

# Extending the Student Trajectory Optimizer for Broader Academic Coverage and Accessibility

A Major Qualifying Project (MQP) Report  
Submitted to the Faculty of  
WORCESTER POLYTECHNIC INSTITUTE  
in partial fulfillment of the requirements  
for the Degree of Bachelor of Science in  
  
Data Science

By:  
Jennifer Kimball,  
Matthew Suyer

Advisor:  
Andrew Trapp, PhD

*This report represents the work of one or more WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on the web without editorial or peer review. For more information about the projects program at WPI, see <http://www.wpi.edu/Academics/Projects>.*

## Abstract

This project extends a course trajectory planner that facilitates optimal course planning for WPI students by minimizing the number of credits needed to finish their programs of study, while secondarily maximizing agency and flexibility. Student body coverage was expanded to approximately 62.9% of the undergraduate population and 35.3% of the graduate population by adding new majors, double majors, and master's programs, and a user interface was developed to increase accessibility. A variety of experiments were performed to test different use cases, and the foundation was laid for future support for combined bachelor's/master's degrees, in addition to other future extensions that incorporate time-sensitive scheduling data and user feedback. The advancements made in this MQP have the potential to help students avoid spending more in tuition on extra courses.

## Acknowledgements

We would like to thank the numerous individuals who have aided in the success of our project. First and foremost, we would like to thank our advisor, Professor Andrew Trapp, for his mentorship and guidance throughout the project. We would also like to thank Interim Provost Arthur Heinricher and Assistant Dean of Student Success Paul Reilly for their vested interest in our project. Finally, we would like to thank our friends, family, and Kendall for their moral support.

# Table of Contents

<b>Abstract</b> .....	<b>2</b>
<b>Acknowledgements</b> .....	<b>3</b>
<b>List of Figures</b> .....	<b>6</b>
<b>1 Introduction</b> .....	<b>7</b>
<b>2 Background</b> .....	<b>11</b>
<b>2.1 The WPI Plan</b> .....	<b>11</b>
<b>2.2 Double Counting</b> .....	<b>12</b>
<b>2.3 Other Trajectory Optimization Tools</b> .....	<b>14</b>
<b>3 Methods</b> .....	<b>16</b>
<b>3.1 The WPI Student Trajectory Optimizer</b> .....	<b>16</b>
3.1.1 Requirements .....	16
3.1.2 Buckets .....	18
3.1.3 Model Overview .....	19
3.1.4 Penalizing Overflow .....	19
<b>3.2 Scope of Implementation</b> .....	<b>23</b>
<b>3.3 Creation of New Programs</b> .....	<b>25</b>
3.3.1 Data Acquisition .....	25
3.3.2 Bucket Scraper.....	26
<b>3.4 M.S. Implementation</b> .....	<b>28</b>
<b>3.5 Online User Interface</b> .....	<b>30</b>
<b>3.6 Independent Studies</b> .....	<b>34</b>
<b>4. Experiments and Results</b> .....	<b>38</b>
<b>4.1 Experimental Setup</b> .....	<b>38</b>
<b>4.2 Experiments on Example Student Schedules</b> .....	<b>38</b>
4.2.1 Experiment A – Common Use .....	39
4.2.2 Experiment B – Overflow Prevention .....	40
4.2.3 Experiment C – Input Error.....	41
4.2.4 Experiment D – Independent Studies.....	44
4.2.5 Experiment E – Double Major .....	45
4.2.6 Experiment F – Suboptimal Double Major .....	46
4.2.7 Experiment G – Master’s Student .....	49
<b>5. Conclusion</b> .....	<b>51</b>
<b>5.1 Discussion</b> .....	<b>51</b>
<b>5.2 Limitations</b> .....	<b>52</b>
<b>5.3 Future Work</b> .....	<b>54</b>
<b>Appendices</b> .....	<b>57</b>

<b>Appendix A – Data Science B.S. Tracking Sheet.....</b>	<b>57</b>
<b>Appendix B – Data Science Major Requirements Sheet .....</b>	<b>58</b>
<b>Appendix C – Data Science Major Bucket Sheet.....</b>	<b>59</b>
<b>Appendix D – Stage I Complete Mathematical Model.....</b>	<b>60</b>
<b>Appendix E – Stage II Complete Mathematical Model .....</b>	<b>63</b>
<b>Appendix F – Cross Restrictions.....</b>	<b>64</b>
<b>Appendix G – The B.S./M.S. Challenge .....</b>	<b>67</b>
<b><i>References .....</i></b>	<b><i>70</i></b>

# List of Figures

Figure 1 – Undergraduate Major Distribution [19].....	24
Figure 2 – Graduate Program Distribution [19].....	25
Figure 3 – Web User Interface for the Course Schedule Optimization Tool.....	30
Figure 4 – Degree Program Input Section of the Online User Interface .....	31
Figure 5 – Instructions Section of the Online User Interface with Welcoming Instructions.....	31
Figure 6 – Course Entry/Upload Section of the Online User Interface .....	33
Figure 7 – Results Page of the Online User Interface.....	34
Figure 8 – Independent Study Form Example .....	36
Figure 9 – Experiment A Results.....	40
Figure 10 – Experiment B Results .....	41
Figure 11 – Experiment C Results – No Major Selected.....	42
Figure 12 – Experiment C Results – Same Major Selected Twice.....	42
Figure 13 – Experiment C Results – Taken Course in Invalid Format.....	43
Figure 14 – Experiment D Results.....	44
Figure 15 – Experiment E Results .....	46
Figure 16 – Experiment F Results .....	48
Figure 17 – Experiment G Results.....	49
Figure 18 – Data Science B.S. Tracking Sheet 2027 [22] .....	57
Figure 19 – Data Science Major Requirements Input Sheet.....	58
Figure 20 – Data Science Major Super-Requirements Input Sheet .....	58
Figure 21 – Data Science Major Bucket Sheet .....	59

# 1 Introduction

With college education becoming an increasingly prevalent requirement to get a full-time job [1], universities nationwide are seeing a large influx of new students [2]. This increase in demand for a college education is partly responsible for the rapid growth of tuition costs, which continue to hit record highs year after year [3]. Every credit a student takes represents a financial cost, and every excess credit a student takes that is not part of an optimal “trajectory” (schedule of future courses to take to meet graduation requirements) adds an unneeded expense to payments on already record-high tuition. According to a July 2017 study done by “Completing College America”, students completing a four-year degree take an average of 13.5 more credits than necessary [4]. While a university may offer financial aid to help offset some of the cost of tuition, students who need additional time to graduate place their four-year scholarships in jeopardy, meaning those students and their families may need to pay most of if not all of the price of those credits completely by themselves. The increased financial difficulty for students graduating late includes more than just paying for additional credits, however. Students who graduate late are ineligible to begin working with the rest of their graduating class, and as such miss earning a full-time-level income for the time they must remain in school. All-in-all, these factors combine to create a total estimated cost of \$88,000 for delaying graduation from four to five years [5]. Additionally, even after controlling for institutional and demographic factors, students who graduate late are more prone to experience stunted mid-career salaries [6].

One of the main contributors to delays in graduation is students creating poor schedules with more courses than necessary for graduation [7]. Resources detailing degree requirements can be confusing and scattered [7], and academic advisors may be limited in their ability to assist due to a lack of tools that help them determine a student’s best path to graduation from where they are.

While course schedule optimization is not a new concept, students at Worcester Polytechnic Institute (WPI) face somewhat of a unique challenge that thus far scheduling optimization literature has failed to address - a four-term schedule. Instead of the typical “two-semester-per-year” structure featured by most four-year universities, WPI structures the majority of its undergraduate courses to each run in one “term,” or half a semester (about seven weeks). A student will typically take three courses per term for a total of 12 courses per year, meaning there are many more possibilities for both the courses a student can take and the terms during which a student can take those courses. Other confounding variables such as course offering patterns and off-campus projects further complicate the scheduling problem, while the lack of firm prerequisites increases the number of possible schedules. Mental health is of immense importance to students, faculty, and staff alike on WPI’s campus, and the sheer number of possibilities of feasible schedules can create mental strain and stress for many.

These difficulties with scheduling courses along with the sizeable benefits of graduating on time prove the need for a course scheduling optimization tool tailored to WPI’s unique course offering style. WPI’s current course registration system is Workday, which contains a degree audit tool accessible to students. While the degree audit system has some utility, there are also a number of inaccuracies, so the Registrar’s office must perform manual checks of every student’s degree audit before graduation. The accumulation of all of these challenges leads students to either manually chart their way by heuristically finding a better path, or simply taking other classes that are ultimately unnecessary.

Presenting students and advisors with a student’s optimal path to graduation (with the fewest number of additional credits) given the number and types of credits they have already earned in addition to other possible paths for them to compare gives students more control of their



four-year college plans and can help take some of the strain off academic advising resources. A course schedule optimization tool can offer considerable benefits to the mental well-being of students by alleviating stress. Laying out a student's entire trajectory can drastically reduce the mental strain of trying to craft schedules for years in advance, and presenting multiple feasible trajectories can eliminate the stress of creating multiple backup plans in case they do not get into a course on registration day. Furthermore, the definition of an "optimal" trajectory can be tailored to each individual student. For example, if many trajectories allow a student to graduate in the same amount of time, the student has the flexibility to choose the trajectory with the most courses of interest, the courses with the least amount of work outside the classroom, etc. In addition to benefits for students and advisors, an optimal scheduling tool can help the university free up space in classes for students who need those classes to graduate, and it allows the university to show prospective students that, should they choose to attend, they will be in full control of their scheduling education.

This project extends functionality for an existing course optimization tool tailored specifically to WPI's unique requirements. The existing mixed-integer linear optimization model served as a fully functional and efficient schedule generation tool that let students perform degree audits themselves and perform "what-if" analysis on future courses of interest. However, as the initial prototype was the product of an initial MQP, the scope of the prototype was incredibly limited – it only served two majors (as well as their double-major combination), and it could not correctly handle any independent study credits the student may have. Additionally, there was not an easy way for students to interact with the tool – the program ran solely on the command line, and each course a student had taken would have to be manually entered. We have expanded the tool's scope to include a variety of the most popular majors, double majors, and master's programs

at WPI. We have also created an interactive online user interface, where students can directly upload their taken courses and provide information about their independent study credits (if any), so those credits fit properly in their schedule.

In this report, expansions to utility and extensions to functionality and usability of a student trajectory optimization model are developed. After a review of the structure of WPI's degree programs and of trajectory optimization models developed for other universities, we explain the implementation of the existing mathematical model developed for WPI students and highlight the changes we made to that implementation to more robustly handle edge-case scenarios that may be encountered. We then define the scope of the additional programs we sought to implement to increase the tool's utility amongst the student body, and we showcase the improved, partially automated methodology we developed to implement them. Additional functionality is created in the form of a new degree program framework for master's degrees, and usability is expanded considerably with the creation of an online user interface for the model. We show how the creation of an online user interface allowed us to develop methodology for the handling of credits earned from independent study projects, and we highlight how this methodology can be expanded to handle other edge-cases within a degree. We then design, run, and analyze the results of several experiments which showcase the robustness of the model. Finally, the tool's current limitations are discussed, and future work to overcome challenges and to further improve functionality, utility, and usability of the model is outlined.

## 2 Background

In this section, we discuss the intricacies of course scheduling at WPI and how courses can be double counted between degrees. Additionally, we discuss other research that has been done in the area of course selection optimization.

### 2.1 The WPI Plan

Course scheduling at WPI is notoriously challenging because of the WPI Plan. The WPI Plan was designed under the principles of accountability and flexibility [8]. Rather than following a strict curriculum, students are given the freedom to explore courses that interest them and partake in project-based learning [9]. There are still core requirements for each major, as well as a series of university-wide requirements. All students must complete six courses in the humanities, satisfying depth (9 credits), breadth (3 credits), elective (3 credits), and capstone (3 credits) requirements [10]. Additionally, all students must complete an interdisciplinary Interactive Qualifying Project (IQP), typically during junior year, and a major-specific Major Qualifying Project (MQP), which is a senior capstone project. The university provides each major with a tracking sheet that displays these requirements. For an example tracking sheet, see Appendix A.

What makes the WPI Plan so challenging to schedule for, however, is the term system. Unlike most major universities, which organize their schedules into two 14-week semesters, WPI organizes its calendar into four 7-week terms. Students at other universities schedule 5-6 classes per semester, while WPI students schedule 3 classes per term. Although the term system increases flexibility, it also increases the number of course offering patterns that can occur, and with it, the number of feasible schedules. Some courses are offered as infrequently as one term every other year, while others have multiple sections every term. This can make planning quite difficult.

The WPI Plan also does not make use of firm prerequisites. While courses may have a recommended background consisting of courses and skills that would be helpful background knowledge, these are not required to be followed. This further increases the number of possible schedules a student can make, but many of these schedules would set the student up to struggle academically.

All of these features of the WPI Plan give students more options for courses to take in addition to more flexibility. However, the tradeoff of having few fixed degree requirements, four terms per year, and a lack of firm prerequisites, is that too much choice can be stress-inducing. When major requirements are only offered in one term, or certain upper-level electives are offered every other year, students need to know years ahead of time to plan their schedules around this. This is why a course-planning optimization tool would be especially beneficial for students in this system.

## 2.2 Double Counting

Developing a tool that can optimally sort courses being used to fulfill requirements is difficult, but this task becomes increasingly difficult for double majors. Double majors are possible due to the principle of double counting, in which one course may be used to satisfy either the same or two different requirements between degrees. For example, a WPI computer science and data science double major could double count ECON 1110 (Introduction to Microeconomics) towards the social science and policy studies requirement in both degrees. That same student could also double count CS 2223 (Algorithms), which is a mandatory course for data science, but a computer science elective.

In addition to double majors, WPI also offers a unique B.S./M.S. program that helps students graduate faster. In this program, students can take certain 4000-level undergraduate courses and graduate courses that double count towards both degrees. In most programs, up to 40% of the MS degree can be fulfilled by these courses. This 40% is equivalent to 12 graduate credits in most programs, which at \$1610 per credit in 2023-24, adds up to \$19,320 saved through optimal double counting [11]. Only a select few courses can be double counted, some of which are offered only in one term during alternate years, so planning should be done well in advance. The double counting system is further complicated by the fact that one graduate class and one undergraduate class do not carry the same type of credits. In most departments, a class taken at the graduate level is worth 3 graduate credits, which is equivalent to 4.5 undergraduate credits. Most of these classes are semester length, while a select few, such as OIE 559, follow this credit distribution pattern over just one term. Then there are departments, most prominently Aerospace Engineering, that offer graduate classes worth only 2 graduate credits in one term. The Aerospace B.S./M.S. program still requires 30 graduate credits for the master's degree, but students can only double count 8 graduate credits [17]. Keeping track of these different conversion systems across different departments makes WPI's double counting system very complex. Students can also be both double majors and B.S./M.S. students at the same time, but no triple counting is allowed. This system is complicated for students to understand and plan for, so a tool that helps solve this problem would be beneficial.

Academic advising is a helpful resource for students looking to double major or participate in a B.S./M.S. program, as advisors are generally capable of filling out tracking sheets and making sure students are on track. However, most advisors are not databases for class offering patterns and requirements, and they have more pressing tasks - such as teaching classes, doing research,

and helping students during office hours - to do instead of studying the course catalog. Students should not have to be that knowledgeable either just to graduate in an optimal timeframe. An optimization tool can help fill in these knowledge gaps and make generating a long-term feasible schedule much easier.

## 2.3 Other Trajectory Optimization Tools

Developing a trajectory optimization tool in itself is not a novel idea. There are a number of other papers that have been published on the subject, each design with its own strengths and weaknesses. Morrow, Hurson, and Sarvestani developed PERCEPOLIS, an “educational cyberinfrastructure that facilitates identification of a personalized education plan tailored to a student’s interests, conformant with all curricular mandates, and with consideration of the target graduation date” [12]. The tool has the potential to be a helpful tool for college students, but it was tested on a very small problem that does not compare to the scale and complexity of WPI’s.

In “A Mathematical Modeling Approach to University Course Planning,” Khamechian and Petering take a different approach - aiming to minimize semesters needed for degree completion instead of total number of courses. This model considers the scheduling patterns of courses and adheres to prerequisites when creating a recommended course plan for students to follow [13]. This model was tested on the University of Wisconsin-Milwaukee’s Industrial Engineering curriculum. One limitation of the model is how it handles broader requirements such as humanities and social science. At UW-Milwaukee, these requirements can be filled by many different electives, most of which have no prerequisites. Therefore, this model would not translate well to WPI mainly because of the complexity of the humanities requirement (with depth, breadth, capstone, and elective sub-requirements) and the crucial placement of one social science class, ID

2050. This class must be taken the term before a student leaves for IQP if they are assigned to an off-campus IQP center [14].

Wu and Havens also attempted to tackle this problem in a similar way in their paper “Modelling and Academic Curriculum Plan as a Mixed-Initiative Constraint Satisfaction Problem.” This model is broken down into two stages, a backtrack-based systematic search method and a systematic local search method [15]. The first produces an initial solution and the user provides recommendations to help construct the final plan in the second. This algorithm is a bit dated, as it was published in 2005, so newer advancements have been made.

One such advancement was made in Olabuyami’s “Application of Optimization Methods for Bachelor’s Degree Planning.” This model identifies the minimum number of credits needed for certain degrees with the university coursework at Southern Methodist University [16]. One unique thing about it is that it can optimally handle double majors. This paper demonstrates results for a management science degree paired with an economics degree. It also explores double counting courses for a 4+1 Bachelor’s/Master’s engineering program, which is similar to the B.S./M.S. Program at WPI. However, this model would not scale well in our case. In Olabuyami’s work, credits appear to be weighted the same at the undergraduate and graduate levels, so double counting only requires constraints to be modified [16]. This model also does not represent solutions as sets, so it does not accurately present the student with choices they may have.

All of these papers approach the course scheduling problem in different ways and do important work in the area of course scheduling and student trajectory optimization. However, none of them can be directly extended to address the complexities of planning WPI student trajectories, so a new approach is necessary.

## 3 Methods

In this section, we discuss the mathematical foundation for our tool, the scope of the project, the process of adding new programs to the tool, and various features of the implementation.

### 3.1 The WPI Student Trajectory Optimizer

In Academic Year '22-'23, a WPI Math and OIE double major named Lindsey Fletcher did an MQP report with Professor Andrew Trapp that developed a course schedule optimization tool that uses integer linear programming to optimally create student trajectories that adhere to WPI's requirements. While Fletcher's work only contained the hand-coded requirements for the Math and Industrial Engineering (OIE) majors as well as the Math and OIE double major, her model implementation is tailored specifically to WPI's courses and requirements and serves as an essential basis for our work [18].

#### 3.1.1 Requirements

Fletcher created a framework for representing degree programs comprised of three major elements – requirements, super-requirements, and buckets (or collections). A requirement is simply a set of courses from which a student must take a certain number of credits to graduate. While multiple requirements may contain the same course in their sets of courses, a course may be applied to at most one requirement – in other words, if a student takes a course to fill a requirement, that course may not be used to fill any other requirement within the same major. For example, a Data Science major at WPI has a Mathematical Science requirement of 15 credits. Calculus III (MA 1023) and Calculus IV (MA 1024) can count towards that requirement or the 36-credit Disciplinary



Electives requirement, but not both. If a student is a double major, a course may be used to fill one requirement in each major simultaneously (where applicable). This is referred to as double-counting.

Super-requirements refer to both “checkboxes” that may be filled by the courses taken across multiple requirements (indicating you must take “at least” a certain number of a certain type of credits) as well as major-wide “rules” that must be followed by all applicable requirements (indicating you may take “at most” a certain number of a certain type of credits). These are referred to as Type 1 super-requirements. For example, Data Science students at WPI must take at least 12 credits of 4000-level (or higher) courses related to their major, although they may choose to apply these credits to any combination of their Disciplinary Elective, Computer Science, and Mathematical Science requirements. As long as 12 or more 4000-level credits are taken between these three requirements, the super-requirement (“checkbox”) is fulfilled. Additionally, Data Science students may take at most 3 credits of 1000-Level Computer Science regardless of which requirement they satisfy with that course. This super-requirement represents a “rule” that within all requirements that accept 1000-Level Computer Science courses, no more than 3 credits may be present.

There are also Type 2 super-requirements. These generally reflect depth requirements where there are sub lists of courses that meet a requirement. The student must take a certain number of credits from one of the sub lists. Take for example, the Basic Science requirement for Computer Science students. There are four basic science departments - Biology, Chemistry, Geology, and Physics – and a student must take 9 credits worth of courses. To satisfy the depth super-requirement, the student must take 6 credits worth of courses from one department. The most prominent other example of these super-requirements is the humanities requirement, which is

required for all majors, but outside of that, they are somewhat rare. For an example of requirement and super-requirement structure, see Appendix B.

### 3.1.2 Buckets

Buckets (or collections) represent courses that fill a certain set of requirements and super-requirements. For example, courses that could be used to fill a Data Science student's Disciplinary Elective, Computer Science, or Mathematical Science requirements will be in one bucket together, while courses that may only fill the Disciplinary Elective or Computer Science (but not the Mathematical Science) requirements will be in another bucket together. While the buckets can be generated purely from the requirements and super-requirements and contain no new information, they offer some advantages to the model. By grouping courses that fill common requirements and super-requirements together, a single optimal solution can actually be a set of solutions. For example, if the model determines a student needs to take a course that exactly fills the Computer Science requirement and the 4000-Level super-requirement, it can present the user with the list of courses in the Computer-Science-and-4000-Level bucket in the solution instead of choosing just one course from that collection and filling it in for the user. This gives students choice in how they wish to go about completing their remaining degree requirements while still having the optimally fewest number of additional credits necessary for graduation.

Using buckets also drastically reduces the complexity of the model. Instead of having binary variables representing whether or not a course is being used to fill a requirement or super-requirement (meaning one variable per course, per requirement and super-requirement that course can fill), we can instead represent our decisions as how many courses from a certain collection will be used to fill each requirement. This grouping considerably reduces the number of decision

variables the model has, which offers large performance benefits in solve time. For a more detailed explanation on the creation and use of buckets, see Fletcher’s report [18]. For an example of a bucket sheet, see Appendix C.

### 3.1.3 Model Overview

Fletcher’s model consists of two stages, each of which has a different objective. In stage I, the model seeks to minimize the number of additional credits the student must take to graduate. In stage II, the model seeks to maximize the sum of the “choice weights” the user gives to each course without exceeding the minimum number of additional credits for graduation it found in stage I. “Choice weights” are designed to represent a student’s preferences regarding which courses they would like to take. Currently, students do not actually have control over the choice weights, and they are instead set by us in a way that maximizes student choice in the model output, though we would like to give users more control in the future. In Fletcher’s model, choice weights are assigned to buckets, although they could be extended to apply to individual courses. These choice weights are integer constants that have a lower bound of the number of courses in the bucket and no upper bound. The higher a bucket’s choice weight, the more likely it appears in the output. Because the stage II solve may not exceed the number of additional credits found in stage I, a student’s preferences may not always be reflected in the final solve should they require a non-optimal number of credits to be taken.

### 3.1.4 Penalizing Overflow

Our contributions consisted of identifying and resolving an important limitation of the output of Fletcher’s model. The problem was that if a student had taken more courses than

necessary, overflow courses would sometimes not get sorted into the unused courses section. If these courses were eligible to fill a requirement, they would get output in that section. This was a problem because it could have falsely led students into believing that they were using those extra courses to meet degree requirements, when in reality they are just unused credits.

It may seem like a strange edge case for someone who has already met all degree requirements to be using the tool, but it has other implications. It is specifically an issue for students whose program combinations are not represented by the tool. Take for example, a Data Science student with a minor in Mathematical Sciences. The minor needs 18 credits of math courses, and 9 of them can be double counted towards the major. Assume the student has taken courses that can only count towards the mathematical sciences requirement in the Data Science degree and no others within that program. Suppose the student has taken 7 total math courses – 2 courses counting towards just the major, 2 for just the minor, and 3 to double count for both. The math requirement for the major needs 5 courses. So ideally, only five math courses should show up under that requirement, and the other two should appear in unused courses. This way, it is clear to the student that those two classes are not being counted towards their major and are free to be used for their minor. If all 7 courses appeared in the math section, it would not be clear what is actually needed for the major. This issue would also affect students doing B.S./M.S. degrees in addition to minors. It was also important to fix for single and double majors so that they understand the degree requirements and are not confused by the extra courses in the output.

Understanding the following sets, parameters, and variables is necessary to understand the changes we made to Fletcher's model. All definitions were first defined in Fletcher's work, except for the variable  $z_r$  that we created to represent the number of overflow credits assigned to requirement  $r$  and the variable  $s$ , which is set equal to the first objective.

<b>Sets</b>	
$P$	Set of programs being evaluated, indexed by $p$
$M$	Set of buckets, indexed by $m$
$R_p$	Set of requirements $r$ that apply to program $p$

<b>Parameters</b>	
$\alpha(m)$	Number of credits for any course in bucket $m$
$\beta(r)$	Number of credits needed to satisfy requirement $r$
$\Gamma(m)$	Choice weight assigned to bucket $m$
$\varepsilon$	Small multiplier to reward overflow
$s^*$	Optimal value of the stage I solve

<b>Variables</b>	
$y_m$	Total number of untaken courses assigned from bucket $m$
$z_r$	Number of overflow credits assigned to requirement $r$
$s$	Continuous variable that represents the sum of the additional credits needed and the overflow penalties (if any)

Fletcher's stage I objective aims to minimize the sum of the products of the number of credits per course in each bucket and the number of untaken courses in each bucket. Our objective keeps that structure but adds an additional sum - the product of  $z_r$  and a small multiplier  $\varepsilon$ , which we set at 0.01. What this does is reward the model for declaring courses as overflow when possible. When we output the results of the solve to the user, we round down to not include any weight from  $z_r$  in the calculation of credits needed.

Fletcher	Kimball and Suyer
$\min \sum_{m \in M} \alpha(m)y_m$	$\min s = \sum_{m \in M} \alpha(m)y_m + \sum_{r \in R_p} \varepsilon * z_r,$ $\varepsilon > 0$

The stage II objective, where the goal is to maximize user choice through the sum product of choice weights and untaken, remains unchanged in our updated model. The mathematical function can be viewed below.

$$\max \sum_{m \in M} \Gamma(m)y_m$$

However, in Fletcher's model, there is a constraint that gets added in stage II to enforce that the maximum number of untaken credits should not exceed the total calculated in stage I. To get that number to match our stage I solve, we have to incorporate  $z_r$ . So, we add that same sum product from the stage I objective into this constraint and set the value equal to  $s^*$ . The reason our modified constraint is satisfied only at equality is because we need to enforce the same amount of overflow courses in the stage II solve. Otherwise, to maximize the choice weights present in the solve, the solver will try to select all applicable courses to a requirement. Note that Fletcher's  $z^*$  is equivalent to our  $s^*$  in the formulations below.

Fletcher	Kimball and Suyer
$\sum_{m \in M} \alpha(m)y_m \leq z^*$	$\sum_{m \in M} \alpha(m)y_m + \sum_{r \in R_p} \varepsilon * z_r = s^*, \quad \varepsilon > 0$

There is one additional constraint present in both stages of the solve that required the addition of  $z_r$ . This constraint exists to enforce that the necessary number of credits get applied to each requirement. For all requirements that apply to the entered programs, it checks that the sum

across all buckets of the number of credits for a course in bucket  $m$  multiplied by the total number of courses from bucket  $m$  getting applied to requirement  $r$  is at least the required minimum. In our case, we do not want it to exceed the required minimum, but we do need it to be possible in case a student has already taken a combination of courses that somehow forces extra credits to be needed. We subtract the overflow courses from the sum, and this ensures that any credits that can be designated as overflow do get designated as overflow.

Fletcher	Kimball and Suyer
$\sum_{m \in M_r} \alpha(m)x_{m,r} \geq \beta(r),$ $\forall p \in P, \quad \forall r \in R_p$	$\sum_{m \in M_r} \alpha(m)x_{m,r} - z_r = \beta(r),$ $\forall p \in P, \quad \forall r \in R_p$

Only the objective functions, the stage II constraint, and the requirement constraint are discussed in this section as that is where modifications were made. The complete models can be viewed in Appendix D (Stage I) and Appendix E (Stage II).

### 3.2 Scope of Implementation

While developing the tool to support every student at WPI would be ideal, the manual setup required for every major and double major combination, as well as every master's program and B.S./M.S. combination exceeded the timeframe of this project. As a result, we had to narrow the scope. We selected a set of programs that serve a large percentage of the WPI student population while demonstrating the full capabilities of the optimization tool. WPI's enrollment data for fall of 2023 (shown below in Figure 1) helped us identify the most popular majors [19]. By selecting six of the seven most popular undergraduate majors (Computer Science, Mechanical Engineering,

Biomedical Engineering, Robotics Engineering, Electrical and Computer Engineering, and Data Science), we can at least partially serve approximately 62.9% of the undergraduate population, as it is not clear how many of these students are double majors. This data also provides no insight into which double majors are the most popular, so we focused on implementing two combinations of these six that have similar content overlap, which we have seen to be common in our experiences (Computer Science/Data Science and Biomedical Engineering/Mechanical Engineering).

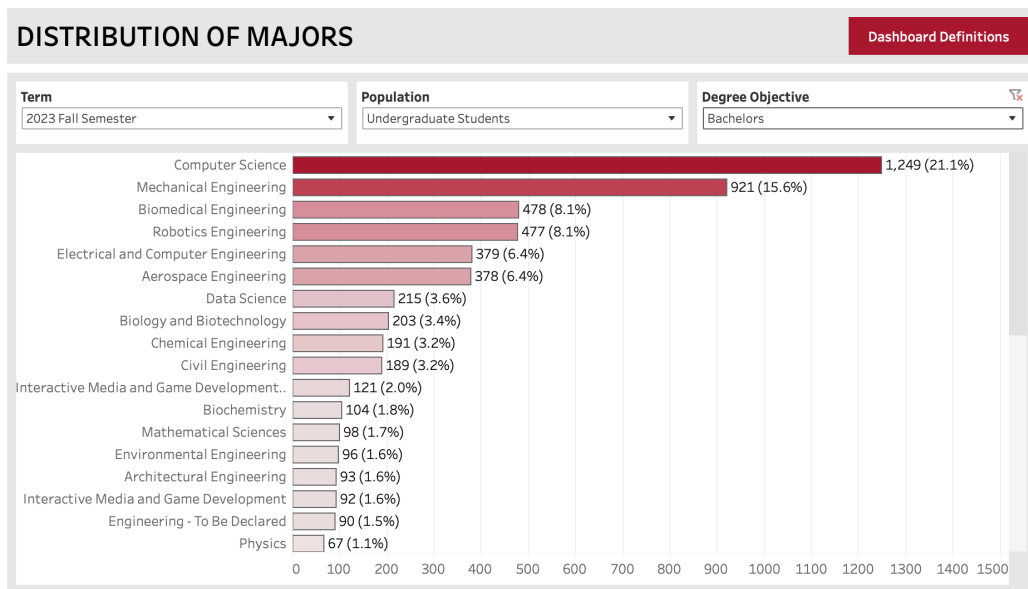


Figure 1 – Undergraduate Major Distribution [19]

Using data from the same source, we identified the three most popular graduate degrees (Robotics Engineering, Computer Science, and Data Science) and decided to implement those. By doing so, the tool can service approximately 35.3% of the graduate student population. We additionally wanted to implement a few B.S./M.S. programs, but we could not successfully implement within the timeframe of our project. Our attempt is discussed in greater detail in Appendix G, however. Successful implementation of these 11 programs will serve as a solid proof of concept for possible future adoption and expansion of the optimization tool.



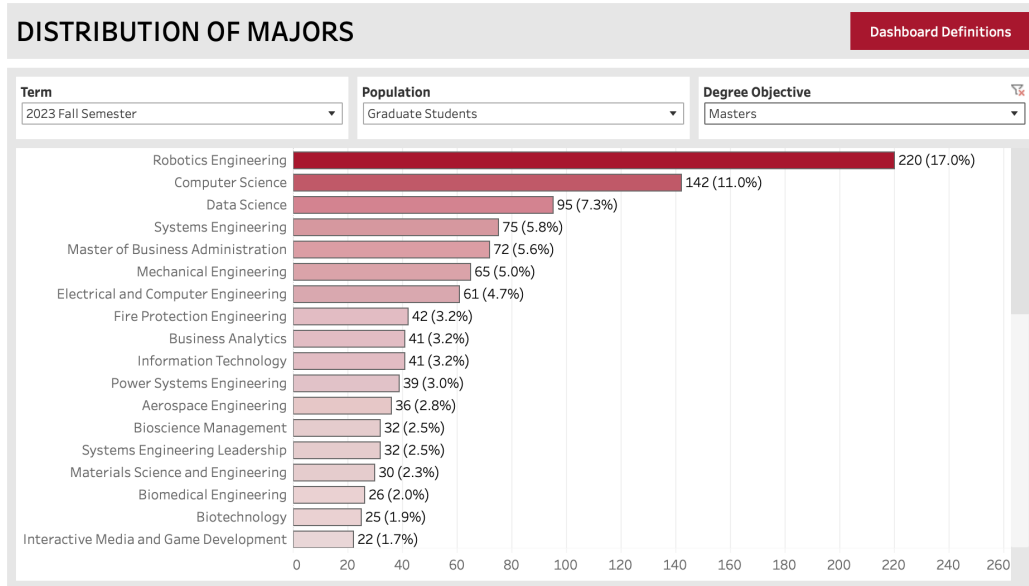


Figure 2 – Graduate Program Distribution [19]

### 3.3 Creation of New Programs

Adding a new program to the tool is a long process that is somewhat automatable, but still requires significant manual editing and testing. The process primarily involves collecting data (the requirements, super-requirements, and courses that satisfy them), sorting the acquired courses into buckets, and testing the output to ensure the user receives as much choice as possible and edge cases produce valid results.

#### 3.3.1 Data Acquisition

To get our 11 chosen programs implemented, the first step was acquiring data. Information about the requirements, super-requirements, and the courses that fill them was gathered primarily using the 2023-24 WPI undergraduate and graduate catalogs, and it was supplemented by the WPI

tracking sheets for students entering in 2023 [20, 21, 22]. There are minor differences between tracking sheets and catalogs of the last few years, such as slightly different sets of courses being applicable to certain requirements or differing amounts of credits needed for electives. We used the most recent version of the tracking sheet to service the largest possible number of students, as students are free to switch to any tracking sheet that was implemented after they enrolled.

These catalogs and tracking sheets also contain various errors and inconsistencies relating to course data, which added to the difficulty in its curation. New courses are also always being added that have yet to make it into the catalog, so the optimization tool can never be completely up to date. However, we attempted to make it as accurate as possible with the available data. Requirements, super-requirements, and the courses that fill them were compiled manually into an Excel sheet. What constitutes a requirement and super-requirement is also not defined by WPI, so we must develop that hierarchical structure separately for each program. We generally follow the outline of the tracking sheets, with sections of the sheet that have numbered slots for courses becoming requirements, and any additional text instructions becoming super-requirements. There are many exceptions to this, however, so we have to edit the structure until we are sure that all rules of the tracking sheets and catalog are being enforced.

### 3.3.2 Bucket Scraper

For the optimization tool to work, these courses have to be sorted into buckets (also called collections), which are specific to every program combination. Manually sorting courses into buckets is tedious and inefficient, so we developed a bucket-creating python script. The script identifies the desired program combination, then looks through the requirements and super-requirements that apply to that combination. It finds the list of courses that fill each requirement

and sorts the courses by the set of requirements and super-requirements they fill. This leaves us with buckets containing unique sets of courses that map to unique sets of requirements. For program combinations that contain data for both graduate and undergraduate classes, buckets are manually divided by credit amount. In these instances, it is the combination of courses and credits that uniquely maps to the set of requirements.

Buckets are identified with a unique bucket key, which serves as a selection ID in the model, and a text description that gets output to the user. Both of these are created manually. While the creation of the bucket key is unimportant and could be automated in the future, creation of the description is very important to the success of the tool. We use these descriptions to give the user as much choice as possible when selecting courses, so we as the creators have to try to imagine situations where these buckets will appear in the output and generate descriptions accordingly. Buckets that contain only a few courses for one or two requirements will have more specific descriptions. Take for example, a Computer Science major who needs a course to fill a spot in the math requirement and the probability super-requirement. The bucket that maps to this set of requirements contains two courses, MA 2621 and MA 2631, so the user must take one of the two. In this case the bucket description is very specific – “Probability – MA 2621 OR MA 2631.” On the other hand, buckets that are larger or can be used to satisfy more requirements need much broader descriptions, such as “Math Electives” or “Basic Science or Engineering.”

While this system generally works well, it is not perfect. There are plenty of cases where a bucket description needs to be narrow for one student’s case yet would be better being broad in another. Take the science depth super-requirement for the Computer Science major. Students are required to take three basic science courses to fill the science requirement, and two of them must be in the same discipline to fulfill the depth super-requirement. For a student using our tool who

is just starting at WPI, it would be better if the bucket description read “Basic science electives – BB, CH, PH, GE,” as they can take any science courses within those four departments. But for a student who has already taken a chemistry course (CH) and a physics course (PH), the output needs to reflect that their third science course must be in one of those two disciplines. So, the bucket that gets recommended will either say “Chemistry” or “Physics”, as both buckets are equally valid in this case. But the student just starting at WPI may get recommended three chemistry classes, when there are viable alternative schedules that contain no chemistry classes. While not ideal, this situation is a better alternative than leaving those buckets broad and having a student take a science class from three different departments, as this would leave the super-requirement unfulfilled and require them to take an extra course. So, while bucket descriptions can artificially restrict choice, they can lead to invalid courses being taken if they are too broad. For this reason, it is important that we pass some responsibility to the user to make sure they understand the limitations of the tool and can interpret the rules of the tracking sheets for themselves. For an example of the results of bucket creation, see Appendix C.

### 3.4 M.S. Implementation

While Fletcher’s implementation provided support for single and double majors, it did not include support for master’s programs. Master’s programs are of course fundamentally different from undergraduate programs in many ways. They do not share university requirements like undergraduate majors do (humanities, social science, IQP, MQP, and PE). Even though most programs require some kind of thesis or capstone, each program handles this differently. There are also no true free electives in most graduate programs. Undergraduate free electives enable students to take any course offered, but graduate free electives are instead generally limited to the discipline

of the program and are specifically defined in the catalog. Another big difference is the total number of credits of graduate programs. They generally require 30 graduate credits, which in most cases converts to 45 undergraduate credits as opposed to the standard 135 for undergraduate majors.

There were some significant edits needed in the source code to account for the differences between undergraduate and graduate programs. We first needed to implement a check to determine whether the user's entered program was a bachelor's or a master's, and then with this information, we excluded master's programs from university requirements, changed the total number of credits needed, removed free electives, and modified various calculations in the code to change the way the remaining credits are calculated.

The way requirements and super-requirements are set up for these programs is also very different. We implemented the undergraduate programs with a combination of many requirements and super-requirements. We had to implement our selected M.S. programs, however, with only one requirement, that being the 45 total credits needed. All additional rules are coded in as super-requirements. The reason for this is because of the B.S./M.S. program. Take the Data Science program as an example. A student needs 4.5 credits in the area of Data Analytics and Mining, and one of the courses in that area is CS 539 – Machine Learning [23]. This is a graduate course worth 4.5 credits, but there is also an undergraduate 3 credit version of this machine learning course – CS 4342. A B.S./M.S. student could double count the three-credit undergraduate version towards both degrees and still fulfill the Data Analytics and Mining requirement. For this reason, it is implemented as a super-requirement, requiring at least 3 credits, instead of exactly 4.5. Aside from providing flexibility for B.S./M.S. students in the future, this structure also provides flexibility in the number of electives for students on the thesis and non-thesis tracks.

### 3.5 Online User Interface

We wanted to make the model as accessible and user-friendly for students as possible so students can generate multiple “what-if” trajectories quickly in addition to the optimal trajectory for their current stage in their trajectory. This meant we did not want students to have to download our model and its dependencies to run it on the command line - instead, we developed a web-based user interface to serve as a user-friendly model frontend, which is shown below in Figure 3.

#### WPI Course Schedule Optimization Tool

Select your major:

Select your second major:

If you're pursuing a master's degree (either standalone or BS/MS), please indicate it here:

Enter your taken courses or upload your Academic Progress Excel sheet from Workday:

No file chosen

Input courses as a comma-separated list with an underscore in between the department code and course number.  
*Ex: MA\_1021, WR\_1010, ...*

If you are manually inputting courses, please omit your MQP and gym credits, but we will output them in your tracking sheet so you don't forget about them.

If you are uploading your Academic Progress, you can also manually input courses above to add to your Workday output (if you'd like to experiment with potential future courses).

Similarly, if you'd like any courses removed from your Academic Progress, enter them here:

**Instructions:**

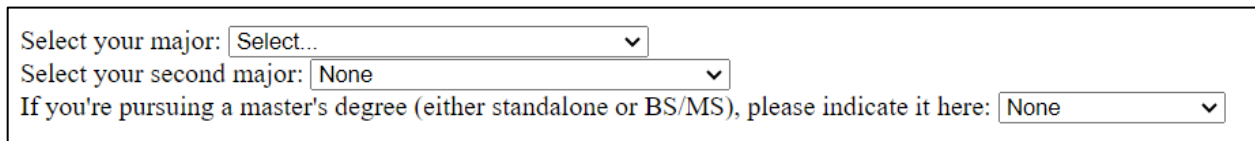
Select at least one major and input and/or upload your courses!

Figure 3 – Web User Interface for the Course Schedule Optimization Tool

The interface collects information from the user using HTML5 form elements (dropdown menus, file uploads, and free-text fields) and passes it to a backend built using Flask, which is a framework for web app development in Python [24]. The interface uses Flask to handle HTTP requests (i.e. the user’s input), process the user’s input to be passed to the backend Python model, and generate dynamic HTML content after receiving a solution from the model. CSS and/or JavaScript elements can be added in future versions of the interface to make it more visually appealing and to make its contents easier to understand and interact with. More text will also be added to provide additional detail on the usage of the interface, how the backend model functions,

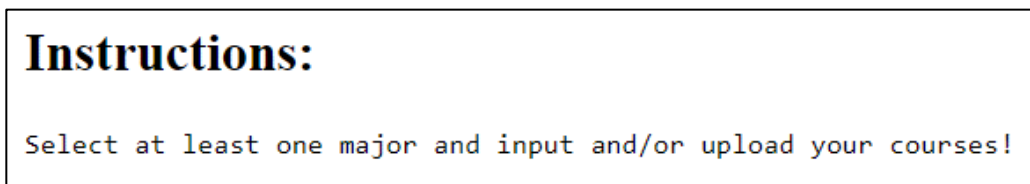
and the ways students can generate “what-if” trajectories containing hypothetical majors and taken courses.

To collect information about the user’s degree program, the interface has dropdowns that contain each bachelor’s and master’s program we have implemented. Users can select a single major, double major, or master’s degree program to generate a trajectory. Support is limited to the implemented double major combinations and does not include B.S./M.S. at this time. If the user enters a combination that is unsupported, or a malformed combination of supported majors (i.e. both majors are the same, or there is a second major selected without a primary major selected, etc.), the Instructions field at the bottom of the page will populate with a message informing users how to fix their input. Figures 4 and 5 below show the dropdowns in the degree program input section and the welcoming directions initially shown in the Instructions section before the user makes an unsupported or malformed input.



Select your major:    
Select your second major:    
If you're pursuing a master's degree (either standalone or BS/MS), please indicate it here:

Figure 4 – Degree Program Input Section of the Online User Interface



**Instructions:**  
Select at least one major and input and/or upload your courses!

Figure 5 – Instructions Section of the Online User Interface with Welcoming Instructions

To collect information about the courses the user has taken, the site has a free-text field where courses can be entered in addition to a field where an Academic Progress Excel sheet from Workday, WPI’s chosen course registration and degree auditing system, can be conveniently

uploaded. This Academic Progress sheet contains each course the student has earned credit for and each course the student is currently registered to take. Students can easily download this sheet from their Workday student portals and upload it unedited to the user interface, which will prompt a Python script on the backend to scrape the file and extract the courses in a format the model can parse. Personal identifying information is not included anywhere in the file or filename, and the only sensitive information included in this file is the grades the student earned in each course and their cumulative GPA, but protecting student privacy is still important. To do so, the only information we access in the file are the names of the courses, and the file is deleted from the server as soon as all courses from it are parsed. In addition to uploading their Academic Progress, students can still use the free-text course field to add potential courses they are interested in taking (that would not appear in their Academic Progress), and the model will treat them as taken. Similarly, if students wish to remove a course from their Academic Progress (perhaps they are anticipating receiving no credit for a course they are presently taking or are unsure about their future registrations), there is a separate free-text field for them to enter those courses. Figure 6 shows the course entry/upload section of the interface, which has instructions specifying the format for free-text course entries, tips on how the free-text fields can be used, and a button for the user to submit their input once they have finished.



Enter your taken courses or upload your Academic Progress Excel sheet from Workday:

No file chosen

Input courses as a comma-separated list with an underscore in between the department code and course number.

Ex: *MA\_1021, WR\_1010, ...*

If you are manually inputting courses, please omit your MQP and gym credits, but we will output them in your tracking sheet so you don't forget about them.

If you are uploading your Academic Progress, you can also manually input courses above to add to your Workday output (if you'd like to experiment with potential future courses).

Similarly, if you'd like any courses removed from your Academic Progress, enter them here:

Figure 6 – Course Entry/Upload Section of the Online User Interface

To begin, users first input information about the degree program for which they would like to generate a trajectory and then input or upload the courses they have taken. After selecting their degree program, users will then need to specify what courses they have taken. They can do so by manually inputting taken courses into the free-text field following the specified format and/or uploading their Academic Progress file from Workday. After a user submits their degree program of interest and their taken courses, the model runs on the backend and creates a text output with the optimized trajectory that is displayed to the user on a separate Results page. The output shows the student the total number of credits in their selected program and a breakdown of their broad categories, how many total credits they have taken, how many of those credits could be applied to their program, and how many were excess. Below that are some statistics about the runtimes of both stages of the model, and an explanation of which courses have been taken and which are prescribed by the model. The output then breaks down each requirement in the degree program, starting with requirements for all undergraduate majors (unless a master's degree program was selected) and then continuing to requirements specific to the selected program. The output also lists the courses that could not be applied to the program. Figure 7 shows the Results page with the text output of an undergraduate Computer Science major who has not yet taken any courses. The output on the Results page is in a single column, but the layouts of the visuals of the Results page will be condensed in this report to save space.



conclusion of the course. Should a student complete this coursework, the extra credit will be applied to their degree in the form of an independent study. Completing this coursework for double-countable courses can potentially give B.S./M.S. students enough credits to safely omit entire classes from their future schedules, meaning B.S./M.S. students are highly incentivized to pursue these extra credit options. As such, many B.S./M.S. students will have at least one independent study applied to their degrees from these double-countable courses. Being able to handle these credits in our model is critical functionality necessary to support B.S./M.S. programs in the future.

Even outside of the B.S./M.S. case, independent studies are not uncommon, so it is imperative to the accuracy of our tool that we not only apply these credits to their degree, but that we also understand exactly how and where in their schedule the student intends to apply those credits. Fortunately, before getting approved for an ISP, students must fill out a form provided by the Registrar to provide details about the intended application of those credits on their degree. This means a student should already know where and how they want their credits applied by the time they would appear in their Academic Progress Report. By designing a form with fields that are almost identical to those found on the form provided by the Registrar, we can accurately apply independent study credits in exactly the way the student intends, even if they have multiple ISPs with the same name.

To correctly apply ISP credits to a student's degree, we need that student to provide four pieces of information for each ISP they have in their course input – the required course in their degree these ISP credits are being directly substituted for (if applicable), the number of credits the ISP is worth, the degree requirement these ISP credits are being applied to (if any), and any degree super-requirements these ISP credits are being applied to. To collect this information, we created

an online form that the user of the online interface will be prompted to fill out for each ISP detected in their taken courses. The form indicates which ISP the user is currently providing information for, and it reminds the user that they should have the information to fill out the form on-hand since they were approved for the ISP by the Registrar. The form is comprised of a free-text field for the substituting course code (if applicable), a numeric field for the number of credits the ISP is worth, a dropdown box for which (singular) requirement the ISP is being applied to (if any), and a checkbox for any super-requirements the ISP is being applied to. The form page is reloaded for each ISP in the user's taken courses after the response for the previous ISP is stored. The ISP's course code is not needed for application of the credits to the degree, meaning ISPs with the same course code can be stored separately and applied to the degree in different ways. Figure 8 shows the online ISP form a user is redirected to upon detection of at least one ISP in their taken courses.

**We've detected that you have some Independent Study Project (ISP) credits!**

**We'll ask you to fill out this form for each Independent Study you've taken. This information is a subset of the information you've provided on WPI's ISP form when you applied for this credit.**

**Current Independent Study: CS\_4999**

Are you using this ISP to directly substitute for a course? If so, enter the course here:

How many credits is this Independent Study worth?

Please indicate which requirement you are fulfilling with this course, if any:

None

Please indicate which super-requirements you are fulfilling with this course, if any:

- ('DS\_4LDER', 'Social imps moved to normal req for DS due to weird overlapping rules with social science')
- ('DS\_BMP', 'Business Modeling and Prediction')
- ('DS\_CS\_1000\_L', 'at most 1 intro CS class')
- ('DS\_DAM', 'data access and management (databases) for a disciplinary elective')
- ('DS\_DMML', 'data mining and machine learning for a disciplinary elective')
- ('DS\_DPE', 'data privacy and ethics')

Submit

Figure 8 – Independent Study Form Example

Handling this information in the model depends on the information the user submitted. All ISP course codes are dropped from the taken courses before solving (as not to suggest to the model that the course is a three-credit free elective), but if the ISP credits are being directly substituted for a required course, the course code the user provided is appended to the list of taken courses. If this is not the case, the specific degree requirements and super-requirements the user indicated

have their total required credits subtracted by the number of credits the ISP is worth. This effectively applies the ISP directly to the degree in the correct place before the model generates an optimal trajectory with the remaining taken courses – reducing the number of credits required for these requirements and super-requirements allows the model to assign fewer credits to them in the same way that the normal application of taken courses to those requirements and super-requirements does.

The one exception to this is Type I super-requirements with an upper bound or “at most” designation. If for example, a student can take at most 3 credits from a bucket, and they take a 3 credit ISP that serves as an equivalent for one of those courses, we do not want that super-requirement to disappear. We still want to enforce that a student can take no more courses from that bucket. So instead, we change the upper bound to 0 (or more generally subtract the taken ISP credits from the original super-requirement credit total) and leave the super-requirement in place.

The main issue with this current setup is that after the solve has been performed, it may not be clear to the user where certain independent study credits got applied if they accidentally applied too many to a single requirement. Even if they were correctly applied to a super-requirement, the student may be confused because super-requirements are not displayed in the output and are instead only coded into the model. In the future, we would like to leave the user output notes to remind them where the ISP credits got applied. These output notes could also have additional features, such as reminding students of various choices for their depth requirements.

## 4. Experiments and Results

To demonstrate how the model generates trajectories for different degree types and handles edge cases, we have devised a set of experiments on example students at different points in their academic programs. We will be varying the degree program of choice and the courses taken from student to student to showcase a variety of possible inputs to the model.

### 4.1 Experimental Setup

The model retrieves the data for each degree program from two JSON files stored locally. All code for the model is written in Python, and the Python Linear Programming (PuLP) library is used to build and solve the model. The solver used by our PuLP implementation is the COIN-OR Branch and Cut (CBC) mixed-integer linear program solver developed by the COIN-OR project [25]. To run each experiment, we will be using the online user interface, which uses the Flask framework to connect the HTML frontend to the Python backend. We will primarily be using the manual course entry feature to create our experiments, although an Academic Progress Report serves as a basis for the experiment with Student B.

### 4.2 Experiments on Example Student Schedules

We will perform experiments on three sets of example students – single-major undergraduate students, double-major undergraduate students, and master’s students. While our aim is to showcase the robustness of the model including its resilience to error in edge-case scenarios, our experiments will not repeat the same exact edge-case premise between sets of students. For example, once we conduct an experiment to show the effect of our overflow penalty

modification on a single-major undergraduate student, we will not run experiments for the sole purpose of demonstrating this effect on double-major undergraduate students and master's students. The model's implementation, adjustments, and edge-case logic remain consistent for all degree program types.

To begin our experiments, we will first examine a variety of example schedules that showcase different points in a single-major undergraduate trajectory, along with some edge cases that may occur. We will introduce example students, explain where they are in their trajectory, and analyze how well the model handled their specific input.

#### 4.2.1 Experiment A – Common Use

Student A is an undergraduate Computer Science student who has completed some requirements, made progress towards completing other requirements, and has not yet started some requirements. This student has not inadvertently exceeded the number of credits that can be applied to any one requirement, and they do not have any independent study credits. Additionally, they have not made any errors in their degree program selection and course inputs when using the user interface. This student represents the case we expect to be most common among users of the model. As such, we expect the model to generate an optimal trajectory for this student without additional user input or edge-case handling required. Figure 9 shows the list of courses Student A has input as taken and where they get placed in the optimal trajectory. For comparison, an undergraduate Computer Science schedule for a student who has not taken any courses can be seen in Figure 7 in Section 3.5.





<<<<<<<<<< Credits >>>>>>>>>>>>	Social Science	Entrepreneurship and Innovation
147.0 TOTAL	GOV 2314 ID_2050	[Entrepreneurship and Innovation]
18 REMAINING	Free Electives	
6 Academic Courses	-----	
0.0 Free Electives	DS 4635	
9.0 MQP	MA 4631	
3.0 PE	PSY 1400	
129.0 TAKEN	Not used	
117.0 Applied	-----	
12.0 Excess	ECON 1110	
	SP 1523	
	EN 1000	
	DS 501	

Figure 10 – Experiment B Results

In Student B's case, they will have 147 credits by the time they complete their degree. The Entrepreneurship and Innovation Requirement is still unsatisfied, but they have taken four extra courses that show up as unused. It is evident from the "Not Used" section that Student B has taken more courses than necessary for the Social Science Requirement. It only needs 6 credits, but the student has taken 12 credits. Six of these credits correctly appear under the Social Science Requirement, three of these credits appear under free electives (PSY 1400) and the remaining three appear as overflow in the not used section (ECON 1110), as the objective function incentivized. This is the correct output in this scenario. Student B also has other unused courses that apply to different requirements, and these courses have correctly been designated as overflow as well.

### 4.2.3 Experiment C – Input Error

Student C is an undergraduate Computer Science student who has made several errors in their model input. In the user interface, this student first forgot to select a major, so they receive an error message (shown in Figure 11) informing them to select a primary major.

Select your major:

Select your second major:

If you're pursuing a master's degree (either standalone or BS/MS), please indicate it here:

Enter your taken courses or upload your Academic Progress Excel sheet from Workday:

No file chosen

Input courses as a comma-separated list with an underscore in between the department code and course number.  
*Ex: MA\_1021, WR\_1010, ...*

If you are manually inputting courses, please omit your MQP and gym credits, but we will output them in your tracking sheet so you don't forget about them.

If you are uploading your Academic Progress, you can also manually input courses above to add to your Workday output (if you'd like to experiment with potential future courses).

Similarly, if you'd like any courses removed from your Academic Progress, enter them here:

**Instructions:**

Please select a primary major!

Figure 11 – Experiment C Results – No Major Selected

Student C then attempts to fix this error, but they accidentally select Computer Science as their secondary major without choosing a primary major. Because they are still missing a primary major, they receive the same error message. To try to fix this, Student C selects Computer Science as the primary major, but forgets to deselect it as the secondary major. They receive a new error message, shown in Figure 12.

Select your major:

Select your second major:

If you're pursuing a master's degree (either standalone or BS/MS), please indicate it here:

Enter your taken courses or upload your Academic Progress Excel sheet from Workday:

No file chosen

Input courses as a comma-separated list with an underscore in between the department code and course number.  
*Ex: MA\_1021, WR\_1010, ...*

If you are manually inputting courses, please omit your MQP and gym credits, but we will output them in your tracking sheet so you don't forget about them.

If you are uploading your Academic Progress, you can also manually input courses above to add to your Workday output (if you'd like to experiment with potential future courses).

Similarly, if you'd like any courses removed from your Academic Progress, enter them here:

**Instructions:**

Please select two different majors or select "None" if you don't have a second major.

Figure 12 – Experiment C Results – Same Major Selected Twice

Notice that Student C has also listed a course twice in the input and has entered a course in the incorrect format. Listing the same course twice and entering a course for removal are already handled intrinsically by the model and user interface. The model creates variables using the course code as a unique identifier for each, so if a course is listed twice, it instantiates a variable for that course and then immediately overwrites that blank instantiation with a new instantiation of a variable with the same name. The user interface cannot process incorrect course formatting at this time, however, so Student C receives the error message shown below in Figure 13.

Select your major:

Select your second major:

If you're pursuing a master's degree (either standalone or BS/MS), please indicate it here:

Enter your taken courses or upload your Academic Progress Excel sheet from Workday:

No file chosen

Input courses as a comma-separated list with an underscore in between the department code and course number.

*Ex: MA\_1021, WR\_1010, ...*

If you are manually inputting courses, please omit your MQP and gym credits, but we will output them in your tracking sheet so you don't forget about them.

If you are uploading your Academic Progress, you can also manually input courses above to add to your Workday output (if you'd like to experiment with potential future courses).

Similarly, if you'd like any courses removed from your Academic Progress, enter them here:

### Instructions:

Taken course with an invalid format detected - please format course codes according to the example above.

Figure 13 – Experiment C Results – Taken Course in Invalid Format

Furthermore, this student has indicated that they wish to remove a course from their Academic Progress, even though no file was uploaded. The user interface requires an Academic Progress file to be uploaded before removing courses from the taken courses list, because if no file is uploaded, this means the user is solely entering their taken courses manually, which means the user can simply omit any courses they would wish to delete from their entry. The tool correctly

catches Student C’s errors and provides helpful instructions so that the student understands how to fix the input.

#### 4.2.4 Experiment D – Independent Studies

Student D is an undergraduate Computer Science student who has completed four three-credit independent study projects, two of which have the same course code. The student enters that they would like to count nine of these credits towards the Science and Engineering requirement, even though it only requires six credits to be completed. The student intends to count the other three credits towards the Statistics requirement. Figure 14 shows the free elective and Computer Science sections of Student D’s output trajectory.

Computer Science	Free Electives
CS Core	[FREE ELECTIVE]
[CS 3043]	[FREE ELECTIVE]
[CS 4000+ Level Electives]	Not used
[CS 4000+ Level Electives]	-----
[CS 4000+ Level Electives]	No Unused Courses
[CS 4000+ Level Electives]	
[CS 4000+ Level Electives]	
[CS Electives]	
[CS Electives]	
[CS Electives]	
[CS Electives]	
[CS Electives]	
[Design - CS_3041, CS_3431, CS_3733, CS_4233]	
[Object-Oriented Programming - CS_2102 OR CS_2103]	
[Systems - CS_3013, CS_4513, CS_4515, CS_4516]	
[Theory - CS_3133, CS_4120, CS_4123, CS_4533, CS_4536]	
General Math	
[1000 Level Math]	
[1000 Level Math]	
[1000 Level Math]	
[1000 Level Math]	
[Math Electives]	
General Science	
[Biology]	
[Biology]	
[Biology]	
Probability	
[Probability - MA_2621 OR MA_2631]	

Figure 14 – Experiment D Results

In Student D's results, we observe that the model has assigned six of the Science and Engineering credits to that requirement, which removes it from the model entirely. Also notice that the model did not ignore the extra three ISP credits, even though they cannot be assigned to that requirement. We have no other information regarding which requirements these credits can be assigned to, so the model adds them to the degree as a free elective and reduces the number of credits in the free elective output from 9 to 6. The model also correctly assigned three ISP credits to the three-credit Statistics requirement and removed it from the output.

#### 4.2.5 Experiment E – Double Major

Student E is an undergraduate Math + Operations and Industrial Engineering double major who has completed some requirements, made progress towards completing other requirements, and has not yet started some requirements. Like Student A, Student E has not inadvertently exceeded the number of credits that can be applied to any one requirement, and they do not have any independent study credits. Additionally, they have not made any errors in their degree program selection and course inputs when using the user interface. Student E also has not taken any courses that interfere with the optimal overlapping of credits between both majors. This student represents the case we expect to be most common among users of the model with double majors. Figure 15 shows Student E's output trajectory.

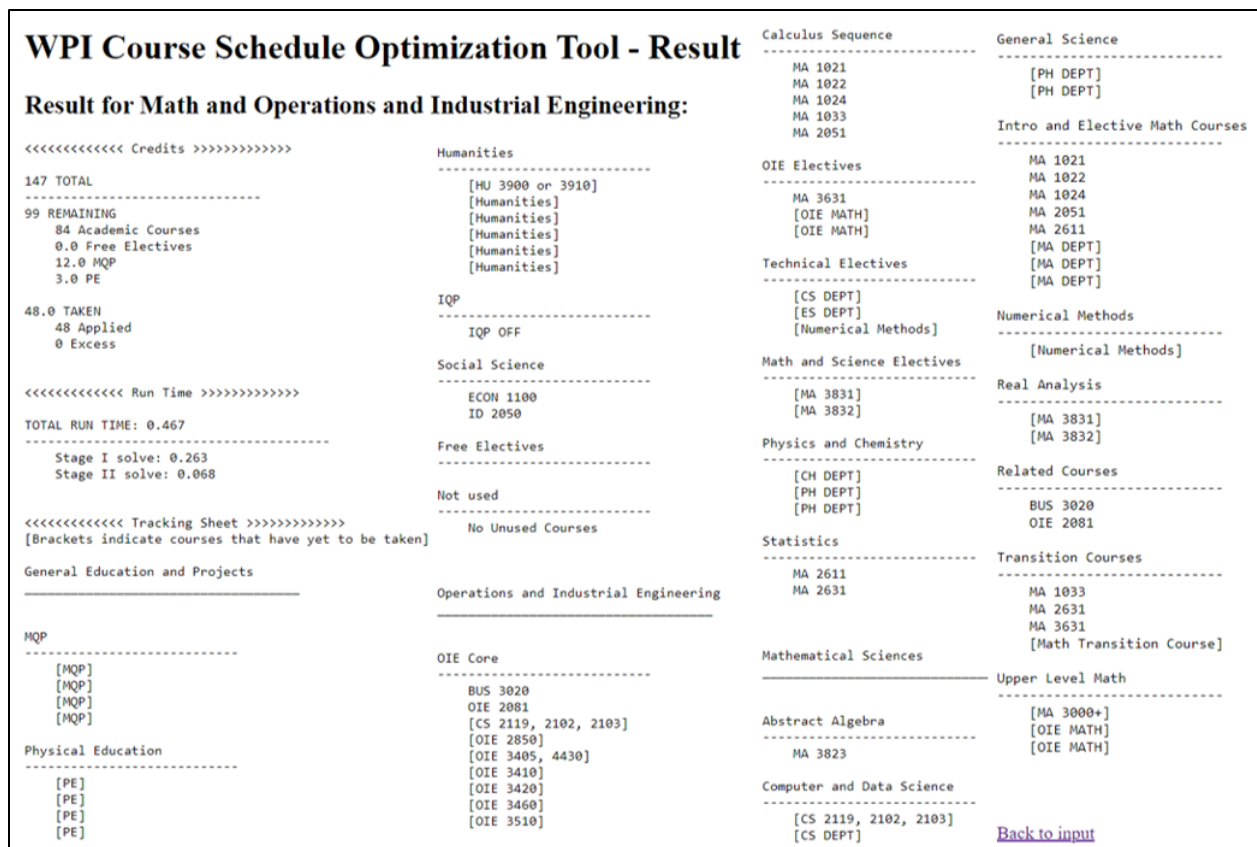


Figure 15 – Experiment E Results

We observe that the model correctly calculated the minimum total number of credits needed to complete the double major with courses that optimally fill the requirements of both degrees at once, and because all of Student E’s taken courses coincide with this calculated optimal overlapping, their output trajectory does not exceed the minimum total credits needed.

#### 4.2.6 Experiment F – Suboptimal Double Major

Student F is an undergraduate Math and Operations and Industrial Engineering double major who has completed some requirements, made progress towards completing other requirements, and has not yet started some requirements. Like Students A and E, Student F has not inadvertently exceeded the number of credits that can be applied to any one requirement, and they

do not have any independent study credits. Additionally, they have not made any errors in their degree program selection and course inputs when using the user interface. However, unlike Student E, Student F has taken a course that does not overlap between both majors in a slot where overlapping is necessary to achieve the optimal minimum number of credits for the program. Specifically, the Math + Operations and Industrial Engineering double major can be completed in as little as 147 credits if the student does not take courses that only can be applied to one of the majors and not the other. For example, Biology courses are applicable to the Math major, but not Operations and Industrial Engineering (outside of a free elective, of which none are available in a double major). Student F has taken one Biology course, but to optimally overlap between both majors, they should have taken a Chemistry or Physics course instead, since that could be applied to the Science requirements of both majors at once. Figure X shows Student F's taken courses (which are the same as Student A's) and their output trajectory.





### 4.2.7 Experiment G – Master’s Student

Student G is a master’s-level Computer Science student who has completed some requirements, made progress towards completing other requirements, and has not yet started some requirements. Like Students A, E, and F, Student G has not inadvertently exceeded the number of credits that can be applied to any one requirement, and they do not have any independent study credits. Additionally, they have not made any errors in their degree program selection and course inputs when using the user interface. This student represents the case we expect to be most common among users of the model pursuing master’s degrees. Figure 17 below shows the results for Student G’s degree optimization.

```
<<<<<<<<<<< Credits >>>>>>>>>>>>>>
45.0 TOTAL
-----
36.0 REMAINING
  36 Academic Courses
  0.0 Free Electives
  0.0 MQP
  0.0 PE

9.0 TAKEN
  9.0 Applied
  0.0 Excess

<<<<<<<<<< Run Time >>>>>>>>>>>>>>
TOTAL RUN TIME: 0.086
-----
  Stage I solve: 0.052
  Stage II solve: 0.016

<<<<<<<<<< Tracking Sheet >>>>>>>>>>>>>>
[Brackets indicate courses that have yet to be taken]

Computer Science Masters (MSCS)
-----

Total MS Credits Req
-----
  CS 539
  CS 542
  [Algorithms: CS_504, CS_584, CS_5084]
  [Elective or Thesis]
  [Elective or Thesis]
  [Elective or Thesis]
  [Electives]
  [Electives]
  [Systems or Networks: CS_502, CS_513, CS_528, CS_529, CS_533, CS_588, ECE_581, CS_535, CS_577, ECE_537]
  [Theory: CS_503, CS_521, CS_559, CS_5003]

Not used
-----
  No Unused Courses
```

Figure 17 – Experiment G Results

We can see that the output for a master's program does not contain the shared requirements across undergraduate degrees, and that the total number of credits has been reduced to 45. In Student G's case, we can see that they have so far completed 9 out of the 45 credits needed for the degree. For Student G's specific program (M.S. in Computer Science), the output also makes it clear that the student has the option to follow a thesis or non-thesis path for their degree (students generally decide by the end of their first semester). This makes it easier on our end to implement master's programs, as we do not have to split the different tracks many of them offer into different programs. Additionally, the output helps Student G recognize that they should be taking a course to fill each of the Algorithms, Systems or Networks, and Theory super-requirements.

## 5. Conclusion

In this section, we discuss the overall successes of the project while addressing the limitations it currently faces. Additionally, we detail our ideas for future extensions.

### 5.1 Discussion

This project expanded the functionality, utility, and usability of an existing WPI course optimization tool. We expanded the set of degree programs for which the model can generate trajectories, focusing on the ones that service the largest number of students, and extended the source code to handle master's programs. We also made it easier to add support for more programs in the future by automating part of the process of bucket creation and detailing the process behind adding a new program. Our mathematical model, which extends Fletcher's work, provides as much choice to the user as possible and penalizes adding too many credits to a requirement so that users can better understand the degree requirements. In combination with our online user interface, the model successfully handles a variety of use cases, as evidenced through our experiments representing students at different points in their academic careers. With the independent study form and built in detection for common user errors, the tool is quite robust and can handle a variety of edge cases. Of course, there are almost certainly edge cases we have not considered, and there are numerous limitations of the model that we already recognize as such, including the limited scope, display of only a single optimal solution, lack of cross restrictions (discussed further in Appendix F), and some missing edge cases for the humanities requirement.

Even with these limitations, we still believe in the educational value of our tool as a supplement to academic advising and the incomplete Workday degree auditing system. We also still believe that the potential for a "what-if" analysis is very valuable for helping students

(especially double majors) deal with the complexities of the WPI plan and that our tool has the potential to save unnecessary tuition expenses by reducing the extra courses they take. Because the tool is only intended to be a supplement to the already available resources, it is important that the users understand its limitations and be able to interpret the rules of the tracking sheets for themselves. There are also many possible future extensions that could increase the serviceability of the tool, with the most prominent being the addition of B.S./M.S. programs; we also have other ideas worth future consideration, including the incorporation of time-sensitive scheduling data, an AI chatbot to help increase user choice, and stochastic scheduling models.

## 5.2 Limitations

While we believe our optimization tool is of value to the WPI student population, we acknowledge that there are a number of limitations. The first and most obvious would be the limited scope. With only the most popular majors and a select few popular master's programs and double majors, it does not service the entire student population. There is also no current support for minors or B.S./M.S. programs, though we do plan to add this in the future.

The optimization tool also requires periodic maintenance to stay relevant. Degree programs can change between academic years, as new classes get added and requirements are altered. This means the tool may require significant updates, depending on how much is changed between years, to be an accurate tool for the students using it.

The other big limitation is that the solution presented to the user is a single credit minimizing solution. The bucket system is designed to make the user aware of the choices they have, but even that does not capture all of the choices the user has. It is therefore critical to the success of our optimization tool that users understand this concept. Additionally, some users may

be unhappy with the large amount of choice provided in the output. There are likely some who wish we would give them a strict schedule to follow to reduce the amount of time spent on schedule planning.

We also understand that the tool is more helpful for double majors trying to figure out the minimum number of credits that need to be taken to finish both degrees, than it is for single majors who just want to fill out the tracking sheet. The tool also helps users gain a broad understanding of what to take but provides no guidance on when to take the recommended courses. Incorporating scheduling data in some capacity in the future would certainly improve the tool.

There are also some technical limitations present in the model that exist because of the complexity of the data. Small things such as the university residency requirement, the limit on AP credits that can be applied to the Humanities requirement, and the ability to override the Humanities Capstone by taking six foreign language classes are absent. And although WPI does not have strict pre-requisite requirements at the undergraduate level, there are recommended background courses, yet our recommendations do not account for this in any way. Our model also does not enforce cross restrictions, which occur when courses with different codes cannot both be taken by the same student for credit. This has much bigger implications for the future of our tool, which is why Appendix F is devoted to further explanation of this topic. Our tool was designed as a supplement to academic advising and student understanding of degree requirements. We developed it with the intention of helping students save money and time, and in spite of its limitations, it can still accomplish that goal.

With all these limitations, our tool cannot serve as a substitute for academic advising or for student understanding of degree requirements, but rather a supplement that can make scheduling easier and potentially help students save money and time.

## 5.3 Future Work

The first extension of this project will likely be to implement B.S./M.S. programs, since planning the two degrees simultaneously and optimizing the double counting between them is a difficult task for students. Our original target scope for this project included the implementation of two of these programs – one with matching undergraduate and graduate programs (Data Science B.S./M.S.) and one without (Computer Science B.S./Data Science M.S.). Unfortunately, implementing these proved to be much more challenging and time-consuming than we had initially accounted for. A more detailed explanation of our approach can be found in Appendix F.

Aside from adding support for B.S./M.S. programs, there are numerous other additions that could improve the project in the future. The most obvious would be expanding the scope – adding more majors and double majors, master’s programs, and minors to service more of the WPI student body. We could also implement tracking sheets from different years instead of only using the most recent, to ensure students of all grade levels can use the tool.

It would also be good to give more choice to the user, perhaps by allowing them to set multipliers on the choice weights that influence bucket selection. As an example, a Data Science student could indicate that they do not like computer science classes and would prefer to take math and business electives where possible. We could then increase the choice weights on those buckets so the student gets recommended more classes they would be interested in. Another way to increase user influence on the results would be to process user text data. Users could communicate with a pre-trained AI that asks questions about what types of courses they would like to take. The AI could also use the courses the student has taken and the grades the student received to make additional recommendations on what courses students should take within the selected buckets.

To improve recommendations further, we could also incorporate WPI OSCAR data. This data is collected from voluntary student course report surveys that are sent out near the end of each term, and it serves as a gauge of course quality and difficulty. The surveys ask students to rate the course and the professor from 1-5 on various metrics such as overall quality, amount learned, and amount of homework [26]. We could use this to influence bucket selection, perhaps by computing an average difficulty rating for each bucket and using that as a tiebreaker when two buckets are equally valid in a solution. This data does suffer from participation bias, but there are many ways it could help further develop the optimization tool down the line.

Another extension of the project would be to further develop the user interface. With more time, we could better customize the output that is displayed to the user, such as providing information about specific cross restrictions the user should watch out for based on their taken courses. It would also help the user understand the limitations of the model if we could develop a way to indicate where there is room for more choice in their schedule, potentially highlighting buckets that could be replaced by others. Another important addition would be a channel for user feedback. This would help us identify missing courses or bugs in the code, as well as which program combinations people want to see implemented. It would also be helpful to hear about what specific aspects of the tool are most useful to people. If it helps double majoring students optimize double counting more than any other students, we could use that feedback to implement more double majors that students request.

Much further down the road, there might be some way to incorporate time sensitive data into the optimization tool. Using historical course offering data, we could develop a stochastic probability-based model that checks the feasibility of finishing the courses you have left within a designated amount of time successfully. This would help address one of the main limitations of

the tool, as we could then help students know both what to take and when to take it. There are many possibilities for the future of this course optimization tool, and its potential to help the WPI community is exciting.



# Appendices

## Appendix A – Data Science B.S. Tracking Sheet

### DATA SCIENCE MAJOR Program Tracking Sheet Effective for students entering AY 2023-2024

Name:	Class Year:
Advisor:	2 <sup>nd</sup> Major:

**NOTES:** Minimum total academic credit = 15 units  
Residency Req.: Min. of 8 units must be completed at WPI

**HUMANITIES AND ARTS (6/3 units)**  
All 5 HUA courses must be completed before beginning the Inquiry Seminar or Practicum.

Depth Component				
Students must complete at least three thematically-related courses prior to the culminating Inquiry Seminar or Practicum in the same thematic area. At least one of the three courses should be at the 2000-level or above.				
	Course	Term	Grade	Units
1				1/3
2				1/3
3				1/3
4	HU 3900 or HU 3910			1/3
Breadth Component				
Students must take at least one course outside the grouping in which they complete their depth component. To identify breadth, courses are grouped in the following manner.				
i. art/art history, drama/theatre, and music (AR, EN/TH, MU);				
ii. foreign languages (AB, CN, EN, GN, SP);				
iii. literature and writing rhetoric (EN, WR, RH);				
iv. history and international studies (HI, HU, INTL);				
v. philosophy and religion (PY, RE).				
Exception: May take all six courses in a foreign language				
5				1/3
Humanities Elective				
6				1/3

**WELLNESS AND PHYSICAL EDUCATION (4 WPE classes = 1/3 unit)**

7				1/12
				1/12
				1/12
				1/12

**SOCIAL SCIENCE (2/3 unit)** ECON, ENV, GOV, PSY, SD, SOC, SS, STS and ID2050

8				1/3
9				1/3

**THE INTERACTIVE QUALIFYING PROJECT (3/3 unit)**

10				1/3
11				1/3
12				1/3

**FREE ELECTIVES (3/3 unit)**

13				1/3
14				1/3
15				1/3

**DATA SCIENCE CORE (3/3 unit)**

16	DS 1010			1/3
17	DS 2010			1/3
18	DS 3010			1/3

**DISCIPLINARY FOUNDATION COURSES (10/3 units)**

**COMPUTER SCIENCE (3/3 unit) (Note 1)**

19	CS 2223			1/3
20				1/3
21				1/3

**MATHEMATICAL SCIENCES (5/3 unit) (Note 2)**

22	MA 2611			1/3
23	MA 2612			1/3
24	MA 2071 or 2072			1/3
25				1/3
26				1/3

**BUSINESS COURSES (2/3 unit) (Note 3)**

27	BUS 2080 or OIE 2081			1/3
28				1/3

**DISCIPLINARY ELECTIVE COURSES (11/3 units) (Note 4)**

29				1/3
30				1/3
31				1/3
32				1/3
33				1/3
34				1/3
35				1/3
36				1/3
37				1/3
38				1/3
39				1/3

**DATA PRIVACY AND ETHICS (1/3 unit)**

40				1/3
----	--	--	--	-----

**SCIENCE (2/3 unit)**

41				1/3
42				1/3

**MQP (3/3 unit)**

43				1/3
44				1/3
45				1/3

**Note 1:** Two of the courses must be chosen from CS 1004, CS 1101, CS 1102, CS 2102, CS 2103, CS 2119, or from CS electives listed in the course catalog.

**Note 2:** Two of the courses must be chosen from MA 1020, MA 1021, MA 1022, MA 1120, or from MA electives listed in the catalog.

**Note 3:** One of the courses must be chosen from OBC 1010, ETR1100, MIS 3010, or ETR 3633

**Note 4:** Disciplinary elective courses can be found in the catalog. Electives must include at least 4/3 at the 4000 level or higher and at least one course in each of the categories below:

- Databases (CS 3431, CS 4432, MIS 3720, CS 4433/DS 4433)
- Data mining/machine learning (CS 4445, CS 4342/DS 4342)
- Business modeling and prediction (MIS 4084, OIE 4430)

Figure 18 – Data Science B.S. Tracking Sheet 2027 [22]

# Appendix B – Data Science Major Requirements Sheet

	A	B	C	D	E
1	Program Key	Req Key	Credits	Req Description	Courses that fill req
36	DS_MAJOR	DS_C	9	Core	['DS_1010', 'DS_2010', 'DS_3010']
37	DS_MAJOR	DS_DE	36	Disciplinary Electives	['CS_2022', 'MA_2201', 'CS_2301', 'CS_2303', 'CS_3041', 'CS_3043', 'CS_3133', 'CS_3431', 'CS_3733', 'CS_4120', 'CS_4233', 'CS_4241', 'CS_4341', 'CS_4342', 'CS_4432', 'CS_4433', 'DS_4433', 'CS_4445', 'CS_4804', 'BCB_4004', 'MA_4603', 'CS_4032', 'MA_3257', 'DS_4635', 'MA_4635', 'MA_1023', 'MA_1024', 'MA_1033', 'MA_1034', 'MA_2051', 'MA_2073', 'MA_2210', 'MA_2431', 'MA_2621', 'MA_2631', 'MA_3231', 'MA_3233', 'MA_3627', 'MA_3631', 'MA_4213', 'MA_4214', 'MA_4222', 'MA_4235', 'MA_4237', 'MA_4631', 'MA_4632', 'MIS_3720', 'MIS_3787', 'MIS_4720', 'MIS_4741', 'MKT_3650', 'OIE_3460', 'MIS_4084', 'OIE_4430', 'CS_534', 'CS_539', 'CS_542', 'CS_548', 'CS_528', 'CS_582', 'CS_583', 'MA_540', 'MA_541', 'MA_543', 'DS_502', 'MA_542', 'MA_554', 'MIS_571', 'CS_561', 'CS_585', 'DS_503', 'DS_541', 'CS_541', 'CS_586', 'DS_504', 'MIS_584', 'MIS_587', 'MKT_568', 'OIE_559']
38	DS_MAJOR	DS_MS	6	Mathematical Sciences	['MA_1020', 'MA_1021', 'MA_1022', 'MA_1120', 'CS_2022', 'MA_2201', 'MA_4603', 'CS_4032', 'MA_3257', 'DS_4635', 'MA_4635', 'MA_1023', 'MA_1024', 'MA_1033', 'MA_1034', 'MA_2051', 'MA_2073', 'MA_2210', 'MA_2431', 'MA_2621', 'MA_2631', 'MA_3231', 'MA_3233', 'MA_3627', 'MA_3631', 'MA_4213', 'MA_4214', 'MA_4222', 'MA_4235', 'MA_4237', 'MA_4631', 'MA_4632', 'DS_502', 'MA_543', 'MA_542', 'MA_554', 'MA_540', 'MA_541']
39	DS_MAJOR	DS_EI	3	Entrepreneurship and Innovation	['BUS_1010', 'ETR_1100', 'BUS_3010', 'ETR_3633', 'OBC_1010', 'MIS_3010']
40	DS_MAJOR	DS_BA	3	Business Analysis	['BUS_2080', 'OIE_2081']
41	DS_MAJOR	DS_STATS	6	Applied Statistics	['MA_2611', 'MA_2612']
42	DS_MAJOR	DS_LA	3	Linear Algebra	['MA_2071', 'MA_2072']
43	DS_MAJOR	DS_ALGOS	3	Algorithms	['CS_2223']
44	DS_MAJOR	DS_NES	6	Natural or Engineering Sciences	['AE_DEPT', 'AREN_DEPT', 'BB_DEPT', 'BME_DEPT', 'CHE_DEPT', 'CE_DEPT', 'CH_DEPT', 'ECE_DEPT', 'GE_DEPT', 'ME_DEPT', 'PH_DEPT', 'RBE_DEPT']
45	DS_MAJOR	DS_CS	6	Computer Science	['CS_1004', 'CS_1101', 'CS_1102', 'CS_2102', 'CS_2103', 'CS_2301', 'CS_2303', 'CS_2119', 'CS_3041', 'CS_3133', 'CS_3431', 'CS_3733', 'CS_4120', 'CS_4233', 'CS_4241', 'CS_4341', 'CS_4342', 'CS_4432', 'CS_4433', 'CS_4445', 'CS_4804', 'CS_534', 'CS_539', 'CS_542', 'CS_548', 'CS_528', 'CS_582', 'CS_583', 'CS_561', 'CS_585']

Figure 19 – Data Science Major Requirements Input Sheet

1	Program Key	Sreq Key	Direction	Credits	Selection Type	Applicable Courses	Applicable Reqs	Sreq Type	Sublists Count
25	DS_MAJOR	DS_4LDER	AT LEAST	12	ANY OF	['CS_4120', 'CS_4233', 'CS_4241', 'CS_4341', 'CS_4342', 'CS_4432', 'CS_4433', 'DS_4433', 'CS_4445', 'CS_4804', 'BCB_4004', 'MA_4603', 'CS_4032', 'DS_4635', 'MA_4635', 'MA_4213', 'MA_4214', 'MA_4222', 'MA_4235', 'MA_4237', 'MA_4631', 'MA_4632', 'MIS_4084', 'MIS_4720', 'MIS_4741', 'OIE_4430', 'CS_534', 'CS_539', 'CS_542', 'CS_548', 'CS_528', 'CS_582', 'CS_583', 'MA_540', 'MA_541', 'MA_543', 'DS_502', 'MA_542', 'MA_554', 'MIS_571', 'CS_561', 'CS_585', 'DS_503', 'DS_541', 'CS_541', 'CS_586', 'DS_504', 'MIS_584', 'MIS_587', 'MKT_568', 'OIE_559']	['DS_DE', 'DS_CS', 'DS_MS']	1a	0
26	DS_MAJOR	DS_DAM	AT LEAST	3	ANY OF	['CS_4433', 'MIS_3720', 'CS_4432', 'DS_4433', 'CS_542']	['DS_DE', 'DS_CS']	1a	0
27	DS_MAJOR	DS_DMML	AT LEAST	3	ANY OF	['CS_4342', 'CS_4445', 'CS_539', 'CS_548']	['DS_DE', 'DS_CS']	1a	0
28	DS_MAJOR	DS_BMP	AT LEAST	3	ANY OF	['MIS_4084', 'OIE_4430', 'MIS_584', 'OIE_559']	['DS_DE']	1a	0
29	DS_MAJOR	DS_DPE	AT LEAST	3	ANY OF	['CS_3043', 'RBE_3100', 'PY_2731', 'PY_2713', 'RE_2731', 'GOV_2314', 'GOV_2315', 'GOV_2313']	['DS_DE', 'SOC_SCI_REQ', 'HUA_CORE']	1a	0
30	DS_MAJOR	DS_CS_1000_L	AT MOST	3	ANY OF	['CS_1004', 'CS_1101', 'CS_1102']	['DS_CS']	1b	0

Figure 20 – Data Science Major Super-Requirements Input Sheet

# Appendix C – Data Science Major Bucket Sheet

	A	B	C	D	E	F	G
	Bucket Key	Bucket Size	Choice Weight	Credits Each	Bucket Description	Bucket Contents	Req Keys
1	ART_CON	20	20	3	Humanities	['AR_DEPT','TH_DEPT','MU_DEPT']	['HUA_CORE','HUA_DEPTH_SL_0','HUA_ART_MAX']
2	FLG_CON	20	20	3	Humanities	['AB_DEPT','CN_DEPT','GN_DEPT','SP_DEPT']	['HUA_CORE','HUA_DEPTH_SL_1']
3	WR_CON	20	20	3	Humanities	['EN_DEPT','WR_DEPT']	['HUA_CORE','HUA_DEPTH_SL_2','HUA_WR_MAX']
4	HI_CON	20	20	3	Humanities	['HI_DEPT','HU_DEPT','INTL_DEPT']	['HUA_CORE','HUA_DEPTH_SL_3','HUA_HI_MAX']
5	PY_CON	20	20	3	Humanities	['PY_DEPT','RE_DEPT']	['HUA_CORE','HUA_DEPTH_SL_4','HUA_PY_MAX']
6	HUA_DPE	3	3	3	PY_2731, PY_2713, RE_2731	['PY_2731','PY_2713','RE_2731']	['HUA_CORE','DS_DPE']
7	HUA_PROJ	2	1	3	HU 3900 or 3910	['HU_3900','HU_3910']	['HUA_PROJ']
8	IQP_OFF_CM	5	100	9	IQP off campus	['IQP_A','IQP_B','IQP_C','IQP_D','IQP_OFF_CAMPUS']	['IQP']
9	IQP_ON_CM	1	25	9	IQP on campus	['IQP_ON_CAMPUS']	['IQP','ON_OFF_CAMPUS']
10	ID_2050	1	1	3	ID_2050	['ID_2050']	['SOC_SCI_REQ','ON_OFF_CAMPUS']
11	ALL_SOC_SCI	60	60	3	Social Science	['ECON_DEPT','ENV_DEPT','GOV_DEPT','PSY_DEPT','SO']	['SOC_SCI_REQ']
12	SOC_SCI_DPE	3	3	3	GOV_2314, GOV_2315, GOV_2313	['GOV_2314','GOV_2315','GOV_2313']	['SOC_SCI_REQ','DS_DPE']
13	DS_CORE	3	3	3	DS_1010, DS_2010, DS_3010	['DS_1010','DS_2010','DS_3010']	['DS_C']
14	MA_ELECT	17	17	3	Math Disciplinary Electives	['CS_2022','MA_2201','MA_3257','MA_1023','MA_1024']	['DS_DE','DS_MS']
15	CS_ELECT	5	5	3	Computer Science Disciplinary Electives	['CS_2301','CS_2303','CS_3041','CS_3133','CS_3733']	['DS_DE','DS_CS']
16	DIS_DPE	1	1	3	CS_3043	['CS_3043']	['DS_DE','DS_DPE']
17	3000L_CS_DAM	1	1	3	CS_3431	['CS_3431']	['DS_DE','DS_CS','DS_DAM']
18	4000L_CS_ELECT	5	5	3	4000-Level CS Disciplinary Electives	['CS_4120','CS_4233','CS_4241','CS_4341','CS_4804']	['DS_DE','DS_CS','DS_4LDER']
19	500L_CS_ELECT	11	11	4.5	500-Level CS Disciplinary Electives	['CS_534','CS_528','CS_582','CS_583','CS_561','CS_584']	['DS_DE','DS_CS','DS_4LDER']
20	DAT_MIN	2	2	3	4000-Level CS Data Mining & Machine Learning	['CS_4342','CS_4445']	['DS_DE','DS_CS','DS_4LDER','DS_DMML']
21	500_L_DAT_MIN	2	2	4.5	500-Level CS Data Mining & Machine Learning	['CS_539','CS_548']	['DS_DE','DS_CS','DS_4LDER','DS_DMML']
22	4000L_CS_DAM	3	3	3	4000-Level CS Data Access and Management	['CS_4432','CS_4433','DS_4433']	['DS_DE','DS_CS','DS_4LDER','DS_DAM']
23	500L_CS_DAM	1	1	4.5	500-Level CS Data Access and Management	['CS_542']	['DS_DE','DS_CS','DS_4LDER','DS_DAM']
24	4000L_ELECT	3	3	3	4000-Level Disciplinary Electives	['BCB_4004','MIS_4720','MIS_4741']	['DS_DE','DS_4LDER']
25	500L_ELECT	3	3	4.5	500-Level Disciplinary Electives	['MIS_571','MIS_587','MKT_568']	['DS_DE','DS_4LDER']
26	4000L_MA_DE	11	11	3	4000-Level Math Disciplinary Electives	['MA_4603','CS_4032','DS_4635','MA_4635','MA_4636']	['DS_DE','DS_MS','DS_4LDER']
27	500L_MA_DE	6	6	4.5	500-Level Math Disciplinary Electives	['MA_540','MA_541','MA_543','DS_502','MA_542']	['DS_DE','DS_MS','DS_4LDER']
28	BUS_DAM	1	1	3	MIS_3720	['MIS_3720']	['DS_DE','DS_DAM']
29	BUS_DE	3	3	3	MIS_3787, MKT_3650, OIE_3460	['MIS_3787','MKT_3650','OIE_3460']	['DS_DE']
30	4000L_BMP	2	2	3	MIS_4084, OIE_4430	['MIS_4084','OIE_4430']	['DS_DE','DS_4LDER','DS_BMP']
31	500L_BMP	2	2	4.5	MIS_584, OIE_559	['MIS_584','OIE_559']	['DS_DE','DS_4LDER','DS_BMP']
32	MATH	4	4	3	Calc I & II	['MA_1020','MA_1021','MA_1022','MA_1120']	['DS_MS']
33	ENTR_INV	6	6	3	Entrepreneurship and Innovation	['BUS_1010','ETR_1100','BUS_3010','ETR_3633','OB']	['DS_EI']
34	BUS_ANALYSIS	2	2	3	BUS_2080 OR OIE_2081	['BUS_2080','OIE_2081']	['DS_BA']
35	STAT	2	2	3	MA_2611 AND MA_2612	['MA_2611','MA_2612']	['DS_STATS']
36	LIN_ALG	2	2	3	MA_2071 OR MA_2072	['MA_2071','MA_2072']	['DS_LA']
37	ALGO	1	1	3	CS_2223	['CS_2223']	['DS_ALGOS']
38	ENG_SCI	150	150	3	Natural and Engineering Sciences	['AE_DEPT','AREN_DEPT','BB_DEPT','BME_DEPT','CHE']	['DS_NES']
39	1000L_CS	3	100	3	CS_1004 OR CS_1101 OR CS_1102	['CS_1004','CS_1101','CS_1102']	['DS_CS','DS_CS_1000_L']
40	INTRO_CS	3	99	3	CS_2102 OR CS_2103 OR CS_2119	['CS_2102','CS_2103','CS_2119']	['DS_CS']
41	DPE	1	1	3	Data Privacy and Ethics	['RBE_3100']	['DS_DPE']

Figure 21 – Data Science Major Bucket Sheet

## Appendix D – Stage I Complete Mathematical Model

The majority of this model comes from Lindsey Fletcher’s “Guiding Course Selection via Degree Evaluation Optimization” paper [18]. Our modifications referenced in section 3.1.4 are reflected in the formulation below. The suggested value for  $\varepsilon$  is 0.01.

---

<b>Sets</b>	
$P$	Set of programs being evaluated, indexed by $p$
$R_p$	Set of requirements $r$ that apply to program $p$
$M$	Set of buckets, indexed by $m$
$M_r$	Set of buckets $m$ that can be applied to requirement $r$
$R_m$	Set of requirements that courses from bucket $m$ can be applied to
$U_{max}$	Set of Type I super-requirements $u$ with an upper bound
$U_{min}$	Set of Type 1 super-requirements $u$ with a lower bound
$M_u$	Set of buckets $m$ to which super-requirement $u$ can be applied
$R_u$	Set of requirements to which super-requirement $u$ can be applied
$W$	Set of Type 2 super-requirements $w$
$V_w$	Set of subsets $v$ such that Type 2 super-requirement $w$ is satisfied if sufficient credits have been taken from a single subset $v \in V_w$
$M_v$	Set of all buckets $m$ such that $m$ is a subset of $v$
$R_w$	Set of requirements to which Type 2 super-requirement $w$ can be applied

---

<b>Parameters</b>	
$\alpha(m)$	Number of credits for any course in bucket $m$
$\beta(r)$	Number of credits needed for requirement $r$
$\mu(u)$	Credit upper or lower bound for a Type 1 super-requirement $w$
$\mu(w)$	Credit lower bound for any subset $v$ that satisfies Type 2 super-requirement $w$
$T(m)$	Number of courses already taken from bucket $m$
$\gamma(m)$	Total number of courses in bucket $m$
$\varepsilon$	Small multiplier to reward overflow
<b>Variables</b>	
$x_{m,r}$	Integer variable that represents the number of courses from bucket $m$ to requirement $r$
$y_m$	Integer variable that represents only the untaken courses from bucket $m$ assigned to any requirement
$q_{v,w}$	Binary variable that takes the value 1 if enough credits have been selected from subset $v$ for Type 2 super-requirement $w$ and 0 otherwise
$z_r$	Continuous variable that represents the number of overflow credits assigned to requirement $r$
$s$	Continuous variable that represents the sum of the additional credits needed and the overflow penalties (if any)

Objective:

$$\min s = \sum_{m \in M} \alpha(m)y_m + \sum_{r \in R_p} \varepsilon * z_r, \quad \varepsilon > 0$$

Subject to:

$$\sum_{r \in R_m \cap R_p} x_{m,r} \leq y_m + T(m), \quad \forall p \in P, \quad \forall m \in M$$

$$\sum_{m \in M_r} \alpha(m)x_{m,r} - z_r = \beta(r), \quad \forall p \in P, \quad \forall r \in R_p$$

$$\sum_{m \in M_u} \sum_{r \in R_m \cap R_u} \alpha(m)x_{m,r} \leq \mu(u), \quad \forall u \in U_{max}$$

$$\sum_{m \in M_u} \sum_{r \in R_m \cap R_u} \alpha(m)x_{m,r} \geq \mu(u), \quad \forall u \in U_{min}$$

$$\sum_{m \in M_v} \sum_{r \in R_w \cap R_m} \alpha(m)x_{m,r} \geq \mu(u)q_{w,v}, \quad \forall v \in V_w, \quad \forall w \in W$$

$$\sum_{v \in V_w} q_{w,v} \geq 1, \quad \forall w \in W$$

$$x_{m,r} \in \{0, 1, \dots, \gamma(m)\} \quad \forall m \in M, \quad \forall r \in R$$

$$y_m \in \{0, 1, \dots, \gamma(m) - T(m)\} \quad \forall m \in M$$

$$q_{w,v} \in \{0, 1\} \quad \forall v \in V_w, \quad \forall w \in W$$

## Appendix E – Stage II Complete Mathematical Model

The stage II formulation includes two additional parameters. All sets, parameters, and variables from stage I are still used here. All constraints from the stage I model are also still necessary for this stage.

---

<b>Parameters</b>	
$\Gamma(m)$	Choice weight assigned to every course in bucket m
$s^*$	Minimum number of untaken credits necessary to complete all requirements; takes the value of the stage I objective

---

Objective:

$$\max \sum_{m \in M} \Gamma(m)y_m$$

Subject to:

$$\sum_{m \in M} \alpha(m)y_m + \sum_{r \in R_p} \varepsilon * z_r = s^*$$

{*Stage I constraints*}

## Appendix F – Cross Restrictions

The lack of enforcement of cross restrictions in our data, which exist to restrict students from taking multiple courses within a defined set, leaves the model with room for error, in some cases more than others. Cross restrictions are present across most departments at WPI, and while they exist as an efficient way for the school to offer the same content to students in multiple departments or in different level programs, they are not well documented and are easily capable of confusing students.

There are two types of cross restrictions that serve different purposes – cross-listed courses and restricted pairs. A cross-listed course serves as a way for multiple departments to offer the same course content. It is essentially one course that has two or more different course codes. One of the most prominent examples in the WPI catalog is Discrete Mathematics. It has both a math course code (MA 2201) and a computer science course code (CS 2022), but it is one class that can be used to fill either a math or computer science requirement. Students are generally more aware that these restrictions exist, as the cross-listed courses have the same name and follow the same offering patterns, so enforcing these in the model is less important.

A restricted pair consists of two courses that cannot both be taken for credit, but they are not the same class. The material covered is similar enough between the two, however, that credit cannot be awarded for both of them. Most restricted pairs exist between advanced undergraduate courses at the 4000 level and graduate classes. For example, CS 539 and CS 4342 are both machine learning, but one is taught at the graduate level across a semester and one at the undergraduate level across a term. Courses in a restricted pair may or may not be taught by the same professor, but they are generally offered at different times, though there are a select few exceptions where the two levels are taught concurrently as one class. The enforcement of restricted pairs is very



important for B.S./M.S. programs because students are taking courses at both levels, and although these programs are not implemented in the current state of the optimization tool, they are first on the list of future extensions. Most restricted pairs are listed on a program's B.S./M.S. page in the catalog, but some are hidden in course descriptions, which means a lot of students do not know about them.

The issue with enforcing both types of cross restrictions is that it significantly restricts user choice. Cross restrictions can easily be added to the model using a super-requirement to enforce at most 3 or 4.5 credits for each pair of courses. One small issue with this is that super-requirements cannot be extended to free electives, so both courses in the restricted pair could technically appear in a solution and the student may not realize the solution is invalid. The bigger issue is that adding all these super-requirements completely changes the structure of the buckets. Each pair of courses ends up in its own bucket, because those two are the only courses that map to that specific cross restriction super-requirement. If these buckets get recommended to users, it suggests that there are only two courses that could fill that spot on the tracking sheet, when this is likely not the case. Even if we lower the choice weights on those buckets so they are only selected when there is no alternative, the chosen bucket will still be missing courses that apply to the same main set of requirements and super-requirements, limiting user choice anyway.

For these reasons, we chose not to enforce cross restrictions for our implemented programs. Because we are missing these restrictions, it is possible that we recommend users a bucket that contains a course they cannot take because they have taken the other half of the pair. We unfortunately have to accept this as a limitation of the model and instead make the user responsible for being aware of cross restrictions. We still believe giving the user more choice is better than

narrowing their view of the courses they can take, even if we must take the chance that users will understand the cross restrictions on their own.

## Appendix G – The B.S./M.S. Challenge

Even though we were unable to implement any B.S./M.S. programs, we felt it was important to include our approach in the paper, firstly because readers may have ideas on how to make it work in the future, and secondly that our resources sunk here explain the more limited scope of the other programs.

A B.S./M.S. degree is generally 162 credits, assuming the ability to double count 18 credits between both degrees. This poses a unique challenge not seen with the double major, because in that case, we aim to double count as many courses as possible between the two. There is not an easy way to limit the number of double countable courses within the solver, so the B.S./M.S. cannot fundamentally be solved the same way. We cannot treat the B.S. and the M.S. as two combined programs solved together, because the solver will double count more courses than are allowed. We cannot treat them as two separate programs and solve them separately, because the courses recommended may not be the same in both. Yet we also cannot treat it as one single program, because we cannot apply the same courses to multiple requirements within the same program.

To try to remedy this, we designed a multi-stage solve where we treat the B.S./M.S. as both a single combined program and two separate programs. First, we solve a combined program that contains 198 total credits - the 135 credits needed for the bachelor's, the 45 credits needed for the master's, and the 18 double-countable credits. We treat the 18 double-countable credits as a requirement, forcing the solver to either sort taken courses there, or recommend courses that can be double counted. To incentivize the solver to sort taken courses into the double count section, we added a reward into the objective function. The objective value gets a small bonus if taken courses get applied to the double count requirement instead of other applicable requirements.

The issue with this first solve is that we have added 18 additional credits to the total instead of subtracting them, because these 18 credits are still present in the bachelor's and the master's as well as having their own requirement. To remedy that, we now take these credits that got placed into the double counting requirement (whether already taken or not) and pass them into two separate solves.

First, we take the list of courses passed in as taken, add the recommended double-countable courses that have not already been taken, and pass it in to solve just the undergraduate degree. This is done to check that the same number of courses appear in the unused section (excluding taken graduate courses that can be applied to the master's). If the same number of unused courses are present in the first solve, this means there is at least one solution in which optimally double counting all 18 credits is possible. So, we then take any applicable graduate courses that appeared in the unused courses section, combine them with the previously chosen double countable courses, and use that list to solve just the master's program. If we do not see any unused courses that were not there previously, we have found an optimal solution, and we display output for the bachelor's degree, master's degree, and this list of double countable courses.

If we get an unused course that we did not expect in any of the three solves, this means there is not room to double count the maximum of 18 courses. Our hope is that most of the users of our tool will be early enough in their academic journeys to still optimally double count, but it is unrealistic to not expect cases where double counting the maximum number of credits is not possible. The reason this idea somewhat works is because most recommended courses exist in their own buckets because they are part of restricted pairs. If we did not include the cross-restrictions, we would be recommending buckets with more than just two courses in them, so our idea of passing untaken courses back into the second stage would not work.

This is as far as we got in solving the B.S./M.S. problem. We can tell if there is not enough room to double count the full 18 credits, but we could not come up with a way to efficiently determine exactly how many credits can be double counted. One idea was to do an exhaustive recursion where we try to decrease the credits in the double count requirement in intervals of 1.5, since this is the difference between an undergraduate class and graduate class. Another idea involved removing recommended double countable courses one by one until the number of unused courses remains the same between solves. This involved developing an order of importance to remove courses that limit user choice before ones that give the user more choice with their remaining untaken courses. All of these seemed like they might eventually work, but because they relied on recursion, none seemed like a very efficient way to solve the problem. Solving the problem of B.S./M.S. (and to a lesser extent minors, since they also have double counting limits) is fundamentally different one than that of single and double majors, so the other alternative is to consider formulating a new model.

## References

- [1] Fuller, J., Raman, M., et al. (October 2017). *Dismissed By Degrees*. Published by Accenture, Grads of Life, Harvard Business School.
- [2] Hanson, M. (2023, October 1). *College Enrollment Statistics [2023]: total + by demographic*. Education Data Initiative. <https://educationdata.org/college-enrollment-statistics>
- [3] Bouchrika, I. (2023, June 28). How much has college tuition increased in the last 10 years? *Research.com*. Retrieved October 13, 2023, from <https://research.com/universities-colleges/college-tuition-increase>
- [4] Barshay, J. (2017, September 4). Wasted time and money on undergraduate classes. *The Hechinger Report*. Retrieved October 15, 2023, from <https://hechingerreport.org/wasted-time-money-undergraduate-classes/>
- [5] S. K. Hayes, “Student employment and the economic cost of delayed college graduation,” *Journal of Business & Leadership*, vol. 6, pp. 129 – 140, 2010.
- [6] D. Witteveen and P. Attewell, “Delayed time-to-degree and post-college earnings,” *Research in Higher Education*, vol. 62, pp. 230 – 257, 2019.
- [7] Complete College America, “New rules: Policies to strengthen and scale the game changes,” 2016. Accessed on October 14, 2023.
- [8] *Future\_of\_Two\_Towers\_Part4.pdf*. (n.d.). Retrieved October 13, 2023, from [https://www.wpi.edu/sites/default/files/docs/Academic-Resources/Gordon-Library/Future\\_of\\_Two\\_Towers\\_Part4.pdf](https://www.wpi.edu/sites/default/files/docs/Academic-Resources/Gordon-Library/Future_of_Two_Towers_Part4.pdf)
- [9] *The WPI Plan | Worcester Polytechnic Institute*. (n.d.). Retrieved October 13, 2023, from <https://www.wpi.edu/project-based-learning/wpi-plan>
- [10] *fully\_updated\_WEBPAGE\_10-18-2021\_HUA-Brochure.pdf*. (n.d.). Retrieved October 13, 2023, from [https://www.wpi.edu/sites/default/files/inline-image/fully\\_updated\\_WEBPAGE\\_10-18-2021\\_HUA-Brochure.pdf](https://www.wpi.edu/sites/default/files/inline-image/fully_updated_WEBPAGE_10-18-2021_HUA-Brochure.pdf)
- [11] *Tuition & Fees | Worcester Polytechnic Institute*. (n.d.). Retrieved October 13, 2023, from <https://www.wpi.edu/offices/bursar/tuition>

- [12] Morrow, T., Hurson, A. R., & Sarvestani, S. S. (2017). A Multi-stage Approach to Personalized Course Selection and Scheduling. *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, 253–262. <https://doi.org/10.1109/IRI.2017.58>
- [13] Khamechian, M., & Petering, M. E. H. (2022). A mathematical modeling approach to university course planning. *Computers & Industrial Engineering*, *168*, 107855. <https://doi.org/10.1016/j.cie.2021.107855>
- [14] *Global Projects Program | Worcester Polytechnic Institute*. (n.d.). Retrieved October 13, 2023, from <https://wpi.cleancatalog.net/global-projects-program>
- [15] Wu, K., & Havens, W. S. (2005). Modelling an academic curriculum plan as a mixed Initiative constraint satisfaction problem. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3501 LNAI*, 79–90. Scopus. [https://doi.org/10.1007/11424918\\_10](https://doi.org/10.1007/11424918_10)
- [16] Olabumuyi, O. (2015). Application of optimization methods to bachelors degree planning. PhD thesis, Southern Methodist University.
- [17] *AE\_MS\_Degree\_Requirements.pdf*. (n.d.). Retrieved October 13, 2023, from [https://www.wpi.edu/sites/default/files/docs/Departments-Programs/Aerospace-Engineering/AE\\_MS\\_Degree\\_Requirements.pdf](https://www.wpi.edu/sites/default/files/docs/Departments-Programs/Aerospace-Engineering/AE_MS_Degree_Requirements.pdf)
- [18] Fletcher, L. (2023). *Guiding Course Selection via Degree Evaluation Optimization*. : Technical Report, Worcester Polytechnic Institute.
- [19] *Enrollment*. (n.d.). Tableau Software. Retrieved December 18, 2023, from [https://public.tableau.com/app/profile/wpi.institutional.research/viz/Enrollment\\_15718046316670/Enrollment](https://public.tableau.com/app/profile/wpi.institutional.research/viz/Enrollment_15718046316670/Enrollment)
- [20] *Programs of Study | Worcester Polytechnic Institute*. (n.d.). Retrieved December 18, 2023, from <https://wpi.cleancatalog.net/programs-of-study>
- [21] *Programs of Study | Worcester Polytechnic Institute*. (n.d.). Retrieved December 18, 2023, from <https://wpi-grad.cleancatalog.net/programs-of-study>
- [22] *Program Tracking Sheets | Worcester Polytechnic Institute*. (n.d.). Retrieved December 18, 2023, from <https://www.wpi.edu/student-experience/resources/academic-advising/program-tracking-sheets>

- [23] *M.S. in Data Science | Worcester Polytechnic Institute*. (n.d.). Retrieved February 18, 2024, from <https://wpi-grad.cleancatalog.net/data-science/ms-in-data-science>
- [24] *Welcome to Flask—Flask Documentation (3.0.x)*. (n.d.). Retrieved March 14, 2024, from <https://flask.palletsprojects.com/en/3.0.x/>
- [25] *CBC User Guide*. (n.d.). Retrieved February 21, 2024, from <https://www.coin-or.org/Cbc/cbcuserguide.html>
- [26] *OSCAR: Title—Tableau Server*. (n.d.). Retrieved February 18, 2024, from <https://tableau.wpi.edu/#/site/WPICommunity/views/OSCAR/Title?:iid=1>