Synthesizing Dynamic Curriculums from Concept Maps Through Matroids and Concept Vectors

Mark L. Dyson, Sheila B. Banks, Eugene Santos, Jr., and F. Alex Kilpatrick
Artificial Intelligence Laboratory
Air Force Institute of Technology
Wright-Patterson Air Force Base, OH 45433
sbanks@afit.af.mil, eugene@engr.uconn.edu

Abstract

The bulk of research to date relating to intelligent tutoring systems (ITS) is focused on the student, and on methods for representing the knowledge itself. From student models to learning schemas to presentation methods, comparatively little attention has been paid to the problem of educators attempting to build viable curriculum plans for use in an ITS environment---yet when this problem is addressed in the literature, it is recognized as a potentially daunting one. This paper presents initial work to address the problem of ITS curriculum development by proposing a practical, computable approach for knowledge engineering that is based on proven classroom methods. We describe our system for dynamically creating student curriculum plans from a knowledge base created using the described methodology, which utilizes already-established algorithms of proven tractability, and then discuss how this system can be integrated into existing and future ITS design.

Introduction

Artificial intelligence has been applied to education for years; in fact, trend analysts from as far back as 1983 declared such application to be "inevitable" (O'Shea & Self, 1983). Nonetheless, despite ever-growing focus on the computer as a dominant medium in the field of educational technology (Ely, Januszewski, & LeBlanc, 1988), expert opinion concerning the utility of artificially intelligent teaching tools ranges from statements that instructional programs "don't know what they're doing" (O'Shea & Self, 1983) to the general conclusion that such programs are of poor quality (Ely, Januszewski, & LeBlanc, 1988).

Before an intelligent tutoring system (ITS) can be employed, it must have a knowledge base from which to teach. That knowledge must be represented in a tractable form to be useful--both from a computing standpoint and from the point of view of presenting that knowledge to a student.

In the literature, one finds numerous examples of knowledge representation schemes, from the idea of concept mapping (Novak & Gowin, 1984) to the intensive, hierarchical databases used in the Air Force's Instructional System Development (ISD) project (HQ AETC/XORR, 1984). Unfortunately, even in the case of automated tools such as provided in the ISD project, educators must face steep learning curves, a lack of a standardized interface, and a significant amount of manual development when constructing lesson plans (HQ AETC/XORR, 1984).

To date, the bulk of ITS research tends to focus on student modeling and knowledge presentation; in short, on the student's perspective within the learning environment. As pointed out in the ISD project, the educator, while attempting to develop instructional curricula for an ITS environment, is often left with a significant amount of development work (HQ AETC/XORR, 1984). What is needed, before attempting to design an ITS, is a methodology for defining and developing student curricula in a form directly suitable to ITS implementation. This methodology needs to be quantifiable both in terms of content and applicability, and able to accept feedback metrics on a given student's progress in order to modify the lessons and the curriculum plan as needed. In addition, any tool providing this capability should be useful without requiring excessive training.

Our research introduces a respected and long-standing method of representing complex knowledge domains in the classroom, *concept mapping*, and illustrates a system for mapping that representation into computer-readable form. We developed a prototype system, based on that form, which is able to accept relatively simple inputs from an educator via a simple world wide web (WWW) interface to Java-enabled WWW browsing software and return a dynamic lesson plan.

This paper will begin by providing background on concept mapping, with an illustrative example of a concept map, and then show how this map is used by a knowledge engineer to define teaching materials within a sample domain. We will introduce concept vectors, motivate their usefulness in representing concepts in computer-readable form, and provide theoretical grounding for an effective algorithm for creating a dynamic curriculum. We will also discuss the prototype system and conclude with a discussion of how the prototype could be enhanced via future research, including automated interfaces between the curriculum builder and an actual ITS.

Concept Mapping

To begin any knowledge-based system, one first needs a workable system for representing that knowledge (Winston, 1993). One very good representation for even potentially abstract knowledge domains was developed in the mid-eighties by Joseph Novak and Bob Gowin (Novak & Gowin, 1984). Not only does this representation, called concept mapping, enable a domain expert to map knowledge in a hierarchical tree structure amenable to computer processing, it has the additional virtue of being a respected and powerful tool in the classroom environment itself (Regis, Giorgio-Albertazzi, & Roletto, 1996).

Concept maps represent an approximation of the relevant concepts and propositions of a given knowledge domain (Novak & Gowin, 1984), and their creation requires both domain expertise and experience with the concept mapping process. Figure 1 provides an illustrative concept map created by a domain expert; in this case, an expert on the topic of Meat Science (Novak & Gowin, 1984). The first step is to identify the major topic to be mapped; in the case of the example, the domain expert has decided that the overarching consideration for this domain is meat quality. This concept of quality leads to the concepts of metrics by which to determine this quality, hence the concepts of judging, and then criteria. From this beginning the relevant criteria follow, and below them are identified the various conditions under which those criteria are affected, such as an animal's age, environment, and feeding habits. In the final form, the domain has been subdivided into a map of 29 nodes, each node representing an atomic concept in the knowledge domain as determined by the domain expert.

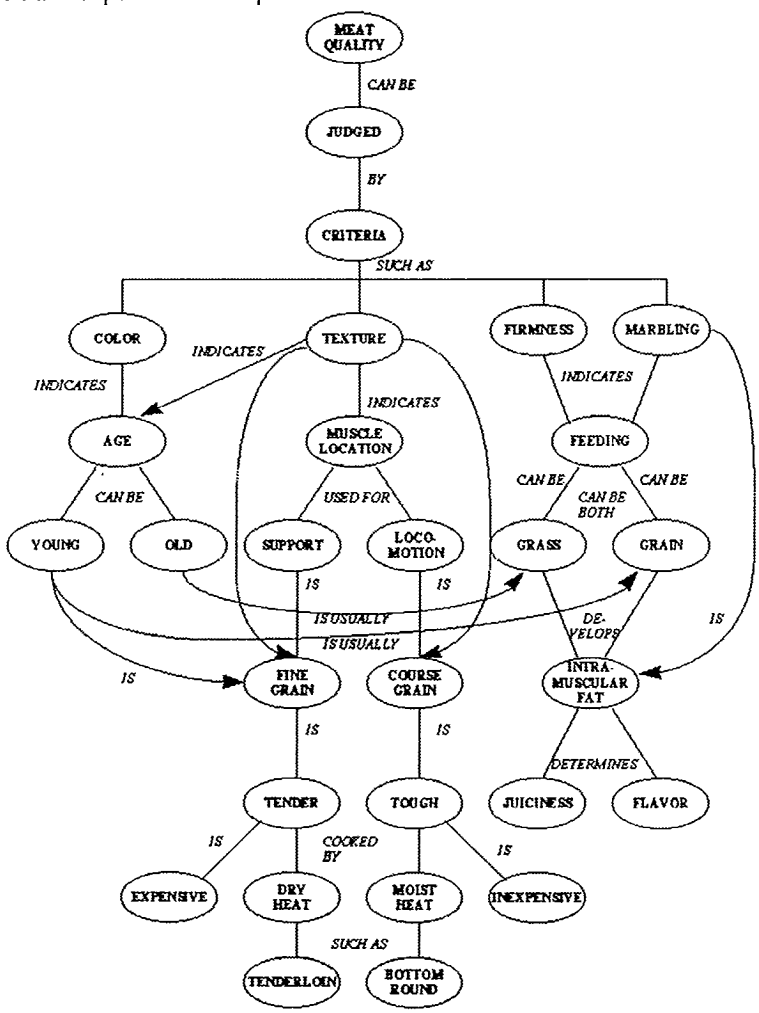

Figure 1. Complete Concept Map of Meat Science Course (Novak & Gowin, 1984)

Mapping Knowledge to Concept Vectors

The first step of our methodology for defining and developing student curricula in a form directly suitable to ITS implementation was to quantify the expression of knowledge within the concept map. This quantification process took the form of mapping the knowledge from the concept map into a vector space, hence the name concept vectors. This section first covers the relationship of concept maps to vector space and then details our concept vector representation.

Relating Concept Maps to Vector Space

The methodical breaking down of a potentially complex domain into progressively smaller conceptual components used in concept mapping is key to the idea of representing knowledge as vectors. A vector is made of components, each representing a magnitude along a single dimension in *n*-dimensional space (Auer, 1991). In order for the vector to be meaningful, each component must exist entirely within its dimension, having no contribution to the magnitudes of other components in that vector. Carrying this analogue to the concept mapping domain, if a vector represents the entire knowledge domain then each concept within the overall concept map can be thought of as a component of that vector. Further, the knowledge engineer must take pains to ensure the concepts are broken down as much as possible to prevent overlap between them---just as vector components provide contribution only in their dimension.

Once a given concept is identified to the desired level of simplicity, the domain expert must draw on available experience to determine what constitutes mastery of that concept. In the meat science case, for example, there exists the concept of "grass" as relates to feeding of meat animals (Novak & Gowin, 1984). The domain expert might conclude that there are a finite number of types of grass an herbivore can encounter, and that mastery of "grass" means that a student can correctly identify each type and relate its effect on an animal's health in relation to the other types.

Given this conclusion, a metric has been established wherein knowledge of "grass" can be measured on a scale from total ignorance to complete mastery. Once such a metric has been established for each concept in the domain, it is normalized so that complete ignorance is represented by zero, and total mastery by one. Applying such a numerical measure to each of the concepts makes it possible to represent the concepts in a machine-readable form, and to do computations based on levels of mastery of the knowledge domain. Further, the ability to represent any given level of knowledge within the domain by a string of numbers--a vector--which follows known properties becomes a powerful tool for knowledge representation.

Concept Vectors

Once a concept map has been established for a given domain, the available teaching material must be reviewed and defined within that context. The key to this process is again to visualize the *n* nodes of the concept map as dimensions in a vector space. The teaching material available---be it a pamphlet, a section of a chapter from a textbook, or a previously-coded instructional module from an existing ITS---must be matched against which node(s) of the concept map to which it relates. For example, in the meat science case, there might exist in the curriculum a textbook on animal nutrition. Within that text might be found a chapter on herbivores, with one section discussing grass and another detailing various grain feeds. In this case the chapter would be treated as two distinct units of instructional material, each relating to a different concept.

The second step is to determine how much of the appropriate concept a given unit covers. Recall that for each concept in the domain, the span from complete ignorance to full mastery is represented as a normalized scale from 0.0 to 1.0. As a unit is evaluated against a given concept, the domain expert determines what degree the concept is covered and assigns appropriate start and end values. It is important to reiterate here that teaching material is assumed to follow a progression from ignorance to mastery. The act of defining a unit of teaching material in this context assumes contiguous coverage from lower to upper bound within the span--the smallest unit of teaching material in this context. If a unit does, in fact, have gaps in coverage as defined by the domain expert, that unit should be further divided into smaller units until no gaps exist in a single unit. This additional division of the concepts into smaller units further illustrates the importance of making the initial concept map as granular as possible.

As an illustrative example, consider Figure 2, a subset drawn from Figure 1. Of the seven concepts depicted, the four highlighted represent the concepts of interest.

Figure 3 shows three lesson modules represented as three concept vectors that address the four concepts of Figure 2. Said another way, Figure 3 presents three concept vectors, each containing four ordered pairs of numbers expressing the amount of each concept covered by the lesson module. The first ordered pair of each concept vector represents the amount of coverage for the first concept of Figure 2, the second ordered pair represents the amount of coverage for the second concept of Figure 2, etc. In this case, the three vectors each only address concept one; the first represents an entry-level module covering the concept from no prior knowledge to a point defined to be three-tenths of the entire range. The second module covers the range from two-tenths to eight-tenths, representing an intermediate level of information, while the third is the most

advanced of the three, beginning seven-tenths of the way along the scale and covering the material up to total master of the concept. Note that, while no one of the three vectors shown covers the whole concept, the three taken together do provide complete coverage.

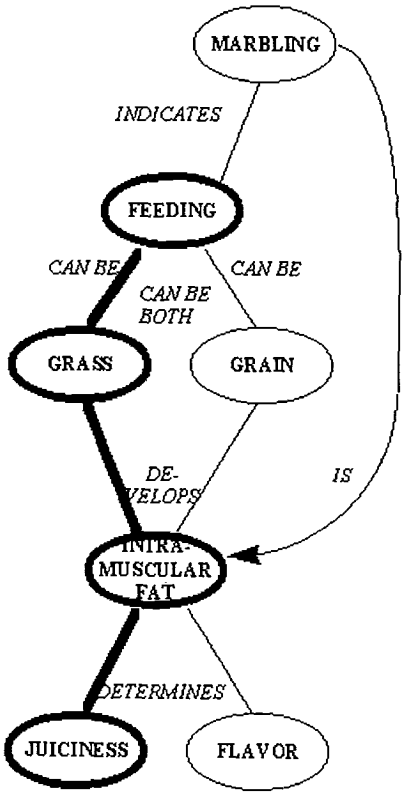

Figure 2. Subset Concept Map of Meat Science Course (Novak & Gowin, 1984)

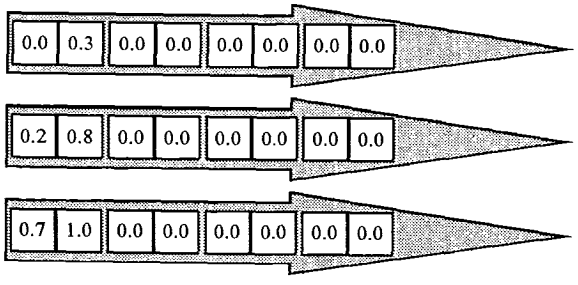

Figure 3. Three Defined Concept Vectors

There is some overlap in the depicted vectors; not all lesson materials available to an educator can be assumed to fit together without some overlap in how they cover a concept. Further, given effective granularity in the concept map definition, one can reasonably expect to see the typical vector providing a contribution in only one concept; in the case of the provided example the three units of material defined as vectors might represent three subsections of a textbook chapter on feeding techniques, with overlap reflecting review and cross-referencing by the author.

With the knowledge domain defined as an *n*-dimensional vector space, and with the available teaching materials defined within that space, the power of this representation becomes clear: the units of teaching material can be summed as vectors either to evaluate the completeness of available material within the context of the overall domain, or in order to reach a desired target vector. In the latter case, the units selected to sum to the target vector map directly to the teaching materials the educator needs to comprise a lesson plan designed to teach given concepts to a desired level.

At this point it is useful to re-visit the forms that lesson modules could take. Previous discussions have mentioned such items as pamphlets and sections from textbooks, but in practice such sources might be difficult to quantify as precisely as described here; they are mentioned mainly for illustrative purposes. It is expected that the design methodology presented will most useful when interfacing with an established ITS, presumably with a database of lesson modules already defined and coded. In such a case the knowledge domain is already defined by the material the ITS is designed to teach, and lesson modules will already be defined in terms which the ITS can use to differentiate between them for selection. Given this situation the knowledge engineering effort becomes less daunting, and the goal of defining discreet concepts that map back to specific subsets of lesson modules intuitively becomes an easier one to attain.

Using Concept Vectors: The Dynamic Curriculum Builder

After completing concept vector definition, we then require a technique for utilizing the concept vectors to produce an ITS student curriculum. The choice and implementation of this technique is the next step in our methodology for dynamically developing ITS student curricula. This section presents the considerations necessary in the choice of a technique, our use of the matriod structure to benefit this algorithm choice, and the implementation of both the representation and manipulation algorithm for the dynamic curriculum builder.

Dynamic Curriculum Generation

The first step was to identify the general form of the problem. As with many schemes involving search, search space pruning and early arrival at optimal solutions is both desirable and often difficult (Winston, 1993). Within the context of this research,

"optimality" is defined as the curriculum generator consistently selecting a lesson module providing the largest overall amount of coverage (which still lies completely within the target range) whenever such a choice exists. Such choices will have the side benefit of tending to cover the target range using the fewest number of lesson modules possible, though in the implementation presented situations can be contrived wherein the number of modules selected is not minimal. The decision as to what is "optimal" is variable; a desired result might be to use as many modules as possible, so long as each provides at least some unique coverage within the range. The definition might be based on some metric defining quality of the lesson modules, or recency, or something else entirely. The important point to keep in mind is that "optimality" must be defined using some metric, and that a given solution can be called "optimal" only within the context of that metric. In our work the term "optimal" is assumed to conform to this restrictive context.

One approach to finding optimal solutions is the so-called greedy approach (Kozen, 1992) (Neapolitan & Naimipour, 1996) (Cormen, Leiserson, & Rivest, 1995). A greedy algorithm selects the "best" choice (based on some metric) from the list of feasible candidates in a list until arriving at a solution. The selection is typically made by sorting the candidates by the optimality metric, in non-increasing order of desirability, so that the selection process is reduced to examining the list in order and selecting viable members until the solution is satisfied or found to be unreachable. By the definition of the desirability metric, each candidate so examined is the most valuable one yet unexamined. The benefit of this approach is that once a solution is found the algorithm terminates without searching for alternative solutions. Unfortunately, although greedy algorithms tend to converge to workable solutions, they can't always be guaranteed to find an optimal solution (Cormen, Leiserson, & Rivest, 1995). The key to insuring optimal solutions with a greedy algorithm is in using the *matroid* property, which Kozen defines as follows (Kozen, 1992):

matroid A pair $(S, \mathfrak{I})$ where S is a finite set and $\mathfrak{I}$ is a family of subsets of S such that

(i) if $J \in \mathfrak{I}$ and $I \subseteq J$, then $I \in \mathfrak{I}$;

(ii) if $I, J \in \mathfrak{I}$ and $|I| < |J|$, then there exists an $x \in J - I$ such that $I \cup \{x\} \in \mathfrak{I}$.

In less rigorous terms, matroids are set structures having the property that subsets can be further broken down into smaller subsets that are still members of the original set, and it is possible to transfer members from one subset to another without leaving the original set (Cormen, Leiserson, & Rivest, 1995). The chief benefit from a search space being a matroid is that there exist numerous cases of greedy algorithms proven to find optimal solutions for matroids (Cormen, Leiserson, & Rivest, 1995).

To show that concept vectors are, in fact, matroids, one needs to demonstrate that both properties from the definition above hold (Kozen, 1992):

- Property (i) is straightforward to demonstrate: take the family of subsets of vectors from the database of concept vectors, S, and call it $\mathfrak{I}$. From that $\mathfrak{I}$ take a subset J, and then from J draw a subset I. It's clear to see that I was drawn directly from the family $\mathfrak{I}$ and therefore $I \in \mathfrak{I}$ holds.
- Property (ii) is similarly straightforward: since I, J are both drawn from $\mathfrak{I}$, then if I has a smaller cardinality (fewer number of vectors) than J, then there will exist some vector x from S which is in J but not in I. If you add that x to the set I, the resulting set will still be a subset of vectors drawn from S, and therefore $I \cup \{x\} \in \mathfrak{I}$ holds.

In summary, concept vectors are in the family of combinatorial structures known as matroids and an optimal solution can be guaranteed (within the context of how the greedy selection is made) using a greedy algorithm.

Once we determined that a greedy approach was suitable, the choice of the specific algorithm to use was based on insight into the problem and its representation. The system is being presented a desired target vector with components constrained within a finite range, has knowledge of a set of candidates each of which can provide some degree of contribution towards meeting the target vector, and asked to compose an optimal subset of those candidates vectors which will sum to that target vector. However, rather than visualizing the target vector as a container into which lesson modules are stored, the true nature of the problem is closer to attempting to cover a given set with as few subsets as possible. In this manner, we chose the set covering algorithm to provide the ability for our system to dynamically construct curriculum plans based upon the concept vector representation.

Dynamic Curriculum Generation Implementation and Results

The implementation vehicle selected for this research is the Java programming language. The intent was to insure portability of the implemented code across multiple platforms, and open up the possibility for creating an intuitive user interface to reduce the burden of learning to use the system. WWW browsers are in widespread use, provide a familiar graphical user interface (GUI), and Java-compatible versions are available for nearly every computer system available to today's educator. In a full implementation, an educator would be able to visit a lesson plan resource page with a Java-enabled WWW browser and select a desired knowledge domain. An example of the initial interface

of our dynamic curriculum generator is shown in Figure 4.

The educator would then be offered a representation of the concept map defined for that domain, and be allowed to select desired concepts from the map, as well as the desired lower- and upper bound for the coverage level for each concept to be taught. This selection would then define a "target vector" representing the level of knowledge the educator desires to impart in each of the appropriate concepts. Using the summation property of component vectors (Auer, 1991), the system would return a set of titles listing the appropriate lesson materials, in order, which will cover the topics in question---in other words, a subset of the vectors in the database which will sums to the target vector.

In this context where the idea of a "best" (or optimal) solution is of interest. The educator might be interested in finding the smallest set of modules which cover the desired material, or might be operating under a set of constraints wherein modules from a certain source or possessing some other attribute are deemed more desirable than others.

Responding to feedback derived from test scores is also possible using this system. Since the teaching materials are already defined in terms of the domain's concept map, it is straightforward to map testing results against the original lesson plan. By adjusting the lower bound of the coverage of a given concept upward (for example) to exclude material successfully tested, the educator can create an updated lesson plan covering only that material the student failed to demonstrate mastery of, based on the test results.

Our prototype system was developed using tested using a notional database of twenty lesson modules, five per concept. The interested reader is directed to Dyson (Dyson, 1997) for an annotated transcript and details of each of the following sessions using the prototype system and further details of actual module implementations.

The first sample execution was a trivial one, designed to insure no spurious modules were accepted outside of the selected range. The system correctly rejected all candidate modules. The next execution was designed to select a range wherein the opportunity arose to select two modules or a single one covering the same range. The system correctly selected the single, larger module that matched the target range and rejected the two module solution. The next execution was designed to show that one of the modules rejected in the previous execution would, in fact, be selected under the proper circumstances. In this case, the previously rejected module was selected demonstrating that the sole criterion for the earlier rejection was, in fact, its suitability to cover the target range within the programmed selection criteria. The next example reinforces the selection of an optimal cover by introducing another case where multiple smaller vectors cover the same range as a larger one, only this time with some overlap. In addition, the expected selection should be two lesson modules instead of just one (if the dual coverage of the same range is incorrectly handled by rejecting all modules in that range) or three (if the system incorrectly selects the two smaller, overlapping vectors instead of the single one covering more of the range). The system rejected the smaller modules and correctly covered the target range with only two lesson modules; the optimal solution.

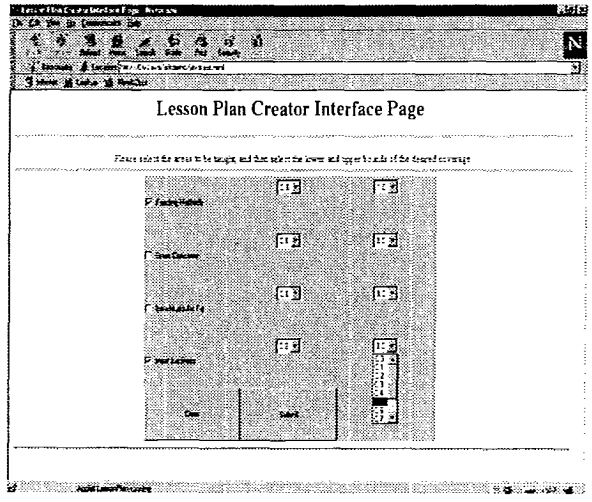

Figure 4. Prototypical WWW Interface

Future Work and Enhancements

One of the most interesting avenues of outgrowth for this system is the possibility of direct integration into an ITS. As discussed in previous chapters, units of instructional material can easily be lesson modules coded into an existing ITS. Using this system, it is entirely possible to have the lesson plan feed directly into the module selection for an ITS session. Further, with the close coupling this system enforces between lessons and concepts, test results could be fed back to the lesson plan generator directly, resulting in dynamic updating of the plan to accommodate needed remedial lessons in the next session.

Additionally, during the course of this research, numerous avenues for prototype enhancements came to mind, or were pointed out during discussions.

Graphics

Ideally, the user interface could present a graphic of the concept map for the desired domain, allowing the educator to click directly on the depicted nodes as desired and enhancing the intuitive feel of interacting with the system. In addition, generating graphical,

flowchart-like output bearing some relation to the original concept map could be useful.

Quality of Instructional Material

Each concept vector could carry an additional field of information: an evaluation of quality. Awarded when evaluated for inclusion into the database of concept vectors, this measurement would allow for qualitative selection of superior lesson materials in cases where more than one unit would otherwise provide the same coverage.

Nonlinear Coverage Within a Concept

Our research assumes blocks of coverage to be selected via an upper and lower bound, with every "bin" in between included. Our method of assigning numeric values through the quantification used in this system would easily permit selecting non-contiguous bins for coverage, but would introduce added complexity in terms of the user interface. Not only would the educator be required to click each individual bin desired, the interface itself would be tied to the number of bins---which might not remain constant between knowledge domains and must be considered.

References

Auer, J.W. (1991). *Linear Algebra, With Applications.* Prentice-Hall.

Cormen, T., Leiserson, C., and Rivest, R. (1995). *Introduction to Algorithms.* McGraw Hill.

Dyson, M. L. (1997). *Concept Vectors: A Synthesis of Concept Mapping and Matrices for Knowledge Representation in Intelligent Tutoring Systems.* Master's Thesis, AFIT/GCS/ENG/97D. Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH.

Ely, D., Januszewski, A., and LeBlanc, G. (1988). *Trends and Issues in Educational Technology 1988.* Syracuse University Press.

HQ AETC/XORR. (1984). *AF HANDBOOK 36-2235.*

Kozen, D.C. (1992) *The Design and Analysis of Algorithms.* Springer-Verlang.

Neapolitan, R. and Naimipour, K. (1996). *Foundations of Algorithms.* Heath.

Novak, J. and Bowin, D. B. (1984). *Learning How to Learn.* Cambridge University Press.

O'Shea, T. and Self, J. (1983). *Learning and Teaching with Computers--Artificial Intelligence in Education.* Prentice-Hall.

Regis, A., Giorgio-Albertazzi, P., and Roletto, E. (1996). "Concept Maps in Chemistry Education," *Journal of Chemical Education,* 73(11).

Winston, P. H. (1993). *Artificial Intelligence (Third Edition).* Addison-Wesley.