Survey on College Course Scheduling Algorithms

Robert Swanson

March 8, 2021

1 Introduction

Academic Planning is a task whose financial and educational importance is grossly understated. The complex requirements of different degrees, differing needs and wants for individual students, and the frequency of unexpected changes make this scheduling task daunting for the undergraduate student. This is a problem not unnoticed by the research community, which has already approached it, and its branching sub-problems, with a myriad of solutions. These strategies solve different variants of this problem, often according to certain assumptions and needs that are relevant to the researcher's undergraduate experience.

Solutions to this problem come in 2 general flavors: genetic algorithms which produce and test potentially invalid options and backtracking searches, which will only search the space of valid options.

2 Implementations

2.1 Genetic Algorithm

Srisamutr, Raruaysong, and Mettanant present a genetic algorithm that creates an optimized plan that considers factors such as prerequisite courses and credit units, also offering historical data to estimate student GPA [15]. They encode plan options into a chromosome and simulate three kinds of mutations between generations: selection, crossover, and mutation. The mutations probabilistically favor chromosomes that score highly according to a fitness function that they define to be a weighted (with coefficients of -7, -6, -5, -4) subtraction of four properties (listed in order): the number of courses assigned after the fourth year, the number of semesters over 22 credit units, the number of prerequisite violations, and the number of course assignments that deviate from a given curriculum plan.

This algorithm offers a few advantages for consideration. First of all, it is simply implemented and its fitness function is easily modified. Although they chose only to consider four qualities, they could have evaluated a chromosome (which represents a complete instance of a plan) according to any number of arbitrary preferences. Additionally, as is true in general for genetic algorithms, this offers the ability to escape local optima and discover new plan variants.

However, this algorithm suffers from one significant shortcoming: run-time. The algorithm offers no method to ensure chromosomes are valid until it operates the fitness function over it, and thereby must search a large number of options that are "clearly" invalid for a smaller number of potentially valid plans. In their experience, they found what they identified as a good solution in 30-60 minutes.

2.2 Backtracking Search

Eckroth and Anderson's Tarot system endeavored to consider more complex scenarios (such as study-abroad semesters). [5] They also chose to only schedule major classes, assuming that a student could easily fill in gen-eds into empty spaces between major classes. This assumption is consistent with recommendations from some human advisors, who often don't consider gen-ed time offerings when developing a course plan, but does limit the system's ability to optimize according to students' preferences relating to those classes.

Tarot was designed to answer questions on four levels: the present, a possible future, all possible futures, and about the rules. It accomplished this by taking advantage of its prolog implementation, which offered a backtracking search which allowed for complex and arbitrary rules that validated a given plan. This allows for the ability to evaluate courses already taken and the requirements already met. By recursively modifying and searching the valid plan space, a plan could be developed to describe a possible future. To answer the question "which semester could I go abroad that minimizes extending my graduation date," Tarot quantifies the number of valid plans for each possible semester taken abroad. Finally, Tarot can compare multiple majors to quantify the overlap in requirements and specify a similarity value.

To minimize backtracking and thereby increase performance, the system schedules higher-level courses (that have more prerequisites) in the last semester, and then works backwards. This strategy could be generalized by starting the search with decisions that have less valid possibilities by taking advantage of known constraints. Such an optimization strategy could potentially be applied elsewhere based on knowledge of the search space (this could be a place for potential work).

2.3 Search Constraints Hybrid

Given two different extremes of solving the problem which vary from offering no constraints (observed in some genetic algorithms) and searching within complete constraints (as in tarot), it's not surprising to see genetic algorithms that implement both paradigms together strategically. Erben and Keppler's genetic algorithm is designed to solve the time-tabling problem (which is a distinct problem mentioned later) and builds a more robust system for generating, mutating, and evaluating chromosomes [6]. Their strategy used Prolog to ensure that all constructed chromosomes are valid (meeting the "hard constraints") and thus the search space serves only to compare known valid timetables based on a weighted sum of violations of "soft constraints".

2.4 Graph Building

Morrow, Sarvestani, and Hurson's PERCEPOLIS [12] [11] system offers another strategy of building a plan which involves the creation and modification of graphs describing the relationship of requirements, and then fitting them into a fixed dimension schedule. This graph uses a virtual root node to fully connect the graph, but otherwise connects nodes to classes for which they are a prerequisite. Every requirement of a degree is included on this graph, including requirements that can be met by more than one class. The algorithm then populates each node with the sum of classes that require it, which is used to specify an "opportunity value" which indicates the number of options created when fulfilling a particular course.

Their system implements a large variety of features designed to make the system "context-aware", such as preferring semester for a class which is known to have higher student performance. It also does this by implementing a summary schema model (SSM) to recommend elective classes based on topics the user expresses interest in. The SSM does this by connecting related words using a thesaurus and by looking at metadata for courses (and their prerequisites), and adding them to the requirements graph.

Once the graph is constructed, the schedule is then filled out by adding first the courses that meet the student's interests (which are assumed to have a large number of required classes). Whenever a class is added, any unmet prerequisites are added first, and the class is placed in the following semester. Once all the classes of interest are inserted, the rest of the classes are inserted in the order of decreasing opportunity value.

This system operates on a set of hard-coded assumptions of preferences which gives it some advantages and disadvantages. For example, this system ought to run faster since it only finds one solution (rather than searching through many solutions), but this also means there is no flexibility to optimize that solution according to other user preferences. Another consequence of this inflexibility is that the user must set a maximum number of courses per semester, which the algorithm will fill up, which does not play well with universities of varied credit-hour courses. Because it has no concept of when courses are offered (assuming they can be scheduled whenever the student wants to), this system would work well for scheduling online classes, but would not be helpful for most universities.

3 Relevant Topics

3.1 The Time Tabling Problem

Srisamutr's paper briefly discusses the university and high school time tabling problem, which takes the university administrator's perspective who need to schedule classes given certain limited number of rooms, faculty, and enrolled students [8] [14]. Extensive research has already been done regarding the complexity of this problem, and some have even developed a formal language structure to define [4]. Researchers have used various algorithms to solve this problem much which fall into the category of genetic algorithms.

One set of researches applied an ant colony algorithm to solve the problem [13], while Adrianto demonstrated that using particle swarm optimization results in significantly smaller penalty than a genetic algorithm [1].

Jat and Yang offer a modification to the genetic algorithm applied to the university timetabling problem which remembers the part of a chromosome which has no penalty, and encourages the passing on of those parts of the chromosome into future generations [7]. Their analysis revealed improved penalty values as compared to the genetic algorithm without the guided searching.

Minh, Thanh, Trang, and Hue used a Tabu search, which works to minimize an objective function [10]. It starts by developing and modifying a "current solution" in solutions that make up its "neighborhood" by operating a "move" to transform the current solution. It tracks the moves already used and adds them to the "tabu list" which will not be used except in certain circumstances (such as if such a move would create the best seen solution).

Aycan and Ayav implemented a simulated annealing algorithm to find an approximation of a global maximum of their cost function [2]. Simulated annealing implements a decreasing tolerance (aka temperature) for investigating solutions which result in a worse cost.

Andersson implemented both Tabu search (TS) and simulated annealing (SA) with the intention to compare runtime and the quality of outputs. He found that both algorithms could find the optimal solution for a small timetable, but only SA could find the optimal solution for a larger solution, as TS is less able to escape local optimum. He also found that SA reached the optimal solution sooner than TS, and suggested using mid-term memory to improve its performance.

Matias, Fajardo, and Medina implemented a genetic algorithm that hybridizes strategies from multiple other implementations [9].

3.2 Other Issues

Work by Chatrurapruek looked into how tools that offer data on courses impact student behavior and performance [3]. Tools such as Carta visualized historical data on student performance and workload, when students took courses in relation to other courses, and the reviews from the course evaluations. When looking into how students behaved when using these tools, they found a reduction in GPA but no evidence of changing in course registration. They believed that students' course decisions were not informed by the tool, but that their behavior when taking the course was negatively impacted by their knowledge of the data.

4 Patterns

4.1 Constraints

Solutions to both the timetabling problem and the student course scheduling problem both fall into two general categories: genetic algorithms, and specialized deterministic searches. Genetic algorithms are more far more flexible in cost function they can minimize, while specialized searches take advantage of certain assumptions of preferences to create a much smaller necessary search space. These constraints are typically not relevant to more than a specific region, and are usually only tested for a specific university, but they do allow for dramatically improved run-time over genetic algorithms, which much do more work to avoid invalid plans.

4.2 Considerations

These optimizations reveal the various possible considerations one might wish to make when scheduling classes.

Most (but not all) had a distinction between hard constraints (requirements) and soft constraints (preferences). Usually, systems implemented prerequisites, course requirements, credit minimums for plans, and credit maximums for individual time periods.

Many such considerations are related to the quantity of courses or credit units taken, whether over a semester or for the entire plan. Many implementations penalized plans that required more than a certain number of semesters or more than a certain number of courses/credits per semester.

Most implementations had some form of considerations that related to the relative ordering of courses, for example, preferring classes that are dependencies for more classes sooner, or penalizing plans for deviating from some sort of manually created plan.

Some systems explicitly worked with common exceptional circumstances, such as double majors, study abroad semester, course override, and transfer credits.

Some systems considered themselves "context-aware" and took into account factors such as historical performance in a particular class or student interests (eg electives similar to keywords).

Most systems only functioned to create an estimation of an optimum plan, but others offered further information, such as outlooks on all possible plans give certain variables and constants (eg. which semester to take abroad for least delay in graduation).

5 Potential Work

Distinctly absent from many systems, is the concept of course offering (eg. which semesters a course is offered), and absent from all systems is any concept of what days time courses will be offered during a semester, and thus doesn't avoid scheduling conflicts within a time period. This is likely because colleges usually can't be confident what days and times they will offer classes and thus don't want to provide recommendations based on it. Nonetheless, such information would likely be more helpful considered and sometimes wrong, then completely ignored, and systems that consider this may offer better plan recommendations. Such a system may observe past years to look for patterns and predict with varying levels of confidence when courses will be offered, even when the course offerings the university provides lacks that information.

Another distinct limitation of all observed work is the specificity with which most of the systems are implemented. All the non-genetic algorithms make a very fine assumption on the context of the user and their university, as well as the preferences they have for an optimal plan. The genetic algorithms, though in concept could be flexible to a large number of preference sets (as the cost is a function of a particular instance of a plan), none of them offered much ability for the user to customize the algorithm towards their particular and specific desires. A system could theoretically offer some ability for the user to customize requirements and preferences, and then dynamically create a cost function to minimize. Such a system could simply implement a single algorithm for all sets of requirements and preferences, or could also intelligently modify the algorithm to optimize for particular preferences (eg set number of semesters vs minimizing semester necessary).

References

[1] Dennise Adrianto. Comparison using particle swarm optimization and genetic algorithm for timetable scheduling. *Journal of Computer Science*, 10(2):341, 2014.

[2] Esra Aycan and Tolga Ayav. Solving the course scheduling problem using simulated annealing. In *2009 IEEE International Advance Computing Conference*, pages 462–466. IEEE, 2009.

[3] Sorathan Chaturapruek, Thomas S Dee, Ramesh Johari, René F Kizilcec, and Mitchell L Stevens. How a data-driven course planning tool affects college students' gpa: evidence from two field experiments. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, pages 1–10, 2018.

[4] Tim B. Cooper and Jeffrey H. Kingston. The complexity of timetable construction problems. In Edmund Burke and Peter Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153, pages 281–295. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

[5] Joshua Eckroth and Ryan Anderson. Tarot: A course advising system for the future. *J. Comput. Sci. Coll.*, 34(3):108–116, January 2019.

[6] Wilhelm Erben and Jürgen Keppler. A genetic algorithm solving a weekly course-timetabling problem. In Edmund Burke and Peter Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153, pages 198–211. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

[7] Sadaf Naseem Jat and Shengxiang Yang. A guided search genetic algorithm for the university course timetabling problem. 2009.

[8] Rhydian Lewis. A survey of metaheuristic-based techniques for university timetabling problems. 30(1):167–190.

[9] Junrie B Matias, Arnel C Fajardo, and Ruji P Medina. A hybrid genetic algorithm for course scheduling and teaching workload management. In *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, pages 1–6. IEEE, 2018.

[10] Khang Nguyen Tan Tran Minh, Nguyen Dang Thi Thanh, Khon Trieu Trang, and Nuong Tran Thi Hue. Using tabu search for solving a high school timetabling problem. In *Advances in intelligent information and database systems*, pages 305–313. Springer, 2010.

[11] Tyler Morrow, Ali R Hurson, and Sahra Sedigh Sarvestani. A multi-stage approach to personalized course selection and scheduling. In *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 253–262. IEEE, 2017.

[12] Tyler Morrow, Sahra Sedigh Sarvestani, and Ali R Hurson. Algorithmic decision support for personalized education. In *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, pages 188–197. IEEE, 2016.

[13] Clemens Nothegger, Alfred Mayer, Andreas Chwatal, and Günther R. Raidl. Solving the post enrolment course timetabling problem by ant colony optimization. 194(1):325–339.

[14] A Schaerf. A survey of automated timetabling. page 42.

[15] A. Srisamutr, T. Raruaysong, and V. Mettanant. A course planning application for undergraduate students using genetic algorithm. In *2018 Seventh ICT International Student Project Conference (ICT-ISPC)*, pages 1–5, 2018.