Topic Dependencies for Electronic Books

Graham Cormode *

July 10, 2002

Abstract

Electronics books consist of a number of topics, and information about dependencies between topics. We examine some theoretical problems related to handling these topic dependencies. In particular we consider the case where these topic dependencies are *acyclic*, and *nondecomposable* (that is, if topic a and b together require c, it is not necessarily the case that a or b alone require c). We formalise the semantics of this system, and give an axiomatisation which we show is sound and complete. We then show that we can easily compute the closure of a set of all topics in this model, which is the set of topics which are required by that set. This closure is computed in an identical manner to that for other formalisations of topic dependencies, showing that for this purpose no distinction need be drawn.

We then consider a different kind of closure, where given a set of desired topics, we must find a subset such that the total running time of all topics in its closure is less than a given time bound. We show that this problem is NP-complete. We analyse some proposed heuristics to approximate the solution of this problem and demonstrate that inputs can be engineered which elicit very poor performance from these heuristics, showing that no factor of approximation can be guaranteed for them. Finally, we transform the problem into a problem over a bipartite graph, which in turn can be formulated as a problem in mathematical integer programming. Hardness results for this problem give us confidence that a guaranteed approximation is unlikely to exist.

*Department of Electrical Engineering and Computer Science, CWRU *grc3@cwru.edu*

1 Introduction

The use of dependencies for electronic books is described in [OBO99]. An electronic book consists of a set of *topics* which are intended to teach subjects. Each topic may require some other topics to have been already taught before it can be taught. These prerequisite topics are encapsulated in a set of dependencies which also form part of the electronic book. These are specified in the form $X \rightarrow y$, meaning "the set of topics X has y as a prerequisite". The semantics of the system are that if a set of topics is given to a user, then the prerequisites for those topics must also be taught. The closure of a set of topics is all the topics which must be taught to satisfy the dependencies. There are a number of models for semantics of dependencies, based on two orthogonal variations.

- The set of dependencies may or may not be allowed to be *cyclic*. Informally, cyclicity allows a topic to be a prerequisite for itself.
- Dependencies may or may not be *decomposable*. Decomposability means that $X \rightarrow y$ is equivalent to $x \rightarrow y$ for each $x \in X$. Nondecomposability means that this decomposition is not allowed.

Of the four possible combinations, three are covered in [OBO99]. Cyclic, nondecomposable dependencies have the same semantics as functional dependencies of database theory. A description of Armstrong's axiomatisation of functional dependencies, and computation of their closure is given in [Ull89]. A set of either kind of decomposable dependencies can be expressed as a graph, and their closure corresponds to transitive closure of that graph.

In this report, we first study the fourth possibility, of nondecomposable acyclic dependencies. We first clarify what it means for a set of dependencies to be acyclic. We show that these dependencies can be axiomatised using a single axiom, and prove that this axiomatisation is both sound and complete. By taking a modified view of what should be included in the closure, we give a simple polynomial time algorithm to compute the closure of a set of topics, and show that this algorithm gives the correct results.

We then consider problems of automated lesson construction. A *lesson* is a closed set of topics (the set is equal to its own closure) which satisfies certain constraints. The input will be a set of desired topics, and the goal shall be to produce a lesson which contains as many of these as possible. Although the closure of any set of topics is computable in polynomial time, when we add the constraint of a time limit on the total length of output topics, finding an optimal solution can become hard. We study this very natural request, to include as many of a set of requested topics be taught given an upper bound on the lesson length. We show that this problem is NP-Complete. We further show that simple heuristics suggested to answer this request have no guaranteed quality, and further, that an algorithm with a guaranteed approximation factor is unlikely to exist.

Throughout this report, we shall use the following conventions: X, Y will denote sets of topics, and XY will be shorthand for $X \cup Y$. x, y will refer to single topics. F will denote a set of dependencies.

2 Acyclic Nondecomposable Dependencies

2.1 Definition of Acyclic Nondecomposable Dependencies (ANDs)

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

Definition 1 *A set of dependencies is* strongly cyclic *if, applying the rule of transitivity, it is possible to deduce that an element depends on itself. For example, the set* $F = \{X \to Y, Y \to Z, Z \to X\}$ *is strongly cyclic. This still holds if* X *represents a subset of elements.*

Definition 2 *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 an element depends on itself. For example, the set* $F = \{WX \to Y, YZ \to V, V \to W\}$ *is weakly cyclic. It is sufficient to consider only single elements.*

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

2.2 Axiomatisation of ANDs

We observe that Armstrong's Axioms, used to axiomatise standard functional dependencies are not appropriate when acyclicity is demanded. The axiom of reflexivity, $X \to Y, Y \subseteq X$ generates trivial (weak) cycles, as does the axiom of augmentation, $WX \to WY$ if $X \to Y$. This leaves transitivity, which is sound, but is not sufficiently powerful.

However, a single axiom is sufficient for dealing with ANDs

Definition 3 *The axiom of* pseudo-transitivity *is as follows: If* $X \to Y$ *and* $WY \to Z$ *then* $WX \to Z$. W *is permitted to be empty, in which case the axiom is simply transitivity.*

In addition to this axiom, we may make use of the split/join rule.

Definition 4 *The split/join rule is that if* $X \to AB$ *then* $X \to A$ *and* $X \to B$*, and vice-versa. This rule follows from the definition of dependencies.*

Note that if a set of ANDs is presented so that the right hand side of every dependency is a single element, then application of the pseudo-transitivity axiom will preserve this condition, and so the split/join rule may never be invoked. However, we may sometimes write $X \to Y$ as shorthand for $X \to y$ for all $y \in Y$.

For ANDs, we need to modify our notion of the closure of a set of attributes.

Definition 5 X^+*, the closure of a set of attributes,* X*, with respect to a set of ANDs,* F*, is the set of attributes* Y*, consisting of all* $y \notin X$ *such that* $X' \to y$ *can be deduced from* F *using pseudo-transitivity, and* $X' \subseteq X$.

Lemma 1 *If $y \in X^+$, then $X' \rightarrow y$ follows from the axiom of pseudo-transitivity where X' is some subset of X. Conversely, if $X \rightarrow y$ follows from the axiom of pseudo-transitivity, then $y \in X^+$*

Proof: The first statement is straightforward from the definition of X^+. The second also follows from the definition *a fortiori* since $X \subseteq X$. □

Definition 6 *$X \Rightarrow Y$ denotes that for each $y \in Y$, there exists an $X' \subseteq X$ such that $X' \rightarrow y$.*

We now give a lemma stating some properties of this notation.

Lemma 2

i) If $Y \subseteq X^+$ then $X \Rightarrow Y$, that is for each $y \in Y$ we can find an $X' \subseteq X$ such that $X' \rightarrow y$.

ii) If $X \Rightarrow Y$, $X''Y \rightarrow w$ and $X'' \subseteq X$, then $X \Rightarrow w$.

iii) If $Xa \Rightarrow y$ and $X' \rightarrow a$ where $X' \subseteq X$ then $X \Rightarrow y$.

Proof: i) This is straightforward from the definition of $\Rightarrow$ and X^+.

ii) This is proved by iteratively applying pseudo-transitivity. Let $n = |Y|$, and define X_i from $X_i \rightarrow y_i$ for $i = 1 \ldots n$, which we know to be the case as $X \Rightarrow Y$. We start with $X''y_1y_2 \ldots y_n \rightarrow w$ and use pseudo-transitivity with this and $X_1 \rightarrow y_1$ to deduce that $X''X_1y_2 \ldots y_n \rightarrow w$. We repeat this procedure with each $X_i \rightarrow y_i$ in turn to eliminate all members of Y from the expression, leaving us with $X''X_1X_2 \ldots X_n \rightarrow w$. Since $(X'' \cup X_1 \cup X_2 \cup \ldots \cup X_n) \subseteq X$, then $X \Rightarrow w$.

iii) We perform a case split, and first consider the case that $Xa \Rightarrow y$ is shown by $X'' \rightarrow y$, where $X'' \subseteq X$. In this case trivially $X \Rightarrow y$. The other case is when $Xa \Rightarrow y$ is shown by $X''a \rightarrow y$. Here, we use pseudo-transitivity to show that $X''X' \rightarrow y$ and hence $X \Rightarrow y$. □

The axiom of pseudo-transitivity seems semantically valid. We will now show that it is sound and complete, using the above lemma.

Theorem 3 *The axiom of pseudo-transitivity is sound and complete for ANDs.*

Proof: Soundness follows from the proof of soundness of transitivity and augmentation for ordinary FDs, which allows us to conclude that the compound rule of pseudo-transitivity is sound. Completeness is shown by adapting the proof of completeness from [Ull89], which we do here in full. For simplicity, we actually prove that the shorthand $\Rightarrow$ is complete.

Let F be a set of dependencies over attribute set U, and suppose $X \Rightarrow y$ cannot be inferred from pseudo-transitivity. Consider two lessons which are constructed so that all topics in $X \cup X^+$ are included in both, but one also includes all topics in $U - X^+ - X$ while the other includes no others. We show that all dependencies in F are satisfied by these two lessons. Suppose $V \rightarrow w$ is in F but not satisfied by these lessons. Then $V \subseteq X \cup X^+$, or else the two lessons could not violate $V \rightarrow w$. Also, w cannot be a member of $X \cup X^+$, or $V \rightarrow w$ would be satisfied by the lessons. Since $V \subseteq X^+ \cup X$, we can split V into $V_X = V \cap X$ and $V_{X^+} = V \cap X^+$, so $V_XV_{X^+} \rightarrow w$ and $X \Rightarrow V_{X^+}$. By Lemma 2 ii), we combine these to show $X \Rightarrow w$, that is there is a subset of X, X', for which $X' \rightarrow w$. Since this follows from the axiom, it means that by definition, w is in

X^+, which we assumed not to be the case. We conclude by contradiction that each $V \rightarrow w$ in F is satisfied by our two lessons.

Now we must show that $X \Rightarrow y$ is not satisfied by these lessons. Suppose it is satisfied. It follows that $y \in X^+$, else the two lessons agree on X but disagree on y. But then Lemma 1 tells us that $X \rightarrow y$ can be inferred from the axiom, a contradiction. Therefore, $X \Rightarrow y$ is not satisfied by the lessons, even though every dependency in F is. We conclude that whenever $X \Rightarrow y$ does not follow from F by the axiom, F does not logically imply $X \Rightarrow y$. That is, the axiomatisation is complete for $\Rightarrow$. □

3 Computing the closure

The following algorithm computes the closure of a set of attributes X:

Algorithm 1

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 $y \notin X$.

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 4 *Algorithm 1 correctly computes X^+.*

Proof: We show that if X^+ contains y, then y is output, and then that if y is output, then X^+ contains y.

We first show that if $X \Rightarrow y$ then y is included in the output by induction on the number of lines of the proof. Our inductive assumption is that if the proof $X \Rightarrow y$ has n lines then y is included in the output of our algorithm. The base case is when there is only one line. This occurs when we prove $X \Rightarrow y$ by observing that $X' \rightarrow y \in F$, for a subset X' of X. In this case, if we are applying the algorithm, then it will add y to X^+, and we are done. In the inductive case, we observe that $X \Rightarrow y$ is deduced from $X' \rightarrow a \in F$ and $Xa \Rightarrow y$, using Lemma 2 iii). By our inductive hypothesis, y will be included in the output, since our algorithm will output a, and then behaves as if it were calculating $(Xa)^+$.

We now show that if y is included in the output, then there is a proof that $X \Rightarrow y$. We do this by showing by induction that if $X \Rightarrow X^{(i)}$ then $X \Rightarrow X^{(i+1)}$. Clearly $X \Rightarrow X^{(0)}$ is vacuously true, since $X^{(0)} = \{\}$. The step to find $X^{(i+1)}$ from $X^{(i)}$ is to find a member of F such that $XX^{(i)} \Rightarrow y$, ie $X_i \rightarrow y$, for $X_i \subseteq (X \cup X^{(i)})$. We apply Lemma 2 ii) to show that we can prove $X \Rightarrow y$, hence we are correct to output y. From the definition of $\Rightarrow$, it is clear that $X \Rightarrow X^{(i+1)} = X^{(i)}y$, since $X \Rightarrow X^{(i)}$ and $X \Rightarrow y$. □

Corollary 5 *This algorithm is applicable to any set of dependencies, irrespective of whether they are cyclic or decomposable. This follows by observing that the algorithm proceeds in the same manner as the algorithm for cyclic nondecomposable dependencies (functional dependencies), despite differing by including X in X^+. The set of topics to be taught given by $X \cup X^+$ computed by*

each algorithm is identical given a set of (cyclic or acyclic) dependencies. We further note that a set of decomposable dependencies presented in canonical form (a single topic on each side of the dependency) is just a special case of a set of nondecomposable dependencies. Hence any algorithm that works for nondecomposable dependencies also applies to decomposable dependencies.

This algorithm can be implemented naively to check through the set of dependencies each iteration to see whether any new attributes can be added. A more efficient implementation is described in [Ull89], which runs in time which is 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 6 *Computation of the closure of a set of attributes X under a set F of acyclic nondecomposable dependencies does not violate acyclicity. That is, $X \Rightarrow X^+$ will not imply any cycles.*

Proof: Since we have shown that X^+ contains y if we can deduce $X' \rightarrow y$, then we just need to show that the axiom of pseudo-transitivity cannot introduce cycles. Recall from Definition 4 that a cycle exists if, by treating dependencies as decomposable, we can deduce $X \rightarrow X$. Suppose that we have a set of dependencies, F, which is acyclic, and we apply pseudo-transitivity to get F'. Then, given dependencies $X \rightarrow Y, WY \rightarrow Z$, we add the dependency $WX \rightarrow Z$. Treating the dependencies as decomposable (and using $\mapsto$ for clarity), we begin with $X \mapsto Y$, $W \mapsto Z$, $Y \mapsto Z$, and we add $X \mapsto Z$. If this creates a cycle, then it is because we can somehow deduce $Z \mapsto X$. But if this can be deduced, then the initial set of dependencies was already cyclic, since we have $X \mapsto Y, Y \mapsto Z$ (that is, we could already deduce $X \mapsto Z$), contradicting our initial assumption. Hence acyclic dependencies remain acyclic under pseudo-transitivity. □

4 Hardness of Request 4

An open problem of [OBO99] was whether a certain style of request was hard, referred to therein as "Request 4". In a sense it asks for a different kind of closure – the closure of a subset such that a time bound is met by the result. We now reproduce the specification of Request 4.

Lesson Request 4 Given (a) the user's knowledge 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 time length, produce a lesson of duration t_{UB} or less that teaches as many of the topics in X as possible.

Theorem 7 *Request 4 is NP-Complete.*

Proof: We split the proof of the hardness of Request 4 into two cases: when the dependencies are decomposable and when they are nondecomposable. As decomposable dependencies are a special case of nondecomposable dependencies, the first proof is unnecessary, but instructive.

Lemma 8 *Request 4 is NP-Hard for nondecomposable dependencies*

Proof: We show a reduction from min 3-SAT, which is shown NP-Complete in [CK99]

Input: An instance of min 3-SAT, over a set of literals U.

Goal: To give an assignment to the literals of U which minimises the number of satisfied clauses.

We shall negate each clause of the input, so the goal will now be to maximise the number of satisfied clauses. We form our instance of Request 4 by creating one topic for each clause C_i, and two for each literal x in U, one to represent x and the other to represent $\bar{x}$. Each topic $x, \bar{x}$ $(x \in U)$ is assigned unit cost, and each topic C_i is assigned zero cost. We also add a topic not in U, \$, to which we assign cost $|U|$, and for each literal $x \in U$ we add the dependency $x\bar{x} \rightarrow \$$. For each input clause of the form $C_i = (a \vee b \vee c)$, where a, b, c are literals or negated literals, we add three dependencies: $C_i \rightarrow \bar{a}, C_i \rightarrow \bar{b}, C_i \rightarrow \bar{c}$. We set the upper time bound t_{UB} to be $|U|$. The desired set of topics, X is all the C_i, and user's initial knowledge is zero.

We now claim that any solution to this instance of Request 4 is a solution to the corresponding instance of min 3-SAT. We observe that clause C_i is falsified if and only if its negation is satisfied. We also note that we for each literal x in U in the instance of Request 4 we can choose at most one of x,$\bar{x}$ from the available topics, corresponding to whether x is set to true or false in the generated truth assignment. Note also that taking topics from U cannot decrease the number of topics available from X, so without loss of generality, we assume that precisely $|U|$ topics are taken, corresponding to an assignment of truth values to U. We can take a topic C_i if and only if we have taken all its prerequisites, which corresponds to the clause C_i being negated. Therefore, this solution to Request 4 solves the min 3-SAT problem, hence Request 4 with nondecomposable dependencies is NP-hard. □

Lemma 9 *Request four is NP-Hard for decomposable dependencies*

Proof: We perform a reduction from the problem of k-clique (the NP Completeness of which is discussed in [GJ79]).

Input: a graph $G(V, E)$ and integer k, $1 \leq k \leq |V|$.

Goal: Find a clique (complete subgraph) with k vertices.

We form an instance of Request 4 from this problem as follows. We create one topic for each vertex, and one for each edge. We set $X = E$, and assign costs of one to each member of V, and zero to each member of E. For each edge in E, $e_i = (v_j, v_k)$, we create two dependencies $e_i \rightarrow v_j$ and $e_i \rightarrow v_k$. We set the time upper bound to $t_{UB} = k$.

We claim that if the number of topics from X selected is $k(k-1)/2$, then the algorithm has found a k-clique. Clearly, since the topics in X represent edges which depend on their two vertices, then if we can select $k(k-1)/2$ edges which connect k vertices, then this defines a clique. Conversely, if we pick k vertices from a graph and find that there are $k(k-1)/2$ edges which have both ends in this subset of vertices, then necessarily we have found a clique. Thus, an algorithm to answer Request 4 in this model can be used to solve k-clique; hence the problem is NP-hard. □

Corollary 10 *The above reduction suggests that Request 4 is also hard in the case where all the topic times are equal (unit cost).* [1] *We show this by modifying the reduction so that each topic in*

[1] It is not sufficient to replace each topic which integer cost c with c topics of unit cost to show this: this transformation can cause an exponential blow-up in the size of the problem

X also has unit cost. We now set t_{UB} to be $k(k+1)/2$, and claim again that an algorithm to solve request 4 will return $k(k-1)/2$ edges from X if and only if there is a k-clique in the generating graph, G. This follows by considering that $k(k-1)/2$ edges in X can only be achieved by selecting k vertices from V. To increase the topics picked from X, we would have to reduce the number of vertices chosen from V, which reduces the number of topics we can choose from X.

These two lemmas allow us to prove the theorem. We finally observe that given a set of dependencies, a requested set X, a time upper bound t_{UB} and a set of proposed topics Y, we can determine in polynomial time that i) the time to teach Y is less than t_{UB} and ii) $Y = Y \cup Y^+$, that is, all the prerequisites necessary are included. This observation allows us to conclude that Request 4 is in NP and hence, as the two above lemmas show that it is NP-Hard, we conclude that it is NP Complete. □

5 Worst case performance for Request 4 Heuristics

Various heuristics are proposed in [OBO99] to give efficient algorithms to answer Request 4. These have been 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.

5.1 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 $0 \leq i \leq k$. We choose F to consist of $x'' \rightarrow x'$ $x' \rightarrow y_i$, for all $1 \leq i \leq k$. 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 which has many prerequisites.

5.2 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 $0 \leq i \leq k$, and create F with the following dependencies: $x_0 \rightarrow y_i$ for $1 \leq i \leq k-1$; and $\forall i, 1 \leq j \leq k$ $x_i \rightarrow z_j$. 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.

6 Reduction of decomposable dependencies to a bipartite graph problem

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 initialise F, the new set of dependencies, to be $x \to x'$. We then add dependencies to F such that $x \to y'$ for each $y \in (x)^+$. This problem is identical to the original hierarchical problem instance.

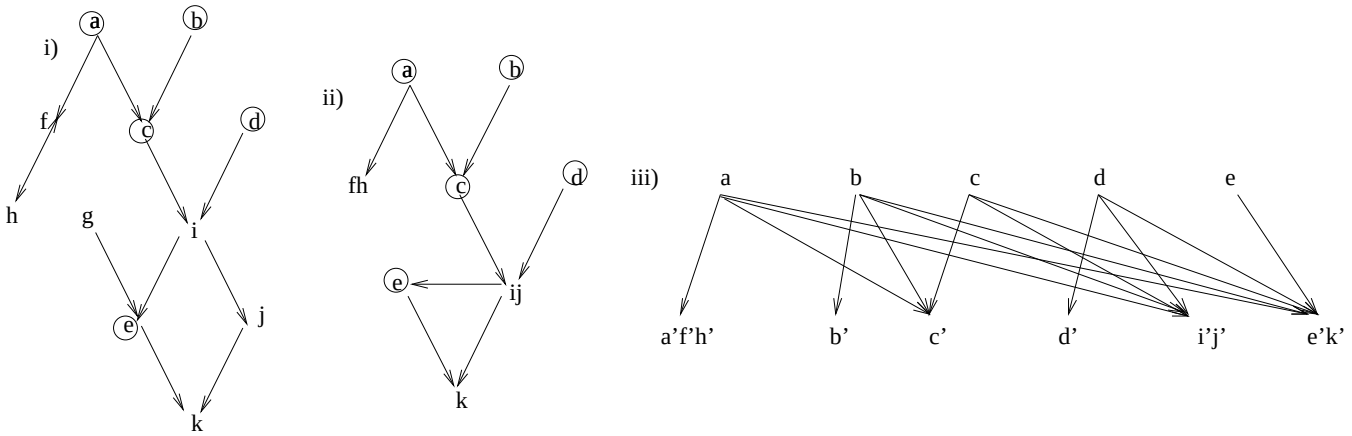

Figure 1: This illustrates an instance of Request 4, with $F = \{a \to f, b \to c, c \to i, d \to i, e \to k, f \to h, g \to e, h \to f, i \to e, i \to j, j \to k\}$ and $X = \{a, b, c, d, e\}$. i) shows these dependencies as a directed graph. ii) simplifies the graph by making the cycle f, h into a compound topic; dropping the topic g which is not needed by any topic in X; merging i and j since any topic in X which requires j also requires i. iii) shows how this is moved to a bipartite model: only topics in X appear on one side of the graph. On the other side are sets of topics, created so that each topic in X requires the inclusion of exactly those that it is linked to.

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. An example of this form is show in Figure 1. 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:

$\max(f(X))$

subject to: $C.X \leq t_{UB}$

$X_i = 0, 1 \quad \forall i$

where $f(X)$ is defined as $\sum_{x \in X} \prod_{y \in x^+} y$.

Unfortunately, problems of this type are hard to approximate. Results from Mathematical Programming Theory ([BR95]) 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 ([BR95]). This leads us to conclude that for requests like Request 4, there are unlikely to be approximation algorithms which can guarantee 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. [2]

An alternative approach is to look for pseudo-polynomial solutions to the problem, that is, algorithms whose running time is polynomial in the size of the input and t_{UB}. Because t_{UB} can be expressed in binary using $\lceil \log_2 t_{UB} \rceil$ bits, such an algorithm would actually be exponential in the problem input size, but as we might expect t_{UB} to be relatively low for many typical problem instances, this would still be a feasible solution. Regrettably, this does not seem possible. Part of the reason for this is that there can be many possible solutions to sub-problems of the main problem which cannot be used interchangeably to build a solution to the whole. The possibility of having distinct topics which have some common prerequisites is the aspect which makes this problem truly hard. If there were no such overlaps, then the problem would be trivially solvable in polynomial time by calculating the closure of each requested topic, and repeatedly picking the cheapest until the time bound is reached.

[2] We should not totally give up hope of their being an approximation algorithm, as the instance of mathematical programming here is more constrained than the general case, in that $f(X)$ contains only products of distinct x_i's, always with a constant factor of one.

7 Conclusion

We have seen that, from the point of view of computing closures, there is no distinction between cyclic and acyclic, and decomposable and nondecomposable topic dependencies, and so this distinction can be dropped. However, acyclic nondecomposable dependencies have some interesting properties and it is useful to know that they can be axiomatised and manipulated in a way that is quite distinct from the traditional functional dependency style of cyclic nondecomposable dependencies.

We have also seen that a very natural style of request, which essentially asks for the best closure given a time bound on the lesson length, is hard to solve exactly, and hard to give even an approximate answer to. It is perhaps the most basic kind of request which involves a time bound, and its hardness is bad news for the implementer of an electronic book system. We have seen that using heuristics can lead to virtually none of the required topics being taught when optimally almost all of them could be made into a lesson. From a pragmatic point of view, perhaps these reservations can be overcome. Since an electronic book system is necessarily an interactive one (because a user must interact with it to use it), we will expect a user to return to the system later if they do not have time to learn all their desired topics in one sitting (in other words, there will be other lessons). If we assume that a user will keep returning to the system until they have learned all their desired topics, then eventually they will have viewed all the topics in the closure of their wish-list exactly once, so overall the user's time will not have been wasted.

Acknowledgments

I am indebted to Mike Paterson for a discussion regarding the hardness of Request 4 under decomposable dependencies.

References

[BR95] M. Bellare and P. Rogaway. The complexity of approximating a nonlinear program. *Math. Programming*, 69:429–442, 1995.

[CK99] Pierluigi Crescenzi and Viggo Kann. A compendium of NP optimization problems. *http://www.nada.kth.se/ viggo/problemlist/compendium.html*, 1999.

[GJ79] Michael R Garey and David S Johnson. *Computers and Intractibility, a Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., 1979.

[OBO99] Gultekin Ozsoyoglu, N Hurkan Bulkir, and Z Meral Ozsoyoglu. Electronic book multimedia databases. *http://erciyes.ces.cwru.edu/tekin/papers/EB.ps*, 1999.

[Ull89] J. D. Ullman. *Database and Knowledge-Base Systems*. Computer Science Press, 1989.