AUTOMATED KNOWLEDGE ACQUISITION AND DYNAMIC CURRICULUM SYNTHESIS FOR INTELLIGENT TUTORING SYSTEMS

Sheila B. Banks, Eugene Santos, Jr.[†], Mark L. Dyson, and F. Alex Kilpatrick[‡]
Artificial Intelligence and Human-Computer Interaction Laboratory
Air Force Institute of Technology
Wright-Patterson Air Force Base, OH 45433
sbanks@afit.af.mil

[†] Computer Science and Engineering Dept.
University of Connecticut, Storrs, CT
eugene@cse.uconn.edu

[‡] Air Force Office of Scientific Research
Bolling Air Force Base, Washington, DC
freeman.kilpatrick@afosr.af.mil

ABSTRACT

To realize the promise of fielding an intelligent tutoring system (ITS), the ITS requires a knowledge base from which to draw instructional source content. That knowledge must be first acquired and then represented in a tractable form; useful from both a computing standpoint and the point of view of presenting that knowledge to a student.

In general, intelligent tutoring systems research to date has focused on the student and on methods for representing the student knowledge. 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 within an ITS environment. What is needed, before attempting to design and develop an ITS, is a methodology for defining and developing student curricula in a form directly related to ITS implementation. This methodology should be quantifiable both in terms of content and applicability, and able to accept feedback metrics on a given student's progress to modify the lessons and the curriculum plan. In addition, the tool should be useful without requiring excessive training.

In the literature, one finds numerous examples of knowledge representation schemes, from the idea of concept mapping to the hierarchical databases used in the Air Force's Instructional System Development (ISD) project. Even when projects provide an automated tool capability, educators face steep learning curves, a wide array of user interfaces, and a significant amount of manual development when constructing student curricula.

Our approach employs an automated knowledge acquisition tool, PESKI, to acquire the necessary information for student curriculum generation, utilizes *concept mapping* to represent that knowledge, and then maps that representation into *concept vectors*. We developed a prototype system, based on concept vectors, that accepts inputs from an educator via a world wide web (WWW) interface and returns a dynamic lesson plan.

ABOUT THE AUTHORS

Sheila B. Banks is an active duty Major in the U.S. Air Force serving as an Assistant Professor of Computer Engineering at the Air Force Institute of Technology, Department of Electrical and Computer Engineering, Air Force Institute of Technology. She received a B.S. in Geology from University of Miami, Coral Gables, FL (1984) and a B.S. in Electrical Engineering from North Carolina State University, Raleigh, NC (1986.) Also from North Carolina State University, she received a M.S. in Electrical and Computer Engineering (1987) and her Ph.D. in Computer Engineering (Artificial Intelligence) from Clemson University, Clemson, SC (1995.) Her research interests include artificial intelligence, intelligent computer generated forces, associate and collaborative systems, distributed virtual environments, intelligent human computer interaction, and man-machine interfaces.

Eugene Santos Jr. is an associate professor of computer science at the University of Connecticut, Storrs, CT. He received his B.S. in Mathematics and Computer Science (1985) and M.S. in Mathematics -- Numerical Analysis (1986) from Youngstown State University (1985) and subsequently completed a Sc.M. (1987) and Ph.D. in Computer Science -- Artificial Intelligence (1992) from Brown University. His research interests include automated reasoning, neural networks, natural language understanding, expert systems, machine learning, operations research, probabilistic reasoning, robotic planning, temporal reasoning, combinatorial optimization, numerical analysis and parallel processing.

F. Alex Kilpatrick is an active duty captain and program manager for software and systems at the Air Force Office of Scientific Research (AFOSR), Bolling AFB, D.C. He holds a Ph.D. in Computer Engineering from the Air Force Institute of Technology. His research interests include machine learning, intelligent tutoring systems, and collaborative learning environment.

AUTOMATED KNOWLEDGE ACQUISITION AND DYNAMIC CURRICULUM SYNTHESIS FOR INTELLIGENT TUTORING SYSTEMS

Sheila B. Banks, Eugene Santos, Jr.[†], Mark L. Dyson, and F. Alex Kilpatrick[‡]
Artificial Intelligence and Human-Computer Interaction Laboratory
Air Force Institute of Technology
Wright-Patterson Air Force Base, OH 45433
sbanks@afit.af.mil

[†] Computer Science and Engineering Dept.
University of Connecticut, Storrs, CT
eugene@cse.uconn.edu

[‡] Air Force Office of Scientific Research
Bolling Air Force Base, Washington, DC
freeman.kilpatrick@afosr.af.mil

INTRODUCTION

The application of artificial intelligence techniques to education practices has, for many years, been declared to be "inevitable" (11). However, despite ever-growing focus on the computer as a dominant medium in the field of educational technology (5), expert opinion concerning the utility of artificially intelligent teaching tools, normally termed *intelligent tutoring systems*, leads to the conclusion that such programs are generally of poor quality (5).

In addition, the bulk of intelligent tutoring system (ITS) research tends to focus on student modeling and knowledge presentation to the student; in short, on the student's perspective within the learning environment. What is required, before beginning work on an intelligent tutoring system, is a methodology for defining and developing student curricula in a form directly suitable to an ITS implementation. In other words, an intelligent tutoring system (ITS) must have a knowledge base from which to teach. Furthermore, for the ITS to utilize that knowledge base in a fieldable system, three steps must be performed. The knowledge for the ITS must be acquired from a domain expert in the area from which the ITS is to teach, the knowledge must be represented in a tractable form from a computing standpoint, and the knowledge must be represented in a way that facilitates presenting that knowledge to a student.

A number of knowledge representation schemes are available for ITS use, from the idea of concept mapping (10) to the intensive, hierarchical databases used in the Air Force's Instructional System Development (ISD) project (7). However, even in the case of automated tools such as provided in the ISD project, educators face steep learning curves, a lack of a standardized, usable interface, and a significant amount of manual development when constructing lesson plans (7).

The requirement, then, is for a methodology to consistently define and develop student curricula in a form directly suitable to an ITS implementation directed towards knowledge presentation to the student. This methodology should 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 necessary. In addition, any tool to acquire the necessary knowledge and provide this capability should be useful without requiring excessive training.

Our approach employs an automated knowledge acquisition tool, PESKI, to acquire the necessary information for student curriculum generation, utilizes a respected and long-standing method of representing this complex knowledge in the classroom, called *concept mapping*, and creates a system for mapping that representation into a form amenable to computer manipulation termed *concept vectors*. We developed a prototype system, based on concept vectors, 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 then return a dynamic student lesson plan.

This paper begins by providing a brief background on the automated knowledge acquisition environment, PESKI, utilized to elicit the necessary concepts for student curricula development. We then discuss the information that forms the educational basis of concept vectors: educational teaching theory and concept mapping, including an illustrative example of a concept map. We then show how a concept map may be used by a knowledge engineer to define teaching materials within a sample domain. We then discuss concept vectors, motivate their usefulness in representing concepts in computer-readable form, and provide theoretical grounding for an effective approach for creating a dynamic curriculum. We will also discuss the prototype system and the paper concludes with a discussion of how the prototype

could be enhanced via future research, including automated interfaces between the curriculum builder and an actual ITS.

BACKGOUND

Concepts from knowledge acquisition and engineering and educational theory form the basis for our ITS dynamic curriculum system development. This section will discuss our automated knowledge acquisition environment, describe concepts utilized from educational theory for curriculum development, and elaborate on the knowledge engineering technique of concept mapping. These concepts were integral in developing our idea of concept vectors and initiating its use for dynamic curriculum building.

Probabilities, Expert System, Knowledge and Inferencing (PESKI) Environment

Within any knowledge acquisition intensive project, such as the one posed by the problem of acquiring the knowledge from which to generate ITS lesson plans, PESKI provides an automated knowledge engineering environment in which to work. The PESKI environment (18, 19, 20) is a complete knowledge engineering tool and consists of four main components: (1) Natural Language and Graphical User Interface that translates natural language questions into inference queries and translates the analyses/inference results back into natural language and supports all communication between the user and the system; (2) Inference Engine that contains the strategies for controlling the selection and application of knowledge and facts in our knowledge base; (3) Explanation and Interpretation that keeps track of the reasoning paths the inference engine used in reaching its conclusions; and (4) Knowledge Acquisition and Maintenance that provides the facility for automatically incorporating new or updated expert knowledge into the knowledge base.

Because there is a natural intersection among the necessary components of a knowledge engineering environment and the four previously discussed components perform multiple functions, PESKI combines the different components together into three architecturally separate subsystems. (1) The PESKI user interface is composed of the Natural Language and Graphical User Interface and the Explanation and Interpretation components. (2) PESKI Knowledge Organization and Validation consists of the Explanation and Interpretation component along with the human expert, optional knowledge engineer, and knowledge engineering tools. Organization is accomplished by communicating with the Knowledge Acquisition and Maintenance component. Validation is similarly accomplished except that we also have feedback from the Reasoning Mechanism through Explanation and Interpretation for debugging purposes. (3) The PESKI Reasoning Mechanism consists of the Inference Engine and the Knowledge Acquisition and Maintenance components. Our premise behind incorporating the Knowledge Acquisition and Maintenance component is that we believe some form of reasoning and possibly learning must take place for any new knowledge to be merged into our existing knowledge base. Problems such as consistency clearly must be addressed within the tool framework. Furthermore, we can also achieve a useful degree of information hiding in this manner. Under this arrangement, it is not necessary for any subsystem outside of the Reasoning Mechanism to be aware of the particular choice for knowledge representation. One exception is the Explanation and Interpretation component; however, it only needs to read and interpret the knowledge base information. As we can see, the PESKI architecture is sufficient for constructing knowledge base systems in nearly any domain, especially the knowledge intensive domain of intelligent tutoring systems.

General Educational Theory

General educational theory on how students actually learn was crucial to the initial idea of concept vectors. Two papers written in the 1930s proposed theories on how people learn. In 1930, E. R. Guthrie published the “contiguity theory” (6) that asserted that once a learner has made a link between an action and learning from that action, repetition of that action is beneficial to the continued learning process. This assertion, as well as that made by E. Thorndike in 1932 (16), states that not only does learning depend on new situations containing elements of earlier ones but also learning responses tend to chain together towards a goal. These assertions contributed to our representation of learning within a single concept as a linear progression, having a beginning and end point, and covering the entire range in between endpoints. Further, Guthrie (6) contends that instruction must present very specific tasks, which is in keeping with our emphasis on keeping our knowledge representation nodes as atomic and specific as possible.

Current theorists in the educational theory literature tend to bear out these earlier works, even in the current culture with the use of computers in the learning process. Trend watchers such as Ely (5) and O'Shea (11) stress how the process of learning transcends computing, rather than being overshadowed by it. In addition, educational

researchers, such as Brown (2), point out that underneath sophisticated computing tools designed to aid learning, the learning process itself has remained relatively unchanged.

Curriculum Building for Intelligent Tutoring Systems

As mentioned previously, ITS literature shows a marked lack of material relating to the educator's problem in composing ITS curricula plans; the bulk of the material available centers on the problems of the teaching process itself. For example, Shute and Psotka's visionary paper detailing their view of the sum of all ITS research (15), only mentions the concept of "curriculum" as an important node in their ITS model. However, this paper glosses over the process of how that curriculum might be obtained, relegating it to a problem to be handled by domain experts. They do point to the issue that this domain expert problem is a difficult one, which underscores the cautions offered in the conclusions of this paper concerning the importance and difficulty of the knowledge engineering effort required to transition our initial prototype system into a real-world system. Shute's presented bias towards student models and learning behavior becomes evident in previous research for the Air Force (13, 14), and sets the scene for the general climate in ITS research to date.

Concept Mapping

To begin any knowledge-based system like an intelligent tutoring system, one first needs a workable system for representing that knowledge (17). One very good representation for potentially abstract knowledge domains was developed in the mid-eighties by Joseph Novak and Bob Gowin (10). Not only does their 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 (12). Our use of concept maps provides a knowledge representation basis well grounded in current educational practice and a framework for representing even complex knowledge domains in a disciplined and easily-computable fashion.

Concept maps represent an approximation of the relevant concepts and propositions of a given knowledge domain (10), 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 (10). 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.

FROM CONCEPT MAPS TO CONCEPT VECTORS

Our methodology for defining and developing student curricula in a form directly suitable to ITS implementation begins with the quantification of the knowledge acquired through PESKI and expressed within the concept map. This quantification process takes 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 decomposition of a potentially complex domain into the progressively smaller conceptual components used in concept mapping is the central idea in representing knowledge as vectors. A vector is made of components, each representing a magnitude along a single dimension in *n*-dimensional space (1). Within a vector we must ensure that the vector maintains consistent meaning; therefore, each component must have no contribution to the magnitudes of the other components in that vector. Relating this aspect to the concept mapping domain, if a vector represents the entire knowledge domain to be used within the ITS, then each concept, shown as a node, within the overall concept map can be thought of as a component of that vector. In addition, when going through the knowledge acquisition process, the knowledge engineer must decompose each concept as much as possible to prevent overlap between concepts---just as vector components provide a magnitude contribution only in their dimension.

Once a given concept is defined at the atomic level, 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 (10). The domain expert might conclude that

there are a finite number of types of grass that 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 of grass.

Given this conclusion, the domain expert establishes a metric where the knowledge of “grass” can be measured on a scale from total ignorance to complete mastery. After establishing such a metric for each domain concept required for inclusion in the ITS, the metrics are normalized so that complete ignorance is represented by zero, and total mastery by one. Applying this numerical measure to each of the concepts makes it possible to represent the concepts in a machine-readable form and to perform computations based on levels of mastery of the knowledge domain. Further, the ability to represent any given level of knowledge within the domain by this string of numbers, a vector, which follows known mathematical properties becomes a powerful tool for knowledge representation.

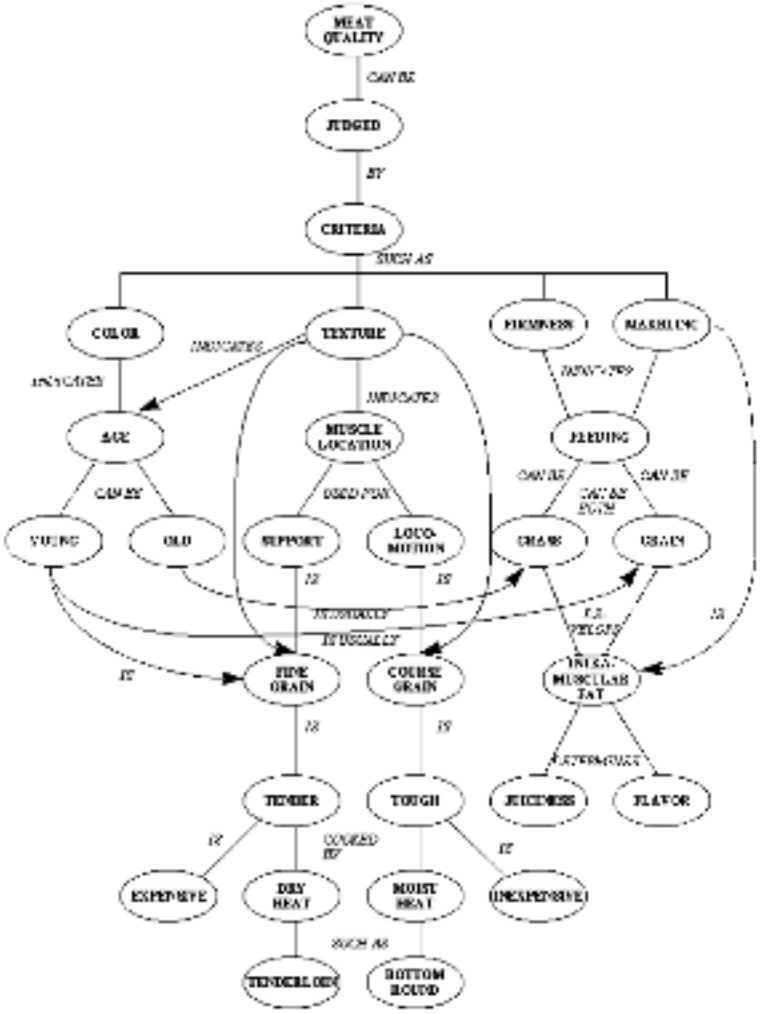

Figure 1. Complete Concept Map of Meat Science Course (10)

Mapping Knowledge to Concept Vectors

After a knowledge engineer defines a concept map for a given domain, the first step of the mapping process requires that the teaching material available to develop student curricula 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 --- 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 the 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 available instructional material, each relating to a different concept.

The second step of the mapping process is to determine how much of the concept each given instructional unit covers. Recall that for each concept in the domain, the span from complete ignorance to full mastery is represented on a normalized scale from 0.0 to 1.0. As an instructional unit is evaluated against a given concept, the domain expert determines what portion of the given concept the instructional unit covers 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 as in general education theory. The act of defining a unit of teaching material in this context assumes contiguous coverage from lower to upper bound within the assigned span defined by the start and end values, which delineates the smallest unit of teaching material in this context. If an evaluated instructional unit does, in fact, have gaps in coverage as determined by the domain expert, that instructional unit should be further divided into smaller instructional 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.

Consider Figure 2, a subset drawn from Figure 1, as an illustrative example. 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 mastery of the concept. Note that, while none of the three vectors shown covers the whole concept, the three taken together do provide complete concept coverage.

In the example given in Figure 3, there is some overlap in the depicted vectors. This case is indicative of the most common situation; not all lesson materials available to an educator can be assumed to fit together without some overlap in how a concept is covered. 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, as shown in Figure 3. In the case of the 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.

After completely defining the knowledge domain as an n-dimensional vector space by a knowledge engineer working with a domain expert, and with the available teaching materials defined within that space by an educational or domain specialist, the power of this representation is apparent. Expressing the teaching material using a numerical range allows it to be summed as vectors. The vector sum enables evaluation of either the completeness and topical coverage of available material within the context of the overall domain or within the context of instructional curriculum determination. Elaborating on the latter case, the units selected to contribute to the target vector map directly to the teaching materials the educator requires to build a lesson plan designed to teach given concepts to a desired level. The units thereby form the basis of an ability to dynamically build and modify instructional curricula either for use directly by the instructor or for input into an ITS for instructional sequencing.

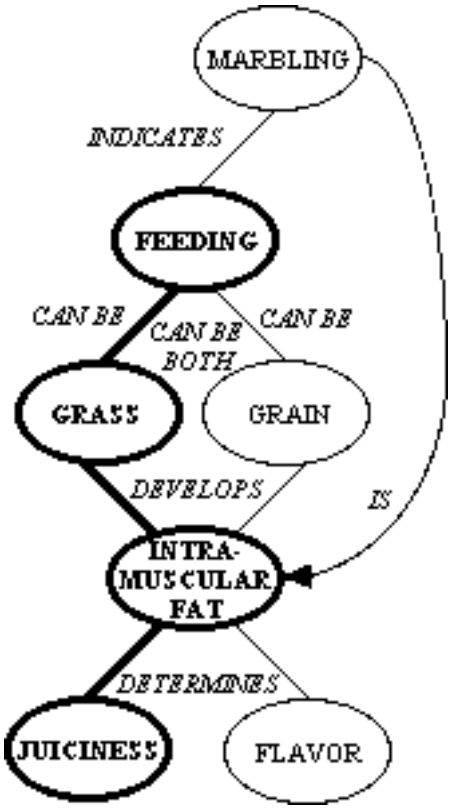

Figure 2. Subset Concept Map of Meat Science Course (10)

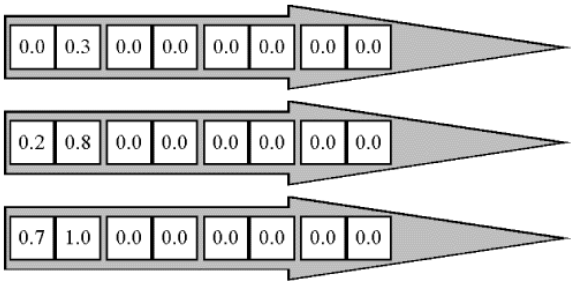

Figure 3. Three Defined Concept Vectors

The forms that lesson modules could take should be re-visited at this point. Previous discussions have mentioned such instructional items in written form as pamphlets and sections from textbooks. In practice, these sources, while of immense value to our methodology, will most likely require time and expertise to quantify as precisely as described within this paper. We expect that the design methodology presented here will be easiest to integrate and will prove most useful when interfacing with an established ITS, presumably with a database of lesson modules already defined and coded for units of instructional content. In this 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 of an obstacle, 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

Upon completing concept vector definition for use for any ITS domain, we then require a technique for utilizing the concept vectors to produce a student curriculum for use within the ITS. 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 technique, and the implementation of both the representation and manipulation algorithm for the dynamic curriculum builder.

Dynamic Curriculum Generation Approach

The first step in any approach is to identify the general form of the problem. As with many systems framed as a general search problem, search space pruning and early arrival at optimal solutions is both desirable and often difficult (17). Within the context of our development, "optimality" is defined as the curriculum generator consistently selecting a lesson module that provides the largest overall amount of coverage (which still lies completely within the target range) whenever such a choice exists. Such a choice has the side benefit of tending to cover the target instructional range using the fewest number of lesson modules possible. However, in the presented implementation, situations can be artificially contrived wherein the number of modules selected is not minimal. The definition of "optimality" can be established by the instructor to meet desired educational objectives. One desired result might be to use as many modules as possible, so long as each provides at least some unique coverage within the range. Another definition of "optimal" might be based on some metric defining quality of the lesson modules, or recency of module use, or some other metric entirely. The important point to keep in mind is that "optimality" must be defined using a chosen metric, and, furthermore, a given solution can be termed "optimal" only within the context of that

metric. In our work, the term "optimal" is assumed to conform to this restrictive context.

To arrive at an optimal solution, we employed a greedy algorithmic approach (8, 9, 3). A greedy algorithm selects the "best" choice (based on some metric previously determined to satisfy the user objective) from the list of feasible candidate solutions. The selection of the candidate to use is typically made by sorting the candidates by the optimality metric, in non-increasing order of desirability, and then testing the list. The initial sort serves to reduce the candidate selection process to examining the candidates on the list in order until a candidate that achieves the desired goal condition is found or the desired goal is found to be unreachable. By the definition of the optimality metric and the sorting of the candidate list, each candidate examined in this method is the most valuable one yet unexamined. The benefit of a greedy approach is that once a solution is found the algorithm terminates without searching for further solutions. Unfortunately, although greedy algorithms tend to converge to workable solutions, they can not always be guaranteed to find an optimal solution (3). The key to insuring optimal solutions while employing a greedy algorithm is in using the *matroid* property, which may be defined as follows (8):

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

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

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

In less mathematically formal terms, matroids are set structures having the property that subsets can be further broken down into smaller subsets where the members of the smaller subsets are still members of the original set, and it is possible to transfer members from one subset to another without leaving the original set (3). 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 (3).

To use the greedy algorithmic approach to find optimal solutions for our dynamic curriculum builder, concept vectors must be shown to be matroids. This demonstration requires that both properties from the definition above hold (8):

- Property (i) is straightforward to demonstrate: take the family of subsets of vectors from the database of concept vectors, S, and call it $\grave{}$. From that $\grave{}$ take a subset J, and then from J draw a subset I. It is clear to see that I was drawn directly from the family $\grave{}$ and therefore $I \in \grave{}$ holds.
- Property (ii) is similarly straightforward: since I, J are both drawn from $\grave{}$, 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 $I \cup \{x\} \in \grave{}$ holds.

In summary, concept vectors are within the family of combinatorial structures known as matroids. Therefore, an optimal solution can be guaranteed to be found using a greedy algorithm. Once again keeping in mind that optimality is only defined within the context of the metric for the greedy algorithm selection process.

After determining that a greedy algorithmic approach was viable, the choice of the specific greedy algorithm to use was based on insight into our problem and its chosen concept vector representation. The system is given a desired target vector with components constrained within a finite range. The ITS has knowledge of a set of candidates that can possibly provide some degree of contribution towards meeting the target vector. Finally, the system is asked to compose an optimal subset of those candidate vectors that will sum to the desired 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.

The Dynamic Curriculum Generation System and Testing Results

The goal of the curriculum generation system was to provide the educator an automated tool that addressed the deficiencies of current instructional generation systems. The idea was to present the educator with a standardized, usable interface supporting a fully automated curriculum generation capability. To this end, we implemented our system utilizing the Java programming language. The implementation method also insures portability of the implemented code across multiple platforms to accomodate the variety of hardware available to educational institutions and provides the possibility for creating an intuitive user interface to increase the usability of the developed

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.

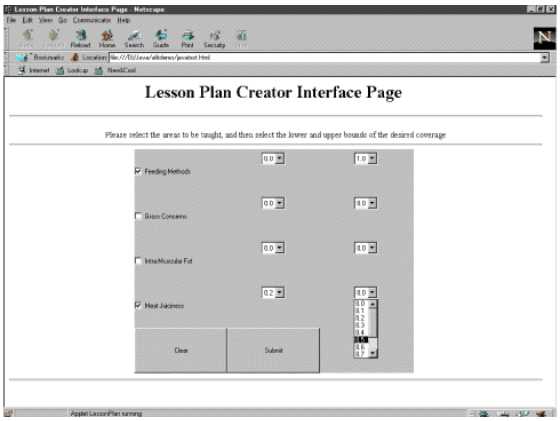

Figure 4. Prototypical WWW Interface

During system use, the educator is offered a representation of the concept map defined for the selected domain. The educator then selects desired concepts from the map along with the desired lower- and upper bounds for the coverage level for each concept to be taught. The selections by the instructor defines a "target vector" representing the level of knowledge the educator desires to impart to the student in each of the appropriate concepts. However, the instructor is removed from this system definition and the use of concept vectors and simply interacts with the ITS interface at the concept map level. After concept and boundary choice by the instructor, the system returns a set of titles, using the summation property of component vectors (1). The returned information shows the appropriate lesson materials, in order, that will cover the topics in question---in system implementation terms, a subset of the vectors in the database that will sum to the target vector.

The choice of a metric to decide the "best" (or optimal) solution is of interest here. The educator might desire to find the smallest set of instructional modules to 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 another instructional module.

Directly and automatically responding to feedback derived from test scores is also possible for the educator or for an ITS utilizing our dynamic curriculum building capability. Since the teaching materials are previously defined in terms of a domain concept map, there is a straightforward comparison process between the testing results against the original lesson plan. Once the comparison is completed, by adjusting the lower bound of the coverage of a given concept upward, for example, to exclude material successfully tested, the educator creates an updated lesson plan covering only that material the student failed to demonstrate mastery of, as demonstrated by previous test results.

Dynamic Curriculum Generation Testing

Our prototype system was tested using a notional database of 20 lesson modules, 5 per concept. The interested reader is directed to Dyson (4) for an annotated transcript and details of each of the following sessions using the prototype system and for further details of actual module implementations.

The first sample execution and test 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 test 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. This selection of the larger module conforms to the previously defined optimality metric utilized within the definition of the greedy algorithm. The next test was designed to show that one of the modules rejected in the previous test condition would, in fact, be selected under the proper circumstances. In this test 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.

FUTURE WORK AND ENHANCEMENTS

The most promising avenue of outgrowth for future work utilizing our system is the direct integration of dynamic curriculum building into an ITS. As discussed previously, units of instructional material can easily be lesson modules coded into an existing ITS. Using this system, the lesson plans may feed directly into the module selection for an ITS session. Further, with the close coupling this system enforces between lessons and concepts, test results can be returned to the curriculum generator directly, resulting in dynamic updating of the plan to accommodate needed remedial lessons in the next session.

Throughout the course of our work, numerous opportunities for prototype enhancements became readily apparent, or were pointed out during discussions.

Graphics

The current system interface is text-based. Ideally, the user interface should present the concept map for the desired domain in its native graphical form, which allows the educator to click directly on the depicted nodes as desired and enhances the intuitive feel of interacting with the system. In addition, generating graphical, flowchart-like output of the selected concepts for instruction and the corresponding modules of instruction to cover the selected concepts would be immediately useful to the educator.

Quality of Instructional Material

The incorporation of a quality measure associated with each instructional unit, an instructional unit weighting factor, is needed to increase the usefulness of the curriculum generating capability. Each concept vector can carry an additional field of information, which would correspond to this evaluation of unit quality. Evaluated and determined 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

Within the current methodology, which maps the concept map to vector space using an assigned range of coverage, we assume blocks of coverage to be selected by an upper and lower bound, with the entire range in between the upper and lower bound included. Our method of assigning numeric values through the quantification used in this system easily permits selecting non-contiguous ranges for coverage. However, the addition of these non-contiguous ranges introduces added complexity in terms of the user interface and its representation of the vector space. Not only would the educator be required to select each individual range desired, the interface representation itself could easily become tied to the number of possible ranges. The number of ranges is certainly not guaranteed to remain constant between knowledge domains and must be examined within the generic system interface design and implementation. To implement this type of concept coverage, alternate presentation of the vector space and the interaction necessary between the educator and the curriculum generator should be carefully considered.

In summary, our work addresses the requirement to automatically define and develop student curricula, utilizing minimal instructor support, in a form directly suitable to ITS implementation. Our methodology is quantifiable and able to accept feedback metrics on student progress to allow automatic update of the presented lessons and the curriculum plan. With the integration of the curriculum generation capability within the PESKI environment, we can directly acquire the necessary knowledge from the ITS domain in the required form. Finally, due to the emphasis placed upon the user interface, the capability provided by our integrated knowledge acquisition and curriculum generation tool is readily useful without requiring excessive training for the educator.

REFERENCES

1. Auer, J.W. (1991). *Linear Algebra, With Applications.* Prentice-Hall.
2. Brown, J. S., Collins, A., and Duguid, S. (1989). “Situated Cognition and the Culture of Learning,” *Educational Researcher*, 18(1).
3. Cormen, T., Leiserson, C., and Rivest, R. (1995). *Introduction to Algorithms*. McGraw Hill.
4. 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.
5. Ely, D., Januszewski, A., and LeBlanc, G. (1988). *Trends and Issues in Educational Technology 1988*. Syracuse University Press.

6. Guthrie, E. R. (1930). "Conditioning as a Principle of Learning," *Psychological Review*, 37.
7. HQ AETC/XORR. (1984). *AF HANDBOOK 36-2235*.
8. Kozen, D.C. (1992) *The Design and Analysis of Algorithms*. Springer-Verlang.
9. Neapolitan, R. and Naimipour, K. (1996). *Foundations of Algorithms*. Heath.
10. Novak, J. and Bowin, D. B. (1984). *Learning How to Learn*. Cambridge University Press.
11. O'Shea, T. and Self, J. (1983). *Learning and Teaching with Computers--Artificial Intelligence in Education*. Prentice-Hall.
12. Regis, A., Giorgio-Albertazzi, P., and Roletto, E. (1996). "Concept Maps in Chemistry Education," *Journal of Chemical Education,* 73(11).
13. Shute, V. J. (1990). "Individual Differences In Learning From an Intelligent Discovery World: Smithtown," Air Force Research Laboratory Technical Report AFHRL-TP-89-57, Manpower and Personnel Division, Brooks Air Force Base.
14. Shute, V.J., et al. (1992). "Modeling Practice, Performance, and Learning," *Simulation-Based Experiential Learning*, D.M. Towne, et al. (Editors), Berlin:Springer-Verlag, pp. 133-145.
15. Shute, V. J. and Psotka J. (1994). "Intelligent Tutoring Systems: Past, Present, Future," Air Force Research Laboratory Technical Report AL/HR-TP-1994-0005, US Air Force, Armstrong Laboratory.
16. Thorndike, E. (1932). *The Fundamentals of Learning*. Teachers College Press.
17. Winston, P. H. (1993). *Artificial Intelligence (Third Edition)*. Addison-Wesley.
18. Santos, E. Jr., Banks, D. O., and Banks, S. B. (1997). "MACK: A Tool for Acquiring Consistent Knowledge Under Uncertainty," *Proceedings of the AAAI Workshop on Verification and Validation of Knowledge-Based Systems*, pp. 23-32, Providence, RI.
19. Santos, E. Jr., Gleason, H. T., and Banks, S. B. (1997). "BVAL: Probabilistic Knowledge-Base Validation," *Proceedings of the AAAI Workshop on Verification and Validation of Knowledge-Based Systems*, pp. 13-22, Providence, RI.
20. Brown, S. M., Santos, E. Jr., and Banks, S. B. (1998). "Supporting Incremental Knowledge Elicitation in Decision-Theoretic Systems," *Proceedings of the AAAI Spring Symposium on Interactive and Mixed-Initiative Decision-Theoretic System*s, pp. 14-15, Palo Alto, CA.