# The Bayesian Advisor Project I: Modeling Academic Advising

**Alex Dekhtyar, Judy Goldsmith, Huaizhi Li** and **Brett Young**
Computer Science Dept.
University of Kentucky
Lexington, KY 40506-0046
{dekhtyar,goldsmit,hli3,brett}@cs.uky.edu,

## Abstract

An academic advisor's job requires that the advisor knows university and departmental requirements; available courses to satisfy those requirements, and a method for predicting relative student success for various course options. Such predictions are necessarily uncertain, both because the advisor has limited information about the student and because student performance can be influenced by unforeseen factors. We present a preliminary Bayes Net model of academic advising. In it, the advisor's predictions are based on student performance so far, as described in a student transcript. We describe the underlying model and some initial experiments.

## 1 INTRODUCTION

Academic advising is a required and often under-rewarded part of many academics' job descriptions. An advisor's job consists of two parts: knowing the course offerings and the requirements relevant to the students' academic careers, and interacting with individual students. Both are complicated, dynamic, and time-consuming.

Ideally, the advisor bases her advice on her knowledge of the student's interests and abilities. Ideally, this knowledge would be accurate. However with a student to faculty ratio of approximately 40:1, the ideal of personal attention to each student's needs and desires is almost inevitably compromised. Much of the faculty advisor's time during an advising session is spent reading student transcripts, which provide the advisor with the bulk of the information about student's performance. The advisor has to discover deficiencies in requirement-meeting, and advise the student to meet those requirements in some manner the advisor presumes to be optimal.

One approach to optimality is to consider the alternatives available for each requirement, to project the *probabilities of success* for each individual course or combination of courses, and to try to find a set of courses that has either optimal or acceptable probability of success and enables the student to move toward her goal of completing the degree with reasonable GPA in reasonable time The key to finding a good suggestion is the advisor's ability to *accurately assess the student's abilities.*

Clearly, such assessement is bound to be *uncertain.* To a large extent, a student's true abilities can be considered *hidden variables* in the advising process. The only indication of student abilities that the advisor has comes in a form of a student transcript and some supplementary materials such as SAT, ACT, GRE or TOEFL scores which all can be considered *observable variables.* The goal of advisor, thus, is twofold: (i) she has to predict the advisee's abilities (hidden variables) based on past performance (observable variables); (ii) based on these predictions, she has to find a satisfactory recommendation (based on the estimate of the probability of success).

The second step of this process must necessarily be *uncertain* as well. In addition to the uncertainty associated with the prediction of student abilities, there is always an inherent uncertainty due to unforseen circumstances. E.g., although a student may begin a semester in full command of all the skills and knowledge to successfully complete a course, may have excellent study skills and habits, personal problems or a sudden dislike of the material may adversely affect the outcome. Conversely, the student may finally decide to get serious about her school work, or may acquire a friendship or other relationship that includes significant tutoring, or may be inspired by a professor to work beyond her earlier limits. The student may suddenly improve beyond the advisor's expectations.

The Bayesian Advisor Project models student success on a course-by-course basis as a Bayes net [14]. The goal is to automate the decision-making component of academic advising, in order to free faculty to focus on the human interactions that form such a vital part of academic advising.

We describe the actual Bayes net model in Section 2, and our pilot implementation in Section 3. We then describe the proposed use of the Bayesian Advisor in Section 4. There are a number of interesting questions that arise in the process of developing the model and acquiring the data. These questions are addressed in in Section 5.

## 2 ADVISING AS A BAYES NET

The goal of the Bayesian Advisor, in small, is to predict a student's performance in subsequent courses based on their performance in courses taken so far. It hinges upon two crucial assumptions about the underlying domain that we are making:

- student performance in future classes *chiefly depends* on particular knowledge and abilities possessed by a student, and
- *observable variables* available to us, such as student transcript and test score information allow fairly accurate prediction of these abilities (*hidden variables*).

### Random Variables

In developing our model, we are (at least initially) restricted to the use of *observable random variables*. We break these variables into two basic categories: *performance in university courses* and *supplemental information*. Thus, performance in each university course will be a unique random variable, and therefore, a node in the network. Supplemental information, or overall predictors, such as current grade point average (GPA), High School GPA, as well as exam scores (SAT, ACT, GRE, TOEFL, etc.) will be the other nodes. Figure 1.a shows some possible random variables for a model of our Computer Science department academic program.

When a correspondence between *observable* and *hidden* variables in a model is assumed to be well established, these hidden variables may be added to the network explicitly. Thus, the Advisor Bayes Net may have nodes for certain skills/abilities, such as mathematical maturity, analytic ability, etc.

**Random variable domains (state spaces).** Random variables representing student performance in university courses have values that are specified by the University grading policies. E.g., at the University of Kentucky, a grade in the course is one the following: A, B, C, D, E (failure), W (withdrawal), X (failure due to cheating), I (incomplete). In addition, for specifying conditional probabilities we can add the value NT, not taken, to the list. In reality, other lists of values can be of use as well. To keep the size of our experiments (see Section 3) under control, we use the set {AB, C, E} of values where AB stands for "A or B" grade. We note that different course variables may have different sets of values used, and in fact, the same random variable can be used with different "compatible" sets of values (i.e., reducible to the same basic set) in different distributions in the Advisor Bayes net. The problem of managing different domains had been addressed by Laskey and Mahoney in [12]

Domains of other variables included in the model need to be determined in advance as well. The original domains of test scores are typically too large, and need not be represented directly. Instead, we may want to represent *qualitative jumps* in the scores. E.g., the set of values for TOEFL score can be {Fail, Probation, OK, Excellent} and would correspond (for old scores) to scores that are respectively less than 530, between 530 and 550, between 550 and 600 and greater than 600. For some other tests, the "break points" are determined by the *percentile* value rather than by a raw score. Thus, the random variable representing GRE Verbal score can have values {Top5, Top10, Top25, Top50, Bot50} which represent the position of the raw score among other scores rather than its absolute value.

### Structure of the model

The procedure for building the structure of an Advisor Bayes Net depends on the scope of the particular model desired. Within the university hiererchy we can identify the following scopes:

- single departmental degree program (e.g., M.S. in Computer Science);
- single department (e.g., all programs offered by Computer Scince department);
- college/group of departments (e.g., all programs offered by College of Engineering);
- university (universal advisor).

We expect that the difference in the size of the Advisor Bayes Net between the smallest and the largest scope within a typical public university is around two

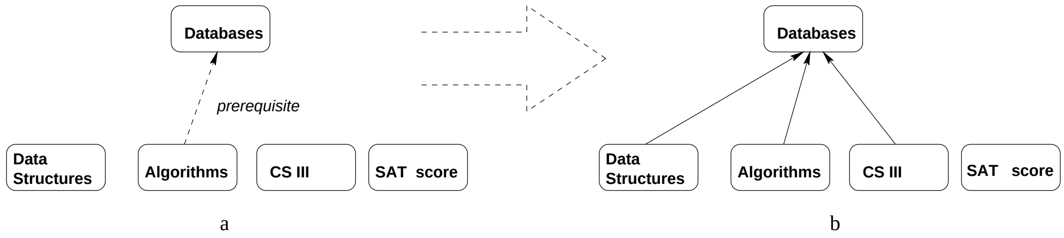


Figure 1: From stand-alone random variables and course prerequisite information (left) to building Advisor Bayes Net (right).

orders of magnitude[1]. Here, we will concentrate on constructing the smaller scope (20 - 200 nodes) Advisor Bayes Nets, addressing the additional problems that arise for large scope nets in Section 5.

For an Advisor model for a single academic program or a single department, a rough estimate of the structure of the Bayes net can be read off of the prerequisite structure for the courses (see Figure 1.a). However, this does not reflect the full information about a student's performance in a given course. Prerequisites, usually decided by departmental committees, tend to reflect the necessity of allowing students to graduate in some small number of semesters (8–12, perhaps). Thus, courses that teach skills or approaches useful for some later courses are often not listed as a prerequisite. Furthermore, a prerequisite may stand in for a skill set and presumed academic maturity, rather than the exact content of the course. For instance, some advanced math courses require calculus in order to bar first-year students and those who have never seen formal mathematics, although they do not depend on calculus. (In the U.S., calculus is a prerequisite for linear algebra; in Russia, the order is reversed.) Therefore, the Bayes net structure used in a Bayesian advisor model will be a modification of the prerequisite structure, based on information elicited from experts (see Figure 1.b for an example of such modification). The experts could be academic advisors, in the case of a reasonably small program, or may need to be professors who are familiar with each individual course.

### Conditional probabilities

There are two potential sources of the conditional probabilties:

- statistical information extracted from a(n anonymous) student transcript database;
- subjective probabilty distributions supplied by experts (faculty).

[1] For example, University of Kentucky Department of Computer Science has roughly 60 courses in the books, while the University catalog contains about 3000.

There are two issues for knowledge discovery: confidentiality, and fluidity of courses. The confidentiality concern arises in accessing data from student records. Theoretically, only statistical information is extracted from the transcript database. In fact, information extraction can be made only using *aggregate database queries* to the transcript database. In practice however, certain steps need to be taken in order to ensure that personal student information is not misused[2].

Although it is difficult to officially revise a curriculum, it is the case in many departments that courses evolve[3] over time, and from professor to professor. The latter can be addressed by eliciting probabilities from each individual professor, and by distinguishing in the knowledge discovery phase which professor taught each section of each course considered. Storage of context-dependent probabilistic data is addressed in Section 4 and in [5].

The problem of courses, curricula, and requirements evolving over time is addressed by considering only relatively recent data, and by periodic updates of the dependencies, probabilities, constraints, and goal states of the system.

There are certain aspects in which a Bayesian model of the advising process is simpler than some other applications of Bayesian Networks. For example, due to the structure of our problem, we are not interested in obtaining posterior probabilities (despite the fact that our model is sufficient to provide those to us). Indeed, the input to a Bayesian Advisor will be a student's transcript, and the reasoning should proceed forward, to predict the probabilities of success in the future classes.

Notice again that the modeling process described here is used to build a Bayes net for a very limited scope.

[2] In order to respect student privacy, we have arranged with the University of Kentucky registrar's office to obtain labeled student data without student IDs or demographic data (gender, race, age, etc.)

[3] Kentucky law no longer requires that we say, "change with time" instead of "evolve".

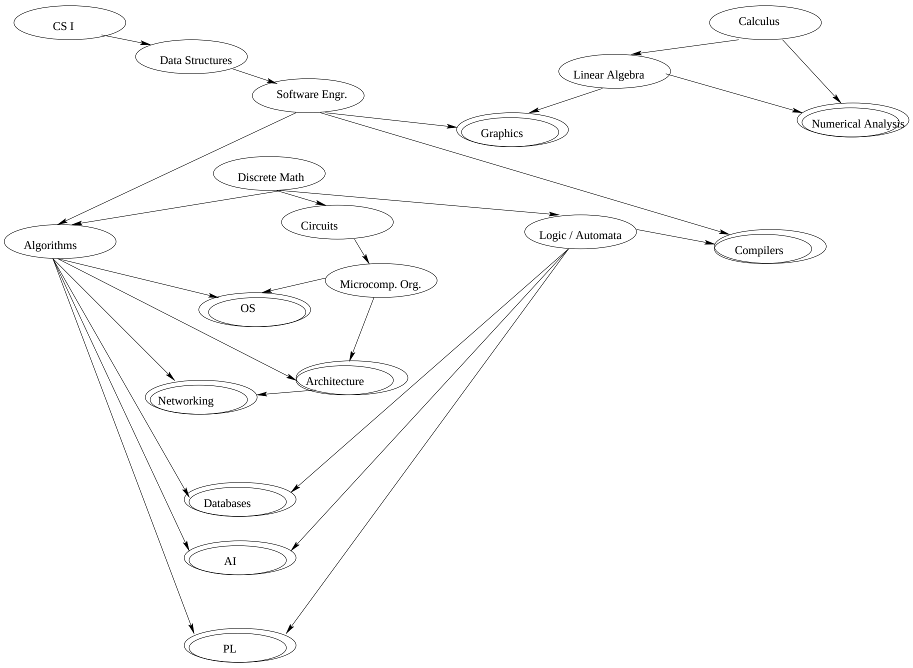


Figure 2: Pilot Advisor Bayes Net model with sample conditional probability distributions

When a full model of the undergraduate curriculum is built, it will *consist of many such smaller units,* each identified by their *context* (student's major, for instance).

# 3 INITIAL IMPLEMENTATION

The pilot model is of a part of the undergraduate Computer Science curriculum for the University of Kentucky. This part represents a simplified three-year sequence of primarily Computer Science classes. Figure 2 depicts the structure of the model. Out of 19 courses captured as nodes in it, three, Calculus, Linear Algebra and Circuit Design are taught by other departments but are required by the Computer Science curriculum. Other required courses featured in the model are Computer Science I, Data Structures, Software Engineering, Algorithms, Discrete Math and Logic/Automata Theory. The remaining courses are the standard junior/senior level Computer Science electives. To keep the pilot model small, we have omitted all graduate level courses available to undergraduates.

To keep the sizes of conditional probability tables small, no course was considered to have more than two parent nodes (an assumption that will be removed in later models). Also, the granularity of the conditional probability tables was coarse: the values were "good" (A/B), "passing" (C), and "failing" (D/E, where E is a failing grade).

Dependencies in our model are based loosely on the prerequisites for the courses. Conditional probabilities were elicited from undergraduate advisors. These probabilities represent subjective opinions and have not been verified.

The Bayes net was implemented in Hugin [2].

## 3.1 EXPERIMENTS

In order to verify the "sanity" of our approach we consider a variety of scenarios. Each scenario consisted of grades for the initial (mandatory) course sequence. We looked at the probability distributions for elective courses in each case. These were compared with each other and with the suggestions elicited from the control group of experts[4]. We *expected* to see the following two phenomena:

- different scenarios induce different probability distributions for elective courses in the pilot model, thus causing different advice;
- expert advice shows similarity with the advice produced by the pilot model in all scenarios.

The scenarios of interest are neither consistently superior nor consistently inferior students; no scenarios include failing required courses. The five scenarios considered were:

[4]The control group of experts was disjoint from the group of advisors who supplied the probabilities.

1. Good programmer, bad at math. The student received A/B in CS I, Data Structures, Software Engineering, Algorithms, Circuits and Microcomputer Organization and C in Discrete Math, Calculus, Logic, Linear Algebra.

2. Bad programmer, good at math. The student received C in CS I, Data Structures, Software Engineering, and A/B in Discrete Math, Calculus, Logic, Linear Algebra.

3. Improving programming skills. This scenario is similar to the previous one, except that the student got an A/B in Software Engineering.

4. Bad first year, improving. The student got C in the first-year courses (CS I, Calculus, Data Structures but improved in the second year getting A/B in Software Engineering, Discrete Math, Logic, and Linear Algebra).

5. Good first year, declining. The grades in the previous example are reversed.

For the second part of the experiment, every member of the control group of experts[5] was presented with five worksheets. Each worksheet contained the description of dependencies (as depicted on Figure 2, with elective courses highlighted) and the description of one of the five scenarios (each known grade printed next to the course node in the network). The general name of the scenario (such as "good programmer, bad in math") was indicated on the worksheet. However, the experts *were not given any information about the actual conditional probability distributions* quantifying the dependencies shown in the pilot model. The experts were asked to list 3-4 elective courses in which they "would expect the student to be most likely to get an A/B".

The measure of the course rank within the control group of experts for each scenario was the frequency that each course was mentioned. Most frequently mentioned courses for each scenario were compared with the ranking generated by the pilot Bayesian Advisor model.

### 3.2 RESULTS

#### Comparing scenarios to each other

In Table 1, we show the order in which our model suggests electives for our five scenarios. The order is based on the probabilities that the students will receive an A or B in each class: the top line indicates a choice with the highest probability for an A/B, for each scenario. Based on these orderings, the advisor or a planning algorithm would make initial suggestions to the student. (Note that, in a realistic planner, other constraints such as time conflicts would have to be considered.) We can see that each scenario has a unique ordering of courses, so would yield a distinct set of recommendations. For instance, consider scenarios 1 and 2: in the first, the student has strong programming skills and weak mathematics; in the second, these are reversed. In the first, the student is steered toward "hardcore" computer science: Architecture, Operating Systems, Networking, while in the second, the student is steered toward courses which require mathematical backgrounds and the ability to abstract concepts. However, simply indicating an improvement in the student's programming skills in their third programming course (scenario 3) opens up other possibilities, in particular Compilers. All probabilities in scenario 3 are at least as large (for their respective courses) as in scenario 2, but some are significantly larger, resulting in a much higher ranking for Compilers. It is clear from the respective orders that the probability of success in Networking grew by the smallest amount.

Similarly, one can contrast scenarios 4 and 5 (the student who began poorly, then improved, and the one who began well, then fell off). The recommendations for scenario 4 are courses that depend heavily on both mathematical and programming skills, and are therefore likely to be most affected by the most recent courses taken (in which the student succeeded).

Other measures can be used to rank courses. Rather than the probability of "success," one can rank them by the probability of not failing the course (i.e., the probability of getting either A, B or C). This is a more appropriate metric for cases such as scenario 5, where all probabilities of success were uniformly low. The orderings under these two metrics are often different, as is clear from Table 3.

#### Comparing pilot model to expert advice

As expected, there was a high correlation between the suggestions of experts and the pilot model's suggestions, as evidenced by Tables 1 and 2. For scenarios 1–4, the correlation was very good, with only one course (Programming Languages) missing from the experts' recommendations in scenarios 2 and 3. In scenarios 2–4, experts seemed to be more eager to recommend Graphics, perhaps because it had no unspecified prerequisites. For scenario 5, there was a significantly lower correlation between the experts advice and the pilot model's ranking of courses based on probabilities of success. However, there was an almost perfect match between the experts' suggestions and the courses that the model predicted the student was most likely to

[5] Total number of experts in the control group was 9.

| Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 | Scenario 5 |
|---|---|---|---|---|
| Architecture | Numerical Analysis | Numerical Analysis | PL | Numerical Analysis |
| OS | PL | PL | Compilers | Networking |
| Networking | AI | Compilers | AI | Architecture |
| Databases | Databases | AI | Databases | Graphics |
| AI | Architecture | Databases | Graphics | AI |
| PL | Networking | Graphics | Architecture | Databases |
| Compilers | Graphics | Architecture | Networking | OS |
| Graphics | OS | OS | OS | PL |
| Numerical Analysis | Compilers | Networking | Numerical Analysis | Compilers |

Table 1: Electives, ranked by pilot Bayesian Advisor model by probability of getting A/B: Scenarios 1–5

| Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 | Scenario 5 |
|---|---|---|---|---|
| OS (8) | AI (9) | Numerical Analysis (7) | Graphics (8) | Numerical Analysis (4) |
| Architecture (5) | Numerical Analysis (8) | Graphics (6) | AI (6) | Compilers (4) |
| Networking (5) | Graphics (6) | AI (5) | PL (3) | Graphics (3) |
| Compilers (5) | Databases (3) | Compilers (4) | Compilers (3) | AI (3) |
| Databases (3) | | Databases (4) | Databases (3) | Databases (3) |
| | | | | Architecture (3) |

Table 2: Expert advice for 5 scenarios (frequencies are reported in partentheses).

| Success | Passing |
|---|---|
| Numerical Analysis | Numerical Analysis |
| Networking | Graphics |
| Arichitecture | Compilers |
| Graphics | AI |
| AI | Databases |
| Databases | PL |
| OS | Architecture |
| PL | Networking |
| Compilers | OS |

Table 3: Electives suggested by pilot Bayesian Advisor Model for scenario 5 using different metrics.

pass. (Compare Table 2 and Table 3.) Given the dismal probabilities of success generated by the model, this was clearly a more appropriate criterion for this scenario.

# 4 THE BAYESIAN ADVISING TOOL

## 4.1 BAYES NET CONSTRUCTION

So far, in our pilot experiment, we have been using conventional methods of Bayes net construction: the initial Bayes net was implemented in Hugin, and the probabilities were hand-elicited from a small set of experts. Future models, however, will require that the construction process be automated, to avoid conflicts. When a student is confronted with a prediction about his or her grades, even if it is probabilistic, the student is likely to modify his/her actions based on that prediction. A professor, confronted with a prediction about her/his grading, is likely to react to that prediction. Thus, the actual construction of the probability tables is best done out of sight of faculty and students.

We did not use actual student data for this initial model; that will be done in later iterations, as anonymous student records become accessible to us. With a student record database, the problem of combining the probabilities elicited from experts and the probabilities extracted from data arises. We need a tool that will do this "conflict resolution" for us automatically.

When we build models based on information obtained from data as well as knowledge elicited from a variety of experts, the extracted and elicited probability distributions will come in a variety of forms, such as single random variable probability distributions, joint probability distributions and conditional probability distributions. This gives rise to the problem of storing and managing this information in a convenient and efficient manner. Using existing probabilistic relational database models [3, 6, 10] for this purpose would be tricky and inefficient. On the other hand, the *semistructured data model* [1] provides the desired level of flexibility in representing the structure of the probability distributions and their properties. It is convenient that semistructured data can be naturally represented and stored in XML format, providing us with the mechanism for data exchange between different tools. We have already developed a representation of such probabilistic information as semistructured data and translated it into XML [5, 8][6]. *Semistructured Probabilistic Objects*, described in [5], will serve as the underlying data model for later versions of the Advisor tool as the means of storing probabilistic information.

The construction of the semistructured probabilistic database from the student records database is a separate task that also needs to be automated. Similarly, the process of eliciting probabilities from a large group of experts (at least one per course, based on recent teaching experiences) would be tedious and time-consuming if done by hand. In the case of the entire undergraduate curriculum of a large state university, this would almost certainly be infeasible. A web-based knowledge elicitation tool is currently under development. The web tools currently available (see [17]) are not designed to elicit a large number of probabilities quickly. For instance, if there are 5 possible grades (A–E, ignoring Pass/Fail and other variations) and three parent nodes, there are $5^3$ probability distributions for that one node. We will be using a variety of techniques, including decision trees, pattern copying, and verbal/numeric scales, for reducing the number of queries and making the elicitation process simpler for the users [15, 16, 13].

### 4.2 PLANNING

Given a Bayes net model of the university program in question, and given a full specification of the goal state (in terms of university and departmental requirements, including number of credit hours), the Advising Tool will have two functionalities: it will be able to compare different plans and rank them according to the probability of success, and it will be able to generate new plans.

The Advising Tool will *not* reveal grade predictions or its internal conditional probability tables to the user. It would be both inappropriate and possibly self-fulfilling to tell a student or teacher what grade the student is likely to receive in an upcoming class. This is the reason that actual probabilities have been suppressed in tables detailing our experiments.

In order to compare plans, the Tool must have a plan evaluator. This may or may not have internal constraint checking to guarantee that the courses planned for future semesters will be or are likely to be offered that semester. The current version does not have that feature. Nor does it check for prerequisites or forbidden pairings such as Physics for Poets and Physics for Majors. These will either be included in later versions, or built directly into the planning algorithms.

Planning algorithms for the Tool will take advantage of a variety of structural constraints, including longest paths through the requirements/prerequisites tree for the student's major department, and possibly information about course availability. This includes schedule constraints such as courses being offered periodically rather than every semester, and information about enrollments. For instance, the University of Kentucky requires a communications course; Public Speaking is the most popular, and tends to close out before sophomores or sometimes juniors are allowed to enroll. Therefore, it should not be recommended to students early in their careers.

Later versions of planning algorithms will take into account the individual's utility function, which may weight GPA higher or lower than a quick degree, and which might put high utility on classes offered at a certain time of day or by certain professors.

## 5 CHALLENGES

The Bayesian Advisor Project raises interesting and challenging questions in every phase. Some, such as the problem of automating probability elicitation, have been discussed in Section 4.

Because courses change over time and according to the instructor, we expect to obtain inconsistent findings. This raises issues of *knowledge fusion versus contextualized data*: what is the tradeoff between keeping each set of found or elicited probabilities, versus fusing them?

Student records are supposed to be confidential, but for our purposes, each student's record requires a unique identifier, so we do not consider (for instance) Brett's transcript after each semester as an independent record. How do we *safely guard the data and conditional probabilities from being output or deduced*? As a first step, the queries to the database of student records will be of two forms: aggregate queries and view construction. However, the views constructed will be strictly internal; the only data output will be statistical. It is difficult to deduce from the statistics the exact performance of any individual.

Typically, a Bayes net structure and probabilities are stored in main memory; under our constructions, probabilities will be stored in a semistructured probabilistic database [5], which supplies the necessary probabilities to the Bayesian Advisor tool on demand. This requires less main memory per application than storing all probabilities in main memory. One of the main

[6]Unlike the work on XMLBIF [4], an XML format for storing Bayes Nets, our work concentrates on a flexible representation of diverse probability tables only.

reasons for this design is that, while multiple Bayes nets must be stored, only one or few need be considered for a given student.

The different Bayes nets constructed will have similar structure to some generic Bayes net for the overall academic program. The variations in structure can be stored as difference sets, although each set of probabilities will have to be stored separately. It is expected, though, that the probability tables will have many similarities. This notion of "overlapping" networks bears certain similarities to the notion of Bayesian multinets [7, 9]. However, in Bayesian multinets, dependencies may vary with changes of values of certain random variables. In our case, the changes in the network depend on variables that are not considered random. This also bears some similiarity to the process of building networks from fragments [11]. The question is, *how can analysis or planning algorithms be constructed to take advantage of these overlapping networks*?

Finally, there will be many challenges in *implementing planning algorithms*. Is it feasible to take into account all of the constraints, including students' preferences in scheduling, courses, and instructors, in the initial plan-generation phase? Note that the University of Kentucky has approximately 3,000 undergraduate courses on the books, including underspecified "topics" courses. There is significant structure possible, given a particular major. Can such structure be utilized to speed up planning?

To the best of our knowledge, this is the first application of Bayes net technology to the problem of academic advising. In these days of increasing demands on faculty, any tools to support such a time-consuming and repetitive task are more than welcome.

**ACKNOWLEDGEMENTS**

We want to thank Rina Dechter for suggesting we pursue this; members of the University of Kentucky Computer Science Department who agreed to participate in the control group for our experiment, and the rest of the Bayesian Advisor group, for teaching us some of the skills we need for this project, and learning the rest with us. Finally, we thank the faculty members from other schools who have encouraged us and asked to be involved in this project, including Eastern Kentucky University; Berea College; University of Dayton; the US Naval Academy, and UC Irvine.

## References

[1] S. Abiteboul, B. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.

[2] S.K. Andersen, K.G. Olesen, F.V. Jensen, and F. Jensen. HUGIN— a shell for building Bayesian belief universes for expert systems. In *Proceedings of 11th IJCAI*, pages 1080–1085, 1989.

[3] D. Barbara, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Trans. on Knowledge and Data Engineering*, 4:487–502, 1992.

[4] Fabio Cozman. XMLBIF: XML BayesNets Interchange Format. http://www.cs.cmu.edu/afs/cs/user/fgcozman/www/Research/InterchangeFormat/.

[5] Alex Dekhtyar, Judy Goldsmith, and Sean Hawkes. Semistructured probabilistic databases. Technical Report 318-01, University of Kentucky, 2001.

[6] D. Dey and S. Sarkar. A probabilistic relational model and algebra. *ACM Transactions on Database Systems*, 21(3):339–369, 1996.

[7] D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and bayesian multinets. *Artificial Intelligence*, 82:45–74, 1996.

[8] Sean Hawkes, Alex Dekhtyar, and Judy Goldsmith. Designing markup languages for probabilistic information. Technical Report 319-01, University of Kentucky, 2001.

[9] David Heckerman. *Probabilistic Similarity Networks*. MIT Press, Cambridge, Massachusetts, 1991.

[10] V.S. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian. Probview: A flexible probabilistic database system. *ACM Transactions on Database Systems*, 22(3):419–469, September 1997.

[11] K. Laskey and S. Mahoney. Network fragments: Representing knowledge for constructing probabilistic models. In *Proceedings of Thirteenth Annual Conference on uncertainty in Artificial Intelligence*. Morgan Kaufman Publishers Inc., 1997.

[12] Kathryn Blackmond Laskey and Suzanne M. Mahoney. Representing and combining partially specified CPTs. In *Proc., 1999 Conf. on Uncertainty in AI*, 1999.

[13] Kathryn Blackmond Laskey and Suzanne M. Mahoney. Network engineering for agile belief networks. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):487–497, 2000.

[14] Judea Pearl. *Probabilistic Reasoning in Intellegent Systems: Networks of Plausable Inference*. Morgan Kaufman, 1988.

[15] Silja Renooij and Cilia Witteman. Talking probabilities: communicating probabilistic information with words and numbers. *International Journal of Approximate Reasoning*, 22(3):195–215, 1999.

[16] Linda C. van der Gaag, Silja Renooif, Cilia Witteman, B.M.P Aleman, and B.G. Taal. How to elicit many probabilities. In *Proceedings of the 15th Conference on Uncertainty in AI (UAI-99)*, pages 647–654, 1999.

[17] Haiquin Wang and Marek J. Druzdzel. User interface tools for navigation in conditional probability tables and elicitation of probabilities in bayesian networks. In *Proc. 16th Conference on Uncertainty in Artificial Intelligence*, 2000.