Research Initial Plan

Student: Jenny Zeng Adviser: Professor David Eppstein

1 Main Idea

Let's consider the problem of scheduling courses for next quarter. To graduate as soon as possible, what courses should a CS student choose? Currently, many students, especially freshmen and transfer students, would like to ask school academic counselors for advice before enrolling to courses of next quarter. Since the number of counselors is limited but many students would go to the office for help, getting suggestions becomes time consuming. Therefore, my goal is to generate a sample study plan similar to the one shown in UC Irvine Department of Computer Science website[1]. Given some completed courses, it should display specific upper courses for each quarter except General Education courses.

If an application of course scheduling is developed, academic counselors can use this tool to shorten the time spent on arranging student schedules and provide accurate information. At the same time, students can check what series of courses they are required to take to satisfy some prerequisites and organize their study plans at ease.

Sample Program of Study — Computer Science

Freshman		
Fall	**Winter**	**Spring**
I&C SCI 31	I&C SCI 32	I&C SCI 33
MATH 2A	MATH 2B	IN4MATX 43
WRITING 39A	WRITING 39B	I&C SCI 6B
I&C SCI 90	General Education III	WRITING 39C
Sophomore		
Fall	**Winter**	**Spring**
I&C SCI 51	I&C SCI 46	Computer Science Spec./Elective
I&C SCI 6D	I&C SCI 53	STATS 67
I&C SCI 45C	I&C SCI 53L	General Education III
	I&C SCI 6N	
Junior		
Fall	**Winter**	**Spring**
COMPSCI 161	Computer Science Spec./Elective	Computer Science Spec./Elective
Science Elective	Computer Science Spec./Elective	Computer Science Spec./Elective
General Education III	I&C SCI 139W	Science Elective
General Education VII	General Education VIII	General Education VI
Senior		
Fall	**Winter**	**Spring**
Computer Science Spec./Elective	Computer Science Spec./Elective	Computer Science Spec./Elective
Computer Science Spec./Elective	Computer Science Spec./Elective	Computer Science Spec./Elective
General Education IV	General Education IV	General Education IV

Figure 1: Sample Study Plan on UCI Department of Computer Science Website [1]

2 Related Algorithm

This is a problem related to job shop scheduling and graph drawing, so I may solve it by implementing the Coffman-Graham algorithm [2] or some other related algorithms in this field.

2.1 Coffman-Graham algorithm

It is an algorithm that can solve the problem of finding a schedule with a fixed number of scheduled items per time period. Here are some characteristics of this algorithm.

1. For number of items in a level, represented by w, if $W > 2$, this algorithm can find a solution within a factor of $2 - 2/W$ of optimal and its time complexity is $O(n^2)$ for n elements. It guarantees a certain quality to the solution.

2. Unlike many shortest path algorithms, such as Dijkstra algorithm, Coffman-Graham algorithm allow managing more than one items in a level.

3. There are precedence constraints for jobs in Coffman-Graham algorithm just as pre-requisites for the courses.

4. The algorithm restricts the number of items in each level by a fixed width bound W, and the course scheduling problem also requires a restriction to limit the number of courses per quarter.

2.2 Layered Graph Drawing Framework

I may try using the Layered Graph Drawing Framework outlined in *Methods for visual understanding of hierarchical system structures* (Sugiyama, Tagawa & Toda) [3].

1. Because the input graph in this problem is already a directed acyclic graph (DAG), I may skip the first stage of choosing a feedback arc set and reversing the edges of the set to convert the input into a DAG.

2. In the framework, A course can be viewed as a vertex and I can use constraints to limit the number of vertices sharing the same y-coordinate. If courses are sharing the same y-coordinate, they are on the same edge.

3. The *dummy vertices* are introduced in the third stage and I have to figure out what they represent in the problem.

4. In the fourth stage, it is noted that for the vertices with the same y-coordinate, they are arranged to minimize the number of crossings, so I may define some rules to achieve this goal.

5. I will try drawing vertices and edges in the graph according to x and y coordinates.

2.3 Difficulties and Interesting Points

Due to some limitations of Coffman-Graham algorithm on course scheduling problem, I will modify it to accomplish the main goal as well as to explore some interesting points below.

1. The Coffman-Graham algorithm specifies a bound w to restrict the number of vertices per level, but this is not proper and flexible in course scheduling. Hence, It would be interesting to apply particular constraints to restrict the number of vertices in a level instead of a fixed width bound.

2. While Coffman-Graham algorithm assigns all of the jobs into the schedule and assumes that the task is finish if all jobs are allocated, course scheduling problem does not require a student to take all CS courses in order to graduate. Accordingly, the actual algorithm should only allocate some of the jobs, rather than all of them, into the schedule and terminates when the schedule satisfies all requirements.

3. There is a presumption in the algorithm that all jobs are available at any time inv the schedule. Nevertheless, in this problem, not all courses are offered every quarter. Thus, it is significant to change the algorithm to produce a solution given a condition

that not all of the jobs are available for every time period. There may be two different ways to resolve this condition:

(a) Before starting to schedule classes in a level, choose which classes to include in the schedule.

(b) Allow more choices to be left open. This might help make the schedule fit.

I will test which approach can get a better result and is easier to implement.

4. It is noted at point 1 in subsection 2.1 that the algorithm gives certain guarantees to the quality of the solution. For w, number of jobs in a level, the solution is within a factor of $2 - 2/w$ of optimal. If in this course scheduling problem, number of courses per quarter is four, $w = 4$ and the quality is $2 - 2/4 = 3/2$. After making changes to the algorithm, it would be meaningful to test the actual quality of the solution.

3 Schedule

This research project will be last for two quarters, and there will be a meeting every week for reporting the process and discussing problems met in the project.

3.1 2017 Winter Quarter

During winter quarter, I will implement the algorithm and resolve difficulties described. I will also have a working application done by the end of this quarter.

- **Week 1**: Find a method to represent the study plan graph.
- **Week 2**: Apply the basic Coffman-Graham algorithm, which defines a fixed width bound W, and test its performance in this particular course scheduling problem assuming that the number of classes a student takes will not exceed a fixed width bound.
- **week 3**: Collect courses information online by using web crawlers or a UCI-Course-API [4] on GitHub.
- **Week 4, 5, 6**: Modify the algorithm to solve difficulties illustrated in the subsection 2.3 and test the quality of the final algorithm.
- **week 7, 8, 9, 10**: Implement the algorithm by using collected data. May make a simple working command-line interface first, and if time permits, develop a web-based application.

3.2 2017 Spring Quarter

During spring quarter, I will focus on writing and revising the project report.

- **Week 1**: An outline of the project report.
- **Week 2**: First 1/3 of the project report.

- **Week 3**: Second 1/3 of the project report.
- **Week 4**: Third 1/3 of the project report.
- **Week 5, 6**: Second draft of the project report.
- **Week 7, 8**: Third draft of the project report.
- **Week 9**: Forth draft of the project report.
- **Week 10**: Final draft of the project report.

References

[1] Sample Program of Study - Computer Science,
http://catalogue.uci.edu/donaldbrenschoolofinformationandcomputersciences/departmentofcomputerscience/#majorstext.html

[2] Coffman, E. G., and R. L. Graham. Optimal Scheduling for Two-processor Systems. *Acta Informatica*, 1: 200213, doi:10.1007/bf00288685, MR 0334913.

[3] Sugiyama, Tagawa, and Toda. Methods for Visual Understanding of Hierarchical System Structures. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 2, 1981, pp. 109125. doi:10.1109/tsmc.1981.4308636.

[4] UCI-Courses-API
https://github.com/djchie/uci-course-api