Algorithms and Data Structures for Automated Teaching Workload Allocation

Rohan Sood

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University in partial fulfilment of the requirement for the degree of Master of Engineering

2016
ABSTRACT

Fair allocation of teaching workload to academic staff is one of the most important administrative duties of an educational institution. However, it is often managed manually with limited data and, often with an arbitrary set of constraints due its complexity, thereby resulting in sub-optimal allocations. The objective of this research is to develop efficient methods to automate the realization of a fair and transparent workload allocation system.

A detailed exploration of past attempts, aimed at improving the workload allocation at tertiary institutions, was undertaken to gain a deeper understanding of the various approaches and challenges associated with the process. In particular, it was observed that various approaches were adopted to quantify a variety of components of the workload during the allocation process covering teaching, research and administrative duties.

This motivated the development of an efficient time-based workload model called the Workload Unit (WLU) Model as the quantitative framework for measuring and comparing different faculty workloads. This involved the utilization of a combination of weightages corresponding to nature of teaching duties, the number and appointment of faculty as well as their availability for undertaking teaching duties, and attributes (such as class size, level of course, preparation requirements, etc.) of the courses to be offered. In addition, the proposed workload model takes into account the research activeness and administrative duties carried out by each faculty. The resulting workload is separated into two components, namely formal and informal teaching for the purpose of systematic workload allocation.

Next, an efficient workload allocation method for formal teaching comprising of lectures, tutorials and labs was proposed by taking into consideration of faculty preference and performance, School policies and priorities and availability of teaching expertise. A novel combined cost function was proposed to determine the feasibility of individual allocations called the Feasibility Index (FI). A greedy approach, with activity-type and course-priority heuristics, is implemented to optimize the quality of the allocations. The proposed approach was validated against the workload allocations made at one of the schools of a University consisting of 100 teaching faculty, 120 courses and 1500 students. The resulting allocations improved workload distribution and quality of assignments over existing manual processes,
improving preferences and performance indices by as much as 18% and 7% respectively, while reducing workload variations by 21%.

Methods for the automatic translation of the resulting workload allocations into faculty and student centric timetable were proposed. This was achieved using an iterated greedy approach, with an objective to reasonably spread the allocated workload across the week for both faculty as well as students while making efficient use of teaching venues. The proposed approach has been shown to improve the spread of time table allocations across the week by 20% for both students and faculty. Moreover, it has also improved the utilization of the teaching venues by 30%.

Realizing Final Year Projects (FYP) allocation is the most challenging part of the informal teaching allocation process, methods for its automation have been proposed by taking into consideration other informal teaching duties such as M.Sc. and Ph.D. supervisions. The proposed methods ensure that the student preference are taken into consideration while adhering to informal workload limits for each faculty. An iterative implementation of the optimal Hungarian algorithm which works on a student-project cost matrix with the preference ranks as the cost of assignment was introduced and multiple iterations were invoked to adjust the cost matrix based on workload constraints to achieve desired results. The resulting solutions comprehensively bettered those achieved by a competing existing system used at a reputable school – improving average student preference for assignments by 10% and workload deviations for faculty by 15%.

The above methods were integrated and validated in a real-world scenario. Overview of the technology and platforms used to implement the above solutions have been presented together with screen captures of the platform that is being deployed in the School for past 5 consecutive semesters. This has unreservedly demonstrated the feasibility for automating an inclusive workload allocation process to realize an efficient and fair workload distribution, which is critical to any educational institution. Finally, avenues for future research and improvements are also stated.
ACKNOWLEDGEMENTS

I take this opportunity to express my profound gratitude and deep regard to my supervisor, Dr. Thambipillai Srikanthan, Professor and Chair, School of Computer Engineering, Nanyang Technological University, for his exemplary guidance, monitoring and constant encouragement throughout the course of this project.

I would also like to thank by Dr. Lau Chiew Tong, Dr Tan Ah Hwee and Dr Goh Woi Boon, whose constant support and critical analysis allowed this project to push way beyond what was planned initially. This project would have also been impossible if not for the support of the administrative staff at SCSE – Mr Jeremiah Chua, Mdm Zarina Banu and Mdm Koh Soh Elaine

Lastly, I thank my family, and the very many friends back home in India and here in Singapore who have made this journey a lot more enjoyable than it would be without them.
# Table of Contents

1. **INTRODUCTION** ............................................................................................................... 11  
   1.1 Motivations .................................................................................................................. 12  
   1.2 Objectives ................................................................................................................... 13  
   1.3 Organisation of the Report ......................................................................................... 13  

2. **LITERATURE REVIEW** .................................................................................................. 16  
   2.1 Study of Workload Allocation Models ......................................................................... 16  
      2.1.1 Early Investigations on Workload Models ............................................................... 16  
      2.1.2 The STAR Model .................................................................................................. 17  
      2.1.3 Computer-Assisted Workload Models ................................................................... 18  
      2.1.4 Time-based Workload Models ............................................................................. 18  
      2.1.5 Limitations of existing workload models .............................................................. 21  
   2.2 Study of allocation strategies for teaching assignments .............................................. 22  
      2.2.1 Objectives of the Workload Assignment Problem ............................................... 22  
      2.2.2 Faculty-Course Preferences ................................................................................. 23  
      2.2.3 Goal Programming .............................................................................................. 24  
      2.2.4 Limitations of existing allocation strategies .......................................................... 25  
   2.3 Survey of Timetabling Models ..................................................................................... 26  
      2.3.1 University Course Timetabling Problem ............................................................... 26  
      2.3.2 Automated Timetabling Techniques ................................................................... 27  
      2.3.3 Limitations of existing timetabling techniques ...................................................... 28  
   2.4 Study of student assignment problem and relevant techniques ................................. 29  
      2.4.1 The Assignment Problem .................................................................................... 29  
      2.4.2 The Hungarian Algorithm .................................................................................... 30  
      2.4.3 Limitations of existing techniques ....................................................................... 30  
   2.5 Summary ...................................................................................................................... 31  

3. **DEVELOPING A FAIR AND COMPREHENSIVE WORKLOAD MODEL** ............... 33  
   3.1 Components of Faculty Workload ................................................................................ 33  
   3.2 Workload Unit Model .................................................................................................. 34  
      3.2.1 Part-Time Faculty and Impact on Workload ............................................................ 36  
   3.3 Teaching Contributions and Relaxations .................................................................... 37  
      3.3.1 Research Relaxation ............................................................................................. 38  
      3.3.2 Administrative and Service Relaxation .................................................................. 39  
   3.4 Calculating Optimal Workload for Individual Faculty ................................................. 40  
      3.4.1 Formal Teaching Workload .................................................................................. 41  
      3.4.2 Informal Teaching Workload ............................................................................... 41  
   3.5 Experimental Set-Up and Results .............................................................................. 43
5.4 Using Hungarian to solve Informal Teaching Allocation ............................ 83
  5.4.1 Simple Hungarian Implementation .................................................. 83
  5.4.2 Iterative Hungarian Implementation ............................................... 86
5.5 Experimental Results ............................................................................. 89
  5.5.1 Workload Distribution .................................................................... 89
  5.5.2 Student Preference ......................................................................... 91
5.6 Summary ............................................................................................... 93
6. ATAS IMPLEMENTATION AND DEVELOPMENT ..................................... 94
  6.1 System Architecture .......................................................................... 94
  6.2 Technologies and Platforms ............................................................... 94
  6.3 System Screenshots ............................................................................ 96
7. CONCLUSION AND FUTURE WORK ....................................................... 99
  7.1 Conclusion .......................................................................................... 99
  7.2 Future Work ...................................................................................... 101
    7.2.1 Improving Data Collection and Integration .................................. 102
    7.2.2 Balancing Teaching Allocations across Semesters ..................... 102
    7.2.3 Improving Allocation Results ...................................................... 102
    7.2.4 Analysing Shortages and Future Requirements ......................... 102
    7.2.5 Economic Analysis ................................................................. 103
    7.2.6 Applications beyond Educational Institutions ....................... 103
List of Tables and Figures .............................................................................. 8
References .................................................................................................... 104
List of Tables and Figures

Table 3.1 Sample values for the different weightage parameters used to compute workload.. 35
Table 3.2 Sample values for the appointment-based teaching contribution values............. 38
Table 3.3 Sample values for the different parameters for research relaxation..................39
Table 3.4 Examples for Service Relaxation Value (hours)............................................. 40
Table 3.5 Examples for Service Relaxation Value (appointments)................................40
Table 3.8 Weightage parameter values for student supervision.................................... 42
Table 3.9 Formal Teaching Workload for different teaching activities......................... 43
Table 3.10 Formal Teaching Workload for part-time faculty........................................ 44
Table 3.11 Teaching contributions and relaxations for full-time faculty.......................... 44
Table 3.12 Teaching contributions and relaxations for full-time faculty.......................... 44
Table 3.13 Teaching contributions and relaxations for full-time faculty.......................... 45
Table 4.1 Workload variation levels for faculty across various implementations.............. 68
Table 4.2 Average workload variation values for manual and automated allocations......... 69
Table 4.3 Distribution of faculty preferences for teaching allocation assignments ............ 70
Table 4.4 Distribution of faculty performance scores for teaching allocation assignments... 70
Table 4.5 Average preference and performance scores for manual and automated allocations.................................................................................................................. 71
Table 4.6 Number of teaching hours in a given day for faculty.................................... 72
Table 4.7 Number of working days in a week for faculty.............................................. 72
Table 4.8 Number of lectures hours in a given day for a student.................................... 73
Table 4.9 Number of continuous lectures scheduled for students across different years of their degree..................................................................................................... 74
Table 4.10 Utilization rate for teaching venues across implementations.......................... 75
Table 5.1 Sample problem to demonstrate the Hungarian Algorithm............................. 81
Table 5.2 Simplified example of the FYP assignment problem...................................... 84
Table 5.3 Final solution using simple Hungarian......................................................... 85
Table 5.4 Final solution using iterative Hungarian....................................................... 88
Table 5.5 Workload deviations for FYP assignments using proposed (ATAS) versus existing system (ES) Sem 2................................................................. 89
Table 5.6 Workload deviations for FYP assignments using proposed (ATAS) versus existing system (ES) Sem 1................................................................. 90
Table 5.7 Comparison of average assigned preference rank (for student FYPs) by the proposed
ATAS versus previously used greed-based allocation system – Sem 2 ................................. 91
Table 5.8 Comparison of average assigned preference rank (for student FYPs) by the proposed
ATAS versus previously used greed-based allocation system – Sem 1 ................................. 92

Fig 4.1 Shortest-Path example problem .................................................................................. 51
Fig 4.2 Shortest-Path example with optimal solution ................................................................. 52
Figure 4.3 Shortest-Path example with sub-optimal solution ..................................................... 53
Figure 4.4 Pseudo-code for the naïve greedy-based formal teaching allocation algorithm .... 54
Figure 4.5 Pseudo-code for the improved greedy-based formal teaching allocation
algorithm ................................................................................................................................. 56
Figure 4.6 Iterated greedy process .......................................................................................... 62
Figure 4.7 Example problem for iterated greedy ...................................................................... 63
Figure 4.8 Construction of greedy candidate solution for the example problem ................. 63
Figure 4.9 Destruction and Construction phases of the iterated greedy process for the example
problem .................................................................................................................................. 63
Figure 4.10 Pseudo-code for the iterated greedy implementation for timetabling ............... 67
Figure 4.11 Comparison of variations in workload allocated to faculty across various
implementations ........................................................................................................................... 68
Figure 4.12 Graphic comparison of average preference and performance scores for
allocations ................................................................................................................................. 71
Figure 4.13 Graphic comparison of lectures scheduled in a day for UG year 2 students across
the week .................................................................................................................................... 74
Figure 5.1 Steps for the Hungarian Algorithm .......................................................................... 81
Figure 5.2 Solution of the sample problem using Hungarian Algorithm ............................... 82
Figure 5.3 Solution of the simplified FYP assignment problem using Hungarian Algorithm.. 85
Figure 5.4 Pseudo-code for the iterative Hungarian implementation for FYP allocation ...... 87
Figure 5.5 Second Hungarian iteration to solve the simplified FYP assignment problem ..... 88
Figure 5.6 Load distribution of FYP allocation using the proposed (ATAS), as well as existing
system (ES) ................................................................................................................................ 90
Figure 5.7 Comparison of average assigned preference rank (for student FYPs) by the proposed
ATAS versus previously used greed-based allocation system ............................................. 92
Figure 6.1 Client-Server Architecture ...................................................................................... 94
Figure 6.2 Interface to edit core system parameters for workload model ......................... 96
Figure 6.3 Interface to add a new faculty record to the database ........................................... 96
Figure 6.4 Pre-allocation report providing an overview of the workload structure as well as ensuring integrity of system data. 97

Figure 6.5 Interface to trigger the allocation process for lectures. 98

Figure 6.6 Overview of allocation results. 98
1. INTRODUCTION

In this chapter, we briefly introduce the problem of automating workload allocations for faculty, and the motivations and objectives of the corresponding research. We also outlay the organisational structure for this thesis.

The unprecedented growth in computational power over the last century has allowed for automation and optimization of processes at a pace last seen during the industrial revolution. While the initial targets were simple, manual processes, the widespread availability of personal computing has allowed for the focus to shift towards increasingly complicated processes. In today's world, systems exist which look to automate and optimize processes as complex as trading of stocks, aviation and even personal transportation with self-driving cars.

Automation is primarily focussed on reducing the manual intervention required in a process to its absolute minimum. It is relatively straightforward if the process can be broken down to a set of simple, step-wise, repetitive tasks. But as these tasks grow in complexity, requiring a certain level of 'expertise' like in the case of human decision making, the problem of automation becomes increasingly challenging.

The idea behind optimization on the other hand is to extract as much as possible from as little as possible, while adhering to the certain constraints governing the extraction. There are different techniques available which optimize different single objective processes like production levels of goods given their demand or the shortest route back home using GPS data. The problem, however, becomes a lot more complex when there exist multiple objective functions, and almost impossible to solve when these objectives are conflicting in nature.

In this research exercise, we aim to automate and optimize the process of allocating teaching workload. We use a harmonious mix of algorithms and data structures which model the domain expertise of a human administrator to achieve automated results with optimization levels that are extremely difficult to achieve manually.
1.1 Motivations

Human Resource is arguably the most important resource for any organization, and tertiary educational institutions are no exceptions. A dedicated workforce is often assigned with the duty to manage this precious resource well, at a considerable cost to the organization. In tertiary institutions, this cost is further exaggerated by a need for members of the senior faculty to double-up as administrators. This is owing to the fact that they possess the required domain knowledge, which serves as a crucial pre-requisite for intelligent decision-making when it comes to the problem at hand i.e. assigning of teaching, supervisory and administrative duties to members of the faculty. But despite their expertise and best efforts, a fair, highly optimal workload allocation system evades us due to the size and complexity of the assignment problem and the limits of manual computational capability.

The allocations of duties for academic staff is not a new problem, as seen by the work done by Ball as far back as 1980 [1] and continues to date [2]. But due to the myriad of constraints and parameters involved, there is no one solution that has been found to fit all. A study to compare most common workload allocation models like ‘Contact Hours’, ‘Actual Hours’ and ‘Point-based’ [3] reported considerable shortcomings, with management and staff dissatisfaction being directly proportional to the complexity of the model – owing partially to the cynicism directed towards a system that cannot be easily understood.

Beyond modelling the workload, expertise matching is another challenge for the allocation process, especially in a school where there exists a considerable overlap in faculty expertise. Faculty preferences are considered to be of paramount importance when making allocation decisions, but school policies and student feedback are often ignored in most existing approaches.

Academic workload allocation is incomplete without timetabling of the assigned teaching activities. It must be done in a manner that is student and staff friendly, while at the same time efficient in its utilization of physical resources like lecture halls and specialized labs within a time-bound system. Extensive research on the topic exists [16][17], with numerous strategies of varying complexity having been implemented [18][19][20]. Yet, the near-infinite possibilities for any reasonably-sized time-table provides an opportunity to find a solution that
works best based-on varying considerations.

There is also the problem of managing supervisory assignments in the form of undergraduate final year projects (FYP) and graduate dissertations for M.Sc. and Ph.D. programmes. We refer to this as the informal teaching load. The problem statement can be approximated to the assignment problem, but has additional constraints in the form of workload limits which make the problem challenging.

To solve these problems, there was need for developing a workload model and allocation strategy that was simple and effective, while at the same time required minimal additional resources to be implemented.

1.2 Objectives

The main objective of this research exercise is to develop an automated teaching allocation system that is fair, equitable and inclusive. It must be based on a workload framework that gives due consideration to the various duties that a member of faculty may be assigned, including teaching, research and administration. The system must be capable of quantifying real-world variables and constraints, as well as realizing the underlying relationships between them.

The system must be driven by an allocation strategy which is optimal and favourable, not just to the faculty, but also to the students and is in line with the needs of the management. At the same time the system must be simple enough for all parties involved to understand so as to minimize adoption-resistance and provide adequate levels of transparency. It must also be able to schedule the allocation workload in the form of a feasible, student and faculty-friendly timetable. Lastly, the system needs to be largely automated, with minimum requirements for human interference so as to be efficient and free of bias.

1.3 Organisation of the Report

In this chapter, we discussed the motivations behind the pursuit of creating an automated teaching allocation system. We have also listed the objectives that will serve as the key design philosophies that will guide the development process.
Chapter 2 offers an overview of existing work done with regards to problem of managing faculty workload. While no attempts have been made to build a comprehensive system that addresses the complete workload problem, there exist multiple research exercises which have tried to solve different parts of the problem. For each of the exercises, we look at the strengths and weaknesses of the proposed approaches, and the need for a better solution.

Chapter 3 deals with the development of the workload model, which is the quantitative framework used to measure and compare the workload for different activities undertaken by an average faculty member. We also discuss how to assess and adjust the workload capacities for individual faculty members based on their involvement in different activities like research and service duties.

In Chapter 4, we focus on the allocation and timetabling for the formal teaching component of the overall faculty workload, i.e. the actual assignment of courses and schedules to faculty members for teaching. We consider the interests of the various stakeholders involved in the decision making process and accordingly construct a problem formulation. We then proceed to solve it using a greedy-based technique for the allocation problem, and then improve upon it based on observed results and real-world feedback. For the timetabling problem, we focus on reasonability of spread and student/faculty friendliness of the timetable. The proposed solution is based on an iterated greedy approach that removes certain components from a greedy solution and re-constructs it to improve the overall results. The final implementation results show the improvement in the quality of the automated allocation and timetabling over their manual counterparts.

Chapter 5 continues the discussion on allocation strategies by shifting the focus to the assignment of supervisory duties to faculty. We consider the problem of informal teaching which involves the assignment of students to projects proposed by the faculty. Using a novel approach involving multiple iterations of the Hungarian algorithm, we are able to find a near-optimal solution to the problem which gives the students the projects they prefer without violating the faculty workload limits. Results of the proposed algorithm are compared with an alternative system used by the school prior to the implementation of this research exercise.
Chapter 6 provides an overview of the technologies and platforms utilized for the development of the Automated Teaching Allocation System (ATAS). Lastly, Chapter 7 provides the set of conclusions drawn, and the possibilities for future improvements and research.
2. LITERATURE REVIEW

Strategies and models for allocation of workload for academic staff have been topics of increasing interest in recent years owing to the growing number, and size, of universities across the globe. In this section, we go through some of the existing work done in this regard to solve the workload allocation issues that are faced by educational institutions almost universally.

2.1 Study of Workload Allocation Models

The workload model is the set of rules which quantifies the faculty’s work into a mathematical framework such that comparisons among them can be made. A number of methods to do this have been proposed over the years.

2.1.1 Early Investigations on Workload Models

R. Ball conducted one of the first studies where concerns regarding availability of staff and their effective utilization via different allocation techniques like student ratios, contact hours and even informal methods (considering the subjective nature of the problem), were investigated [1]. Each technique was observed to have its own set of merits and de-merits.

1. Student to Staff Ratios were simple to utilize, but betrayed the economies of scale that one might argue exist with increasing number of students [24]. This was because the cost-effort of teaching a bigger class did not necessarily increase linearly with its size. This limitation was fixed in a variation of the ratio model by Fielden and Lockwood [15], though it was still observed to be too crude to realistically model the workload distribution.

2. The ‘Contact Hour’ method was another technique, which at the time was favourably received in a number of North American institutions. It calculated workload as a direct reflection of the number of teaching hours associated with a course/subject, and was reported to work well due to the tightly constrained and highly specific relationship between teaching time and academic units awarded. However, Ball argued that differing teaching duties like (tutorials versus lectures) had differing implications in terms of the effort required.
3. Other techniques like ‘Q-ratio’ (statistical approach using national survey data) or an ‘Informal method’ (case-by-case evaluation, no consistency) were also discussed.

However, notable among these was the ‘Workload Model’ which used mathematical models to generate workload for courses/subjects independently using multiple parameters like class size, group size (for tutorials/labs), and teaching hours (course-specific, as well as overall)[25][26]. This technique was found to be the most inclusive, but concerns were raised regarding the practicality of implementing it and the ‘large’ quantity of data that would need to be processed at a time when personal computing as a concept was still nascent. Regardless, it was a technique which showed the most potential - enough to merit ‘further research and development’.

2.1.2 The STAR Model

Multiple, un-related research work [26][27][28] carried out across universities re-affirmed the need for a well-structured approach towards a quantitative workload model. A notable step in this direction was the Supervision, Teaching, Administration and Research Model or ‘STAR’ as introduced by Finlay and Gregory [21]. In this model;

- Supervision referred to the guiding of undergraduate students through their projects and industrial training assignments, as well as the research work of post-graduate students.
- Teaching constituted the conventional lectures and tutorials for courses offered by the university/department.
- Administration was the carrying out of duties essential to meet the accountability set by the educational institution.
- Research was the component that included work done on a faculty’s own research interest beyond what was directed towards their students.

The motive behind the STAR model was to find the fair amount of teaching workload that can be assigned to a faculty, based on their relative involvement in the supervision, administration and research domains. The paper covered various techniques to measure each component, and deduce the teaching workload based on different models like linear, exponential and their variation thereof. Overall, it provided an in-depth yet objective way to look at faculty workload.
2.1.3 Computer-Assisted Workload Models

A case study by Burgess in [5] surveyed the use of computer assisted systems for workload allocations and planning purposes. The study looked at the workload planning systems implemented at three different institutions in the UK.

1. Case ‘A’ studied a conventional teaching-only contact-hour based model implemented using a Decision Support System (DSS) built in dBASE running on DOS-based 8086 personal computer. Case ‘B’ was a STAR based system (as described above), which converted faculty activities into actual hours and ran on a Windows 486 PC.

2. Case ‘LUBS’ was also a STAR based system, but was based on a conversion of activity hours into utility values and did not touch upon the nature of computer support used. Case ‘A’ was inherently inferior owing to its limited approach, and while Case ‘LUBS’ offered a more comprehensive coverage, it had its own set of limitations like variations within teaching activities and capacity flexibility.

3. Case ‘B’ overcomes these limitations with weighted parameters and an in-built capacity limit. It had its own set of shortcomings when it came to dealing with accuracy of workload quantification and part-time faculty, though these existed for other the other two cases as well. The takeaway however was this - comparisons and computations for different workload models which were previously deemed too expensive now became viable owing to the advancements made in the world of computing.

2.1.4 Time-based Workload Models

An attempt to find a fair and equitable workload model was carried out by Blitzer [2] in South Africa with an attempt to find a fair and equitable workload distribution, while at the same time ensuring general faculty satisfaction. The foundation for the attempt was based on the survey carried out by Soliman [29] where two frameworks for faculty workload were explored. The first was an economic costs and earnings based framework, where workload for an activity was equated to its financial output. The second framework was based on the time spent by faculty on various activities. The survey clearly articulated that the time-based approach was preferred widely over the earnings-based approach owing largely to the fact that a number of essential academic activities like development work and service to communities, were unfunded. These findings were in line with reports from other researchers [30][31][32], and it was asserted that
time-based models dominated when it came to judging academic workload and as a result was chosen as the preferred system.

Blitzer illustrated the phased-development of the time-based model at a particular university’s faculty of business. In the first phase, categories and sub-categories of workload activities were decided in consultation with members of the faculty. The conclusion was similar to the STAR system above, with the addition of a community-engagement category. Next, the base number of typical hours associated to activities in each category was determined. Once fixed, various weighted adjustments called ‘indicators’ were considered for each of the activity.

For example, teaching 1 hour of lecture entailed an additional hour of preparation for the faculty, hence a factor of 2 needed to be introduced. Furthermore, if a course was being taught for the first time, it was agreed that the preparation time in this case would be about 1½ hours instead, so a factor of 2.5 would be applicable. Similar adjustments were considered for undergraduate versus graduate courses, assessment time based on number of students and outside the class interaction. While the above example is just for one of the activities under the teaching category i.e. lectures, similar considerations were made for other activities and categories. Graduate students were deemed to require 300/200 hours respectively for Ph.D./Master’s thesis completion. Organizing a workshop entitled a faculty 50 hours of work, while writing an article in a respectable journal entitled a 100. All of these considerations were made on the back of a 2200 working-hours annually for the average full-time faculty.

In summation, these hour-conversion for each activity were computed, and an attempt to balance workload distribution across faculty was made. The resulting system found general acceptability, owing largely to the active contribution of members of the faculty in arriving at the hours for each activity. The author offered the following conclusion based on the exercise:

- It is preferable to involve faculty in the workload modelling and allocation process
- Individual faculty performance should be included in assessing workload
- Validity and accuracy of data used is extremely important for an effective system
The strength of a time-based workload model can be seen in multiple implementations, like in the study by Iris Vardi [3] who compared Workload Allocation Models or WAMS categorized into three main categories - contact hours, actual hours and points based models.

1. ‘Contact hours’ refers to a time-based model where typical values for teaching are used, with no consideration to the type of contact (lecture, tutorial, and lab) or other factors (like class size, novelty). Standardized additions to the number of actual teaching hours are made for preparation, assessment etc. The administrator compensates for factors like class size outside the system by providing additional support. The basis of the system is to achieve relative equitability among staff using a system that is simple to use and comprehend. While this system is observed to work well when expectations for teaching hours are reasonable and supplements can be made, it has its limitations when it comes to accounting for non-direct duties like research, community work and clinical trials.

2. ‘Actual hours’ on the other hand is a time-based model with due considerations made for the nature of work that the faculty is engaged in. For example, for marking activities the type of assessment and number of students is factored in, while for consultation activities time to meet students as well as answer email queries is considered. The resulting system is one that is more comprehensive and can be easily related to human resourcing and costing. However, it is also more complex, leading to a lower degree of transparency which in turn was perceived by members of the faculty as a form of distrust by the administrator. The system was also perceived by some as being too controlling due to its fine-grained nature.

3. The ‘Points model’ is another attempt to quantify workload using a time-related metric by formulating methods to compute ‘points’ based on the activity assigned. Faculty were avoided points for the various activities they undertook, with the points themselves being loosely tied to the hours of effort involved. This model was perceived to be the least effective, both by faculty and the administration. The faculty complained about the lack of relationship between ‘hours’ and ‘points’, and the bias towards points awarded for activities that had higher financial or budgetary returns. The administration found the system to be lacking when it came to identifying the over/under-loaded faculty, and were unhappy with the resulting negligence of ‘non-point’ activities.

The time-based models ended up being widely preferred over the points model. Between the two time-based models of ‘contact hours’ and ‘actual hours’, the simplicity of the former was
found to be most desirable and ensured lowest levels of faculty dissatisfaction. However, this simplicity also resulted in a system deemed to be incomprehensive by the management due to its inability to account for the real-world complexities that form a considerable share of the overall workload.

2.1.5 Limitations of existing workload models

The models discussed so far address the core principles of quantifying faculty workload. However, there exist a number of limitations among them:

1. **Non-teaching activities** – Time-based models do not account well for non-teaching activities that cannot be easily quantified into hours. These include research, administrative and supervisory duties. Research and administrative duties are especially harder to quantify into hours, as they’re extremely varied based on an individual’s field of interest and capacity.

2. **Hard Capacity Constraints** - By assigning a fixed number of hours to an activity, teaching or otherwise, a hard limit is enforced on the total teaching capacity available to the institution. Theoretically, it is possible for a faculty to justify an inability to carry out teaching assignments owing to the hours awarded from research and administrative activities. This is undesirable, as it forces the administration to ‘over-load’ faculty.

3. **Equivalence of Workload** - With time (contact-hours) as the sole measure, all quantifiable activities of similar duration are marked as equivalent from the workload perspective. As we realized, this is not true in practice. Lectures require greater preparation and active effort than tutorials, which in turn require more effort than lab sessions.

4. **Initial Efforts** - The enumeration of each activity and computing/deciding their corresponding number of hours, despite being a one-off exercise for the most part, is a challenging task.

5. **Historic Data** – Most time-based model estimate components like research output based on historic data. For example, the number of hours reduced from a faculty’s total available hours would be based on the number of publications in the previous year, while the teaching allocation is being carried out for the subsequent year.

6. **Micro-management** - As discussed in the paper by Vardi [3], this model risks being perceived as micro-management and ‘too controlling’ by the faculty.
2.2 Study of allocation strategies for teaching assignments

A workload model solves the quantitative element of the workload distribution problem, but it still leaves the qualitative section unaddressed. Regardless of the model used to quantify workload, faculty satisfaction will continue to be low if there is not a methodical approach to the actual workload allocation itself. The problem is not limited to just ‘how much’ work a faculty member must do, but also includes the question of exactly ‘what’ must be assigned to them, especially in cases where there are multiple, seemingly-viable options like in case of course-selection and student supervision. The workload assigned to a faculty must be well-spread out across the various activities, while must also give due consideration to the faculty’s expertise, capabilities and preferences. This results in a problem statement that has multiple, and often conflicting, objectives.

2.2.1 Objectives of the Workload Assignment Problem

Harwood and Lawless [23] detailed the multiple objectives of the workload assignment problem by identifying three key stake-holders and their respective objectives, or ‘goals’ as they preferred to call it:

- The student body, which requires that a certain minimum variety of courses be offered in line with student interests.
- The administration, which requires that all course offerings are reasonably staffed and the workload is fairly distributed devoid of any bias
- The faculty, who are the key players and have multiple goals with regards to the type and spread of workload assigned

While the objectives of the students and administrations were observed to be fairly simple, the faculty’s needs were more complex and required exploring. To do this, Harwood and Lawless carried out a faculty survey asking them to rank their specific goals in order of priority. The results of the survey were then collated to arrive at a list of goals in order of decreasing importance. The ordered list is presented below:
1. Quantity of workload assigned
2. Flexibility in terms of time-slots assigned
3. Minimize assignment of new courses to save preparation time
4. Minimize time-gaps between classes assigned
5. Minimize repetition in terms of number of sessions of the same course
6. Minimize number of teaching days in a week
7. Minimize number of classes outside regular working hours

These above goals were used to define the constraints of an objective function, which was subsequently solved using the mixed-integer goal-programming approach. During the solving process, it was realized that some of these goals were conflicting and could not be attained in their absolute form and as a result minimal deviation was allowed and deemed acceptable. Also, while computing resources in the form of a Univac 1108 were utilized, the program was terminated at the first feasible solution – meaning the optimality of the solution was neither pursued nor achieved. However, the results were deemed acceptable enough by the department where the system was implemented - albeit manual adjustments were made post the program-execution.

2.2.2 Faculty-Course Preferences

A more direct approach to the teaching workload allocation problem incorporating faculty-course preferences as a key consideration into the allocation strategy was carried by Shih and Sullivan in 1977 [6]. They proposed a zero-one programming model to allocate courses to faculty based on their preference, while also generating a time-table in parallel. This approach required faculty to provide a list of course preferences in a weighted form, with the sum of weights being equal to one. The problem was then formulated as an optimization problem with the primary objective to maximize the total course preference score, followed by assigning the faculty-course pairs into viable time-slots. Additional constraints can were also introduced to model real-world issues, like assigning related courses to the same faculty or limiting over-all workload for sub-set of the faculty etc.

Once the problem was formulated along with its constraints, it was solved using ‘Code RIP30C’ [22]. The resulting solution was severely limited by the number of variables it could handle, so while in theory the solution should provide the required results, it was almost impossible to
achieve them in practice. However, it may be noted that this was 40 years ago at a time when capabilities of computing hardware were extremely limited.

Alternative approaches have also been considered to circumvent the issue of sourcing the actual preferences from faculty. One such approach was based on external factors as explored by Tang et al [4] where they offered a method to estimate preferences from available literature published by the faculty. Indirect sourcing allowed for the faculty’s preferences to be considered loosely, but had the distinct advantage of being based on solid data and discounting other arbitrary considerations like preparation time or course timings which faculty may unfairly affect their choice of courses.

2.2.3 Goal Programming

Marc et al [10] proposed a goal programming model which was built upon the research of Harwood and Lawless, while at the same time gave higher precedence to faculty-course preferences during workload allocation. Goal programming in itself is a generalization of linear programming in order to deal with multiple, conflicting objectives.

The key motivation in the proposed approach was to simplify the complexity of the existing systems to a level that support staff could carry out the workload allocations with little assistance from key members of the administration and/or the management. Additionally, by setting an objective function which had a heavy bias to minimize the sum of all preferences, a more faculty friendly workload distribution was achieved. However, it was also noted that owing to the inherently conflicting nature of ranked preferences, it was almost always impossible to achieve a perfect ranked distribution.

This approach was further built upon by Badri in [11] to modify it into a sequential two-stage process. The first stage dealt with the faculty-course allocations, while the second stage handled faculty-time allocations by introducing additional constraints and variables. With a dual system where both faculty-course as well as faculty-time allocations could be adjusted, the problem complexity grew exponentially. It was realized that the workload problem is multi-objective problem with conflicting constraints owing to the different stakeholders involved – administration, students and faculty. Of these, faculty have been made the primary focus.
To solve this goal-programming problem, a matrix based approach was proposed. The faculty were represented as rows, courses as columns and their ranked preference as the cell value in the first stage. For the second stage, the columns were replaced with time slots and the values are determined by the faculty preferences and other variables like resource availability. The problem was then solved using techniques very similar to those in [10]. The author claims the staggered approach allowed for capturing dynamic aspects of the assignment problem while continuing to be a convenient decision support system.

2.2.4 Limitations of existing allocation strategies

As comprehensive as these proposed workload allocation strategies are, they still left much to be desired owing to the following limitations:

1. **Single Workload Priority** – All existing approaches to the workload allocation problem for faculty-course assignments consider all courses to be of equal priority and status. In practice, this is not the case. Lower year courses, and courses which serve as pre-requisites for multiple advanced courses are more important and play a bigger role in determining the quality of educated offered and received at the university. As a result, there is a need to prioritize some courses over others.

2. **Student Involvement** – While faculty and the management / administration are the key share-holders of the workload allocation system, it can be argued that students are the ones most impacted by it. Yet, students are not represented in most of the allocation algorithms, and where they are it is for trivial purposes like the quantity and variety of courses to be offered.

3. **Dynamic nature** – A university (or for that matter any workplace) is inherently dynamic in nature. The workload and the resources available to carry them out keep changing, like a course being scrapped due to students dropping out, or a faculty leaving for any reason. It is not feasible to disturb the entire workload assignment due to these unexpected changes by re-running the entire allocation algorithm from scratch. Instead, the algorithm must provide for a module that can quickly ‘fix’ these issues, while making the least amount of changes to existing assignments.

4. **Complexity versus Optimality** – Comprehensive and complex algorithms can provide near-optimal allocation results in theory, but in practice this is rarely the case. With hundreds of variables and constraints incorporated using as many as a few thousand equations
representing them, the resulting solution can become increasingly difficult to break down and rationalize to the various stake-holders. The resulting lack of transparency can lead to even lower faculty satisfaction, which is counter-intuitive to the purpose of this research exercise to begin with. Therefore, there is a need to strike a balance between system complexity and optimality.

2.3 Survey of Timetabling Models

Timetabling is the scheduling of activities into different time-slots while simultaneously allotting the necessary resources, in a conflict free manner. In this section, we discuss course timetabling in universities and different ways in which it can be has been done.

2.3.1 University Course Timetabling Problem

The university course timetabling problem (UCTP) refers to the problem of assigning time-slot, venue and relevant equipment where applicable to each of the faculty-course assignments made at a university, in a way that is conflict-free for both student and faculty.

The main differentiator between UCTP and other forms of timetable, like school timetabling, is that a student takes multiple courses and these courses cannot clash. So while two courses may take up completely different resources that are readily available, it may still not be feasible to schedule them at the same time. The exact problem formulation has been detailed in multiples works, like those by de Werra [37] and Tripathy [38], but to summarize there are two main sets of constraints:

1. Hard Constraints
   - A student/faculty cannot attend/teach more than one class at a time
   - A venue can only be used by one class at a time
   - The scheduled venue must be large enough for the class
2. Soft Constraints
   - Minimize number of consecutive classes for student/faculty
   - Minimize the spread of classes in a day for student/faculty
   - Minimize the number of days in a week that a student/faculty has classes
Any solution which violates the hard-constraints even once, or is incomplete, is deemed as infeasible. All feasible solutions can be evaluated relative to each other based on the number of soft-constraint violations and the weightage assigned to each constraint.

2.3.2 Automated Timetabling Techniques

Applying the above conditions to a mathematical formulation of the problem leads to a large number of variables and constraints, making manual solving extremely long and cost-prohibitive. As a result, there is a need to automate the process. Below are some of the surveyed techniques used to automate and solve the UCTP [39][40]:

- **Linear Programming** – As discussed earlier in section 2.2, several authors solved the combined workload-timetabling problem using variations of the linear programming techniques [6][10][11].

- **Direct heuristics** – A simplistic approach, where the timetabling system uses a user-defined heuristic to sequentially assign time-slots/venues one course at a time in a conflict-free manner as far as possible. In the event that the system is unable to meet the hard-constraints, it goes back to prior assignment to try and swap some of them in order to create a feasible slot. An implementation of this kind of system can be seen in the form of the SCHOLA system design by Junginger [41], which uses ‘urgency’ of the classes as a heuristic.

- **Graph Colouring** - Selim [42] and de Werra [37] have both proposed solutions involving a reduction of the UCTP to the graph colouring problem. The basic underlying principle is to associate each class (lecture, tutorial or lab) to a vertex in a graph, and for each pair of classes that cannot be scheduled together for any reason - faculty clash, student clash or venue clash - there exists an edge between the two vertices. Other factors like unavailability and pre-assignments are handled using additional colour-constraints. By performing the necessary time-based coloration, a feasible timetable can be generated.

- **Network flow techniques** – Some authors [42], [43] propose a timetabling algorithm based on network models. The problem is formulated as a multi-level network, plus an additional source and sink. There is a level each for department, faculty and time/venue, with vertices on each level for each of the options available. The vertices on one level are connected by edges to the vertices on the other level if and only they make a feasible pair. The edges values are binary i.e. 0 or 1, and due to uni-modularity, an optimal solution is guaranteed to possess all integer values. The coefficients of the objective function are assigned based
on availabilities and preferences, and can be solved in polynomial time. However, multiple iterations are generally required due to the constraint of a faculty delivering only one class at a time is not captured in this mode. In fact, human intervention may be required at times to fix the infeasibility issues.

- **Rule-based approach** - A technique based on expert systems is given in [44], where Solotorevsky et al. define a rule-based language, called ‘raps’. Raps was used for specifying general resource allocation problems, including UCTP where time periods are considered as resources need to be assigned to the activities/classes. ‘raps’ has 5 types of rules – assignment, constraint, local change, context, and priority. Assignment rules assign classes to periods, linearly, and are supplied by the user. Constraint rules specify positive (must be satisfied) and negative (must fail) constraints that the solution must adhere to, help in identifying conflicts as soon as they arise. Local change rules specify the action to perform when the system gets stuck at an assignment. Context rules select the active context and priority rules determine the priorities of the classes and the periods in those contexts. The system then works in one of two possible ways: greedy (skip failed assignments) and non-greedy (manage failed assignments using local change rules).

- **Constraint Logic Programming (CLP)** - A (CLP) system [45] allows for modelling specific search problem, along with the ability to declare variables, domains, and constraints. The CLP finds solutions using a propagation and pruning technique to identify and remove inconsistencies. It is essentially a backtrack search, with a look-ahead attempt to identify failures sooner. A depth-first, branch and bound search is used as a basis by the CLP to solve optimization problems. A notable implementation to solve the timetabling problem was carried out by [46] using the CLP language ‘domlog’.

- **Others techniques** - Genetic algorithms have been used for timetabling purposes with good results [47] [48]. These algorithms generally involve modifying individual solutions in line with the theory of natural selection – involving mutation, inheritance, selection and crossover. Logic programming is another approach used by [49], wherein a priority-based heuristic implementation using prolog was carried out. Logic programming techniques are also used in [50] and [47]. Some other approaches like those of [23] using mixed integer programming have already been mentioned in previous sections.

### 2.3.3 Limitations of existing timetabling techniques

Timetabling techniques proposed in existing literature cover a large variety of constraints and
problem variants. However, with regards to the objective of this research thesis which targets fair and equitable workload distribution in faculty-centric manner, there exist a few constraints that have not been addressed:

1. **Reasonability of Spread** – While a lot of focus is exerted on ensuring that the time-tables are clash free, there exists almost no techniques to tune a viable timetable to be more faculty and student friendly by spreading classes in an intelligent manner.

2. **Faculty Preference** – Existing timetabling have limited options in terms of allowing faculty to choose when and where their classes are scheduled.

3. **Flexibility** – Universities exist in a dynamic world, where there exists a need to be re-scheduled or add classes due to a number of reasons which has not been explicitly covered by the surveyed techniques.

4. **Comprehensibility** – The algorithm behind the timetabling process should be easily comprehensible and simple to explain so as to improve transparency between the schools administration and faculty.

### 2.4 Study of student assignment problem and relevant techniques

Assignment of students to faculty for supervision of their Final Year Projects or graduate thesis is termed as the informal teaching workload. This section looks at techniques used for allocating this informal teaching workload in a manner that is fair to both faculty and students.

#### 2.4.1 The Assignment Problem

The assignment problem is the problem of choosing an optimal assignment of \( n \) tasks to \( n \) agents, assuming that a numerical rating is given for each agent’s viability for each task [14]. This forms the basis of our informal teaching or student assignment problem – we must assign \( n \) projects (agents) to \( m \) students (tasks), given each student’s preference for the given project (numerical rating).

Proll [33] provided an objective description of student assignment problem and insights into how to formulate it as an optimization problem. The author proposes a matrix-based approach, where students make the rows and available projects make the columns. The cell values then reflect the ranked preference for a given project by the respective student. He then continues
to formulate an optimization problem based on minimizing the sum of preferences, and solves it using existing programming solutions. While there exists merit in the proposed formulation and representation of the problem, the paper offers limited novelty and lacks insights into actually solving it.

Gao et al [34] attempted to solve a similar problem titled the ‘Freshman-Seminar Assignment Problem’. A similar set-up exists, with students preferring seminars instead of projects and rank them in an ordered manner. Apart from minimizing the sum of preferences, the authors attempted to minimize the variance of student preferences so as to avoid having seminars that consist only of students who ranked them lower on their preference scale. As for solving the problem, two algorithms are proposed – an approximation technique titled ‘balanced algorithm’ and an exact technique using the famous ‘Hungarian algorithm’.

2.4.2 The Hungarian Algorithm

The Hungarian Algorithm is a combinatorial optimization algorithm that solves the assignment problem in polynomial time, having a time complexity of \(O(n^4)\) reducible to \(O(n^3)\). It was developed and published in 1955 by Harold Kuhn [13], and is the standard go-to-algorithm for solving one-to-one assignment problems as can be seen by its 5000+ citations [34][35][36] to date. The Hungarian algorithm works well when the problem statement can meet two broad conditions:

1. The number of agents equals the number of tasks, leading to a square cost matrix
2. There exists a simplified cost for each agent-task assignment

The paper describes the method to compute a solution such that minimum cost is achieved, and proves the optimality and completion of the solution via multiple theorems. The only shortcoming of the proposed solution is its time complexity.

2.4.3 Limitations of existing techniques

The informal teaching allocation problem is a complex variant of the assignment problem, as there is an additional faculty workload constraint over the one-to-one student-project assignment. None of the existing techniques cover the incorporation of this additional
constraint in an optimal manner.

2.5 Summary

We began the literature survey with an exploration of the existing models used for measuring faculty workload. It was established that the core workload components are supervision, teaching, administrative duties and research, with each component having its own set of intricacies. Among the models studied, time-based models seemed to be the most popular and generally offered the best results when compared to others with regards to quantifying workload. They were limited however by their ability to deal with real-world constraints and the absolute nature of the workload quantization process.

A number of techniques like linear programming and goal programming, have been used to allocate the faculty workload realized above in a manner that improves the overall fairness and equitability of the distribution while simultaneously ensuring faculty satisfaction. As realized from these attempts, the workload problem is multi-objective problem with conflicting constraints owing to the different stakeholders involved – administration, students and faculty. Of these, faculty have been the primary focus in existing works, stressing the importance of incorporating their preferences into the allocation strategy as far as possible. However, most proposed approaches have been overly complex, while at the same time ignorant of certain key real-world constraints like priorities and student feedback.

We also studied the assignment problem and the Hungarian algorithm in relation to the informal teaching assignment i.e. the assignment of students to faculty for supervision based on the student’s ranked preferences. While the Hungarian algorithm works well as far as optimizing the student preferences of the overall assignment, it struggles to deal with the workload constraint that limits the number of students that can be assigned to the faculty.

Timetabling as a research topic has been explored extensively, as is seen by the multiple works discussed in this literature survey. The university course timetabling problem (UCTP) has an added constraint (compared to the generalize version) that students can choose to attend multiple courses and these must not be offered at the same time. The different techniques like linear programming, direct heuristics, network flow, graph colouring etc. use to solve UCTP work well to provide feasible solutions to the core problem that are largely satisfactory in nature. Having said that, there are certain features that are left to be desired. These include
scheduling remedial/make-up lessons, incorporating individual faculty preferences and timetable spread.

In conclusion, the existing research provided two necessary insights into teaching workload allocation problem. Firstly, it provided motivation in the sense that the problem of optimal workload allocation exists across universities and is not limited to one school or university. And secondly, it allowed for a sound understanding of the problem at hand and possible approaches that can be used, along with their merits and demerits. The various limitations observed across workload models, allocation techniques and time-tabling, need to be overcome. In the subsequent sections, we engineer a solution which does so, while simultaneously meeting the rigorous demands and high-expectations of a real-world, automated decision support system in academia.
3. DEVELOPING A FAIR AND COMPREHENSIVE WORKLOAD MODEL

As mentioned previously in Section 2.5.1, existing workload models are limited by numerous shortcomings. To overcome these limitations, we engineer a hybrid model that complements the absolute nature of time-based systems with a non-teaching engagement system that is predominantly relative in nature. We call this proposed system a contribution-based, workload unit model. This model served as the back-bone of the Automated Teaching Allocation System described later.

3.1 Components of Faculty Workload

Teaching, research, administration/service, and student supervision – these are the four core-components of an average faculty’s workload, as have been described in [21]. However, the quantization and comparison of different types of workload is a challenging task, and makes the resulting workload model extremely complex and difficult to comprehend from an external (faculty) point of view.

Components like research offer very little flexibility in terms of assignments made by the administration. Conversely, there is a consistent set of administrative duties which need to be distributed among faculty regardless of their other involvements. So any model, which tries to balance workloads across domains, be it time-based or otherwise, is bound to fail in one way or another.

For the reasons above, we propose that the faculty workload be broken down into connected, but independent modules. In the resulting approach, each faculty is expected to carry out a certain minimum amount of activities in each of the 3 modules of teaching (formal and informal), research, and administrative services. However, unlike existing models, the workload is balanced in each of these domains individually. This is a simple strategy that is a lot more intuitive and fair, while also promoting the faculty’s development by pushing them into all the domains essential for an academic’s success.
Consequently, we focus primarily on the allocation and balancing of teaching workload, as it is the most variable and controllable component of the overall workload assigned to a faculty. Research and administrative service, while largely fixed in terms of allocation, are used for balancing the workload by impacting teaching. This is discussed in detail in Section 3.3.

3.2 Workload Unit Model

We start with defining a Workload Unit or WLU, which we will use to quantify formal teaching workload in our proposed model. Basically, WLUs are contact hours adjusted using certain ‘weight’ parameters, inspired by the ‘actual hours’ model mentioned in the study by Vardi [3]. The weight parameters cover all the various adjustments that should be considered while comparing different teaching activities. These include the type of teaching activity (lecture, tutorial, and lab), the size of the class, the novelty of the course being taught, etc. By multiplying the corresponding weightage values to the actual contact hours of the teaching activity, we arrive at a standardized quantization of the workload in terms of WLUs. A basic mathematical representation can be made as below.

Let,

\[ T_p \] be the teaching activity type parameter;
\[ S_p \] be the class size parameter;
\[ N_p \] be the novelty parameter;
\[ H \] be the number of hours of the teaching activity per week;
\[ W \] be the number of weeks the teaching activity is carried out;

Then,

\[ \text{Workload (in WLU)} = T_p \times S_p \times N_p \times H \times W \]  
(Eq 3.1)

This simple-yet-comprehensive formula allows for a fair and standardized quantization of the workload for different activities. By accounting for the differences via utilization the parameters, it becomes easier to understand the absolute and relative workloads of faculty members leading to a more equitable workload model.

The weightage values of the respective parameters are left open for the school/department to decide. This may be done in different ways - a democratic consultation with all faculty members involved could be carried out, or a small representative set of faculty members may
decide the values in discussion with the management. Alternatively, one could use historic data to train the values, provided the faculty were reasonably satisfied by previous workload allocations. A set of sample values are detailed below.

<table>
<thead>
<tr>
<th>Parameter Type</th>
<th>Parameter Value</th>
<th>Weightage Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teaching Activity, $T_p$</td>
<td>Lecture</td>
<td>1.50</td>
</tr>
<tr>
<td></td>
<td>Tutorial</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td>Laboratory</td>
<td>1.00</td>
</tr>
<tr>
<td>Class Size, $S_p$</td>
<td>$&gt;300$</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td>151 – 300</td>
<td>1.50</td>
</tr>
<tr>
<td></td>
<td>61 – 150</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td>0 – 60</td>
<td>1.00</td>
</tr>
<tr>
<td>Novelty, $N_p$</td>
<td>New for School</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td>New for Faculty</td>
<td>1.50</td>
</tr>
<tr>
<td></td>
<td>Regular</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 3.1 Sample values for the different weightage parameters used to compute workload.

To illustrate the computation of workload, let us consider lectures for an existing course C1 (previously taught by the faculty) which runs for 2 hours a week for 13 weeks, having a class size of 400.

Then,

$$Workload(C1_{LEC}) = T_{LEC} \times S_{400-} \times N_{REG} \times 2 \times 13$$

$$= 1.50 \times 2.00 \times 1.00 \times 2 \times 13 = 78$$

Therefore,

$$Workload(C1_{LEC}) = 78 \text{ WLU}$$

Continuing with the illustration, let us consider a tutorial for the same course, running 1 hour a week for 13 weeks, with a class size of 40 since tutorials are held in smaller groups.

Then,

$$Workload(C1_{TUT}) = T_{TUT} \times S_{60-} \times N_{REG} \times 1 \times 13$$

$$= 1.25 \times 1.00 \times 1.00 \times 1 \times 13 = 16.25$$

Therefore,

$$Workload(C1_{TUT}) = 16.25 \text{ WLU}$$
As we can see, even though the lecture hours are only twice as many as tutorial hours, the effective workload difference between an hour of lecturing versus an hour of tutoring is ~2.5 times using the WLU model. This is a more accurate and fairer representation of the real-world effort required to undertake these activities.

Now, let us compute the total course workload of C1 assuming it has 10 tutorial groups, in addition to the lectures, and no labs.

Then,

\[
Workload(C1) = Workload(C1_{LEC}) + N_{TUT} \times Workload(C1_{TUT}) + N_{LAB} \times Workload(C1_{LAB})
\]

\[
= 78 + 10 \times 16.25 + 0 \times 0 = 240.5
\]

Where,

\[
N_{TUT} \text{ is the number of tutorial groups;}
\]

\[
N_{LAB} \text{ is the number of lab sessions;}
\]

Therefore,

\[
Workload(C1) = 240.5 \text{ WLU}
\]

Using the above computations, the total teaching workload can be quantified as a summation of individual course workloads.

That is,

\[
Total \ Teaching \ Workload = \sum Workload(Ci)
\]

(Eq 3.2)

It would seem that with this, we have a way to quantify overall teaching workload. However, there is one key consideration that is yet to be made – Part-time Faculty.

3.2.1 Part-Time Faculty and Impact on Workload

Any school or educational institution often engages a certain number of teaching staff on a contractual basis. These may be industry experts holding full-time jobs, or an esteemed faculty at another school, or just other highly specialized individuals. These engagements allow the school to offer specialized courses to its students, without having to deal with full-time costs of hiring individuals. It is then obvious that these ‘part-time’ faculty members affect the teaching workload. This effect needs to be accounted for.
As part-time faculty do not generally engage in research and administrative duties, it is futile to consider them for it. Additionally, while they are involved in teaching, their teaching assignments are pre-determined and part of their contracts – meaning they are adequately informed and compensated for their efforts. Considering all of this, a part-time faculty’s inclusion into the workload allocation system is unwarranted.

However, this does not mean that adjustments to the system are not required. As we do not consider the part-time faculty, we should also not consider their workload. That is, all workload assigned to them must be removed from the calculation of the total teaching workload. Therefore, we re-define the total teaching workload as follows,

\[
\text{Total Teaching Workload} = \sum \text{Workload}(Ci) - \sum \text{Workload}(Cp)
\]

Where,

\[
\sum \text{Workload}(Cp) \text{ is the sum of course-loads taught by part-time faculty}
\]

With the total workload accounted for, it now becomes necessary to quantize the available teaching capacity in terms of faculty, before allocations can be made. This is discussed in the next section.

### 3.3 Teaching Contributions and Relaxations

The teaching and supervision components of the workload are the ones that can be adjusted the most by the administration. For further simplification, we combine the teaching and supervision aspects of the popular STAR workload [21] into teaching itself – with a distinction of ‘formal teaching’ for lectures/tutorials/labs and ‘informal teaching for student supervisions. So effectively, the ‘base teaching contribution’ covers both these aspects.

To quantify the available teaching capacity, we start with assigning every member of the faculty a ‘base teaching contribution’ in the percentage format. The ‘base teaching contribution’ is the approximate share of teaching that a faculty member is expected to undertake. This share can be defined by a number of factors. However, we propose that it reflect the appointment of the faculty for following reasons:
1. Appointments are the clearest indicators used for comparing different faculty members. It is considered almost a given that faculty holding the same position would essentially be assigned the same amount of overall workload.

2. Appointments are made based on certain expectations from the faculty. While Professors are expected to give equal time to research and teaching, lecturers are expected to focus on teaching alone. Similarly, Research Fellows who aspire to be professors soon require experience in teaching, and as a result must contribute more to it.

With the above in mind, an example of the base teaching contribution for faculty is given below.

<table>
<thead>
<tr>
<th>Appointment</th>
<th>Base Teaching Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lecturer, Research Fellow/Scientist</td>
<td>150%</td>
</tr>
<tr>
<td>Assistant/Associate/Full Professor</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 3.2 Sample values for the appointment-based teaching contribution values.

With the base values established, faculty were provided ‘relaxations’ based on their research and administrative/service contributions. These relaxations were calculated separately using the following techniques.

3.3.1 Research Relaxation

Research contributions have been measured in terms of publications, patents and research grants secured in existing literature [21][2][3]. Each of these measurements has their own flaws. With publications and patents there is a question about quantity versus quality, and how to objectively define quality. While with research funding there are variable components in terms of actual fund utilization – a fund that is primarily spent on acquisition of expensive, essential equipment for example would possibly contribute less in terms of research as compared to one that is used primarily on hiring a research team.

We thus propose to compute research relaxation as a percentage value based on the number of research staff employed under each faculty, combined with a set of user-defined parameters. This approach was preferred because the university has its own guidelines governing the ‘key performance indicators’ of individual research staff, controlling the variation in capabilities and research output of the hired staff. As a result, the research team led by different faculty members are maybe not perfect, but good indicators of their respective research contributions.
For the actual computation, a weighted sum of all research staff employed by a faculty was computed and then converted into the percentage-based research relaxation values, as follows

Let,

- \( N_r \) be the number of research staff employed of type ‘r’
- \( W_r \) be the weight parameter for the research staff of type ‘r’
- \( R_p \) be the research relaxation parameter

Then,

\[
\text{Research Relaxation} = R_p \times \sum (N_r \times W_r)
\]

(Eq 3.4)

The value of parameters can vary. An example set of values is given below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight Parameter for Post-Docs</td>
<td>0.5</td>
</tr>
<tr>
<td>Weight Parameter for Research Assistants/Associates</td>
<td>1.0</td>
</tr>
<tr>
<td>Relaxation Parameter</td>
<td>2.5%</td>
</tr>
</tbody>
</table>

Table 3.3 Sample values for the different parameters for research relaxation.

To illustrate with an example, let us compute the research relaxation for Prof. P1 with 2 Post-Docs and 1 Research Assistant employed under him/her.

Then,

\[
\text{Research Relaxation(P1)} = R_p \times \sum (N_r \times W_r)
\]

\[
= 2.5\% \times (N_{PD} \times W_{PD} + N_{RA} \times W_{RA})
\]

\[
= 2.5\% \times (2 \times 0.5 + 1 \times 1.0) = 5.0\%
\]

Therefore,

\[
\text{Research Relaxation(P1)} = 5.0\%
\]

The computed relaxation can now be subtracted from the faculty’s base teaching contribution to arrive at the effective teaching contribution value.

3.3.2 Administrative and Service Relaxation

The administrative or ‘service’ relaxation were slightly trickier to compute owing to the varying nature of the service duties. With the numerous granularities and no fair way to generalize the duties from a workload perspective, a variation of the actual hours model was used. Each service duty was enumerated and assigned a number of hours, as well as a criticality factor.
Considerations were made to ensure that the number of effective service hours did not exceed a set value for each faculty. However, in the limited cases where the hours were exceeded, relaxations were given.

The amount of relaxations can vary, but an example is given below. Note that lump-sum, relaxations can also be given for specific administrative posts.

<table>
<thead>
<tr>
<th>Service Hours</th>
<th>Relaxation</th>
</tr>
</thead>
<tbody>
<tr>
<td>300+</td>
<td>20%</td>
</tr>
<tr>
<td>200-300</td>
<td>10%</td>
</tr>
<tr>
<td>0-200</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 3.4 Examples for Service Relaxation Value (hours)

<table>
<thead>
<tr>
<th>Administrative Appointment</th>
<th>Relaxation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair</td>
<td>100%</td>
</tr>
<tr>
<td>Associate Chair</td>
<td>50%</td>
</tr>
<tr>
<td>Assistant Chair</td>
<td>30%</td>
</tr>
</tbody>
</table>

Table 3.5 Examples for Service Relaxation Value (appointments)

These relaxations, similar to research relaxations, are also subtracted from the base teaching to arrive at the faculty’s effective teaching contribution.

It may be worth noting that we use ‘relaxations’ instead of modularized research and administrative workload components. This is because that while research and administrative duties are essential to a faculty’s career progression, they are not marked as mandatory at the school i.e. it is not necessary for every staff member to be involved in research and/or administration matters.

### 3.4 Calculating Optimal Workload for Individual Faculty

With the WLU model and teaching contribution in place, we can now compute the optimal workload values for individual faculty members. This is computed separately for formal and informal teaching, in line with modular nature of the overall workload policy.
3.4.1 Formal Teaching Workload

To calculate optimal workload, we first calculate the average teaching workload by taking the sum of all weight-adjusted contact hours and dividing it by the sum of effective teaching contribution (i.e. base minus relaxations). Then, for each individual faculty, their optimal workload is computed as the product of their respective teaching contribution and the average workload. This is illustrated by the set of equations below.

\[
\text{Total Teaching Workload} = \sum \text{Workload}(C_i) - \sum \text{Workload}(C_p)
\]

where \(C_i\) is all courses and \(C_p\) is courses taught by part-time faculty

\[
\text{Total Teaching Contribution} = \sum \text{Teaching Contribution}(P_i) - \sum \text{Teaching Relaxation}(P_i)
\]

where \(P_i\) is all full-time faculty

\[
\text{Avg. Teaching Workload} = \frac{\text{Total Teaching Workload}}{\text{Total Teaching Contribution}}
\]

\[
\text{Optimal Workload (Pi)} = \text{Effective Teaching Contribution (Pi)} \times \text{Avg. Teaching Workload}
\]

We can thus compute the amount of workload to be assigned to the respective faculty during the actual allocation process, which will be covered in the next section.

3.4.2 Informal Teaching Workload

The workload distribution for informal teaching is isolated from formal teaching to avoid unnecessary complexities with regards to comparing formal contact hours and informal student supervisions. This is also done because informal teaching is a mandatory requirement for all teaching faculty under the school policy.

The average and individual faculty workloads are computed similar to the method in formal teaching, with the only difference being that workload is measured as number of students and not weighted contact hours. However, weightages exists for different levels of supervision (FYP/M.Sc./Ph.D.) which helps quantitatively compare the informal teaching workload among faculty members. An example of the weightages is given below:
Similar to the rationale behind formal teaching, weightage parameters are introduced to better reflect and compare the differences in effective workload for supervising undergraduate and graduate students.

Graduate students are in most cases personally undertaken by the faculty members on a case-by-case basis. As a result, there is limited flexibility in terms of their assignments. However, the graduate student numbers can be used to adjust the expected workload of a faculty in terms of the number of undergraduate students assigned - the key component of the informal teaching workload.

To compute the number of F.Y.P. students that must be assigned to an individual faculty, we first compute the overall informal teaching workload.

\[
\text{Informal Teaching Workload} = W_{\text{FYP}}N_{\text{FYP}} + W_{\text{MSc}}N_{\text{MSc}} + W_{\text{PhD}}N_{\text{PhD}} \quad \text{(Eq 3.8)}
\]

Where,

- \(N_{\text{FYP}}, N_{\text{MSc}}\) and \(N_{\text{PhD}}\) are the total number of FYP, MSc and PhD students
- \(W_{\text{FYP}}, W_{\text{MSc}}\) and \(W_{\text{PhD}}\) are the supervision weightages for FYP, MSc and PhD respectively

Next, we divide the overall informal teaching workload by the total teaching contribution to get average informal teaching workload. We then adjust the average value by the faculty’s individual teaching contribution levels to get their optimal individual workload. From this, we subtract the weighted sum of MSc and PhD student supervisions to get the optimal FYP supervision load. This number, divided by the FYP weightage parameter gives us the optimal number of FYP students that must be assigned to individual faculty. The values generated in this way are rounded off to the smallest integer value greater than the value itself due to the atomic nature of student supervisions i.e. a student cannot be split between faculties, nor can a project be split between two students. Additionally, the individual FYP numbers may be capped or bumped up to meet the maximum and minimum criteria respectively. To summarize, the

<table>
<thead>
<tr>
<th>Student</th>
<th>Supervision Weightage</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Full-Time</td>
<td>Part-Time</td>
</tr>
<tr>
<td>F.Y.P.</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>M.Sc./M.Eng.</td>
<td>1.25</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>Ph.D.</td>
<td>1.75</td>
<td>0.9</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.8 Weights parameter values for student supervision
informal teaching allocation problem can be limited to the assignment of FYPs based on students’ preferences while adhering to workload limits that are influenced by other supervisory roles undertaken by the faculty for M.Sc. and Ph.D. students.

3.5 Experimental Set-Up and Results

To validate the practicality of the proposed workload model, and the allocation algorithms discussed later, data from a school at a reputable university was used. The school has an academic staff comprising of over 80 full-time faculty members, and offers more than 120 courses in an academic year across 2 semesters to the 1500 students enrolled. With regards to research, the school houses over 200 researchers supervised by the faculty which produce close to 500 publications every year. Additionally, members of the faculty are engaged in over 50 different administrative committees and duties.

By the end of this research exercise, experimental analysis and implementation was carried out on data over 4 full academic years. After analysing the system results for each years, improvements were made based on the feedback received from the school’s management and administration.

3.5.1 Results for Formal Teaching Workload

Using the WLU Model, we quantified the time-based data to determine overall, average and individual teaching workload values for 4 academic years. The values for the various workload parameters were set by the school’s management and remained constant across the academic years (refer table 3.1 for exact values). Based on this, the total formal teaching workload was determined as follows (in WLU):

<table>
<thead>
<tr>
<th>Academic Year</th>
<th>Lec. Workload</th>
<th>Tut. Workload</th>
<th>Lab Workload</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8994</td>
<td>6512</td>
<td>3493</td>
<td>18999</td>
</tr>
<tr>
<td>2</td>
<td>8438</td>
<td>5856</td>
<td>4040</td>
<td>18334</td>
</tr>
<tr>
<td>3</td>
<td>7938</td>
<td>5977</td>
<td>4223</td>
<td>18138</td>
</tr>
<tr>
<td>4</td>
<td>7859</td>
<td>6018</td>
<td>3977</td>
<td>17854</td>
</tr>
</tbody>
</table>

Table 3.9 Formal Teaching Workload for different teaching activities.

Of this, a fixed amount of workload was pre-assigned to part-time/visiting faculty. Subtracting
the pre-assigned workload from the total computed above provided the total effective formal teaching workload.

<table>
<thead>
<tr>
<th>Academic Year</th>
<th>Total Workload</th>
<th>Part-Time/Visiting Faculty Workload</th>
<th>Effective Formal Teaching Workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18999</td>
<td>967</td>
<td>18032</td>
</tr>
<tr>
<td>2</td>
<td>18334</td>
<td>1291</td>
<td>17043</td>
</tr>
<tr>
<td>3</td>
<td>18138</td>
<td>982</td>
<td>17156</td>
</tr>
<tr>
<td>4</td>
<td>17854</td>
<td>844</td>
<td>17010</td>
</tr>
</tbody>
</table>

Table 3.10 Formal Teaching Workload for part-time faculty.

To quantify the teaching capacity across the 4 years, parameter values were again set by school management. The values can be found in Table 3.2 - 3.5.

<table>
<thead>
<tr>
<th>Academic Year</th>
<th>No. of Faculty</th>
<th>Research Staff</th>
<th>Relaxation (FTE)</th>
<th>Total Teaching Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Res. Fellows</td>
<td>Res. Asst/Assoc</td>
<td>Research</td>
</tr>
<tr>
<td>1</td>
<td>84</td>
<td>28</td>
<td>71</td>
<td>2.1</td>
</tr>
<tr>
<td>2</td>
<td>84</td>
<td>61</td>
<td>113</td>
<td>3.6</td>
</tr>
<tr>
<td>3</td>
<td>85</td>
<td>86</td>
<td>104</td>
<td>3.7</td>
</tr>
<tr>
<td>4</td>
<td>80</td>
<td>77</td>
<td>119</td>
<td>3.9</td>
</tr>
</tbody>
</table>

Table 3.11 Teaching contributions and relaxations for full-time faculty.

Based on the combination of the above values, we can compute the average formal teaching workload per faculty, which is the workload that would be assigned to a faculty with a 100% effective teaching contribution after relations.

<table>
<thead>
<tr>
<th>Academic Year</th>
<th>Avg. Lec. WLU</th>
<th>Avg. Tut. WLU</th>
<th>Avg. Lab WLU</th>
<th>Avg. WLU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>73</td>
<td>37</td>
<td>211</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>73</td>
<td>53</td>
<td>227</td>
</tr>
<tr>
<td>3</td>
<td>93</td>
<td>75</td>
<td>54</td>
<td>222</td>
</tr>
<tr>
<td>4</td>
<td>97</td>
<td>79</td>
<td>53</td>
<td>229</td>
</tr>
</tbody>
</table>

Table 3.12 Teaching contributions and relaxations for full-time faculty.

The above workload values can be reversed to estimate the effective teaching hours for the faculty by dividing them by their weightage parameters. For example, Year 4 requires that faculty undertake 64 hours of lectures, 63 hours of tutorials and 53 hours of labs in a year.
3.5.2 Results for Informal Teaching Workload

The informal teaching allocation was validated across two academic years, twice a year for a total of 4 sets of data. This was because the informal teaching problem was not attempted until the 2nd academic year, and data for the 4th academic year was unavailable at the time of writing of this thesis. The workload was computed based on the date on number of students and the weightage parameters as set by the school management (refer table 3.8 for exact values).

<table>
<thead>
<tr>
<th>Academic Year</th>
<th>Semester</th>
<th>No. of Post-Graduate Student Supervisions</th>
<th>F.Y.P.</th>
<th>Total</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>M.Eng/M.Sc</td>
<td>Ph.D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>347</td>
<td>24</td>
<td>40</td>
<td>3.0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>308</td>
<td>20</td>
<td>306</td>
<td>6.3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>308</td>
<td>20</td>
<td>59</td>
<td>5.3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>275</td>
<td>18</td>
<td>290</td>
<td>5.1</td>
</tr>
</tbody>
</table>

Table 3.13 Teaching contributions and relaxations for full-time faculty.

The workload values computed above would be used later in Chapter 4 and 5 during for the allocation process.

3.6 Summary

In this chapter, we began with understanding the four major components of faculty workload – supervision, teaching, administration and research – and how they impacted one another. Research and administration activities are largely inflexible from the management point of view as their corresponding ‘assignments’ are fixed for the most part. The teaching and supervision activities can however be controlled to a much greater degree, and can be used to adjust the overall faculty workload.

The workload model is the quantitative framework which serves as the foundation of the overall teaching allocation system. It defines how the various activities must be quantified keeping in mind the numerous variables and constraints which impact the effective workload of each activity. We proposed the Workload Unit Model (WLU) for this purpose, which modifies contact hours for teaching duties using weightage parameters derived from corresponding workload constraints. Using the model, we can quantify the total teaching
workload that needs to be distributed among faculty, after discounting for part-time staff. The WLU model also quantifies the teaching capacity of individual faculty members. It does so by assigning a base teaching contribution to the faculty based off their appointment, and then allowing for relaxations based on other activities they might be involved in. In the WLU model, we allow two types of relaxation – research and service. Research relaxations are based off the number of research staff employed by the faculty, whereas service relaxations are based on administrative duties like being the Chair/Associate Chair of the school.

The quantification of teaching workload and capacity allows for the computation of optimal workload values for each faculty. These are based on the global average workload values, which are further adjusted based on the net teaching contribution of each faculty member. With the amount of workload to be assigned to each faculty now being known, we can proceed to the actual allocation strategies for the said assignment.
4. FORMAL TEACHING ALLOCATION USING GREEDY TECHNIQUES

With the workload model in place, the next step is to design an allocation algorithm that aligns with the objectives of having a fair and equitable distribution of the said workload. In this section, we propose greedy based approaches to the problem of allocating and timetabling formal teaching workload. Formal teaching, as described earlier, comprises of the lecture, tutorials and labs for the courses offered by the school.

We begin with an overall description of the necessary features. In the subsequent sections, we first discuss the allocation problem and its proposed solution. This is followed up with a similar discussion on the problem of timetabling. Finally, we discuss the results of the proposed algorithms to demonstrate the viability of the solutions.

4.1 Features of the Allocation System

The teaching allocation system is a complex one that can be broken down into two major components. First, is the actual allocation of workload to faculty which deals with the decision making process of choosing courses and activities that are to be assigned to each member of the faculty. Second, is timetabling which deals with the problem of scheduling the assigned workload in a conflict free, student and faculty friendly manner.

4.1.1 Allocating Formal Teaching Workload

As mentioned in section 2.2, workload allocation strategies proposed across the research literature were found to be lacking in one way or another. Additionally, in practice we observed that a number of real-world constraints were not represented. To summarize, the algorithm designed for the formal teaching allocation must have the following features:

1. It must distribute workload to the faculty in a fair and equitable, respecting their preferences and capabilities.
2. It must utilize the pool of faculty efficiently, avoiding under/over utilization of the human resource from an administrative point of view.
3. It must make due considerations towards students, by incorporating their feedback for the faculty and prioritizing core courses.

4. It must be able to handle dynamic changes, like addition/removal of courses and faculty movements, in a non-disruptive manner.

5. Lastly, the algorithm must be as simple so as to allow greater transparency and acceptance among the various stakeholders involved.

4.1.2 Timetabling

The university course timetabling problem is the problem of scheduling faculty-course assignments in a conflict-free manner across multiple venues and time-slots. Beyond this basic requirement, a timetabling system must ideally possess the following features:

1. It must be faculty-friendly, reasonably spreading the assigned teaching activities across the week. The system must accommodate faculty preferences as far as possible, and must balance the spread to avoid extreme cases like assigning teaching duties on all five days or stacking all assignments into a single day.

2. It must also follow a student-centric approach, ensuring that the classes are spread out across the week and students receive reasonably-sized breaks between classes.

3. It should efficiently utilize infrastructural resources like labs and lecture halls by matching venue capacity with class-size

4. It should generate timetables that have an element of flexibility to schedule make-up or additional classes that were not part of the original teaching schedule

4.2 Problem formulation for Allocation

Formal teaching comprises of lectures, tutorials and lab sessions. As mentioned in Chapter 3, each of these activities have a weighted parameter which is a better reflection of the real-world workload and allows for a fairer comparison and distribution. The workload associated with each activity assignment can thus be computed individually. Additionally, the workload capacity for each individual faculty can also be worked out as shown in section 3.4. The problem can then be modelled in its most trivial form as a job-shop scheduling problem (JSP). It can be defined as:
We are given \( n \) jobs (teaching activities) \( J_1, J_2, ..., J_n \) of varying processing times (workload), which need to be scheduled (assigned) on \( m \) machines (faculty members) with varying processing power (teaching contribution), while trying to minimize the makespan (overload). The makespan is the total length of the schedule (workload distribution).

This can be represented mathematically as:

Let \( P \) and \( C \) be the set of \( n \) faculty members and \( m \) teaching activities, respectively. Let \( P_i \) represent the teaching contribution (capacity) for faculty \( i \), and \( C_j \) denoting the workload associated with activity \( j \). Additionally, let \( X \) be an \( m \)-by-\( n \) binary matrix, with \( X_{ij} \) denoting if faculty \( P_i \) is assigned teaching activity \( C_j \). Then, our objective is

\[
\text{Minimize } \sum_{i=0}^{m} X_{ij} C_j / P_i, \text{ for all } i
\]

\[
\text{Subject to } \sum_{i=0}^{n} X_{ij} = 1, \text{ for all } j
\]

While the above JSP representation covers the workload aspect of the problem, it leaves out some of the other features discussed in section 4.1. In its current state the problem does not completely model the real world scenario - the school does not wish to simply allocate all courses to faculty with workload being the only criteria for allocation. There are two other considerations as previously noted – faculty preference and student satisfaction. These are incorporated using a combined function we called the assignment feasibility index (F.I.).

### 4.2.1 Assignment Feasibility Index

The assignment feasibility index (F.I.) is defined as an indicator of optimality for faculty-course assignment. In its most basic form, it is computed as a weighted sum of the faculty’s ‘preference’ for the course and the inverted student feedback score for the faculty we termed as ‘performance’. Faculty preferences were sourced directly from the faculty using an online form where they were asked to rank courses they would like to teach, with separate forms for each of the three teaching activities of lecture, tutorial and lab.

The performance scores on the other hand were calculated using a two-step process. First, we normalized the absolute student feedback scores (received on a scale of 1-5) into a percentile-based discretised scale. These discretized values were then inverted on a ten-point integer scale, meaning the 90th percentile and above feedback scores equated to a value of 1, 80th-90th
percentile equated to 2, and so on. Consequently, lower the value of the F.I. (high preference, better feedback), lower would be the cost of assignment.

\[ \text{Feasibility Index (Pi, Cj)} = \text{Performance (Pi, Cj)} + \text{Preference (Pi, Cj)} \]  

(Eq 4.1)

where \( Pi \) and \( Cj \) refers to individual faculty \( i \) and course \( j \)

Using this, we can now incorporate an additional objective function of optimizing assignments by minimizing FI. This introduces a ‘profit’ metric which is inversely proportional to the FI for each assignment. With these considerations, we transform the earlier problem from a job-shop scheduling problem to a ‘generalized assignment problem’, which is defined as:

There are a number of agents (faculty) and a number of tasks (teaching activities). Any agent can be assigned to perform any task, incurring some cost (workload) and profit (assignment feasibility) that may vary depending on the agent-task assignment. Moreover, each agent has a budget (teaching contribution) and the sum of the costs (overall faculty workload) of tasks assigned to it cannot exceed this budget. It is required to find an assignment in which all agents do not exceed their budget and total profit of the assignment is maximized.

The resulting problem formulation cannot be simplified into any existing combinatorial optimization problem with a proven solution strategy. To devise a solution for the same, the computational complexity of the problem needed to be examined.

### 4.2.2 Computational Complexity Analysis

The computational complexity analysis of a problem can provide insights into possible algorithms and techniques that can be used to solve it. The JSP approximation of the workload only problem is a generalization of the travelling sales-man problem (TSP). TSP has been proven to be a non-deterministic polynomial time hard, or NP-hard problem. The generalized assignment problem used to formulate the complete teaching allocation problem is even more complex, as it has been proven to be an NP-complete problem.

The NP-completeness of the problem implies that an optimal solution for the teaching allocation problem cannot be guaranteed in polynomial time. Being NP-complete, the time required to solve the problem would in fact grow exponentially with the size of the input data. Considering the large input data set (faculty, courses, activity types) and its exponential impact
on the time-complexity of the problem, there exists a need to devise a solution that is fast and computationally inexpensive. This can be achieved by utilizing one (or more) of the following techniques:

- **Approximation** – Finding a near-optimal solution, instead of an optimal one.
- **Randomization** – Using randomness to generate solutions, while actively controlling the randomization function so that the solution moves towards optimality.
- **Restriction** – Limiting the structure of the input data to lower computational complexity.
- **Parameterization** – Fixing the problem’s input parameters.
- **Heuristics** – Problem-tailored techniques to achieve ‘good enough’ results

In the next section, we discuss the greedy algorithm – a technique used to approximate solutions to the generalized assignment problem. We shall also discuss the advantages and disadvantage of using the greedy algorithm, and how it can be used to solve the formal teaching allocation problem.

### 4.3 The Greedy Algorithm

The greedy algorithm is a simple algorithm which given a heuristic, sequentially solves the problem by making locally optimal choices with the hopes of finding a globally optimal solution. We can illustrate the algorithm with a simple implementation for the shortest path problem described below.

![Fig 4.1 Shortest-Path example problem.](image-url)

The shortest-path problem is to devise a path to get from node O to node D while travelling the minimum distance. The path must traverse through the intermediary nodes in a manner directed
by the graph edges, with the weights denoted above them being the path distance between nodes. At every node, a greedy approach would choose the next node that is closest to it. For the above problem, it would devise a solution as follows:

![Fig 4.2 Shortest-Path example with optimal solution.](image)

The solution O-A-D with path distance of 3 happens to be optimal i.e. shortest distance in this case owing to the structure of the problem. However, if the problem was structured differently this would not be the case as can be illustrated below.

![Figure 4.3 Shortest-Path example with sub-optimal solution.](image)

The solution O-A-D with path distance of 8 in this case is not the optimal solution, as the path O-B-D has a path distance of 7. This is the key short-coming of the greedy algorithm – its inability to provide optimal results due to its ‘short-sighted’ and ‘non-recoverable’ nature. To produces optimal results, the greedy algorithm requires the problem to that have two key features. First, a greedy choice property using which a decision can be made at every step. Second, an existence of optimal substructures which implies than an optimal solution to a problem contains optimal solutions to its sub-problems.
However, the greedy algorithm offers multiple advantages, especially over other alternatives like Genetic Algorithms and Linear programming. It is simple to implement, and as a result simple to explain in an organizational setting like a university. It allows for smaller set of parameters (< 10) that can be directly co-related to real-world constraints. The greedy algorithm is also more efficient in terms of resource utilization and can provide results much faster than alternative exhaustive search algorithms owing to its lower time complexity O(n). Based on these considerations, the greedy algorithm was chosen as the starting point for designing the algorithm for allocating formal teaching workload.

4.4 Implementing Greedy for Formal Teaching Allocation

For the general assignment problem, the greedy algorithm would make for a poor approximation considering the lack of heuristics or greedy choice property. In a situation where all agents and tasks are of equal priority, the random ordering of the assignments would lead to sub-optimal results. However, for the specialized case of formal teaching allocation there exists an additional feature – workload priority.

4.4.1 Workload Priority

The need to give priorities to different activities/courses implies that some activities/courses are more important than others, and hence must be assigned first. This makes for a great priority based heuristic, ensuring that locally optimal assignment decisions lead to a globally optimal teaching allocation. The priorities to assignments were designated on two levels.

1. **Teaching Activity Type** – Lectures, tutorials and labs were prioritized in decreasing order based on the rationale that lectures formed the foundation of a course, and their teaching quality directly impacted the students’ understanding of the subject as well as their performance in the tutorial and lab sessions. Tutorials were similarly prioritized over labs, as they provided greater opportunities for interacting with the faculty.

2. **Academic Year** - Lower year courses were prioritized over more advanced and specialized courses. This was done because lower-year courses provide the necessary foundation that served as pre-requisites for later-year courses and electives. Additionally, lower-year courses have larger class sizes and prioritizing them maximizes the quality of the teaching allocation for a larger number of students.
Allocations can thus be made with the objective of minimizing the sum of feasibility indices for all assignments while utilizing workload priorities as the heuristic.

### 4.4.2 Naïve Greedy Implementation

The naïve greedy implementation assigned individual teaching activities sequentially in order of their given priority. For each assignment, it considered all faculty members that preferred the course/activity type and their corresponding performance scores by assessing the respective feasibility indices. The choice of assignment was based on picking the faculty assignment which had the minimum FI score.

Additionally, after each assignment was made, the assigned faculty was checked to see if they had reached their optimal workload values. If they had, the faculty was removed from the allocation pool. If no faculty could be found for an assignment due to the relevant faculty members being overloaded, the teaching activity was marked as unassigned. The basic algorithm is illustrated below.

```plaintext
procedure FORMAL_TEACHING_ALLOCATION
begin

1. for all courses
   assign course priority based off their academic year;
2. initialize teaching activity = lectures/tutorials/labs, allocation pool = all faculty;
3. while (there exist untraversed courses) {
4.   pick highest priority course that is yet to be traversed;
5.   for all faculty that have preferred the course and are still in allocation pool
       compute FI based on preference rank and performance scores;
6.   pick faculty with lowest FI score (most feasible);
7.   if (no faculty available)
       mark as unassigned;
8.   else {
       assign course to faculty;
       update faculty workload;
       if (faculty workload > optimal workload)
           remove faculty from allocation pool;
   } mark course as traversed;
} mark course as traversed;
end
```

Figure 4.4 Pseudo-code for the naïve greedy-based formal teaching allocation algorithm.

54
The above naïve implementation provided a solution which adhered to the principles of equity and fairness in workload distribution but had two major shortcomings. First, the overall FI scores dropped considerably as we moved to lower priority courses. Second, the solution was often ‘incomplete’ owing to certain courses not being assigned because the faculty who preferred to teach them had already been assigned to other courses and met or exceeded their optimal workload values. Also, certain teaching activities were left unassigned due to the atomic nature of the workload – while multiple faculty had a small portion of their capacities available, none could take the entire activity and neither could the activity be broken down into smaller parts.

4.4.3 Improved Greedy Implementation

Upon closer examination of the naïve implementation results, it was found that the issues of feasibility-drop and incompleteness occurred largely due to the hard checks on faculty workload violation. To address this, we initially experimented by replacing the hard constraint on workload with a soft, flexible limit using a user defined ‘workload variation’ parameter. Users could now define an acceptable variation level for individual faculty workload, and the system would accordingly allow over/under loading the faculty by the chosen percentage. For e.g., a value of 20% would mean that a faculty can be allocated workload up until it hits 120% of its optimal value, only after which it would be removed from the allocation pool. This replacement effectively increased the teaching capacity by making over-loads acceptable to a degree. As a result, feasibility indices improved and the number of non-assignments went down, though they still continued to exist.

Considering the nature of the teaching allocation problem where non-assignments can simply not be allowed, an effort was made to focus on ensuring ‘completeness’ of the solution above all else. With this in mind, the greedy algorithm was altered and even the soft, flexible limits were instead replaced with additional dynamic costs to assignments.

The modified algorithm leveraged on the naïve implementation’s objective function of minimizing the FI values for all assignments. Instead of removing faculty from the allocation pool once their optimal workload values were reached, the modified algorithm significantly increased the FI cost of all subsequent assignments for the said faculty. This added a ‘Faculty Load’ metric to the FI equation (Eq 4.1).
Feasibility Index \((P_i, C_i)\) = Performance \((P_i, C_i)\) + Preference \((P_i, C_i)\) + Faculty Load \((P_i)\) (Eq 4.2)

Where,

\[
\text{Faculty Load} (P_i) \begin{cases} 
0, & \text{if Assigned Workload} (P_i) < \text{Ideal Workload} (P_i) \\
10, & \text{if Assigned Workload} (P_i) \geq \text{Ideal Workload} (P_i)
\end{cases}
\]

The addition of the faculty load metric meant that while an over-loaded faculty would effectively never be considered for an assignment as long as other alternatives existed, it would still continue to be an option-of-last-resort instead of marking a course as unassigned. The FI score now not only represented faculty preference and student feedback, but also reflected the impact of workload distribution for any given faculty-course assignment. This eliminated the case of ‘non-assignments’, while still ensuring reasonable FI scores (preference and feedback) and workload distribution. The resulting improved algorithm can be expressed as shown on the following page.

```plaintext
procedure IMPROVED_FORMAL_TEACHING_ALLOCATION
begin
1. for all courses
   assign course priority based off their academic year;
2. initialize teaching activity = lectures/tutorials/labs;
3. while (there exist untraversed courses) {
4.   pick highest priority course that is yet to be traversed;
5.   for all faculty that have preferred the course and are still in allocation pool
       compute FI based on preference rank, performance scores and faculty load;
6.   pick faculty with lowest FI score (most feasible);
7.   assign course to faculty;
8.   update faculty workload;
9.   mark course as traversed;
}
end
```

Figure 4.5 Pseudo-code for the improved greedy-based formal teaching allocation algorithm.

The improved-greedy algorithm thus ensured a complete solution, albeit with a slightly skewed workload distribution. However, more importantly, the FI manipulation technique used for workload violations allowed for a way to incorporate other dynamic constraints. For e.g., the number of new courses assigned to a faculty could be limited by adding weights to subsequent FI computations once the faculty’s quota was met. Similarly, for annual allocations across two semesters the spread of workload could be controlled. Thus, any additional limitation that needed to be introduced could be incorporated as a soft-constraint.
4.4.4 Secondary Pass

The primary allocation algorithm was limited in its ability to forecast future allocations. Consequently, minor imperfections like increased preparation time as a result of assignment of single tutorials of two different courses preferred equally instead of two tutorials of a single course were present in the results. No matter the constraints and variables introduced into the objective function, these imperfections were almost impossible to fix without compromising on the overall allocation quality. To avoid this compromise, a secondary iteration was added to the algorithm to check for these issues once the primary allocation was complete.

In this secondary pass, the system looked at the overall allocations and looked to swap certain allocations in a bid to achieve better real-world reasonability. This would lead to over-ruling the ‘priority’ heuristic to an extent, but this was deemed acceptable as the heuristic was a soft-constraint to begin with. Some of the issues that were fixed as a result of this secondary pass were the reduction in number of different courses assigned to the faculty, consistency in lecture-tutorial-lab allocations and better load balance across semesters.

After some additional fine-tuning the algorithmic parameters, a solution deemed ‘acceptable’ by the school could finally be computed. While ‘acceptable’, feedback received on the solution found it lacking in enforcing certain real-world sensibilities and constraints. This necessitated a re-examination of the problem constraints followed by another round of refinements to the algorithm. These constraints and refinements are discussed in the next sub-section.

4.5 Addressing Real-World Allocation Concerns

The formal teaching allocation solution computed by the improved greedy algorithm met all the pre-conditions and constraints as detailed in Section 4.2. However, there existed a set of non-intuitive, real world concerns that were not addressed by the solution in its current state.

4.5.1 Change Management

The manual allocation process used previously was based on a minimum-change policy. The course assignments to faculty would remain largely unchanged from year-to-year so as to minimize disruptions and the resulting workload for the administrators. Changes to the
allocation were few in number, and were mostly limited to addressing faculty movements and variations in course offerings.

With the implementation of the bias-free automated teaching allocation system, the paradigm changed resulting in a major re-organization of faculty-course assignments. This led to a number of faculty having to move away from courses that they had taught for multiple years at a stretch, and instead take up new courses. The resulting disruption was deemed difficult to manage in the course of a single year, and the administration requested that changes to the allocation be made in a phased-manner instead. Based on the discussion and feedback received, it was decided that an additional constraint for ‘new courses’ would be added.

A ‘new course’ was defined as a course that a faculty had not previously taught at the school. The constraint restricted the number of ‘new course’ assignments made to a faculty to 1 per semester. It was implemented by adding a ‘new’ parameter to the feasibility index (Eq 4.2).

\[
\text{Feasibility Index} (P_i, C_i) = \text{Performance} (P_i, C_i) + \text{Preference} (P_i, C_i) + \text{Faculty Load} (P_i) + \text{New} (P_i, C_i)
\]

\[
\text{Where,}
\]

\[
\text{New} (P_i, C_i) \begin{cases} 
0, & \text{if } P_i \text{ has not been assigned a new course OR } C_i \text{ is not a new course} \\
10, & \text{if } P_i \text{ has been assigned a new course AND } C_i \text{ is a new course}
\end{cases}
\]

The check for whether a course was new to the faculty or not was done based on previous feedback scores for the faculty. If a feedback score record for the course existed, it meant that the faculty had taught the particular course before. If not, it was a new course.

Additionally, concerns were also raised regarding the number of different courses assigned to a faculty member. It was preferred that a faculty should not be engaged in more than 5 different courses in an academic year. To implement this, a ‘Number of Courses’ parameter was added to the feasibility index equation.

\[
\text{Feasibility Index} (P_i, C_i) = \text{Performance} (P_i, C_i) + \text{Preference} (P_i, C_i) + \text{Faculty Load} (P_i) + \text{New} (P_i, C_i) + \text{Number of Courses} (P_i)
\]

\[
\text{Where,}
\]

\[
\text{Number of Courses} (P_i) \begin{cases} 
0, & \text{if } P_i \text{ has been assigned } < 4 \text{ courses} \\
10, & \text{if } P_i \text{ has been assigned } \geq 4 \text{ courses}
\end{cases}
\]
4.5.2 Impact of Course Priority

The course priority heuristic is used by the allocation algorithm to prioritize certain courses over other when computing the order of assignments. For the most part, this works as intended. However, corner cases were discovered where the strict ordering based on course priority led to undesirable results. This happened where the greedy nature of the algorithm severely impacted the quality of allocation for later assignments. Typical scenarios included assigning faculty that were better suited to lower priority courses to high priority courses as their FI scores were marginally better than the alternative. In fact, the alternative would actually have a much lower score for the lower priority course but would be assigned the course due to workload constraints. Consequently, marginal improvement for the high priority course cascaded into major quality drops for the lower priority courses in manner that defeated practical reasonability.

To address this issue, the secondary pass of the improved greedy algorithm was expanded to include assignments with high base FI score. The secondary pass would now check for alternatives to ‘poor’ assignments, and perform swaps where overall gains in preference/performance scores were substantially greater than the corresponding local losses for the higher priority assignments. To keep the secondary pass constrained to a finite run-time, only one-to-one swaps were allowed.

4.6 Problem Formulation for Timetabling

In section 2.4.1, we defined the university course timetabling problem and its two corresponding set of constraints – hard and soft. The hard constraints included the physical limitations of a student/faculty to be at one course at a time, as well as the venue being large enough and assigned to only one course at any given point in time. The soft constraints focussed on improving quality of spread by minimizing continuous classes and number of teaching days, while simultaneously minimizing the length of breaks within classes on the same day. The features listed in the preceding section also add to the list of soft constraints. The objective of the timetabling exercise is thus to construct a timetable that satisfies the hard constraints, and minimizes the violation of the soft constraints. Using the works of De Werra [37], a mathematical representation of the university timetabling problem can be constructed as described next.
Let there exist $n$ courses $C_1, C_2, ..., C_n$, with each course $C_i$ consisting of $c_i$ classes (lectures/tutorials/labs) for all $i$. There exist $m$ sets of clashing-courses $S_1, S_2, ..., S_m$ that cannot be assigned at the same time due to overlapping students or teachers. Let there be $p$ time-periods or slots across a typical work week, and let the number of venues available at time $k$ be $v_k$. This limits the number of classes scheduled in parallel at a given point in time to be less than or equal to $v_p$. Then, the formulation is to find $y_{ik}$ ($i = 1...n, k = 1...p$), such that

$$\sum_{k=1}^{p} y_{ik} = c_i \quad (i = 1...n)$$

$$\sum_{t=1}^{q} y_{it} \leq v_k \quad (k = 1...p)$$

$$\sum_{t \in S_t} y_{it} \leq l \quad (t = 1...r, k = 1...p)$$

While maximizing the objective function,

$$\sum_{t=1}^{n} \sum_{k=1}^{p} d_{ik} y_{ik}$$

Where, $y_{ik}$ is a binary value set to 1 if class $c_i$ is assigned to time period $p_k$, or else is set to zero. And $d_{ik}$ is the desirability index of assigning class $c_i$ to time $p_k$, a measure of how optimal the assignment is. Since there is no readily available metric that can be used for the desirability index, it is necessary to construct one.

### 4.6.1 Desirability Index

For a given partial timetable and a yet to-be-assigned class, there will always exist a set of timeslots and/or venues which are more desirable than others. For example, a lecture-hall which is the exact capacity as the class to be scheduled is preferable over a lecture-hall that is twice as large. Or, a starting time one hour after the last class taken by the same professor is preferable that assigning non-stop classes. Such ‘preferences’ when it comes to scheduling options for a class must be computed objectively. For this purpose, we introduce the desirability index – a timetabling equivalent of the feasibility index (section 4.2.1)

The desirability index is effectively a reverse cost-function for every class-time assignment, with the value being determined using the soft-constraints of the timetabling problem. For the purpose of this research exercise, we define it as a sum of soft-constraint violations:

$$\text{Desirability Index (DI)} = W_{C.C\text{ontinuous}} + W_{F.H\text{Faculty}} + W_{S.H\text{Student}} + W_{V.C\text{Venue}} + W_{P.C\text{Packed}} \quad (\text{Eq 4.5})$$
For a given class S to be scheduled from time T1 to T2 at venue V,

- **C\textsubscript{Continuous}** is the number of clashing-course classes (i.e. classes taught by faculty teaching S or students attending S) that end at T1 or start at T2. It is a check for evaluating if scheduling the class will lead to continuous sessions for students and/or faculty.
- **H\textsubscript{Faculty}** is a step function for the number of hours of classes scheduled in the same day for the faculty teaching the class. Value is set to 0 as default, but changes to 10 if the maximum number of hours as defined by the administration is reached to reflect faculty-friendliness.
- **H\textsubscript{Student},** similar to **H\textsubscript{Faculty},** is a step function for the number of hours of classes scheduled in the same day for the students attending the class. Default value is 0, but changes to 10 if the maximum number of hours exceeds the limit set. It is a measure of student-friendliness.
- **C\textsubscript{Venue}** is the multiple by which the venue capacity exceeds the size of the to-be-scheduled class. It is a measure of the efficiency of utilization for a given venue.
- **C\textsubscript{Packed}** is the number of non-clashing classes scheduled or ‘packed’ at the same time, multiplied by -1. It is used as a packing measure to incentivize scheduling unrelated classes in parallel so as to keep as many options open as possible for later classes.
- **W\textsubscript{C}, W\textsubscript{F}, W\textsubscript{S}, W\textsubscript{V} and W\textsubscript{P}** are weightage parameters, which can be tuned based on the need to focus-on or relax the respective soft-constraints.

### 4.6.2 Computational Complexity

The resulting problem is one that is NP-complete, meaning it cannot be solved optimally in polynomial time using well-defined techniques. The NP-complete nature of the timetabling problem has been theoretically proven in the works by Cooper and Kingston [51]. As noted by Schaerf [40], for a practical scenario where the number of courses is greater than 10, a heuristic based method must be used.

For the university course timetabling problem, the optimality of the solution is not a critical feature. As long as the solution is feasible and reasonably accommodative of faculty and student needs – it is acceptable. This consideration allows us to use a relaxed approach in way of the iterated greedy algorithm, which is explained in the next section.
4.7 Iterated Greedy Algorithm

We first used the greedy algorithm in Section 4.3 for the workload allocation problem. The iterated greedy algorithm is an extension of the simple greedy algorithm that allows for better results at the cost of increased complexity. The use of the iterated greedy algorithm for scheduling proposed by Ruiz et al [52].

The iterated greedy algorithm at its core performs multiple partial-runs of the greedy algorithm by selectively altering the results at the end of each run. It has two main phases – destruction and construction, which run iteratively on the greedy solutions. The destruction phase takes the complete or ‘candidate’ solution at the end of the first iteration, and removes certain components of it. The construction phases then re-constructs a new complete solution by running the greedy algorithm on the removed components. The newly constructed solution is compared with the solution from the previous iteration. If it is better, the new solution becomes the candidate solution, else the candidate remains unchanged. This is done iteratively until an acceptance criteria is met or a fixed number of iterations are complete. Formally, it can be represented by a five-step process:

Step 1 – Compute initial greedy solution which will be the candidate solution
Step 2 – Destruction – Remove a fixed number of assignments made in the candidate solution
Step 3 – Construction – Re-introduce the removed assignments using greedy
Step 4 – If new solution is better than candidate solution, make new solution the candidate solution
Step 5 – If solution meets criteria for acceptance or number of iterations, terminate. Else, go back to Step 2.

The algorithm can be illustrated with an example – Consider a problem of finding the minimum number of schools which cover a set of eleven towns. The only limitations are that a school must be in a town, and that every school covers all towns which are within 30 miles of it. Let the towns be distributed as denoted by the graph on the following page, with each node denoting a town and each connected edge indicating a distance of less than 30 miles between the towns.
We use a greedy heuristic which assigns a school to the town which has the most number of uncovered towns within the 30-mile distance. This gives us the solution which assigns 4 schools to towns a, c, j and f.

Using iterated greedy, we remove schools in towns a and f, and re-apply the greedy heuristic. Town e now becomes the preferred choice by the algorithm, and the solution is now complete with only 3 schools at e, c and j.
As can be seen, the iterated greedy algorithm can improve simple greedy results by repeatedly performing the destruction and construction phase. It is simple to implement and explain, which makes it the suitable for an organization set up. Additionally, it offers solutions which are closer to optimal than those offered by naïve greedy implementations at the cost of slightly increased time complexity. For the UCTP, where optimality is not a critical constraint, a near-optimal solution is deemed acceptable if processed in reasonable time. Lastly with regards to this research exercise, the prior greedy implementation affords a level of familiarity with the nuances of the algorithm making it a preferred choice.

4.8 Solving UCTP using Iterated Greedy

The greedy algorithm executes based on the assumption that making locally optimal decisions in a sequential order can lead to a globally optimal solution. In case of the workload allocation problem, this holds true due to the presence of the ‘course priority’ heuristic, where the priority of a course determined the order in which assignments would be made, with each assignment being made in a locally optimal manner. However, for the timetabling problem, no such heuristic exists. While a random heuristic could be used as default, constructing a heuristic which guides the greedy algorithm towards optimality is preferred.

4.8.1 Class-Clash Heuristic

In the absence of a heuristic, the greedy algorithm would process the scheduling of classes in a random order. The locally-optimal decision making policy of the algorithm in this case would fail to produce a globally-optimal solution in most cases. As mentioned earlier, the optimality of the constructed timetable is not critical as long as it is feasible and ‘good enough’ in terms of minimizing the violation of soft constraints. We therefore only need a heuristic that can approximate the results of the greedy implementation in a manner that is ‘good enough’, and guide it towards optimality.

This lead to the development of the class-clash heuristic. A ‘clash’ is said to occur when there exist two faculty-course assignments (or classes) such that they cannot be scheduled together. It happens when the two classes are either – a) to be taught by the same faculty or b) to be attended by an overlapping set of students. The class-clash heuristic for a given faculty-course assignment is thus the number of classes that cannot be scheduled in parallel to it.
By processing courses in decreasing order of their class-clashes, the greedy algorithm will process the hardest classes to schedule first. This allows the system to have a greater range of options when assigning highly clash-constrained courses, improving the quality of the scheduled slot and simultaneously reducing the probability of a non-assignments.

4.8.2 Iterated Greedy Implementation

With the course-clash heuristic and the desirability index defined earlier, the timetabling problem can be readily solved using the greedy algorithm. We design the algorithm to process classes in decreasing order of clashes. For each class, a feasibility check is first conducted to identify time-slots where there are no clashing classes and a venue with sufficient capacity/facility is present. Next, we compute the desirability index for each of the feasible time-slots. The time-slot with best value is chosen in line with greedy approach to make the assignment. The venue is also chosen in a greedy manner by selecting the smallest available venue that can accommodate the class. Once all classes have been considered, a simple greedy solution is found, which serves as the initial candidate solution for the iterated greedy approach.

As mentioned in the previous section, the iterated greedy algorithm consists of two phases that are iterated post the generation of the candidate solution – destruction and construction. In the destruction phase, we remove some components of the candidate solutions whereas in the construction phase, we re-construct another solution by performing greedy algorithm on the removed components. For the timetabling problem, we design the destruction and construction phases as follows:

- **Destruction** - Remove n percent of all scheduled classes, where n is a user-defined value in the range 0-100. The classes that are to be removed can be chosen randomly if n is large enough. But for smaller values of n (<10), we propose a lowest-DI heuristic to choose the classes that have the highest number of soft-constraint violations.
- **Construction** – Re-insert the removed classes by performing the greedy algorithm on them. For a large enough value of n, the greedy construction can use a random execution order to determine the sequence in which classes are to be re-scheduled. For smaller values of n (<10), a modified class-clash heuristic which counts the number of clashing classes that are already scheduled in the partial solution is used.
For the replacement criteria, which checks if the solution achieved at the end of an iteration should replace the candidate solution, we use a simple comparison of the average DI score of all scheduled classes. The solution with the better score is chosen as the candidate solution for future iterations.

The stopping criteria is controlled by two user-defined values. First, is the processing time which terminates the algorithm once it executes for a set amount of time. Second, is the target average DI score that also leads to termination if the candidate solution has a score that is equal or better than the user-defined score. The following pseudo-code illustrates the algorithm:

```
procedure ITERATED_GREEDY
begin
1. \( C = \) set of all classes that need to be scheduled;
2. \( CA = SIMPLE\_GREEDY(C) \);
3. while(acceptance criteria is not met) {
4. \( DP = DESTRUCTION\_PHASE(CA, n) \);
5. \( D = CA – DP \);
6. \( CP = CONSTRUCTION\_PHASE(DP, D, n) \);
7. if(CP is better than CA) \( CA = CP \);
8. } if (faculty limits are not violated in M){
9. final_solution = CA;
end
procedure SIMPLE\_GREEDY(C, H = default course-clash heuristic);
begin
11. while(there exists untraversed classes)
12. {
13. pick class in C with highest H value;
14. find feasible time-slot T with best DI value;
15. find smallest available venue V with capacity >= class size of C;
16. schedule C for time T at venue V;
17. mark C as traversed;
18. }
end
procedure DESTRUCTION\_PHASE(S, n);
begin
19. if(n>10) \( D = S – (\text{random n}/100 \text{ scheduled classes}) \); 
20. else if(N<10)
```
For the above implemented iterative greedy solution, the complexity is \( O(n^2 m) \) where \( n \) is number of classes and \( m \) is the number of timeslots in a week. The complexity in this case is manageable, but can cause processing times to be considerably long for bigger timetables.

4.8.3 Round-Robin Scheduling

To improve the average-case processing time of iterated greedy, the round-robin scheduling technique was incorporated into the algorithm. This was achieved by incorporating a day of the week counter for each academic year of undergraduate courses, and an additional counter for post-grad courses. The counter was set to advance by one after every class scheduled, ensuring that the algorithm spread the classes across the week for majority of the students. This also reduced the search space for feasible time-slots by a factor of 5, thereby improving average run-time for the iterated greedy algorithm and spread of classes across the week.

4.9 Experimental Results

The objective of the allocation process for formal teaching was to allocate workload to faculty in a fair and equitable manner, while taking into consideration their preferences and capabilities. To assess how successful the proposed algorithm was in achieving these goals, we compare the results for the 4 academic years against the actual manual allocation results for year 1.
4.9.1 Workload Distribution for Teaching Allocation

The first metric used to assess the quality of results was the variation of workload from optimal values. Below is a table which lists a set of workload variation and the corresponding number of faculty which fell in that category for each academic year of implementation.

<table>
<thead>
<tr>
<th>WL Variation (%)</th>
<th>Number of Faculty</th>
<th>Manual</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
<th>Year 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>-50</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-45</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-40</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>-35</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-30</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-25</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-20</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>-15</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>-10</td>
<td>1</td>
<td>2</td>
<td>10</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>-5</td>
<td>2</td>
<td>7</td>
<td>11</td>
<td>12</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>5</td>
<td>15</td>
<td>15</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>+5</td>
<td>3</td>
<td>4</td>
<td>21</td>
<td>16</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>+10</td>
<td>3</td>
<td>8</td>
<td>9</td>
<td>14</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>+15</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>+20</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>+25</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>+30</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>+35</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>+40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>+45</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>+50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1 Workload variation levels for faculty across various implementations.

![Workload Variation versus Number of Faculty](image)

Figure 4.11 Comparison of variations in workload allocated to faculty across various implementations.
In the above graph, the X-axis is the net overload (or under-load, as in case of negative values), which is the variation from the optimal workload for a faculty. The Y-axis is the number of faculty which had the given variation. For eg, there were 2 faculty with 0% (+/-2.5) overload for the manual allocations. Ideally, all faculty should be in the 0% (marked using a dotted line) net overload bracket, as it denotes perfect loading.

The above representation can be summarized by calculating the average variation (standard deviation from 0) of the workload results, as tabulated below:

<table>
<thead>
<tr>
<th>Academic Year of Implementation</th>
<th>Average Workload Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>34.5%</td>
</tr>
<tr>
<td>Year 1</td>
<td>28.8%</td>
</tr>
<tr>
<td>Year 2</td>
<td>10.2%</td>
</tr>
<tr>
<td>Year 3</td>
<td>10.5%</td>
</tr>
<tr>
<td>Year 4</td>
<td>13.2%</td>
</tr>
</tbody>
</table>

Table 4.2 Average workload variation values for manual and automated allocations.

As we can see, before using ATAS, the manual system resulted in a highly skewed workload distribution, with considerable number of faculty members being severely over or under loaded. The Year 1 ATAS implementation used the Naïve Greedy approach, while in later years the Improved Greedy approach was used. The naïve method made modest initial gains, and the distribution became considerably fairer with the improved implementation.

### 4.9.2 Faculty Preference and Performance for Allocations

Apart from the workload distribution, the actual quality of allocation must also be assessed to validate the inclusiveness of the ATAS. For this purpose, we first compare the faculty preference distribution of the assignments made as shown in the table below. The preference column refers to the rank of the course assigned on the faculty’s preference list, and the values under each academic year denote the number of assignments made at the corresponding rank.
Table 4.3 Distribution of faculty preferences for teaching allocation assignments.

Next, we compare the performance distribution, which is a reflection of the student feedback that a faculty received. The values are reflected in the table below, and are arranged similar to the preference distribution table. Lower performance values denote better performance/feedback score.

Table 4.4 Distribution of faculty performance scores for teaching allocation assignments.
The preference and performance data can be summarized better by computing their workload adjusted averages across the various academic years/implementations. This was done by summing up the product of individual assignment workload and the faculty’s preference/performance score for it, and dividing it by the total workload for the year. The results were as follows, with lower values denoting better assignment quality:

<table>
<thead>
<tr>
<th>Academic Year of Implementation</th>
<th>Average Preference</th>
<th>Average Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>2.77</td>
<td>5.41</td>
</tr>
<tr>
<td>Year 1</td>
<td>2.80</td>
<td>5.23</td>
</tr>
<tr>
<td>Year 2</td>
<td>2.09</td>
<td>5.26</td>
</tr>
<tr>
<td>Year 3</td>
<td>2.11</td>
<td>5.22</td>
</tr>
<tr>
<td>Year 4</td>
<td>2.14</td>
<td>5.15</td>
</tr>
</tbody>
</table>

Table 4.5 Average preference and performance scores for manual and automated allocations.

As visualized, the trends for preference and performance distribution mirrored those seen for workload distribution. Notably, considerable gains were made with regards to preference distribution with the improved greedy implementation in years 2, 3 and 4.

Assessing the quality of results for the timetabling problem is not a straightforward task. This is because unlike formal and informal teaching, there isn’t a readily available metric in terms like workload distribution, preference or performance scores which can be directly compared. As a result, we instead compare the various soft-constraint violations - continuous classes, number of classes in a day, number of working days and utilization of teaching venues – from the point of view of the faculty, students and management separately. We compare the ATAS generated results for Y4S1 with the available manually generated timetable.
4.9.3 Faculty-friendliness of the Timetable

The number of classes in a day is an indicator of timetable quality for both faculty and student. Ideally, the number of hours must neither be too low nor too high to ensure a reasonability to the spread of the timetable. The tabulated results for faculty are given below.

<table>
<thead>
<tr>
<th>Teaching Hours in a Day</th>
<th>Number of Faculty Instances</th>
<th>Manual</th>
<th>ATAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>128</td>
<td>190</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>107</td>
<td>37</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>90</td>
<td>79</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>54</td>
<td>41</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>11</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td><strong>Standard Deviation</strong></td>
<td></td>
<td>2.2</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Table 4.6 Number of teaching hours in a given day for faculty.

The above table represents the number of instances when a faculty was assigned the corresponding number of teaching hours in a day. As stated earlier, faculty prefer having a reasonable amount of classes in a day rather than having too many or too few. In the given implementation, this amount was determined to be 4 hours. Additionally, faculty prefer having ‘free days’ i.e. days where they have no classes. As can be seen, ATAS performed better than the existing manual timetabling process considerably. It’s allocation policy followed the workload reasonability policy by ensuring that faculty workweeks could be broken down into two fairly distinct sets of ‘teaching’ and ‘free’ days.

The number of instances of zero teaching hour or ‘free days’ only provides quantitative insight by suggesting that ATAS condenses the timetable for faculty better overall. However, to check if all faculty benefit equally from it, we must consider the overall distribution of working days.

<table>
<thead>
<tr>
<th>Number of Working Days in a Week</th>
<th>Number of Faculty Instances</th>
<th>Manual</th>
<th>ATAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>9</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>19</td>
<td>38</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>32</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>23</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.7 Number of working days in a week for faculty.
As can be seen from the data above, ATAS considerably reduces the number of teaching days for a faculty. This is ideal as it allows the faculty to dedicate complete days to research and other duties including service/administration as well as student supervision. Additionally, it was also observed that the number of instances where a faculty was assigned two different classes back to back was reduced from 40 (manual) to 14 (ATAS).

4.9.4 Student-friendliness of the Timetable

Students are the other major stakeholders in the timetabling process. An ideal timetable must reasonably spread out classes for a student across the week to optimize the learning process. An indicator of this is the number of lectures scheduled in a day for courses that a student is eligible for. It is not an exact metric as students may choose to register for only a subset of the eligible courses offered on a given day, but provides a good enough measure from a statistical standpoint.

<table>
<thead>
<tr>
<th>Number of Lectures Hours in a Day</th>
<th>Number of Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Manual</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>Standard Deviation from 5</td>
<td>2.12</td>
</tr>
</tbody>
</table>

Table 4.8 Number of lectures hours in a given day for a student.

The above data was generated by grouping together courses offered in the same year of the degree programme, as most students are likely to register for courses under one of such groups. This resulted in 5 groups, 4 for under-graduates and 1 for graduate students. Combined with the 5 working days in a week, there were 25 instances – one for each year-day pair. The distribution above shows that for ATAS the range lies in 3-9 hours of lectures in a day, versus 0-11 hours for manual. Additionally, a typical day in ATAS is 4 hours of lectures for a student.
in a day, while that for the manual timetable was 5. The data for under-graduate year 2 is illustrated below to provide a clearer example.

![Graph of lectures in a day (UG Year 2)](image)

Figure 4.13 Graphic comparison of lectures scheduled in a day for UG year 2 students across the week.

Another metric to asset student-friendliness is the number of continuous classes that a student could need to attend. Ideally, a student must have a break between classes, especially for lectures. Below, we compare the number of instances when a student would have to attend continuous lectures in a day for each degree year of the under-graduate/post-graduate programmes.

<table>
<thead>
<tr>
<th>Degree/Year</th>
<th>Number of Continuous Lectures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Manual</td>
</tr>
<tr>
<td>UG Year 1</td>
<td>12</td>
</tr>
<tr>
<td>UG Year 2</td>
<td>10</td>
</tr>
<tr>
<td>UG Year 3</td>
<td>9</td>
</tr>
<tr>
<td>UG Year 4</td>
<td>10</td>
</tr>
<tr>
<td>PG</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.9 Number of continuous lectures scheduled for students across different years of their degree.

As can be seen, the instances where a student may need to attend continuous lectures is greatly reduced in case of the proposed automated system.

4.9.5 Utilization of Teaching Venues

From a management stand-point, it is essential to utilize infrastructural resources like teaching venues efficiently. To measure the utilization efficiencies of a given timetable, we define a utilization rate metric. The utilization rate is the number of students in a class divided by the capacity of the venue where the class is held. Typical values lie in the 0 to 1 range, with higher
values denoting more efficient usage. To illustrate the results achieved using the iterated greedy approach, we compare it against the existing school timetable for utilization of lecture venues.

<table>
<thead>
<tr>
<th>Venue</th>
<th>Capacity</th>
<th>Utilization Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Manual</td>
</tr>
<tr>
<td>V1</td>
<td>122</td>
<td>0.60</td>
</tr>
<tr>
<td>V2</td>
<td>122</td>
<td>0.38</td>
</tr>
<tr>
<td>V3</td>
<td>122</td>
<td>0.61</td>
</tr>
<tr>
<td>V4</td>
<td>122</td>
<td>0.67</td>
</tr>
<tr>
<td>V5</td>
<td>343</td>
<td>0.36</td>
</tr>
<tr>
<td>V6</td>
<td>577</td>
<td>0.03</td>
</tr>
<tr>
<td>V7</td>
<td>240</td>
<td>0.35</td>
</tr>
<tr>
<td>V8</td>
<td>240</td>
<td>0.41</td>
</tr>
<tr>
<td>V9</td>
<td>240</td>
<td>0.47</td>
</tr>
<tr>
<td>V10</td>
<td>331</td>
<td>0.43</td>
</tr>
<tr>
<td>V11</td>
<td>36</td>
<td>0.83</td>
</tr>
<tr>
<td>V12</td>
<td>36</td>
<td>0.83</td>
</tr>
<tr>
<td>V13</td>
<td>600</td>
<td>0.51</td>
</tr>
<tr>
<td>V14</td>
<td>240</td>
<td>0.60</td>
</tr>
<tr>
<td>Average Utilization</td>
<td>0.50</td>
<td>0.680</td>
</tr>
</tbody>
</table>

Table 4.10 Utilization rate for teaching venues across implementations.

As can be seen, ATAS utilized teaching venues more efficiently with a utilization rate of 0.68 compared to a utilization rate of 0.50 for the manually drafted time-table. Furthermore, it allowed for two venues to be freed up completely for use by additional courses.

**Notes**

1. Year 4 results are slightly worse than Year 3 and Year 2 for workload allocation owing to the fact that at the time of writing of this thesis there were still changes happening at the school.

2. The number of assignments for the performance distribution table is lower than that for preference because lab sessions were not surveyed for student feedback scores as a matter of school policy.

3. While all policy constraints were considered in the timetabling implementation, it is possible that certain real-world source constraints like unavailability of a venue at a particular time were missed. Nevertheless, the proposed solution allows for venues to be marked available/unavailable for certain periods only.
4.10 Summary

We began the chapter with a short discussion about the necessary features for an automated teaching allocation system that is inclusive in nature for students, faculty as well as the school management. The system was broken down into two major components – allocation and timetabling.

We first formulated the teaching allocation problem as a variant of the generalized assignment problem, and discussed its computational complexity which necessitated the use of an approximation algorithm. This was followed by an explanation of the greedy algorithm, as well as its merits and demerits. Its simple implementation, as well as the presence of priority heuristic made it the preferred choice for solving the formal teaching allocation problem. The naïve implementation of the greedy algorithm as well as its subsequent improvements were then described in detail. Finally, we addressed certain real-world concerns that came to light post initial implementation by modifying the algorithm accordingly.

Next, we discussed the multi-constrained problem of academic timetabling. The objective of scheduling classes in a way that clashes are avoided, while simultaneously spreading them in a manner that is faculty and student friendly is, is difficult to achieve. Additionally, there exists a need to efficiently utilize infrastructural resources like lecture halls, tutorial rooms and labs. Keeping all this in mind, a comprehensive problem formulation was carried out and the various constraints and parameters were identified. A desirability index was defined, as a combined function for the numerous soft constraints like number of classes in a day, reasonability of spread, and venue utilization.

The problem statement translated the timetabling process as a NP-complete problem. This necessitated the need for an approximation algorithm to produce acceptable results within a reasonable time-frame. As a result, a greedy-based approach was chosen for its simplicity and familiarity owing to the prior implementation for the formal teaching problem.

The proposed approach was an iterated variant of the greedy algorithm. This iterated greedy algorithm would first create an initial candidate solution based on the desirability index and a clash-based heuristic. This candidate solution would then be selectively altered by removing certain parts of the timetable during the destruction phase. As a follow-up, the solution would
be reconstructed in a construction phase using greedy techniques again. If the new solution is found to better than the candidate solution, it becomes the new candidate solution itself. The construction and destruction phases are iterated in this way until a certain number of iterations are complete or the quality of the solution meets a certain acceptance criterion.

We concluded the chapter with a discussion on the experimental results of the proposed solutions. The allocation algorithm yielded results which bettered manual allocations by a considerable margin, and were deemed more than acceptable by the school. Preference and performance indices improved by 18% and 7% respectively, while workload distribution improved by 21%. The iterated greedy approach too resulted in a solution which is better than simple greedy implementations, while at the same time not too computationally expensive. Deviations from the acceptable limit of 4 hours of daily teaching were reduced by 20% and 98% of all faculty were scheduled at least 1 day off. Deviations for students also reduced by 20%, ensuring they had better spread out weeks while venue utilization was improved by 30%.

In the next chapter, we deal with the allocation of informal teaching workload, which is the assignment of students to faculty for supervision of their Final Year Projects and M.Sc. dissertations.
5. ALLOCATING FINAL YEAR PROJECTS USING HUNGARIAN ALGORITHM

Informal teaching refers to the supervisory roles that faculty members must play for undergraduate final year projects (FYP), in addition to their supervision of Master’s and Ph.D. students. While assignment for Master’s and Ph.D. are mostly fixed owing to the faculty choosing their own sets of students, the allocation of FYP remains flexible. In this chapter, we propose a system to optimally assign FYPs to students based on their ranked preferences for them using the Hungarian algorithm.

5.1 Features of the informal teaching allocation system

The allocation strategy for FYPs (a subset of informal teaching) focusses on much of the same goals as formal teaching, with the exception of a slight change in priorities. To summarize, the system must incorporate the following features:

1. It must follow a student-centric approach towards their project preferences during the allocation process
2. It must ensure that faculty utilization and workload distribution is as close to ideal as possible
3. It should guide students’ project preferences towards their areas of interest and academic capabilities, while still allowing the students to have the final say
4. It must allow for multiple rounds of supplementary assignment to manage changes in the student/faculty make-up

In the next section, we discuss the differences between the natures of the informal teaching problem versus the formal teaching problem described previously in Chapter 4. We then formulate the problem statement and discuss its computational complexity.
5.2 Problem Formulation

As discussed in Chapter 3, informal teaching comprises of the supervisory roles played by the faculty for 3 different classes of students – Under-graduate Final Year Project (FYP), Master’s of Science/Engineering Dissertations (M.Sc.) and Doctor of Philosophy Thesis (Ph.D.). Each of these supervision have a weighted parameter which helps better compare them in terms of their real-world impact on a faculty’s workload. Similar to the formal teaching setting, there exist individual capacities for the number of students a faculty can supervise.

Next, faculty propose a number of projects, with the number being slightly higher than their capacity to facilitate greater variety for students. Each student then chooses a subset of these projects and ranks them in order of personal preference. Based on this preference data, assignments are to be made.

At first glance, the informal teaching problem seems to be similar to the formal variant, with the only change being the replacement of lectures/tutorials/labs by FYP/MSc/PhD student supervisions. However, a closer look reveals considerable differences:

1. **Dimensionality** - Instead of a two-dimensional ‘task’ (course) and ‘agent’ (faculty) system, the informal teaching is a three-dimensional system – there are students, projects and faculty. While project-faculty combinations are locked, the workload constraint keeps the problem from being reduced to two-dimensions.
2. **Completeness** - Unlike formal teaching where all courses had to be assigned to the available faculty, in informal teaching certain projects are guaranteed to go unassigned. This is because faculty propose more projects than are expected from them. As a result, the definition of completeness changes and the algorithm must choose which projects get assigned and which ones do not.
3. **Flexibility** – Formal teaching allocation allowed the system almost complete freedom when it came to assigning different types of teaching activities i.e. lectures/tutorials/labs. For informal teaching however, the number of M.Sc. and Ph.D. students is largely fixed.
4. **Heuristic** - There is no clear heuristic like ‘priority’. In the absence of a credible heuristic, a linear approach to the allocations like the greedy algorithm can lead to results which diverge greatly from optimal values.
Keeping the above in mind, the informal teaching problem is effectively reduced to the problem of allocating FYPs. It can be approximated to an ‘assignment problem’ with slight variations. The ‘assignment problem’ differs from the ‘generalized assignment problem’ (Section 4.2) in the way that it is a specialized case of generalized assignment problem where the assignments must be one-to-one. It is defined as:

*There are a number of agents (students) and a number of tasks (projects). Any agent can be assigned to perform any task, incurring some cost (preference) that may vary depending on the agent-task assignment. It is required to perform all tasks by assigning exactly one agent (student) to each task (project) and exactly one task (project) to each agent (student) in such a way that the total cost of the assignment (sum of ranked preferences) is minimized.*

There are two adjustments that must be made to this assignment problem for it to match the informal teaching allocation problem. First is the additional constraint that each FYP assignment (task) requires a faculty supervisor (resource) that is limited by the individual teaching contribution values. Second, is the fact that not all projects (tasks) are to be assigned, as the number of projects (tasks) is greater than the number of students (agents). Mathematically, we can represent the problem as follows:

*For two equal sized sets of agents A and tasks T, with a cost function $C_{ij}$ denoting cost of assigning agent $i$ to task $j$, the assignment problem can be described by the object function*

$$\text{Minimize } \sum_{i \in A} \sum_{j \in T} C_{ij} \cdot X_{ij}$$

*Subject to $\sum_{j \in T} X_{ij} = 1, \sum_{i \in A} X_{ij} = 1, X_{ij} \geq 0$ for $i,j \in A, T$*

*Where $X_{ij}$ is 1 when agent $i$ is assigned to task $j$, and 0 otherwise*

The informal teaching allocation problem is thus a special case of the assignment problem, which in turn makes it a specialized case of the GAP. Consequently, the problem is NP-Hard in terms of computational complexity. Assigning $n$ tasks to $n$ agents allows for $n!$ possible assignments. As $n$ grows larger, the time-complexity grows to a point that an exhaustive search is no longer feasible. We could use some of the techniques discussed earlier in section 4.2 that are used for NP-hard problem, but there exists a specialized technique specifically for the assignment problem – the Hungarian Algorithm.
5.3 The Hungarian Algorithm

The Hungarian Algorithm has been mentioned before in the literature survey at the beginning of this report (Section 2.3.2). The algorithm offers a polynomial-time, optimal solution to the assignment problem, and involves computations on a cost matrix with the objective of finding minimum cost. It was proposed H.W. Khun [13].

The algorithm first transforms the problem data into a square cost matrix by assigning a row to every agent and a column to every task. For n agents and tasks, the result is an n-by-n matrix M. The value of the M_{ij} cell is the cost of assigning task j to agent i. To find the optimal assignment so as to minimize overall cost, the algorithm performs the following steps:

- **Step 1** - Subtract the smallest value (lowest cost) in each row (agent) from all the cells of its row.
- **Step 2** - Subtract the smallest value (lowest cost) in each column (task) from all the cells of its column.
- **Step 3** - Draw ‘lines’ through appropriate rows and columns so that all cells having ‘zero value’ in the cost matrix are covered using the minimum number of lines possible.
- **Step 4** - If the number of covering lines is equal to n, an optimal assignment can be made based on zero values. Else, if the number of covering lines is less than n, the solution is not yet complete and we go to step 5.
- **Step 5** - Determine the smallest value of the reduced cost matrix that is not covered by a line. Subtract this value from each uncovered cell, and add it twice to each covered cell. Go back to Step 3.

Once the cost-matrix meets the optimality condition, the solution can be constructed by assigning the j^{th} task to the i^{th} agent wherever M_{ij} is now equal to zero. In case of multiple zero values in a row or column, assign agents/tasks with least number of feasible alternatives.

The problem can be better illustrated with the help of an example. Consider three agents A, B and C which must be assigned three tasks X, Y and Z, with the respective costs for each task being as described in the cost matrix below:

<table>
<thead>
<tr>
<th>Agents/Tasks</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>25</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>B</td>
<td>45</td>
<td>60</td>
<td>35</td>
</tr>
<tr>
<td>C</td>
<td>20</td>
<td>40</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 5.1 Sample problem to demonstrate the Hungarian Algorithm.
Then the optimal solution with minimum cost would be computed as follows:

### Step 1 – Subtract smallest value from each row

<table>
<thead>
<tr>
<th>Agents/Tasks</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>25</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>B</td>
<td>45</td>
<td>60</td>
<td>35</td>
</tr>
<tr>
<td>C</td>
<td>20</td>
<td>40</td>
<td>25</td>
</tr>
</tbody>
</table>

\[\Rightarrow\]

<table>
<thead>
<tr>
<th>Agents/Tasks</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>20</td>
<td>5</td>
</tr>
</tbody>
</table>

### Step 2 – Subtract smallest value from each column

<table>
<thead>
<tr>
<th>Agents/Tasks</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>20</td>
<td>5</td>
</tr>
</tbody>
</table>

\[\Rightarrow\]

<table>
<thead>
<tr>
<th>Agents/Tasks</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

### Step 3 – Cover all zero-values with minimum lines

<table>
<thead>
<tr>
<th>Agents/Tasks</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

### Step 4 – Construct optimal solution based on zero-values

<table>
<thead>
<tr>
<th>Agents/Tasks</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

\[\Rightarrow\]

<table>
<thead>
<tr>
<th>Agents/Tasks</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>25</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>B</td>
<td>45</td>
<td>60</td>
<td>35</td>
</tr>
<tr>
<td>C</td>
<td>20</td>
<td>40</td>
<td>25</td>
</tr>
</tbody>
</table>

Figure 5.2 Solution of the sample problem using Hungarian Algorithm.

The resulting solution A-Y, B-Z, C-X has the optimal minimum cost of 95.

The time complexity of the Hungarian algorithm has been proven to be O(n^4). However, it can be further reduced to O(n^3). The exponential time-complexity makes the algorithm preferable for problem sets which have a limited set of agents and tasks. In the next section, we describe the implementation of the Hungarian algorithm to find optimal results for informal teaching allocation.
5.4 Using Hungarian to solve Informal Teaching Allocation

While the time-complexity of the Hungarian algorithm is exponential, it is still acceptable owing to its ability to guarantee optimality and completeness of results unlike other approximation algorithms. Additionally, the student preferences for projects make a true cost function that can be leveraged by implementing Hungarian.

5.4.1 Simple Hungarian Implementation

The initial solution we developed for the informal teaching allocation problem was a direct implementation of the Hungarian algorithm. It involved constructing a cost matrix with each row denoting a student and each column denoting a project. The values of each cell were defined as the rank of the project as per the student’s preference. Each student was allowed to choose a maximum of 15 projects, so the values were in the 1 to 15 range. If the student didn’t prefer a project, a value of infinity (999 in practice) was assigned to that cell.

The Hungarian Algorithm requires the matrix to be square, i.e. the number of rows and columns must be equal. As mentioned earlier, in practice the number of projects is greater than the number of students to allow for a greater pool of options. To compensate for the additional projects, ‘dummy’ students were appended to the matrix rows. The value for each ‘dummy student’-project pair was set at 16, which was one lower than the lowest possible rank a student could assign to a preferred project. This ensured that no dummy student would be prioritized in the assignment of a project over an actual student.

Upon applying the Hungarian algorithm, an optimal solution was received which ensured that the sum of the ranks of the project assigned was the minimum for all students. This was in line with expected results. To illustrate, consider the simplified example below:

The matrix here represents 4 students (S1, S2, S3, S4) and their preferences for 5 different projects (F1, F2, F3, F4, F5) which are supervised by two faculty members (P1, P2). Each student prefers 3 projects, and the rest are denoted by a preference value of 99. A dummy student D1 is added and is marked with preference value 1 greater than highest allowed value in accordance with the algorithm.
Table 5.2 Simplified example of the FYP assignment problem.

Upon solving the above matrix with naïve Hungarian, the assignments would be as follows:

<table>
<thead>
<tr>
<th>Student/FYP</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>F2</td>
</tr>
<tr>
<td>S1</td>
<td>1</td>
<td>99</td>
</tr>
<tr>
<td>S2</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>S3</td>
<td>99</td>
<td>2</td>
</tr>
<tr>
<td>S4</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>DS</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Upon solving the above matrix with naïve Hungarian, the assignments would be as follows:

<table>
<thead>
<tr>
<th>Student/FYP</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>F2</td>
</tr>
<tr>
<td>S1</td>
<td>0</td>
<td>98</td>
</tr>
<tr>
<td>S2</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>S3</td>
<td>98</td>
<td>1</td>
</tr>
<tr>
<td>S4</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>DS</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

=>

<table>
<thead>
<tr>
<th>Student/FYP</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>F2</td>
</tr>
<tr>
<td>S1</td>
<td>0</td>
<td>98</td>
</tr>
<tr>
<td>S2</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>S3</td>
<td>97</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>DS</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Figure 5.3 Solution of the simplified FYP assignment problem using Hungarian Algorithm.

Ignoring the dummy student, we get the following final assignment

<table>
<thead>
<tr>
<th>Student/FYP</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>0</td>
<td>98</td>
</tr>
<tr>
<td>F2</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>F3</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5.3 Final solution using simple Hungarian.

This solution gives all students the highest preferred projects as far as possible, with the sum of preferences being 5. The solution can be verified to be optimal via exhaustive search.

While this was a great result to start with, the real world constraint of workload limits were not adhered to in the solution. In the above example, consider a case where the workload requirement was to have a maximum of 2 students per faculty. In this case, the naïve Hungarian algorithm cannot provide us a solution that meets the workload constraint as it assigns 3 students to faculty P2. We thus needed to improve the current algorithm.
5.4.2 Iterative Hungarian Implementation

The native Hungarian algorithm does not allow us to enforce limits on the number of students assigned to a single faculty due to its non-linear assignment process. The algorithm doesn’t know how many students have been assigned to a faculty until the final result has been achieved. This meant that the limit had to be enforced by altering the results post allocation.

The first attempt at enforcing the workload/student limits included a secondary pass similar to the one suggested in section 4.4.2. The modified algorithm would scan through the assignments made at the end of the Hungarian allocation and identify faculty that were assigned more students than expected as per their optimal workload values. The system would then scan for alternative projects for each of the students assigned to such faculty. The student with the best alternative that led to the smallest drop in preference ranks would be chosen to be swapped. In this way, number of student assignments to a faculty could be brought down to acceptable limits. However, this method was computationally expensive, sub-optimal and selectively biased against certain students who happened to have been assigned to certain faculty members for no fault of their own. Additionally, it was observed that altering a single student-project assignment could have cascading effects on other assignments.

The second attempt was a managerial fix, which involved randomly removing students from faculty who exceeded their limits, and asking them to participate again with a fresh ranked set of preferences in the secondary rounds of allocation along with students who had missed the first round. While devoid of bias, this still negatively impacted a sub-section of students and was not acceptable.

To counter these side-effects, a novel approach was proposed. The likelihood of a student being assigned a project, and in turn a faculty supervisor, was based solely on the student’s ranked preference of the project which served as the cost function. There existed a need to negatively impact the cost function for all students once a faculty had reached their designated limit. As this could not be done mid-execution of the algorithm, an iterative approach was needed.

At its core, the proposed approach is simple. Conduct a preliminary run of the Hungarian algorithm on the original cost matrix, and identify faculty that have been assigned more students than optimally required as per the workload calculations. Update the cost matrix by
increasing the student preference ranks of all projects offered by the faculty. Next, re-run the Hungarian algorithm on the new cost matrix. Continue iteratively until there are no violations to the student limits, or until a set of runs number of runs (15) is reached. This can also be explained by the following pseudo-code:

```
procedure ITERATIVE_HUNGARIAN
begin
31. create n-by-n matrix M, where n is number of projects
32. for all student-project preference
33.   Mij = preference rank of project j by student i;
34.   limit = 1;
35. do {
36.   S = Hungarian(M);
37.   if (faculty limits are not violated in M){
38.     limit = 0;
39.     final solution = S;
40.   }
41.   else{
42.     P = set of all faculty who have been assigned more students over their limit;
43.     for all projects j supervised by faculty in P
44.       {
45.         For all students i
46.         Mij = Mij +1;
47.       }
48.   }
49. } while (limit=1 OR n=15)
50. final solution;
end
```

Figure 5.4 Pseudo-code for the iterative Hungarian implementation for FYP allocation.

Due to the increased preference rank or ‘cost’ of allocating projects belonging to the faculty that were over-loaded, the likelihood of them being assigned in subsequent runs decreases. This leads to possible reassignments of students to the projects they preferred supervised by other faculty with lower workload/number of assignments. Consequently, subsequent iterations are less likely to over-load faculty members, and a fairer workload distribution is achieved. All of this is achieved without compromising on optimality as every iteration of the Hungarian is optimal by itself. Additionally, by incrementing ranks for all project-preferences under the faculty, we ensure that the re-assignments are fair and the impact on students is minimized.
Using the above technique, the solution to the earlier example constrained with the 2 student per faculty workload limit would be:

**Updated cost matrix**

<table>
<thead>
<tr>
<th>Student/FYP</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>F2</td>
</tr>
<tr>
<td>S1</td>
<td>1</td>
<td>99</td>
</tr>
<tr>
<td>S2</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>S3</td>
<td>99</td>
<td>2</td>
</tr>
<tr>
<td>S4</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>DS</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Post Hungarian Re-run**

<table>
<thead>
<tr>
<th>Student/FYP</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>F2</td>
</tr>
<tr>
<td>S1</td>
<td>0</td>
<td>98</td>
</tr>
<tr>
<td>S2</td>
<td>97</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>97</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>97</td>
<td>0</td>
</tr>
<tr>
<td>DS</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 5.5 Second Hungarian iteration to solve the simplified FYP assignment problem.

Ignoring the dummy student, we get the following final assignment

<table>
<thead>
<tr>
<th>Student</th>
<th>FYP</th>
<th>Preference</th>
<th>Supervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>F1</td>
<td>1</td>
<td>P1</td>
</tr>
<tr>
<td>S2</td>
<td>F3</td>
<td>1</td>
<td>P2</td>
</tr>
<tr>
<td>S3</td>
<td>F2</td>
<td>2</td>
<td>P1</td>
</tr>
<tr>
<td>S4</td>
<td>F5</td>
<td>1</td>
<td>P2</td>
</tr>
</tbody>
</table>

Table 5.4 Final solution using iterative Hungarian.

As can be seen, the iterative approach fixes the workload while having limited impact (in this case no impact) on the preference of projects assigned to students. The sum of preferences is still 5, but the assignment is now more favourable in terms of workload distribution. From a practical perspective, this is deemed as a more desirable solution as it is better for a student to receive full attention and guidance for a project he/she preferred less, instead of receiving limited guidance for a highly preferred project owing to the supervisor being over-loaded.
5.5 Experimental Results

For FYP allocation, an assessment of the proposed algorithm was carried out in a way similar to the formal teaching assessment. Results were compared on the basis of workload distribution and quality of assignments across 4 continuous semesters. It may be noted that the number of assignment to be made are considerably different across the even and odd semesters. Semester 1 has about 300 students, while Semester 2 typically has about 50 students.

The comparison further differed from the one for formal teaching owing to the fact that informal teaching allocation was done using an existing automated system, instead of a manual process. The existing system had no documentation to support it, but an analysis of the results pointed towards a possible greedy-based approach.

5.5.1 Workload Distribution

For informal teaching, we computed the variations for informal teaching by referring to the exact number of students assigned over or below the optimal numbers as computed in Chapter 3. The recorded data is tabulated below, with comparisons broken down into two sets. The first set compares results for Sem 2 across 2 academic years, and the second set compares results for Sem1. The breakdown allows for a clearer visualization owing to similar student numbers.

FYP Deviation refers to the number of additional/fewer students assigned to faculty, compared to the optimal computed value. ES refers to existing system.

<table>
<thead>
<tr>
<th>FYP Deviation</th>
<th>Number of Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ES Y2S2</td>
</tr>
<tr>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>44</td>
</tr>
<tr>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>+1</td>
<td>9</td>
</tr>
<tr>
<td>+2</td>
<td>1</td>
</tr>
<tr>
<td>+3</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.5 Workload deviations for FYP assignments using proposed (ATAS) versus existing system (ES) Sem 2.
Table 5.6 Workload deviations for FYP assignments using proposed (ATAS) versus existing system (ES) Sem 1.

<table>
<thead>
<tr>
<th>FYP Deviation</th>
<th>Number of Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ES Y3S1</td>
</tr>
<tr>
<td>-7</td>
<td>1</td>
</tr>
<tr>
<td>-6</td>
<td>1</td>
</tr>
<tr>
<td>-5</td>
<td>4</td>
</tr>
<tr>
<td>-4</td>
<td>2</td>
</tr>
<tr>
<td>-3</td>
<td>6</td>
</tr>
<tr>
<td>-2</td>
<td>9</td>
</tr>
<tr>
<td>-1</td>
<td>12</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>+1</td>
<td>13</td>
</tr>
<tr>
<td>+2</td>
<td>11</td>
</tr>
<tr>
<td>+3</td>
<td>9</td>
</tr>
<tr>
<td>+4</td>
<td>2</td>
</tr>
</tbody>
</table>

A graphical representation of the above data is given below:

Figure 5.6 Load distribution of FYP allocation using the proposed (ATAS), as well as existing system (ES).
The above graph is similar to the one used for formal teaching. The X-axis is the net over-assignment of students (or under-assignment, as in case of negative values), which is the variation from the ideal number of supervisions. The Y-axis is the number of faculty which had the given variation. For eg, there were 25 faculty who were assigned 2 additional students over their optimal supervision numbers during the ATAS Y3S1 run. The peak for ATAS at 2 over-assignments is due to the user-defined acceptable variation being set at 2 supervisions.

The impact on workload can be summarized by computing the average variation from optimal number of supervisions, which should be as low as possible for an ideal system. The resulting value was 2.2 students for ATAS, versus 2.5 for the existing system. This means that while the existing system over/under-loaded a faculty by 2.5 students, ATAS only did so by 2.2 - an improvement of about 12%.

### 5.5.2 Student Preference

The second metric used to assess the performance of the proposed iterated Hungarian approach is the distribution of student preferences across the assigned projects. Ideally, the system should assign projects that are as high up on the students’ preferred list as possible. The results are tabulated below, broken down again into semester-wise sets.

<table>
<thead>
<tr>
<th>Student Preference</th>
<th>Assigned Students</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ES Y2S2</td>
</tr>
<tr>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

| Average Assignment Rank | 1.50 | 1.75 | 2.20 | 2.37 |

Table 5.7 Comparison of average assigned preference rank (for student FYPs) by the proposed ATAS versus previously used greed-based allocation system – Sem 2.
<table>
<thead>
<tr>
<th>Student Preference</th>
<th>Assigned Students</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ES Y3S1</td>
</tr>
<tr>
<td>1</td>
<td>140</td>
</tr>
<tr>
<td>2</td>
<td>48</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>Unallocated</td>
<td>19</td>
</tr>
<tr>
<td>Average Assignment Rank</td>
<td>3.46</td>
</tr>
</tbody>
</table>

Table 5.8 Comparison of average assigned preference rank (for student FYPs) by the proposed ATAS versus previously used greed-based allocation system – Sem 1.

Graphically, the above data can be summarized based on average assignment rank:

![Average Student Preference](image_url)

Figure 5.7 Comparison of average assigned preference rank (for student FYPs) by the proposed ATAS versus previously used greed-based allocation system.
In the system previously used at the School, there was no provision for workload balancing and hence no constraints were enforced on the allocation of projects to students. Despite this, as shown above, the system performed poorly for semesters with larger sets of students (Sem1) due its inherently greedy nature with an average assignment preference of 3.46 versus 3.06 and 2.52 versus 2.30. Not only that, the number of unallocated students was reduced also reduced from 19 and 14 to 11 and 0 respectively. The improvement in ranks was over and above the gains in workload distribution.

For semesters with fewer students, the existing greedy system performed marginally better with regards to preference distribution with average scores of 1.50 versus 1.75 and 2.20 versus 2.37. However, these gains came at an extremely high cost of overloading faculty as discussed in section 5.5.1.

### 5.6 Summary

The problem of informal teaching allocation, which deals with assignment of students for supervision to the faculty, was discussed. The differences between formal and informal teaching problems were explained to better understand the need for a different allocation strategy.

Hungarian algorithm, a popular algorithm that provides optimal solutions to the assignment problem in polynomial time, was chosen as the preferred algorithm for solving the problem. While it performed well with regards to optimizing student preferences, it failed to address the workload distribution issue comprehensively. This necessitated a modification which resulted in the iterated Hungarian algorithm. The improved algorithm adjusted the cost matrix after every iteration so as to decrease the likelihood of over-loading faculty. Consequently, the iterative implementation provided results which satisfied the needs of both student and faculty, without any inherent biases. Same was validated by comparing against the existing system used at a reputable school.

In the next chapter, we provide an overview of the actual implementation of the proposed system as integrated stand-alone tool – Automated Teaching Allocation System (ATAS).
6. ATAS IMPLEMENTATION AND DEVELOPMENT

In this section, we discuss the implementation of the aforementioned models and algorithms to create the Automated Teaching Allocation System (ATAS), including an overview of the platforms and technology used.

6.1 System Architecture

The proposed teaching allocation system needed to be accessible via multiple access points. Additionally, it required separate interfaces for different types of users, yet needed a common database that reflected across users. Furthermore, the system was required to fetch data from multiple sources on the university network. Keeping these requirements in mind, a web-based architecture with a browser-based interface was chosen.

![Figure 6.1 Client-Server Architecture](image)

The result was a simple server-client set up. All system files were transferred to the server by the administrator using FTP. Data was stored in a relational database implementation on the server as well. By utilizing the server’s address and a set of verified credentials, any user connected the local university network could have access to the system via any major web-browser.

6.2 Technologies and Platforms

The system was hosted on a local university server running the Apache HTTP Server. Apache was chosen owing to its open-source nature and the massive user adoption which has led to a robust support system for it.
MySQL, an open source variant of the traditional SQL, was used for the relational database on the server, as most existing data was in a tabular format. A single database with multiple tables was utilized, with minimal redundancies across tables. The MySQL triggers and procedures were used extensively to ensure integrity and consistency of the data, while scheduled back-ups ensured reliability of the system. Upon completion, the database held over 60 tables with over a 50,000 rows of data,

For server-side scripting, including interfacing import/export queries as well as logic implementation, PHP was used. It is open source, highly stable and works extremely well with MySQL. While nowhere as fast as traditional platforms like C, utilizing PHP for realization of the algorithms and related logic greatly simplified the maintenance process by keeping the number of platforms used to the minimum. Additionally, the transmission of parsed data to the client in PHP limited the system’s exposure to undesirable elements.

The client side interface was designed using standard web-technologies like HTML, CSS and JavaScript. Special consideration was made to ensure cross-browser compatibility. Efforts were also made to ensure the interface was responsive to allow for a better user experience on mobile platforms.

Since the ATAS dealt with a host of confidential data, including but not limited to faculty emails and performance scores, it was necessary to secure the system from malicious users. The limited availability of the system over the university network provided the first layer of security by requiring that a user must first gain access to the network. For a second layer of security, the users were required to re-authenticate before every use. For the third and final layer, a comprehensive security scan was carried out to search for vulnerabilities across scripts and subsequently fixed.
6.3 System Screenshots

**Figure 6.2** Interface to edit core system parameters for workload model.

**Figure 6.3** Interface to add a new faculty record to the database.
Figure 6.4 Pre-allocation report providing an overview of the workload structure as well as ensuring integrity of system data.
Figure 6.5 Interface to trigger the allocation process for lectures.

Figure 6.6 Overview of allocation results.
7. CONCLUSION AND FUTURE WORK

In the section below, we conclude the discussion on the state of the system as well as the observations made during the course of this research exercise. This is followed by potential areas for future research and development.

7.1 Conclusion

This thesis addresses the problem of automating the allocation of workload to faculty members of a university department in a fair and equitable manner. The proposed solutions are holistic and inclusive in nature, and give due consideration to all parties involved – faculty, students and the administrative management.

The problem of workload allocation was broken down into four major components based on key activities undertaken by a faculty – research, administrative/service duties, teaching and supervision. To quantify these activities into comparable workload values, a novel Workload Unit Model (WLU) was proposed. The WLU model is a time-based model, which uses a combination of weighted parameters and actual contact hours to compute workload values. It partitions the three different workload components into individual modules that were interconnected using a relaxation system. The relaxation system adjusted teaching and supervision workloads based on the faculty’s involvement in research and administrative activities as the latter were largely fixed. Recognizing that the key variable component for workload allocation were the teaching and supervision activities, the two were formulated separately as formal and informal teaching workloads respectively.

Formal teaching workload involved the allocation of direct teaching activities of undertaking lectures, tutorials and lab sessions while satisfying conflicting objectives of the stakeholders involved. Faculty were represented by their preferences for various courses, as well as their teaching appointment which necessitated a fixed contribution towards teaching duties. The school was represented by its need to balance workload across the limited faculty resource while adhering to certain policies. Lastly, students were represented via the teaching feedback scores they assign to the faculty members. A greedy-based approach was proposed to automate
and optimize the workload allocation by making assignments in manner which maximized the returns for each of the stakeholders involved. The said returns were quantified in terms of a feasibility value - a combined function proposed for the constraints of the formal teaching workload problem based on faculty preference, student feedback, workload limits and other constraints. The resulting allocation solutions were largely acceptable, but certain assignments made by the system posed challenges in the real world. One such challenge was the magnitude of change as the workload allocated to faculty by the proposed system often differed substantially from what they have been teaching in the past in terms of courses taught, and required the faculty to put in considerable preparation effort. To resolve this, additional constraints in the form of a limit to the number of previously untaught courses were introduced.

Another challenge was the imbalance of faculty preference for assignments of courses and activities that were considered low priority due to the strict ordering followed by the proposed greedy algorithm. Though theoretically in line with school policies, this was found to be too rigid in the real-world and severely impacted allocation quality. To address this, a secondary pass of the greedy results was computed to fix inefficiencies introduced due to the priority-based ordering. The resulting approach considerably improved workload distribution and quality of assignments over manual process. Preference and Performance indices improved by as much as 18% and 7% respectively. Simultaneously, the average deviation from optimal workload of individual faculty fell from 35% to 14%, a drop of about 21%.

With the workload allocation problem for formal teaching resolved, it was essential to propose a scheduling algorithm for a student and faculty-friendly timetable so as to create a comprehensive solution. It was deemed essential to propose a solution that went beyond simply avoiding clashes and accounted for adequate breaks, a reasonable spread and an efficient utilization of teaching space. Additionally, the NP-complete complexity of the timetabling problem necessitated an efficient approximation algorithm that provided near optimal results. To achieve this, the greedy algorithm used for formal teaching allocation was built upon to propose an iterated greedy solution. The proposed algorithm removed and reconstructed components of an initial solution repetitively in a bid to find better results. A feasible result with a favourable spread of teaching activities was obtained this way – reducing typical work-day deviation for teaching hours by 20% for both students and faculty, while also improving class-venue fit by 30%. Additionally, the number of continuous classes were reduced for both faculty and students considerably.
Informal teaching workload was the other variable component that was addressed as a part of the overall teaching allocation system. It involved the assignment of students to faculty for supervision of their projects/thesis (predominantly final year projects or FYPs), with a key focus on students and their preferences. Optimizing for this single objective was straightforward with the use of the Hungarian algorithm that reduced the problem to square cost-matrix for each student-faculty assignment. While this approach produced optimal distribution of student ranked preferences for the assigned FYP, it was limited in its ability to enforce a fair distribution of workload. To eliminate this limitation, a novel approach involving multiple iterations of the Hungarian algorithm was proposed, as the Hungarian algorithm is unable to dynamically alter the cost constraint based on the state of the assignment. By using an iterative approach, the cost matrix was updated in between iterations to increase the cost of assignments for faculty who exceeded the number of students they should be assigned in the previous iteration. This increased the likelihood of over-loaded faculty being assigned fewer students in future iterations while simultaneously ensuring that there existed no bias when it came to dropping a few preference ranks for the students in favour of a better workload distribution. The resulting solutions comprehensively bettered those achieved by a competing existing system – improving student preferences by 10% and workload deviations by 15%. However, it was noted that the impact on preference improvement was reduced for smaller sets of students owing to the smaller workload limits, reduced overload tolerance and student tendencies to prefer certain popular FYPs over others.

Lastly, an overview of how the proposed models and algorithms can be implemented into an integrated tool which automates the entire workload allocation process was provided. A web-based system built on the Windows-Apache-MySQL-PHP stack is recommended, keeping in mind the database and multiple-access requirements of such a tool.

7.2 Future Work

While a strong foundation for solving the workload allocation problem has been built by this thesis, there exist a number of avenues to improve and extend it. Some of the possibilities are listed below.
7.2.1 Improving Data Collection and Integration

The quality of results produced by ATAS are highly dependent on the quality of the input data provided to the system. For the most part, data collection for ATAS was carried via a manually controlled import of spreadsheets and database exports from other systems used at the school/university which often introduced errors and inconsistencies into the database. By programming automated direct data-links between ATAS and existing systems, and scripting additional self-checks for database consistency, the required data can be collected quickly without any loss of accuracy.

7.2.2 Balancing Teaching Allocations across Semesters

The proposed workload allocation system computes workload and makes assignments for the entire academic year, especially with regards to formal teaching. The semester-wise breakdown of the resulting allocation is often skewed as a result, leading to faculty being often assigned most of their teaching duties in a single semester – something which the faculty may or may not prefer. By recording individual faculty preferences for workload across semesters, an additional constraint of semester balancing can be introduced into the allocation process.

7.2.3 Improving Allocation Results

The formal teaching and timetabling algorithms proposed in this thesis were limited by the need for a simple and transparent allocation logic that could be utilized to justify moving away from manual allocations towards an automated system that could be trusted. Once this trust has been established, the underlying algorithms can afford to grow in complexity if the resulting solutions are closer to optimal. Based on the appetite for complexity and availability of computational resource, certain approaches like convex optimization and genetic algorithms or even Hungarian implementation for the formal teaching workload can be explored with the hopes of further improving the results.

7.2.4 Analysing Shortages and Future Requirements

The large amounts of data collated for the automated teaching allocation system along with the set of results produced can be used to identify domains where there exists a shortage or excess of faculty experts. For example, finding courses that are preferred less by the faculty can help identify possible areas in which faculty must be hired.
7.2.5 Economic Analysis

A major part of workload allocation to faculty members is the financial cost of employing the faculty as well as that of utilizing the physical resources like lecture halls, labs and equipment. These have not been factored in as yet, but offers potential cost-saving opportunities from the management perspective. By including financial data like student fees, staff salaries and running costs for infrastructure, a support-system for core economic decision making at the university level can be made.

7.2.6 Applications beyond Educational Institutions

Workload allocation is a tricky problem to solve not just in the educational industry, but also in other industries like aviation, medicine, hospitality and manufacturing. By making relevant modifications to the algorithms and data structures proposed in this thesis, a number of real-world problems can be solved.
References


