



University of Kentucky
UKnowledge

Theses and Dissertations--Electrical and
Computer Engineering

Electrical and Computer Engineering

2019

Curricular Optimization: Solving for the Optimal Student Success Pathway

William G. Thompson-Arjona

University of Kentucky, wgthompson@uky.edu

Digital Object Identifier: <https://doi.org/10.13023/etd.2019.147>

[Right click to open a feedback form in a new tab to let us know how this document benefits you.](#)

Recommended Citation

Thompson-Arjona, William G., "Curricular Optimization: Solving for the Optimal Student Success Pathway" (2019). *Theses and Dissertations--Electrical and Computer Engineering*. 139.

https://uknowledge.uky.edu/ece_etds/139

This Master's Thesis is brought to you for free and open access by the Electrical and Computer Engineering at UKnowledge. It has been accepted for inclusion in Theses and Dissertations--Electrical and Computer Engineering by an authorized administrator of UKnowledge. For more information, please contact UKnowledge@lsv.uky.edu.

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

William G. Thompson-Arjona, Student

Dr. Gregory Heileman, Major Professor

Dr. Aaron Cramer, Director of Graduate Studies

Curricular Optimization: Solving for the Optimal Student Success Pathway

THESIS

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical
Engineering in the College of
Engineering at the University of
Kentucky

By

William Guillermo Thompson-Arjona
Lexington, Kentucky

Director: Gregory Heileman, Ph.D, Professor of Electrical Engineering
Lexington, Kentucky

2019

Copyright© William Guillermo Thompson-Arjona 2019

ABSTRACT OF THESIS

Curricular Optimization: Solving for the Optimal Student Success Pathway

Considering the significant investment of higher education made by students and their families, graduating in a timely manner is of the utmost importance. Delay attributed to drop out or the retaking of a course adds cost and negatively affects a student's academic progression. Considering this, it becomes paramount for institutions to focus on student success in relation to term scheduling.

Often overlooked, complexity of a course schedule may be one of the most important factors in whether or not a student successfully completes his or her degree. More often than not students entering an institution as a first time full time (FSFT) freshman follow the advised and published schedule given by administrators. Providing the optimal schedule that gives the student the highest probability of success is critical.

In efforts to create this optimal schedule, this thesis introduces a novel optimization algorithm with the objective to separate courses which when taken together hurt students' pass rates. Inversely, we combine synergistic relationships that improve a students probability for success when the courses are taken in the same semester. Using actual student data at the University of Kentucky, we categorically find these positive and negative combinations by analyzing recorded pass rates. Using Julia language on top of the Gurobi[®] solver, we solve for the optimal degree plan of a student in the electrical engineering program using a linear and non-linear multi-objective optimization. A user interface is created for administrators to optimize their curricula at *main.optimizeplans.com*.

KEYWORDS: Optimization, Curricular Analytics, Multi-Objective, Cloud LAMP Stack

Author's signature: William Guillermo
Thompson-Arjona

Date: May 2, 2019

Curricular Optimization: Solving for the Optimal Student Success Pathway

By

William Guillermo Thompson-Arjona

Director of Thesis: Gregory Heileman, Ph.D

Director of Graduate Studies: Aaron Cramer, Ph.D

Date: May 2, 2019

Esta tesis está dedicada a mis padres.

ACKNOWLEDGMENTS

Without the support and vision of my advisor and chair, Dr. Gregory Heileman, this thesis would not have been possible. Under his guidance and leadership I have grown as an engineer and a professional. Many thanks to Orhan Akbar and Gokhan Bakal, colleagues and friends with help programming this optimization. Also many thanks to Adam Roth for his invaluable help and guidance with the LAMP stack.

CONTENTS

Acknowledgments	iii
Contents	iv
List of Figures	v
List of Tables	vi
Chapter 1 Introduction	1
1.1 Background	1
1.2 Previous Work	3
Chapter 2 Tools, Structure, and User Interface	10
2.1 Julia	10
2.2 JuMP	11
2.3 Gurobi [®] Solver	11
2.4 Jupyter	12
2.5 User Interface → <i>main.optimizeplans.com</i>	12
Chapter 3 Optimization	17
3.1 Overall Constraints and Considerations	17
3.2 Bin Filling and Optimal Time to Completion	19
3.3 Multi-objective Optimization with Toxicity Avoidance	23
Chapter 4 Applications in Electrical Engineering	39
4.1 Power Transmission, Optimization of Damping Control	39
4.2 Smart Grid	41
Chapter 5 Conclusions and Moving Forward	44
Bibliography	45
Vita	47

LIST OF FIGURES

1.1	Tuition costs at public universities and colleges [3]	1
1.2	B.S. Electrical Engineering Program at the University of Kentucky, 2018	2
1.3	Example four course curricula subset demonstrating typical progression to circuits 1 in the electrical engineering curriculum	4
1.4	Curricular complexity metrics in relation to small adjustment in course placement	7
2.1	User interface created at <i>main.optimizeplans.com</i>	14
2.2	CSV data input front end linking to AWS-PHP LAMP stack to AWS S3 bucket	15
3.1	B.S. Electrical Engineering program at the University of Kentucky, 2018. Optimized using “bin filling” approach, function decomposition	22
3.2	B.S. Electrical Engineering program at the University of Kentucky, 2018. Optimized using “bin filling” approach, function combination	23
3.3	Example Curricula, Unoptimized	33
3.4	optimized example curricula, objective of toxicity minimization	34
3.5	B.S. Electrical Engineering program at the University of Kentucky, 2018. Optimized using toxicity avoidance objective	35
3.6	B.S. Electrical Engineering program at the University of Kentucky, 2018. Optimized using term imbalance minimization objective	36
3.7	B.S. Electrical Engineering program at the University of Kentucky, 2018. Optimized prerequisite string minimization objective	37
3.8	B.S. Electrical Engineering program at the University of Kentucky, 2018. Optimized using toxicity avoidance, term balancing, and prerequisite string minimization objectives	38
4.1	192Bus WECC Windfarm	40
4.2	Variability in power output of wind farm by day, for one month [18]	42

LIST OF TABLES

2.1	EC2 instance t2.medium specifications	13
2.2	LAMP stack description	15
3.1	Course toxicity example relationship	25
3.2	Course toxicity relationship between two courses in the electrical engineering curricula	27
3.3	User defined limits for optimization	29
3.4	Toxicity score (Ts) matrix	30
3.5	Binary course placement matrix X	31
3.6	Possible course combinations in the first term of the example curriculum with representative toxicity scores	33
3.7	First iteration of binary matrix x of example curriculum	34
4.1	Objective and constraints associated with inter oscillatory minimization	40
4.2	Objective and constraints associated with smart grid topology	42

Chapter 1 Introduction

1.1 Background

Completing a degree in higher education is a necessity in the 21st century global economy. According to the *bureau of labor statistics*, the average wage in 2017 for an individual with a bachelors degree with respect to an individual with a high school diploma was 39% more, a testament to higher education as “the gateway to the middle class” [1]. Attaining this level of education however is not trivial as many prospective students are faced with significant headwind. The decision to attend and invest the time and financial resources necessary makes the decision quite significant for most individuals. This also comes at a time of rising tuition costs greatly outpacing inflation. During the past decade, there have been exceptional and perhaps unprecedented increases in tuition at public colleges and universities. Poor economic conditions and subsequent state budget cuts have created a fertile landscape for large tuition increases. Although many of these year-to-year increases are in the neighborhood of 4% or 5%, a considerable number are above 10%, 15%, and even 20% [2].

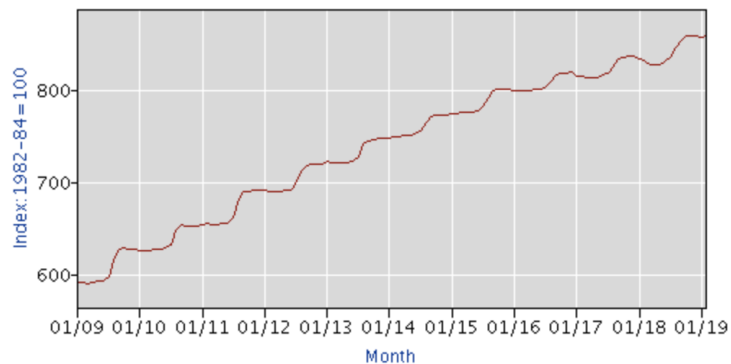


Figure 1.1: Tuition costs at public universities and colleges [3]

In light of these challenges placed on students and often time their families, it becomes evident that administrators in higher education have an obligation to dedicate significant thought and resources to assuring the success of their students. Frequently this comes

in the form of financial aid and work study programs. Large emphasis is also placed in other areas such as mental health counseling, student engagement and recreation initiatives, and corporate outreach. However, many times an underlining cause of student attrition is overlooked, that being the fundamental ways a student chooses to schedule his or her classes, the degree plan.

Degree plans are usually laid out by administrators to facilitate student course selection. They explain, usually in a graphical format, the expected course progression. Unfortunately, unbeknownst to the administrators, some of the semesters they have curated will adversely affect their students. Some courses when taken in the same term are much more difficult to pass. Specifically in electrical engineering this is the case for many course combinations.

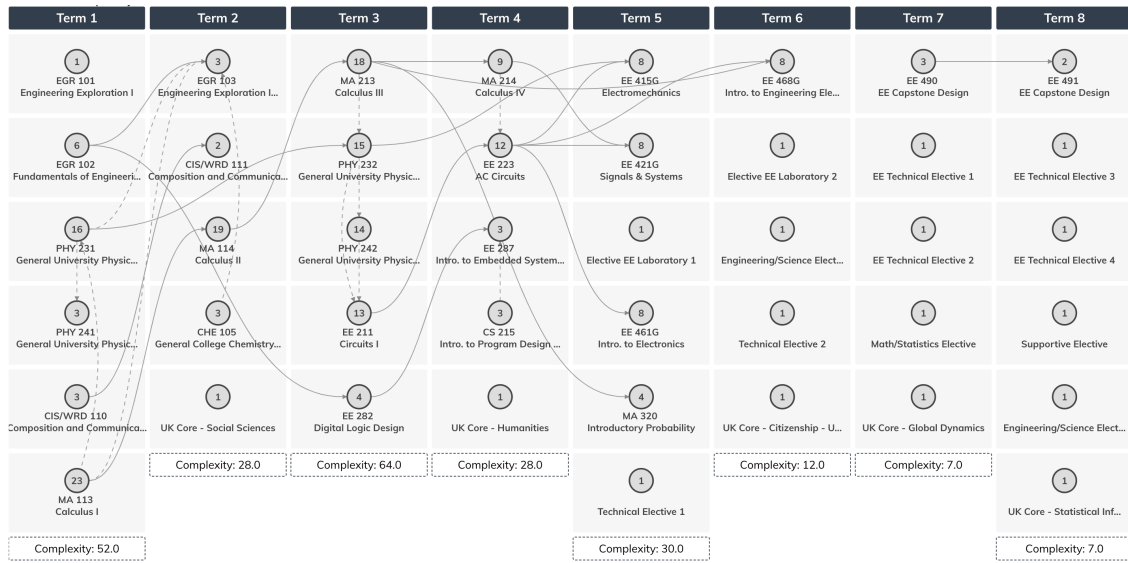
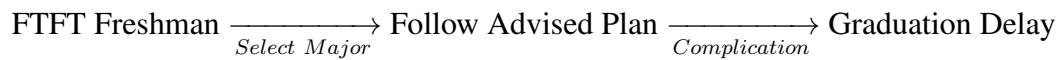


Figure 1.2: B.S. Electrical Engineering Program at the University of Kentucky, 2018

The negative effect one course may have on the other may be attributed to a multitude of factors, one of which is the sheer instructional difficulty and material needed to be digested by the student in a semesters' time. It would be more advantageous to spread the complexity of a curricula throughout the time the student is enrolled while trying to not overload any particular semester.

We previously demonstrated a direct relationship between the complexity of a curriculum and a student's ability to complete that curriculum [4]. In order achieve the same learning

outcomes needed to graduate from any particular field, students at universities across the country face drastically different complexity. More often than not, it is the student facing the lower complexity that successfully graduates and more specifically, graduates *on-time*. In fact, it has been proven that institutions categorically ranked higher in the *U.S. News and World Report* consistently offer less complex curricula than those ranked lower [6]. It is therefore of particular interest to administrators to quantitatively analyze their degree plans with efforts to lower their complexity, ultimately improving student success outcomes.



To improve these outcomes, degree plans must be given much more attention and analysis before being introduced to the student population. Everything from basic complexity metrics to advanced optimization techniques should be run in efforts to provide the optimal road map to success. This all begins with defining areas of concern within current pathways. If troubled course combinations can be found, articulated, and avoided in a degree plan, student success outcomes may be greatly improved.

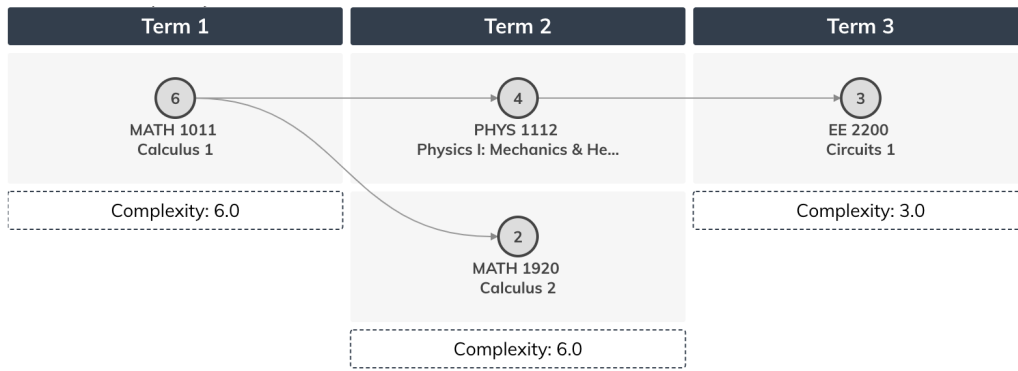
1.2 Previous Work

Curricular Analytics

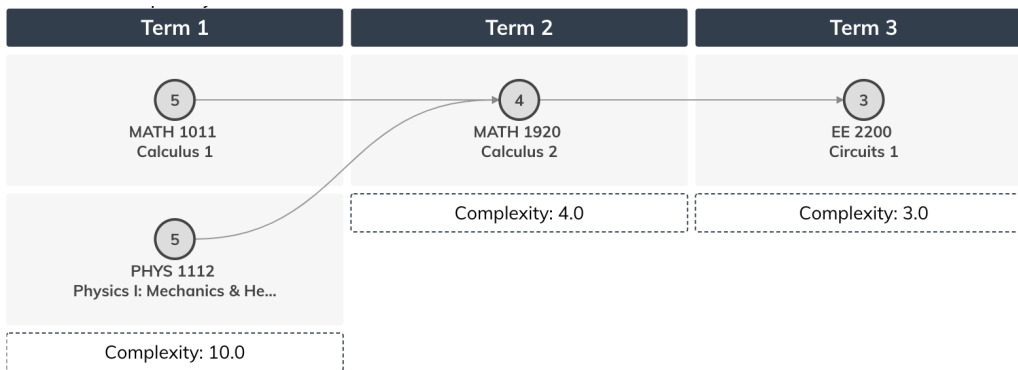
A great amount of work has been done in both the fields of curricular analytics and optimization techniques, but both concepts have rarely been used symbiotically. Much treatment has been devoted to quantitatively analyzing a course relative to its difficulty and attribution to student success outcomes. Analyzing this metric of a course, or complexity with respect to its placement and its relationship to the other courses, is crucial in understanding the reasoning and methodology behind the curricular optimization techniques.

We first must analyze the relationship of course within a degree plan, its placement crucial to the progression of a student. This placement relative to requisites is known as its *structural complexity*.

In a degree plan, it is observed that the graph structure of the mandated university curricula



(a)



(b)

Figure 1.3: Example four course curricula subset demonstrating typical progression to circuits 1 in the electrical engineering curriculum

and its corresponding structural complexity is a major factor that impacts a student's ability to complete the curricula. Specifically, we define this cruciality of a course within a degree plan as being associated with two main features, its delay factor and its blocking factor.

To understand what is meant by this we observe how slight variations in scheduling can effect the complexity of a degree plan. We analyze a typical progression to circuits one in the electrical engineering curriculum, a known path of high complexity.

To fully recognize the differences in structural complexity posed by the different degree plans in Figure 3.1, we must first define the relative complexity of a course in terms of the classes downstream that depend on it as prerequisite.

Delay Factor

Some courses have a critical impact on the academic progress of a student in the sense that any failure in these courses (or delays in taking them at the appropriate time) subjects the student to the risk of not finishing on time. Such is the case many times in science, technology engineering, and math (STEM) fields, which contain a set of courses that must be completed in sequential order. It is not uncommon to find prerequisite pathways consisting of up to eight courses, in effect spanning nearly every term in any possible degree plan. The ability to successfully navigate these long pathways without delay is critical for student success and on-time graduation [4]. Not only are these long pathways attributing to delay among their corresponding prerequisite strings, but they are often attributing to student delay across other facets of the curriculum.

We can more specifically define this *delay factor* associated with a given course v_k in a curriculum c , denoted $d_c(v_k)$, as the number of vertices in the longest path in G_c that passes through v_k [5].

G_c is known as the *curriculum graph*, where each vertex $v_1, \dots, v_n \in V$ represents a requirement (i.e., course) in a curriculum c . There is a directed edge $(v_i, v_j) \in E$ from requirement v_i to v_j if v_i that must be satisfied prior to the satisfaction of v_j [5].

$$d_c(v_k) = \max_{i,j,k,l} \{\#(v_i \rightsquigarrow v_k \rightsquigarrow v_j)\} \quad (1.1)$$

The delay factor associated with an entire curriculum c is:

$$d(G_c) = \sum_{v_k \in V} d_c(v_k) \quad (1.2)$$

Blocking Factor

Another structural factor arises when one course serves as the gateway to many other courses in the curriculum. In this case, if a student is unable to pass the gateway course, they are **blocked** from attempting many of the other courses in the curriculum [4]. This cruciality metric is a necessity when analyzing the effect of later semesters relative to the

completion of key classes near the beginning of a curriculum. If there is high enough blocking across many classes in a curricula, the result will be high stop out and attrition due to the inability to attempt many other courses.

In our example, *Calculus I* is a foundational first-term course that must be completed before taking the other major-specific classes in subsequent terms leading to our end goal of completing *Circuits I*. A course which is a prerequisite for a large number of other courses in a curriculum is a highly important course in that curriculum with regards to on-time degree completion.

Specifically we will define this *blocking* metric in relation to when course v_j is reachable from course v_i , via any prerequisite pathway, using $v_i \rightsquigarrow v_j$, and $v_i \nrightarrow v_j$ will be used if course v_j is not reachable from course v_i . The blocking factor associated with course v_i in curriculum $G_c = (V, E)$, denoted $b_c(v_i)$, is then given by [5]:

$$b_c(v_i) = \sum_{v_j \in V} I(v_i, v_j) \quad (1.3)$$

where I is the indicator function (i.e. the indication of whether or not the course downstream requires the course of interest as a prerequisite) :

$$= I \begin{cases} 1 & \text{if } v_i \rightsquigarrow v_j \\ 0 & \text{if } v_i \nrightarrow v_j \end{cases} \quad (1.4)$$

We define the blocking factor associated with an entire curriculum c as:

$$b(G_c) = \sum_{v_i \in V} b_c(v_i) \quad (1.5)$$

Complexity Score

After computing blocking and delay factor, a unit-less measure for structural complexity can be applied to every course in any curriculum. We keep in mind that the experimental

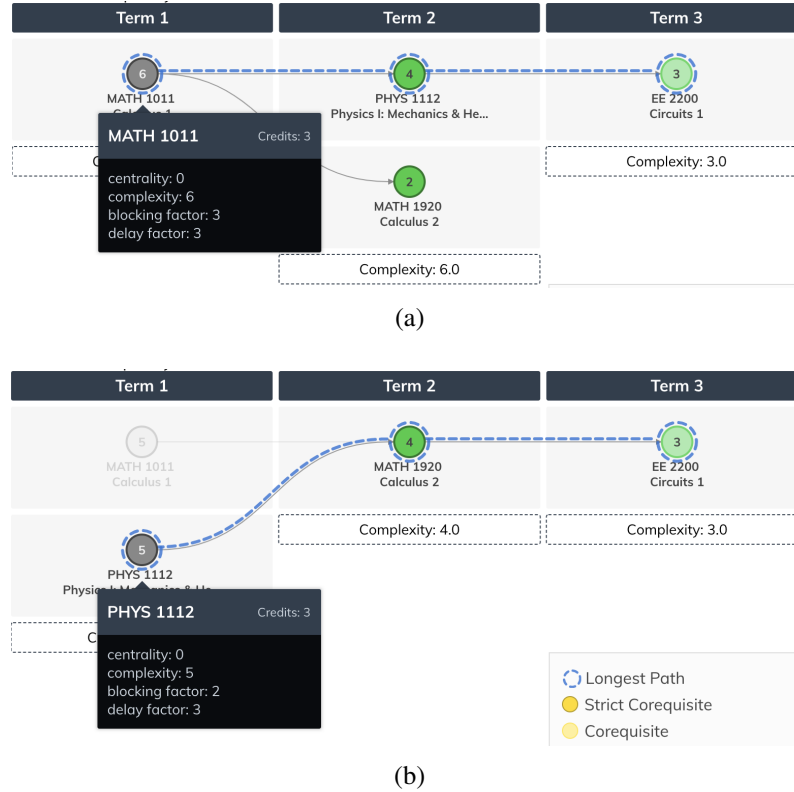


Figure 1.4: Curricular complexity metrics in relation to small adjustment in course placement

design explicitly relates this measure to the likelihood that a student can complete a curriculum. In order to achieve this overall *Complexity* metric, we simply add the blocking and delay factors of the entire curricula:

$$Complexity = b(G_c) + d(G_c) \quad (1.6)$$

It is beneficial to now analyze the difference between the two curricula in Figure 3.1 in relation to the formal structural metrics that have been defined. With the adjustment of *Physics I* from the second term to the first term (sub-figure *a* to sub-figure *b* in Figure 3.1 respectively) we see the complexity not only change for the course itself, but for courses in the same prerequisite chain. Observed is a chain reaction that leads to the complexity change of relational classes such as *Calculus 1*, which has its complexity score drop from 6 to 5. This is due to a reduction in *blocking factor*, in that PHYS 1112 is no longer inaccessible if a student does not progress past MATH 1011. Small adjustments such as

these demonstrate the causal sequence of events that transpire when a degree plan is modified. Hence, it is instrumental that administrators quantitatively analyze and study any adjustments that are proposed, as they may have a significant impact on student success outcomes. Any change has a significant chance to detrimentally or advantageously affect the progression of a student. It is therefore through the power of iterative optimization techniques founded in real student data that an optimal plan may be produced.

Optimization

Optimization techniques are essential in many modern day applications. They drive profit for shareholders of many corporations by maximizing resources and minimizing sources of cost. Not only used by corporations, these techniques can be found in all areas of applied science, engineering, economics, and statistics. In fact, optimization techniques are used in most decision making algorithms where a “best” choice should be made. The most obvious example of optimization is the way in which the text of this thesis is laid out using \LaTeX typesetting. The aim of the system is to produce a visually appealing arrangement of text subject to the constraints of margins, and spaces between letters, words, and paragraphs. The parameters can be slightly adjusted in order to achieve the best objective, a process that includes many elements of a general optimization problem.

All optimization problems follow the same general format in that they minimize or maximize a defined objective subject to constraints. The model can take on a general form such that[7],

$$\begin{aligned}
 & \underset{x \in R^n}{\text{minimize}} && F(x) \\
 & \text{subject to} && c_i(x) = 0, \quad i = 1, 2, \dots, m', \\
 & && c_i(x) \geq 0, \quad m' + 1, \dots, m
 \end{aligned} \tag{1.7}$$

where $F(x)$ represents the general objective function that can be either minimized or maximized subject to the real valued scalar functions c_i .

These optimization techniques in relation to curricular analytics have received limited treat-

ment. In their work presented in the European Journal of Operational Research, Ünal and Uysal present a balancing academic curriculum problem (BACP). The BACP schedules courses to different semesters, while balancing the total workload per period. In the study they create a *Revalency Score*. This score represents the level of interdependency between two courses [8]. The optimization works to minimize the difference between two courses that are highly relevant to each other. Their methodology is surrounded by the use of what they refer to as the Relevance Based Curriculum Balancing (RBCB) that is formulated as bi-objective Mixed Integer Linear Programming (MILP) problem. The RBCB has to ensure the workloads are being distributed to the semesters in a balanced fashion. To achieve this goal, the degree of “balance violation” (i.e. deviation from the average workload) is calculated for each semester and penalized in the objective function [8]. The optimization posed is represented such that,

$$\text{minimize } R(x) + B(L(y)) \quad (1.8)$$

where $R(x)$ aims at minimizing the total layout cost (i.e. distance with respect to relevance score), and $B(L(y))$ represents the minimization of imbalance of workloads per semester.

Although thorough in its treatment of degree plan optimization as a MILP problem, the paper leaves questions with regards to the aspect of which an optimized plan can be rooted in actual student pass rates. Also, user interactability is limited (e.g. administrator ability to input constraints such as desired term count and credit hours per term). In the following chapter we will discuss how the user is put in a position of control over their optimization by product of a novel user interface.

Chapter 2 Tools, Structure, and User Interface

Language

2.1 Julia

The optimization was implemented in the open-source Julia programming language. Julia is described as a high level, high performance dynamic programming language for technical computing that “has the performance of a statically compiled language while providing interactive dynamic behavior and productivity like Python, LISP or Ruby” [9]. It is built upon an LLVM- based just-in-time (JIT) compiler which allows it to reach performance close to that of C and, in many cases, speeds faster than that of R or MATLAB [9]. Julia’s defining feature is multiple dispatch, but some other notable features are module support, a type system, parallelism, and a built-in package manager. It also has a thriving community of developers contributing high-quality open source libraries spanning a range of applications such as machine-learning, statistical analysis, graph analysis, plotting, and data-handling [10].

Each of these features contributed to choosing Julia as the language of implementation. Julia’s type system, ease of use, and dynamic nature made it simple to implement the optimization’s components, methods logic, and support for file formats such as CSV and JSON that make data IO simple. In addition, it has a built-in package manager with many packages that enable uses such as powerful machine learning, statistical models, and visualization capabilities [10]. In this case specifically, we use packages such as Multi-objective, JuMP, and LinearAlgebra, which we will describe in greater detail.

Most importantly however, Julia was chosen for its integration and support on top of the Gurobi[®] solver. The clearly defined language interface that seamlessly integrated the powerful solver allowed for easy implementation and definition of constraints and objectives.

Choosing Julia also allowed for integration of the optimization into the Curricular Analytics toolbox, allowing for further advanced analysis [11]. The toolbox allows for the resultant

degree plan produced from the optimization to be visualized and analyzed for validity, among many other features.

2.2 JuMP

JuMP is an open-source modeling language that allows users to express a wide range of optimization problems (linear, mixed-integer, quadratic, conic-quadratic, semidefinite, and nonlinear) in a high-level, algebraic syntax [12]. In essence, it is a domain-specific modeling language for mathematical optimization embedded in Julia. It easily and seamlessly acts as the bridge between the solver and the higher level programming language. Through its intuitive syntax with respect to defining variables, constraints, and objectives in an optimization, more time could be spent in design of the methodology than time spent finding a way to program. Speed is also quick in that JuMP communicates with most solvers in memory, avoiding the need to write intermediary files. There is computational evidence that JuMP is able to produce quadratic and conic-quadratic optimization models, in a format suitable for consumption by a solver, as fast as state-of-the-art commercial modeling languages such as MATLAB and Python [12].

$$Julia \xleftrightarrow{\text{JuMP}} Gurobi^{\text{®}}$$

2.3 Gurobi[®] Solver

Due to the high computation requirement drawn by our optimization, a powerful solver was needed. Thousands of iterations of the degree plan need to be run quickly in order to find the optimal result. Along with this, the solver needed to integrate seamlessly with the Julia language. Gurobi met or exceeded all of our requirements. Gurobi is a commercial solver for both linear programming and mixed integer linear programming. According to the MIPLIB 2017 Benchmark, Gurobi is the fastest to optimality, feasibility, and infeasibility [13]. The proven solver is used in applications ranging from optimization methods for genome scaffolding to optimal inmate assignment programs, which saved the Pennsylvania prison system \$3 million USD in just the first year of use [14].

2.4 Jupyter

Jupyter, an application originating from within anaconda, is an interactive coding platform that integrates Julia seamlessly. The step wise nature of a jupyter notebook made troubleshooting and collaborative coding possible during the development of the optimization. The graphical file structure make the linking of dependent files within the notebook simple. Jupyter notebooks may be run locally through the installation of the IJulia package within the Julia REPL. Specifically, live code, equations, narrative text, and the result degree plan visualizations were all produced within this interactive environment.

In order to create an interactive environment where code could be easily distributed and modified by various parties in the research group, the notebook was mounted on an Amazon Web Services (AWS) Elastic Cloud Compute (EC2) Ubuntu instance. Here all necessary dependencies are installed within Julia. Once the environment is prepared, the notebook is then initialized through the provided *tornado* server, inherent to anaconda, and optimized for asynchronous input/output. This “hosted” notebook benefited research in that versioning control of the underlying curricular analytics toolbox examples and dependencies [11]. The notebook in which the curricular optimizations reside can be found at <https://optimizeplans.com:8080>

2.5 User Interface → *main.optimizeplans.com*

The ease of which the optimizations can be used is paramount to making meaningful progress in improving student success. In an industry that is already reluctant to change, providing an easy to use dashboard to select and run the optimization with the users’ specific curricula is critical. An intuitive dashboard was built on AWS with ends to create a centralized compute hub where administrators can quickly and easily get results by optimizing to one of the supported objectives. In order to instantiate this service, we would need to first select and register a domain name.

Optimizeplans.com was chosen and registered on AWS Route 53 service. In order to connect to the infrastructure, a variety of record sets would need to be created. First would be the aliased record (A record) set pointing to our particular IPv4 address. This would link

the domain to our service. In particular, a canonical name (CNAME) was produced as to properly denote the host name.

In order to host the compute capability of the infrastructure, AWS Elastic Cloud Compute (EC2) was used. Considering the memory and compute requirements a t2.medium linux instance was chosen for the jupyter compute section of the framework.

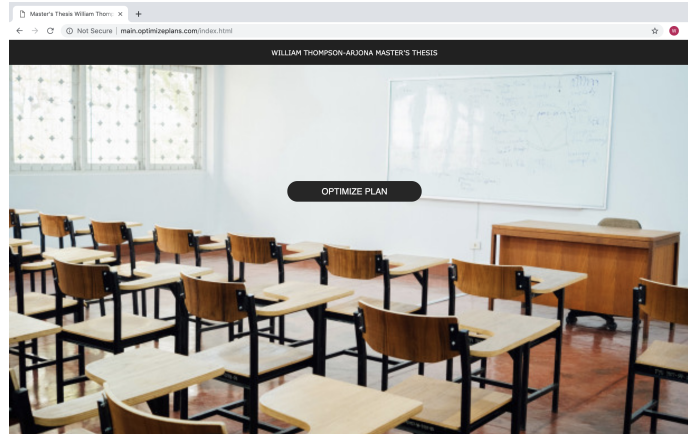
Table 2.1: EC2 instance t2.medium specifications

<i>vCPU</i>	2@ 3.3GHz
<i>CPU Credits/Hour</i>	24
<i>Memory(GiB)</i>	4

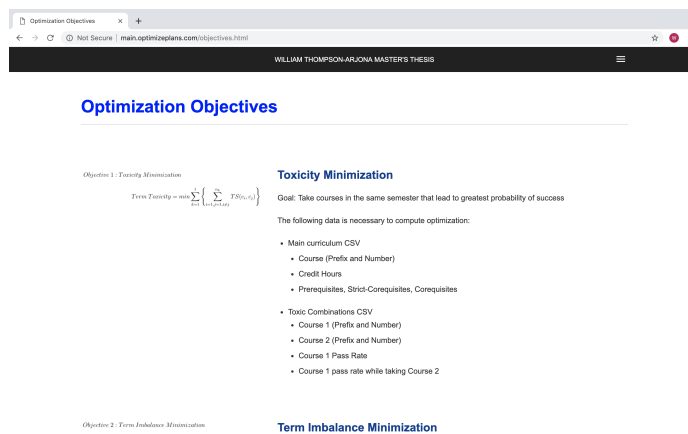
Once the instance was initialized the security groups were established as to allow for front facing public access. Due to the specification of the apache *tornado* server inherent to anaconda and hence the notebook, the design decision was made to create a separate EC2 instance for the customer facing user interface.

The second EC2 instance uses Ubuntu 18.04 as the OS due to its popularity in server architectures along with an Apache web server. PHP was chosen due to its large community, prevalence as a server side language, and ease to develop web applications. Composer was used as the PHP dependency manager, which was used to download the AWS PHP software development kit (SDK). This SDK allows for PHP integration into a variety of necessary services such as AWS S3. Once the general graphical user interface was laid out through basic HTML, links to each particular jupyter notebook were set to correspond to each one of the optimization objectives under the *Optimization Objectives* tab. In production, users will be able to input their data directly into the examples folder located within the notebook framework and run the optimization from the convenience of their machine, without direct administrator contact. However, due to Gurobi being a commercial solver, a cloud license will need to be purchased in order to host the capability through the supported optimization notebooks.

Per Gurobi, pricing is variable based off use, with an unlimited perpetual use license costing \$30000 USD per year. In this application the optimizations only need be run once per



(a) Home screen



(b) Optimization objectives with linked jupyter notebook

Figure 2.1: User interface created at *main.optimizeplans.com*

curricula, not exerting extreme compute resources. This being the case, a *silver* package could be purchased for \$10000 USD per year with an additional cost of \$8 USD per hour of use. This option is ideal for active development and deployment situations where the hourly charge is more of a factor. This package could be hosted directly by our EC2 instance as it includes a compute server license.

Until the Gurobi cloud license is purchased and Gurobi is installed as a dependency on the notebook EC2 instance, users will have to input the necessary CSV data and the administrator will have to run the optimizations. In order to achieve this data handshake, a bridge to an AWS S3 bucket was created. The created bucket was given corresponding IAM access role with programmatic access. This would be needed in order to create the access keys necessary to link the input data into the bucket. The user uploads the CSV file

Input CSV

WILLIAM THOMPSON-ARJONA MASTER'S THESIS

Main Curricula

Choose File No file chosen

Upload Curricula CSV

The following are the required fields:

- Course (Prefix and Number)
- Credit Hours
- Prerequisites, Strict-Corequisites, Corequisites

Toxicity

Choose File No file chosen

Upload Toxicity CSV

Upload for the most advanced optimization techniques, identifying toxic and synergistic course combinations. Toxic courses are separated while synergistic courses are placed within the same term, subject to the described constraints. The following are the required

Course 1	Course 2	Take C1 Only	Take C1 Only Passed	Take Both	Take Both Passed C1
CHE 111	CHE 109	7664	5747	84	2
CHE 105	CHE 109	20382	7428	112	1
MA 193	ABS 300	9700	7273	37	4
MA 113	ABS 300	9924	6432	40	4
CHE 105	ABS 111	3727	2542	36	6
ABS 100	NUR 882	5107	4223	35	10
CHE 107	ASC 325	5871	3804	41	6
ANA 109	UK 090	2204	3305	53	16
CHE 107	ASC 310	12290	8689	31	8
CHE 212	ABS 100	4770	3449	50	25
POY 206	BI0 102	5226	3989	32	11
ECD 201	UK 090	12170	10549	31	14
MA 193	ABS 100	4649	3529	32	10
ANA 109	UK 100	4692	3582	48	17
CHE 105	ASC 320	21458	15087	34	10

Figure 2.2: CSV data input front end linking to AWS-PHP LAMP stack to AWS S3 bucket in an HTML form, in which a web request is sent to the Apache web server. Web server (in PHP) handles the request by processing and verifying the file, and then uploads the file to the S3 bucket sub directory, which are partitioned based off the file type uploaded.

Table 2.2: LAMP stack description

LAMP Stack Component	Description
(L)inux	Open-source Operating system
(A)pache	Open-source cross-platform web server software
(M)ySQL	Open-source relational database management system
(P)HP	Server side general purpose programming language

Through AWS, a MySQL db.t2.medium instance was set up by the maintained RDS service. MySQL was chosen due to its large community, wide popularity, open-source nature, and database structure that integrates well with CSV files. The instance includes adequate amount of RAM for the application (4 GiB) and CPU sufficient for the amount of uploads and retrievals the application will see (24 Credits/hour at 3.3 GHz). RDS was configured

with a database name and key, so it can only be accessible through authenticated credentials. Security groups were set to restrict incoming access to the database based upon IP, so that only port 3306 is accessible from optimizeplans.com's server's private IP address. This database was linked to our system through an RDS endpoint that allowed the server to connect via PHP. Optimization data will be stored in a table with the columnar structure representing data from the CSV optimization (curricula and toxicity). Columns in this table store each field as the proper data type (e.g. credit hours as an integer). This way all data types are properly converted to and from the back end system.

main.optimizeplans.com is now active and ready to accept administrator's curricula and toxicity CSV file types, ultimately providing an easy access point for them to directly improve student success outcomes through optimized degree plans.

Chapter 3 Optimization

3.1 Overall Constraints and Considerations

It is crucial for a student in higher education to complete their degree requirements in a timely manner. In large part this is due to financial strain caused by the addition of further semesters as well as the probability to complete a degree successfully when time to completion is minimized [4].

In undergraduate curricula of higher education institutions, there are generally 8 semesters and around 40 courses. A course establishing the fundamentals of a more advanced course is treated as the prerequisite and scheduled earlier in the curriculum [8]. The path to completion from the initial term to the final term is fraught with peril, such that the course placement on a term over term basis is critical. The degree plan in this case should be treated as a “live document” in which each subsequent term changes relative to the courses completed successfully in the current term. While this optimization sets to find the optimal degree plan in its totality, future applications of the tool could be used in advising situations where students enter with varying backgrounds and transfer credits.

All things considered and irrespective of varying student backgrounds, a set of fundamental truths apply for all students at the start of their undergraduate journey. The first of which is the objective to graduate in a timely manner. In its most basic sense, the introduced algorithms follow a linear integer programming model that outputs optimal course progression in the minimum amount of time, considering a student’s desired course load.

Goal: Minimize time to degree completion.

min = Minimize number of terms

While the goal, or *objective*, may be straightforward in degree plan optimization, there are a variety of considerations that must be addressed in order to keep the degree plan

valid. The first of which is that the course load should be balanced throughout a student progression term over term. This will avoid any unnecessary loading on one term over the other, insuring no one term is drastically more credit hour intensive than the others in the plan. If l is the term course load in credit hours, this can be represented by,

Objective: Keep course load even throughout a student's progression

$$\min \sum_{i=1}^m |l_{i-1} - l_i|$$

Where m is the number of courses

To insure the student completes his or her degree in the desired time frame, the credit hour per term must be kept at a maximum as desired by the student such that,

Objective: Keep course load as maximum per semester to maximum desired by student

$$\min \sum_{i=1}^m |l_i - \beta|$$

Where β is the number of credit hours

Finally, in order to insure validity of the plan, all prerequisites, corequisites, and strict corequisites must be honored during the optimization. This insures that students have the required learning outcomes required before attempting a more advanced course.

Constraint: Prerequisite classes will be honored during the optimization

$$\sum_{i=1}^n i * x_{ai} < \sum_{j=1}^n j * x_{bj}$$

In the simplest sense, the algorithm will output a binary result with respect to whether or not the particular term is optimal such that,

$x = 0$ when semester count not optimal

$x = 1$ when semester count optimal

Once the optimal term is found, the optimization iteratively moves on to the next term. This generalized process is repeated through many, sometimes thousands, of iterations in order to find the optimal result. In the following sections the optimization techniques are more closely described with the end goal of delivering the best degree plan for the student.

3.2 Bin Filling and Optimal Time to Completion

From a student's perspective, there is perhaps no more important objective than to complete his or her degree in the quickest amount of time. For the discerning student who is driven by financial circumstances, a desire to begin graduate school early, or a want to become financially independent in the optimal time, the quickest route to completion is desired. This is where the "bin filling" approach may be deployed. Although not taking into consideration some of the advanced constraints we have developed in our multi-objective optimization approach, the filling algorithm, in a literal sense, fills a degree plan term by term, up to a credit hour limit desired by the user. The approach be loosely defined such that given n number of courses in n number of terms, where c is the *capacity* of each term in maximum credit hours, and h_j the credit hours of the course, such that

$$\begin{aligned}
 & \text{minimize} && \sum_{i=1}^n y_i \\
 & \text{subject to} && \sum_{j=1}^n h_j x_{ij} \leq c y_i \quad i \in N = \{1, \dots, n\}, \\
 & && \sum_{i=1}^n x_{ij} = 1 \quad j \in N, \\
 & && y_i = 0, 1 \quad i \in N, \\
 & && x_{ij} = 0, 1 \quad i \in N, j \in N
 \end{aligned} \tag{3.1}$$

The objective would be to fill each term with the maximum of credit hours h . By this we accomplish a degree plan with the minimum possible amount of terms used. A *find minimum terms* function was created to find the minimum number terms possible *while*

respecting all requisite conditions. The algorithm was designed such that it would return a tuple with three elements.

- Boolean value which shows the term list created for the total term count.
- A term list which contains all courses that were placed in a designated term, grouped by a term ID
- The term count is a integer value to show minimum number of terms possible to satisfy the placement of all courses in a curricula to designated terms.

It must be noted that the *find minimum terms* function introduces no balancing function, hence the possibility arises to produce uneven terms (e.g. terms with unequal distribution of credit hours). This is due to the objective of the bin filling approach to be to place all corresponding courses that “fit” within a term as early as possible, while honoring requisites.

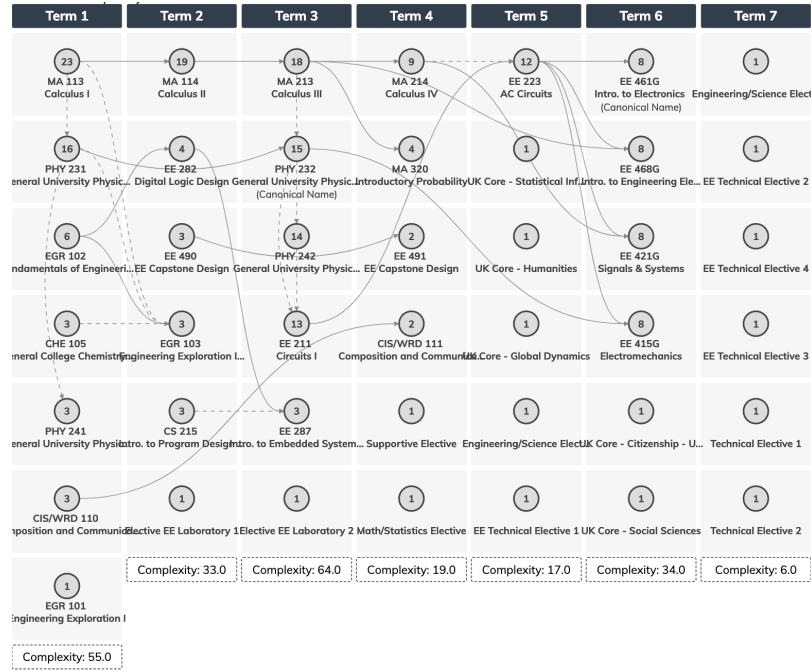
To balance the terms with respect to credit hour load, a second function would be introduced, aptly named *balance terms*. The objective of this function would be to equally distribute the course load amongst filled terms, such to minimize the course load difference between the terms subject to the maximum allowable term load (in credit hours) as defined by the user. As in the *find minimum terms* function, a 3 element tuple is returned upon the introduction of a curricula.

- Boolean value which shows the term list created for the total term count.
- A term list which contains all courses that were placed in a designated term, grouped by a term ID
- *max credits* is a integer value to represent the maximum number of credit hours assigned to any of the terms.

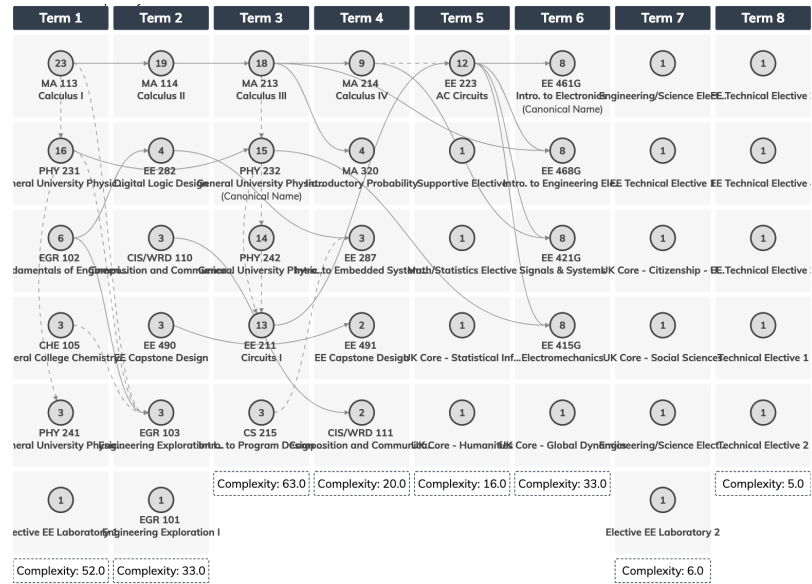
Subsequently, the *balance terms* function and the *find minimum terms* function can be combined to create a powerful degree plan creation tool that simultaneously provides the fastest

time to completion while maintaining credit hour load homogeneity across the degree plan, all while honoring the requisite relationships. The following are the resultant plans output from the above functions, visualized using the curricular analytics toolbox [11].

Results



(a) *find minimum terms function*



(b) *balance terms function*

Figure 3.1: B.S. Electrical Engineering program at the University of Kentucky, 2018. Optimized using “bin filling” approach, function decomposition

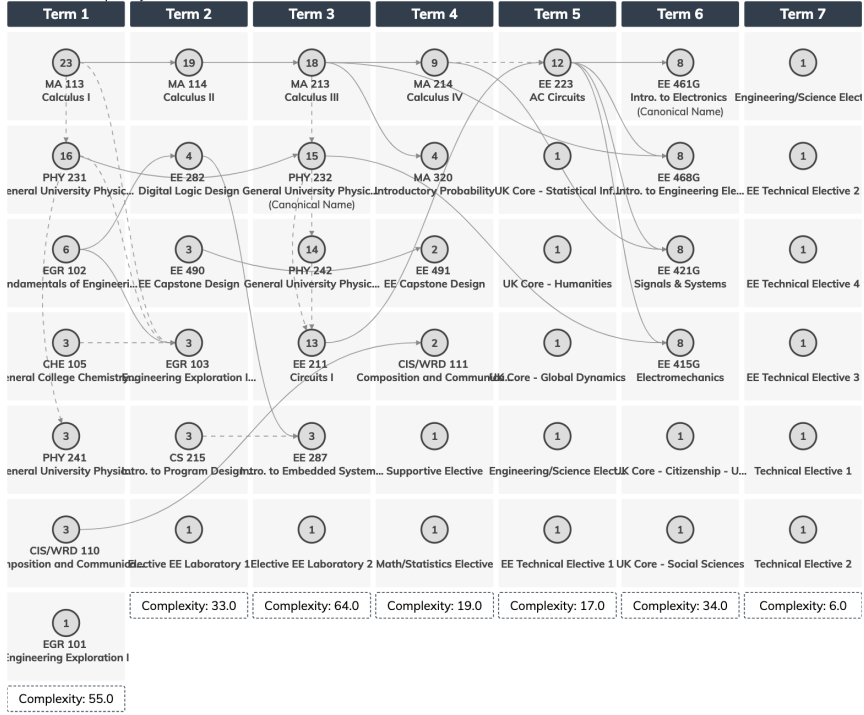


Figure 3.2: B.S. Electrical Engineering program at the University of Kentucky, 2018. Optimized using “bin filling” approach, function combination

While the bin filling approach does have its limitations (e.g. no fixed course array to place senior design in last terms), the objective of minimizing the amount of terms subject to the desired credit hour load is accomplished. If targeted specifically to ambitious students, these degree plans may be useful as a road map to graduate early. As demonstrated with a credit hour maximum set at 19, electrical engineering students at the University of Kentucky can graduate 1 semester early.

3.3 Multi-objective Optimization with Toxicity Avoidance

Before introduction of the multi-objective optimization methodology, the concept of toxic and synergistic course combinations are introduced.

Definition of Toxic Course Combination

Consider a population of students P and a set of courses C . For all courses $c_i \in C$, let $X_{c_i} \in P$ denote the set of all students who attempted course c_i for the first time. We

can partition this set based upon the outcomes of the students' course completion attempts. Namely, let $X_{c_i}^p$ denote the subset of students who completed course c_i with a passing grade on the first attempt, and let $X_{c_i}^{\bar{p}}$ denote the set who did not.¹ *Toxic course combinations* are ones that have a negative impact on student progression when paired together in a given term. More formally,

Definition 1 (Toxic Course Combination). *Consider the two courses $c_i, c_j \in C$. These courses are considered a toxic combination if*

$$\Pr\{X_{c_i}^p\} - \Pr\{X_{c_i}^p | X_{c_j}^p\} > \theta_t.$$

□

In other words, courses c_i and c_j are toxic in combination if a student's probability of passing c_i is significantly reduced in the event they attempt to successfully complete c_j at the same time (i.e., in the same term).

Definition of Synergistic Course Combination

We may also similarly define *synergistic course combinations* as those course pairs that tend to facilitate student progression. That is,

Definition 2 (Synergistic Course Combination). *Consider the two courses $c_i, c_j \in C$. These courses are considered a synergistic combination if*

$$\Pr\{X_{c_i}^p\} - \Pr\{X_{c_i}^p | X_{c_j}^p\} < \theta_s.$$

□

In this case, taking course c_j in combination with course c_i improves a students chances of passing c_i .

¹We assume $X_{c_i}^p \cup X_{c_i}^{\bar{p}} = X_{c_i}$, i.e., all students attempting the course must be placed into one of these two categories.

Identification of Toxic/Synergistic Combinations

One can begin to formulate a design of experiments to identify these combinations relative to student data. This is done by observing the pass rates of the course as a singular and uncorrelated event. It is then compared to the pass rate of the course when taking a second course of interest. The following demonstrates this principle:

Example 1. Consider two courses c_1 and c_2 with the following characteristics:

Table 3.1: Course toxicity example relationship

	enrollment	# passed
c_1	20	12
c_2	30	21
$c_1 \cap c_2$	10	5

Letting frequencies approximate probabilities, we can estimate the toxicity/synergism of c_1 and c_2 . Specifically, $\Pr\{X_{c_1}^p\} \approx 12/20 = 0.6$, $\Pr\{X_{c_2}^p\} \approx 21/30 = 0.7$, and $\Pr\{X_{c_1}^p, X_{c_2}^p\} \approx 5/50 = 0.1$. Thus,

$$\Pr\{X_{c_1}^p | X_{c_2}^p\} = \frac{\Pr\{X_{c_1}^p, X_{c_2}^p\}}{\Pr\{X_{c_2}^p\}} = \frac{0.1}{0.7} = 0.14.$$

and

$$\Pr\{X_{c_2}^p | X_{c_1}^p\} = \frac{\Pr\{X_{c_1}^p, X_{c_2}^p\}}{\Pr\{X_{c_1}^p\}} = \frac{0.1}{0.6} = 0.16.$$

The toxicity value (relative to c_1) associated with attempting c_1 and c_2 at the same time is:

$$\Pr\{X_{c_1}^p\} - \Pr\{X_{c_1}^p | X_{c_2}^p\} \approx 0.6 - 0.14 = 0.46,$$

and the toxicity value (relative to c_2) associated with attempting c_1 and c_2 at the same time is:

$$\Pr\{X_{c_2}^p\} - \Pr\{X_{c_2}^p | X_{c_1}^p\} \approx 0.7 - 0.16 = 0.54.$$

If the toxicity threshold is $\theta_t = 0.4$, then the course combination is toxic, with c_1 having a slightly more toxic impact on c_2 than c_2 has on c_1 .

Design of Optimization

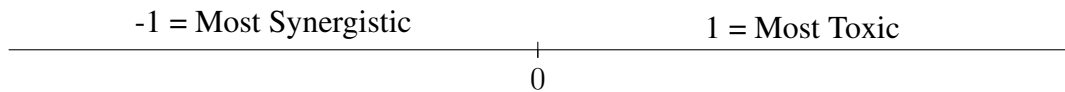
Data Collection and Toxicity Score Calculation

The most powerful aspect of any optimization is the degree in which it is rooted in real data. For our optimization, we directly analyze all 15723 course combinations pulled directly from the University of Kentucky's main sequel database, SAP HANA®. For each one of the respective course combinations, c_1 and c_2 , the pass rate was noted over the entire history of the course (data available since 2009).

Using the unique student identifier assigned by the university, we recorded the number of students that were enrolled in c_1 and c_2 in the same semester, subsequently finding the pass rate of those specific students. From this the *toxicity score* was calculated using the equation above:

$$\Pr\{X_{c_2}^p\} - \Pr\{X_{c_2}^p|X_{c_1}^p\} = \textit{Toxicity Score}.$$

If the toxicity score is greater than zero the combination is deemed to be toxic. As the score approaches 2, the more toxic the combination. Inversely, those scores approaching 0 are deemed to be the most synergistic:



An example data type and corresponding *toxicity score* are demonstrated when comparing EE 480, *Advanced Computer Architecture*, with EE 380, *Introduction to Embedded Systems* at the University of Kentucky.

Using this data we can calculate the *toxicity score* to be $= 0.81 - 0.61 = 0.2$. Considering this value is greater than zero we can deem the combination of EE 480 and EE 383 as being toxic if taken in the same semester.

In our optimization, this calculation is done across all course combinations with efforts to minimize the *toxicity score* per semester. Priority of avoidance is given to those courses

Table 3.2: Course toxicity relationship between two courses in the electrical engineering curricula

	enrollment	# passed	probability to pass
<i>EE</i> 480	145	117	0.81
<i>EE</i> 480 \cap <i>EE</i> 383	51	31	0.61

with highest *toxicity scores*.

Objectives

Methodology for the optimization was centered around the objective to minimize the total *toxicity score*, or *TS*, of each term. Considering that course combinations become more toxic as their toxicity scores increase, it is appropriate to minimize the score in each term in efforts to separate detrimental course combinations while combining those that are synergistic. Once each term is minimized the total minimized score of the degree plan is found by summing the score across all the terms. In each term, if n is the number of total courses in a curricula, and t the terms,

- *Objective 1 : Toxicity Minimization*

$$Term\ Toxicity = \min \sum_{k=1}^t \left\{ \sum_{i=1, j=1, i \neq j}^{n_k} TS(c_i, c_j) \right\}$$

This however cannot be the only objective of the optimization, as degree plan infeasibility could arise as course combinations are grouped together to form skewed terms with too many credit hours and inversely small terms with too few credit hours. In order to provide validity to the plan balancing credit hours across the terms must also be considered, where l is the term course load in credit hours and t is the number of terms in a curricula,

- *Objective 2 : Term Imbalance Minimization*

$$Credit\ Hour\ Imbalance = \min \sum_{i=1}^t |l_{i-1} - l_i|$$

While considering the minimization will drastically change the positioning of courses in the spacial array of the set maximum term length, it is necessary to make sure prerequisite linkage to their corresponding course is corresponding to a normal course progression. For example, the optimization would not be considered to produce acceptable degree plans of calculus II was separated by over 2 terms from calculus I. By this principle, our third objective minimizes the distance between the two requisites where e are the course edges, more specifically the requisite links between all the courses. We minimize the distance the course edges are placed such that the term where c_2 is placed is minimized relative to the term that c_1 is placed.

- *Objective 3 : Perquisite String Minimization*

$$\text{Perquisite String} = \min \sum_{i=1}^e (t_{c_2} - t_{c_1})$$

Constraints

There are many underlying principles of a degree plan that must be met even as we minimize the objective functions. Degree plans must be tightly regulated with respect to their course progression, course load, and requisite relationships. Considering these regulations we impose a variety of constraints to reinforce the efficacy and validity of the plan.

User Defined Limits

It is necessary for the user to define the general outline of what the degree plan will be. While optimal course placement is taken care of by the solver, it is necessary for the user to define the number of terms the sequence must fall under. Maximum and minimum number of credit hours per term must also be defined. These user defined limitation set a template for which the optimization may calculate.

Dynamic Constraints

During the optimization, course placement is cycled through every course, subject to various constraints. One important constraint to consider in a curricula is that of requisites.

Table 3.3: User defined limits for optimization

Limit	Description
Terms	Maximum allowable terms in degree plan
Maximum credits	Maximum allowable credit hours in a term
Minimum credits	Minimum allowable credit hours in a term

Throughout the optimization all prerequisite courses should be placed in terms preceding their target course such that,

- *Constraint 1* : If course c_2 has course c_1 as prerequisite:

$$\sum_{j=1}^n j * x_{c_1j} < \sum_{k=1}^n k * x_{c_2j}$$

In the case that the course is listed as a co-requisite, the requisite course must either be placed in the same term or precede the target course such that,

- *Constraint 2* : If course c_2 has course c_1 as co-requisite:

$$\sum_{j=1}^n j * x_{c_1j} \leq \sum_{k=1}^n k * x_{c_2j}$$

In the case that the course is listed as a strict co-requisite, the requisite course must be placed in the same term such that,

- *Constraint 3* : If course c_2 has course c_1 as strict co-requisite:

$$\sum_{j=1}^n j * x_{c_1j} = \sum_{k=1}^n k * x_{c_2j}$$

There are some courses that should not be moved in the optimization, for example, *Senior Design* should always fall in the last term before completion. This course along with many others, mainly in the first or last terms, are known as *fixed courses*. The creation of an array

designating *fixed courses* within a desired amount of terms assures they are held in desired term t ,

- *Constraint 4* : Fixed courses should remain in their designated term:

$$\text{Fixed Course Array} = \begin{pmatrix} fc_{1,1} & \cdots & fc_{1,t} \\ fc_{1,2} & \cdots & fc_{2,t} \\ \vdots & \ddots & \vdots \\ fc_{n,1} & \cdots & fc_{n,t} \end{pmatrix}$$

More specifically, we have defined the constraint of *Consecutive Courses*, that is two courses that should be taken in two consecutive terms, such as the case for *Engineering Capstone Design I* and *Engineering Capstone Design II* which should be taken consecutively in terms 7 and 8 respectively.

Methods

Before optimization were to commence course pairs across the entire university would need to be calculated and compared for their relative toxicity. The best way to go about this calculation would be the construction of a two dimensional array. This *Toxicity Score* matrix, or TS_{matrix} , can be constructed with TS indicting toxicity score of the course combination C_1 to C_n such that

Table 3.4: Toxicity score (Ts) matrix comparing relationships between all possible course combinations

	C_1	C_2	C_3	C_n
C_1	TS_{C_1,C_1}	TS_{C_1,C_2}	TS_{C_1,C_3}	TS_{C_1,C_n}
C_2	TS_{C_2,C_1}	TS_{C_2,C_2}	TS_{C_2,C_3}	TS_{C_2,C_n}
C_3	TS_{C_3,C_1}	TS_{C_3,C_2}	TS_{C_3,C_3}	TS_{C_3,C_n}
C_n	TS_{C_n,C_1}	TS_{C_n,C_2}	TS_{C_n,C_3}	TS_{C_n,C_n}

The diagonal of the matrix can be discarded considering a toxicity score of the identical course is extraneous.

Once all *toxicity score* combinations of the courses pairings are found, optimal placement of the courses relative to the other courses in a designated term are sought.

The solver must rotate through all possible combinations in efforts to find the optimal course placement relative to the other courses in the term. To accomplish this a binary matrix is created in order to assign combinations to a term. The size of the *x axis* is defined by the user defined limit t_m where m is the maximum number of terms. The *y axis* is the total number of courses in the curricula from C_1 to C_n where n denotes the total number of courses. The first iteration of the solver in the example curricula arbitrarily assigns courses as being placed in the in first term, denoted in binary classification as being 1. If courses were to be assigned to a subsequent term, their value in t_1 would be zero. This course positioning matrix is denoted by x such that

Table 3.5: Binary course placement matrix X

	t_1	t_2	\dots	t_m
C_1	{0, 1}	{0, 1}	\dots	{0, 1}
C_2	{0, 1}	{0, 1}	\dots	{0, 1}
\vdots	\vdots	\vdots	\vdots	\vdots
C_n	{0, 1}	{0, 1}	\dots	{0, 1}

Once we have created both the binary course positioning matrix x and the toxicity score matrix TS_{matrix} , the term vector in x is multiplied by each column in TS_{matrix} as to find the dot product, for the first term

$$Optimality Matrix = \begin{bmatrix} T_1 C_1 \in \{0, 1\} \\ T_1 C_2 \in \{0, 1\} \\ \vdots \\ T_1 C_n \in \{0, 1\} \end{bmatrix} \cdot (TS_{matrix})$$

This dot product will give us the toxic relationship for every permutation in every term unrelating to the courses chosen ad hoc by the solver. In order to specifically find the relationships for the courses selected by the solver as being related to any particular term, we then transpose the binary course positioning matrix x for the particular term of interest and multiply by the rows of the found *Optimality matrix*, finding the dot product such that:

$$Final\ Matrix = \left[T_1C_1 \in \{0, 1\}, T_1C_2 \in \{0, 1\}, \dots T_1C_n \in \{0, 1\} \right] \cdot (Optimality\ Matrix)$$

The sum of the toxicity values in the *Final matrix* make up the value we are attempting to minimize relative to the toxicity minimization objective. With respect to the load balancing objective, cross term comparisons of credit hour totals will be compared and discrepancies minimized. Optimal course placement will also be cross checked according to the edge minimization objective. These numerical analyses across all objectives are driven from the course placement found through the *Final matrix* calculation.

Gurobi[®] will run through however many iterations necessary in order to find the optimal solution relative to the objectives described.

With regards to objective two, *Term Imbalance Minimization*, finding the absolute value is necessary in order to adequately find the credit hour difference between each term irrespective of which term is of greater value. In order to accomplish this a series of manipulations must be considered. Let us consider the values for l_{i-1} and l_i in credit hours which we will designate a and b respectively. We iteratively find these differences across all the subsequent terms up to the limit defined by the user. This *credit hour difference* array will be noted by $y[i]$ such that,

$$y[i] \geq \begin{cases} a - b, & \text{if } a \geq b \\ b - a, & \text{if } b \geq a \end{cases}$$

Example Methodology

We consider finding the optimal course placement for a degree plan with a total of 6 courses of 3 credits each in a curricula, with a user defined limit of 9 credit hours in each term.







Term 1	Term 2
 C_1	 C_2
 C_3	 C_4
 C_6	 C_5

Figure 3.3: Example Curricula, Unoptimized

In order to find the optimal solution relative to the objectives given, the solver will iteratively progress through all possible combinations of courses such that the minimum is found. There are six possible course combinations that must be considered for the first term. Representative toxicity scores are given to these combinations. In production, these scores are extracted from the overall toxicity file which contains the pass rates from every combination at the institution.

Table 3.6: Possible course combinations in the first term of the example curriculum with representative toxicity scores

<i>Combination 1</i>	C_1	C_3	.24
<i>Combination 2</i>	C_1	C_6	.15
<i>Combination 3</i>	C_3	C_1	-.20
<i>Combination 4</i>	C_3	C_6	0
<i>Combination 5</i>	C_6	C_1	.1
<i>Combination 6</i>	C_6	C_3	-.16

The first iteration of the solver in the example curricula arbitrarily assigns courses C_1 , C_3 , and C_6 as being placed in the in first term, denoted in binary classification as being 1. These selected courses are noted in the course positioning matrix x as being associated with the first term.

Table 3.7: First iteration of binary matrix x of example curriculum

	t_1	t_2
C_1	1	0
C_2	0	1
C_3	1	0
C_4	0	1
C_5	0	1
C_6	1	0

Although defined by our algorithm with respect to the JuMP/Gurobi framework, the binary course positioning matrix x and the optimality matrix will not be output. These are iterative steps that transpire without user manipulation or need for data ingestion. Gurobi will proceed to run through however many iterations necessary to find the optimal result (There are cases in which no optimal result may be found).

Once the result has been found by Gurobi, the optimal course progression on a term by term basis may be output and visualized using the curricular analytics toolbox [11]. The resultant degree plan found from our example curricula is as follows:

Term 1	Term 2
1 C 1 C1	1 C 3 C3
1 C 2 C2	1 C 5 C5
1 C 4 C4	1 C 6 C6
Complexity: 3.0	Complexity: 3.0

Figure 3.4: optimized example curricula, objective of toxicity minimization

We will now consider the undergraduate electrical engineering curricula at the University of Kentucky, resolving separate degree plans relative to the objective sought.

Results

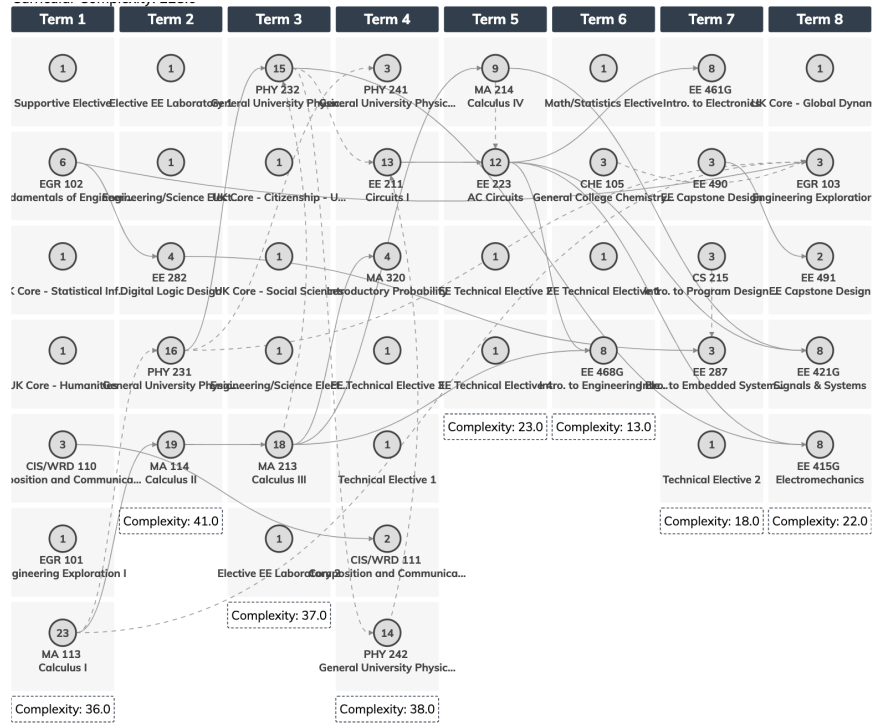


Figure 3.5: B.S. Electrical Engineering program at the University of Kentucky, 2018. Optimized using toxicity avoidance objective

Figure 3.5 displays the resultant degree plan after optimizing for toxic course combination avoidance. During the iterative solving for the plan, toxic course combinations were separating from residing in the same semester while synergistic courses were combined. All this was done subject to the constraints detailed (including honoring requisite relationships). However, it must be observed that the semesters are not balanced with respect to credit hours. For this we introduce the load balancing objective.

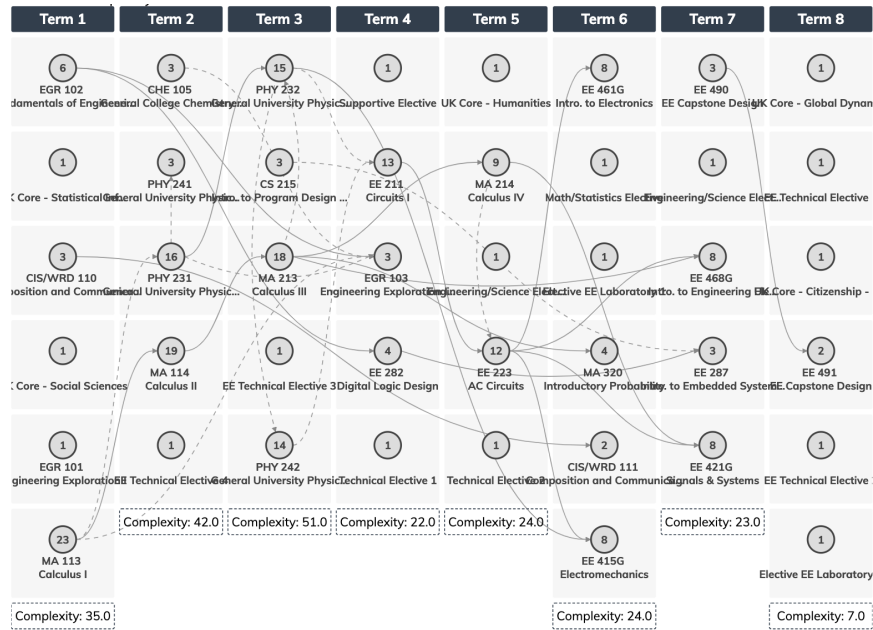


Figure 3.6: B.S. Electrical Engineering program at the University of Kentucky, 2018. Optimized using term imbalance minimization objective

Figure 3.6 displays the resultant degree plan after optimizing for term imbalance minimization. All semesters are now balanced term over term. However it must be pointed out that prerequisite strings can be sometimes extremely long. Such is the case between WRD110 and WRD111 which are separated by 4 semesters. In efforts to place requisite learning outcomes as close to the course which employs its use, we introduce the requisite string minimization objective.

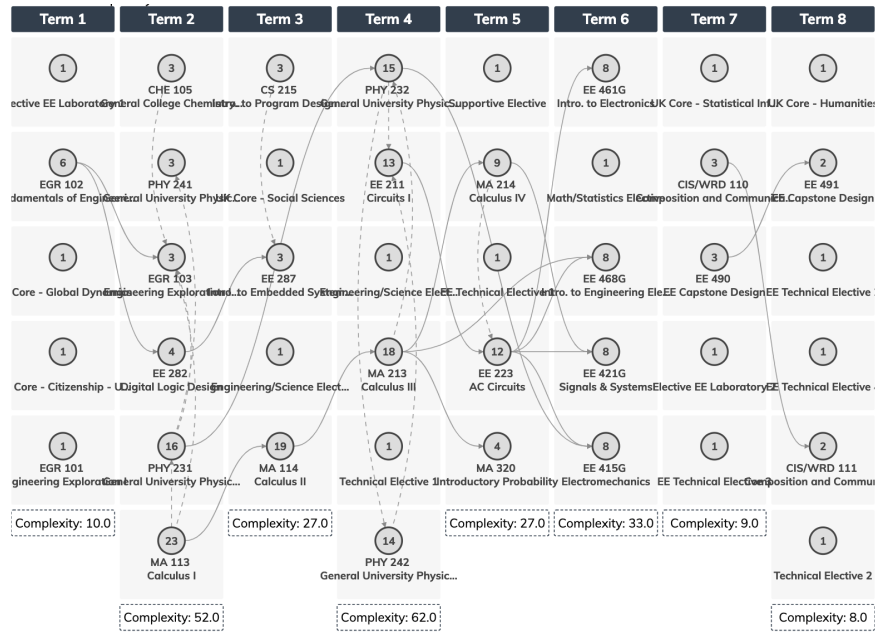


Figure 3.7: B.S. Electrical Engineering program at the University of Kentucky, 2018. Optimized prerequisite string minimization objective

Figure 3.7 displays the resultant degree plan after optimizing for prerequisite string minimization. All requisites are placed at most with one semester of separation. Now all objectives are combined to give the ultimate plan:

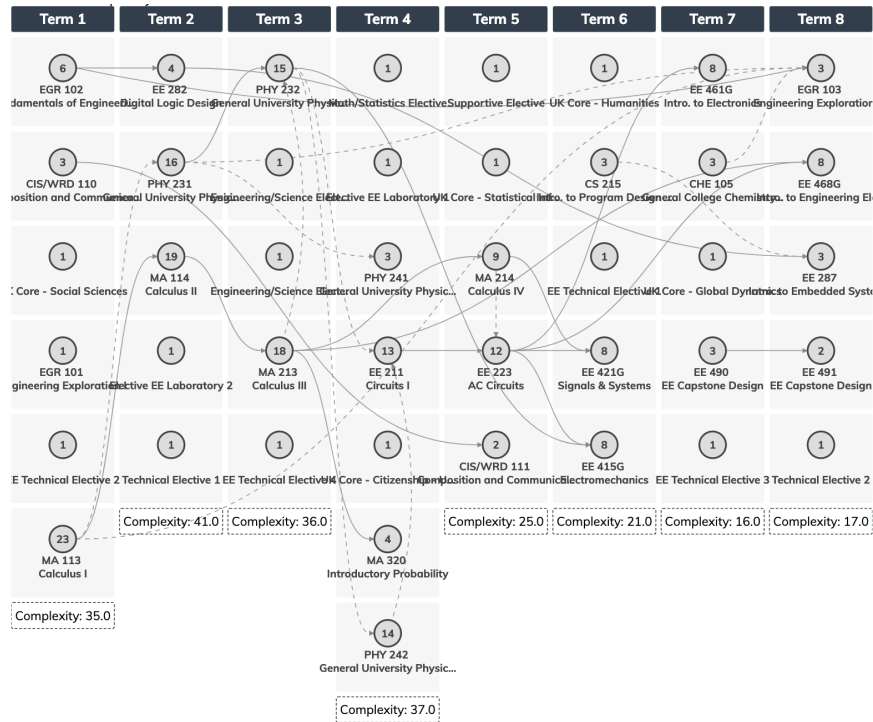


Figure 3.8: B.S. Electrical Engineering program at the University of Kentucky, 2018. Optimized using toxicity avoidance, term balancing, and prerequisite string minimization objectives

Figure 3.8 displays the resultant degree plan after optimizing for all three objectives, including term credit hour balancing, prerequisite string minimization, and toxicity avoidance. While the toxicity avoidance is vital in optimizing for student success, the degree plan introduces validity in the objectives of balancing course load and keeping requisite courses close to one another. The *fixed course array* constraint makes sure term specific courses are not moved during the optimization (e.g. Capstone Design in terms 7 and 8) while the other constraints, automatically checked for during the algorithm, insure all requisites are honored, all while falling in the user defined limitations.

This *ultimate* plan combines real world, institution specific pass rates to create the optimal schedule, thus providing a truly powerful tool for administrators across the world.

Chapter 4 Applications in Electrical Engineering

Electrical engineering is truly a beautiful field of study considering the expansiveness of its applications. Even though there are quite a bit of differences between nanoscale semiconductor devices and large megawatt power systems, fundamental electrical engineering principles hold true. In the same sense, applications of linear and non-linear optimization extend far beyond that of degree plan optimization. The same treatment, principles, and methodologies described can easily be translated to problems posed in the field of electrical engineering.

4.1 Power Transmission, Optimization of Damping Control

In power transmission engineering many design factors need to be accounted for including signal fidelity across long distances and power outage mitigation. Once such mechanism implemented to maintain signal fidelity are *Wide Area Damping Controllers*, or WADCs. Optimization techniques could be used with regards to the placement and signal allocation priority of these controllers in large multi-nodal power systems.

Stability in power systems is of key concern for many system designers. The ever-increasing amount of noise introducing or transient nodes in a system introduces many failure modes not previously warranting mitigation techniques. These failure/fault modes may be attributed to the large-scale implementation of renewable power generation technology by many power companies. In the presence of these large scale renewable systems across long distances arise the nuisance of small signal instability, specifically inter area oscillations. When this is the case strategically placed damping actuators throughout a system can be coordinated in such a way to dampen these oscillations. This is where optimization techniques could first be introduced, with regards to the placement of the controllers between the two optimal nodes to most quickly resolve the oscillations.

While novel modal-based control allocation techniques have been introduced and proven across a complex wind farm scenarios with healthy and affected actuators randomly dis-

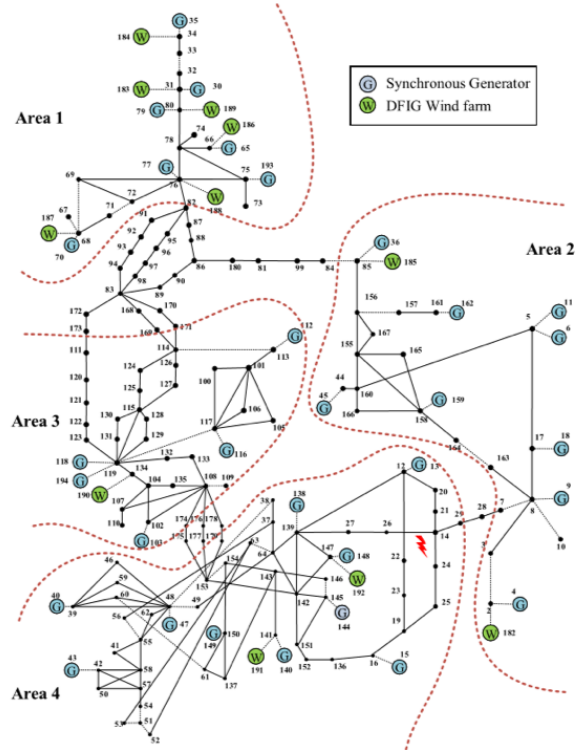


Figure 4.1: 192Bus WECC Windfarm

Table 4.1: Objective and constraints associated with inter oscillatory minimization

Objectives: Minimize Disturbance	Example Constraints
1. Optimal placement of WADC within power system	1. Geographic limitation
2. Priority of oscillatory damping signal	2. Limit on amount of WADCs
	3. WADC damping characteristics

persed throughout, optimization could be used to further the efficacy of the system [15]. Redundancy and protection in the face of these oscillation scenarios is crucial in maintaining a resilient power transmission system considering the increased use of renewable technology seen today.

Electro-mechanical oscillations between interconnected synchronous generators are phenomena inherent to power systems. The stability of these oscillations is of vital concern and is a prerequisite for secure system operation. For many years, the oscillations observed to be troublesome in power systems, were associated with a single generator, or a very closely connected group of units at a generating plant. Some low frequency unstable oscillations were also observed when large systems were connected by relatively weak tie lines,

and special control methods were used to stabilize the interconnected system. These low frequency modes were found to involve groups of generators, or generating plants, on one side of the tie oscillating against groups of generators on the other side of the tie [16]. In recent times, many instances of unstable oscillations, involving inter-area modes in large power systems have been observed, both in studies and in practice, such as in the western region of the United States. Low frequency synchronizing oscillations (particularly around 0.1 hertz) between the Pacific Northwest and Pacific Southwest have long been a characteristic of the western power system. These oscillations were primarily caused by the negative damping effect of hydro governors on the swing mode between the two regions, which were connected by a weak system of 230 kV inter-ties [17]. Such oscillations are increasingly becoming a cause of concern. This has led to a renewed interest in the nature of these modes, methods for systematically studying them, and control optimization methods by which they can be stabilized. In the face of ever increasing deployment of renewables into the modern power system, the need for damping of low frequency oscillations will only increase. Optimization techniques will prove invaluable as this problem becomes more and more prevalent, mitigating the threat of power system disruption across the world.

4.2 Smart Grid

Another application where optimization methods could easily be applied is that of the smart grid. We are living in the midst of a significant technological transition, away from the passive electric power transmission and distribution system to a connected and resilient platform leveraging many emerging and proven technologies. New telemetry and long distance real time data acquisition techniques allow for the implementation of more concise load generation, leading to less waste and a more ecologically conscience utility company. Granular load monitoring at the local customer level allows for specific generation benchmarks in relation to a variety of different economic and meteorological conditions. Monitoring devices are able to be tied together using common communication protocols creating an ‘internet of things’ environment. The adjoining of these ‘connected’ components to a centralized data acquisition hub where data driven decisions can be made has given rise to

the term ‘Smart Grid’. Now that a prevalent amount of data exists, optimization techniques can be employed across a variety of use cases.

Table 4.2: Objective and constraints associated with smart grid topology

Objectives: Maximize Power throughput	Example Constraints
1. Load matching	1. Capital expenditure
2. Transient disturbance mitigation	2. Transient characteristics
3. Outage minimization	3. Power output and storage

Optimization techniques can be deployed using the data collected from the smart grid to minimize over production of power. During load transients, the optimal amount of power can be produced for the varying power requirements. This in essence would provide “load matching”, so that the power generated can more closely follow the load required by the system. Also, regulators can be constructed for line performance characteristics at optimized placements though out the grid, much in the same way that was discussed with respect to WADCs .

Another area that the optimization techniques can be deployed is with respect to outage mitigation, or “self healing” systems. In order to insure the most customers have their power outage resolved (e.g. after a catastrophic storm), an optimization can be run with respect to the locations in which crews should address first. This will insure that the most customers benefit from the limited resources available by the utility company.

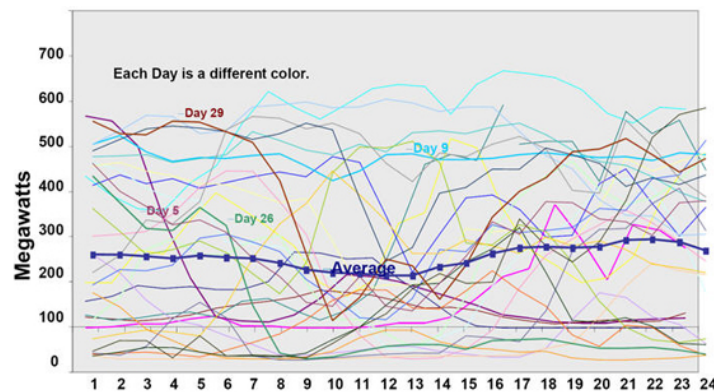


Figure 4.2: Variability in power output of wind farm by day, for one month [18]

When thinking about renewable energy as an integrated power generation member of the smart grid system, power output variability must be taken into consideration in relation to

non-predictable energy output (wind meteorological duty cycle variance). This variance gives rise to key smart grid concepts such as load forecasting and load scheduling. Wind farms are known for the ability to reliability to produce adequate power for 40% of the hours in a year, yet it is very difficult to predict when those hours will occur [18].

With “connected (IoT)” solid-state relay devices to a centralized data center that can monitor real time meteorological conditions, optimization algorithm development can be designed to open or close specific sectors of a wind farm in order to maximize renewable energy production or minimize low wind speed ‘cut-in’ energy waste.

Chapter 5 Conclusions and Moving Forward

The ramifications that curated degree plans have on incoming students are often not fully grasped. The plan in essence will dictate the students attempt to follow his or her dream to graduate from their chosen institution. The degree plan represents trials and tribulations, long nights in the library, and often unimaginable stress levels. The degree plan epitomizes students' futures, their successes, and their failures. All told, the degree plan may be the single most powerful advising tool in higher education.

In that, administrators must deeply analyze and quantitatively assess their plan before they release to students. This thesis attempts to solve for the ultimate student success pathway leveraging the most advanced commercially available solver with optimization algorithms tailored to creating valid, meaningful degree plans. An easy to use user interface has been created in order to facilitate the use of these powerful algorithms (*main.optimizeplans.com*).

Moving forward, much work remains to be done. Ongoing efforts include the creation of degree plans tailored to the particular student. This will include using their ACT score, high school GPA, and demographics to create a degree plan custom to them. It will leverage trained machine learning models to find areas of concern, basing problematic "stop out" areas relative to historical student data of students with similar backgrounds.

As administrators, the obligation exists to provide the best probability of success to all students, irrespective of their background. It is essential for the future and vitality of the American economy and way of life. Until this probability of success remains equally high for all students who, through hard work, dedicate their lives to the completion of their degree, work remains to be done.

Onward.

Bibliography

- [1] Elka Torpey *Measuring the value of education,” Career Outlook, U.S. Bureau of Labor Statistics* April 2018.
- [2] Hemelt, S. W., and Marcotte, D. E. *The Impact of Tuition Increases on Enrollment at Public Colleges and Universities. Educational Evaluation and Policy Analysis*, 33(4), 435–457. <https://doi.org/10.3102/0162373711415261> 2011
- [3] Bureau of Labor Statistics *College tuition and fees in U.S. city average, all urban consumers, not seasonally adjusted.* https://data.bls.gov/timeseries/CUUR0000SEEB01?output_view=data.
- [4] Gregory L. Heileman, Chaouki T. Abdallah, Ahmad Slim, and Michael Hickman. *Curricular Analytics: A Framework for Quantifying the Impact of Curricular Reforms and Pedagogical Innovations*. Albuquerque, NM., 2018.
- [5] Slim, A. *Curricular Analytics in Higher Education*. PhD thesis, University of New Mexico, Albuquerque, NM. 2017
- [6] Heileman, G. L., Thompson-Arjona, W. G., Free, H. W., and Abar, O. *Does Curricular Complexity Imply Program Quality?* Lexington, Kentucky, 2018.
- [7] Gill, P. E., Murray, W., Wright, M. *Practical Optimization* San Diego, California., 1981
- [8] Ünal, and Uysal. *A New Mixed Integer Programming Model for Curriculum Balancing: Application to a Turkish University*. European Journal of Operational Research (2014)
- [9] J. Bezanon, S. Karpinski, V. Shah, and A. Edelman. *Julia: A fast dynamic language for technical computing*. In Lang.NEXT, Apr. 2012.
- [10] Hickman, Michael. *Masters Thesis* Albuquerque, NM., 2017.
- [11] Heileman, G. L., Free, H. W., Abar, O. and Thompson-Arjona, W. G. *CurricularAnalytics.jl Toolbox*. <https://github.com/heileman/CurricularAnalytics.jl>
- [12] Dunning, I., Huchette, J., Lubin, M. *JuMP: A Modeling Language for Mathematical Optimization* SIAM Review 2017
- [13] Jablonský, J. *Benchmarks for Current Linear and Mixed Integer Optimization Solvers* <http://dx.doi.org/10.11118/actaun201563061923>, 07 Number 6, 2015
- [14] *Gurobi Software Drives Award-winning Inmate Assignment Project*. MS Today 44.6, 2017

- [15] M. E. Raoufat, K. Tomsovic and S. M. Djouadi *Dynamic Control Allocation for Damping of Inter-Area Oscillations* IEEE Transactions on Power Systems, vol. 32, no. 6, pp. 4894-4903 Nov. 2017
- [16] Klein, M., Rogers, G.J., and Kundur, P. *A fundamental study of inter-area oscillations in power systems* IEEE Transactions on Power Systems, vol. 6, no. 3, pp. 914-921 Aug 1991
- [17] Cresap, R. L., and Hauer, J. F. *Emergence of a New Swing Mode in the Western Power System* IEEE Transactions on Power Apparatus and Systems, vol. PAS-100, no. 4, pp. 2037-2045 April 1981
- [18] Shaffer, Walter. *The Role of Smart Grids in Integrating Renewable Energy* ISGAN Synthesis Report www.nrel.gov/docs/fy15osti/63919.pdf.

Vita

William G. Thompson-Arjona

Education

- B.S. Bioelectrical Engineering, Marquette University, Milwaukee, WI, 2015

Appointments

- Graduate Research Assistant, Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY, 2017 – Present.
- Associate Development Engineer, Power Control Business, Rockwell Automation, Milwaukee, WI, 2015-2018
- Undergraduate Teaching Assistant, Circuits I/II Laboratory, Marquette University, Milwaukee, WI, 2014-2015
- Undergraduate Intern, Enthermics Medical Systems, Milwaukee, WI, 2014
- Undergraduate Intern, Cardiovascular Innovation Institute, Department of Regenerative Medicine, Louisville, KY, 2013

Product Development

- KYdegreeplans.com
Centralized degree plan creation, visualization, and analytics tool used by the Center for Postsecondary Education across the state of Kentucky
- CanBad Autonomous Weighing Systems, Provisional Patent 119769-1
Designed autonomous weighing systems complete with patent pending hardware. This included custom PCB with PIC32MZ as MCU interfacing via UART to load cell, SPI to FDTI processor (display), and Bimba/PIAB pneumatics.
- PowerFlex 755T, Allen Bradley, Rockwell Automation. 2015-2018
Key contributor for successful product launch of state-of-the-art variable frequency drive for heavy industry motor control. PCB design and implementation of three phase fault and inbalance protection.

- Virtual Interactive Patient, Medtronic, 2015
Senior design project. Programmed FPGA to test pacemakers/left ventricular assist devices for robustness against ECG/EKG abnormalities