Tarot: A Course Advising System for the Future*

Joshua Eckroth and Ryan Anderson
Math & Computer Science Department
Stetson University
DeLand, Florida 32723
{jeckroth,randerson1}@stetson.edu

Abstract

Course advising plays an important role in a student's college experience. Uninformed decisions about which courses to take during which semesters can cause a student to fail to graduate on-time or struggle with a major not ideal for the student. Likewise, faculty and administration routinely ask questions about when courses should be offered, how many teachers are needed to cover the courses, and how changes to prerequisites or major requirements impact schedules. However, course advising is challenging because it can demand complex planning years into the future while considering numerous rules about major requirements, course prerequisites, maximum course loads, and course offering schedules. We have developed Tarot, a course advising system that uses a planning engine to develop multi-year course schedules for complex scenarios such as double majors, study-abroad semesters, course overrides, early graduation, and transfer credits. This paper describes Tarot's design and operation and distinguishes its capabilities from existing course advising tools.

1 Introduction

Course advising plays an important role in ensuring a student's success in their college career. This is particularly true for students attending private,

often costly colleges and students on academic or athletic scholarships. In these cases, time-to-graduation is a hard constraint on their ability to afford their education. Likewise, colleges and universities are ranked according to 4-year graduation rates, further emphasizing the importance of ensuring students finish their degrees on-time. Without a strong course advising infrastructure, including software tools and well-trained advisors, student success and the institution's ranking may suffer.

We have developed a new software tool called TAROT to assist advisors and students. TAROT is designed to help with the complex constraints and rules inherent in planning multi-year course schedules. Although many academic departments design prototypical two- or four-year schedules to help guide entering students, not all students follow the same path or come from the same background. Once a student deviates from this pre-defined plan, e.g., they bring transfer credit, or add a second major, or study abroad, or need a course override, or must finish in 3.5 years, etc., then the pre-defined plan is useless. Now, the student and advisor must think through the intricacies of major requirements, course prerequisites, and offering times to find a plan for this student's particular circumstances. This cognitive burden introduces the possibility of mistakes and precludes any more advanced forecasting such as finding the best time to study abroad or finding every possible major elective that would satisfy graduation requirements.

Yet handling constraints and managing complex interactions is the raison d'être of planning engines from the field of artificial intelligence. TAROT is a planning engine designed specifically for course advising. Its use cases focus on developing a student's schedule across multiple semesters rather than scheduling courses for times/days in the week, rooms, etc. We implemented TAROT in Prolog and exploit this language's ability to perform backtracking search to find solutions that satisfy arbitrary constraints. TAROT is also designed for efficiency, and time-to-answer is reported for each use case described below. At this time, TAROT handles just a few majors (math, computer science, computer information systems, and cybersecurity), and while we have plans to extend to more majors, we do not plan to include every available university course. General education requirements, for example, are marked as "gen. ed." slots in a schedule since the courses are mostly interchangeable and available in any semester.

In order to best explain how TAROT differs from other course advising systems, we define four levels of sophistication in terms of the kinds of questions students and/or advisors may wish to answer.

Level 0: Questions about the present. At the most basic level, students and advisors wish to know: What courses has this student taken? What is the student's current GPA, and how many credits have they earned?

Level 1: Questions about a possible future. This level introduces forecasting, i.e., establishing a schedule for the student's future. Relevant queries include: What courses should the student take, and in which semesters, in order to graduate on time? What is a sequence of courses that allows the student to achieve a double major? How does the schedule change if the student wishes not to take more than two major courses per semester? How does the schedule change if the student studies abroad during a certain semester or takes a semester off? (Note, we assume students are unable to transfer their study abroad credits, though of course this is not always the case in practice.)

Level 2: Questions about all possible futures. This next level considers more than one possible future, or often, all possible futures. For example: When should this student study abroad to ensure on-time graduation? Which semester should a particular course be taken to remain maximally flexible? What grade must the student earn in a certain class to ensure a desired GPA?

Level 3: Questions about the rules. Finally, staff and faculty may also ask about the rules themselves. For example: How different are two majors in terms of shared and distinct courses? How many teachers are required to cover the courses that students may take to complete their degrees?

In the remainder of this paper, we will show how TAROT is able to answer questions from all four levels. We begin by discussing related works and how they fit in terms of these levels. We then describe TAROT's internal implementation and showcase how TAROT handles a variety of questions.

2 Related Work

Several systems have been designed to help with academic advising. For simple advising, i.e., our Level 0, the use cases involve the present and the near future. For example, a student comes to an advisor after one or two semesters and wants to know his or her eligibility for some specific classes in the upcoming semester. Engin et al. [4] created a system that attempts to answer those questions by comparing courses to a set of rules created using *Oracle Policy Automation* (OPA). The student's level (freshman, sophomore, etc.) can be determined using credits earned, or the student could check his or her eligibility for a course in terms of prerequisite courses. Another system from Vincenti and Bennett [7] is designed for the student to use on his or her own ("self-advising"). Once the student selects courses for the current semester, a list is generated showing which courses will be available and relevant in the following semester. Guerin and Goldsmith [2] created an academic advising system that uses a dynamic Bayes net. This system suggests courses based on the academic background of the student. For example, students are directed to different sequences of Computer Science courses depending on whether they were stronger

in mathematics but weaker in programming, or vice versa. Dodson et al. [3] use Markov decision processes to likewise recommend next-semester courses based on the student's history. Systems like these are effective when the student and/or advisor have a clear understanding of the student's desires in the near term, but are not capable of developing customized concrete plans multiple semesters into the future.

The systems described above operate mostly on Level 0 since they do not generate full schedules. Now we consider systems that operate at Level 1. The IDiSC+ system by Mohamed [6] uses a sophisticated algorithm to consider several factors while forecasting semesters. This system, like those in Level 0, considers prerequisites and availability, but the IDiSC+ system can also work with constraints such as ensuring minimum credit hours per semester, number of courses per semester, required electives, minimum GPA, course priority, and even the student's budget. A system with this type of complexity is effective for planning the archetypal undergraduate education, but falters when even more complex factors – double majors, when to study abroad, or if a semester will be missed – are introduced.

To our knowledge, no systems are available that operate at Level 2 or 3, i.e., finding all possible schedules and making inferences on this set, and asking questions about the major requirements and multitude of course schedules themselves.

These prior efforts demonstrate a reasonable ecosystem for assisting students and advisors in general advising scenarios. Students have access to databases and planning engines that allow them to lay out an appropriate schedule, perhaps for their entire undergraduate career. Unfortunately, more complexity exists in real-world academic advising. This added complexity is where Tarot is most unique. In particular, Tarot can identify all possible scenarios meeting user-provided constraints, rather than just the most likely. Consider an example where a first-year student is certain that she will study abroad for one semester. She asks her advisor which semester taken abroad would delay her graduation the least, assuming she is unable to transfer any credits (the safest assumption for the hypothetical scenario). An advisor would normally have to spend a substantial amount of time working this issue out, taking into consideration long term questions like course frequency/availability, GPA, and double majors; and this would have to be repeated for each potential semester abroad. For Tarot, this kind of search can easily be constructed by specifying just a few constraints. Tarot is designed to work with arbitrary constraints, some of which may introduce exhaustive search to find all possible solutions to a question.

In the following sections, we comment on Tarot's implementation and then evaluate its capabilities and efficiency across all Levels 0 through 3.

3 Implementation

TAROT is implemented in Prolog, whose backtracking search paradigm is an ideal fit for TAROT's use cases. Course information, including major requirements, prerequisites, and course offering times (Fall/Spring; odd/even years) are stated as Prolog facts or, in some cases, more complex rules that check for specific conditions. For example, our computer science major's required senior research course has complex prerequisites: students must have completed three 300-level courses, two of which must be computer science (CSCI) courses, one of which may be a CSCI or a computer information systems (CINF) course.

For most advising questions, TAROT internally fills out a multi-year schedule. In some cases, TAROT also proceeds to find all possible ways of building the schedule in order to answer questions from Levels 2 and 3. The schedules are built by establishing a blank schedule full of "free slots" and then recursively replacing free slots with courses. Prolog's backtracking allows us to specify arbitrary constraints on the schedules, such as maximum number of major courses to take each semester, without having to define code that specifically handles these constraints. Any schedules that fail to meet the constraints will be thrown out and the search for satisfying schedules will continue. As a case in point, finding a schedule for a student who wishes to double major is as simple as finding a schedule for the first major, then starting with this schedule (and its remaining free slots) and filling in the requirements for the second major using the same code path that filled in the first major's courses.

Due to space constraints, we refrain from showing much Prolog code in this report. However, it is worth noting that considerable effort has been focused on efficiency while developing TAROT. For example, the system avoids excessive backtracking by first scheduling senior-level courses in the last or second-to-last semester. Next, junior-level courses are scheduled, and so on. This ensures introductory courses that serve as prerequisites for later courses, but which may be taken any time since they have the fewest prerequisites, are scheduled under the most constrained environment, when most courses have already been scheduled. This design follows the general design paradigm of constraining the search as early as possible to avoid generating results that are ultimately thrown away by subsequent constraints.

Presently, TAROT suffers from a lack of a proper user interface. Advising questions must be specified as Prolog queries on a command line. In future work, we plan to investigate an appropriate web-based interface that best exposes TAROT's capabilities in a form that is clear and efficient for non-programmers.

4 Evaluation

We evaluate TAROT by demonstrating some example questions and TAROT's responses across the three levels.

Level 0: Questions about the present. Our university uses Ellucian Degree Works [1] to show information about a student's class history and required courses. TAROT has a built-in parser for Degree Works' class history format. We can read this class history and then ask TAROT to calculate the student's GPA:

```
readStudentFile('classHistoryH1.txt', Taken),
calcGPA(Taken, GPA, Credits, AllCredits).
```

The result is a table of the student's class history, including grade earned and individual course credits, followed by their GPA and overall earned credits. We note that the classes and grades shown below are simulated. The query completes in 0.002 seconds on a laptop with an Intel i5 processor and 8 GB RAM.

Transfer	astr180 (Tr, 3)	chem110 (Tr, 4)	csci111 (Tr, 4)	
2016 Fall	csci142 (B+, 4)	csci211 (A-, 4)	rels390 (P, 4)	
2017 Spring	csci201 (B, 4)	csci221 (A-, 4)	hlsc219 (A, 4)	math141 (C+, 4)

GPA: 3.33, Credits: 24, All credits (including transfer): 39.

Level 1: Questions about a possible future. Using this same simulated class history, we can next ask for a schedule that would allow the student to complete his computer science degree in two years. The following query reads the class history, generates four semesters covering the next two years, fills in a schedule, and further ensures there are at least five general education courses in the schedule.

```
readStudentFile('classHistoryH1.txt', Taken),
generateBlankSemesters(2018, 4, Semesters),
finishDegrees([csci], Taken, [], Semesters, PlanSemesters),
ensureGenEdCount(PlanSemesters, 5).
```

The result is a table that gives the proposed schedule. Note that major requirements, course prerequisites, and course offering times are considered while developing the schedule. The query completes in 0.006 seconds.

2018 Fall	cinf401	csci311	csci321	math142
2019 Spring	csci331	csci431	gen. ed.	gen. ed.
2019 Fall	csci498	phys141	gen. ed.	gen. ed.
2020 Spring	csci301	csci499	phys142	gen. ed.

The student may also wish to limit the number of CSCI and CINF and MATH courses to take in the same semester to, say, two courses. TAROT fails

to find a successful schedule (in 1.742 seconds), indicating the constraint is not possible to achieve. Adding another semester does succeed (in 0.437 seconds):

```
readStudentFile('classHistoryH1.txt', Taken),
generateBlankSemesters(2018, 5, Semesters),
finishDegrees([csci], Taken, Semesters, PlanSemesters),
ensureCourseLoad([csci,cinf,math], PlanSemesters, 2),
ensureGenEdCount(PlanSemesters, 5).
```

2018 Fall	csci321	math142	phys141	gen. ed.
2019 Spring	csci331	csci431	phys142	gen. ed.
2019 Fall	cinf401	gen. ed.	gen. ed.	gen. ed.
2020 Spring	csci301	csci498	gen. ed.	gen. ed.
2020 Fall	csci311	csci499	gen. ed.	gen. ed.

Now consider a new student, with no class history, who wishes to study abroad (which we assume involves no transferrable major courses) during 2020 Spring semester. TAROT can plan around this missing semester to find a successful four-year schedule (in 0.006 seconds):

```
generateBlankSemesters(2018, 8, Semesters),
setMissingSemester(Semesters, 2020, spring, Semesters2),
finishDegrees([csci], [], Semesters2, PlanSemesters).
```

Double-majors are also trivial to specify, for example, a MATH+CSCI double major (completed in 0.009 seconds):

```
generateBlankSemesters(2018, 8, Semesters),
finishDegrees([math,csci], [], Semesters, PlanSemesters).
```

Level 2: Questions about all possible futures. If a student wishes to know *when* to study abroad, we can write a query that forces TAROT to find all schedules for all possible study abroad semesters, and then tally the number of schedules that work for each different study abroad selection. We use Prolog's 'findall' feature for this purpose. Naturally, as we develop a more friendly user interface, direct use of Prolog code will be entirely avoided.

Assuming the student is a freshman starting college in Fall 2018, the system finds the following number of working schedules for each possible study abroad semester:

2019 Spring – 47,340 (20%)	2019 Fall – 51,549 (21%)	2020 Spring – 35,469 (15%)
2021 Spring – 32,870 (14%)	2021 Fall – 45,092 (19%)	2022 Spring – 28,026 (12%)

Thus, the student has the most flexibility in her schedule if she studies abroad in 2019 Fall. Note that 2020 Fall is missing due to the frequency in which some required major courses are offered (Fall/Spring; odd/even years).

A student may wish to know all the possible semesters to take a particular course, for example, our department's artificial intelligence course. To answer this question, we first generate all possible schedules with 'findall' starting in Fall 2018, then tally all the different semesters the course appears. The query completes in 112 seconds and reports that the only semester available to take the course is 2021 Spring if the student starts as a freshman in our introductory programming course. If the student has prior programming experience and starts in CSCI 141, the AI course may be taken in 2020 Spring in 20% of possible schedules or 2021 Spring in the other 80% of possible schedules.

Sometimes students wish to know what grades are required in certain courses to achieve a certain GPA. This is straightforward to request from Tarot. We can simply request Tarot to generate a schedule (with unknown grades in classes not yet completed), calculate their GPA with the query 'calcGPA(Taken, GPA, Credits, AllCredits)', and then add a constraint like 'GPA >= 3.5'. We use the CLPR library (constraint logic programming in real numbers) [5] to efficiently establish constraints on real-valued variables, in this case, course grades. CLPR uses a solver for linear equations rather than backtracking search.

Level 3: Questions about the rules. Finally, Tarot can report how similar two majors are by finding schedules for each major separately and then tallying how many courses are in common. This operation takes as much time as generating all possible schedules, twice (about four minutes).

Tarot can also help staff and faculty determine how many teachers are needed to cover all possible schedules for freshmen entering in odd or even years (assuming one section of each course). This information can serve as evidence in a proposal to support a new tenure line – i.e., an argument can be made that at least some number of teachers are required to cover a new major within an existing department. The query involves finding all distinct possible schedules for each relevant major, for freshman starting in an odd year and even year separately. Then, Tarot calculates the number of distinct courses in each semester and divides that number by the course load for teachers (e.g., 3/3). Naturally, this is Tarot's most time-consuming query, requiring 9 minutes, but it is also likely to be the least frequently executed query.

5 Conclusion

This paper described Tarot, a tool that aids students and advisors in everyday, yet sophisticated course advising scenarios. By making use of an integrated planning engine, Tarot is capable of developing long-term course schedules while respecting major requirements and prerequisites. Tarot can also handle arbitrary constraints such as maximum course load, study abroad semesters,

and double majors. By producing and analyzing all possible schedules for a given set of constraints, TAROT can also provide information about the best time to take a certain course or the minimum number of teachers required to cover all courses. In future work, we plan to develop an appropriate user interface for students and advisors that best exposes TAROT's advanced features.

References

[1] Ellucian Degree Works. https://www.ellucian.com/Software/Ellucian-Degree-Works/.

[2] DEKHTYAR, A., GOLDSMITH, J., LI, H., AND YOUNG, B. The bayesian advisor project i: Modeling academic advising. Tech. Rep. 323, University of Kentucky Dept of Computer Science, 2001.

[3] DODSON, T., MATTEI, N., GUERIN, J. T., AND GOLDSMITH, J. An english-language argumentation interface for explanation generation with Markov decision processes in the domain of academic advising. *ACM Transactions on Interactive Intelligent Systems (TiiS) 3*, 3 (2013), 18.

[4] ENGIN, G., AKSOYER, B., AVDAGIC, M., BOZANLI, D., HANAY, U., MADEN, D., AND ERTEK, G. Rule-based expert systems for supporting university students. *Procedia Computer Science 31* (2014), 22–31.

[5] HOLZBAUR, C. Ofai clp(q, r) manual. Tech. Rep. TR-95-09, Austrian Research Institute for Artificial Intelligence, Vienna, 1995.

[6] MOHAMED, A. A decision support model for long-term course planning. *Decision Support Systems 74* (2015), 33–45.

[7] VINCENTI, G., AND BENNETT, V. A JSON-based self-advising system. *Journal of Computing Sciences in Colleges 32*, 3 (2017), 39–45.