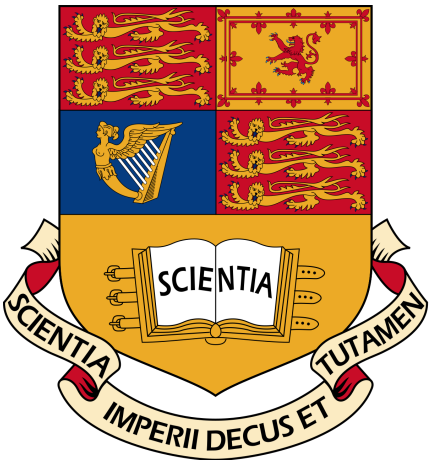
Imperial College London

Department of Electrical and Electronic Engineering

Final Project Report 2019



Project Title:	**Resource Allocation via Integer Programming**
Student:	**Mwanakombo Hussein**
CID:	**00936105**
Course:	**EIE4**
Project Supervisor:	**Prof. Alessandro Astolfi**
Second Marker :	**Dr. Fei Teng**

Acknowledgements

I would like express my sincerest appreciation to my project supervisor Prof. Alessandro Astolfi for agreeing to supervise my impromptu self-proposed project. For without his guidance, support, and insight throughout the duration of the project, this project would not have been realized.

Furthermore, I would like to thank Dr. Fei Teng for his time and careful consideration in the evaluation of this project.

I would also like to thank Dr. Thomas Clarke for his assistance in ensuring I worked on a project I'm passionate about.

Finally, I would like to thank my family for their love and support throughout my studies. Without them, I would not have had the opportunity to attain a fulfilling education at Imperial College London.

Abstract

This paper takes a Binary Integer Programming (BIP) approach to solving the general resource allocation problem faced in various industries such as manufacturing in production planning and finance in portfolio selection. The goal is optimizing the allocation of finite resources to specific tasks to minimize cost or maximize profit. This problem is widely approached using Linear Programming, as such, the challenge is tackling the same problem with a narrower scope of BIP which has been practiced far less often in literature as it constricts the problem to binary variables. As a paradigm in this general scope, the focus is the 'Student Project Allocation' problem faced ubiquitously in higher education institutions where individual projects are often a graduation requirement. Initially, this problem considers the preferences of the students solely, and later incorporates the lecturers' capacities of supervision. This paper shows that utilization of Integer Programming performs faster and more efficiently, yielding higher rates of both student and lecturer satisfaction compared to existing methods used to solve resource allocation problems such as gradient descent algorithms, the Hungarian method, and complete enumeration. Finally, this paper highlights the applications of the aforementioned method in other industries and related challenges with this approach.

Table of contents

List of figures

List of tables

Nomenclature

Greek Symbols

β_k	total number of projects supervised by lecturer k
χ_j	total number of workers/residents allocated to job/hospital j
$\eta_j^{T_{PS}}$	lecturers' acceptable supervision
π	shadow price
τ_1	student satisfaction
τ_2	lecturer satisfaction
Z	difference between $\eta_j^{T_{PS}+1}$ and $\eta_j^{T_{PS}}$

Input Parameters

T_C	total number of project choices/preferences
T_L	total number of lecturers
T_P	total number of projects
T_{PS}	maximum number of projects supervised by each lecturer
T_S	total number of students

Superscripts

i	student , $i \in I = \{1, \ldots, T_S\}$
j	project, $j \in J = \{1, \ldots, T_P\}$
k	lecturer, $k \in K = \{1, \ldots, T_L\}$

Acronyms / Abbreviations

AJSSP Assembly Job Shop Scheduling Problem

AP Assignment Problem

CBC Coin-or Branch and Cut

EEE Electrical and Electronic Engineering

EIE Electronic and Information Engineering

GPS Global Positioning System

HR Hospital/Residents

BIP Binary Integer Programming

IP Integer Programming

NRMP Job/Worker

LP Linear Programming

MAX-SPA-P Maximum stable matching in SPA-P

MILP Mixed Integer Linear Programming

MLP Mixed Linear Programming

NP Non-deterministic Polynomial-time

NRMP National Resident Matching Program

P Polynomial-time

RHS Right Hand Side

SPA-P SPA with Preferences over Projects

SPA Student Project Allocation

TF Taxi Firm

TP Transportation Problem

Chapter 1

Introduction

The resource allocation problem is one faced ubiquitously across varied industries whereby the challenge faced is the optimal distribution of finite resources to competing tasks in order to minimize cost or maximize profit. The design of logistic distribution systems is one of the most critical issues in industrial facility management commonly referred to as the facility location problem [2]. The facility location problem is defined in which a company such as Amazon has one set of demand points, and a set of potential facility locations, with specified costs for opening. The decision of placing an Amazon fulfillment center in a new location requires considering the trade-offs between the revenue considerations as local customers are exposed to sales tax and the costs minimized from reducing the shipping distance to those customers [3]. As such the aim is to optimize the choice of which facilities to open so as to minimize the cost of fulfilling a customer (demand point) from a facility location. Within the energy industry, we observe a resource allocation problem researched by Klauw et al. in decentralized energy management. Klauw et al. focused on devices that schedule their flexible load profile based on steering signals received from centralized controllers whereby the objective is finding optimal device schedules given the steering signals. Other electronic and electrical engineering resource allocation problems are in Quality of Service (QoS) management [4], Device-to-Device (D2D) communication [5], and computer resource allocation [6] as archetypes. Excluding electronics, other applications include load distribution, production planning, portfolio selection, and apportionment [7].

In the education industry, public school systems require an automated system to allocate students to public high schools such as the 'New York City Department of Education' mechanism that matches approximately 90,000 entering students to public high schools each year in the city [8]. This challenge transitions to university systems as graduating students when they submit university applications that Gale and Shapley first realized in their

paper ‘College Admission and the Stability of Marriage’ [9]. Gale and Shapley discuss the challenges of colleges considering a set of students with a constraint of admitting a certain quota incorporating intricacies such as the likelihood of students accepting university offers, impact of wait-listing students and the uncertainty of the students’ university preference. In the UK, the likelihood of students meeting their conditional offer would be included. In France, the ‘French Ministry of National Education’ publicizes the positions that need to be filled in the departments of its many universities as graduating students submit applications [10]. For each position, the university strictly ranks the candidates in order of preference and informs them of their respective rankings; the applicants are then required to express their preferences among the positions offered. The objective is to make a stable assignment between institutions and students alike. A similar approach is implemented in Turkey, however the universities base their student ranking solely on the academic performance of the graduating students on a series of national examinations [11]. On a granular scope in higher education institutions, Tounsi discusses in his paper resource allocation problem that arise in the scheduling of resources for both courses and examinations every year. This allocation problem is complex due to various factors such as a large number of students, module/course options, required examinations and invigilator constraints. As such, his paper concludes that solving this problem requires an automated resource allocation system that can produce feasible and effective timetables [12]. What these cases have in common is the necessity of an efficient, automated system or model that can handle the allocation process for a large number of students or variables in a timely manner.

As an archetype, this paper aims to apply BIP, from here-on-out denoted as BIP to optimize the student project allocation problem using the Electrical and Electronic (EEE) Department at Imperial College London as a case study. The student project allocation problem seeks to optimally allocate students to projects whereby the student, project and lecturer relationship is depicted in Figure 3.1 in Section 4. Each student can only be allocated to one project thus forming a one-to-one relationship between students and projects whereas, each lecturer can propose and supervise multiple projects forming a one-to-many relationship between lecturers and projects/students. The allocation of the resources (projects/lecturers) to the tasks (students) is formulated as an assignment problem.

The student allocation problem is faced by countless universities as research projects are usually a requirement for graduating purposes at higher education institutions [13]. The allocation of these research projects is usually modeled with a set of lecturers, students, and projects. Usually, the lecturers at these institutions propose available projects, then the students choose the projects they deem acceptable as their project preferences. In some cases,

lecturers can also choose the students they deem acceptable to pursue their project. Dependent on the institution or department certain constraints are imposed on this optimization problem such as a maximum number of projects each lecturer may supervise or the minimum number of projects a student must choose as a preference. The paper here-on-out refers to the student project allocation problem as the 'SPA problem' [14].

The Electrical and Electronic Engineering Department at Imperial College London, aims to automate the allocation of projects to students, due the large number of students in the graduating class and in the process, optimize the project allocation algorithm. Although the current algorithm is automated, as realized, some aspects of the optimization are hard-coded to account for certain decisive factors such as the splitting of popular projects or consideration of over-burdened lecturers hence rendering the SPA algorithm semi-automated. The need to hard code certain constraints is a challenge as certain lecturers and students require a different allocation objective function without altering the current allocation solution. Furthermore, an allocation process that is not fully automated takes a much longer time to yield allocation results due to the manual intervention of re-coding certain aspects or changing particular constraints on-demand.

To that end, the principal aim of this paper will be to establish mathematical models that automate the allocation process allowing all students to be optimally allocated their preferred projects whilst considering university or departmental constraints effectively without any manual interference. Additionally, the paper will not only enhance and implement the new established mathematical models, but also identify and implement known models in literature that currently strive to solve the SPA problem to effectively gauge the performance of the approach followed in this paper .

1.1 Project Outline

Following the resource allocation and SPA introduction, Chapter 2 comprises of background research that provides context on existing approaches, algorithms and perspectives relevant to the student project allocation problem. This chapter will also discuss the project requirements outlining the objectives and approach as well as the deliverables.

Chapter 3 will delve into the formulation of the SPA mathematical models using BIP where dynamic programming will be used into implement objective functions with multiple goals striving to meet the challenge of fully automating the optimization of SPA problem.

Chapter 4 will discuss the techniques of implementing optimization models and the challenges of the implementation process.

Chapter 5 will test the SPA BIP implementation to verify the models function as expected. The tests will be done in Python using the PuLP optimization library where the source code for the optimization models can be found in the appendix.

Chapter 6 will analyze the optimal results gained from the SPA integer programming models. These integer programming SPA models results will be compared to the actual allocation algorithm used in the 2018/2019 EEE Department for allocating projects to the 2019 graduating class. This section will compare the allocation from the realistic departmental algorithm and the SPA models evaluating which results in higher student satisfaction and lower lecturer supervision workload.

Chapter 7 will evaluate the SPA model results against the current departmental implementation and widely used algorithms. This evaluation will narrow-down which approach performs better and whether the BIP method explored in this paper meets original objectives.

Chapter 8 will discuss the manner the SPA problem formulated in the paper can be applied to additional applications of the SPA problem outside the university arena in a much broader context. These additions aim to depict the adaptability and impact of the SPA problem in solving other real-world assignment problems using the mathematical models formulated in this paper.

Chapter 9 will discuss any further work that would be interesting to explore given more time based on the achievement and learning points of this paper.

Chapter 10 will discuss the conclusion of the paper summarizing the achievements, the challenges, and the impact of solving the SPA problem using BIP.

1.2 Project Specification

The paper is oriented to answering the following questions. “Can the SPA problem formulated in this paper:”

- be computationally faster?
- be fully automated requiring no human intervention?

- perform better resulting in a higher student satisfaction rate compared to the current algorithm and existing methods?
- perform better resulting in a higher lecturer satisfaction rate compared to the current algorithm and existing methods?
- support multiple objectives when required?
- dynamically support a dual goal objective function?
- be extended to different assignment problems in various industries such as Transportation?

1.3 Project Requirements

This chapter comprises of the project requirements and deliverables to be achieved in solving the SPA problem via BIP using the EEE Department at Imperial College London as an exemplification.

The project will deliver Integer Programming optimization models that will optimize the Student Project Allocation (SPA) problem specifically for individual student projects as opposed to group projects. The SPA model in this paper will focus on the variant whereby lecturers propose available projects and students provide a strict order of preference over the proposed projects. The paper will not explore the SPA model whereby lecturers have preferences over the students who've shortlisted their proposed projects as it is a model that is not implemented often in higher education institutions for reasons discussed in Chapter 1. The paper will then introduce complexity by further defining the SPA problem by considering constraints such as lecturer capacity and distinct decision factors.

1. **Mathematical Formulations**
   The optimization models explored will be described with mathematical formulations that aim to optimize the SPA problem based on various objective functions that a university may want to achieve, such as:
   Necessary:
   (a) Allocate projects to students solely based on their preferences.
   (b) Allocate the highest preference project to as many students as possible based on their preference rankings.

(c) Allocate the highest preference project to as many students as possible whilst putting an upper bound on the number of projects each lecturer can supervise.

Desirable:

(d) Minimize the number of projects each lecturer supervises.

(e) Allocate the highest preference project to as many students as possible whilst minimizing the number of projects each lecturer supervises.

(f) Possible further models that may emerge from analysis and further literature research.

2. **Constraint Controls**
Distinct constraint controls will be analyzed to gauge how minor changes to certain constraints impact the optimal allocation results and objective function behavior. The primary controls will be the student's project preference methodologies where different project preference approaches will be explored and varied throughout the paper. By student preference methodology, we refer to the manner a student creates a project preference list, which could be one of the following:
Necessary:

(a) A set of project preferences with no ranking (thereby, of equal ranking).

(b) A set of project preferences with ranking indicated by a fixed range of integers. For example, a project ranked 1 would be a student's highest preference whilst a project ranked 5 would be a student's fifth preference.

Desirable:

(c) A set of project preferences with ranking indicated by weightings in percentages. For example, a set of project preferences weighted: $50\%, 30\%, 10\%, 20\%$, would have a respective proportional impact on the allocation algorithm. Hence from above, the student indicated that their highest preference project is the project weighted 50%. Hence, this project should have a greater impact on the allocation algorithm than any of the other project preferences as their weightings are below 50%.

3. **Model Simulation**
Each model will be simulated using MATLAB and/or Python's optimization library PuLP for simulation purposes with test data-sets. The simulations are to test and verify whether the designed model functions as expected. Once verified, the SPA

models will be implemented using realistic data-sets used in the 2018/2019 EEE/EIE Final Year Project allocations. Python's optimization library PuLP is well designed to enable the utilization of multiple solvers using the same code such as Gurobi [15], CPLEX [16], and GLPK [17] which are commercial and capable of solving large scale problems. All the solvers mentioned have detailed results that provide information such as problem solve time, upper-bounds, feasibility status which allows for further analytical diagnosis on the solutions found compared to the default PuLP solver (CBC [18]). For these reasons, Python will be the primary programming language used for simulation of the optimization models of the SPA problem as opposed to MATLAB.

4. **Analysis**
   Each model and its respective allocation solution will be analyzed in order to establish a performance criterion to rank the effectiveness of each model with respect to their objective function. The result from the analysis will be compared to the performance and allocation of the 2018/2019 EEE Departmental student allocation algorithm to highlight whether the BIP approach performs better than the current departmental implementation.

   Analysis aspects for each Model :

   - *Student Satisfaction*: The number of students allocated either of their top 3 project preferences.
   - *Lecture Satisfaction*: The number lecturers allocated 3 or fewer projects to supervise post allocation.
   - *Execution Time*: Computation time for the program to run and yield a feasible optimal allocation output.

5. **Optimal Solution Analysis**
   Post optimal analysis should yield the most effective models and these models will utilize the realistic data-sets used during 2018-2019 EEE/EIE Final Year Project allocations as a proof of concept in solving the SPA problem efficiently and optimally. The most effective models will be applied to distinct assignment problems countered in various industries seen in Chapter 8.

6. **Project Extension: Group SPA Problem**
   The extension is implementing the BIP SPA approach to a group allocation case whereby students list project preferences, and multiple students are allocated to projects forming group project teams. This case considers distinct constraints such as module prerequisites and group maximum and minimum student quotas.

Chapter 2

Background

This chapter comprises of background research that provides context on existing approaches, algorithms and perspectives relevant to the student project allocation problem. It also delves into optimization background relevant to the SPA problem such as linear programming, integer programming, and the assignment problem.

2.1 Literature Overview

The generalization of the SPA problem is the allocation of junior doctors to hospitals nationwide in the US [19] coined as the Hospitals/Residents problem (HR) [20]. The HR problem is a two-sided matching problem where the aim is to form a stable matching of residents to hospitals considering constraints such as hospital capacities, the preference of residents over hospitals and vice versa [14]. As such, the generalization of the SPA problem can be applied to several real-world challenges such as the allocation of students to secondary schools and higher education institutions across Europe [21]. Hence, exploring the optimal solution of the SPA problem would have a wide reaching impact.

There are two Student Project Allocation problem models recurrent in research: the first simply considers the student's preferences [22][23][24] and the second incorporates the preference of the lecturers as well [25][26]. The first variant views the SPA problem as a specific variation of the generalized assignment problem whereby we aim to assign X students to Y projects, [13] and the second variant as a two-sided matching problem, where the challenge is allocating a set of agents to another set of agents based on preferences [27][26]. The latter scenario is one where the SPA problem defines that students have preferences over proposed projects, whilst the lecturers have preferences over the students. It

has been proven when both students and lecturers have preferences, then the critical property the matching should satisfy is stability [14].

An interview conducted with the Head of Student Project Allocation at the EEE department at Imperial College London, Dr. Tom Clarke, expresses it is not desirable to impose a constraint whereby lecturers have preference over students. The reason being it imposes unfair project allocation conditions on certain students if the lecturer's student preference, for instance, is deemed solely on academic merit. It is simple to deduce that students who are less academically inclined would be less likely to be allocated their preferable projects if the projects are also preferable to more academically inclined students. This reasoning is also seen at other institutions such as the University of Manchester [28], where a survey conducted showed that there is a high student dissatisfaction rate experienced especially by less academically-inclined students with the latter approach. This two-sided matching introduces new and distinct complications such as what factors will be used to derive the lecturer preference list and how will the factors affect the optimal allocation solution without bias. However, it should also be considered that exceptions can be made to this situation and certain projects would require a two-sided matching approach as occurred with one project proposal during this year's project allocation process for the EEE/EIE 2019 graduates. This is attributed to the fact that some projects require certain skills and prerequisites of knowledge in which case lecturers would benefit from having a preference list on the students they deem acceptable to take on their projects. Whether or not lectures have preferences over students depends on different institutions and departments alike. As mentioned in the project specification, this paper will focus on the variant of the SPA problem where the student imposes a preference for projects only. Hence, the SPA problem in this paper will not be a matching problem but a specific case of the assignment problem [28].

The higher education institutions such as the Department of EEE at Imperial College London strive to automate the SPA process as much as possible if not achieve complete automation. This need to automate the allocation process is also observed at several other universities and their departments such as the Department of Computing at Imperial College London, the School of Computing Science at the University of Glasgow [24], the Department of Computer Science at the University of Manchester [29], School of Chemical Engineering and Analytical Science at the University of Manchester [28], and the Department of Computing Science at the University of York [30] to mention but a few. Automating the SPA process reduces the time and effort required to obtain a feasible solution enabling re-evaluation and re-optimization of the allocation solution when required. Automation is a necessity with

solving the SPA problem and the use of BIP will effortlessly enable the modeling of logical relationships and decisions.

Of the two most recurrent SPA models in research, Anwar and Bahaj follow the first SPA model mentioned only considering the student's preference [22]. They propose two models that followed an Integer Programming from here-on-out denoted as IP, approach in their paper. The first IP model, is an individual project model that requires students to select a subset of the proposed projects and rank their choices. This model aims to make a lecturers' workload in the supervision of individual student projects as even as possible and also strives to allocate students their first project preference. This was implemented following a dynamic programming approach that defined two models: one that minimized the lecturer workload and another that maximized the student preferences, in that order. The two models were combined sequentially such that the objective function result of the previous model acted as a constraint for the following model. It was concluded that this dynamic programming approach was effective in achieving two distinct objective functions. The second IP model, is a group project model that has the same students requirements as mentioned in the individual model and by optimally allocating each individual student a project, implicitly allocates projects to groups of students. It is worth noting that although the SPA problem is one that has been researched throughout the years, the concept of approaching the SPA problem using integer programming is seen rarely in research.

It is observed that Anwar and Bahaj combined two distinct objective functions to achieve a dual goal objective function in their first IP model. Pan et al., expanded on the multi-objective concept by defining a goal programming formulation for the SPA problem with three objective functions [31]. The three objective functions were as follows: 1) maximizing the number of students allocated, 2) maximizing students' satisfaction and 3) maximizing supervisors' satisfaction, in that order. Similar to Anwar and Bahaj's approach, these three objective functions were sequentially solved considering the previous model's objective function solution as the constraint of the following model.

Other researchers follow the second SPA model in research viewing the SPA problem as a matching problem and strive to find stable matching between students and projects such as Abraham, Irving and Manlove [26]. They proposed two algorithms to find a stable matching in the SPA problem such that students have preferences over projects, whilst lecturers have preferences over students. The first algorithm produced a stable student-optimal matching whereby students are allocated the best projects possible in any stable matching while the second algorithm produced a stable lecturer-optimal matching whereby lecturers have the best possible students in any stable matching. Another paper that viewed the SPA problem as a

matching problem is Manlove and O'Malley's paper, however, differently to Abraham et al.'s approach, they explored a SPA problem where both students and lecturers have preferences over projects (SPA-P), which is disparate to the perspectives usually considered in research.

In the SPA-P variant of the SPA problem discussed by Manlove and O'Malley, lecturers rank in strict order of preference the projects that they offer implying that each lecturer is indifferent among the students who find acceptable a given project that they offer [14]. With this distinct approach they discover that stable matchings can have different sizes, and prove that finding the maximum cardinality stable matching referred to as MAX-SPA-P is NP-hard and not approximable within δ, for some $\delta > 1$, unless P=NP [14]. Lastly, they give an approximation algorithm with a performance guarantee of 2 for MAX-SPA-P. This supports that an integer programming approach is more effective, as it would result in an optimal solution rather than an approximation especially for problems that are NP-hard. Hence, reiterating why following an integer programming approach is far more effective when solving the SPA problem rather than a linear programming approach.

Whereas Manlove and O'Malley proposed the SPA-P problem and provided a polynomial-time 2-approximation algorithm, four years later, their approximation algorithm was improved by Iwama, Miyazaki, and Yanagisawa [32]. This improvement yielded an approximation algorithm with a performance guarantee that has an upper bound of 1.5 and a lower bound of $\frac{21}{19}$ for MAX-SPA-P [33]. This development on Manlove and O'Malley's algorithm meant given P=NP, the MAX-SPA-P is not only approximable within δ, for $\delta > 1$ but instead is not approximable within $\frac{21}{19} - \alpha$, for any $\alpha > 0$. As seen from Manlove and Iwama, the existing algorithms for MAX-SPA-P only guarantee to produce an approximate solution. Hence, Manlove, Mline, and Olaosebika [27] utilized another technique to enable MAX-SPA-P to be solved optimally by using integer programming.

BIP is utilized frequently in the optimal scheduling of courses i.e course timetabling in universities as explored by Bakir and Aksop [34], Tripathy [35] and Phillips et al. [36] however rarely seen in the optimal assignment of students in the SPA problem in literature. Other than the IP SPA approach Anwar and Bahaj [22] brought to the table, Manlove, Mline and Olaosebika [27]'s integer programming approach to the SPA problem is the only other IP SPA paper I could acquire in research. Their IP SPA model enables MAX-SPA-P to be solved optimally and present a correctness result. By comparing the feasible solutions produced by the approximation algorithms and the optimal solution produced by their IP model w.r.t the size of that table matching constructed, their paper shows that the $\frac{3}{2}-$approximation finds stable matching that is very close to maximum cardinality. Hence, showing that an IP model can run practical SPA-P instances, to find maximum cardinality matchings that admit

no blocking pair. The rarity of solving the SPA problem using BIP in literature creates an exciting and novel prospect in solving the problem that will be explored in this paper.

Finally, R. Calvo-Serrano et al. proposed a mixed-integer linear programming (MILP) model for the SPA problem that poses a drastically different approach to either of the two recurrent SPA models in research discussed above (2.1). Their MILP model is uniquely based on a flexible definition of the SPA problem which allocates students to project categories and project supervisors, rather than allocate students solely to projects as seen in previous research [28]. This aims to maximize student satisfaction by fulfilling two criteria rather than one. Calvo-Serrano et al. allocated students to their preferred supervisor and/or to their preferred project category, which provides the usual SPA model with more flexibility leading to better solutions that maximize student satisfaction.

Having, discussed the various SPA models and approaches explored in literature, this paper will focus on the first recurrent model mentioned earlier (2.1) whereby the students have preferences over the projects solely and lecturers have no preferences on the students. The SPA problem will be formulated as a *specific case* of the assignment problem hence the following section discusses relevant methodologies regarding the assignment problem.

2.2 The Assignment Problem

A *special case* of the assignment problem refers to a problem whereby the number of students is not equal to the number of projects i.e a non-square cost matrix; whereas the general assignment problem makes several assumptions :

- A1. The number of students must be equal to the number of projects.
- A2. No two students can be assigned to a project.
- A3. No student can be assigned to more than one project.
- A4. Every student must be allocated to a project.

As such, the general assignment problem would require the number of students to be equal to the number of projects i.e a square cost matrix [37]. The decision to formulate the SPA using BIP as a *special case* of the assignment problem is to enable the adaptation to the current EEE/EIE Department 2019 graduating class data which consists of 109 students and 181 projects. Clearly, in this scenario $109 \neq 181$, hence the cost matrix would be non-square

which is realistic in real-world applications. Additionally, scenarios where the number of students is not equal to the number of projects are practical and more common as the number of projects proposed tends to be greater than the student population to allow for project variety and multiple project preferences.

Therefore, the SPA problem in this paper is a *special case* of the assignment problem enabling the number of students to be distinct from the number of projects i.e a non-square cost matrix. It is important to appreciate that the generalized assignment problem (AP) is a *special case* of the Transportation Problem (TP), which is a *special case* of the minimum cost flow problem, which is inherently a *special case* of a linear program depicted in Figure 2.1. Although any of these problems can be solved using the simplex algorithm, a more efficient algorithm that leverages each specialization's unique structure can be used yielding a better performance. The generalized assignment problem is a *special case* of the transportation problem, as the TP does not require a square cost matrix i.e the number of sources does not need to be equal to the number of destinations; AP requires a square cost matrix i.e the number of assignees to be equal to the number of tasks [37]. As such, we observe our SPA implementation formulated as a *special case* of the AP has a very similar approach to that of the TP, as opposed to the generalized AP.

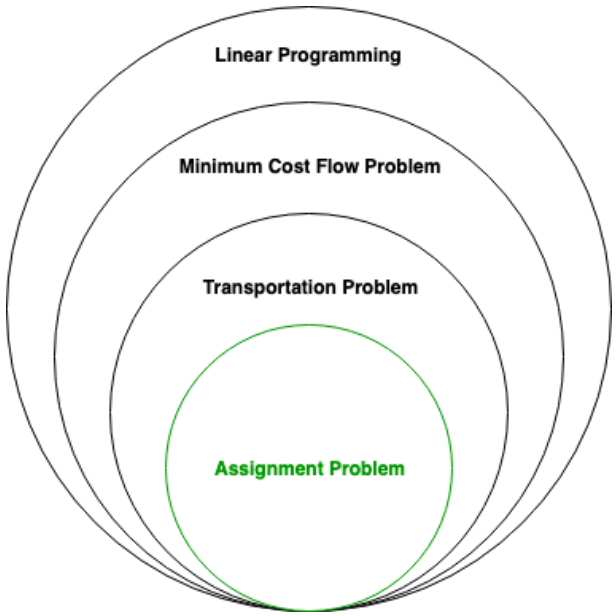


Fig. 2.1 Assignment Problem as a Subset of Linear Programming

It is worth noting that the AP is one of the most studied, well solved and predominant problems in mathematical programming whereby the assignment problem has applied in various industries such as manufacturing, education, etc [38]. The AP solves for scenarios with N resources that are needed to be assigned to N activities considering resource capacity such that overall cost is minimized. The assignment of resources is such that each resource can only associate with one activity i.e a one-to-one basis. Hence, if $C_{i,j}$ denotes the cost of assigning the i^{th} resource to activity j^{th} and $X_{i,j}$ denotes the assignment of the i^{th} resource to activity j^{th} then assignment problem would be formulated as [39]:

The Assignment Problem Formulation:

- *Decision Variables*

$$X_{i,j} = \begin{cases} 1, & \text{if resource } i \text{ is allocated to activity } j \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

- *Objective Function*

$$\text{Minimize } Z = \sum_{i=1}^{N} \sum_{j=1}^{N} (C_{i,j} \times X_{i,j}) \tag{2.2}$$

- *Constraints*

$$\begin{aligned} &\sum_{j=1}^{N} X_{i,j} = 1, \text{ where, } 1 \leq i \leq N \\ &\sum_{i=1}^{N} X_{i,j} = 1, \text{ where, } 1 \leq j \leq N \\ &X_{i,j} \in \{0,1\}, \text{ where, } 1 \leq i,j \leq N \end{aligned} \tag{2.3}$$

Due to the unimodularity of the constraints, the AP can be solved either as a linear program using the simplex algorithm or BIP. Hence, the AP can be implemented and solved using various methods few of which are [40]: Complete Enumeration, Hungarian Method, Simplex LP Method, Munkres Method and Deepest Hole Method. We will explore *Complete Enumeration*, the *Hungarian Method* and the Binary (0-1) Integer Programming method which is formulated as a variation of the assignment problem, and is a subset of a *Simplex LP method*. The Munkres and Deepest Hole Methods are not relevant to this paper, as the Munkres Method is considered a variant of the Hungarian Method which is already being explored hence would be repetitive. Furthermore, the Deepest Hole method cannot guarantee

to always select the lowest overall cost assignment, whereas the BIP method can rendering the Deepest Hole method redundant in comparison [40].

2.2.1 Complete Enumeration

The most direct way of solving an assignment problem such as the SPA problem is by complete enumeration. This entails:

1. **Enumerating through of all N possible assignments of students to projects denoted with matrix $X_{i,j}$.**

   - The number N, of possible assignments of n projects to m students is such that each project can only be associated with one student i.e a one-to-one basis:

     When the possible assignment matrix $X_{i,j}$ is a square matrix that is $n \times n$:

     $$N = n! \text{ operations} \tag{2.4}$$

   - When the matrix $X_{i,j}$ is $m \times n$ where $m < n$:

     $$N = \frac{n!}{(n-m)!} \text{ operations} \tag{2.5}$$

2. **Calculating the objective value of each of those assignments is denoted with $Z_{X_{i,j}}$.** Hence, by enumeration, there would be N possible assignments, $X_{i,j}$ whose objective value $Z_{X_{i,j}}$ (the cost) needs to be calculated for each distinct assignment hence resulting in N objective values.

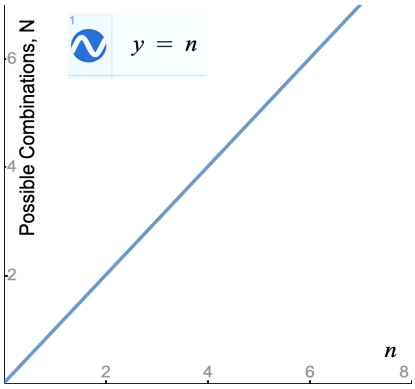
   $$\text{Minimize } Z_{X_{i,j}} = \sum_{i=1}^{M} \sum_{j=1}^{N} (C_{i,j} \times X_{i,j}) \tag{2.6}$$

   The objective function above gives as many students as possible, a high-ranked project since higher-ranked projects have smaller costs than lower-ranked projects. For example, the highest-ranked project 'Rank 1' represents a cost of 1 whereas a lower-ranked project would represent a cost > 1.

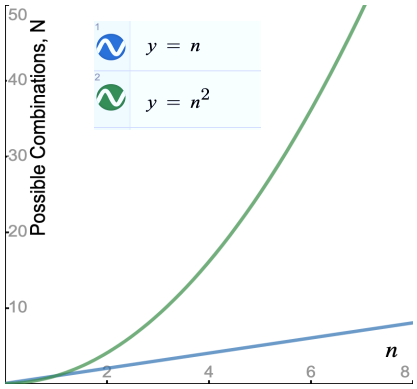
3. **Identifying the optimum assignment which results in the minimum cost.**
   It is necessary for all the N objective values, $Z_{X_{i,j}}$ to be enumerated through to identify the smallest/minimal value and thereby identify the most optimal assignment.

$$\text{Optimal Assignment} = \text{Minimize}[Z_{X_{i,j}}] \quad \forall i, j \tag{2.7}$$

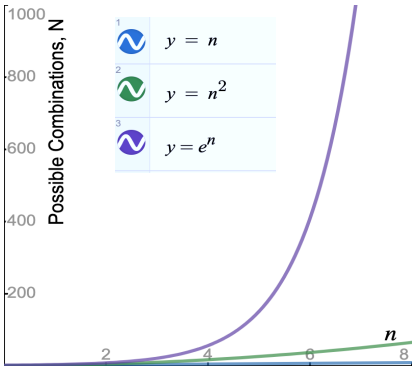
As discussed above, in the square matrix scenario, there are N combinations of assigning n resources to n tasks where $N = n!$ Hence as n increases, N becomes obscenely large. Consider Figure 2.2d below.



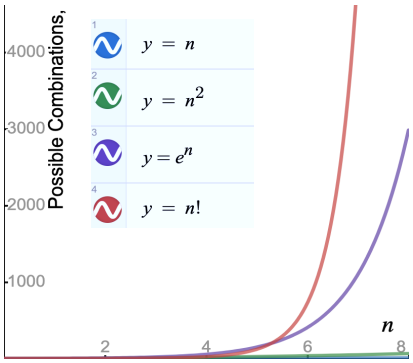
(a) With n possible combinations



(b) With n^2 possible combinations



(c) With e^n possible combinations



(d) With $n!$ possible combinations

Fig. 2.2 Square matrix scenarios: n resources to be assigned to n tasks

As observed in Fig: 2.2, as n increases the factorial function $n!$ increases significantly as opposed to the linear function n, quadratic function n^2 or even the exponential function e^n. Given there are $n!$ or $\frac{n!}{(n-m)!}$ possible assignments depending on the shape of the cost matrix, this informs us that enumerating all possible assignments is only feasible for very small problems. Hence, alternative approaches to solving the SPA problem are required that will be time and computationally efficient. Therefore, due to the slow computational speed of this method, the complete enumeration method cannot possibly be implemented.

2.2.2 Hungarian Method

A common way of solving an assignment problem such as the SPA problem is by using the Hungarian Method. This method was initially developed by Harold Kuhn in 1955 and reviewed later in 1957 after James Munkres discovered that this method is strongly polynomial [41] [42]. To implement the Hungarian Method there are two ways to formulate the problem either as a matrix or as a bipartite graph of which the former approach will be followed.

Matrix Formulation

The matrix formulation requires a non-negative $n \times n$ matrix, where the element in the i^{th} row and j^{th} column represents the cost of assigning the j^{th} project to the i^{th} student. The objective is to find an assignment of the projects to the students such that each project is assigned to one student, and each student is assigned one project whilst minimizing the total cost of the assignment. It is worth noting that this Hungarian method SPA matrix formulation is identical to that of the BIP approach followed in this paper. Given that Munkres proved that the Hungarian Method is strongly polynomial and polynomial-time algorithms are said to be fast, this method can be practically implemented.

2.2.3 Binary Integer Programming

This the cardinal approach that will be used throughout this paper to solve the SPA problem. Requiring the decision variables to be integer values only, generally makes optimization problems harder. This is because techniques that work well for continuous variables can no longer be used. In particular, linear and convex optimization (minimizing a convex function subject to convex constraints) can be done efficiently, but requiring integer values renders the feasible set non-convex. However, it is vital to consider existing literature and research on the SPA problem has shown that the majority of the existing algorithms for the SPA problem that use linear programming will only produce an approximate solution [27] [14]. Whereas, the BIP approach is capable of producing optimal solutions to a range of optimization problems, even those that are NP-hard with optimization solvers such as Gurobi [15], CPLEX [16], and GLPK [17]. These commercial solvers allow for BIP models to be solved relatively fast for large real-world applications. Furthermore, BIP formulations make it simpler to encode distinct decisions and constraints with binary variables as well as enables multi-objective optimization [22][31].

The conclusions observed in existing research indicate that the BIP approach results in an optimal solution rather than an approximate solution to the variants of the SPA problem. It is

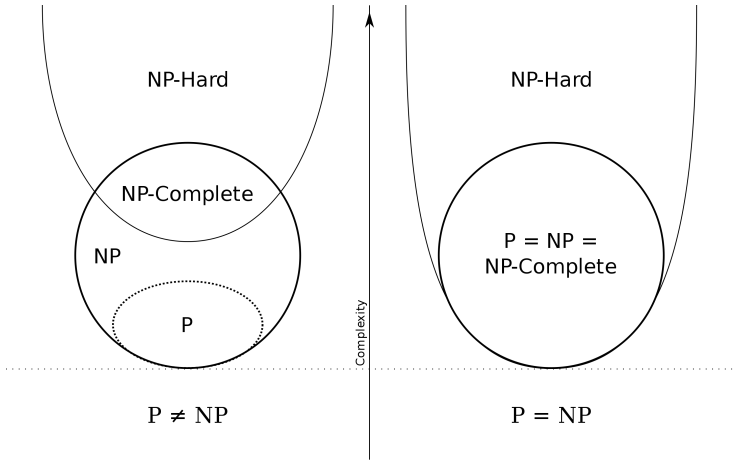


Fig. 2.3 Euler diagram for P, NP, NP-complete, and NP-hard set of problems [1]

essential to note that integer programming is NP-complete and particularly binary integer linear programming is one of Karp's 21 NP-complete problems. Observe in Figure 2.3 that NP-complete problems are in the NP set where all decision problems have solutions that can be verified in polynomial time [43] which is computationally fast. The methodology followed in modelling and implementing the BIP SPA models in this paper follow Figure 2.4 below:

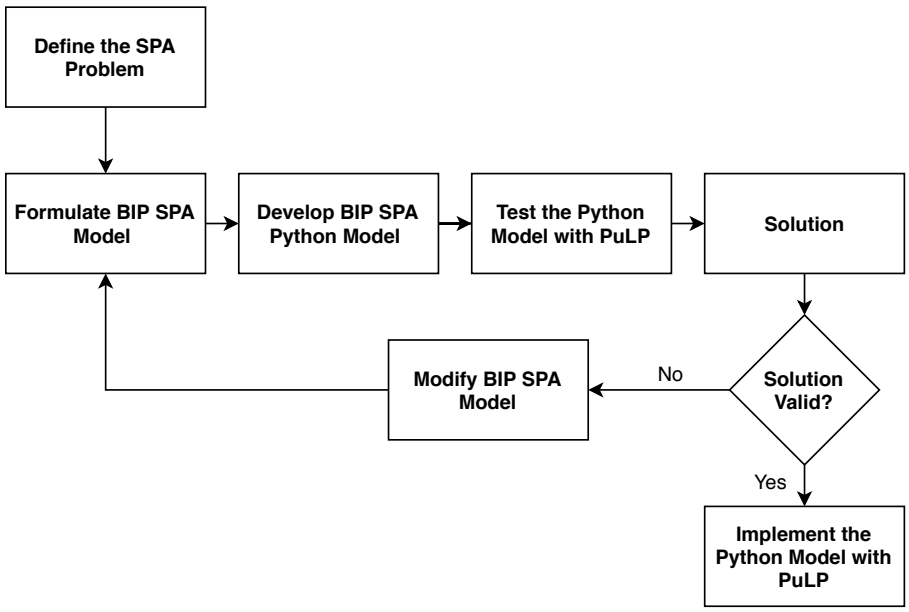


Fig. 2.4 BIP SPA Methodology

Chapter 3

Individual Student Project Allocation Modeling

This chapter comprises of six SPA models using BIP mathematical formulations. The modelling process is defined below and this chapter explores the first two steps whilst the last three steps will be explored in the Chapters 5, 6, and 7.

The modeling process can be broken down into five steps [42]:

1. Forming the problem description
2. Formulating the mathematical program
3. Solving the mathematical program
4. Performing post-optimal analysis
5. Presenting the solution and analysis

As seen above, the modeling process starts with a well-defined model description. The model description is then translated into a mathematical program. The next step is to use solvers such as Gurobi, CPLEX, GLPK, PuLP's CBC to solve the model. The solution is then utilized to make decisions that support the model description.

It is important to note, Model 1 is modeled to ease into the mathematical formulation of the SPA problem using binary (0-1) integer programming providing the base model for the following formulations. Model 2 and Model 3 will be implemented, tested and evaluated as their objective functions target the primary goals of this paper which is to solve the SPA problem maximizing student satisfaction and considering the lecturers' project supervising

workload. Model 4 is modeled as a variation of Model 3 that optimizes the previous model. Given the modification in Model 4 is minor, the implementation would be trivial, furthermore, testing would require realistic data regarding the project supervision workload for each lecturer which is not available, hence, comparative analysis would not be possible. Model 5 and Model 6 show a dynamic approach to Model 3's non-dynamic approach which tackles two objectives by asserting one objective as a constraint whereas the former tackles two objectives by sequentially solving them and considering the previous model's objective function solution (Model 5) as the constraint of the following model (Model 6). The non-dynamic approach (Model 3) and the dynamic approach (Model 5 and Model 6) are inherently identical as they have the same objective function hence implementation would be repetitive. To summarize, Model 2 and 3 will be implemented whilst Model 1, 4, 5, and 6 are used for qualitative analysis to optimize the optimization models and investigate distinct approaches.

The student, project, and lecturer relationship is depicted below in Figure 3.1 where each student can only be allocated to one project thereby forming a one-to-one relationship between students and projects. Observe, each lecturer can propose and supervise multiple projects forming a one-to-many relationship between lecturers and projects (or equivalently a one-to-many relationship between lecturers and students) .

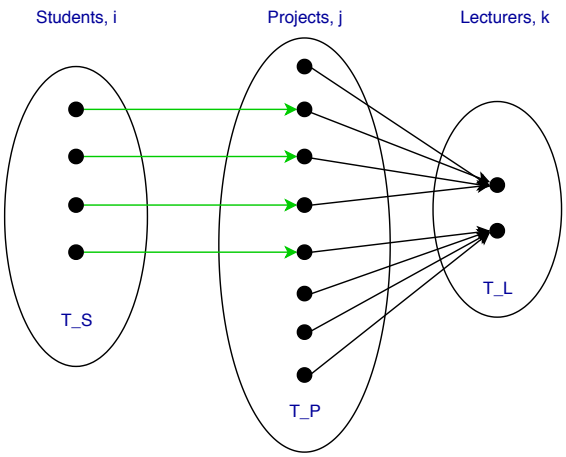


Fig. 3.1 SPA Model

3.1 Model 1: Naively allocate projects to students based solely on their preferences.

Step 1 - Problem Description:
As mentioned in Section 2.1, there two main recurrent SPA models in research. The first simply considers the student's preferences and the second incorporates the preference of the lecturers as well. Model 1 follows from the first model recurrent in research, as it allocates projects to students from a subset of project preferences that the students shortlist from the entire list of project proposals. This model is naive as it is far from optimal but allows for an intuitive problem formulation which will then be improved. Model 1 is designed such there are a total of T_S students. Each student can shortlist a total of T_C projects from a total of T_P proposed projects. In this model, a student can only be allocated to one project and a project can only be allocated to one student - although, in distinct real-world applications such as group project allocations, a project can be allocated to multiple students.
Where,

- T_S, is the total number of students.
- T_L, is the total number of lecturers.
- T_C, is the total number of projects required to be ranked by each student.
- T_P, is the total number of projects proposed by lecturers.
- T_{PS}, is the maximum number of projects each lecturer can supervise.

Step 2 - Mathematical Program Formulation:

- *Decision Variables*
  - $X_{i,j}$: A student can only be allocated a project that has been proposed by a lecturer. This decision can be defined with the binary variable,

$$X_{i,j} = \begin{cases} 1, & \text{if student } i \text{ is allocated to project } j \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

  Note: The index, i represents the i^{th} student, $\{i | 1 \leq i \leq T_S, i \in \mathbb{Z}\}$. The index, j represents the j^{th} project, $\{j | 1 \leq j \leq T_P, j \in \mathbb{Z}\}$.

– $C_{i,j}$: A student can only be allocated a project that has been shortlisted as one of their preferences.

$$C_{i,j} = \begin{cases} 1, & \text{if project } j \text{ is chosen as a preference by student } i \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

- *Constraints*

1. **Each student should only be allocated to one project**
   Where student i has to be allocated to 1 of the total T_P proposed projects,

$$X_{i,1} + \cdots + X_{i,T_P} = 1 \tag{3.3}$$

   This is generalized as:

$$\sum_{j=1}^{T_P} X_{i,j} = 1 \tag{3.4}$$

2. **Each project should be allocated to at-most one student**
   The "$\leq$" in the constraint takes into account, that some projects may be unallocated due to various factors such as not being chosen as a preference by any student.

$$\sum_{i=1}^{T_S} X_{i,j} \leq 1 \tag{3.5}$$

3. **Each student can select a maximum of T_C projects as part of their preferences subset**
   Where, student i has to choose T_C project preferences,

$$C_{i,1} + \cdots + C_{i,T_P} = T_C \tag{3.6}$$

   Notice, if project j is not chosen by student i, then $C_{i,j} = 0$ and project j would not be applicable for allocation to student i. This is generalized as:

$$\sum_{j=1}^{T_P} C_{i,j} = T_C \tag{3.7}$$

4. **Each student can only be allocated to a project that is part of their preferences subset**
   Out of the T_c possible preferences, student i can only be allocated to 1 of their

project preferences.

$$C_{i,1} \times X_{i,1} + \cdots + C_{i,T_P} \times X_{i,T_P} = 1 \tag{3.8}$$

This is generalized as:

$$\sum_{j=1}^{T_P} C_{i,j} \times X_{i,j} = 1 \tag{3.9}$$

- *Objective Function*

$$Z_1 = \sum_{i=1}^{T_S} \sum_{j=1}^{T_P} (C_{i,j} \times X_{i,j}) \tag{3.10}$$

3.2 Model 2: Allocate projects to students based on their preference rankings.

Step 1 - Problem Description:

This model is an improvement on Model 1, as students are required to strictly rank their shortlisted project preferences rather than simply selecting a subset of project preferences as done in Model 1. Although rankings can be defined by percentage weights, Model 2 will explore integer weights. The weights, $C_{i,j}$ of student i on project j will range such that $C_{i,j} \in [1, T_C]$, where T_C represents the total number of project preferences allowed to be ranked by each student. Consider a case where $T_C = 5$, students should assign a weight of 5 to their least preferable project (their fifth preference) and a weight of 1 to their most preferable project (their first preference) and 0s to projects they are not interested in. To favor lower rankings i.e most preferred projects in this optimization problem, Model 2 will be a **minimization** problem. If the weights were reversed such that a weight of 5 refers to their first preference, and a weight of 1 refers to their fifth preference then we would formulate the problem as a maximization problem. Recall, the variable definitions outlined in Model 1 (3.1).

Step 2 - Mathematical Program Formulation:

- *Decision variables*
  - $X_{i,j}$: Remains unchanged from 3.1

$$X_{i,j} = \begin{cases} 1, & \text{if student } i \text{ is allocated to project } j \\ 0, & \text{otherwise} \end{cases} \tag{3.11}$$

– $C_{i,j}$: Is no longer a binary variable as in Model 1 and is an integer variable in Model 2. $C_{i,j}$ indicates the ranking assigned to project j by student i. If student i did not list project j as a possible preference then $C_{i,j} = 0$. If student i lists project j as preference 1 then $C_{i,j} = 1$, if student i lists project j as preference 2 then $C_{i,j} = 2$ and so on. Hence, $C_{i,j}$ can take any value where, $C_{i,j} \in [1, T_C]$ where T_C is the total number of projects required to be ranked by each student. Hence, if $T_C = 5$:

$$C_{i,j} = \begin{cases} 1, & \text{if student } i \text{ ranks project } j \text{ as } 1^{st} \text{ preference} \\ 2, & \text{if student } i \text{ ranks project } j \text{ as } 2^{nd} \text{ preference} \\ 3, & \text{if student } i \text{ ranks project } j \text{ as } 3^{rd} \text{ preference} \\ 4, & \text{if student } i \text{ ranks project } j \text{ as } 4^{th} \text{ preference} \\ 5, & \text{if student } i \text{ ranks project } j \text{ as } 5^{th} \text{ preference} \end{cases} \tag{3.12}$$

- *Constraints*

1. **Each student should only be allocated to one project**
   Remains unchanged from 3.4

$$\sum_{j=1}^{T_P} X_{i,j} = 1, \text{ where, } 1 \leq i \leq T_S \tag{3.13}$$

2. **Each project should be allocated to at-most one student**
   Remains unchanged from 3.5

$$\sum_{i=1}^{T_S} X_{i,j} \leq 1, \text{ where, } 1 \leq j \leq T_P \tag{3.14}$$

3. **Each student can only be allocated a project that is part of their preferences**

$$X_{i,j} \leq C_{i,j} \tag{3.15}$$

   This constraint doesn't allow for $X_{i,j} = 1$ if $C_{i,j} = 0$, demonstrating that a student cannot be allocated a project that hasn't been listed as a preference i.e $C_{i,j} = 0$. Hence, $C_{i,j} = 0$ asserts $X_{i,j} = 0$ indicating student i is not assigned to project j.

- *Objective Function*
  The objective is to give as many students as possible their highest-ranked project.

$$\text{Minimize } Z_2 = \sum_{i=1}^{T_S} \sum_{j=1}^{T_P} (C_{i,j} \times X_{i,j}) \tag{3.16}$$

3.3 Model 3: Allocate projects to students based on their preference rankings whilst putting an upper-bound (T_{PS}), on the number of projects each lecturer can supervise.

Step 1 - Problem Description:
It has been noted from Dr. Clarke - Head of Student Project Allocation within the EEE Department at Imperial College London, that lecturers are happier if they're allocated a maximum of 3 projects or fewer in-order not to over-extend themselves. It was also noted that lecturers who supervised a maximum of 3 or fewer projects yielded students that perform at much higher standards delivering academically stronger projects than students whose projects were being supervised by lecturers who were overseeing 4 or more projects. This observation is attributed to the time a lecturer can afford a student if they are over-burdened with a large project supervision workload as well as day-to-day teaching and faculty responsibilities.

Model 3 considers the strict project rankings the students have assigned to their project preferences as well as the supervision workload implications on the lecturers that supervise the projects. The objective of this model is to limit the lecturer supervision workload up to T_{PS} projects, whilst where possible, allocate students their highest-ranked projects.

This model has a similar objective as the dynamic approach of Model 5 and Model 6, however, it is implemented by following a non-dynamic approach. This model builds on Model 2 that has students rank their shortlisted project preferences, but also limits the number of projects each lecturer will supervise by including a new constraint. This approach will aim for the number of projects each lecturer supervises to be bounded by a maximum number of project supervisions allowed simultaneously, T_{PS}. Recall, the variable definitions outlined in Model 1 (3.1).

Step 2 - Non-dynamic Mathematical Program Formulation:

- *Decision variables*
  - $X_{i,j}$: Remains unchanged from 3.1

$$X_{i,j} = \begin{cases} 1, & \text{if student } i \text{ is allocated to project } j \\ 0, & \text{otherwise} \end{cases} \tag{3.17}$$

  - $C_{i,j}$: Remains an integer variable unchanged from 3.12

$$C_{i,j} = x, \text{ if student } i \text{ ranks project } j \text{ as } x^{th} \text{ preference} \tag{3.18}$$

  - $P_{k,j}$: Binary variable that determines whether lecturer k proposed project j

$$P_{k,j} = \begin{cases} 1, & \text{if lecturer } k \text{ proposed project } j \\ 0, & \text{otherwise} \end{cases} \tag{3.19}$$

  - $S_{k,i}$: Binary variable that determines whether a lecturer will supervise a student. Due to the one-to-one relationship of a student and project this variable also indicates whether a lecturer will supervise a project.

$$S_{k,i} = \begin{cases} 1, & \text{if lecturer } k \text{ will supervise student } i \\ 0, & \text{otherwise} \end{cases} \tag{3.20}$$

    Where,

$$S_{k,i} = P_{k,j} * X_{i,j}^{\mathsf{T}} \tag{3.21}$$

- *Constraints*

  All constraints in Model 2 remain unchanged and one constraint is added - the lecturer supervision constraint.

  1. **Each student should only be allocated to one project**
     Remains unchanged from 3.4

$$\sum_{j=1}^{T_P} X_{i,j} = 1, \text{ where, } 1 \leq i \leq T_S \tag{3.22}$$

  2. **Each project should be allocated to at-most one student**
     Remains unchanged from 3.5

$$\sum_{i=1}^{T_S} X_{i,j} \leq 1, \text{ where, } 1 \leq j \leq T_P \tag{3.23}$$

3. **Each student can only be allocated a project that is part of their preferences**
   Remains unchanged from 3.15

$$X_{i,j} \leq C_{i,j} \tag{3.24}$$

4. **Added constraint: Each lecturer can only supervise a given number of projects, T_{PS}**
   Model 2 constraints ensure that the majority of the student would get allocated their first choice project. Adding this constraint requires that each lecture does not supervise more than T_{PS} number of projects, where T_{PS} will be the maximum number of projects each lecturer can supervise.
   This can be achieved by:

$$\sum_{j=1}^{T_P} P_{k,j} \times \sum_{i=1}^{T_S} X_{i,j}^{\intercal} \leq T_{PS}, \text{ where, } 1 \leq k \leq T_L \tag{3.25}$$

   Or equivalently:

$$\sum_{i=1}^{T_S} S_{k,i} \leq T_{PS}, \text{ where, } 1 \leq k \leq T_L \tag{3.26}$$

   Note: Summing $S_{k,i}$ across students yields S_k which indicates the number of students/projects each lecturer k is supervising. Hence,

$$\begin{aligned} &\sum_{i=1}^{T_S} S_{k,i} \equiv S_k \\ &S_k \leq T_{PS}, \text{ where, } 1 \leq k \leq T_L \end{aligned} \tag{3.27}$$

- *Objective Function*
  The objective is to give as many students the highest-ranked project as possible whilst limiting the number of projects each lecturer supervises by T_{PS} thereby limiting the lecturer supervision workload. This is incorporated by the lecturer constraint 3.27 seen above. This model enables lecturer supervision efficiency, increasing the prospects of

a majority of students performing academically well on their allocated projects.

$$\text{Minimize } Z_3 = \sum_{i=1}^{T_S} \sum_{j=1}^{T_P} (C_{i,j} \times X_{i,j}) \tag{3.28}$$

3.4 Model 4: Allocate projects to students based on their preference rankings whilst putting an upper-bound (β_k), on the number of projects lecturer k is allowed to supervise.

Step 1 - Problem Description:
Practically, it would be unrealistic to limit every lecturer to a constant T_{PS} as all lecturers do not have uniform teaching and faculty responsibilities. Existing factors such as varied university teaching workloads, tutorial sessions, post-doctorate dissertation supervisions, and research obligations differ lecturer to lecturer. As such, some lecturers may have the leeway to supervise more students than other lecturers without compromising on student supervision quality and time dedicated. This vital circumstance can be considered with a minor modification to Model 3's fourth constraint i.e the lecturer supervision constraint as all other aspects of the optimization formulation of Model 3 remain unchanged. To accommodate this aspect, instead of having a constant upper-bound (T_{PS}) for all lectures, a variable upper-bound (β_k) specific to the lecturer k will be applied. This ensures each lecturer has a distinct maximum number of project supervisions acceptable dependent on their availability. The values of β_k would depend on lecturer k and their current workload to ascertain the respective project supervision workload they can manage successfully. To achieve this, Model 3's fourth constraint is modified by replacing T_{PS} with β_k, also recall the variable definitions in outlined in Model 1 (3.1).

Step 2 - Mathematical Program Formulation:

- *Decision variables*
  - $X_{i,j}$: Remains unchanged from Model 1 (3.1)
  - $C_{i,j}$: Remains unchanged from Model 2 (3.12)
  - $P_{k,j}$: Remains unchanged from Model 3 (3.19)

- $S_{k,i}$: Remains unchanged from Model 3 (3.20)

- β_k : Maximum number of projects lecturer k is allowed to supervise at a time i.e the lecturer supervision workload for lecturer k

- *Constraints*

  The first three constraints in Model 3 remain unchanged whilst the fourth constraint, the lecturer supervision workload constraint is modified.

  1. **Each student should only be allocated to one project**
     Remains unchanged from 3.4

$$\sum_{j=1}^{T_P} X_{i,j} = 1, \text{ where, } 1 \leq i \leq T_S \tag{3.29}$$

  2. **Each project should be allocated to at-most one student**
     Remains unchanged from 3.5

$$\sum_{i=1}^{T_S} X_{i,j} \leq 1, \text{ where, } 1 \leq j \leq T_P \tag{3.30}$$

  3. **Each student can only be allocated a project that is part of their preferences**
     Remains unchanged from 3.15

$$X_{i,j} \leq C_{i,j} \tag{3.31}$$

  4. **Modified constraint: Each lecturer can only supervise a given number of projects, β_k**
     This constraint is similar to that of Model 3, except the right-hand-side is changed from T_{PS} to β_k. This constraint requires that each lecture does not supervise more than β_k number of projects, where β_k will be the maximum number of projects lecturer k is allowed to supervise.

     This can be achieved by:

$$\sum_{j=1}^{T_P} P_{k,j} \times \sum_{i=1}^{T_S} X_{i,j}^{\intercal} \leq \beta_k, \text{ where, } 1 \leq k \leq T_L \tag{3.32}$$

Or equivalently:

$$\sum_{i=1}^{T_S} S_{k,i} \leq \beta_k, \text{ where, } 1 \leq k \leq T_L \tag{3.33}$$

Note:

$$\begin{aligned} &\sum_{i=1}^{T_S} S_{k,i} \equiv S_k \\ &S_k \leq \beta_k, \text{ where, } 1 \leq k \leq T_L \end{aligned} \tag{3.34}$$

- *Objective Function*
  The objective is to give as many students the highest-ranked project as possible whilst limiting the number of projects lecturer k supervises by β_k. This is incorporated by the lecturer constraint 3.27 seen above.

$$\text{Minimize } Z_4 = \sum_{i=1}^{T_S} \sum_{j=1}^{T_P} (C_{i,j} \times X_{i,j}) \tag{3.35}$$

3.5 Model 5: Allocate projects based on minimizing the number of projects each lecturer supervises.

Step 1 - Problem Description:
This model naively allocates projects to students from a subset of project preferences whilst minimizing the number of projects supervised by each staff member. The student's project preferences are not weighted i.e all their preferences have an equal possibility of being assigned as the primary goal is minimizing the lecturer supervision workload of the lecturers. Recall, the variable definitions in outlined in Model 1 (3.1).

Step 2 - Mathematical Program Formulation:

- *Decision variables*
  - $X_{i,j}$: Remains unchanged from Model 1 (3.1)
  - $C_{i,j}$: Remains unchanged from Model 2 (3.12)
  - $P_{k,j}$: Remains unchanged from Model 3 (3.19)

- $S_{k,i}$: Remains unchanged from Model 3 (3.20)

- *Constraints*

1. **Each student should only be allocated to one project**
   Remains unchanged from 3.4

$$\sum_{j=1}^{T_P} X_{i,j} = 1, \text{ where, } 1 \leq i \leq T_S \tag{3.36}$$

2. **Each project should be allocated to at-most one student**
   Remains unchanged from 3.5:

$$\sum_{i=1}^{T_S} X_{i,j} \leq 1, \text{ where, } 1 \leq j \leq T_P \tag{3.37}$$

3. **Each student can only be allocated to a project that is part of their preferences subset**
   Remains unchanged from 3.9

$$\sum_{j=1}^{T_P} C_{i,j} \times X_{i,j} = 1 \tag{3.38}$$

4. **Each lecturer can only supervise a given number of projects, T_{PS}**

$$\sum_{j=1}^{T_P} S_{k,j} \leq T_{PS}, \text{ where, } 1 \leq k \leq T_L \tag{3.39}$$

- *Objective Function*

$$\text{Minimize } Z_5 = T_{PS} \tag{3.40}$$

This minimizes the variable T_{PS}, hence minimizes the total number of projects supervised by each lecturer.

3.6 Model 6: Allocate projects to students based on their preference rankings whilst minimizing the number of projects each lecturer supervises.

Step 1 - Problem Description:
Model 6 will explore the same objective function as Model 6 however explored using a dynamic approach. This model builds on Model 5 that minimizes the number of projects supervised by each lecturer by allocating student a project from their list of project preferences. Instead of simply allocating from a list of project preferences, students will be required to rank their project preferences similar to Model 2. This dynamic approach combines Model 5 and Model 2 such that the first model (Model 5), minimizes the number of projects supervised by each lecturer and the second model (Model 2), distinguishes between equivalent solutions from Model 5. This results in a solution where the maximum number of students are allocated their first choice whilst minimizing the number of projects supervised by each lecturer [22]. Model 5 and Model 2 are combined sequentially, such that the objective function result from Model 5 is utilized as a constraint in Model 2, which defines Model 6. Recall, the variable definitions in outlined in Model 1 (3.1).

Step 2 - Dynamic Mathematical Program Formulation:

- *Decision variables*
  - $X_{i,j}$: Remains unchanged from Model 1 (3.1)
  - $C_{i,j}$: Remains unchanged from Model 2 (3.12)
  - $P_{k,j}$: Remains unchanged from Model 3 (3.19)
  - $S_{k,i}$: Remains unchanged from Model 3 (3.20)

- *Constraints*
  The first two constraints(3.36, 3.37) in Model 5 remain unchanged, the third constraint (3.38) is modified whilst the final constraint is omitted (3.39).

  1. **Each student should only be allocated to one project**
     Remains unchanged from 3.36

$$\sum_{j=1}^{T_P} X_{i,j} = 1, \text{ where, } 1 \leq i \leq T_S \tag{3.41}$$

2. **Each project should be allocated to at-most one student**
   Remains unchanged from 3.37:

$$\sum_{i=1}^{T_S} X_{i,j} \leq 1, \text{ where, } 1 \leq j \leq T_P \tag{3.42}$$

3. **Modified constraint: Each student can only be allocated to a project that is part of their preferences subset**
   The third constraint in Model 5 (3.38) is modified such that the constrained is no longer $= 1$ but ≥ 1 as seen below. This is because, $C_{i,j}$ is no longer a binary variable as in Model 5 (3.2) but an integer indicated ranking as in Model 2 (3.12)

$$\sum_{j=1}^{T_P} C_{i,j} \times X_{i,j} \geq 1, \text{ where, } 1 \leq i \leq T_S \tag{3.43}$$

4. **Added constraint: Allocate whilst maintaining optimum uniformity of lecturer supervision from Model 5**
   This constraint is added to ensure the lecturer supervision workload for Model 6 is the same as that found in Model 5. For reference, $Z_5 = T_{PS}$ as seen in Model 5 (3.40), where T_{PS} is the lecturer supervision workload derived in Model 5.

$$\sum_{j=1}^{T_P} S_{k,j} \leq Z_5, \text{ where, } 1 \leq k \leq T_L \tag{3.44}$$

- *Objective Function*
  The objective is to give as many students as possible as high a ranked project as possible identical to the objective function of Model 2 seen in 3.16. However, this is implemented by distinguishing between the feasible solutions in Model 5 which minimizes the number of projects supervised by each lecturer, to select the solution where the highest number of students are allocated their first choice.

$$\text{Minimize } Z_6 = \sum_{i=1}^{T_S} \sum_{j=1}^{T_P} (C_{i,j} \times X_{i,j}) \tag{3.45}$$

Chapter 4

Implementation of SPA Models

This chapter comprises of the implementation process of the existing methods, and the BIP approach distinctly formulated in the previous chapter. The following methods discussed in section 2.2 will be investigated to deem whether whether they are adequately effective to be implemented. The methods are *Complete Enumeration*, the *Hungarian Method*, and the *BIP* approach.

4.1 Existing Methods

Firstly, the currently existing methods: *Complete Enumeration* and *Hungarian Method* will be implemented to understand the complexity, time and number of operations required to assign a student to a project. Complete Enumeration does not require the number of students to be equal to the number of projects as desired using the Hungarian Method. Therefore, complete enumeration will use the realistic '2018/2019 Student Project Data' where the number of students m, is not equal to the number of projects n. However, since the Hungarian method strictly requires $m = n$, the EEE/EIE '2018/2019 Student Project Data' must be modified such that its $m \times n$ dimension where $m \neq n$ is changed to a $m \times m$ (square) dimension. This adaptation is done by extending the n dimension by '$m-n$' elements with a much higher value than the current maximum value.

4.1.1 Complete Enumeration Implementation

To gauge the time and number of operations needed to assign every student to a project, the complete enumeration is simulated using the current EEE/EIE Department 2019 graduating class data. Since the data has 109 students and 181 projects, the assignment cost matrix $C_{i,j}$ is of dimension 109×181 as $m = 109$, and $n = 181$. As such, using complete enumeration

would result in N possible assignments:

$$\begin{aligned} N &= \frac{n!}{(n-m)!} \\ N &= \frac{181!}{(181-109)!} \\ N &= 5.93 \times 10^{227} \text{ operations} \end{aligned} \tag{4.1}$$

As shown, there are 5.93×10^{227} possible assignments ($X_{i,j}$). Given that each of these possible assignments need to be formed, and its respective cost/objective value ($Z_{X_{i,j}}$) calculated, there would be 5.93×10^{227} ($Z_{X_{i,j}}$) objective values. This is an obscenely large number of variables! In order to identify the optimal assignment that results in the minimum objective value, each of the 5.93×10^{227} objective values would need to be analyzed to determine the minimal objective value.

Time Impact

Complete enumeration requires a program that iterates through 5.93×10^{227} variables to find the minimum objective value i.e Min$[Z_{X_{i,j}}]$ $\forall i, j$. Hence, the time taken to enumerate through 5.93×10^{227} values to determine the smallest objective value can be estimated using the following program:

```
#Create empty array
values = []
#N : Number of variables iterated eg. 5000 variables
N=5000
#Fill array with N randomly generated numbers
for _ in range(N):
   values.append(randint(0, 10))
#Start timer seconds
start = time.time()
#Find minimum value out of N variables
find_min_value = np.array(values).min()
#Stop timer
end = time.time()
#Print time elapsed finding the minimum of N values
print("Time elapsed:", end - start)
```

The program above mimics the computation required to find the minimum value out of N possible values. In the above code, the for-loop in line 6 creates a randomly generated array with N values between 0 and 10 that represent the N objective value variables $Z_{X_{i,j}}$. Line 10 starts the program timer as line 11 determines the minimum value of the N values using the '.min()' function. The time taken to compute the minimum value of the N values is found for different values of N declared in line 4 and extrapolated using the *'TREND'* function in MS 2019 Excel Software to find the time taken when $N = 5.93 \times 10^{227}$ as shown in Table 4.1. The *'TREND'* function is used to extrapolate as setting $N = 5.93 \times 10^{227}$ in line 4 stalls the program and Macbook Pro machine even with 8 threads running simultaneously.

N, **Number of Possible Assignments**	**Time, seconds**
1	4.03×10^{-5}
10	6.91×10^{-5}
100	7.22×10^{-5}
1000	1.36×10^{-4}
10000	7.09×10^{-4}
100000	6.25×10^{-3}
1000000	5.70×10^{-2}
10000000	5.96×10^{-1}
100000000	6.15
1000000000	98.05
Extrapolated point: 5.93×10^{227}	$5.82 \times 10^{220} \approx 6.74$ days

Table 4.1 Time taken to find minimum value as N increases

As observed above, searching through 5.93×10^{227} objective values, would take 6.74 days to determine the minimum value and optimal assignment! Complete enumeration is an extremely slow method that requires extensive CPU resources rendering this method as one that could not possibly be implemented as speculated earlier in section 2.2.1 .

4.1.2 Hungarian Method Implementation

Given that this method strictly requires $m = n$, the EEE/EIE '2018/2019 Student Project Data' is modified such that its dimension $m \times n$ where $m \neq n$ becomes a square dimension of $m \times m$. As there 109 students and 181 projects, $m = 181$.

Given that the Munkres established that Hungarian method runs in polynomial time, Kuhn derived the equation that calculated the maximum number of operations needed to solve completely any $m \times m$ assignment problem, N_{max} as [40]:

$$\begin{aligned} N_{max} &= \frac{m}{6}\left(11m^2 + 12m + 31\right) \\ &= \frac{181}{6}\left(11(181)^2 + 12(181) + 31\right) \\ &= 1.09 \times 10^7 \text{ operations} \end{aligned} \tag{4.2}$$

Practically N_{max} is important as it is much smaller (for $m \geq 6$) than the $m!$ operations necessary to solve the assignment problem using the *Complete Enumeration* method [40] discussed earlier. Completing 1.09×10^7 operations is fairly fast as the Hungarian method runs at polynomial time. Hence, this method can be implemented as speculated earlier in section 2.2.2.

4.2 BIP Method

Three methods of implementing a BIP optimization model using Python were considered.

- PuLP - powerful open-source modeling package in Python
- Gurobi - powerful commercial optimization solver with Python interface (gurobipy)
- CPLEX - powerful commercial optimzation solver with Python interface (docplex)

After weighing the ease of adaptation of the various methods the PuLP optimzation library was the simplest approach. PuLP enables the code to call Gurobi, CPLEX and other solvers to solve the MILP, making it more flexible than utilizing gurobipy or docplex which are limited to one solver, Gurobi and CPLEX respectively. From here-on-out, the Gurobi solver was selected as the primary solver for this project rather than PulP's default CBC solver. The case studies shown in the PuLP optimization library documentation [44], and Knight's implementation [45], guided on how to program the following integer programs in Python.

4.2.1 Model 2 Implementation

To implement an optimization with PuLP, the following components must be defined. The below code are excerpts from Model 2 seen in Appendix A.1.

1. Reading input matrix $C_{i,j}$ denoted as C

```
#Retrieving Student Preferences data from excel
C = fnc.read_preferences('Workbook_name.xls','Worksheet_name')
```

2. Define the linear problem using *'LpProblem'*

```
# Create the 'prob' variable to contain the problem data
prob = LpProblem("SPA Model_2", LpMinimize)
```

3. Define the decision variables using *'LpVariable'*

$$X_{i,j} = \begin{cases} 1, & \text{if student } i \text{ is allocated to project } j \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

```
# a) Variable x
# A dictionary called 'x' is created to contain the referenced
    Variables
# Var x_{i,j} has a Lower-bound of # of students, Upper-bound of #
    of projects and is of type 'LpBinary'
x = LpVariable.dicts("x",
    itertools.product(range(number_of_students),
    range(number_of_projects)),cat=LpBinary)
```

4. Define the objective function and include it in the optimization problem using as seen in line 7 in the code excerpt below

$$\text{Minimize } Z_2 = \sum_{i=1}^{T_S} \sum_{j=1}^{T_P} (C_{i,j} \times X_{i,j}) \tag{4.4}$$

```
#Objective Function
objective_function = 0
for student in range(number_of_students):
    for project in range(number_of_projects):
        if C[(student, project)] > 0:
            objective_function += x[(student, project)] *
                ((C[(student, project)]))
prob += objective_function
```

5. Define the constraints and include it in the optimization problem

   (a) **Each student can only be allocated a project that is part of their preferences**

$$X_{i,j} \leq C_{i,j} \tag{4.5}$$

```
#a) Each student can only be allocated a project that is part
    of their preferences subset
for student, project in x:
    prob += x[student, project] <= float(C[student, project])
        #, "Preference_Constraint"
```

   (b) **Each student should only be allocated one project**

$$\sum_{j=1}^{T_P} X_{i,j} = 1, \text{ where, } 1 \leq i \leq T_S \tag{4.6}$$

```
#b) Each student should be allocated to only 1 project
for student in range(number_of_students):
    prob += sum(x[(student, project)] for project in
        range(number_of_projects)) == 1 #, "Student_Constraint"
```

   (c) **Each project should be allocated to at-most one student**

$$\sum_{i=1}^{T_S} X_{i,j} \leq 1, \text{ where, } 1 \leq j \leq T_P \tag{4.7}$$

```
#c) Each project should be allocated to at most 1 student
for project in range(number_of_projects):
    prob += sum(x[(student, project)] for student in
        range(number_of_students)) <= 1 #, "Project_Constraint"
```

6. Solve the optimization problem with the solver of your choice

```
# The problem is solved using PuLP's open-source CBC solver
prob.solve()
# The problem is solved using commercial Gurobi solver
prob.solve(GUROBI())
# If the problem is solved using commercial CPLEX solver
prob.solve(CPLEX())
```

7. Print the solution status either: Optimal, Infeasible, Unbounded, Undefined or Not-Solved

```
# The status of the solution is printed to the screen
print("Status:", LpStatus[prob.status])
```

8. Print the optimized variables (Student Project Allocations)

```
# Each of the variables is printed with it is resolved optimum value
for v in prob.variables():
    print(v.name, "=", v.varValue)
```

9. Print the optimized objective function value

```
# The optimized objective function value is printed to the screen
print("Objective Fnc= ", value(prob.objective))
```

A challenge with using PuLP optimization library was seen in printing the optimized variables in step 8 above. Given that PuLP allocates the values of the optimal solution to coordinates of the form $(x,\_y)$, which are of type 'String' rather than 'Int', meant that the coordinates would follow lexicographical numbering rather than conventional numerical ordering.

Hence, the output of 11 variables will be lexicographically ordered as $[0,1,10,2,3,4,5,6,7,8,9]$ rather than $[0,1,2,3,4,5,6,7,8,9,10]$ as shown in Table 4.2 as string item 11 of the form $(0,\_10)$ would be seen as a substring of $(0,\_1)$ when it should not. This lead to issues in the interpretation of the allocation results.

Printing the optimized variables lexicographically, resulted in an incorrect interpretation of the allocation as shown in Table 4.3 'String Error Allocation Mapping'. Hence, the code modification seen in Appendix A.2, was implemented instead of step 8 above. This modification shown below ensured the optimized variables were printed numerically, resulting in the correct mapping of indices in the allocation matrix which produced the 'Ideal Allocation Mapping' in Table 4.3

String Error	Ideal Output
x_(0,_0) = 1.0 x_(0,_1) = 0.0 x_(0,_10) = 0.0 x_(0,_2) = 0.0 x_(0,_3) = 0.0 x_(0,_4) = 0.0 x_(0,_5) = 0.0 x_(0,_6) = 0.0 x_(0,_7) = 0.0 x_(0,_8) = 0.0 x_(0,_9) = 0.0	x_(0,_0) = 1.0 x_(0,_1) = 0.0 x_(0,_2) = 0.0 x_(0,_3) = 0.0 x_(0,_4) = 0.0 x_(0,_5) = 0.0 x_(0,_6) = 0.0 x_(0,_7) = 0.0 x_(0,_8) = 0.0 x_(0,_9) = 0.0 x_(0,_10) = 0.0

Table 4.2 String Error Analysis

String Error Allocation Mapping	Ideal Allocation Mapping
Status: Optimal [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.] [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]] Objective Fnc= 3.0	Status: Optimal [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.] [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]] Objective Fnc= 3.0

Table 4.3 String Error Allocation Mapping

8. Modified: Print the optimized variables (Student Project Allocations)

```
# Each of the variables is printed with it is resolved optimum value
result = np.zeros((number_of_students,number_of_projects))
for v in prob.variables():
    #filtered is (0,_0) to 0,_0
    filtered = re.search('x_\((.+?)\)', v.name).group(1)
    #find the comma's index which is 1
    comma_idx = filtered.find(',_')
    #find the underscore's index (not used)
    underscore_idx = comma_idx + 1
    #set coords as integers from filtered to get int (0,0) from string 0,_0
    coords = int(filtered[:comma_idx]), int(filtered[comma_idx+2:])
    #create array result with coords and set their respective values
    result[coords] = v.varValue
print(result)
```

4.2.2 Model 3 Python

The implementation of Model 3 seen in Appendix A.3, modifies step 1 and step 5 from the previous model (Model 2). The changes to these steps are shown below:

1. Reading input matrix $P_{k,j}$ denoted as L

```
# Retrieving Lecturer Project Proposals data from excel
L = fnc.read_preferences('Workbook_name.xls','Worksheet_name')
```

5. The constraints in Model 2 remain unchanged and the lecturer project supervision constraint was added as c) shown below.
   (c) Each lecturer can only supervise a given number of projects, T_{PS}

$$\sum_{j=1}^{T_P} P_{k,j} \times \sum_{i=1}^{T_S} X_{i,j}^{\intercal} \leq T_{PS}, \text{ where, } 1 \leq k \leq T_L \tag{4.8}$$

```
#c) Each lecturer can only supervise T_PS projects/students
for lecturer in range(number_of_lecturers):
    prob += sum(L[lecturer, project] * sum(x[(student, project)]
    for student in range(number_of_students))
    for project in range(number_of_projects)) <= T_PS
```

Chapter 5

Model Testing

This chapter examines the SPA optimization model implementations by applying simple scenarios to establish the correctness of the models. The Hungarian method is tested to further analyze efficiency of this well-known method compared to the BIP SPA models. This chapter continues to explore step 3 of the modelling process (3) particularly using testing data.

3. Solving the mathematical program (using test data)

Solving the mathematical programs using test data ensures the optimization models function as expected prior to applying the '2018/2019 Student Project Data' to the SPA models. The verification process is as follows,

- Each model will be defined with a simple conceptual scenario and evaluated manually to obtain the expected result.
- Each model will be implemented using Python's optimization library PuLP to obtain a practical result that should be identical to the expected result obtained prior.

5.1 Hungarian Method

Prior to testing the primary BIP SPA implementation, we test the most commonly used method to solve assignment problems - the Hungarian Method.

Conceptually:
The Hungarian Model takes a square input cost matrix $C_{i,j}$ that is non-negative and of the dimension $n \times n$. The cost matrix $C_{i,j}$ denotes the project preferences for student i regarding project j. Note that the indexing of i, and j are initialized from 1. Hence, when $n = 3$, index

of $i = 2$ and $j = 3$, implies student 2 ranks project proposal 3 as their second most preferred project highlighted in red below.

Where a student would normally leave a preference 'null' i.e $C_{i,j} = 0$ when they're not interested in ranking a certain project, in the Hungarian algorithm that changes such that the student who has zeroes to represent a lack of interest in projects will have their $C_{i,j} = 0$ values changed to $C_{i,j} = max(row\ cost) + 1$. This ensures that the student can only be assigned one of their ranked project preferences as projects that the student doesn't rank have the highest cost thereby eliminating any likelihood of them being allocated. As such, the usual unmodified $C_{i,j}$ matrix below is modified to $C^B_{i,j}$ after being adjusted for the Hungarian algorithm highlighted in blue seen in Eq: 5.2.

$$C_{i,j} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

$$C^B_{i,j} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} \tag{5.2}$$

Fig. 5.1 Unmodified: $C_{i,j} = 0$

Fig. 5.2 Modified: $C^B_{i,j} = max(row\ cost) + 1$

Intuitively, the *unmodified matrix* $C_{i,j}$ above shows that only 'Student 1' ranks project 1 as their highest preference, only 'Student 2' ranks project 2 as their highest preference, and only 'Student 3' ranks project 3 as their highest preference. Hence, the optimal solution can be naively seen to be without any use of solvers or integer programming as:

$$X_{i,j} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

However, with the **Hungarian Method**, the *modified matrix* $C^B_{i,j}$ is used to find the optimal allocation using the algorithm outlined by Sivarethinamohan below [37]:

1. **Finding the opportunity cost matrix**:

   - **Subtracting the smallest entry in each row from all the entries of its row.** Therefore, subtracting each row by 1 results in:

$$C^B_{i,j} = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{5.4}$$

- **Subtracting the smallest entry in each column from all the entries of its column.**
  However, since all columns have a zero, subtracting by 0 which has no effect and leaves the matrix unchanged from above. Notice, only 3 lines are required to cover all zeros.

2. **Drawing lines through appropriate rows and columns ensuring that all the zero entries of the cost matrix are covered and the minimum number of such lines is used.**
   This results in:

$$C^B_{i,j} = \begin{bmatrix} \cancel{0} & \cancel{1} & \cancel{2} \\ 2 & 0 & \cancel{1} \\ \cancel{1} & \cancel{1} & \cancel{0} \end{bmatrix}$$

Fig. 5.3 Crossed rows so that all zero entries are covered

3. **Once the all zeros are covered, we check for optimality.**
   Since, the minimum number of covering lines is equal to n i.e 3, an optimal assignment of zeros is found seen in Figure 5.4a, and the algorithm ends.

$$C^B_{i,j} = \begin{bmatrix} \textcircled{0} & 1 & 2 \\ 2 & \textcircled{0} & 1 \\ 1 & 1 & \textcircled{0} \end{bmatrix} \qquad X_{i,j} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(a) Optimal Assignment of Zeros (b) Student Project Allocation Matrix

Fig. 5.4 Final Allocation

The objective function of the Hungarian Method is to find an assignment of the projects to the students ($X_{i,j}$), such that each project is assigned to one student and vice-versa resulting in a minimum total cost of assignment. Summing up the total cost of matrix $X_{i,j}$ shown in Figure 5.4b, results in a minimum objective value of $Z_H = 3$.

Practically:
Utilizing Hirsch's implementation of the Hungarian Method with VBA in Excel 2019 with cost matrix $C^B_{i,j}$ above yields the following allocation matrix $X_{i,j}$ [46]:

Input, Cost Matrix $C^B_{i,j}$				
		Project 1	Project 2	Project 3
	Student 1	1	2	3
	Student 2	3	1	2
	Student 3	2	2	1

(a) Input Matrix: $C^B_{i,j}$

Output, Allocation Matrix $X_{i,j}$				
		Project 1	Project 2	Project 3
	Student 1	1		
	Student 2		1	
	Student 3			1

(b) Output Matrix: $X_{i,j}$

Fig. 5.5 Practical VBA Implementation of Hungarian Method using Excel 2019

The maximum number of operations needed to solve completely any $n \times n$ assignment problem, N_{max} is shown below [40]:

$$\begin{aligned} N_{max} &= \frac{n}{6}\left(11n^2 + 12n + 31\right) \\ N_{max} &= \frac{3}{6}\left(11(3)^2 + 12(3) + 31\right) \\ N_{max} &= 83 \text{ operations} \end{aligned} \tag{5.5}$$

Interestingly, it is worth noting that the Hungarian algorithm requires 83 operations to solve a 3×3 assignment problem which is far worse than the complete enumeration method outlined in Equation 2.4. The complete enumeration method would only require $n!$ operations, hence $3! = 6$ operations to solve the same assignment problem. This exhibits the complete enumeration method performing ≈ 14 times faster than the Hungarian Method for $n = 3$. As such, for $n < 6$ the complete enumeration method performs faster than the Hungarian method [40]. Consequentially, for $n \geq 6$ the value of the Hungarian methods' N_{max}, is much smaller than the complete enumeration methods' $n!$ hence declaring the former method more computationally efficient than the latter for large problems.

It is worth noting that N_{max} is generally larger than the number of operations needed to solve the assignment problem using some of the other methods such as Deepest Hole algorithm mentioned briefly that doesn't guarantee to always select the most-optimal overall cost

assignment [40]. Hence, we will focus on testing the BIP SPA models formulated in this paper to solve the SPA problem both efficiently and optimally.

5.2 Model 2

Objective Function: Allocate projects to students based on their preference rankings.

Conceptually:
To initially test out this model, we input the following simple scenario: We have 3 students, 5 projects proposed by lecturers and each student is required to rank 3 project preferences. Hence, $T_S = 3$, $T_P = 5$ and $T_C = 3$.

Identical to the Hungarian Method, this SPA model takes an input matrix $C_{i,j}$ that are the project preferences of the student i regrading project j.

$$C_{i,j} = \begin{bmatrix} 1 & 2 & 3 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & 1 & 2 & 3 \end{bmatrix} \tag{5.6}$$

The above matrix shows that only 'Student 1' ranks project 1 as their highest preference, only 'Student 2' ranks project 2 as their highest preference, and only 'Student 3' ranks project 3 as their highest preference. Hence, the optimal solution can be naively seen as:

$$X_{i,j} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{5.7}$$

Given that the objective function is as follows, as seen in 3.16,

$$\text{Minimize } Z_2 = \sum_{i=1}^{T_S} \sum_{j=1}^{T_P} (C_{i,j} \times X_{i,j}) \tag{5.8}$$

Minimizing the total sum of the rankings for all students. Hence, the optimal solution to the objective function can be manually calculated in this manner:

$$\begin{aligned} \text{Minimize } Z_2 = {} & C_{1,1}X_{1,1} + C_{1,2}X_{1,2} + C_{1,3}X_{1,3} + C_{1,4}X_{1,4} + C_{1,5}X_{1,5} \\ & + C_{2,1}X_{2,1} + C_{2,2}X_{2,2} + C_{2,3}X_{2,3} + C_{2,4}X_{2,4} + C_{2,5}X_{2,5} \\ & + C_{3,1}X_{3,1} + C_{3,2}X_{3,2} + C_{3,3}X_{3,3} + C_{3,4}X_{3,4} + C_{3,5}X_{3,5} \end{aligned} \tag{5.9}$$

$$\text{Minimize } Z_2 = 1+1+1 = 3 \tag{5.10}$$

Any other allocation (change in $X_{i,j}$) would have led to $Z_2 > 3$ which is a sub-optimal solution to the problem.

Practically:
Given we can infer the optimal solution above, this expected solution will be compared to the practical result obtained from implementing the integer program. Model 2 and the test data above are implemented using python's optimization library PuLP and Gurobi as the solver discussed in the previously in Chapter: (4)
The optimization model results:

```
Optimal solution found (tolerance 1.00e-04)
Best objective 3.000000000000e+00, best bound 3.000000000000e+00, gap 0.0000%
Gurobi status= 2
Status: Optimal
[['1.0' '0.0' '0.0' '0.0' '0.0']
 ['0.0' '1.0' '0.0' '0.0' '0.0']
 ['0.0' '0.0' '1.0' '0.0' '0.0']]
Objective Fnc=  3.0
```

Fig. 5.6 Model 2 - Test 1 Allocation Result

As can be observed from the Gurobi solver, the student project allocation result is identical to the allocation expected above 5.7 :

$$X_{i,j} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{5.11}$$

The optimal solution to the objective function (5.8) is $Z_2 = 3$ as stipulated in the manual calculation, hence we have verified that Model 2 is valid and functioning as expected.

5.3 Model 3

Objective Function: Allocate projects to students based on their preference rankings whilst putting an upper-bound (T_{PS}), on the number of projects each lecturer can supervise.

Conceptually:
To initially test out this model we input the following simple scenario: We have 3 students, 5 projects proposed by 2 lecturers and each student is required to rank 3 project preferences. Hence, $T_S = 3$, $T_P = 5$,$T_C = 3$, $T_L = 2$ and T_{PS} will be defined below.

This SPA model uses three matrices: $C_{i,j}$, $P_{k,j}$ and $X_{i,j}$. Matrix $X_{i,j}$ is the student project allocation output from the optimization problem that is used as an input in the problem constraint. $C_{i,j}$ and $P_{k,j}$ are input matrices, where $C_{i,j}$ is a matrix whose entries represent the ranking of project j indicated by student i and $P_{k,j}$: Is a binary variable that determines whether lecturer k proposed project j. The matrices are defined as follows:

$$C_{i,j} = \begin{bmatrix} 1 & 2 & 3 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & 1 & 2 & 3 \end{bmatrix} \tag{5.12}$$

$$P_{k,j} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{5.13}$$

The matrix $X_{i,j}$ is the output of the optimization problem used as an input in the constraint. Given the values of $C_{i,j}$ above, the resulting student project allocation matrix $X_{i,j}^{\intercal}$ is,

$$X_{i,j}^{\intercal} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.14}$$

The objective function as seen in 5.8 results to,

$$\text{Minimize } Z_3 = \sum_{i=1}^{T_S} \sum_{j=1}^{T_P} (C_{i,j} \times X_{i,j}) = 3 \tag{5.15}$$

The objective function remains unchanged from Model 2, however, the added constraint in Model 3 ensures that each lecturer can only supervise a given number of projects, T_{PS}. Hence the lecturer workload constraint is as follows:

$$\sum_{k=1}^{T_L} S_{k,i} = \sum_{j=1}^{T_P} P_{k,j} \times \sum_{i=1}^{T_S} X_{i,j}^{\intercal} \leq T_{PS} \tag{5.16}$$

Using testing matrices above $X_{i,j}$ and $P_{k,j}$,

$$\sum_{k=1}^{T_L} S_{k,i} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

When $T_{PS} = 2$,

$$S_{k,i} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{5.17}$$

When $T_{PS} = 1$, the problem is infeasible as it limits each lecturer to supervise one student. Hence, only 2 students can be supervised leaving the 3rd student without supervision which is an infeasible scenario as every student should be supervised by a lecturer i.e every student must be allocated to a project.

Practically:
Given we can infer the optimal solution above, this expected solution will be compared to the practical result obtained from implementing the integer program. Model 3 and the test data above are implemented resulting in:

When $T_{PS} = 2$,

```
Optimal solution found (tolerance 1.00e-04)
Best objective 3.000000000000e+00, best bound 3.000000000000e+00, gap 0.0000%
Gurobi status= 2
Objective Fnc=  3.0
Status: Optimal
S{k,i} for all k:
[[1. 0. 1.]
 [0. 1. 0.]]
S{k,i}: [2. 1.]
```

Fig. 5.7 Model 3 - Test at $T_{PS} = 2$ Student Supervisor Result

As observed above, the optimal objective value is 3.0 indicating the student project allocation matrix $X_{i,j}$ is the same as inferred above. Given a limit of $T_{PS} = 2$, has Lecturer 1 supervising two students and Lecturer 2, supervising one student ensure all three students in this scenario are supervised within the constraint.

When $T_{PS} = 1$,

```
Model is infeasible
Best objective -, best bound -, gap -
Gurobi status= 3
Objective Fnc=  None
Status: Infeasible
S{k,i} for all k:
[[nan nan nan]
 [nan nan nan]]
S{k,i}: [nan nan]
```

Fig. 5.8 Model 2 - Test at $T_{PS} = 1$ Student Supervisor Result

As inferred conceptually above, the problem is infeasible as the T_{PS} bound is too tight to ensure all students are supervised. Since the practical implementation results are identical to the expected results are stipulated in the conceptual section above, we have verified that Model 3 is valid and functioning as required.

Prior to applying the realistic data to the tested and verified models, the input $C_{i,j}$ needed to be populated with the realistic 2018/2019 EEE graduating class project preferences. This was a challenge as the data available was an incompatible format that listed students and their respective project preferences rather than a matrix whereby the respective ranking of every project for each student is given. As such, a Python data processing program was written to translate the students preferences from the given format to the required non-negative matrix format of dimension 'students $\times$ projects'. This program used JSON to create project name aliases and output the 'student project preference' matrix in .csv format.

The source-code for all implementations and supporting functions can be found in the Appendix (A) which points to a Github repository.

Chapter 6

Results

This chapter comprises of the results yielded from applying the '2018/2019 Student Project Data' to the BIP SPA models formulated in this paper. This chapter explores step 3 using realistic data as well as performs post-analysis as outlined by step 4 of the modeling process (3).

3. Solving the mathematical program (using realistic data)
4. Performing post-optimal analysis

The previous chapter showed how functional correctness is established for the SPA optimization models formulated in Chapter 4. This chapter delves into the qualitative and quantitative analysis of how well the SPA optimization models perform.

6.1 Current EEE/EIE Department SPA Results

The current implementation derives its data from the 2018/2019 EEE/EIE Final Year Project data used in the EEE department at Imperial College London. The anonymised 137 student project preferences data was cleaned to ensure,

- Consistency of data
- Exclusion of self-proposed project students
- Exclusion of students allocated projects in the Department of Computing (DoC)
- Exclusion of students that were allocated projects manually by splitting popular projects into 2 or 3 to accommodate preferences.

Out of the 137 students, data analysis highlighted 26 students had self-proposed projects, 2 students were allocated projects in DoC, whilst 15 students had projects allocated to them post splitting of popular projects manually. As such, these 43 students are excluded from analysis as their preferences were not run in the department algorithm. The algorithm allocated projects to 94 students out of the whole student body of 137 students. Hence, although there are 137 students i.e $T_S = 137$, considering the exclusions $T_S = 94$.

Although 193 projects were proposed i.e $T_P = 193$, to speed up the optimization algorithm, projects that were not shortlisted as a preference by any student were excluded. As 12 projects were not listed as a preference by any student, $T_P = 181$ from the original $T_P = 193$. It is worth noting provided 10 choices of preferences, the average number of choices utilized by the 94 students was ≈ 5 choices.

6.1.1 Ranking Analysis

The realistic data has the following statistics: $T_S = 94$, $T_P = 181$, and $T_C = 10$. Table 6.1 depicts the project allocations for the EEE 2018/2019 cohort. The algorithm allocated student preferences in the range: Rank 1 - Rank 8 with no students being allocated their 7th, 9th or 10th preferences. As shown, 54% of students were allocated their 1st preference, 14% their 2nd preference, 11% their 3rd preference, 7% their 4th preference, 11% their 5th preference, 2% their 6th preference, and 1% their 8th preference.

Rankings	**Students Allocated**	**Percentage**
Rank 1	51	54%
Rank 2	13	14%
Rank 3	10	11%
Rank 4	7	7%
Rank 5	10	11%
Rank 6	2	2%
Rank 7	0	0%
Rank 8	1	1%
Rank 9	0	0%
Rank 10	0	0%
Total	94	

Table 6.1 EEE 2018/2019 Algorithmic Allocation (94 Students)

Rankings	Students Allocated
Total Algorithmic Allocated	94
Manual Allocation	15
Total	109

Table 6.2 EEE 2018/2019 Algorithmic and Manual Allocation (109 Students)

6.1.2 Number of Projects Each Lecturer Supervises

The '2018/2019 Student Project Data' had the following statistics with T_L now taken into account whereby T_L is the number of lecturers: $T_S = 109$, $T_P = 181$, $T_C = 10$, $T_L = 57$. Analysis of the EEE 2018/2019 departmental implementation is shown in Table 6.3 where 14% of lecturers supervised 0 projects, 30% of lecturers supervised 1 project, 25% of lecturers supervised 2 projects, 19% of lecturers supervised 3 projects, 9% of lecturers supervised 4 projects, 2% of lecturers supervised 5 projects, and 2% of lecturers supervised 6 projects.

Number of Lecturers Supervising,	Number of Lecturers	Percentage
0 Projects	8	14%
1 Project	17	30%
2 Projects	14	25%
3 Projects	11	19%
4 Projects	5	9%
5 Projects	1	2%
6 Projects	1	2%
Total	57	

Table 6.3 EEE 2018/2019 Algorithmic Allocation (109 Students)

6.2 Model 2 SPA Integer Programming Results

Objective Function: Allocate projects to students based on their preference rankings.

$$\text{Minimize } Z_2 = \sum_{i=1}^{T_S} \sum_{j=1}^{T_P} (C_{i,j} \times X_{i,j}) \tag{6.1}$$

Prior to analyzing the results from Model 2 and Model 3 using the 2018/2019 EEE student project data, it is critical to define variables to aid in measuring the effectiveness of these models:

Definitions:

(D_1) **Student Satisfaction,** τ_1: To evaluate which implementation is more student-optimal we compare the student satisfaction achieved by each algorithm - to do so, *Student Satisfaction*, τ_1 needs to be defined. Students are happier if they're allocated either of their top three preferences i.e projects shortlisted as Rank 1, Rank 2 or Rank 3. Whereas students are less happy if they are allocated projects ranked lower than their third preference i.e Rank 4 - Rank 10. Hence the *Student Satisfaction*, τ_1 will represent the percentage of students allocated one of their top 3 project preferences (Rank 1, Rank 2 or Rank 3).

Student Satisfaction, τ_1:

$$\tau_1 = \frac{1}{T_S}\left(\sum_{x=1}^{3} n_x\right) \times 100\% \tag{6.2}$$

Where,

- n_x represents the number of students allocated their x-th project preference where, $x \in [1,3]$.
- T_S represents the total number of students.

(D_2) **Lecturer Satisfaction,** τ_2: Similarly, to evaluate which implementation is more lecturer-optimal we compare the lecturer satisfaction achieved by each algorithm - to do so, *Lecturer Satisfaction*, τ_2 needs to be defined. As mentioned briefly in Chapter 3, Dr. Clarke - Head of Student Project Allocation within the EEE Department at Imperial College London noted that lecturers who supervised a maximum of 3 projects or fewer had seen their supervisees perform better than students that were being supervised by lecturers who supervising 4 or more projects. Furthermore, he noted that lecturers are much happier when supervising 3 or fewer projects due to the manageable workload. Hence, *Lecturer Satisfaction*, τ_2 is defined as the percentage of lecturers who are allocated to supervise a maximum of 3 projects or fewer (this is asserted in Model 3 when $T_{PS} = 3$).

Lecturer Satisfaction, τ_2:

$$\tau_2 = \frac{1}{T_L}\left(\sum_{x=1}^{3} n_x\right) \times 100\% \tag{6.3}$$

Where,

- n_x represents the number of lecturers supervising x number of projects where, $x \in [1,3]$.
- T_L represents the total number of lecturers.

To interpret the effectiveness of the student satisfaction, τ_1 rate, and lecturer satisfaction τ_2 rate, we refer to Table 6.4 shown below derived from commonly used ranges to establish satisfaction quality. This table correlates the satisfaction percentage with a qualitative description to ease understanding of the satisfaction achieved by the algorithm being evaluated.

Student and Lecturer Satisfaction / %	
Very Satisfied	> 80
Satisfied	60 - 80
Average	40 - 60
Dissatisfied	20 - 40
Very Dissatisfied	< 20

Table 6.4 Quantitative Satisfaction Values with Corresponding Qualitative Descriptions

6.2.1 Ranking Analysis

Whereas the department allocation algorithm allocated 94 students algorithmically and 15 students manually, Model 2 will allocate the total of 109 students using the integer programming optimization model. Recall, the data has the following statistics: $T_S = 109$, $T_P = 181$, and $T_C = 10$. Table 6.5 depicts the project allocations using the optimization model - Model 2 (3.2). This optimization model allocates students preferences in the range: Rank 1 - Rank 5 with no students being allocated their 6th, 7th, 8th, 9th or 10th preferences. As shown, 57% of students were allocated their 1st preference, 24% their 2nd preference, 9% their 3rd preference, 7% their 4th preference, and 3% their 5th preference. This yielded an objective value of 191 from the objective function in Model 2 seen in Equation 6.1.

6.2.2 Number of Projects Each Lecturer Supervises

Notice the objective function of Model 2 (5.2) does not take into account the number of projects each lecturer supervises, and only maximizes the student satisfaction by allocating students their most preferred project where possible. As such, the variables T_L and T_{PS} are not included in the optimization problem, where T_L is the number of lecturers proposing

Rankings	Students Allocated	Percentage
Rank1	62	57%
Rank 2	26	24%
Rank 3	10	9%
Rank 4	8	7%
Rank 5	3	3%
Rank 6	0	0%
Rank 7	0	0%
Rank 8	0	0%
Rank 9	0	0%
Rank 10	0	0%
Total	109	

Table 6.5 Model 2 - Student Project Allocation Results

a project and T_{PS} is the upper bound of the number of projects each lecturer is allowed to supervise.

Model 2 has an unbounded limit to the number of projects each lecturer can supervise maximizing the student rankings only. Hence, it was initially expected that the current algorithm and Model 2 would achieve identical results due to Model 2's objective function being lecturer-independent. However, in this particular case, integer programming performs differently from the current algorithm where the impact on the lecturers without an upper bound T_{PS} is shown below in Table 6.6. It is observed that 14% of lecturers supervised 0 projects, 33% of lecturers supervised 1 project, 23% of lecturers supervised 2 projects, 16% of lecturers supervised 3 projects, 7% of lecturers supervised 4 projects, 5% of lecturers supervised 5 projects, and 2% of lecturers supervised 6 projects.

Number of Lecturers Supervising,	Number of Lecturers	Percentage
0 Projects	8	14%
1 Project	19	33%
2 Projects	13	23%
3 Projects	9	16%
4 Projects	4	7%
5 Projects	3	5%
6 Projects	1	2%
Total	57	

Table 6.6 Model 2 - Lecturer Supervision Allocation Results

6.3 Model 3 SPA Integer Programming Results

Objective Function: Allocate projects to students based on their preference rankings whilst putting an upper-bound (T_{PS}), on the number of projects each lecturer can supervise.

6.3.1 Rankings Analysis

As T_{PS} is incremented by one, the maximum possible supervision of each lecturer is increased which inherently loosens the constraints on the project rankings assigned to each student allowing for an improved allocation (smaller objective value) leading to a higher student satisfaction. This theory prompts a hypothesis that states, as student satisfaction τ_1 increases, lecturer satisfaction τ_2 decreases which is proved in Figure 6.1 below.

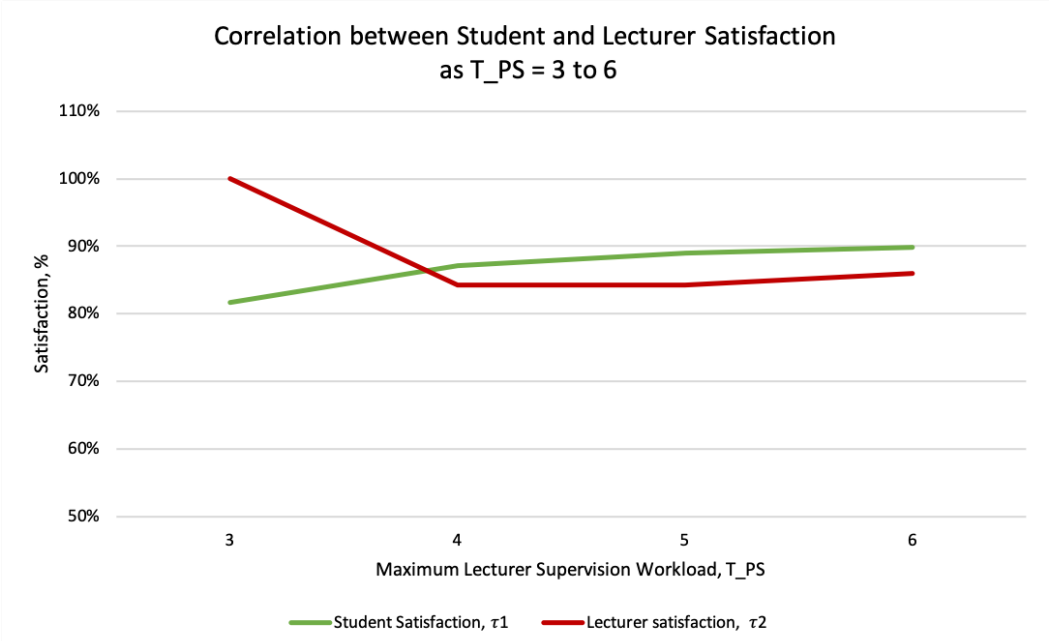


Fig. 6.1 Variation of Student and Lecturer Satisfaction as T_{PS} increases

It is important to note the previous model (Model 2) has a lecturer who is supervising up to 6 projects. Hence, assigning $T_{PS} = 6$ in Model 3 is expected to and yields the same ranking as Model 2. Table 6.7 below depicts the variation of the objective value, student satisfaction and the worst ranking allocated as T_{PS} is increased. Observe in Figure 6.2 the objective values decreases as student satisfaction increases which is expected since increasing student satisfaction corresponds to increasing the number students being allocated their highest preferences i.e their low-value ranks which are summed up to yield the objective value Z_3.

Projects Supervised per Lecturer, T_{PS}	**3**	**4**	**5**	**6**
Objective Value, Z_3	235	204	195	191
Student Satisfaction, τ_1	81.65%	87.16%	88.99%	89.91%
Number of students allocated Rank 1 - Rank 3	89	95	97	98
Lowest Rank Allocated	7	6	6	5

Table 6.7 Model 3 - Student Satisfaction, τ_1 and Objective Value, Z_3

When $T_{PS} > 6$, the objective value is no longer improved and remains constant hence the variation of the objective value with respect to T_{PS} is shown below in Figure 6.2. This is because at $T_{PS} = 6$ the solution is degenerate which is analyzed further in Chapter 7.

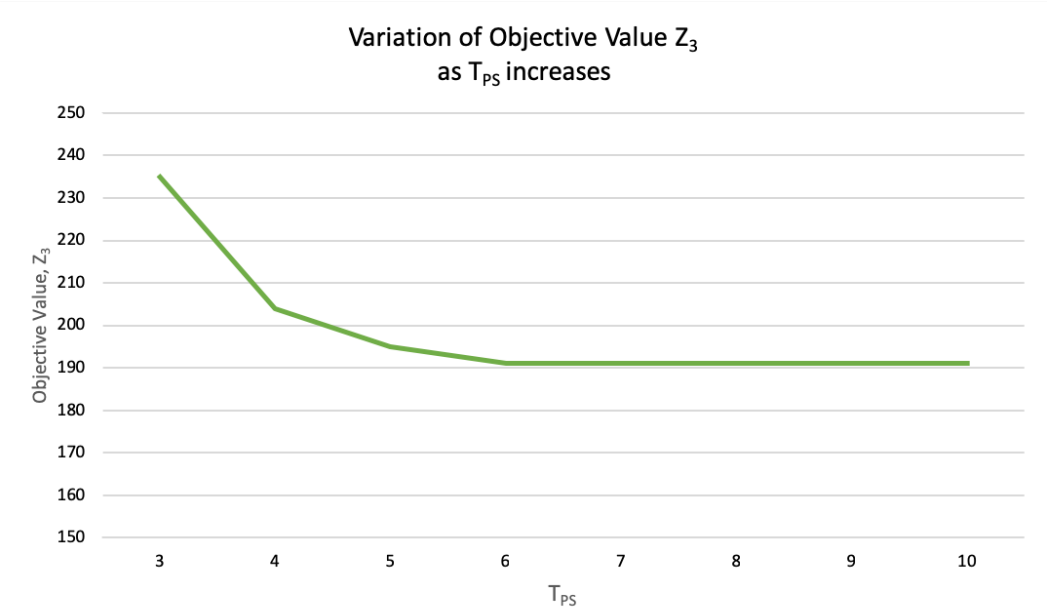


Fig. 6.2 Variation of Objective Value Z_3 w.r.t T_{PS}

6.3.2 Number of Projects Each Lecturer Supervises, T_{PS}

Table 6.8 below depicts the variation of the objective value, lecturer satisfaction, and the simplex iterations as T_{PS} is increased. Firstly, as Model 3's objective function is a minimization problem, we observe as T_{PS} increases the objective function decreases. This occurs because as T_{PS} increases, this allows more students to assigned to a particular lecturer which enables more students to be assigned to one of their higher-ranked projects. For example, if a student ranks their 1st preference to a project proposed by a lecturer already supervising 3 students, and the maximum workload $T_{PS} = 3$, the student will be assigned their 2nd, 3rd or lower-ranked project instead of their 1st option. However, had $T_{PS} > 3$ this student would

be allocated their first choice project consequently increasing the project workload for that specific lecturer from 3 projects to 4 projects. The former scenario with $T_{PS} = 3$ is a sub-optimal, as students are allocated lower-ranked projects (with correspondingly high-value) which consequently increases the objective function. Secondly, as T_{PS} increases, lecturer satisfaction decreases. This is attributed to the objective value reasoning above because as the objective value increases so does the lecturer satisfaction.

Projects Supervised per Lecturer, T_{PS}	**3**	**4**	**5**	**6**
Lecturer Satisfaction, τ_2	100%	84.21%	84.21%	85.96%
Simplex Iterations	289	184	170	169

Table 6.8 Model 3 - Lecturer Satisfaction, τ_2 and Simplex Iterations

Thirdly, as T_{PS} increases, the feasibility region increases, which allows for a more flexible problem space. This consequently decreases the number of simplex iterations required to solve the problem as depicted below in Figure 6.3

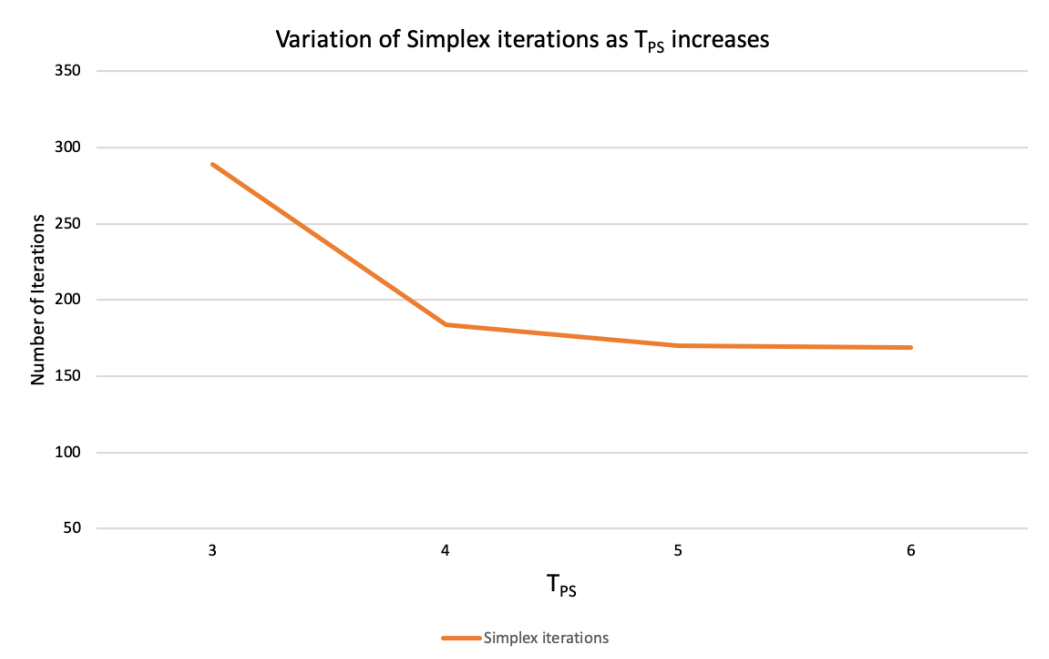


Fig. 6.3 Variation of Simplex Iterations w.r.t T_{PS}

Note: all results can be found in the Github repository listed in the appendix (A).

Chapter 7

Evaluation

This chapter comprises of the evaluation of the results gained from applying the '2018/2019 Student Project Data' to the BIP SPA models formulated in this paper compared to the departmental algorithm. This chapter explores the last step of the modeling process (3) evaluating and presenting the best solution to the SPA problem.

5. Presenting the solution and analysis

Prior to evaluating the current departmental algorithm and the BIP SPA integer models, recall the evaluation metrics defined in the previous section.

(D_1) **Student Satisfaction,** τ_1: Represents the percentage of students allocated one of their top 3 project preferences (Rank 1, Rank 2 or Rank 3) as defined in Item (D_1) in Chapter 6.

Student Satisfaction, τ_1:

$$\tau_1 = \frac{1}{T_S}\left(\sum_{x=1}^{3} n_x\right) \times 100\% \tag{7.1}$$

(D_2) **Lecturer Satisfaction**: Represents the percentage of lecturers who are allocated a maximum of 3 projects or less to supervise - in Model 3 this is when $T_{PS} = 3$ as defined in Item (D_2) in Chapter 6.

Lecturer Satisfaction, τ_2:

$$\tau_2 = \frac{1}{T_L}\left(\sum_{x=1}^{3} n_x\right) \times 100\% \tag{7.2}$$

To understand the effectiveness of the Student Satisfaction, τ_1 and Lecturer Satisfaction, τ_2 variables, we refer to Table 6.4 shown in Chapter 6.

7.1 Current Algorithm Evaluation

This section will evaluate the performance the allocation algorithm currently implemented in the EEE Department at Imperial College London to allocate students to their final year projects.

7.1.1 Ranking

The current gradient descent algorithm implemented in the EEE Department at Imperial College London has the following student satisfaction, τ_1 referring to the project ranks allocated in Table 6.1. The table depicts 51 students allocated their 1^{st} preference, 13 students allocated their 2^{nd} preference, and 10 students allocated their 3^{rd} preference we observe τ_1 below.

Student Satisfaction, τ_1:

$$\begin{aligned} \tau_1 &= \frac{1}{94}\Big(51+13+10\Big) \times 100\% \\ &= 78.72\% \end{aligned} \tag{7.3}$$

Referring to Table 6.4, the impact of $\tau_1 = 78.72\%$ has the current algorithm attaining a solid *'Satisfied'* student satisfaction level during the allocation of projects to the 94 students.

7.1.2 Lecturer Supervision Workload

This algorithm achieves the following lecturer satisfaction, τ_2 with 8 lecturers unallocated to a project, 17 lecturers supervising 1 project, 14 lecturers supervising 2 projects and 11 lecturers supervising 3 projects we observe τ_2 below.

Lecturer Satisfaction, τ_2:

$$\begin{aligned} \tau_2 &= \frac{1}{57}\Big(8+17+14+11\Big) \times 100\% \\ &= 87.72\% \end{aligned} \tag{7.4}$$

We can see that with regards to $\tau_2 = 87.72\%$, the current algorithm performs very well attaining a solid *'Very Satisfied'* lecturer satisfaction level during the allocation of projects for supervision to the 57 lecturers.

7.2 Model 2 Evaluation

This section will evaluate the performance of Model 2's integer programming approach compared to the existing current algorithm being used within the EEE Department at Imperial College London.

7.2.1 Rankings

Directly comparing the current algorithm and Model 2 as observed in Figure 7.1, Model 2 allocates 3% more students their 1^{st} choice, 10% more students their 2^{nd} choice thereby performing much better than the current algorithm in place, w.r.t the objective of allocating as many students as possible to their most preferred preferences.

Model 2 performs far better than the current algorithm with respect to student satisfaction τ_1, achieved using integer programming. Referring to Table 6.5, Model 2 allocates 62 students their 1^s choice, 26 students their 2^{nd} choice, and 10 students their 3^{rd} choice we see τ_2 below.

Student Satisfaction, τ_1:

$$\begin{aligned} \tau_2 &= \frac{1}{109}\Big(62+26+10\Big) \times 100\% \\ &= 89.91\% \end{aligned} \tag{7.5}$$

Model 2 results in an increase of 11.2% in the student satisfaction rate by among the students compared to the current implementation. This Model attains a solid *'Very Satisfied'* student satisfaction level during the allocation of projects to the 109 students.

The following graph depicts a comparable view of the student project rankings achieved by both algorithms where the integer programming approach of Model 2 is represented by the green bars and the current implementation is represented by the red bars.

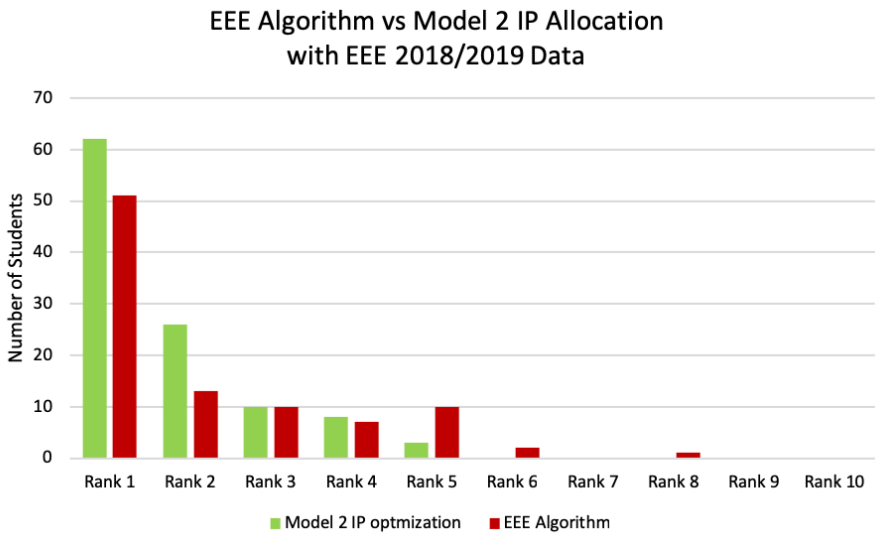


Fig. 7.1 Rankings Achieved: EEE Algorithm and Model 2 IP Allocation

7.2.2 Lecturer Supervision Workload

Although Model 2 was initially expected to perform identically to the current departmental implementation, due to Model 2's objective function being lecturer-independent, we observe from the results that Model 2 instead achieves a slightly lower lecturer satisfaction τ_2, than that achieved by current algorithm as shown in Figure 7.2. This optimization model achieves the following lecturer satisfaction, τ_2 with 8 lecturers unallocated to a project, 19 lecturers supervising 1 project, 13 lecturers supervising 2 projects, and 9 lecturers supervising 3 projects we see τ_2 below.

Lecturer Satisfaction, τ_2:

$$\begin{aligned} \tau_2 &= \frac{1}{57}\Big(8+19+13+9\Big) \times 100\% \\ &= 85.96\% \end{aligned} \tag{7.6}$$

With only a small lecturer satisfaction difference of 1.76% between the current algorithm and Model 2, it supports the initial hypothesis that due to the lecturer-independent objective function the two models would perform very similar to another regarding the lecturer supervision workload. As such even at $\tau_2 = 85.96\%$, Model 2 identically attains the same lecturer satisfaction level at *'Very Satisfied'*.

The following graph depicts a comparable view of the lecturer project workload achieved by both algorithms where the integer programming approach of Model 2 is represented by the green bars whereas the current implementation is represented by the red bars.

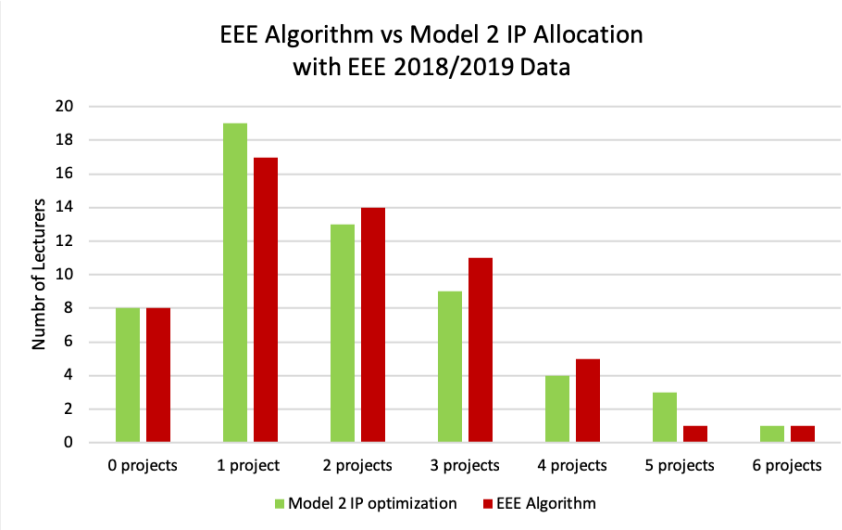


Fig. 7.2 Lecturer Workload Achieved: EEE Algorithm and Model 2 IP Allocation

7.3 Model 3 Evaluation

This section will evaluate the performance of Model 3's integer programming approach compared to the existing current algorithm being used within the EEE Department at Imperial College London. Model 3 focuses on achieving dual objectives, as it not only aims to maximize the student satisfaction but also to maximize the lecturer satisfaction. As such, referring to Table 6.7, we observe the student satisfaction increasing as T_{PS} increases.

7.3.1 Ranking

As expected student satisfaction, τ_1 and the objective value, Z_3 of Model 3 are **negatively correlated** - as student satisfaction, τ_1 increases the objective value Z_3 decreases and vice versa. This is observed in Figure 7.4a below and the data is presented in Table 6.7. Note that regardless of the T_{PS} value, this model attains a solid *'Very Satisfied'* student satisfaction level during the allocation of projects to the 109 students as the satisfaction rate increases from 81.65% to 89.91% as T_{PS} increases.

Model 3 at $T_{PS} = 3$ achieves 100% lecturer satisfaction, hence is the lecturer-optimal solution. Whereas, Model 3 at $T_{PS} = 6$ achieves 89.91% the highest achievable student satisfaction,

hence is the student-optimal solution. Figure 7.3 depicts a comparable view of the student project rankings achieved by Model 3 at the lecturer-optimal solution (blue bars), and the student-optimal solution (green bars), as compared to the current EEE algorithm (red bars).

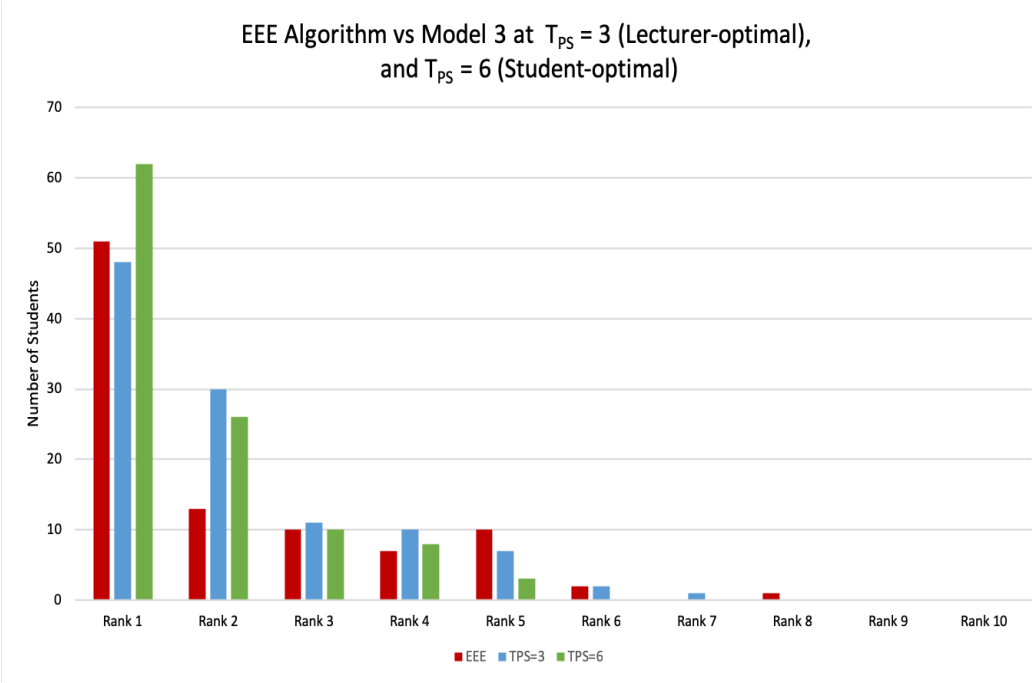
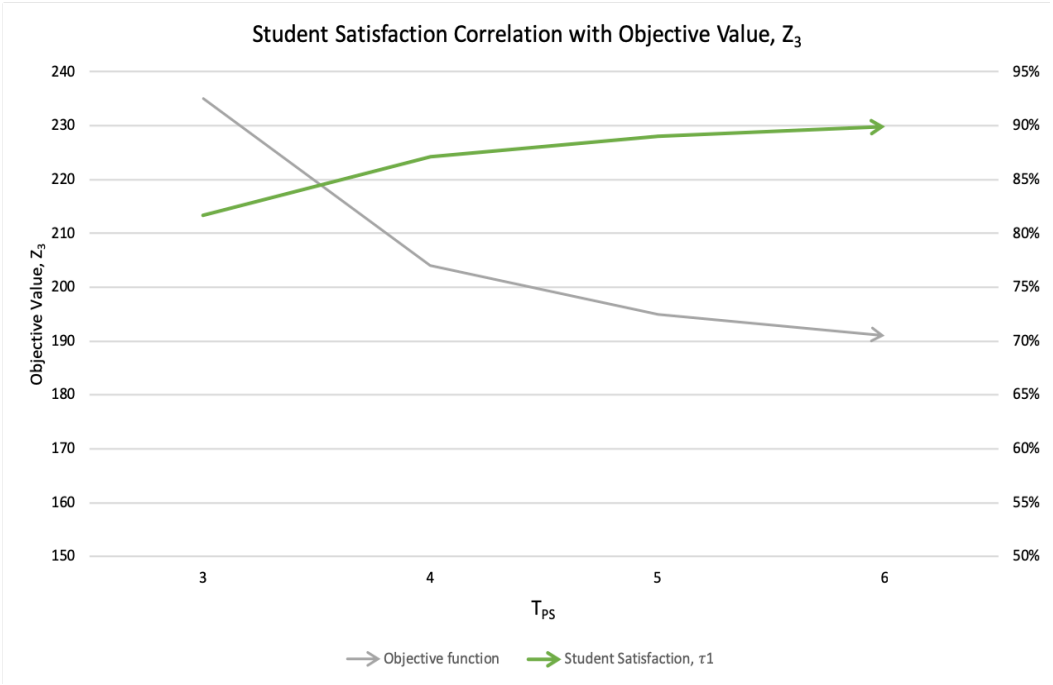
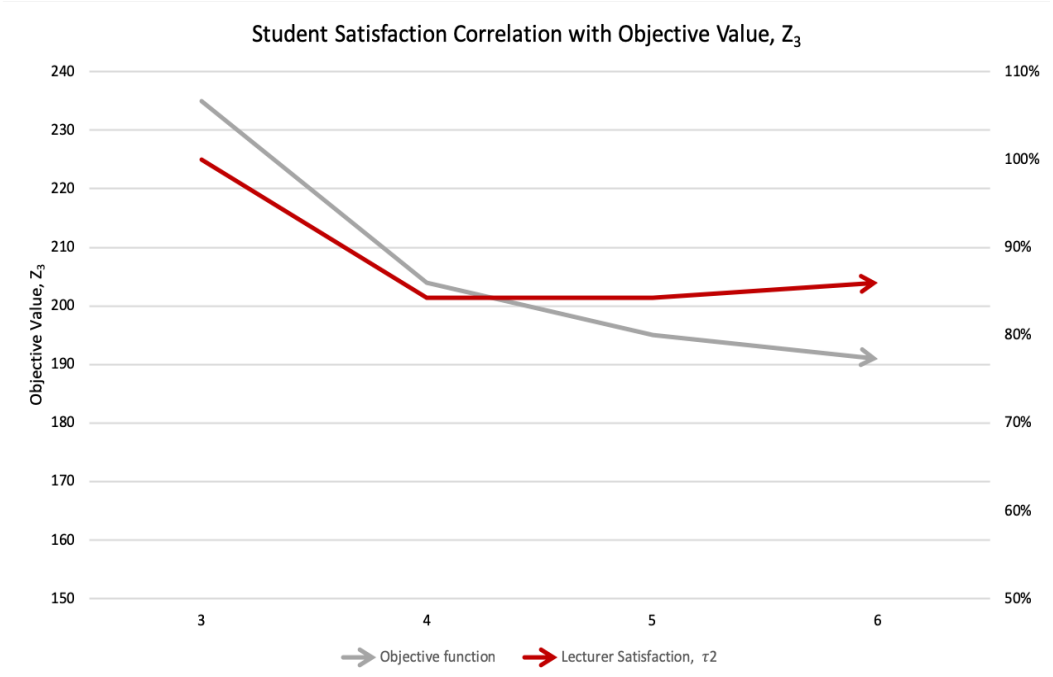


Fig. 7.3 Rankings Achieved: EEE Algorithm and Model 3 IP Allocation



(a) Student Satisfaction



(b) Lecturer Satisfaction

Fig. 7.4 Satisfaction Correlations to the Objective Value Z_3

7.3.2 Lecturer Supervision Workload

Unlike student satisfaction, lecturer satisfaction τ_2 and the objective value Z_3 of Model 3 are **positively correlated** - as lecturer satisfaction, τ_2 decreases the objective value Z_3 decreases and vice versa until $T_{PS} = 5$. When $T_{PS} = 6$ the correlation is no longer present as the objective value continues to decrease while the lecturer satisfaction increases rather than decrease. When $T_{PS} > 6$, the objective function and lecturer satisfaction both remain constant as this point generates a degenerate solution. The degeneracy halts any improvement in the objective value therefore not affecting the lecturers' satisfaction. This is observed in Figure 7.4b below and the data is presented in Table 6.7. Note that regardless of the T_{PS} value, this model also attains a solid *'Very Satisfied'* lecturer satisfaction level after the project allocation as the satisfaction rate decreases from 100% to 85.96% as T_{PS} increases.

Figure 7.6 compares the variation of student satisfaction and lecturer satisfaction with the number of students who get allocated their top three preferences (Rank 1 - Rank 3) with respect to T_{PS}. As expected, as the number of students who get allocated their top three preferences increases, so does the student satisfaction. However, we observe from the results in Figure 7.6b that the relationship is inversely regarding lecturer satisfaction. In the lecturer perspective, as students get happier, the lecturer project supervision workload T_{PS} increases, making them unhappy. However, at this point we are unaware whether the increase of T_{PS} creates unhappy lecturers uniformly or only affects particular lecturers. It is essential, to evaluate further what the impact is on each lecturer, as the maximum supervision limit T_{PS} increases. This will be examined using discrete sensitivity analysis in the following section.

Figure 7.5 depicts a comparable view of the lecturer project workload achieved by Model 3 at $T_{PS} = 3$ (blue bars), and by Model 3 at $T_{PS} = 6$ (green bars), as compared to the current EEE algorithm (red bars).

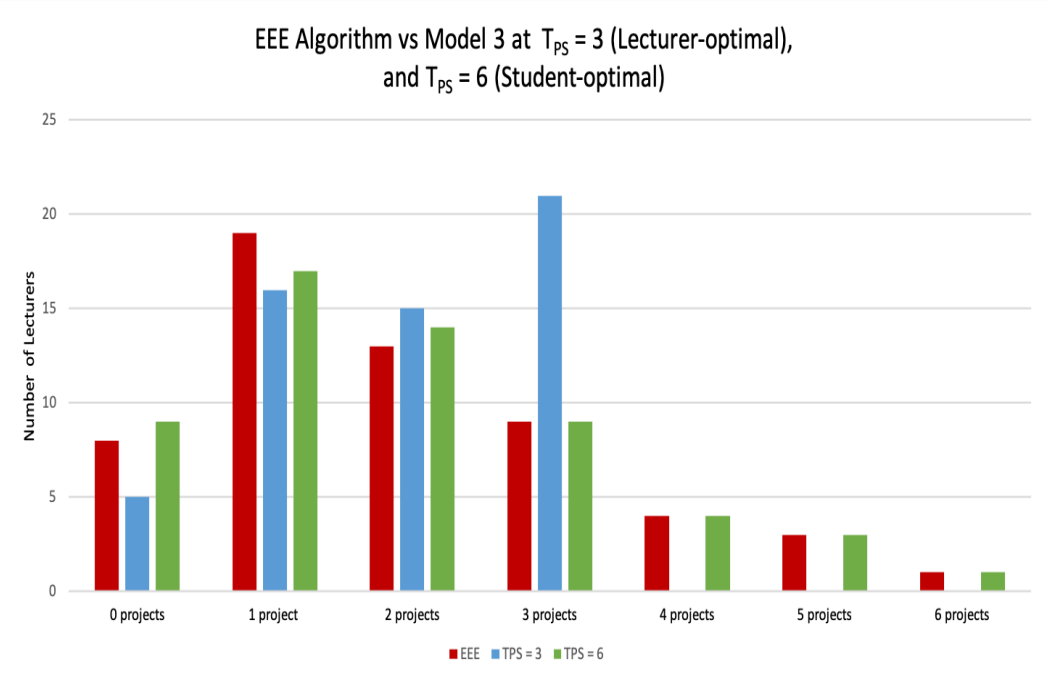


Fig. 7.5 Lecturer Workload Achieved: EEE Algorithm and Model 3 IP Allocation

7.3.3 Discrete Sensitivity Analysis

Given that we vary the right-hand-slide value, T_{PS}, of Model 3's lecturer constraint recalled below (7.7), it is important to carry out a sensitivity analysis to understand the impact of increasing T_{PS} as it gives an insight into possible technological improvements of the process being modeled. For example, it may be the case that the available resources are not balanced properly in the model, in which case, sensitivity analysis assists in discerning whether the issue can be solved by reallocating the current resources or by acquiring additional resources to eliminate possible bottlenecks [47].

$$\begin{aligned} &\sum_{i=1}^{T_S} S_{k,i} = \sum_{j=1}^{T_P} P_{k,j} \times \sum_{i=1}^{T_S} X_{i,j}^{\mathsf{T}} \\ &\sum_{i=1}^{T_S} S_{k,i} \equiv S_k, \text{ where, } 1 \leq k \leq T_L \\ &S_k \leq T_{PS}, \text{ where, } 1 \leq k \leq T_L \end{aligned} \tag{7.7}$$

Sensitivity analysis investigates the effect of changes to the right-hand-side (RHS) variables on the optimal value of the objective function and the range of validity of these effects [48]. The range of validity refers to the range over which the value of a single RHS variable can vary without changing the basis associated with an optimal solution. The simplex method plays a vital role in sensitivity analysis as this method allows for easy determination of which variables leave the basis as changes occur to the constraint RHS values. As Lin and Wen observed, the optimal basic solution of the assignment problem is inherently degenerate, in that, the optimal bases changes but the optimal assignment is unchanged. The optimal assignment corresponds to several optimal basic solutions/bases with the same non-degenerate basic variables and different degenerate basic variables [49].

Degeneracy occurs when the objective value and basic solution does not change when there is an exiting variable in the simplex method i.e one of the constraint RHS values in the simplex tableau is equal to zero thereby yielding a degenerate basic feasible solution.

What a degenerate solution informs us about the model, is that at least one constraint is redundant and the optimum point, in this case, would be over-determined [50]. This is because two different bases correspond to the same basic feasible solution/allocation at this optimum point. As such, the knowledge that at least one of the resources is surplus is valuable to further optimize the model and its implementation. We are going to determine the values of T_{PS} that yield degenerate solutions:

1. **At $T_{PS} < 3$, infeasible solutions are realized:**
   In this range, the Model 3 derives no student project allocation that satisfies all constraints at $T_{PS} \leq 3$ as expected. This infeasible solution indicates there is no allocation that ensures all lecturers are supervising a maximum of 2 projects/students. It is worth noting that the slack variables will *always* provide a feasible solution if *all* the constraints are‘ $\leq$’ with non-negative RHS. This is because, for other types of constraints, artificial variables are used for commutation as opposed to slack variables. However, this is not the case with Model 3 one of the constraints is an ‘$=$’ constraint rather than ‘$\leq$’ constraint.

2. **At $3 < T_{PS} < 6$, feasible optimal solutions are realized:**
   In this range, the simplex algorithm attempts to increase a non-basic variable and if there is no degeneracy, the non-basic variable will be positive after the pivot and the objective value is improved [51]. This situation occurs in Model 3 in that as T_{PS} varies within this scope, the objective value is improved i.e decreased. Hence, we will carry

out sensitivity analysis on the optimal solution and its respective objective value to changes in the value of T_{PS} i.e the lecturer supervision constraint (7.7) within this range. For ease of understanding, T_{PS} is also described as the maximum number of supervision slots each lecturer is allowed to have.

To analyze these feasible optimal solutions, we define the following:

(D_1) Lecturers' Acceptable Supervision, $\eta_j^{T_{PS}}$,
the number of lecturers supervising j projects given the maximum supervision limit, T_{PS}. Hence, for example, $\eta_5^6 = 3$ indicates 3 lecturers supervised 5 projects given the maximum supervision of 6 projects. Hence, as the maximum supervision limit increases within the range $3 \leq T_{PS} \leq 6$ the lecturer acceptable supervision values, $\eta_j^{T_{PS}}$, produced in Model 3 are seen below in Table 7.1.

Projects Supervised, j \ Max Limit, T_{PS}	3	4	5	6
0	5	6	9	9
1	16	19	17	17
2	15	15	14	14
3	21	8	8	9
4	0	9	5	4
5	0	0	4	3
6	0	0	0	1

Table 7.1 Values of $\eta_j^{T_{PS}}$ for $j \in [0, ..., T_{PS}]$

(D_2) Shadow Price, Π,
the rate of change of increasing the maximum supervision limit, T_{PS} from a to b resulting in a change in the objective value from Z_a to Z_b respectively:

$$\begin{aligned} \Pi &= \frac{\text{Change in objective value, } \Delta Z_3}{\text{Change in supervision slots, } \Delta T_{PS}} \\ \Pi &= \frac{Z_b - Z_a}{b - a} \\ \Pi &= \frac{\Delta Z_3}{\Delta T_{PS}} \end{aligned} \tag{7.8}$$

The shadow price provides a direct link between the model input (lecturers' maximum supervision limit, T_{PS}) and its objective value (sum of project ranks allocated to students) thereby providing a direct link between the maximum supervision limit, T_{PS}, and student satisfaction, τ_1.

Consider the following cases as T_{PS} varies within $3 < T_{PS} < 6$:

- **As T_{PS} increases from 3 projects to 4 projects:**
  The rate of change in the objective value that results from increasing the lecturers' maximum supervision from 3 projects to 4 projects, also known as the shadow price Π, can be computed as follows:

$$\begin{aligned} \Pi &= \frac{\text{Change in objective value}}{\text{Change in supervision slots,}} \\ &= \frac{204-235}{4-3} \\ &= -31 \end{aligned} \tag{7.9}$$

  The shadow price must be normalized with the number of lecturers directly affected with this change. This change is denoted by Z, when T_{PS} is incremented by 1. In this scenario, when T_{PS} is increased from 3 to 4, Z represents the number of lecturers who were initially supervising 3 projects and after the increase to $T_{PS} = 4$ are now supervising 4 projects. This value is represented by calculating the difference between $\eta_j^{T_{PS}+1}$ and $\eta_j^{T_{PS}}$ at $j = T_{PS}+1$ defined earlier (D_1).

  Hence, when $T_{PS} = 3$:

$$\begin{aligned} Z &= \eta_j^{T_{PS}+1} - \eta_j^{T_{PS}} \\ &= \eta_4^4 - \eta_4^3 \\ &= 9-0 = 9 \end{aligned} \tag{7.10}$$

  Therefore after normalization, $\Pi_{norm} = \frac{-31}{9} = -3.444$.

  The shadow price indicates increasing the maximum supervision workload from 3 to 4 projects directly impacts the objective value by -3.444 units. Given that the immediate change in objective value shown (7.9) is -31 before normalization, this indicates that the sacrifice the nine lecturers who initially supervised 3 projects now supervising 4 projects contributed to 11.11% of the objective value decrease. It is interesting to observe whilst nine lecturers had their supervision increased by 1 project (initially 3 projects to 4 projects), one lecturer had their supervision decreased by 2 projects (initially 3 projects to 1 project), six lecturers had their supervision decreased by 1 (initially 3/2 projects to 2/1 projects) and one lecturer

who was initially supervising 1 project ended up supervising zero projects as can be interpreted from Table 7.1. Clearly, the change from $T_{PS} = 3$ to $T_{PS} = 4$ is *not uniform* across lecturers.

- **As T_{PS} increases from 4 projects to 5 projects:**
  The rate of change in the objective value that results from increasing the lecturers' maximum supervision from 4 projects to 5 projects, can be computed as follows:

$$\begin{aligned} \Pi &= \frac{\Delta Z_3}{\Delta T_{PS}} \\ &= \frac{195-204}{5-4} \\ &= -9 \end{aligned} \tag{7.11}$$

  Normalization,

$$\begin{aligned} Z &= \eta_5^5 - \eta_5^4 \\ &= 4-0=4 \\ \Pi_{norm} &= \frac{-9}{4} = -2.25 \end{aligned} \tag{7.12}$$

  In this case, the shadow price shows increasing the maximum supervision workload from 4 to 5 projects directly impacts the objective value by -2.25 units. Given that the immediate change in objective value shown (7.11) is -9 before normalization, this indicates that the sacrifice the four lecturers who initially supervised 4 projects now supervising 5 projects contributed to a significant 25% of the objective value decrease. Note, whilst four lecturers had their supervision increased by 1 project (initially 4 projects to 5 projects), one lecturer had their supervision decreased by 1 project (initially 2 projects to 1 project), and three lecturers who were initially supervising 1 project ended up supervising zero projects as can be interpreted from Table 7.1. The change from $T_{PS} = 4$ to $T_{PS} = 5$ is *not uniform* across lecturers.

- **As T_{PS} increases from 5 projects to 6 projects:**
  The rate of change in the objective value that results from increasing the lecturers' maximum supervision from 5 projects to 6 projects, can be computed as follows:

$$\begin{aligned}\Pi &= \frac{\Delta Z_3}{\Delta T_{PS}} \\ &= \frac{191-195}{6-5} \\ &= -4\end{aligned} \tag{7.13}$$

Normalization,

$$\begin{aligned}Z &= \eta_6^6 - \eta_6^5 \\ &= 1-0=1 \\ \Pi_{norm} &= \frac{-4}{1} = -4\end{aligned} \tag{7.14}$$

The shadow price shows increasing the maximum supervision workload from 5 to 6 projects directly impacts the objective value by -4 units. Given that the immediate change in objective value shown (7.13) is -4 before normalization, this indicates that the sacrifice the single lecturer who initially supervised 5 projects now supervising 6 projects contributed wholly (100%) of the objective value decrease.

The lecturer satisfaction rate observed earlier in Table 6.8 indicated as T_{PS} increases, the general consensus of the lecturer population is less happy. However, the shadow prices have highlighted that as T_{PS} increases, individually, some lecturers get happier whilst other get overburdened and less happy. As such, there is no uniformity in ensuring all lecturers are happy and realistically, not every lecturer will be happy and satisfied with their project supervision workload. As such, relying on the lecturer satisfaction rate, τ_2, is the best approach to ensure that a *majority* of the lecturers are satisfied although not all lecturers will be.

3. **At $T_{PS} \geq 6$, degenerate optimal solutions are realized:**
   In this range when $T_{PS} > 6$, as T_{PS} increases the value of the objective function remains constant indicating that within the basic feasible solution, one of the basic variables is equal to zero. This inherently informs us that there two different bases that correspond to the same basic feasible solution yielding degenerate solutions.

Another way to analyze the impact on the objective value Z_3, as T_{PS} is increased by a single unit is to simply calculate the percentage change of the objective function shown in table

7.2. Recall, that ΔT_{PS} is the change in T_{PS} and ΔZ_3 is the change in the objective value of the Model 3 objective function. As observed in Table 7.2, as T_{PS} increases the percentage change of the objective function decreases. This occurs as the optimal solution and its respective objective value tend towards being degenerate thereby no longer improving the objective value and having it remain constant for $T_{PS} > 6$.

ΔT_{PS}	**3 to 4**	**4 to 5**	**5 to 6**
% ΔZ_3	-15.2%	-4.6%	-2.1%

Table 7.2 Change in Objective Value Z_3, as T_{PS} increases

7.4 Summary

Initial Objectives v. Project Outcomes:
Comparing the project's objectives with initial project specifications captured in section 1.2, it is observed that all the specifications have been met. Comparing the outcome with the initial requirements captured in section 1.3, it is observed that all '*Necessary*' requirements have been achieved. The '*Desirable*' requirements within the initial 'Mathematical Formulations': 1 have all been developed, however, the '*Desirable*' requirements within the initial 'Constraint Controls': 2 were not achieved. This was a conscious decision as implementing a model whereby the project preferences are ranked by weightings in percentages was deemed redundant in order to exhibit the impact of the integer programming approach in solving the SPA problem. Finally, the 'Project Extension: Group SPA Problem' was not implemented purposefully, as it would not deliver the versatility and extensive possibilities of the BIP approach. Instead, as an extension, the next chapter explores the impact and varied utility of the BIP method explored in this paper by adapting the models formulated in Chapter 3 to resource allocation problem faced in various and distinct industries. Thus, the SPA model formulated in this paper was applied to a Vehicle Routing problem occurring at taxi firms, the well-known Hospital/Residents problem and the ubiquitous Job/Worker problem thereby enriching the impact of using integer programming in solving resource allocation problems.

Project Scope:
The project scope differs from related work primarily on the integer programming aspect. Current approaches widely use linear programming, assignment methods such as the Hungarian, Munkres, Deepest Hole methods, and even gradient descent algorithms. Approaching a resource allocation problem formulated as an assignment problem solved using BIP, is unique

and rarely seen in literature due to the challenges attached to constricting variables to binary integers only. As can be derived from Table 7.3, the advantages of my approach compared to related work is that it is faster, and yields the most-optimal allocations achieving the highest student and lecturer satisfaction. Additionally, formulating the SPA problem with binary variables creates a realistic perspective, as all decision variables particularly to this problem are either 1 or 0, as a student is either allocated a project or not. Existing approaches form an unrealistic understanding of the SPA problem for example if linear programming is utilized, it is impractical for a student to be allocated to a 'half' or a 'third' of a project. However, the disadvantages of my approach are regarding computation; solving integer programs is much more difficult than solving linear programs. Furthermore, limiting the variables to binary integers although realistic in the SPA problem, makes it more difficult to model than a linear program.

Summary of the Most Effective Approach:
In summary, approaching the SPA problem via IP proves simple using Python, faster regarding execution time, and completely automated as it ensures that any feasible allocation set allocates projects to *all* students, otherwise the solution is rendered infeasible leaving no room for manual intervention. Table 7.3 clearly depicts the performance of the integer programmed models (Model 2 and Model 3) compared to the existing gradient descent algorithm implemented currently in the department, the Hungarian method which is a widely pervasive methodology used to solve assignment problems across several industries and complete enumeration. Note that the complete enumeration method was not implemented due to the lengthy computation time of 6.72 days to execute. Additionally, due to this lengthy duration, it was immediately eliminated in the running of which model is the most effective in solving the SPA problem.

On a lecturer-optimal approach, Model 3 at $T_{PS} = 3$ is the best approach as it achieves 100% lecturer satisfaction ensuring all lecturers are supervising 3 or fewer projects. As discussed earlier, the lecturer-independent objective function of Model 2 without an upper bound on the lecturer supervision workload resulted in a single lecturer supervising 6 lecturers. As such, this model yields the same allocation solutions as Model 3 at $T_{PS} = 6$, making them equivalent in the evaluation process. Therefore, on a student-optimal approach, Model 2 equivalently Model 3 at $T_{PS} = 6$, is the best achieving the highest student satisfaction rate at 89.91%. It is worth noting, if lecturers do not want to supervise up to 6 projects, then Model 3 at $T_{PS} = 5$, is a good compromise that results in relatively high student and lecturer satisfaction rates as seen in Figure 7.7. As observed from Table 7.3, although the Hungarian method, and Model 3 at $T_{PS} = 6$ have the same student satisfaction as Model 2, Model 2

	Execution Time	Simplex Iterations	Max Rank Allocated	Student Satisfaction	Lecturer Satisfaction
Model 2	$\approx$ 5 secs	169	6	89.91%	85.96%
Current Algorithm	$\approx$ 5 mins	N/A	6	78.72%	87.72%
Model 3 at T_{PS} = 3	$\approx$ 50 secs	289	7	81.65%	100.00%
Model 3 at T_{PS} = 4	$\approx$ 50 secs	184	6	87.16%	84.21%
Model 3 at T_{PS} = 5	$\approx$ 50 secs	170	5	88.99%	84.21%
Model 3 at T_{PS} = 6	$\approx$ 50 secs	169	5	89.91%	85.96%
Hungarian Method	66.72 secs	N/A	6	89.91%	85.96%
Complete Enumeration	6.72 days	N/A	6	N/A	N/A

Table 7.3 Overall Evaluation Metrics

surpasses both in computational speed executing in approximately 5s. This makes Model 2 (equivalently Model 3 at $T_{PS} = 6$), the fastest and most effective approach that achieves the most optimal balance for students and lecturers alike.

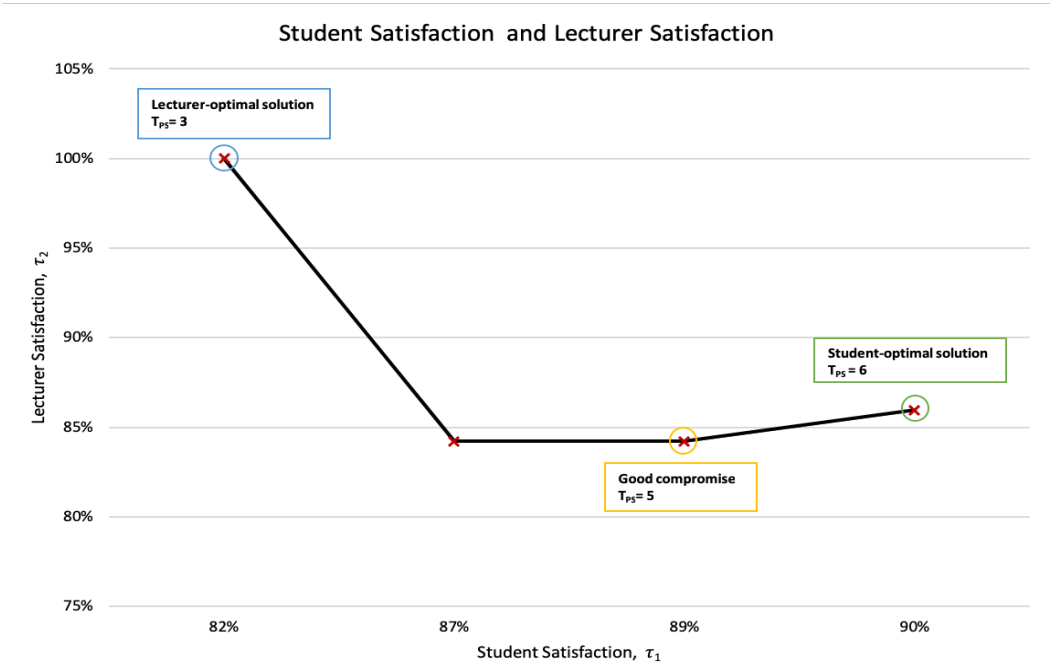
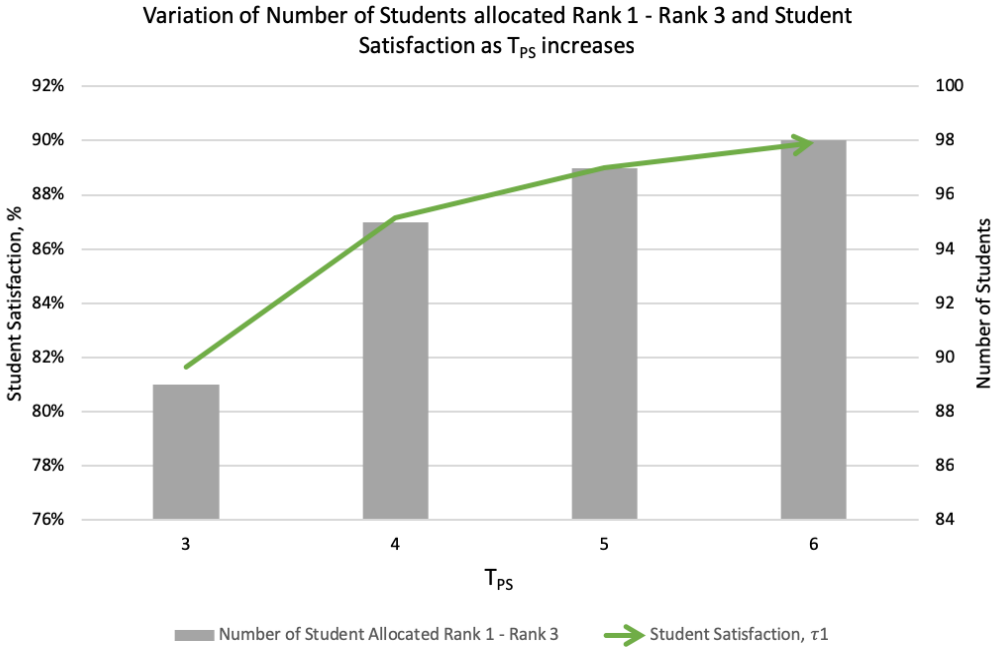
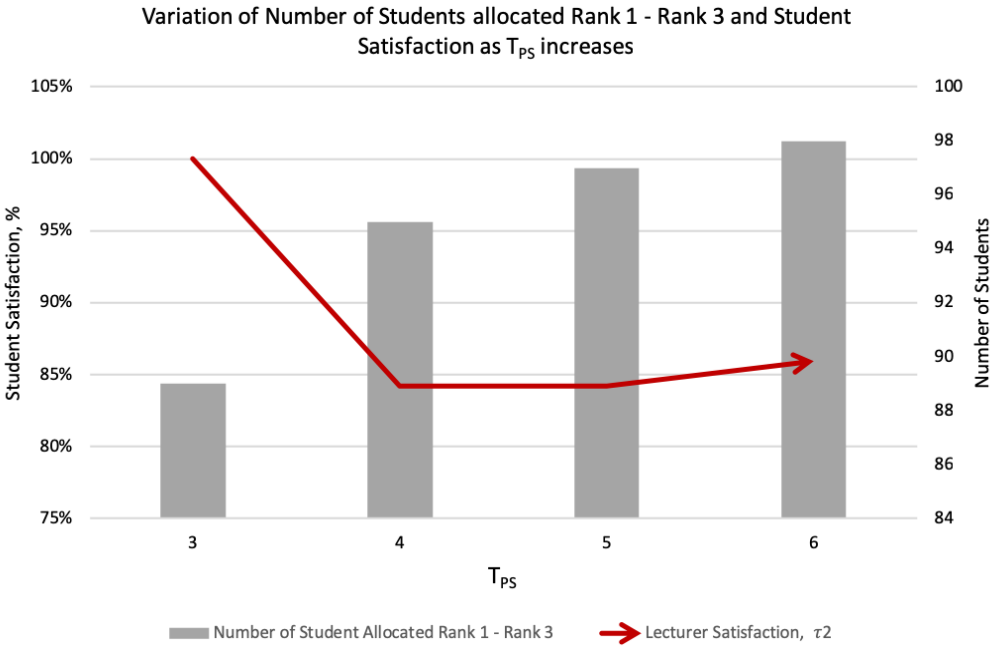


Fig. 7.7 Student Satisfaction and Lecturer Satisfaction Analysis for $3 \leq T_{PS} \leq 6$



(a) Student Satisfaction



(b) Lecturer Satisfaction

Fig. 7.6 Satisfaction scenarios as the number of students who get allocated their top three preferences increases

Chapter 8

Further Applications of the SPA Problem

This chapter presents the practicability and adaptability of the BIP approach examined in this paper on varied industry resource allocation problems faced in the real-world as discussed initially in Chapter 1.

As observed in this papers' introduction, there are numerous cases where the resource allocation challenge occurs. This chapter, however, investigates three particular resource allocation problems that occur in the transportation, health, and corporate sectors. We will discuss the *Vehicle Routing* problem specifically regarding taxi services in urban cities, the well-known *Hospital/Residents* problem which was derived from Gale and Shapley's *College Admissions* problem [9], and the *Job/Worker* problem that exists ubiquitously across industries.

8.1 Vehicle Routing

The VR problem on taxi allocation is adapted from BIP SPA Model 2 seen in Chapter 3
This adaptation requires no modification to Model 2; the mathematical formulations are directly applied to the case of vehicle routing unchanged.

Taxi services are a popular mode of transport especially in large cities offering comfort, convenience, and speed of travel compared to public transportation [52]. However, public transport is constantly in use as it is designed to cater to all whereas taxi cabs suffer low capacity rates where up to 50% of the time the cabs are empty [53]. The low capacity rates are attributed to factors that affect taxis such as the weather - where rain motivates more customers to use a taxis [53], and high tourist seasons - where tourists tend to be unaware of how to navigate public transport systems in new countries and rely on taxis. These factors

create uncertainty within the taxi firm reservation systems that consequently cause low taxi capacity rates. To mitigate this, Souza et al. noted that in Belo Horizonte, a city in Brazil, 10% - 15% of customers who prefer ordering a taxi by phone, gave up because of the lengthy wait for the service. [52]. These lengthy waiting times are attributed by the inefficiency of allocating cabs that are too far from the potential customer, forcing the customer to wait as the cab travels to the pick up point. Hence, an effective vehicle routing system is required to ensure cabs spend more time serving to customers rather than waiting (being unproductive). The use of GPS to correlate the distance between a taxi and a potential customer greatly enables the solution to this problem.

An approach to evaluate how to optimally assign taxi cabs to customers has been sought to be solved in several cities and businesses such as the city of Singapore [54] and Dazhong Company in Shanghai [55]. Yang and Wong noted that Singapore uses an automated centralized system that handles the taxi service: the Automatic Vehicle Location and Dispatch System (AVLDS). The system requires their drivers to send their positions, and as customers calls arrive, the system assigns the closest vehicle to each customer. This approach is emulated at Dazhong Company in Shanghai as documented by Xu et al., both scenarios resulted in a reduction of 16% - 32% on traveled distances without a passenger, and a reduction of 15 - 30 minutes in the customer average waiting time [52]. However, these problems although conventionally solved with LP have not been explored directly with a BIP approach.

Suppose that a taxi firm has T_T taxis available, and T_{Cu} customers wishing to be picked up as soon as possible. The time taken for the taxi to reach the pick up point establishes the cost of each taxi as taxi firms such as Uber, pride themselves on fast customer pick ups [56]. It is essential to note that the time taken for the taxi to reach a customer, directly corresponds to the Euclidean distance between the taxi and the customer. Therefore, minimizing the aforementioned time/distance optimizes the driver's experience, as they have less idle time whilst customers enjoy shorter pick-up times.

To that end, the 'cost' of each taxi picking up a particular customer will depend on the Euclidean distance between the taxi and the customer. Souza et al. noted that the distance between the taxi and the customer can be treated in a more elaborate way, bt taking into account the pre-determined route of the taxi, speed limits on the route, and traffic among others factors [52]. The optimal solution to the assignment problem will be the combination of taxis and customers that result in the least total cost [57] i.e the shortest Euclidean distance travelled in total. As such, the decision of whether or not a driver picks up a customer is dependent on how close the driver is to the potential customer, and the Euclidean distance

between them is used to prioritize picks ups. The shorter the distance, the less time it takes for the taxi to pick up the customer thus reducing the 'cost' for taxi j to pick up customer i.

The SPA problem explored in this paper has students rank projects with their most preferred project as 'Rank 1', and incrementing their ranking such that their least preferred project is 'Rank 10'. In this scenario, the Euclidean distances between a particular taxi and potential customers are calculated (8.1) whereby the shortest distance is the most preferred 'customer pick up' equivalently weighted as 'Rank 1' and the longest distance is the least preferred 'customer pick up' equivalently weighted as 'Rank 10'; if the taxi's algorithm is only allowed to calculate and shortlist the 10 closest potential customers. Note that 0s are set to the Euclidean distance calculations of potential customers not shortlisted i.e too far way.

Note that the SPA formulation of Model 2 is an assignment problem following a BIP approach, which is directly adapted to solve the Taxi Firm problem faced in various businesses globally.

Step 1 - Problem Description:
Modeling the Taxi Firm problem as an assignment problem that considers the shortest distance between a taxi and a potential customer as the most preferred taxi assignment can be adapted from the formulation of Model 2 explored earlier. In this case, the taxi algorithm will assign a weight $C_{i,j}$, of customer i on taxi j that ranges such that $C_{i,j} \in [1, T_C]$ where T_C represented the total number of distances required to be ranked by each taxi. For example, if $T_C = 5$, a taxi would assign a weight of 5 to their least preferable 'customer pick up', a weight of 1 to their most preferable 'customer pick up' and 0s to customers they're not interested in picking up due to lengthy distances. The Taxi Firm problem is formulated as a **minimization** problem to favor lower rankings i.e allocating customers closest to the taxi, therefore the most preferred 'customer pick ups' using integer programming.

Variable Definitions:

- T_T is the total number of taxis.
- T_{Cu} is the total number of customers.
- T_C is the total number of distances required to be ranked by each taxi between a particular taxi and potential customers.
- $C_{i,j}$ is the cost/distance matrix that represents the Euclidean distances between taxis j and customers i. The Euclidean distances are calculated using the GPS coordinates of the taxi (x_j, y_j), and the customer (x_i, y_i):

$$C_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \quad \forall\ i, j \tag{8.1}$$

Step 2 - Mathematical Program Formulation:

- *Decision Variables*
  - $X_{i,j}$: Binary variable that determines whether customer i is allocated to taxi j. This decision can be defined as:

$$X_{i,j} = \begin{cases} 1, & \text{if customer } i \text{ is allocated for pick up with taxi } j \\ 0, & \text{otherwise} \end{cases} \tag{8.2}$$

  - $C_{i,j}$: Is the ranking assigned to customers i by taxis j. If taxi j Euclidean distance calculation did not shortlist customer i as a possible pick up preference, then $C_{i,j} = 0$. If taxi j shortlists customer i as preference 1, then $C_{i,j} = 1$ highlight that this customer is the closest to taxi j as it has the shortest Euclidean distance. If taxi j shortlists customer i as preference 2, then $C_{i,j} = 2$ and so on. Hence, $C_{i,j}$ can take any value where, $C_{i,j} \in [1, T_C]$. Recall, T_C is the total number of Euclidean distances required to be ranked by each taxi between that particular taxi and potential customers. Hence, if $T_C = 3$:

$$C_{i,j} = \begin{cases} 1, & \text{if customer } i \text{ ranked } 1^{st} \text{ preference pick up by taxi } j \\ 2, & \text{if customer } i \text{ ranked } 2^{nd} \text{ preference pick up by taxi } j \\ 3, & \text{if customer } i \text{ ranked } 3^{rd} \text{ preference pick up by taxi } j \end{cases} \tag{8.3}$$

- *Constraints*

1. **Each taxi should only be allocated to one customer**
   Remains unchanged from Model 2. However, note that one customer may refer to one set of customers i.e a couple, a group of four people etc.

$$\sum_{i=1}^{T_{Cu}} X_{i,j} = 1, \text{ where, } 1 \leq j \leq T_T \tag{8.4}$$

2. **Every customer should only be allocated to at-most-one taxi**
   Remains unchanged from Model 2

$$\sum_{j=1}^{T_T} X_{i,j} \leq 1, \text{ where, } 1 \leq i \leq T_{Cu} \tag{8.5}$$

3. **Each taxi can only be allocated a customer that is shortlisted in the Euclidean distance preferences**
   Remains unchanged from Model 2

$$X_{i,j} \leq C_{i,j} \tag{8.6}$$

   Because when taxi j does not take customer i as a preference, $C_{i,j} = 0$ and hence $X_{i,j} = 0$ i.e taxi j is not assigned to customer i.

- *Objective Function*
  The objective is to give as many taxis as high a ranked customer on their Euclidean distance preference list. This translates to assigning as many taxis to their closest customer i.e shortest time/distance to customer pick up. Hence, minimizing the passenger waiting time and the taxis' idle time.

$$\text{Minimize } Z_{VR} = \sum_{j=1}^{T_T} \sum_{i=1}^{T_{Cu}} (C_{i,j} \times X_{i,j}) \tag{8.7}$$

8.2 The Hospital/Residents Problem

The HR problem is adapted from BIP SPA Model 2 seen in Chapter 3
This adaptation requires a slight modification to Model 2, as an extra constraint (8.11) is added to the initial mathematical formulation to accommodate the case of the Hospital/Residents problem.

An instance of the Hospitals / Residents problem (HR) involves a set of residents i.e junior medics, and a set of hospitals where each hospital has a given resident quota. In this two-sided matching problem, the residents have preferences for the hospitals, as do hospitals for residents [58]. The solution to the HR problem is a stable matching between residents and hospitals with respect to the capacity constraints and preference lists in a precise way [59]. As such, the largest implementation of the HR problem is the 'National Resident Matching Program' (NRMP) [60] in the US which is a centralized matching scheme that aims to optimally match residents to hospitals across the country. This matching scheme handles the allocation of approximately 30,000 residents graduating from medical schools to the hospital

of their choice. The NRMP employs an efficient algorithm that solves the Hospitals/Residents problem finding a stable matching of residents to hospitals that is resident-optimal [61]. This means, each resident obtains the best hospital that he/she could obtain in any stable matching [20].

Note that the SPA formulation in Model 2 is an assignment problem that is student-optimal by design whereby only the student has preferences over the projects/lectures. Hence, translating the resident-optimal matching as discussed above into the BIP approach seen in this paper, is easily adaptable.

Step 1 - Problem Description:
Modeling the HR problem as an assignment problem that is resident-optimal i.e only considering the resident's ranking of preferred hospitals can be achieved with a modification of the Model 2 SPA BIP optimization model explored earlier.

In the case of the HR problem, residents i will assign a weight $C_{i,j}$, on hospital j that ranges such that $C_{i,j} \in [1, T_C]$ where T_C represented the total number of hospital preferences allowed to be ranked by each resident. For example, if $T_C = 5$, residents would assign a weight of 5 to their least preferable hospital and a weight of 1 to their most preferable hospital and 0s to hospitals they're not interested in working in. The HR problem is formulated as a **minimization** problem to favor lower rankings i.e allocating residents to their most preferred hospitals using integer programming.

Variable Definitions:

- T_R is the total number of residents.
- T_H is the total number of hospitals.
- T_C is the total number of hospitals required to be ranked by each resident.
- $C_{i,j}$ is the cost/preference matrix that represents the ranking of hospital j by resident i.

Step 2 - Mathematical Program Formulation:

- *Decision Variables*
  - $X_{i,j}$: Binary variable that determines whether resident i is allocated to hospital j. This decision can be defined as:

$$X_{i,j} = \begin{cases} 1, & \text{if resident } i \text{ is allocated to hospital } j \\ 0, & \text{otherwise} \end{cases} \tag{8.8}$$

– $C_{i,j}$: Is the ranking assigned to hospital j by resident i. If resident i did not list hospital j as a possible preference then $C_{i,j} = 0$. Hence, if $T_C = 3$:

$$C_{i,j} = \begin{cases} 1, & \text{if resident } i \text{ ranks hospital } j \text{ as } 1^{st} \text{ preference} \\ 2, & \text{if resident } i \text{ ranks hospital } j \text{ as } 2^{nd} \text{ preference} \\ 3, & \text{if resident } i \text{ ranks hospital } j \text{ as } 3^{rd} \text{ preference} \end{cases} \tag{8.9}$$

– χ_j: A hospital can only be allocated a number of times. That is to say, multiple residents i can be allocated to a single hospital j depending on its resident-intake quota as discussed by Askalidis et al. [62]. This differs on a hospital by hospital basis depending on hospital size, location, number of available positions and other factors.

- *Constraints*

1. **Each resident should only be allocated to one hospital**
   Remains unchanged from Model 2

$$\sum_{j=1}^{T_H} X_{i,j} = 1, \text{ where, } 1 \leq i \leq T_R \tag{8.10}$$

2. **Every hospital can only be allocated to a given number of residents**
   This is added to Model 2's initial formulation to take into account a hospitals' resident-intake quota, χ_j.

$$\sum_{i=1}^{T_R} X_{i,j} \leq \chi_j, \text{ where, } 1 \leq j \leq T_H \tag{8.11}$$

3. **Each resident can only be allocated a hospital that is part of their preferences**
   Remains unchanged from Model 2

$$X_{i,j} \leq C_{i,j} \tag{8.12}$$

   Because when resident i does not take hospital j as a preference, $C_{i,j} = 0$ and hence $X_{i,j} = 0$ i.e resident i is not assigned to hospital j.

- *Objective Function*

The objective is to give as many residents as high a ranked hospital on their preference

list whilst ensuring that each hospitals' resident-intake quota is not exceeded.

$$\text{Minimize } Z_{HR} = \sum_{i=1}^{T_R} \sum_{j=1}^{T_H} (C_{i,j} \times X_{i,j}) \tag{8.13}$$

8.3 Job/Worker Problem

The J/W problem is adapted from BIP SPA Model 3 seen in Chapter 3
This adaptation requires a slight modification to Model 3, as an extra constraint (8.19) is added to the initial mathematical formulation to accommodate the case of the job/worker problem.

The most common scenario to apply distinct assignment problem approaches is the job/worker scenario. Given that the SPA problem in this paper is formulated as a specific variation of the assignment problem, we can apply the BIP SPA formulations to this problem as well. It is important to note the Job/Worker problem is applied ubiquitously throughout various industries such as manufacturing, production, corporate sector and more. Jiang and Aftab et al. delve into the Job/Worker problem in the manufacturing industry known as the 'Assembly Job Shop Scheduling Problem' (AJSSP) whereby the processing time of each job in the assembly depends on its starting time and the number of workers assigned to process it [63]. The objective is to find the optimal scheduling and worker assignment to assembly machines in order to minimize the production cost [64]. Whilst Jiang solves the AJSSP problem using greedy heuristics, Aftab et al. use an 'Ant Colony Optimization' approach.

In the production industry, Chauvet et al. explored the Job/Worker problem applied in the 'The National Company for Industry and Mining of Mauritania' with n employees available to perform m maintenance tasks whereby $n > m$. The employees at this company are not specialized to perform particular tasks. Hence, due to their varied background, the times that two employees require to perform the same task might be quite different [65]. The objective of the company is to assign employees to at-most-one task each while minimizing the time to complete all the tasks. Finally, Caron et al. investigated the application of the Job/Worker problem in the corporate sector whereby employees belong to given seniority classes, and jobs have given priority levels. In this case, the seniority and priority constraints imposed that the solution be such that no unassigned person can be given a job unless an assigned person with the same or higher seniority becomes unassigned [66]. The objective is to assign employees to at-most-one task each while minimizing the time to complete all the tasks using a mixed linear programming (MLP) approach. Note that neither of the Job/Worker applications above utilize a BIP approach. Hence, this paper will similarly explore the

corporate applications of the Job/Worker problem, as did Caron et al. however, rather than use greedy heuristics, 'Ant Colony Optimization', or a MLP approach, a BIP approach will be implemented.

The corporate setting is outlined as follows, an office has T_W workers and T_J jobs to be performed whereby $T_W > T_J$. As Chauvet et al. realized, workers differ in their efficiency of accomplishing tasks dependent on their prior experience and background. In this case, the cost matrix would relate to the 'time each worker takes to accomplish each job'. The objective is to assign jobs to workers such that the overall time spent completing jobs is minimized. This inherently minimizes the monetary costs to the company. However, in a part-time setting such as working at a restaurant, the cost matrix would relate to the 'pay each worker requires to accomplish each job' rather than the efficiency of the workers. A part-time worker w_1, may require payment of £x_1 to accomplish a certain job in a restaurant whereby another part-time worker w_2, may require payment of £x_2 to accomplish the same job where, $x_1 < x_2$. Given the objective is to assign jobs to workers resulting in the lowest cost to the restaurant, in this case, part-time worker w_1 would be allocated to the job.

Note that the SPA formulation in Model 3 is an assignment problem following a BIP approach that can be directly adapted to solve the Job/Worker problem in various scenarios, and currently in the corporate setting.

Step 1 - Problem Description:
Modeling the Job/Worker problem as an assignment problem that considers the time each worker takes to complete a certain job can be achieved with a modification of the Model 3 SPA BIP optimization model explored earlier. The time a worker takes to accomplish a certain job is effectively the efficiency of worker i in completing job j. In the SPA model, the values of cost matrix, $C_{i,j}$ are filled with project rankings filled by student i depicting their preferences, however, in the Job/Worker problem, for company efficiency, the worker does not input their preferences and instead the company often decides how to assign workers. This ensures that the company can assign workers to jobs such that their cost and time wastage is minimized. Note that practically, the decision of assigning workers to a job can be treated more elaborately taking factors such as availability, division preferences, and seniority into account.

The 'cost matrix' in the Job/Worker scenario, is not filled by the worker, but by the employer, which differentiates it slightly from the SPA problem. Practically, each manager is responsible for a number of jobs. However, whilst allocating workers to jobs, it is vital to not over-allocate workers to a particular job thereby over-burdening the manager responsible. Depending on

the managers' role and seniority, each manager will be expected to manage a certain number of jobs, and inherently a number of workers simultaneously, thus this constraint will be taken into account. The Job/Worker problem is formulated as a **minimization** problem to favor low-cost/time i.e allocating the most efficient worker for a particular job using integer programming.

Variable Definitions:

- T_J is the total number of jobs.
- T_W is the total number of workers.
- T_M is the total number of managers.
- T_{PS} is the maximum limit of the total number of workers each manager can supervise.
- T_C is the total number of job times required to be input by the employer for each worker.
- $C_{i,j}$ is the cost/efficiency matrix that is filled by the employer and represents the time worker i takes to accomplish job j.

Step 2 - Mathematical Program Formulation:

- *Decision variables*
  - $X_{i,j}$: Binary variable that determines whether worker i is allocated to job j. This decision can be defined as:

$$X_{i,j} = \begin{cases} 1, & \text{if worker } i \text{ is allocated to job } j \\ 0, & \text{otherwise} \end{cases} \tag{8.14}$$

  - $C_{i,j}$: Is an integer variable indicating the efficiency assigned to worker i by the employer with respect to job j. If the employer i did not set the time/efficiency of worker i on job j, indicating they would not like to consider worker i for that particular job, then $C_{i,j} = 0$. If the employer assigns $C_{i,j} = 1$, this indicates that the time worker i takes to accomplish job j is the shortest/most efficient. If the employer assigns $C_{i,j} = 2$, this indicates that the time worker i takes to accomplish job j is the second shortest/efficient and so on. Hence, $C_{i,j}$ can take any value where, $C_{i,j} \in [1, T_C]$. Recall, T_C is the total number of job completion times required to be input by the employer for each worker.

- $P_{k,j}$: Binary variable that determines whether manager k proposed job j. This decision can be defined as:

$$P_{k,j} = \begin{cases} 1, & \text{if manager } k \text{ is responsible for job } j \\ 0, & \text{otherwise} \end{cases} \tag{8.15}$$

- $S_{k,i}$: Binary variable that determines whether manager k will supervise worker i. This decision can be defined as:

$$S_{k,i} = \begin{cases} 1, & \text{if manager } k \text{ will supervise worker } i \\ 0, & \text{otherwise} \end{cases} \tag{8.16}$$

  Where,

$$S_{k,i} = P_{k,j} * X_{i,j}^{\mathsf{T}} \tag{8.17}$$

- χ_j: A job can only be allocated a number of times. That is to say, multiple workers i, can be allocated to a single job j depending on its worker-intake capacity. This differs on a job by job basis depending on job size, location, number of available positions and other factors.
- β_k : Represents the number of workers manager k is allowed to supervise at a time i.e the worker supervision workload for manager k

• *Constraints*

1. **Each worker should only be allocated to one job**
   Remains unchanged from Model 3

$$\sum_{j=1}^{T_J} X_{i,j} = 1, \text{ where, } 1 \leq i \leq T_W \tag{8.18}$$

2. **Every job can only be allocated to a given number of workers**
   This is added to Model 3's initial formulation to take into account a jobs'worker-intake capacity, χ_j.

$$\sum_{i=1}^{T_W} X_{i,j} \leq \chi_j, \text{ where, } 1 \leq j \leq T_J \tag{8.19}$$

3. **Each worker can only be allocated a job that has been listed part of the employer's preferences to ensure time-efficient assignments**
   Remains unchanged from Model 3

$$X_{i,j} \leq C_{i,j} \tag{8.20}$$

4. **Each manager can only supervise a given number of workers, β_k**
   Remains unchanged from Model 3
   This constraint requires that each manager does not supervise more than β_k number of workers simultaneously. Practically, this is a key constraint as managers who are over-extended tend to perform poorly. It is important to appreciate that not all managers will have the same job supervision workload depending on factors such as seniority, experience, and team size, etc. However, it is reasonable that managers within the same seniority level will have the same job supervision workload. For example, all vice-presidents would have a job supervision workload x, whereas all directors would have job supervision workload y where $x < y$. The constraint below will assume there's a unified allocation system that takes an input of all the managers irrespective of their seniority level. Hence, the job supervision workload will not be a constant T_{PS}, it will be a variable β_k that varies depending on the manager k and their respective seniority within the firm.

   This can be achieved by:

$$\sum_{j=1}^{T_J} P_{k,j} \times \sum_{i=1}^{T_W} X_{i,j}^{\mathsf{T}} \leq \beta_k, \text{ where, } 1 \leq k \leq T_M \tag{8.21}$$

   Or equivalently:

$$\sum_{i=1}^{T_W} S_{k,i} \leq \beta_k, \text{ where, } 1 \leq k \leq T_M \tag{8.22}$$

   Note:

$$\begin{aligned} &\sum_{i=1}^{T_W} S_{k,i} \equiv \beta_k \\ &S_k \leq \beta_k, \text{ where, } 1 \leq k \leq T_M \end{aligned} \tag{8.23}$$

- *Objective Function*

  The objective is to assign as many workers to their most efficient jobs i.e the job that takes the shortest time to accomplish, in order to minimize the overall time spent and cost to the company in completing business tasks. This Job/Worker optimization model also ensures each manager has an acceptable job supervision workload that is not exceeded.

$$\text{Minimize } Z_{JW} = \sum_{i=1}^{T_W} \sum_{j=1}^{T_J} (C_{i,j} \times X_{i,j}) \tag{8.24}$$

Chapter 9

Further Work

This section outlines the lessons learned, and how the work explored in this paper could be taken further given more time. It depicts distinct areas of research that build upon the successes of the BIP approach in solving resource allocation problems achieved so far in this paper.

There are some critical lessons to be aware of prior to advancing to future work. Firstly, it is vital to decide how to formulate a resource allocation problem prior to any modelling. This is because a resource allocation problem can be formulated in various ways, such as an assignment problem, transportation problem, matching problem among other variants. This initial decision sets the fundamental modeling course that is followed throughout the optimization methodology as shown in Figure 2.4. Additionally, during the mathematical formulations of an optimization model, it is advantageous to initially formulate the model such that it meets the general requirements, and later add constraints to ensure all edge cases are accounted for. Furthermore, it is essential to test a complex optimization model, prior to implementation to verify its functionality. This is done effectively with a simple and concise scenario whose outcomes/outputs are known, rather than using complex data whose outcomes are unknown for model validation.

Secondly, this paper outlined that although optimality is achievable, no solution can be optimal to every person/variable involved. As such, where each feasible solution yields a general optimal allocation to the problem, it does not guarantee that every student and lecturer in the SPA scenario would be satisfied. As such, creating evaluation metrics and analytical variables to measure an optimization models' performance w.r.t to its objective function, was vital in this paper. The establishment of variables such as student satisfaction, lecturer satisfaction, among others played a key role in interpreting the performance and impact of the SPA models formulated. This allowed for a deeper understanding that enabled

certain modifications to increase lecturer satisfaction within the lecturer population or strike a greater balance between the students and lecturers. Hence, whist delving in further work, it is imperative to be aware that there is no 'ideal' solution that is perfect for everyone involved. The goal, and challenge is to strive for a solution that is the most optimal for all variables/people involved. This optimality should yield a general consensus of satisfied students and lecturers.

With those lessons in mind, further work may expand the scope from the specificity of the SPA problem, to general resource allocation problems faced in various industries discussed briefly in Chapter 8. This papers' BIP approach can be applied more effectively in resource allocation problems such as network routing, computer resource allocation, load balancing or power distribution in the domain of EEE. This would exploit the success of BIP in execution speed, and solution optimality compared to existing methods as proved by this paper. Note that since the assignment problem is a special case of the transportation problem, this trajectory also widens the scope from an assignment problem to a transportation problem.

On the other hand, an extension to this project would be to narrow, rather than widen the scope on the SPA problem. Whilst narrowing the scope, the focal point would shift to a group SPA problem, rather than an individual SPA problem as currently explored in this paper. The group SPA problem is entirely different as it requires consideration of more variables, and constraints. This increase in constraint, and variable complexity would demand a **multi-objective**, **goal programming** and/or **dynamic programming** approach. Group projects are challenging to model as they create a different set of constraints that are not considered in the individual case of the SPA problem. Such constraints are:

- Maximum and minimum number of students that can be allocated to a group.
- Module prerequisites or knowledge requirements needed to implement the group project as observed for 3rd year EEE/EIE students in the EEE department at Imperial College London.
- Student cross-departmental ratios within a group to enable a multi-disciplinary project requiring distinct skills across courses.
- Certain group dynamic diversity constraints that may be required for allocation such as: Student gender ratio, cultural ratio etc.

Furthermore, the group project allocation problem is one faced at several universities throughout all education years, as opposed to the individual SPA problem that is encountered only in the final year of studies.

Chapter 10

Conclusion

This paper delved into solving a resource allocation problem using BIP, an approach not often seen in research as discussed in Chapter 2, particularly in section 2.1. This project was successful in showing that an integer programming approach yields an optimal feasible solution to the SPA problem within 5s for Model 2, and 50s for Model 3, which is much faster than existing methods. Furthermore, these BIP solutions result in allocations that have higher student and lecturer satisfaction rates within this record time; as compared to existing methods that perform less efficiently and much slower.

Prior to implementation, the limits of Python's optimization library PuLP, were not explored. Hence, some learning points were encountered in using PuLP for the BIP implementation. At specific junctures, the exploration of implementing dual objective functions arose in order to allocate students their highest preference, whilst minimizing the number of projects lecturers have to supervise. This multi-objective function decision was initially enabled as integer programming supports multiple objective functions [22][31]. However, since PuLP does not have the capability to declare multiple objective functions, a dual objective approach or multiple objective approach could not be implemented in Python.

However, PuLP's limitation did not hinder the implementation of the SPA problem, as the problem required objective functions that could be formulated *either* as a single objective function, or as a multi-objective function to achieve the required result, and purpose of this paper. Nevertheless, more complex problems within the resource allocation scope exist, and may depend upon multi-objective functions. In this case, it is vital to acknowledge that the Python PuLP implementation in this paper would not have been suffice. Hence, the use of Gurobi not as a solver, but as a linear programming framework using gurobipy would meet the challenge, as it provides tools that blend multiple objectives hierarchically.

From the results and evaluation process, the best approach was the BIP SPA model formulated in this paper (Model 2), as it was the fastest implementation offering the most optimal balance for lecturers and students alike in $\approx$ 5s. The speed of execution is invaluable, as it enables fast re-computation of distinct optimal allocations using different student and lecturer preferences when new information needs to be considered by the optimization model. As such, Model 2 is the recommended optimization model to effectively solve the SPA problem. In addition, the results in Table 7.3 have demonstrated that the other BIP SPA implementations formulated in this paper, perform generally much faster whilst producing much higher student and lecturer satisfaction rates than the current departmental implementation, and existing methods. Hence, these BIP SPA implementations are a great improvement on the aforementioned methods, and would be the preferred implementation in comparison nevertheless. Furthermore, implementing Model 3 that takes approximately 50s to execute is a successful precedent. As it presents the efficiency of the BIP models in solving SPA problems with more complex objective functions as detailed in Chapter 3 with Model 4, 5, and 6.

In conclusion, the use of BIP has resulted in a simple, yet effective manner of solving the SPA problem that has proven to be faster than conventional methods seen before. The models were implemented using Python in a manner that allows them to utilize distinct commercial solvers such as Gurobi, CPLEX or GLPK, by a single-line modification in the code. This ability to call on different commercial solvers enables this papers' implementation to be adapted on a large scale, solving the SPA problem for $\approx 10^4$ students as opposed to the $\approx$ 100 students solved as a proof of concept in this paper. Additionally, the ease of implementation, requiring less than 100 lines of code facilitates fast accommodation of this approach in solving the SPA problem across departments and universities alike. Regarding execution time, the fastest BIP model formulated in this paper (Model 2) yields the most optimal student and lecturer allocation in $\approx$ 5s allowing for distinct data-sets to be run quickly reducing delay between possible allocation results. Furthermore, the mathematical formulations not implemented in Chapter 3 i.e models 4-6 display pragmatic use-cases that can be used in varied circumstances in allocating final year project to students. Lastly, Chapter 8 has successfully shown the versatility of the BIP approach achieved by simply adapting the SPA formulations in Chapter 3 in vital real-world applications.

Appendix A

Appendix

All code programmed throughout this project is available in the Github Repository.

A.1 Model 2

All functions being called are written in M2_functions.py that can be found in the repository above.

```
"""
Model 2 : Allocate projects to students based on their preference rankings
Author: Mwanakombo Hussein
"""

from pulp import *
import M2_functions as fnc
import numpy as np
import time

#Retrieving data from excel : Input workbook name, worksheet name
C = fnc.read_preferences('M2_InputData.xls','Realistic')

#Data : 109 students, 10 choices, 181 project preferences
number_of_students, number_of_projects = C.shape
print("Students: ", number_of_students, "Projects:", number_of_projects)

# Time it takes to run M2 start point
start = time.time()
```

```
20
21 # Create the 'prob' variable to contain the problem data
22 prob = LpProblem("SPA_M2", LpMinimize)
23
24 # A dictionary called 'x' is created to contain the referenced Variables
25 # Var x_{i,j} has a LB of # of students, UP of # of projects and is of type
       'LpBinary'
26 x = LpVariable.dicts("x", itertools.product(range(number_of_students),
       range(number_of_projects)),
27                       cat=LpBinary)
28
29 #Objective Function
30 objective_function = 0
31 for student in range(number_of_students):
32     for project in range(number_of_projects):
33         if C[(student, project)] > 0:
34             objective_function += x[(student, project)] * ((C[(student,
                   project)]))
35 prob += objective_function
36
37 #Constraints
38 #1 Each student should be allocated to only 1 project
39 for student in range(number_of_students):
40      prob += sum(x[(student, project)] for project in
           range(number_of_projects)) == 1 #, "Student_Constraint"
41
42 #2 Each project should be allocated to at most 1 student
43 for project in range(number_of_projects):
44     prob += sum(x[(student, project)] for student in
           range(number_of_students)) <= 1 #, "Project_Constraint"
45
46 #3 Each student can only be allocated a project that is part of their
       preferences subset
47 for student, project in x:
48     prob += x[student, project] <= float(C[student, project]) #,
           "Preference_Constraint"
49
50 # The problem is solved using PuLP's choice of solver : Gurobi
51 # prob.solve() to used PuLP's CBC Solver
```

```
# or
prob.solve(GUROBI())

# The status of the solution is printed to the screen
print("Status:", LpStatus[prob.status])

# Time it takes to run M2 end point
end = time.time()
print("Time elapsed:", end - start)

# Each of the variables is printed with it's resolved optimum value
# using function sort_allocation to fix lexicographical string error
allocation = np.zeros((number_of_students,number_of_projects))
fnc.sort_allocation(prob,allocation)

# The optimized objective function value is printed to the screen
print("Objective Fnc= ", value(prob.objective))

# Retrieve allocation rankings in array 'rank' for post-optimal analysis
rank = []
for student in range(number_of_students):
    for project in range(number_of_projects):
        if allocation[student, project] == 1:
            rank.append(C[student, project])

# Export allocation and ranks to xls workbook
# using write_allocation function in functions.py
fnc.write_allocation(allocation,rank)
```

A.2 Lexicographical Error Correcting

This snippet it taken from M2_functions.py which is also used for Model 3.

```
# Fixing the string lexicographical order to conventional integer
    (numerical) order
def sort_allocation(prob,unsorted_allocation):
    for v in prob.variables():
        #filter: from (0,_0) to 0,_0
```

```
        filtered = re.search('x_\((.+?)\)', v.name).group(1)
        #find the comma's index which is 1
        comma_idx = filtered.find(',_')
        #find the underscore's index
        underscore_idx = comma_idx + 1
        #set coordinates as integers from filtered to get int (0,0) from
            string 0,_0
        coords = int(filtered[:comma_idx]), int(filtered[comma_idx+2:])
        #create allocation array with new int coordinates and set their
            respective values
        unsorted_allocation[coords] = v.varValue
    #print(unsorted_allocation)
```

A.3 Model 3

All functions being called are written in M3_functions.py that can be found in the repository listed above.

```
"""
Model 3 : Allocate projects to students based on their preference rankings
whilst putting a upper bound (T_PS),on the number of projects each lecturer
    is allowed to supervise.
Author: Mwanakombo Hussein
"""

from pulp import *
import M3_functions as fnc
import numpy as np
import time

# Maximum number of projects each lecturer can supervise simultaneously
T_PS = 6
print("Superviser Limit:", T_PS)

#Retrieving Student Preferences data from excel : Input workbook name,
    worksheet name
C = fnc.read_preferences('M3_InputData.xls','ij_choice_realistic')
    #ij_choice_realistic or ij_choice_simple
```

```
# Data : 109 students, 181 projects, 10 project preferences
number_of_students, number_of_projects = C.shape
print("Students: ", number_of_students, "Projects:", number_of_projects)

# Retrieving Lecturer Project Proposals data from excel : Input workbook name, worksheet name
L = fnc.read_preferences('M3_InputData.xls','kj_realistic') #kj_realistic or kj_simple
# Data : 57 lecturers, 181 projects
number_of_lecturers, number_of_projects = L.shape
print("Lecturers: ", number_of_lecturers, "Projects:", number_of_projects)

# Time it takes to run M3 start point
start = time.time()

# Create the 'prob' variable to contain the problem data
prob = LpProblem("SPA Model_3", LpMinimize)

# A dictionary called 'x' is created to contain the referenced Variables
# Var x_{i,j} has a LB of # of students, UP of # of projects and is of type 'LpBinary'
x = LpVariable.dicts("x", itertools.product(range(number_of_students), range(number_of_projects)),
                        cat=LpBinary)
#OBJECTIVE FUNCTION
objective_function = 0
for student in range(number_of_students):
    for project in range(number_of_projects):
        if C[(student, project)] > 0:
            objective_function += x[(student, project)] * ((C[(student, project)]))
prob += objective_function

#CONSTRAINTS
#1 Each student can only be allocated a project that is part of their preferences subset
for student, project in x:
    prob += x[student, project] <= float(C[student, project]) #,
        "Preference_Constraint"
```

```
50
51 #2 Each student should be allocated to only 1 project
52 for student in range(number_of_students):
53     prob += sum(x[(student, project)] for project in
           range(number_of_projects)) == 1 #, "Student_Constraint"
54
55 #3 Each project should be allocated to at most 1 student
56 for project in range(number_of_projects):
57     prob += sum(x[(student, project)] for student in
           range(number_of_students)) <= 1 #, "Project_Constraint"
58
59 # l lecturers
60 #4 Each lecturer can only supervise T_PS projects/students (sensitivity on
       number of t_ps)
61 for lecturer in range(number_of_lecturers):
62     prob += sum(L[lecturer, project] * sum(x[student, project]
63                                            for student in
                                                range(number_of_students))
64                 for project in range(number_of_projects)) <= T_PS
65
66 # The problem data is written to an .lp file
67 prob.writeLP("SPA Model_3.lp")
68
69 # The problem is solved using PuLP's choice of solver : Gurobi
70 prob.solve(GUROBI())
71
72 # The optimised objective function value is printed to the screen
73 print("Objective Fnc= ", value(prob.objective))
74
75 # The status of the solution is printed to the screen
76 print("Status:", LpStatus[prob.status])
77
78 # Time it takes to run M3 end point
79 end = time.time()
80 print("Time elapsed:", end - start)
81
82 # Each of the variables is printed with it's resolved optimum value
83 # using function sort_allocation to fix lexicographical string error
84 allocation = np.zeros((number_of_students,number_of_projects))
```

```
85 fnc.sort_allocation(prob,allocation)
86
87
88 # Calculate and print the number of projects each lecturer is supervising
89 # Multiply Pkj * Xij resulting in Ski
90 Ski = np.dot(L,allocation.T)
91 # Sum the number of projects each lecturer is supervising
92 # across columns i.e across projects
93 lect_count = np.sum(Ski,axis=1)
94
95
96 # Retrieve allocation rankings in array 'rank' for post-optimal analysis
97 rank = []
98 # final_rank = fnc.rank(ranks)
99 for student in range(number_of_students):
100     for project in range(number_of_projects):
101         if allocation[student, project] == 1:
102             rank.append(C[student, project])
103
104 # Change lect_count from ndarray to list
105 # Export allocation,ranks and number of projects each lecturer is
      supervising (lect_count)
106 # to xls workbook
107 fnc.write_allocation(allocation,rank,lect_count.tolist())
```

References

[1] Behnam Esfahbod. Euler diagram on p, np, np-hard and np-complete. URL https://commons.wikimedia.org/w/index.php?curid=3532181. CC BY-SA 3.0.

[2] Riccardo Manzini and Elisa Gebennini. Optimization models for the dynamic facility location and allocation problem. *International Journal of Production Research*, 46 (8):2061–2086, Apr 15, 2008. doi: 10.1080/00207540600847418. URL http://www.tandfonline.com/doi/abs/10.1080/00207540600847418.

[3] Jean-François Houde, Peter Newberry, and Katja Seim. Economies of density in e-commerce: A study of amazon's fulfillment center network. Working Paper 23361, National Bureau of Economic Research, April 2017. URL http://www.nber.org/papers/w23361.

[4] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. A resource allocation model for qos management. pages 298–307. IEEE Comput. Soc. Press, 1997. ISBN 1052-8725. doi: 10.1109/REAL.1997.641291. URL https://ieeexplore.ieee.org/document/641291.

[5] Bin Wang, Li Chen, Xiaohang Chen, Xin Zhang, and Dacheng Yang. Resource allocation optimization for device-to-device communication underlaying cellular networks. pages 1–6. IEEE, May 2011. ISBN 1550-2252. doi: 10.1109/VETECS.2011.5956157. URL https://ieeexplore.ieee.org/document/5956157.

[6] Brian W. Unger. A computer resource allocation model with some measured and simulation results. *IEEE Transactions on Computers*, C-26(3):243–259, Mar 1977. doi: 10.1109/TC.1977.1674813. URL https://ieeexplore.ieee.org/document/1674813.

[7] Toshihide Ibaraki and Naoki Katoh. *Resource allocation problems*. MIT Pr, Cambridge, Mass, 1988. ISBN 9780262090278.

[8] Atila Abdulkadiroglu, Parag A. Pathak, and Alvin E. Roth. The new york city high school match. *The American Economic Review*, 95(2):364–367, 2005. doi: 10.1257/000282805774670167. URL http://www.fachportal-paedagogik.de/fis_bildung/suche/fis_set.html?FId=737704.

[9] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, Jan 1, 1962. doi: 10.1080/00029890.1962.11989827. URL http://www.tandfonline.com/doi/abs/10.1080/00029890.1962.11989827.

[10] Mourad Baiou and Michel Balinski. Student admissions and faculty recruitment. *Theoretical Computer Science*, 322(2):245–265, 2004. doi: 10.1016/j.tcs.2004.03.011. URL https://www.sciencedirect.com/science/article/pii/S0304397504001628.

[11] Michel Balinski and Tayfun Sönmez. A tale of two mechanisms: Student placement. *Journal of Economic Theory*, 84(1):73–94, Jan 1999. doi: 10.1006/jeth.1998.2469. URL https://www.sciencedirect.com/science/article/pii/S0022053198924693.

[12] M. Tounsi. A heuristic-based technique for university resource allocation problems. pages 1–6. IEEE, Mar 2006. doi: 10.1109/IEEEGCC.2006.5686243. URL https://ieeexplore.ieee.org/document/5686243.

[13] H. O. Salami and E. Y. Mamman. A genetic algorithm for allocating project supervisors to students. *International Journal of Intelligent Systems and Applications*, 8(10):51–9, 2016. doi: 10.5815/ijisa.2016.10.06. URL http://dx.doi.org/10.5815/ijisa.2016.10.06.

[14] David F. Manlove and Gregg O'Malley. Student-project allocation with preferences over projects. *Journal of Discrete Algorithms*, 6(4):553–560, 2008. doi: 10.1016/j.jda.2008.07.003. URL https://www.sciencedirect.com/science/article/pii/S1570866708000476.

[15] Gurobi optimization. URL https://www.gurobi.com.

[16] Cplex optimizer. URL https://www.ibm.com/analytics/cplex-optimizer.

[17] Glpk (gnu linear programming kit). URL https://www.gnu.org/software/glpk/.

[18] Cbc (coin-or branch and cut). URL https://projects.coin-or.org/Cbc.

[19] Alvin E. Roth. The evolution of the labor market for medical interns and residents: A case study in game theory. *Journal of Political Economy*, 92(6):991–1016, December 1, 1984. doi: 10.1086/261272. URL https://www.journals.uchicago.edu/doi/10.1086/261272.

[20] David F. Manlove and Frances Cooper. Two algorithms for the student-project allocation problem. *Journal of Discrete Algorithms*, 5(1):73–90, 2007. doi: 10.1016/j.jda.2006.03.006. URL https://www.sciencedirect.com/science/article/pii/S1570866706000207.

[21] Kolos Csaba Ágoston, Péter Biró, and Richárd Szántó. Stable project allocation under distributional constraints. *Operations Research Perspectives*, 5:59–68, 2018. doi: 10.1016/j.orp.2018.01.003. URL https://www.sciencedirect.com/science/article/pii/S2214716018300022.

[22] A. A. Anwar and A. S. Bahaj. Student project allocation using integer programming. *IEEE Transactions on Education*, 46(3):359–67, 2003. doi: 10.1109/TE.2003.811038. URL http://dx.doi.org/10.1109/TE.2003.811038.

[23] P.R. Harper, V. de Senna, I.T. Vieira, and A.K. Shahani. A genetic algorithm for the project assignment problem. *Computers and Operations Research,*, pages 1255–1265, 2005.

[24] A. Kwanashie, R.W. Irving, D.F. Manlove, and C.T.S. Sng. Profile-based optimal matchings in the student–project allocation problem. *In Proceedings of IWOCA '14: the 25th International Workshop on Combinatorial Algorithms, volume 8986 of Lecture Notes in Computer Science*, pages 213–225, 2015.

[25] D. Kazakov. Co-ordination of student-project allocation, 2001. URL http://www-users.cs.york.ac.uk/kazakov/papers/proj.pdf. Manuscript, University of York, Department of Computer Science.

[26] D.J. Abraham, R.W. Irving, and D.F. Manlove. Two algorithms for the student- project allocation problem. *Journal of Discrete Algorithms,*, pages 79–91, 2007.

[27] David Manlove, Duncan Milne, and Sofiat Olaosebikan. An integer programming approach to the student-project allocation problem with preferences over projects. In *5th International Symposium on Combinatorial Optimization, ISCO 2018, April 11, 2018 - April 13*, volume 10856 LNCS, pages 313–325, Marrakesh, Morocco, 2018 2018. School of Computing Science, University of Glasgow, Glasgow, United Kingdom, Springer Verlag. ISBN 0302-9743. doi: 10.1007/978-3-319-96151-4_27. URL http://dx.doi.org/10.1007/978-3-319-96151-4_27. Compilation and indexing terms, Copyright 2018 Elsevier Inc.; T3: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics);.

[28] Raúl Calvo-Serrano, Gonzalo Guillén-Gosálbez, Simon Kohn, and Andrew Masters. Mathematical programming approach for optimally allocating students' projects to academics in large cohorts. *Education for Chemical Engineers*, 20:11–21, Jul 2017. doi: 10.1016/j.ece.2017.06.002. URL https://www.sciencedirect.com/science/article/pii/S1749772817300301.

[29] Samuel Lightfoot. Solving the student project allocation problem. Technical report, May 6 2016.

[30] Jonathan Dye. A constraint logic programming approach to the stable marriage problem and its application to student project allocation, 29 June 2001.

[31] Li Pan Sydney C. K. Chu Guangyue Han and Joshua Z. Huang. Multi-criteria student project allocation: A case study of goal programming formulation with dss implementation, 2009.

[32] K. Iwama, S. Miyazaki, and H. Yanagisawa. Improved approximation bounds for the student-project allocation problem with preferences over projects. *Journal of Discrete Algorithms,13*, pages 59–66, 2012.

[33] Jon Lee, Giovanni Rinaldi, and Ali Ridha Mahjoub. *Combinatorial optimization*, volume 10856. Springer International Publishing, Cham, 2018. ISBN 9783319961514. URL http://bvbr.bib-bvb.de:8991/F?func=service&doc_library=BVB01&local_base=BVB01&doc_number=030516382&sequence=000001&line_number=0001&func_code=DB_RECORDS&service_type=MEDIA.

[34] Cihan Aksop and M. Akif Bakir. A 0-1 integer programming approach to a university timetabling problem. *Hacettepe Journal of Mathematics and Statistics*, 37(1):41–55, 2008. URL http://uvt.ulakbim.gov.tr/uvt/index.php?cwid=9&vtadi=TMUH&c=ebsco&c=summon&c=ebsco&ano=84904_12d3e6baeb2ffbeeb25ac2bfb0550f6f.

[35] Arabinda Tripathy. School timetabling–a case in large binary integer linear programming. *Management Science*, 30(12):1473–1489, Dec 1, 1984. doi: 10.1287/mnsc.30.12.1473. URL http://mansci.journal.informs.org/cgi/content/abstract/30/12/1473.

[36] Antony E. Phillips, Hamish Waterer, Matthias Ehrgott, and David M. Ryan. Integer programming methods for large-scale practical classroom assignment problems. *Computers and Operations Research*, 53:42–53, Jan 2015. doi: 10.1016/j.cor.2014.07.012. URL https://www.sciencedirect.com/science/article/pii/S0305054814001956.

[37] R. Sivarethinamohan. *Operations Research*. Tata McGraw-Hill, New Delhi, 2008.

[38] Special cases in the simplex method. URL https://www.brainkart.com/article/Special-Cases-in-the-Simplex-Method_11207/.

[39] SHWETA SINGH. A comparative analysis of assignment problem. *IOSR Journal of Engineering*, 2(8):1–15, 2012. doi: 10.9790/3021-02810115.

[40] Athanasios Vasilopoulos. The assignment problem: Searching for an optimal and efficient solution. *Journal of Business and Economics*, 6(1):1–12, Jan 20, 2015. doi: 10.15341/jbe(2155-7950)/01.06.2015/001.

[41] Jameer G. Kotwal and Tanuja S. Dhope. Solving task allocation to the worker using genetic algorithm. *International Journal of Computer Science and Information Technologies*, 6, 2015. URL http://www.ijcsit.com/docs/Volume%206/vol6issue04/ijcsit2015060498.pdf.

[42] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, pages 32–38, 1957.

[43] Alan Cobham. The intrinsic computational difficulty of functions. In Yehoshua Bar-Hillel, editor, *Logic, Methodology and Philosophy of Science: Proceedings of the 1964 International Congress (Studies in Logic and the Foundations of Mathematics)*, pages 24–30. North-Holland Publishing, 1965.

[44] Pulp Documentation Team. Pulp 1.6.0 documentation - a blending problem, 2009. URL https://pythonhosted.org/PuLP/CaseStudies/a_blending_problem.html.

[45] Vince Knight. Allocating final year projects to students;, Sept 26 2017. URL https://vknight.org/unpeudemath/math/2017/09/26/allocating-final-year-projects-to-students.html.

[46] Thomas Hirsch. Hungarian algorithm in vba/excel, Nov 13 2014. URL https://www.thomashirsch.co.uk/2014/11/hungarian-algorithm-in-vbaexcel.html.

[47] Emanuele Borgonovo. Sensitivity analysis. *European journal of operational research*, 249(3):869–887, 2015. URL http://www.econis.eu/PPNSET?PPN=845699377.

[48] K. Kavitha and P. Pandian. Sensitivity analysis of supply and demand in a fully interval transportation problem. *International Journal of Engineering Research and Applications*, 2(3):1900–1910, May 2012. URL http://www.ijera.com/papers/Vol2_issue3/LJ2319001910.pdf.

[49] Chi-Jen Lin and Ue-Pyng Wen. Sensitivity analysis of objective function coefficients of the assignment problem. *Asia-Pacific Journal of Operational Research*, 24(2):203–221, Apr 2007. doi: 10.1142/S0217595907001115. URL http://www.worldscientific.com/doi/abs/10.1142/S0217595907001115.

[50] Special cases in the simplex method. URL https://www.brainkart.com/article/Special-Cases-in-the-Simplex-Method_11207/.

[51] James Orlin and Ebrahim Nasrabadi. Optimization methods in management science, 2013. URL https://ocw.mit.edu.

[52] Matheus P. Souza, Marconi Pereira, Abilio Oliveira, Felipe Reis, Paulo Almeida, Daniel Crepalde, and Eder Junio Da Silva. Optimization of taxi cabs assignment using a geographical location-based system in distinct offer and demand scenarios. *Revista Brasileira de Cartografia*, 68, 04 2016.

[53] Felipe Reis, Marconi Pereira, and Paulo Almeida. Location-based service to reduce the waiting time for taxi services. 01 2013.

[54] Hai Yang and S. C. Wong. A network model of urban taxi services. *Transportation Research Part B*, 32(4):235–246, 1998. doi: 10.1016/S0191-2615(97)00042-8. URL https://www.sciencedirect.com/science/article/pii/S0191261597000428.

[55] Zhengchuan Xu, Yufei Yuan, Huiliang Jin, and Ling Hong. Investigating the value of location information in taxi dispatching services: A case study of dazhong taxi. page 111, 01 2005.

[56] Betsy Masiello. Faster pickup times mean busier drivers, April 5 2017. URL https://www.uber.com/newsroom/faster-pickup-times-mean-busier-drivers/.

[57] Gaglani Mansi Suryakant. A study on transportation problem, transshipment problem, assignment problem and supply chain management, Oct 22 2011. URL https://core.ac.uk/download/pdf/11822284.pdf.

[58] David F. Manlove School of Computing Science, University of Glasgow, Glasgow, and UK. The hospitals / residents problem (1962; gale, shapley).

[59] D. M. Manlove, G. O'Malley, P. Prosser, and C. Unsworth. A constraint programming approach to the hospitals/residents problem. Springer, Jun 2007. URL http://eprints.gla.ac.uk/3504/.

[60] National resident matching program, 2002. URL http://www.nrmp.org/.

[61] Dan Gusfield and Robert W. Irving. *The stable marriage problem*. MIT Press, Cambridge, Mass. u.a, 1989. ISBN 0262071185.

[62] Georgios Askalidis, Nicole Immorlica, Augustine Kwanashie, David F. Manlove, and Emmanouil Pountourakis. Socially stable matchings in the hospitals / residents problem, Mar 8, 2013. URL https://www.openaire.eu/search/publication?articleId=dedup_wf_001::cf3f0a26c1e494b565383832fbfc296e.

[63] Tianhua Jiang. Production scheduling and worker assignment problems in assembly job shop with deterioration effect. In *2nd International Conference on Electronics, Network and Computer Engineering (ICENCE 2016)*, China, 2016 2016. Atlantis Press 664.

[64] Mian Tahir Aftab, Muhammad Umer, and Riaz Ahmad. Jobs scheduling and worker assignment problem to minimize makespan using ant colony optimization metaheuristic. 2012. doi: 10.5281/zenodo.1058337. URL https://search.datacite.org/works/10.5281/zenodo.1058337.

[65] F. Chauvet, J. M Proth, and A. Soumare. The simple and multiple job assignment problems. *International Journal of Production Research*, 38(14):3165–3179, Sep 1, 2000. doi: 10.1080/002075400418207. URL http://www.tandfonline.com/doi/abs/10.1080/002075400418207.

[66] Gaetan Caron, Pierri Hansen, and Brigitte Jaumard. The assignment problem with seniority and job priority constraints. *Operations Research*, 47(3):449–453, May 1, 1999. doi: 10.1287/opre.47.3.449. URL http://or.journal.informs.org/cgi/content/abstract/47/3/449.