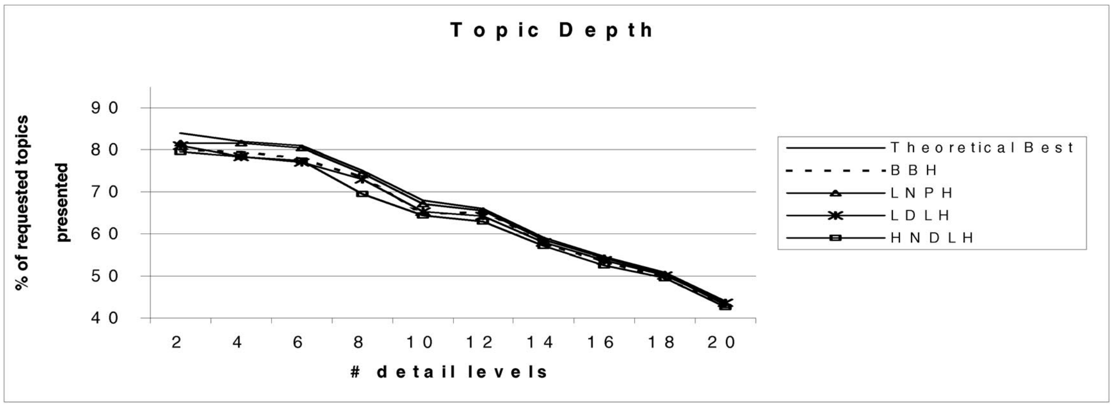

Fig. 8. Effects of changing the number of detail levels.

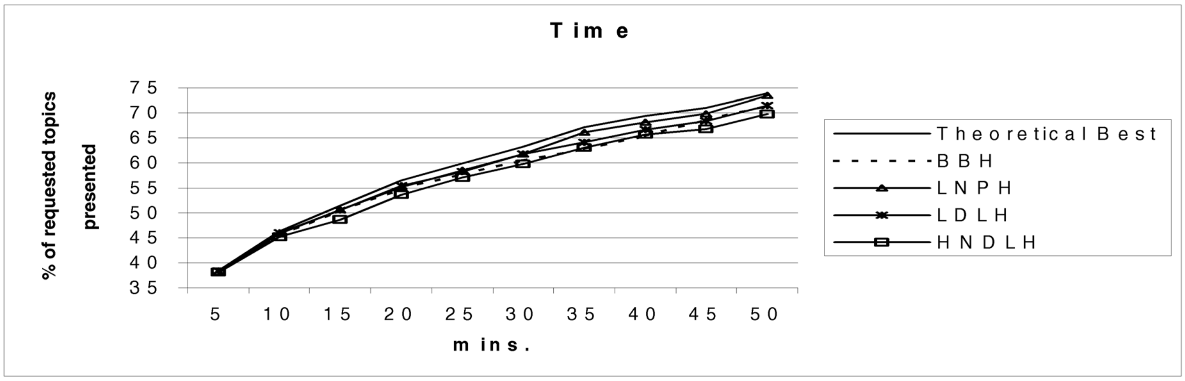

Fig. 9. Effects of changing the time upper bound of a lesson.

theoretical maximum. Among the heuristics, LNP performed the best, while HNDL performed the poorest.

Fig. 8 shows the effect of the topic depth (i.e., the number of detail levels) on the percent of requested topics presented. In this experiment, we have kept the following parameters constant: number of topics: 1,000, instructional module length: 10 minutes, presentation length: 60 minutes, and size of requests: 10 topics. The number of prerequisite dependencies (400-4,000) is changed proportionally to the change in the number of detail levels (2-20). The results are similar to prerequisite dependency results. As the topic depth increases, topics at higher detail levels are included into the requests. Instructional modules for 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.

Fig. 9 illustrates the effect of increasing the time upper bound t_{UB} on the presentation. In this experiment, we have kept the following parameters constant: number of topics: 1,000, topic depth: 12, instructional module length 10: minutes, number of prerequisite dependencies: 2,400, and size 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 has no effect on the performance of the heuristics, which can be observed in Fig. 10. In this experiment, we have kept the following parameters constant: topic depth: 12, instructional module length: 10 minutes, presentation length: 60 minutes, and size of requests: 10 topics.

Changing the request length (i.e., the number of topics in X) has an effect similar to changing the prerequisite dependencies or changing the topic depth, as shown in Fig. 11. In this experiment, we have kept the following parameters constant: number of topics: 1,000, topic depth: 12, instructional module length: 10 minutes, presentation length: 60 minutes, and the number of prerequisite dependencies: 2,400. Since the time limit on the lesson does not change, the percentage of requested topics that are presented decreases. The 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.

Lesson request calculation times of all four heuristics, assuming a main-memory-based representation of topics and dependencies, have been about the same (in reality, topics and dependencies would be disk-resident). As the length of the lesson increases, the time to calculate the best solution increases exponentially. When the lesson includes 18 topics, using a simulation implemented in C and running on a Pentium 4 with Windows NT operating system, *all* heuristic

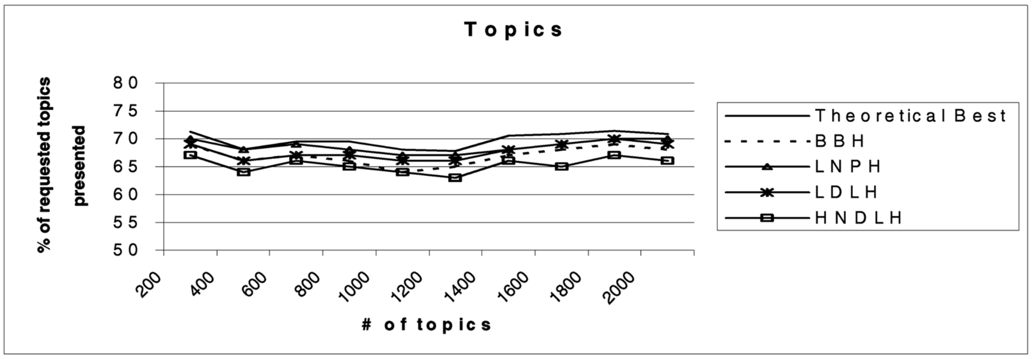

Fig. 10. Changing the number of topics.

algorithms produced a solution under 5 milliseconds, while it took more than 167 seconds for the best solution by enumeration. Clearly, as the request length increases, heuristic solutions become a must for an efficient implementation. Thus, all of the four heuristics perform well with results that are within 7 percent of the best solution (we have computed the best lesson by enumeration). Similar results can be shown for lesson requests 3 and 5.

6.4 Worst-Case Performance Guarantees 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. For worst-case performance guarantees of lesson request algorithms, we now 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, a, on the left side of the bipartite graph, G. The length of the instructional module for this topic (i.e., the *cost* of the topic) is set to zero. We also create a topic, a', on the right-hand side of the bipartite graph whose cost is that of the length of the instructional module for the topic. We initialize F', the new set of dependencies, to be $a \rightarrow a'$. We then add dependencies to F' such that $a \rightarrow b'$ for each $b \in X^+, X = \{x\}$, derivable from the original set F of dependencies. In our graph, we shall represent this by putting an edge in G for each dependency in F', linking the zero-weight topics on the left to their weighted prerequisites on the right. 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 b not in X has closure $\{b\}^+$ such that no member of $\{b\}^+$ is in X, then we can replace the whole of $\{b\}^+$ with a single topic whose length is the sum of the lengths of the component topics. We also can 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. Let $\aleph$ represent a bit vector where a one in position i indicates that x_i, the ith topic from X, is chosen to be taught. Also, let C be a vector where C_i is the length of x_i. We can then create a function $f(\aleph)$, which

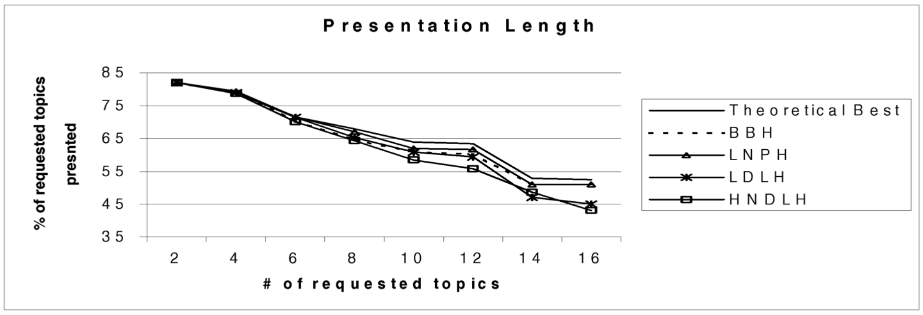

Fig. 11. Changing the request length (the number of requested topics).

evaluates the number of topics that are taught and have all their prerequisites included. In formal terms, this mathematical program is:

$$\text{Maximize } f(\aleph) \text{ subject to} : C.\aleph \leq t_{UB}, \aleph_i = 0, 1\ \forall i,$$
$$\text{where } f(\aleph) \text{ is defined as } \Sigma_i\ \aleph_i \cdot \Pi_{Xj \in Xi+\aleph_j}.$$

Unfortunately, problems of this type are hard to approximate. Results from Mathematical Programming Theory [8] 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 a then $\{a\}^+$ 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 [8]. This leads us to conclude that, for requests like Request 4, there are unlikely to be approximation algorithms that can guarantee their results are within any factor of the optimal, and so we should be content with using heuristics to solve real instances of the problems.

7 Conclusions and Future Research

This paper opens up a new problem domain (electronic textbooks) for database researchers; we are not aware of any database research that attaches a data model for electronic textbooks. We design an electronic textbook environment for the automated assembly of multimedia lessons, and study the use of a multimedia database, and database techniques for electronic textbooks containing precaptured multimedia presentations about topics in the book. We believe that this paper is a first step toward formulating and solving what may be a very large number of practical research problems about electronic textbooks.

One research direction to pursue is to design and evaluate a query language for users to construct their lesson requests. Another research direction is to introduce multiple experts, and handle advice conflicts between experts.

Acknowledgments

This research is supported by the US National Science Foundation grants IRI96-31214, CDA65-29513, and INT-9912229. A preliminary version of this paper was published in the IEEE Advances in Digital Libraries Conference, May 2000.

References

[1] S. Altingovde, O. Ozel, O. Ulusoy, G. Ozsoyoglu, and Z.M. Ozsoyoglu, "Topic-Centric Querying of Web-Based Information Resources," *Proc. Database and Expert Systems Applications Conf.*, Sept. 2001.

[2] Ariadne Project Web site, http://www.ariadne-eu.org/en/system/index.html, 2003.

[3] N.H. Balkir, "Managing Multimedia Presentations," PhD thesis, Electrical Engineering and Computer Science Dept., Case Western Reserve Univ., May 1999.

[4] D. Brickley and R.V. Guha, "Resource Description Framework Schema (RDFS)," W3C Proposed Recommendation, http://www.w3.org/TR/PR-rdf-schema, Mar. 1999.

[5] B.S. Bloom, *Taxonomy of Educational Objectives, Cognitive Domain.* New York: Longman Publishers, 1956.

[6] P. Barker, "Designing Interactive Learning Systems," *Education and Training Technology Int'l*, vol. 27, pp. 125-150, 1990.

[7] T. Boorsok and N. Higginbothan-Wheat, "Interactivity: What is it and What Can it do for Computer-Based Instruction," *Educ. Technol.*, pp. 11-17, 1991.

[8] M. Bellare and P. Rogaway, "The Complexity of Approximating a Nonlinear Program," *Math. Programming,* vol. 69, pp. 429-442, 1995.

[9] BookWorks, OverDrive Systems, Inc., http://www.overdrive.com, 2003.

[10] M.J. Carey and D. Kossmann, "Reducing the Braking Distance of an SQL Query Engine," *Proc. Very Large Database Conf.*, 1998.

[11] S. Chaudhuri and L. Gravano, "Evaluating Top-k Selection Queries," *Proc. Very Large Database Conf.*, 1999.

[12] W.W. Cohen, "Integration of Heterogeneous Databases Based on Textual Similarity," *Proc. SIGMOD*, 1998.

[13] DBLP Bibliography, http://www.acm.org/sigmod/dblp/db/index.html, maintained by Michael Ley, 2001.

[14] Electronic Classroom Project, http://erciyes.ces.cwru.edu/ecp.html, masters projects of J. Aithal, A. Jain, and G. Kalele, Electrical Engineering and Computer Science Dept., Case Western Reserve Univ., Dec. 1998.

[15] V. Hakkoymaz and G. Ozsoyoglu, "A Constraint-Driven Approach to Automate the Organization and Playout of Presentations in Multimedia Databases," *J. Multimedia Tools and Applications,* vol. 4, 1997.

[16] V. Hakkoymaz, J. Kraft, and G. Ozsoyoglu, "Constraint-Based Automation of Multimedia Presentation Assembly," *ACM Multimedia Systems J.*, Nov. 1999.

[17] J.H. Han and M. Kamber, *Data Mining Concepts and Techniques.* Morgan Kaufmann, 2001.

[18] R.B. Kozma, "Learning with Media," *Rev. of Educational Research,* vol. 61, no. 2, pp. 179-211, 1991.

[19] O. Lassila and R.R. Swick, "Resource Description Framework (RDF) Model and Syntax Specification," *W3C Recommendation,* http://www/w3.org/TR/REC-rdf-syntax, Feb. 1999.

[20] T. Lee, L. Sheng, B. Bozkaya, G. Ozsoyoglu, and Z.M. Ozsoyoglu, "Querying Multimedia Presentations Based on Content," *IEEE Trans. Knowledge and Data Eng.*, 1999.

[21] T. Lee, L. Sheng, A. Al-Hamdani, G. Ozsoyoglu, and Z.M. Ozsoyoglu, "Query Processing Techniques for Multimedia Presentations," *J. Multimedia Tools and Applications,* 1999.

[22] NetLibrary, http://www.netlibrary.com/, 2003.

[23] G. Ozsoyoglu et al., "Sideway Value Algebra for Object-Relational Databases," *Proc. Very Large Databases Conf.*, 2002.

[24] Open eBook Forum, an Int'l Trade and Standards Organization, http://www.openebook.org/aboutOEBF.htm, Year?

[25] Reusability, Collaboration, and Learning Troupe Web site, http://rclt.usu.edu/research.html, (contains the Instructional Architect research), 2003.

[26] R.A. Reiser, "Clark's Invitation to the Dance: An Instructional Designer's Response," *Educational Technology Research and Development,* vol. 42, no. 2, pp. 49-53, 1994.

[27] R.G. Ragsdale and A. Kassam, "The Magic of Multimedia in Education," *Multimedia Computing,* S. Reisman, ed., Idea Publishers, 1994.

[28] C. Seeberg et al., "iTeach—Interactive Teaching and Learning," *Proc. ACM Multimedia Conf.*, 1998.

[29] G. Salton, *Automatic Text Processing.* Addison-Wesley, 1989.

[30] W. Savitch, *Problem Solving with C++, the Object of Programming.* Addison-Wesley, 1996.

[31] R.C. Schak, "Learning via Multimedia Computers," *Comm. ACM,* vol. 35, no. 5, pp. 54-55, 1993.

[32] A. El Saddik, S. Fischer, and R. Steinmetz, "ITBeankit: An Educational Middleware Framework for Bridging Software Technology and Education," *Proc. EdMedia Conf.*, 2000.

[33] A. El Saddik, S. Fischer, and R. Steinmetz, "Reusability and Adaptability of Interactive Resources in Web-Based Educational Systems," *ACM J. Educational Resources in Computing,* Mar. 2001.

[34] M. Biezunski, M. Bryan, and S. Newcomb, eds., ISO/IEC 13250, Topic Maps, http://www.ornl.gov/sgml/sc34/document/0058.htm, 2003.

[35] The Walden's Path Project Web site, http://www.csdl.tamu.edu/walden, 2003.

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

Gultekin Ozsoyoglu received the PhD degree in computing science from the University of Alberta, Edmonton, Alberta, Canada, in 1980. He is a professor in the Department of Computer Engineering and Science, Case Western Reserve University, Cleveland, Ohio. He has published in database, security, multimedia computing, and real-time computing conferences and journals such as *ACM Transactions on Database Systems*, *IEEE Transactions on Software Engineering*, *IEEE Transactions on Knowledge and Data Engineering*, and the *Journal of Computer and System Sciences*. He has served on program committees and panels of database conferences such as ACM SIGMOD, VLDB, and IEEE Data Engineering. He was an ACM national lecturer, program chair of the Third Statistical and Scientific Database Conference, workshops general chair of the CIKM'94 and CIKM'96 Conferences, Research Prototypes chair of ACM SIGMOD'94 conference, a guest editor for *IEEE Transactions on Knowledge and Data Engineering*, general chair of the 11th Scientific and Statistical Database Management Conference in 1999, and cochair of the US National Science Foundation 2000 Information and Data Management Workshop. He is a senior member of the IEEE Computer Society.

Nevzat H. Balkir received the PhD degree in managing multimedia presentations in the Computer Engineering and Science Department, Case Western Reserve University (1998). He received the MS degree from Case Western Reserve University, and the BSc degree from Bilkent University, Ankara, Turkey in 1995 and 1993, respectively. He is a solutions architect at Experian North America. His primary research interests are in the areas of object-oriented databases, multimedia databases, query optimization, and data distribution on very large databases.

Z. Meral Ozsoyoglu received the BSc degree in electrical engineering, and the MSc degree in computer science, both from Middle East Technical University, Turkey. She received the PhD degree in computer science from the University of Alberta, Canada. She is a professor of computer science at Case Western Reserve University, Electrical Engineering and Computer Science Department. Her primary research work and interests are in the areas of principles of database systems, database query languages, database design, object-oriented databases and complex objects with applications in scientific, temporal, and multimedia databases. She has published several papers on these topics in computer science journals and conferences. She was ACM PODS program chair in 1997, and 11th SSDBM Conference program chair in 1999. She has served on organizing and program committees of several database conferences including ACM SIGMOD, ACM PODS, VLDB, IEEE DE. She was the ACM SIGMOD vice-chair from 1997-2001. She has been a recipient of several awards including an IBM Faculty Development Award, a US National Science Foundation Faculty Award for Women in science and engineering (FAW) award, and a University of Alberta Distinguished Alumni award. She is presently an associate editor of the *ACM Transactions on Database Systems*. She is a senior member of the IEEE Computer Society.

Graham Cormode received the bachelors degree in computer science from the University of Cambridge in 1998. He received the PhD degree, with a thesis titled "Sequence Distance Embeddings," from the University of Warwick in 2002. He is currently a postdoctoral fellow at the Center for Discrete Mathematics and Computer Science (DIMACS), based at Rutgers University, Piscataway, New Jersey. He is working on small space algorithms for massive data sets.