A decision-tree-based system for student academic advising and planning in information systems programmes

Naoufel Werghi* and Faouzi Kamoun

College of Information Technology
University of Dubai
P.O. Box 14143, Dubai, UAE
E-mail: nwerghi@ud.ac.ae
E-mail: fkamoun@ud.ac.ae
*Corresponding author

Abstract: This paper presents a Decision Support System (DSS) for student advising. The system aims to provide students with an automated programme planning and scheduling service that best fits their profiles while meeting academic requirements. After the literature survey and description of the system’s architecture, the paper describes the new paradigm that models student advising as a search problem, whereby the search space is represented by a decision tree that embeds virtually all the instances of a student academic plan. Our approach has several advantages over previous rule-based advising systems. The system implicitly implements, via the decision tree, many academic rules; it allows a systematic and exhaustive browse of the different student plan instances; and it permits a methodological assessment and measurement of the appropriateness of a given student academic plan. An advanced prototype of the proposed advising system was successfully implemented.

Keywords: information systems; decision-tree; academic advising.

Reference to this paper should be made as follows: Werghi, N. and Kamoun, F. (2010) 'A decision-tree-based system for student academic advising and planning in information systems programmes', Int. J. Business Information Systems, Vol. 5, No. 1, pp.1–18.

Biographical notes: Naoufel Werghi received a PhD in Computer Vision from the University of Strasbourg, France, in 1996 and an Msc in Instrumentation and Control for Computer Vision from the University of Rouen, France, in 1993. He has been a Research Fellow at the Division of Informatics at the University of Edinburgh, UK, a Lecturer at the Department of Computer Sciences at the University of Glasgow, UK and a Visiting Professor at the Department of Computer and Electrical Engineering at the University of Louisville, USA. Currently, Dr. Werghi is an Assistant Professor at the College of Information Technology, University of Dubai (UD), UAE. His research interests are decision support systems and multimedia information processing and retrieval.
1 Introduction

Academic advising is meant to help students develop educational plans that are consistent with their academic career and life goals, and to provide the information and the guidance they need to pursue those goals. This support is based on the establishment of educational collaboration between students and their academic advisors. Academic advisors are the students’ consultants in their educational planning. Advisors assist students by guiding them through the university’s educational requirements, helping them schedule the most suitable modules, introducing them to pertinent resources, promoting leadership and campus involvement, assisting in career development, helping them with the timely completion of their studies and helping them find ways to make their educational experience personally relevant (Pizzolato, 2006).

The idea of a Decision Support System (DSS) for academic advising is an enticing one. The benefits of such an assistance are manifold. Such a system will improve the advising process and help overcome the many problems that can occur, such as the limited number of advisors in contrast to the large number of students, the fact that advisors are not available all the time and the fact that some advisors are new to the university and do not have enough advising knowledge/experience. A DSS will also simplify the tasks of faculty, staff, students and professional advisors, help save time and effort, prevent human-induced errors and help the university introduce new technologies (Beaudin and Breiner, 2001; Golbuski, 2008; Wilson, 2004). At the same time, being related to scheduling and sourcing, student advising and planning can borrow or inspire DSS solutions that address such problems (Foulds and Zhao, 2007; Dominic et al., 2008).

A DSS is capable of taking, as input, information about a student, including academic programme and course history, and making a reasonable recommendation of courses or a sequence of courses for the next term or terms, respectively. Such a system will enable the advisor to save time and concentrate on the more substantive advising issues such as the choice of electives.

Academic Advising (AA) is a good domain and application to test the adequacy of DSS paradigms and approaches (Hamdi, 2007). AA also provides a complex and dynamic environment and constitutes a challenging experimental test-bed for
investigating DSS issues. In addition, a DSS will allow better evaluation of the advising process and facilitate the implementation of any eventual improvement actions (Hester, 2008).

In this paper, we propose a novel paradigm for intelligent academic advising. The innovative aspects of this work are twofold:

1. the deployment of a decision tree based on which the system can determine systematically and exhaustively the sequence of courses a student can take during each term or over a multiyear planning horizon

2. the implementation of mechanisms for assessing and measuring the appropriateness of a given plan according to certain academic criteria.

2 Literature review

The deployment of DSSs in academia was pioneered by Cox and Jesse (1981). They applied and used backward scheduling logic of material requirements for planning class scheduling. The system can determine the modules that can be offered in each term across a multiyear plan. This system brought computerised solutions to the prerequisite rules, but their study was macro in nature. Dinkel et al. (1989) developed an expert system for course time and venue scheduling in a specific term. Wehrs (1992) developed an expert system that could assist the advisor in evaluating student records and propose course schedules.

While the above systems had the merit of promoting the idea of exploiting DSSs in course planning and scheduling, they have been criticised for not being geared towards the student. Other studies attempted to address this issue to make the DSS more student-centric. Murray and Le Blanc (1995) proposed a backward scheduling approach, which is similar to the one proposed by Dinkel et al. (1989), but which was supported by the hierarchical rules paradigm of Kosaka and Hiroushi (1982). The system processes the student records in the database and gets input from the student. Valtorta et al. (1984) proposed a Prolog implementation that deploys an inference engine (modelled after MYCIN) for a rule-based expert system dedicated to graduate students in computer science. Golumbic et al. (1986) suggested an expert system that encompasses student information, a departmental knowledge base and a planning module. A similar system was developed by Frank et al. (1988) whereby a relational database model was adopted for rules and student records. The system can suggest the number of course-hours a student can take and the sequence in order to satisfy college degree requirements. Cutright et al. (1991) developed an expert system that generates a list of courses that is consistent with prerequisites, and which starts from an initial assertion list. Gunadhi et al. (1995) proposed an expert system that embeds object-oriented representation with knowledge-based paradigms. The advent of web technology incited the development of web-based expert systems such as those proposed by Htay et al. (2006) and Grupe (2002). In the same vein, Pokrajac and Rasamny (2006) proposed the ‘Virtual Expert System for Advising’, which was mainly geared towards avoiding schedule conflicts. This approach has also been investigated by McDonald and Prosser (2002), who applied a constrained programming framework to implement a student advisory system.
The use of rule-based expert systems was a concrete step towards intelligent advising and planning; yet it was not sufficient to reach this objective for the following reasons:

- the sheer amount of knowledge required, the complexity of the advising task and the diversity of factors that intervene in academic advising all make the formulation of sequential rules that can span all these aspects an extremely difficult task
- the dynamic nature of academic programme requirements would turn the updating and maintenance of such systems into a crippling task
- the intrinsic disadvantages of rule-based expert systems, namely the opaque relationship between rules, ineffective search strategy and inability to learn.

Motivated by these concerns, researchers have tried other strategies. Sandvig and Burke (2005) proposed a case-based reasoning paradigm. The approach is based on the assumption that similar students will have similar course histories. The system uses the experience and history of graduate students as a template to propose potential appropriate courses for the current students. However, to be successful, the approach requires matching between students’ histories to find similar cases. Unfortunately, some aspects of this process (for instance, measuring course similarities) are pretty problematic, as the authors recognised. Hamdi (2006) placed the student advising problem in the context of the Information Customisation Framework, by suggesting an elegant architecture of a multi-agent system; however no specific recommendations were given regarding the decision techniques. In the same vein, Marivate et al. (2008) developed a multi-agent system for course recommendation whereby the course’s source is the whole web. However, the scope of the work is rather oriented towards specific training courses recommendation rather than academic planning. Juang et al. (2007) developed a similar framework based on genetic algorithm optimisation tools.

Other approaches aimed to reduce the scope of the advising task to a specific term. For instance, Farzan and Brusilovsky (2006) developed an interactive hypermedia navigation system that employs a social navigation approach (Dieberger et al., 2000) to tackle the problem of information overload. The system delivers recommendations for courses based on students’ assessments of their particular career goals. Bendakir and Aimeur (2006) developed a system that combines association rules with user-preference input to recommend relevant courses. O’Mahony and Smith (2007) proposed a method for proposing electives based on simple statistics of the core courses taken by the students. They also used text-based retrieval approaches for proposing similar modules. However, for large-scale curricula, this procedure should be performed offline in order to keep the advising within a reasonable interactive time. Ho and Lu (2005) proposed web-based expert system called class schedule planner. While their approach adopts the standard rules-based system, it has the merit of allowing dynamic knowledge management.

It is also worth mentioning other studies that were driven by the belief that the key to good advising is the advisor’s competence in accurately assessing the student’s abilities. These studies investigated the forecasting of student performance based on course history. Samples of such studies are Dekhtyar et al. (2001), Deniz and Erzan (2002) and Isa et al. (2007). There have also been some contributions that tackled specific aspects of the advising process, such as Biletskiya et al. (2008), whereby the authors developed an expert system to automate the process of transferring course credits between academic institutions.
3 The advising domain

The College of Information Technology in the University of Dubai, UAE, offers a BSc degree in Computing Information Systems. The programme is modular and courses (modules) are offered during two regular (15-week) terms each year. The programme encompasses 53 modules segmented into six categories. The student needs to complete 43 modules according to the breakdown in Table 1.

**Table 1** Breakdown of the Bsc Information Systems Modules

<table>
<thead>
<tr>
<th>Category and number of available modules</th>
<th>Number of modules required</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Education (9 modules)</td>
<td>7</td>
</tr>
<tr>
<td>Humanities and Social Science (9 modules)</td>
<td>2</td>
</tr>
<tr>
<td>Natural and Applied Science (6 modules)</td>
<td>2</td>
</tr>
<tr>
<td>Business (13 modules)</td>
<td>13</td>
</tr>
<tr>
<td>IT Core (12 modules)</td>
<td>12</td>
</tr>
<tr>
<td>Specialisation (7 modules)</td>
<td>7</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>43</strong></td>
</tr>
</tbody>
</table>

The above course distribution reflects the vision of the college to give students a broad education while focusing on the information systems area. Courses are also categorised into four grades or levels; for example, the first grade corresponds to the foundation courses, whereas the fourth level corresponds to the most advanced ones.

Each course is labelled by code and catalogue number. The code specifies the category to which the course belongs. For example, the core and the specialisation courses have the codes ITGN (for Information Technology General) and ITIS (for Information Technology Information Systems) respectively. The ITGN/ITIS courses are offered within the College of Information Technology. Courses which are coded differently are offered by other colleges or departments. The course catalogue number is composed of four digits, the fourth of which indicates the course level.

The programme has a modular structure that offers great flexibility and allows for a better usage of resources, yet it does involve a number of rules and constraints (prerequisite rules in particular) that tend to break the programme into phases. Figure 1 depicts a flowchart that illustrates the relation of the course prerequisites in the model.

Students can take from two to six courses per term. In selecting courses, they must know about the different possible combinations, taking into account the list of offered modules, while respecting academic rules and recommendations. Academic rules cover the prerequisite requirements and the maximum number of modules a student can take (depending on whether the student is full-time or part-time and the cumulative GPA). These rules should be strictly obeyed, as per the university policies.
In addition to strict rules and restrictions, academic recommendations are soft by nature and are seen as good practices in selecting courses. They are not compulsory, but the student is strongly advised to follow them. The two main recommendations are as follows:

1. For core and specialisation modules, it is recommended that students complete all the modules within a given level before enrolling in a higher-level course. For instance, if the student can choose between a third-level course and a fourth-level course, priority should be given to the third-level course.

2. For courses within the same level, priority should be given to the course having the lowest catalogue number.

From the above description, it is clear that the choice of a suitable academic plan is not a trivial task for the student. A systematic and methodological selection of the optimal plan manually would be a tedious and time-consuming task for both the student and the advisor. In fact, there is a need to derive all the possible alternate programme plans that concur with the academic regulations. Then, the academic plan instances must be browsed and evaluated using well-defined criteria and eventually compared, in order to select the plan that would best suit the student. The risk of error is considerable in such manual processes and this can emanate from many factors, e.g., the lack of objectivity (from the student and the advisor), unfamiliarity with academic regulations and time...
constraint pressures. It is unlikely that a manual process would generate the optimal plan. For this reason we thought of the adoption of a DSS approach that has the potential to bring both effectiveness and efficiency to the advising and planning task.

4 The decision support system

The DSS consists of three standard components: the database, the modelling and the user interface components. The database is a repository of student data. From this database one can extract a student profile, which encompasses all the information needed for advising. This information spans the list of courses that the student has completed, his/her performance in each of these courses and the cumulative GPA. The student profile can also capture other information about the student, such as status (part-time or full-time), work experience, programming languages mastered, spoken languages and special needs/interests.

The modelling part is the reasoning and the computing part of the system. It embeds the task-specific knowledge, its representation and the human expertise required for the advising process. It is the inference engine in charge of treating the spectrum of information involved in advising, in order to produce the optimal choice(s). Depending on the degree of autonomy of the system, its observation and adaptation abilities, a modelling subsystem (for instance, an observing agent) can be dedicated to the construction of the student profile. Its task consists of observing and monitoring the student’s progress in his/her studies and incrementally building and updating the student’s profile.

The user interface provides mechanisms and tools for the user to communicate and dialogue with the system.

The proposed DSS architecture is depicted in Figure 2.

Figure 2 DSS architecture and information flow (see online version for colours)
The administrative database is a relational database that stores the statistical information about students, course information, schedules and faculty. The student database stores the customised profiles of students. It contains the data needed for the modelling unit. The student profile essentially captures the student’s academic history and his/her past records; it can also include other information such as educational focus, work experience, special interests and skills. A human administrator is in charge of building and updating the student profile. This task can also be delegated to a monitoring module, acting as mediator between the administrative database and the student database. This module will observe the students’ progress and update their profile accordingly.

The modelling unit composed of the decision tree and the constraint filter (see Figure 2) is the heart of the system. This unit acts as a virtual advisor and embeds the processes associated with the advisor’s tasks, such as searching for different alternatives and options for course planning, and selecting those courses which are compliant with academic regulations while providing the best match to the student’s profile.

The motivations behind the adoption of a decision tree approach and how it will simplify the implementation of the advising tasks are elaborated in the next section.

5 The decision tree approach

Academic planning and advising can be conceptually perceived as a search for the best choice among many alternatives. In fact, to fulfil a programme’s requirements, there is a large variety of course sequences a student can follow. The student’s aim is to find the best sequence of courses that meets some criteria across a single term or a multiterm horizon plan. Computerising this search task requires a computing framework that allows exploring all the plans systematically and exhaustively and embedding the criteria and the conditions (or at least part of them) that would lead to the best course selection.

Search techniques (Russel and Norvig, 2001) can meet these requirements to a large extent. In general, search techniques are used to find a sequence of physical or abstract actions that will lead from an initial state to a target state. For example, a freshman student starts from an initial state where no courses have been taken yet. The final state can be, for example, a specific group of courses he/she will enrol in for the first term, or the last group of courses he/she will take before graduation. Here, the sequence of actions is the sequence of enrolments to be made over a single or multiterm plan. Search techniques operate in the search space, which is defined as the set of all possible states. The tree structure is an elegant framework for representing the search space and the sequence of actions. Basically a tree is a hierarchical structure composed of nodes and edges. The tree starts from a root node, from which emerges other nodes (referred to as children) which are connected (linked) to their parent node via edges. From each child node emerges other child nodes and so on. The nodes and edges represent the states and the actions respectively. A path in the tree is defined as the sequence of linked nodes.

As illustrated in Figure 3, a node \( n \) in the tree is associated with a group of modules and the children of that node are the groups of modules that the student can take after successfully passing all the courses in node \( n \). The root node represents the initial state, which corresponds to the case where no modules have been taken yet. The tree is organised hierarchically, where each level corresponds to a term. The distribution of the courses across the tree levels, depicted in Figure 3, depends on the prerequisite rules. For example, node 9, which contains the modules [ENGL 105, GMAT 105, ITGN 120, ITGN
A decision-tree-based system for student academic advising and planning

215, GUAS100], cannot be part of the first tree level, as three modules in this group (ENGL 105, GMAT 105, ITGN 120) need the prerequisites ENGL 100, GMAT 100 and ITGN 115 respectively.

**Figure 3** Sample of a distribution of course groups in the tree structure (see online version for colours)

According to university regulations, the number of courses that can be taken per term must not exceed six and should not be less than two. However, we allowed a node to hold a number of courses that ranged from one to six. This allows the tree to span a large variety of student states (e.g., the student is constrained to take a single course in one term for some special reasons, or the student fails two courses out of three). For example, consider the case of a student who enrolled in the group of courses [ENGL 100, CISL 100, GMAT 100] in a first term (node 2 in Figure 3). Let us suppose that he failed GMAT 100. For the next registration, this student’s state will correspond to node 3 ([ENGL 100, CISL 100]).

The tree captures virtually all the possible instances of a student academic plan. A full academic plan covers a sequence of course groups from the first set of enrolled courses up to the last group of courses taken before graduation. A full academic plan is associated with a full path, i.e., a path in the tree that starts at the first level of the tree and ends at a leave node (a node that has no children). On the other hand, a partial academic plan covers the sequence of courses starting from any group of courses taken in a given term (excluding the first one) until the last group of courses that the student enrolled in before graduation. A partial plan is associated with a partial path that starts at a node not located on the first level of the tree and terminates at a leave node.

We also note that the length of a path, defined by its number of edges, does not necessarily reflect the number of terms. This can happen, for instance, when a student fails a course or decides to retake an already-passed course. For example, if we consider the failure case mentioned previously, the student’s state after failing the GMAT 100 course is just at node 2, which corresponds to the path [0, 2] of length 1. However, the student has actually spent two terms. For re-enrolment, consider the case where a student, having passed the group of courses [ENGL 100, CISL 100] (node 3 in Figure 2), decides to take a new course GMAT 100 and to enrol again in the ENGL 100 and CISL 100
courses. After passing this group of courses, the next state will be at node 2 of Figure 3; thus the current path, which is [0, 2], is of length 1, yet this student was actually enrolled in two terms.

It should be noted that the path length can be made equivalent to the number of terms by allowing pairs of parent-child nodes to share similar courses. However, this will dramatically increase the number of nodes in the tree. Therefore, we decided not to embed the failure and the re-enrolment cases in the tree structure, but we record them when they do occur in the student profile.

The decision tree implicitly implements the prerequisite rules via the parent-children link. The prerequisite rules restrict the number of children that a parent node can have. A path embeds the perquisite rules and the number of terms (i.e., the number of edges in the path if we ignore failure and re-enrolment) required to achieve a partial or a full academic plan. Further, the more prerequisite rules, the fewer the number of paths in the tree; and thus the fewer number of academic plan options. Using tree search browsing techniques, it is possible not only to explore the different student academic plans systematically and exhaustively, but also to assess their appropriateness.

We will have better insight on how a sequence of course groups is organised in the tree as well as how to conduct a search for student academic plans using the simple hypothetical case discussed below. It corresponds to a student who still has to enrol in three courses before graduation. These courses are ITIS 302, ITIS 411 and ITIS 440, and they are not related by any prerequisite rule.

The subtree related to this case is depicted in Figure 4. For simplicity, these courses are referred to by the letters a, b and c. The subtree starts at the current state node, and there is no need to know the courses associated with it. Its child nodes correspond to the different course combinations that can be taken in the next term.

Figure 4 Sub-tree corresponding to a student case having three courses left before graduation (see online version for colours)

The child nodes are arranged from left to right in descending order with respect to the number of courses that a node carries. As illustrated above, the first child node of the current state node is the one with the maximum number of courses (three). Then come the child nodes with two courses, and finally those with a single course. This arrangement
implies that the short-time academic plans tend to be on the left side of the tree (as Figure 4 clearly shows). This enables us to speed up the search by focusing on the left area of the tree, given that the student’s priority is to achieve graduation with a minimum number of terms.

The arrangement described above permits us to tailor the search to the type of user query as well as to establish shortcuts to the most popular queries. For example, if the user just wants to know the different alternatives for a next term enrolment, the system will simply retrieve the child nodes of the current states. To know all the possibilities of course sequences across m next terms, the system can perform breadth-first or a depth-first search up the next m levels in the tree and retrieve all the paths found. To retrieve all the academic plan instances up to graduation, the system can perform a depth-first search and retrieve all the partial paths. Table 2 depicts a list of the three most popular queries and the corresponding shortcut search actions.

<table>
<thead>
<tr>
<th>Query</th>
<th>Shortcut action</th>
</tr>
</thead>
<tbody>
<tr>
<td>The largest group of courses that can be taken next term</td>
<td>Retrieve the first child node of the current state</td>
</tr>
<tr>
<td>All the possible groups of courses that can be taken next term</td>
<td>Retrieve all the child nodes of the current state</td>
</tr>
<tr>
<td>The shortest academic plan for graduation</td>
<td>Retrieve the first partial path starting at the current state</td>
</tr>
</tbody>
</table>

5.1 Searching for the optimal student academic plan

In the proposed decision-tree-based system, determining the best choice for a student academic plan can be formulated as searching in the tree for the path that best meets a set of requirements. At this stage we propose an approach that can handle three requirements, namely:

1. the prerequisite rules
2. the minimum time (the least number of terms)
3. the academic recommendations.

Requirement 1 is already embedded in the decision tree, as mentioned earlier. For requirements 2 and 3, we propose to associate a cost function to each path in the decision tree. The optimal path is the one that minimises the cost function while meeting the requirements.

The cost function is defined as follows:

Let a path $P$ be composed of $k$ nodes ($n_1, n_2, \ldots, n_k$). The cost function associated with this path is:

$$C(P) = \text{Length}(P) + F(P) + G(P)$$

where $\text{Length}(P)$ is the length of the path, given simply by the number of edges in the path (approximately equal to the number of terms if we ignore failure and re-enrolment).
\( F(P) \) and \( G(P) \) are penalty functions that penalise the violation of the two academic recommendations. Recall that the academic recommendations are as follows:

1. For core and specialisation modules, it is recommended to terminate all the modules within a given level before enrolling in a higher-level module.
2. For courses within the same level, priority should be given to the course having the lowest catalogue number.

The penalty function associated with academic recommendation 1 is defined as follows:

\[
F(P) = \sum_{i=2}^{n} f_i, \quad f_i = \sum_{k=1}^{n_{ci}} \alpha(c_k)
\]

where:

- \( n_{ci} \) = the number of courses in the node \( n_i \)
- \( c_k, k = 1..n_{ci}; \) a course in node \( n_i \)
- \( \alpha(c_k) \) = binary function defined as follows
  
  \[
  \alpha(c_k) = 1, \text{ if there is a course } x \text{ in } n_{ci-1} \text{ such that level}(c_k) > \text{level}(x)
  \]
  
  = 0 otherwise.

The function \( \alpha(c_k) \) penalises each module taken before another module with a greater level. For example, let us consider path 4 in Figure 4, which is composed of the nodes \((n_1, n_2, n_3)\):

\[
F(P) = f_2, \quad f_2 = \sum_{k=1}^{\alpha(\text{ITIS} \ 302)} \alpha(\text{ITIS}411) + \alpha(\text{ITIS}440)
\]

Because \( \text{level}(\text{ITIS}411) > \text{level}(\text{ITIS}302) \) and \( \text{level}(\text{ITIS}440) > \text{level}(\text{ITIS}302) \). Thus \( F(P) = 2 \).

The penalty function associated with academic recommendation 2 is defined as follows:

\[
G(P) = \sum_{i=2}^{n} g_i, \quad g_i = \sum_{k=1}^{\beta(c_k)}
\]

where \( \beta(c_k) \) is defined as follows:

\[
\beta(c_k) = 0.5, \text{ if there is a course } x \text{ in } n_{ci-1} \text{ such that level}(c_k) = \text{level}(x) \text{ and number}(c_k) > \text{number}(x)
\]

= 0 otherwise.

The function \( \beta(c_k) \) penalises each module taken before another one which is in the same level but having a greater catalogue number. We note here that the function \( \beta \) has been assigned a penalty weight which is half of the function \( \alpha \)'s weight. This is because the violation of academic recommendation 2 is considered less severe than that of academic recommendation 1.
For example, let us compute $G(p)$ for path 3 in Figure 4, which is composed of the nodes $(n_1, n_2, n_3)$:

$n_1$: the current state node \hspace{1em} n_2 = [ITIS 302, ITIS440], \hspace{1em} n_3 = [ITIS411]

$$G(p) = g_2, \hspace{1em} g_2 = \sum_{i=1}^{k} \beta(c_i) = \beta(c_1) + \beta(c_2) = \beta(\text{ITIS 302}) + \beta(\text{ITIS 440})$$

$$\beta(\text{ITIS 302}) = 0, \hspace{1em} \beta(\text{ITIS 440}) = 0.5.$$  

Because $\text{level}(\text{ITIS 440}) > \text{level}(\text{ITIS 411})$, thus $G(p) = 0.5$

Table 3 illustrates the breakdown of the cost function for each path in Figure 4. We notice that the cost function ranges from 1 to 4.5. These two values correspond to path 1 and path 13 respectively. Path 1 is the shortest path that satisfies all the academic recommendations. At the other extreme, path 13 has a maximum length and it violates the two academic recommendations.

<table>
<thead>
<tr>
<th>Path</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L(p)$</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$F(p)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$G(p)$</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>$C(p)$</td>
<td>1</td>
<td>2</td>
<td>2.5</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3.5</td>
<td>3</td>
<td>4</td>
<td>3.5</td>
<td>4</td>
<td>4.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

5.2 The constraint filter

The system retrieves, via the decision tree, all ranked potential academic plans that can be followed. This raw data goes through the constraint filter module, which will eliminate the plan instances that cannot be realised. This task is performed according to a constraint satisfaction process. There are two types of constraints in the system: domain-driven constraints and student-driven constraints. Domain-driven constraints may include constraints derived from the schedule of classes for the current term (e.g., list of offered courses, the group of courses offered at the same time slot) and place availability (depending on whether or not the enrolment ceiling has been reached). Student-driven constraints are derived from two sources:

1. the student profile, which includes the maximum number of courses a student can take
2. the student input, which includes the number of courses the student wants to take per semester.

6 Implementation

The administrative database and the student database are currently operational. Both databases are built within web applications employing JavaServer Pages and JavaServer Face technology.
The implementation should ensure the following system qualities:

- user-friendly and easy to use
- reasonable response time
- easy to maintain and to update.

A prototype model that successfully implements a partial decision tree, the constraint filter and the user interface has been developed for testing and validation. The decision tree was implemented using a modular object-oriented paradigm that supports hierarchical knowledge representation with inheritance of object properties. This paradigm also facilitates system scalability. At this prototyping stage, the decision tree is constructed manually; however, for a large tree size, an automatic construction would be more efficient. Many methods can be employed for this purpose as described in Murthy (1998), and Tsutomu and Testuro (2003). To speed up the computation of the path’s cost function, we precalculated the cost function at each node in the tree and appended the result to the node’s data. Likewise, the cost function of a given path (i.e., academic plan) is determined by retrieving the cost function estimate as the last node of the path.

The student user interface contains five search queries which allow the exploration of a variety of options regarding short-term and long-term academic plans. The queries are as follows:

1. the largest group of courses the student can take next term
2. all the groups of courses the student can take next term
3. the shortest academic plan up to graduation
4. all the academic plans up to the next semesters (number specified by the user)
5. all the academic plans up to graduation.

For queries 2 to 5, the student can specify the maximum number of courses that he/she can take per semester.

Figure 5 depicts the system output to query 2, related to the student case of Figure 4, whereby the seven possible groups of courses are displayed. Figure 6 shows the output corresponding to query 5. Here the different academic plans are displayed in order, according to their respective cost function values.

At the current version, the planning system presents some limitations. The intrinsic structure of a tree does not allow embedding academic plans in which the student can re-enrol in an already-passed module. There are two possible methods for integrating such scenarios. The first is by adding other nodes that embed re-enrolment cases to the tree. However, this will prohibitively increase the size of the tree. This also raises the issue of setting the appropriate size of the tree, as the number of times a student wants to re-enrol in a given module cannot be known in advance. The second method consists of adopting a graph structure rather than a tree for representing academic plans. A graph allows a trivial representation of academic plans, showing replicated courses (caused either by failure or desire to retake). However, information on the term progress would be less visible and harder to derive than with the tree implementation. In addition, the system does not consider the current student performance when estimating the plan cost.
functions. This limitation can be addressed in future by implementing local rules that involve the student’s performance history with the academic plan’s modules. These further enhancements are left for future research.

Figure 5  Sample output of query 2 (see online version for colours)

Figure 6  Sample output for query 5 (see online version for colours)
7 Conclusion

The paper proposes a DSS for student advising and planning. Academic planning is a knowledge-intensive process for determining the optimal course progression that meets a set of requirements. Contrary to previous contributions, which were intensively based on rule-based approaches, we formulated the student planning and advising tasks as a search problem. The innovative contribution of our decision-tree-based approach lies in several enticing aspects:

- an effective embedding of the prerequisite rules
- a systematic and exhaustive search of the academic plans over a time scale that ranges from the next term up to graduation
- a methodological assessment of the appropriateness of the academic plans.

These features would have been extremely difficult to implement in a rule-based system.

There are many directions in which this work can be further explored. First, we plan to integrate more decision variables in the path’s cost function, including the student’s past performance. Second, with the decision-tree-based system, it will be easier to compare student profiles, in particular with regard to the course history. In fact, the path associated with the students’ academic plan can be used to derive a metric that measures the similarity of the students’ course history. This will be quite useful for mining student profiles and analysing and predicting student performance. These are left for future research.

References


N. Werghi and F. Kamoun


**Note**

1 http://java.sun.com/products/jsp/