Case Study 2: Harvey Mudd CS Major Scheduler

Hannah Davalos, Tenny Liu, Camille Simon & Ian Taylor

May 7th, 2019

1 Abstract

There is nothing harder for college students than planning out their four year schedule; there are so many required classes to take, pre-requisites to consider, and subject areas beyond their major that they wish to explore. This is especially true for Harvey Mudd Computer Science (Mudd CS) majors, whose required classes fill up at the blink of an eye and whose major has so much flexibility that it is hard plan their schedule so that the workload stays relatively balanced. In order to address these problems, we created a course scheduling tool that can be used by Mudd CS majors.

To use our scheduler, these students will only have to input the following information into an Excel spreadsheet: classes they've taken, classes they wish to take in a particular semester (if any), the number of semesters they have left, and the season (fall/spring) of the upcoming semester. It then returns a schedule for the student's remaining semesters considering requirements needed to graduate from Mudd. Our scheduler's objective is to minimize workload each semester while penalizing for over-loading or under-loading on credits, thus encouraging a balanced workload across the remaining semesters and graduating on-time. Our scheduler works especially well for Mudd CS majors because it includes a pinning feature that allows students to customize their schedule to ensure a certain class is or is not taken during a particular semester. Our scheduler does a great job in creating feasible schedules, but it does not work well when our simplifying assumptions do not apply to the student, and there often exist major differences between the amount of credits suggested to take between semesters. Our scheduler and these findings will be further discussed in this report.

2 Executive Summary

We know that it can be overwhelming for students to create a class schedule for their remaining semesters that allows them to take both required and desired classes, all while balancing their workload. This problem is particularly applicable for CS majors whose classes are in high demand and whose number of requirements makes it hard to schedule their classes while keeping their workload balanced. Thus, we decided to create a scheduler that helps Mudd CS majors create their schedules for their remaining semesters at Mudd while minimizing the workload each semester.

To achieve a balanced workload between semesters, our scheduler needs to know the workload of each class that CS majors may take. We found this workload by sending a survey out to the student body of Harvey Mudd College seeking responses solely from CS majors. The survey asked respondents to give input on how many hours (outside of class) each CS major or elective class requires if they have already taken the class. We took this data and averaged the results to come up with a workload metric in hours for each class a CS major may encounter in their four years. Our scheduler balances workload by minimizing the determined workload of the classes it chooses for them to take each semester, as it calculates the workload directly from these class choices. To further ensure a balance of workload, our scheduler penalizes

for under-loading (taking under 12 credits typically, but customizable) and over-loading (taking over 18 credits typically, but customizable) in order to encourage an unstressed, on-time graduation.

In order for our scheduler to optimize a schedule in this way, it uses a combination of binary integer decision variables. Our scheduler uses one binary variable to determine whether or not a class is taken in a particular semester, and it uses two integer decision variables to determine the number of credits a student over-loads or under-loads during each semester.

To create an optimized schedule, our scheduler also considers several constraints to account for the wants and needs of CS majors at Mudd. These constraints are based on parameters that inform the scheduler of the requirements and preferences of a student seeking a schedule. The following are the parameters that our constraints are based on:

- CS major courses and the season (fall/spring) they are being offered
- the workload each courses requires (hours)
- the prerequisites of each CS major course
- the credits that each course is worth
- Mudd graduation credit requirements
- the credit requirements for over-loading and under-loading (or the minimum and maximum amount of credits the user wishes to take each semester)
- the number of semesters that the student has left
- classes a student wishes to take or not taken during a particular semester

Thus, our scheduler determines whether or not a class is taken in consideration of these parameters. It does this while aiming to balance the workload across the semester without over-loading, under-loading, or exceeding the bounds of credits that the student inputs into the scheduler.

3 Technical Summary

3.1 Introduction

Our scheduler aims to help Mudd CS majors create a class schedule for their remaining semesters that balances the workload for each semester. Our scheduler does this by considering a series of sets, parameters, decision variables, and simplifying assumptions.

In order to create schedules, our scheduler has to consider sets that outline classes that the student must take at some point in their time at Mudd, as well as the number of semesters that the student has left and is looking to make a schedule for:

- Classes (j is an element of this set) is the set of all CS courses being offered
  - Major Requirements, a subset of Classes, is the set of all courses required for the CS major
  - Core Requirements, a subset of Classes, is the set of all courses required to satisfy the Core requirements
  - HSA Requirements, a subset of Classes, is the set of generically named courses required to satisfy the HSA requirement

- Semesters (i is an element of this set) is the set of all semesters that the student user has left in their four years

In order to create the schedule and determine the amount of credits a student will take each semester, our scheduler utilizes the the following decision variables:

- y_i is the number of credits by which the student underloads per semester i
- l_i is the number of credits by which the student overloads per semester i
- $z_{i,j}$ is a binary variable that represents whether or not a student is taking class j in semester i, and l_i is the number of hours the student goes over the average workload in a given semester i
- $n_{i,j}$ is a binary variable that represents whether or not a student is opposed to taking class j in semester i
- $t_{i,j}$ is a binary variable that represents whether or not a student inputs that they take class j in semester i

In order to create a schedule with balanced workload across each semester, the following parameters must be considered:

- w_j is the workload that each class j requires (hours)
- c_j is the number of units for each class j
- *Min_credits* is the minimum number of credits needed each semester to be a full-time student (according to student choice)
- *Max_credits* is the maximum number of credits a student can take without overloading (according to student choice)
- *Min_grad_cred* is the minimum number of credits needed to graduate from Mudd
- CS major courses and the season (fall/spring) they are being offered
- the prerequisites of each course j
- the number of semesters that the student has left
- $pinTake_{j,i}$ is whether or not course j is wished to be taken during semester i
- $pinNoTake_{j,i}$ is whether or not course j is wished to not be taken during semester i
- $alreadyTaken_j$ is whether or not course j has already been taken by the student in a previous semester

The parameters that our scheduler considers are all easily accessed information either from the requirements of the CS major or from the student inserting their information for the scheduler to consider, with the exception of the workload that each class requires. In order to determine this parameter, we created and distributed a survey to gather student input on the workload in hours beyond class that are required for every CS class offered at Mudd. We received 77 responses from students on the student mailing list at Mudd which offered feedback that covered 68.5% of the CS courses. We took these responses and averaged them to determine the workload for each class. For the courses that we did not receive feedback on, we estimated the workload based on the amount of credits of the course.

Our scheduler also considers some simplifying assumptions. The first of these being that the student using our scheduler will or did not take CS 42. While the student can alternatively input that they want to

or have taken both CS 5 and CS 60, this will cause the scheduler to inaccurately determine the amount of credits taken by the time of the student's graduation in the schedule that it creates. Another simplifying assumption that we made was that the student will not take summer courses; our scheduler only determines schedules based on the classes available during the fall and spring.

To use our scheduler, see Appendix A. The next section provides the algebraic formulation of the scheduler that was outlined in this section.

3.2 Scheduler Formulation

$$\min Z_{i,j} + My_i + Ml_i$$

such that: the total workload of every semester is less than a variable Z, which we attempt to minimize,

$$Z \geq \sum_{j \in \text{Classes}} w_j z_{i,j} \qquad \forall i \in \text{Semesters},$$

every class can only be taken once,

$$1 \geq \sum_{i \in \text{Semesters}} z_{i,j} \qquad \forall j \in \text{Classes}text,$$

the total credits of all classes taken and suggested to take satisfy the minimum credit requirement for graduation,

$$\text{Min\_grad\_credits} \geq \sum_{j \in \text{Classes}} \sum_{i \in \text{Semesters}} c_j z_{i,j},$$

student would have completed the prerequisites for a certain class before the semester it's suggested to be taken,

$$\sum_{s'=0}^{s-1} z_{p_j,s'} \geq z_{j,s} \qquad \forall j \in \text{Classes}, p \in \text{Prerequisites}, s \in \text{Semesters},$$

student complete all classes in major, core, and HSA requirements,

$$\sum_{i \in \text{Semesters}} z_{i,j} = 1 \qquad \forall j \in \text{Major, Core \& HSA Requirements},$$

the sum of credits to be taken and underloading credits for each semester satisfy the minimum credit requirement (underloading is heavily penalized in this scheduler in order to to remain full-time for scholarship/fin-aid, but use can remove penalty or set lower Min credits if they wish to do so),

$$\sum_{j \in \text{Classes}} z_{i,j} c_j + y_i \geq \text{Min\_credits} \qquad \forall i \in \text{Semesters},$$

student doesn't take more credits than their ideal maximum credits for each semester (doesn't overload),

$$\sum_{j \in \text{Semesters}} z_{i,j} c_j - l_i \leq \text{Max\_credits} \qquad \forall i \in \text{Semesters},$$

a certain class to be taken in a certain semester

$$z_{i,j} = 1 \qquad \forall i \in \text{Semesters}, j \in \text{Pin\_Take}_i,$$

a certain class not to be taken in a certain semester (e.g. slots filled up, preference of professor, etc.)

$$z_{i,j} = 0 \qquad \forall i \in \text{Semesters}, j \in \text{Pin\_No\_Take}_i,$$

courses can be offered only in the spring or in the fall

$$z_{i,j} \leq \text{offered\_fall}_j \quad \forall i \in \{s \in \text{Semesters} : s \bmod 2 = \text{next\_semester\_fall and } s \neq 0\}, j \in \text{Classes}$$
$$z_{i,j} \leq \text{offered\_spring}_j \quad \forall i \in \{s \in \text{Semesters} : s \bmod 2 \neq \text{next\_semester\_fall and } s \neq 0\}, j \in \text{Classes}$$

3.3 Results

Our scheduler outputs a 2D matrix that consists of 1s and 0s. The rows are the optional and required classes the student can take and the columns are the semesters that the student has left at Harvey Mudd. A 1 in a given row and column indicates that the student should take that class in the given semester. A 0 indicates that the student should not take the class at that time. Semester 0 has 1s corresponding to classes that the student has already taken. The full output is included in Appendix D.

When running the scheduler to determine a schedule for Ian as an incoming Junior(his user input can be found in Appendix C), we achieved the following result:

Semester	Junior Fall	Junior Spring	Senior Fall	Senior Spring
Classes	Breadth_4	CS_140	CS_105	CS_159
	Breadth_5	CS_151	CS_131	CS_184
	CS_133	FREE_10	CS_155	HSA_10
	CS_153	FREE_4	CS_183	HSA_FREE_1
	CS_195	HSA_FREE_2	CS_121	
	FREE_11	PE_4	Depth_5	
	FREE_3			
Total Credits/unit	17	14.5	18	12
Total Workload/hr	26	23	30	19

In the schedule, HSA_FREE and FREE respectively indicate that Ian has the freedom to choose any humanities and any general class to fulfill requirements. It's important to note here that the workload for senior Fall appear to be higher because he "pinned" several higher level CS electives and reserved a spot for an art class(Depth_5) he wishes to take for his HSA concentration for that semester. Otherwise, the workload and credits for each semester appear to be consistent and well-balanced in terms of STEM and humanities.

4 Conclusions/Future Applications

Our scheduler suggests the classes a student should take each semester for the rest of their college career in order to complete all requirements of the Harvey Mudd CS major. We tested our scheduler with various input data and found that our scheduler does provide a feasible recommendation to students regarding classes they should take. Our scheduler is also highly customizable because students can pin certain elective classes to particular semesters or even add them to the major classes list to ensure that

they are taken. The user can also make use of the pinning classes feature to account for retaking classes and unpinning for classes being filled up or preference of a different professor. The "Next is Fall?" input can tackle fall/spring scheduling even in situations where the student has taken semesters off.

However, there are some edge cases that our scheduler could not fully encompass, including the differences related to taking CS5 or CS42 during freshman year. If the user took CS42, they must input CS5 and CS60 as courses they have taken instead because both are prerequisites for CS70. The scheduler does count that the student took 6 credits instead of 3, but the student could see this and possibly add another elective class to their schedule to account for the 3 additional credits. Another case to consider is when a student takes summer courses, as the scheduler currently does not have summer as an option to take a course. So, if a students plans on taking a summer course, they would just not take that course in the semester specified by the scheduler. However, if a student has already taken a summer course, they would just input that into the scheduler and it would count as a class that has already been taken. Although these edge cases make it slightly harder for the user to navigate the schedule produced by our scheduler, it is far more rewarding to have a feasible schedule that can be used during future semesters at Mudd and that eliminates the stress from choosing classes each semester.

Additionally, though our scheduler minimizes workload per semester, it does not necessarily provide a consistent number of credits to take across semesters. In future applications of our scheduler, this could be addressed such that the student will have a schedule that minimizes the differences in the number of credits taken across semesters so that the workload is truly balanced.

Even though our scheduler is designed for Mudd CS majors, our final product can be easily modified to accommodate students of other majors and of other colleges graduation requirements by changing the content of subsets. We have made this program more accessible by producing a script that reads in .csv files of course information and requirements into our scheduler and yields the optimal schedule. We have made our program public and envision departments adopting our program as a registration advising tool for their students. In other words, our user can simply modify .csv based on our given template and create a customized schedule for their own school/major. A link to use our scheduler with full instructions and templates can be found in Appendix A.

5 Appendices

5.1 Appendix A: Link to Use Scheduler

To use our scheduler, see the following link: https://github.com/ianrabt/HMC-Scheduler

5.2 Appendix B: Course List

Course Code	Fall?	Spring?	Credits	Workload	Prereqs	...
BIO_52	0	1	3	3		
Breadth_1	1	1	3	3		
Breadth_2	1	1	3	3		
Breadth_3	1	1	3	3		
Breadth_4	1	1	3	3		
Breadth_5	1	1	3	3		
CHEM_23A	1	0	3	3		
CHEM_23B	0	1	1.5	3		
CS_105	1	1	3	5	CS_70	
CS_111	1	1	3	6	CS_70	
CS_121	1	1	3	8	CS_70	
CS_124	1	1	3	6	CS_60	
CS_125	1	1	3	6	CS_105	
CS_131	1	1	3	8	CS_70	CS_81
CS_132	0	1	3	6	CS_105	CS_131
CS_133	1	0	3	8	CS_70	CS_81
CS_134	0	1	3	12	CS_105	
CS_136	1	0	3	6	CS_105	
CS_137	1	0	3	6	CS_105	
CS_140	1	1	3	8	CS_70	CS_81
CS_142	1	0	3	6	CS_81	
CS_144	0	1	3	2	MATH_60_MATH_65	CS_60
CS_147	0	1	3	6	CS_70	MATH_30_MATH_35
CS_151	1	1	3	4	CS_70	
CS_152	1	0	3	4	CS_60	
CS_153	1	0	3	6	CS_60	
CS_154	0	1	3	6	CS_60	
CS_155	1	0	3	7	CS_70	
CS_156	0	1	3	6	CS_105	CS_140
CS_157	0	1	3	6	CS_155	
CS_158	1	0	3	7	CS_151	
CS_159	0	1	3	5	CS_81	
CS_181	1	1	3	7		
CS_183	1	0	3	8		
CS_184	0	1	3	8		
CS_189	1	1	1	4	CS_5	
CS_195	1	1	0.5	2		
CS_5	1	0	3	5		
CS_60	1	1	3	5	CS_5	
CS_70	1	1	3	8	CS_60	
CS_81	1	1	3	6	MATH_55	CS_60
Depth_1	1	1	3	3		
Depth_2	1	1	3	3		
Depth_3	1	1	3	3		
Depth_4	1	1	3	3		
Depth_5	1	1	3	3		
Depth_6	1	1	3	3		
ENGR_79	1	0	3	3	MATH_40_MATH_45	
FREE_1	1	1	3	1		
FREE_10	1	1	1.5	1		

Course Code	Fall?	Spring?	Credits	Workload	Prereqs	...
FREE_11	1	1	1.5	1		
FREE_12	1	1	1.5	1		
FREE_2	1	1	3	1		
FREE_3	1	0	3	1		
FREE_4	0	1	3	1		
FREE_5	0	1	3	1		
FREE_6	1	0	3	1		
FREE_7	1	1	3	1		
FREE_8	1	1	1.5	1		
FREE_9	0	1	1.5	1		
HSA_10	0	1	3	3	WRIT_01	
HSA_FREE_1	1	1	3	3		
HSA_FREE_2	1	1	3	3		
LAB_1	1	1	1	2		
LAB_2	1	1	1	2		
MATH_30_MATH_35	1	0	3	3		
MATH_40_MATH_45	0	1	3	3		
MATH_55	1	1	3	6	MATH_30_MATH_35	MATH_40_MATH_45
MATH_60_MATH_65	1	0	3	3		
PE_1	1	1	1	0		
PE_2	1	1	1	0		
PE_3	1	1	1	0		
PE_4	1	1	1	0		
PHYS_22	1	1	1	3		
PHYS_23	1	0	1.5	3		
PHYS_24	0	1	3	3	PHYS_23	
PHYS_51	1	0	3	3	PHYS_24	
WRIT_01	1	0	1.5	3		

5.3 Appendix C: User Input

Classes Taken	Sem. Left	Min credits	Max credits	Next is Fall?	Pinned Course	Sem.	Take/Don't take
CS_70	4	12	18	1	CS_105	2	0
MATH_55					CS_140	2	1
CS_81					Depth_5	3	1
CS_60					CS_155	3	1
MATH_30_MATH_35					CS_159	4	1
CHEM_23A					CS_183	3	1
LAB_1					CS_184	4	1
WRIT_01							
PE_1							
FREE_8							
MATH_60_MATH_65							
PHYS_51							
LAB_2							
PE_2							
Breadth_1							
Breadth_2							
BIO_52							
MATH_40_MATH_45							
CHEM_23B							
PHYS_22							
Depth_1							
Depth_2							
PHYS_24							
CS_81							
CS_189							
FREE_9							
Depth_3							
Breadth_3							
Depth_4							
PE_3							
PHYS_23							
ENGR_79							
CS_5							

5.4 Appendix D: scheduler Output

```
ampl: include schedule.run;
CPLEX 12.8.0.0: optimal integer solution; objective 97
0 MIP simplex iterations
0 branch-and-bound nodes
take [*,*]
:                0   1   2   3   4    :=
BIO_52           1   0   0   0   0
Breadth_1        1   0   0   0   0
Breadth_2        1   0   0   0   0
Breadth_3        1   0   0   0   0
Breadth_4        0   1   0   0   0
Breadth_5        0   1   0   0   0
CHEM_23A         1   0   0   0   0
CHEM_23B         1   0   0   0   0
CS_105           0   0   0   1   0
CS_111           0   0   0   0   0
CS_121           0   0   0   1   0
CS_124           0   0   0   0   0
CS_125           0   0   0   0   0
CS_131           0   0   0   1   0
CS_132           0   0   0   0   0
CS_133           0   1   0   0   0
CS_134           0   0   0   0   0
CS_136           0   0   0   0   0
CS_137           0   0   0   0   0
CS_140           0   0   1   0   0
CS_142           0   0   0   0   0
CS_144           0   0   0   0   0
CS_147           0   0   0   0   0
CS_151           0   0   1   0   0
CS_152           0   0   0   0   0
CS_153           0   1   0   0   0
CS_154           0   0   0   0   0
CS_155           0   0   0   1   0
CS_156           0   0   0   0   0
CS_157           0   0   0   0   0
CS_158           0   0   0   0   0
CS_159           0   0   0   0   1
CS_181           0   0   0   0   0
CS_183           0   0   0   1   0
CS_184           0   0   0   0   1
CS_189           1   0   0   0   0
CS_195           0   1   0   0   0
CS_5             1   0   0   0   0
CS_60            1   0   0   0   0
CS_70            1   0   0   0   0
CS_81            1   0   0   0   0
Depth_1          1   0   0   0   0
Depth_2          1   0   0   0   0
```

```
Depth_3           1   0   0   0   0
Depth_4           1   0   0   0   0
Depth_5           0   0   0   1   0
Depth_6           0   0   0   0   0
ENGR_79           1   0   0   0   0
FREE_1            0   0   0   0   0
FREE_10           0   0   1   0   0
FREE_11           0   1   0   0   0
FREE_12           0   0   0   0   0
FREE_2            0   0   0   0   0
FREE_3            0   1   0   0   0
FREE_4            0   0   1   0   0
FREE_5            0   0   0   0   0
FREE_6            0   0   0   0   0
FREE_7            0   0   0   0   0
FREE_8            1   0   0   0   0
FREE_9            1   0   0   0   0
HSA_10            0   0   0   0   1
HSA_FREE_1        0   0   0   0   1
HSA_FREE_2        0   0   1   0   0
LAB_1             1   0   0   0   0
LAB_2             1   0   0   0   0
MATH_30_MATH_35   1   0   0   0   0
MATH_40_MATH_45   1   0   0   0   0
MATH_55           1   0   0   0   0
MATH_60_MATH_65   1   0   0   0   0
PE_1              1   0   0   0   0
PE_2              1   0   0   0   0
PE_3              1   0   0   0   0
PE_4              0   0   1   0   0
PHYS_22           1   0   0   0   0
PHYS_23           1   0   0   0   0
PHYS_24           1   0   0   0   0
PHYS_51           1   0   0   0   0
WRIT_01           1   0   0   0   0
;

ampl: display sum{c in Classes} credit[c]*take[c,1];
sum{c in Classes} credit[c]*take[c,1] = 17

ampl: display sum{c in Classes} credit[c]*take[c,2];
sum{c in Classes} credit[c]*take[c,2] = 14.5

ampl: display sum{c in Classes} credit[c]*take[c,3];
sum{c in Classes} credit[c]*take[c,3] = 18

ampl: display sum{c in Classes} credit[c]*take[c,4];
sum{c in Classes} credit[c]*take[c,4] = 12
```