Proceedings of the International Conference on Industrial Engineering and Operations Management
Pilsen, Czech Republic, July 23-26, 2019

Teaching Assistantship Assignment Optimization using Hungarian Algorithm – A Case Study

Chandra Mouli R. Madhuranthakam
Chemical Engineering Department,
Abu Dhabi University
Abu Dhabi, United Arab Emirates P.O.Box. 59911
chandra.mouli@adu.ac.ae

Mukhtar Al-Ismaily[1] and Ali Elkamel[1,2]
1- Chemical Engineering Department,
University of Waterloo,
Waterloo, Ontario, Canada
mukhtar.al-ismaily@uwaterloo.ca
aelkamel@uwaterloo.ca

2- Chemical Engineering Department,
Khalifa University of Science and Technology,
Abu Dhabi, United Arab Emirates, P.O.Box. 2533
ali.elkamel@ku.ac.ae

Abstract

Teaching assistantship (TA) assignment is typically a time-consuming and enduring process due to conflicting constraints and multiple objectives. This project involves assigning TA assignments to a large pool of candidates over a smaller set of offered courses, all the while ensuring that fairness is maintained during the assignment. A program was developed to address and optimize based on the given constraints and objectives. It begins with a data collection phase where all relevant information on the candidates and courses is acquired and then a software program is used to optimize the TA assignment using the Hungarian algorithm. The model was initially formulated as a multi-objective function consisting of three objectives viz., Course/Instructor Preference, Student Preference and Income deviation among candidates. It was then scalarized into a single weighted-sum objective function. The optimizer was tested on candidates applying for teaching assistantship towards three separate semesters, yielding an average of 60-70% matching with the manual assignments that were previously carried out by the department. The statistical dispersion in optimization cost and matching extent towards the manual assignment was observed under the different set of weights. Higher assignment deviations would extend the Pareto domain, which also results in more favorable optimization costs.

Keywords
Multiobjective optimization, Hungarian algorithm, Data analysis, Conflicting constraints.

1. Introduction

Before the start of a semester, departments in various institutions must carry out the crucial duty of assigning TAs (teaching assistants) to courses offered in an upcoming semester. The supply and demand between available courses and candidates varies across various departments. In some cases, the supply prevails and TA tasks (typically assigned to graduate students) are given to undergraduates or graduates from other departments. In such matters, especially when the sample size of tasks to assign are low (under 10), the selection process becomes trivial and can be completed rather quickly. In other cases, for instance, the department of Chemical Engineering in the University of Waterloo has to undergo the process of assigning numerous tasks (approximately 20) from a pool of at least forty candidates. The assignment of TAs is carried out manually by a committee of staff and faculty members. This process involves a large investment of their time and effort. At times, candidates may not be satisfied with their assignment, or aggrieved for not being selected after many attempts in the past. Also due to human error, rare cases may also exist where conflicts and inconsistencies may arise with the assignments made.

The purpose of this article is to optimize the TA assignment process, which is primarily geared towards the matching of a large number of tasks across an even larger pool of candidates. In such a case it is impossible to generate an assignment on which everybody agrees with, however it would be possible to achieve an outcome that is most desirable for all parties (faculty, students) involved. The optimized selection process presented in the project is computerized, which eludes to the fact that the TA assignment will be accomplished in a matter of seconds as opposed to numerous tedious hours using the manual method. A few past studies have employed combinatorial mathematics to optimize assignments in the academic setting. Amongst the earliest accounts was carried out by He (2002) who produced an application that primarily assisted in managing TA-course scheduling conflicts and TA workloads within a Java™ platform, while assignments were still carried out manually [1]. In automated assignments however, CPLEX solver in GAMS that employs the branch and bound method became a common choice in numerous publications that followed. Üney-Yüksektepe & Karabulut (2011) proposed a mixed integer linear programing (MILP) assignment problem that accounts for TA conflicts with student timetables, workload, and history to minimize preparation time due to prior knowledge [2]. The article by Ozdermir and Gasimov (2004) and the follow-up by Ismayilova et al. (2007) were more geared towards assigning courses to faculty member by managing scheduling conflicts and workloads, in addition to considerations from administration preferences. An optimization is carried out where multiple objectives are synthesized into a single objective function, with the weights of each objective determined via an analytic hierarchy process (AHP) [3 & 4]. Daskalaki et al. (2004) suggested a more comprehensive model which also manages timetable conflicts between students, instructor, and classroom availability [5].

Maya et al. (2012) developed an application for assigning TAs to help disabled children by optimizing TA workload distribution and travelling costs using the Greedy randomized adaptive search procedure (GRASP) [6]. In Glaubius' (2001) proposed model, considerations were made to TA preferences and their history, which are aspects that inspired the current paper [7]. Another source of influence is the application reported by Fuentes (2014) that utilizes the Hungarian algorithm to optimize the TA assignment [8]. This article starts with the mathematical description of the assignment model, then proceeds with a brief review of the Hungarian algorithm used to optimize the selection process. A complete description and hierarchy of the developed application is then presented, followed by the discussion of outcomes obtained on sample cases, with suggestions on any potential future work and recommendations.

2. Mathematical Modeling

Mathematical development of this assignment is presented in the form of an MILP with the goal of minimizing the final cost objective function. The aim of this model is to assign a set of available tasks of size m from a pool of applicable candidates of size k. An important distinction needs to be made between the number of tasks (m) and number of courses (n), since some [of the larger] courses need more than one TA, and consequently offer more tasks. Therefore, m is always greater than or equal to n. This section will begin with a description of the known parameters, then proceeding with the decision variables and the objective function, and concluding with the constraints imposed on the problem.

2.1 Known Parameters

One of the most important parameters involved in the decision making is the preference. There are two sets of preferences provided: 1- TA → Course Preference, and 2- Course → TA Preference. The former is obtained as a set of inputs provided by the candidates where they rank their preferences in the respective courses, while the latter has the course instructor rank their preferred candidates. The selection of these variables is similar to the task

assignment model proposed by Dinkel et al. (1989) which included instructor preferences towards courses, timeslots, and classrooms [9].

$$SP_{ij} \quad \in \{0,1,2,3\}, \quad \forall ij \tag{1}$$

$$CP_{ij} \quad \in \{0,1,2,3,4,5\}, \quad \forall ij \tag{2}$$

$$i \in \{1,2,...,k\}$$

$$j \in \{1,2,...,m\}$$

Where SP_{ij} corresponds to student preferences, while CP_{ij} is the instructor/course preference. The values {1, 2, 3...} are the respective ranks provided, and if no rank is provided then it is assigned a value of 0. Also, when a student ranks a specific course *l* that contains more than 1 task, the rank is imposed on all its tasks. This is discussed further in the next section.

$$I_j \quad \in \{0.5,0.75,1\}, \quad \forall j \tag{3}$$

$$T_i \quad \in Z, \quad \forall i \tag{4}$$

$$L_i \quad \in \{0.8,1\}, \quad \forall i \tag{5}$$

$$H_i \quad \in \Re, \quad \forall i \tag{6a}$$

Tasks are divided into 3 types {C, B, A} which correspond to different working hours/assigned budgets, I_j. The normalized fractions are used instead of dollar amounts; this is because changes in budgets are more probable in future semesters than the ratios between the hours of different task types. T_i represents the duration student *i* was present in the program in the form of registered terms. L_i corresponds to the candidate's level in the form of degree sought {Masters, Doctorate}. Another important parameter which is accounted for is the TA history, H_i. This is the cumulative earnings made by candidate *i*. It plays a crucial role in an attempt to reduce the disparity between the funding of candidates. It is a cumulative sum of multiple task-type values of [0.5, 0.75, 1] given to said candidates in past semesters. One may observe from equations (3) to (6a) that freedom to change the values or eliminate any level-based biases is possible on the client-end.

2.2 Decision Variables

In a nutshell, a decision needs to be made on the assignment of TA *i* on course *j*. For an assignment problem, there will exist an $m \times k$ number of decision variables.

$$X_{ij} \quad \in \{0,1\}, \quad \forall ij \tag{6b}$$

Where X_{ij} represents a binary decision made on the assignment of TA *i* to task *j*. An assignment is made via a value of 1 or 0.

2.3 Objective Function

The objective (also called as cost) function is modeled using the parameters described in Eq. (1) to (6). This is a multi-objective optimization problem, where the aim is to minimize each individual objective (f_1, f_2 and f_3) of the cost function.

$$f_1(x) = \sum_{i=1}^{k}\sum_{j=1}^{m} CP_{ij} X_{ij} \tag{7}$$

$$f_2(x) = \sum_{i=1}^{k}\sum_{j=1}^{m} \frac{SP_{ij} X_{ij}}{L_i} \tag{8}$$

$$f_3(x) = \sum_{i=1}^{k}\sum_{i'=1}^{k} \left| \frac{H_i + \sum_{j=1}^{m} X_{ij} I_j}{\min[T_i + 1, 12]} - \frac{H_{i'} + \sum_{j=1}^{m} X_{i'j} I_j}{\min[T_{i'} + 1, 12]} \right| \qquad \textit{where } i \neq i' \tag{9}$$

The first objective given by equation (7) corresponds to the course preference, while the second objective equation (8) accounts for the student preference, with their level playing an important role if needed. Finally, the third objective given by equation (9) computes the income-fairness amongst the candidates. This is done by calculating the total income deviation between students. The numerator takes a minimum between the total number of terms or it is equal to 12 (typical term-duration for doctorate students). The longer the student has been registered, and the

lower the cumulative funding and the more the function cost will be to his/her advantage. Deviation between the candidates ensures proper parity between their earnings. A similar concept for candidate deviation was also carried out by Güler et al. (2015) where penalty costs are computed from a candidate's target load (soft constraints) [10].
A multi-objective problem is often solved by combining its multiple objectives into one single-objective scalar function. This approach is in general known as the *weighted-sum* or *scalarization* method. Essentially, the weighted-sum method minimizes a positively weighted convex sum of the *sub*-objectives [11],

$$\min \quad \sum_{p=1}^{3} \frac{w_p \cdot f_p(x)}{f_{p,\max}}$$

$$\sum_{p=1}^{3} w_p = 1 \qquad (10)$$

$$w_p > 0$$

$$p \in Z, \quad \forall p$$

By substituting the f_i from equations (7) through (9), the resulting single unique objective function is given by equation (11).

$$\min \frac{w_1}{f_{1,\max}} \sum_{i=1}^{k} \sum_{j=1}^{m} CP_{ij} X_{ij} + \frac{w_2}{f_{2,\max}} \sum_{i=1}^{k} \sum_{j=1}^{m} \frac{SP_{ij} X_{ij}}{L_i} + \frac{w_3}{f_{3,\max}} \sum_{i=1}^{k} \sum_{i'=1}^{k} \left| \frac{H_i + \sum_{j=1}^{m} X_{ij} I_j}{\min[T_i + 1, 12]} - \frac{H_{i'} + \sum_{j=1}^{m} X_{i'j} I_j}{\min[T_{i'} + 1, 12]} \right| \qquad (11)$$

Each sub-objective is divided by its maximum possible value; this is to ensure they are all between a value of 0 and 1, which would suppress any biases between the different sub-objectives. For instance, if the maximum value of CP_{ij} is a rank of 5, then

$$f_{1,\max} = CP_{\max} X_{\max} = (5)(1) = 5$$

2.4 Constraints

The problem tends to be overly constrained when the demands largely outweighs the supply. The constraints applied to this problem are typical of most classical assignment problems and are shown in equation (12) and (13).

$$\sum_{i}^{k} X_{ij} \leq 1 \qquad (12)$$

$$\sum_{j}^{m} X_{ij} = 1 \qquad (13)$$

Where the first constraint given by equation (12) ensures that no candidate is assigned more than one task, while leaving room for the possibility that some of them don't get assigned at all. The second constraint given by equation (13) ensures that every task is assigned.

This model is only applicable for the cases where $k \geq m$. Otherwise, both constraints would be overly restrictive, and a feasible solution won't be possible.

3. Optimization using Hungarian Algorithm

Microsoft Excel was used as a platform for modeling the TA assignment problem. This is done for various reasons. Firstly, the input data (model parameters) are obtained as set of metadata in the form of an excel spreadsheet. This set of data is compiled from the faculty database in combination with the list of preferences provided by candidates and course instructors beforehand. Also, Excel is accompanied with a Visual Basic for Applications (VBA) editor where scripts may be written to carry out custom operations. This is particularly useful, since it allows for the freedom of catering the spreadsheet to the TA assignment model. Finally, its popularity and user friendly interface allows the flexibility for the future assignment of TAs on the intradepartmental level. Having also established itself as the industry standard for spreadsheets, and with the growing compatibility it has with many other 3rd party applications, Excel has designated itself as the de facto platform for this problem. The most important reason however, is that Excel comes accompanied with an optimization tool (SOLVER). The original intention is to use this tool to carry the optimization. However, it wasn't until it was later realized that SOLVER comes with a severe

limitation in handling assignment problems, where the number of decision variables cannot exceed 200. It was therefore decided that the Hungarian method is to be used - also scripted from the ground up - to carry out the optimization for this problem. It is the most appropriate method due to its relative simplicity compared to other combinatorial algorithms, and exactly having the same constraints as the problem statement at hand. Combinatorial optimization is a subset of mathematical optimization. It involves finding the optimum solution from a set of finite objects. The most common combinatorial problems are shortest path, transportation, and assignment-type problems [12]. Many assignment combinatorial optimization techniques have been developed in the past, such as the Hungarian method, Branch and Bound method, and the Simplex method. In this application, the Hungarian method is adopted due to its relative simplicity, and its effectiveness for handling assignment problems within the scope of our constraints. It may also be asserted that a pencil and paper Hungarian method can complete a (10x10) assignment problem faster than a computer that utilizes a brute force method [13]. It is termed the Hungarian method due to contributions of four Hungarian mathematicians throughout the 19th and 20th century: Jacobi, Dénes Konig, Jeno Egerváry, and Harold Kuhn.

The problem is represented by a $k \times k$ cost matrix, where each element in the matrix is assigned a cost coefficient C_{ij} (from objective function) for the assignment of candidate i to task j. An important requirement for the matrix is that it has to be a square matrix (for e.g. of equal dimensions as denoted by $k \times k$). In the case where more candidates exist such as the current assignment problem ($k \times m$), additional "imaginary" columns/tasks are added containing a cost slightly higher than the cost of the highest coefficient. Imaginary coefficients are denoted as $C^*_{ii^*}$ as shown in Figure 1.

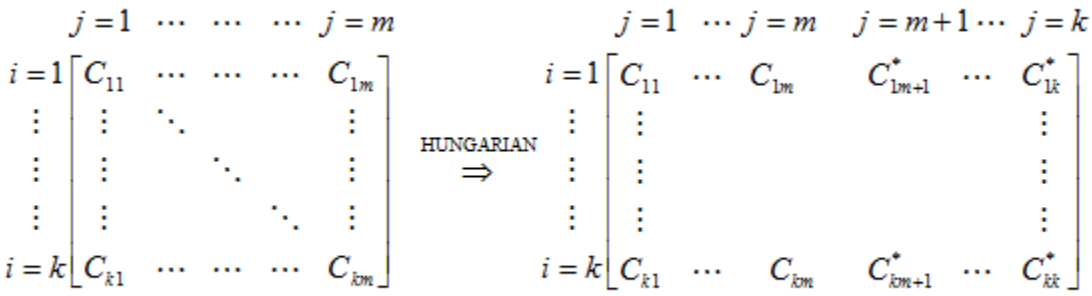

Figure 1 – Remapping of original $m \times k$ cost matrix to a $k \times k$ Hungarian form

Where,

$$C_{ij} \in \mathbf{C} \qquad i = \{1,2...k\},\ j = \{1,2...m\}$$

$$C^*_{ii^*} = \max[\mathbf{C}] + 1 \qquad i^* = \{m+1, m+2...k\},\quad i^* \neq j$$

Initially, in the case where no student has noted a preference on a course/task, a coefficient of $C_{ij} = \max[\mathbf{C}] + 1$ is assigned, which denotes a high cost and should be avoided during the minimization. It is required to assign all tasks, where there is exactly one candidate to each task and exactly one task to each candidate in such a way that the total cost of the assignment is minimized. Candidates assigned to the "imaginary" tasks are the ones who weren't able to get a position for the semester.

The problem is described as a standard linear program given by equation (14) (Fuentes (2014) [8]):

$$\min Z = \sum_{i=1}^{k}\sum_{j=1}^{m} C_{ij} X_{ij} + \sum_{i=1}^{k}\sum_{i^*=m+1}^{k} C_{ii^*} X_{ii^*} \qquad (14)$$

$$
\begin{aligned}
s.t. \quad & \sum_{j=1}^{m} X_{ij} + \sum_{i^*=m+1}^{k} X_{ii^*} = 1 && \forall i \\
& \sum_{i=1}^{k} X_{ij} = 1 && \forall j \\
& \sum_{i=1}^{k} X_{ii^*} = 1 && \forall i^* \\
& X_{ij}, X_{ii^*} \geq 0 && \forall i, j, i^*
\end{aligned}
$$

Where X_{ij}/X_{ii^*} represents the assignment of candidate i to task j/i^*. The first constraint ensures that every candidate is assigned, while the second and third constraints guarantee that every real task and imaginary task is assigned, respectively. The goal is to find a solution to vector $X \in \mathbf{R}^n$ such that all constraints are satisfied and the objective function is minimized. The overall optimization model is given by eqution (15).

$$
\min \quad C_{ij} = V_{ij}\left(\frac{w_1}{f_{1,\max}} CP_{ij} + \frac{w_2}{f_{2,\max}} \frac{SP_{ij}}{L_i} + \frac{w_3}{f_{3,\max}} \sum_{i'=1}^{k} \sum_{j'=1}^{m} V_{i'j'} \left| \frac{H_i + I_j}{\min[T_i + 1, 12]} - \frac{H_{i'} + I_{j'}}{\min[T_{i'} + 1, 12]} \right| \right)
$$

$$
V_{ij} = \begin{cases} 1, & \text{If } SP_{ij} > 0 \\ 0, & \text{If } SP_{ij} = 0 \end{cases}
$$

$$
\text{where } i \neq i' \qquad j \neq j' \tag{15}
$$

It is important to note that the cost coefficients are computed before maximizing the zero and imaginary coefficients. The third objective has been slightly modified, where only the deviation against candidates *i'* that have placed a preference on tasks *j'* is evaluated (as shown in Figure 2). This feature is handled by the coefficient V_{ij}.

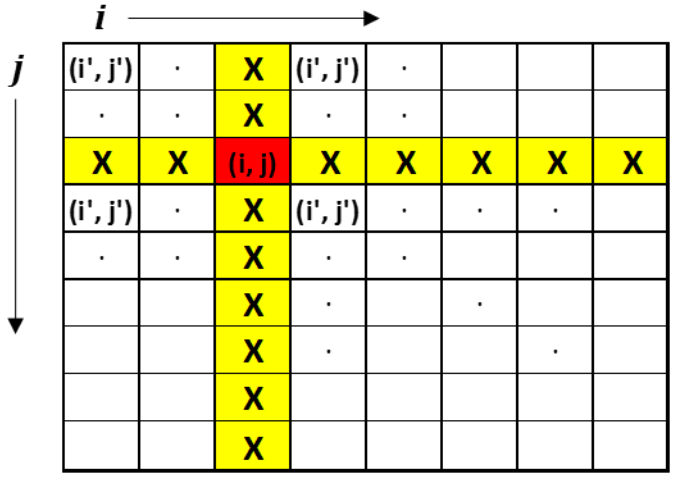

Figure 2 - Illustrating how the deviations (3rd objective in Eq. (15)) is computed within the cost matrix. Indices (*i'*, *j'*) are never on the same column or row of the referenced (*i*, *j*)

3.1 Hungarian Algorithm

Let **C** be the $k \times k$ cost matrix, with $C_{ij}, C_{ii^*} > 0$ representing the cost assigning candidate i (rows) to task j, i^* (columns). The goal is the optimum assignment of candidates to tasks and Hungarian algorithm provides a solution to this problem with $O(k^4)$ complexity.

1- For each row, subtract the minimum cost. Repeat the same for each column where no zeros exist. The updated matrix is called **C**'.

2- Find the minimum number of lines (p), through both rows and columns that cover all of the zeros in **C**'.

3- If $p < k$, let d equal to the minimal element in C_{ij},C_{ii^*} such that C_{ij},C_{ii^*} is not covered by any of the p lines. For all uncovered elements C_{ij},C_{ii^*} subtract d, while for elements covered twice, add d. Update matrix and return to step 2.

4- If $p > k$, construct a set of k independent zeros, where for all C_{ij},$C_{ii^*} = 1$, else C_{ij},$C_{ii^*} = 0$. This assignment output is the final new matrix [8]. The Hungarian method's default optimization position is minimization. In case the user would prefer to maximize, then every element in the Hungarian Matrix is first subtracted off some arbitrary maximum, before proceeding with the algorithm.

4. Model Development

This section provides a brief overview of the structure and hierarchy of the TA assignment model constructed in MS Excel via the VBA editor. The selection process goes through four phases (as shown in Figure 3), starting with the data collection process, then proceeding with the problem formulation, then finally with the optimization procedure and delivering the final TA allocations. The computations occur via interactions between the Excel worksheets and the compiled VBA code.

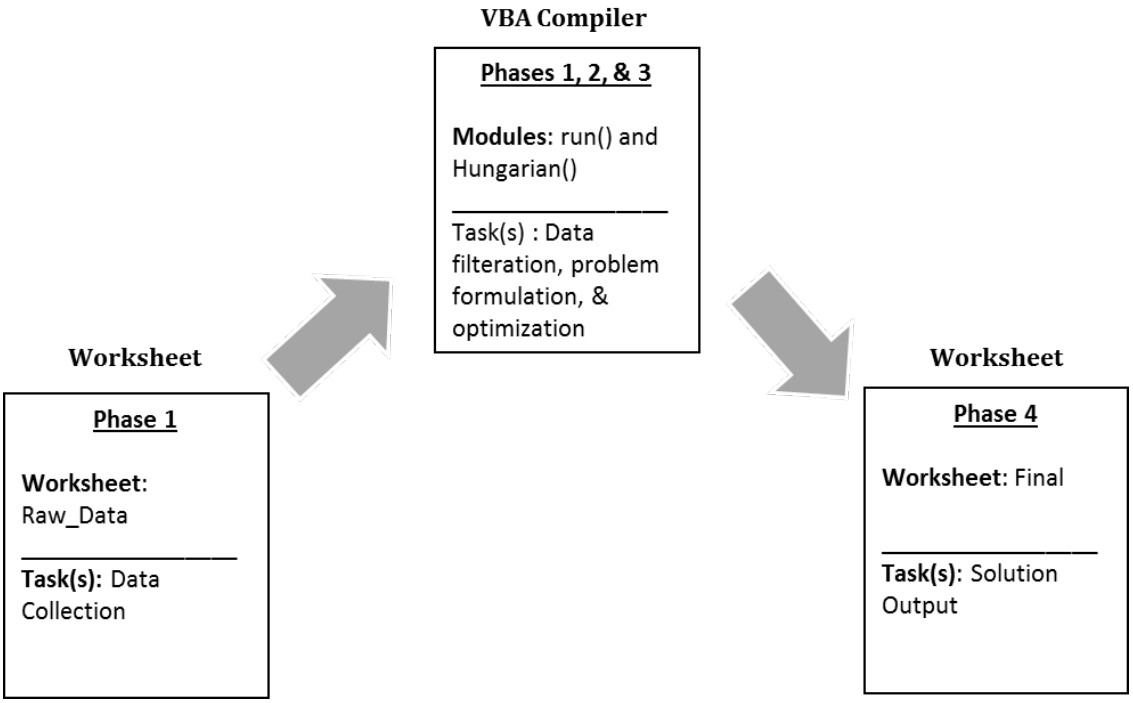

Figure 3 - Illustration of transition of phases within model

The Excel workbook for handling the TA assignment is designed to expect the final input data in a very specific type of format. The initial input data is tabulated on the server level which contains the academic information and history of the teaching assistantship for all students registered in the department's programs (as shown in Figure 4).

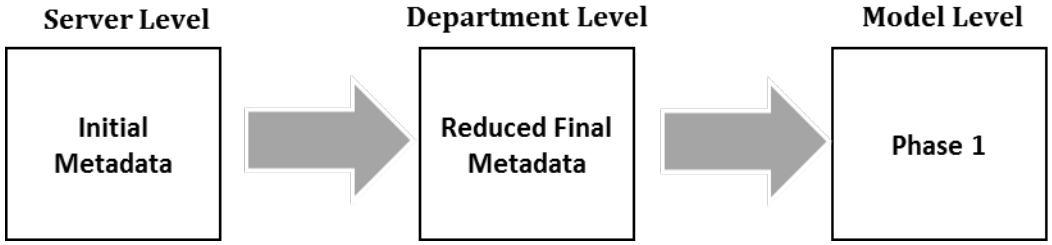

Figure 4 - Illustration of the transfer of raw input data to developed model

The data also includes information on the courses that need to be assigned TAs for the next semester, number of tasks, type of tasks and information on the preferences. Only a fraction of all available information is needed, this fraction is extracted and delivered via the department's technical staff. After, data collection the compiled code begins to extract all required information for the TA assignment, and store them in arrays of varying dimensions depending on the nature of the data. Before extraction, the data is filtered by removing non-applicable candidates, who don't meet the "YES" requirement in columns 9 → 13 as shown in Figure 5.

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8
NexusID	Last Name	First Name	Preference	Preflevel	Instructorpreference	tahistory	positions

Column 9	Column 10	Column 11	Column 12	Column 13	Column 14	Column 15	Column 16
SafetyTraining	TAManual	Expectations	SafetyManual	SupervisorApproval	PhDComp	Plantoregister	Registered

Column 17	Column 18	Column 19
FT/PT	Level	Cum Term

Figure 5 - Column-wise format of Reduced Final Metadata required for Phase 1

This is because students are expected to have undergone the training sessions (Safety Training, ExpecTAtions…etc.) to be eligible for teaching assistantship. This is a university-wide requirement. However, as with the case in Eq. (3) and (6), these restrictions may also be relaxed on the client's side. The order of columns and their contents are shown in Figure 5. Figure 6 demonstrates a sample of how the metadata is extracted, transformed and stored into various VBA data-type arrays. For instance, to obtain TA_List, the NexusIDs (unique) from Column 1 are collected and stored into the one-dimensional array. Then based on the order or TAs listed in TA_List, another one-dimensional array is developed by combining the first and last names in columns 3 and 2, respectively. Data extraction and storage is also carried out on columns 7, 18, and 19 to provide data teaching assistantship history, degree sought, and cumulative terms registered (not illustrated in Figure 6).

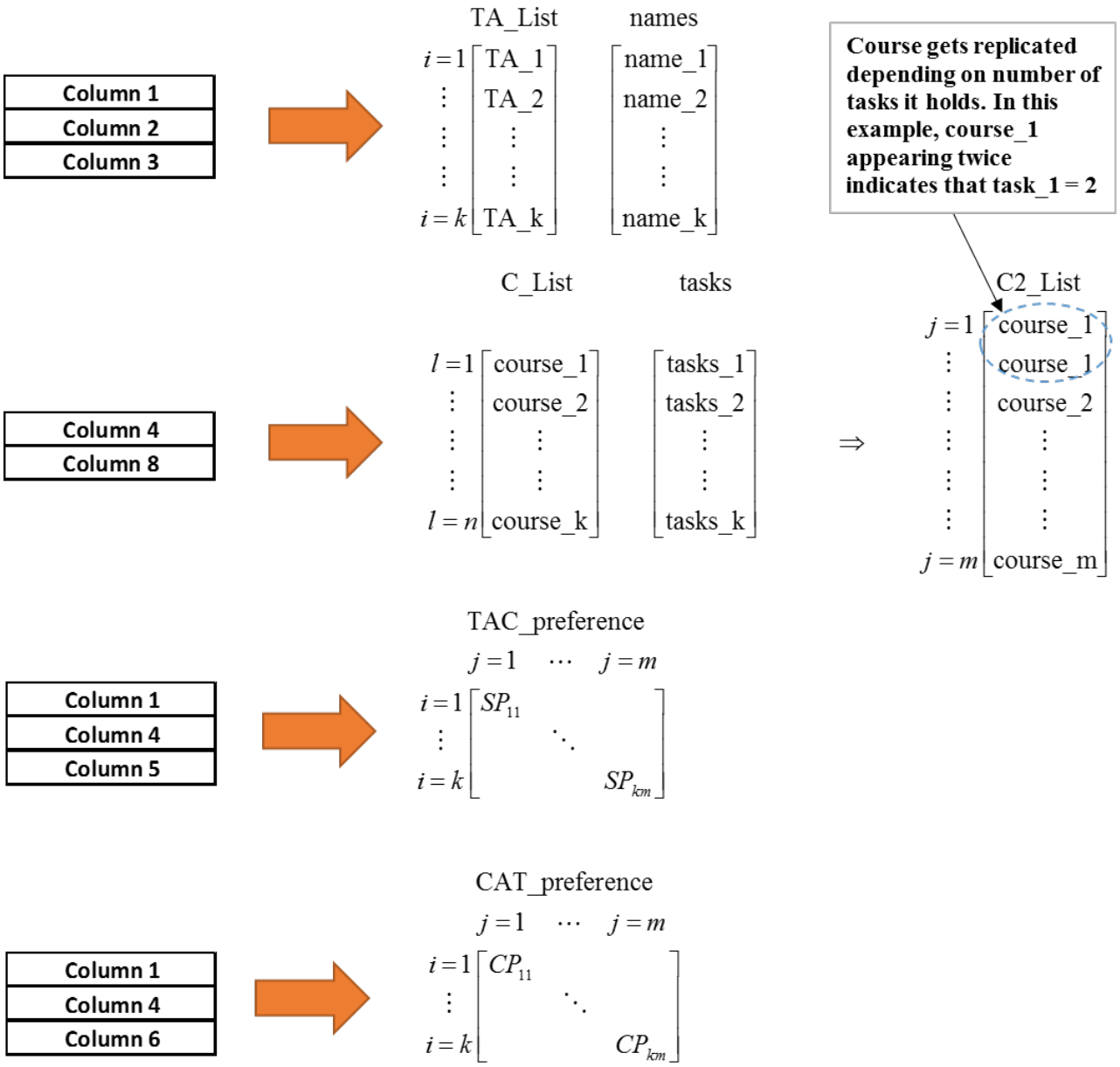

Figure 6 - Metadata extraction and data storage [SAMPLE] from the column sets in Figure 5

In addition to sorting out the data, this stage is also designed to check if the number of tasks don't exceed the number of applicable candidates, otherwise that would violate the constraints in equations (12) and (13) and a feasible solution cannot be obtained. In the Mathematical Modelling section, a model was developed for optimizing the TA assignment. This was later modified in the model development section to adopt to the Hungarian algorithm format via the cost coefficient matrix. The problem formulation stage essentially involves populating the cost matrix of size $k \times k$ using the equation (15) and passing it into the Hungarian() function located in its designated VBA module. The cost matrix passed to Hungarian() is multilayered, making it a three-dimensional array. The third dimension is used to keep track of the evolution of the cost matrix from the steps outlined in the Hungarian Algorithm section, and to keep track of zero covered rows and columns. The final assignment matrix located on level 2 of the 3D dimensional Hungarian evolution array is returned, containing ones (1s) on the TA(i)-task(j) assignment, and excluding the columns containing the imaginary tasks. The resulting matrix should satisfy

constraints given by equations (12) and (13). Outputting allocation results involves translating the resulting assignment matrix and matchmaking the allocation of TAs to the courses and their respective tasks.

5. Results and Discussion

The assignment model was tested on the raw metadata provided for three past semesters: fall 2014, spring 2015 and winter 2016. Table 1 summarizes the assignment parameters for each semester. Semester 3 for instance is supplied a total of 65 candidates, where 8 were disqualified for not meeting the safety or expectations or approval conditions (Columns 9-13, Figure 5), and resulting in a total of k = 57 applicable candidates.

Table 1: Assignment parameters for Semesters 1, 2, and 3 where the optimization is carried out

Semester	**1**	**2**	**3**
Total Candidates	49	51	64
Applicable Candidates (k)	47	44	57
Courses (n)	20	18	25
Tasks (m)	46	31	38
Decision Variables ($m \times k$)	2162	1364	2166

The problem formulation is carried out on m = 38 tasks (from n = 25 courses), leading to a total of 2166 decision variables (X_{ij}). This example provides ample justification for developing an in-house optimizer instead of using Microsoft Solver, which is limited to 200 decision variables. Optimizing via brute force is also futile, since $2^{m \times k}$ permutations will need to be generated and computed. For the current scenario 2.4 x 10^{636} permutations are to be generated, and current office PCs wouldn't possess the processing power of carrying out a brute force assignment within a feasible time-frame. Before generating an output and mapping the cost matrix, the sub-objective weights from equations (10), (11), and (15), [w_1, w_2, w_3] must be provided. To justify their selection, a collection of all possible set of weights are produced as a series of Barycentric coordinates, where the unit step size of $\Delta w_p = 0.1$ and the non-zero weight ($w_p > 0$) conditions are still maintained, e.g. [0.1, 0.1, 0.8], [0.1, 0.2, 0.7], [0.1, 0.3, 0.6] …etc., including an additional point of [w_1= 1/3, w_2 = 1/3, w_3 = 1/3] where all weights are equal. As a result, this leads to a total of 37 possible weight vector runs to be analyzed.

The assignment optimization is carried out for each possible set of weights on all three semesters, where two outputs are examined: the total cost C_{tot}, and the mismatching extent to the manual assignment, μ are given by equations (16) and (17).

$$C_{tot} = \sum_{i=1}^{k} \sum_{j=1}^{m} C_{ij} X_{ij} \tag{16}$$

$$\mu = \frac{1}{2} \sum_{i=1}^{k} \left| \sum_{j=1}^{m} \left(X_{ij} \right)_{man} - \sum_{j=1}^{m} \left(X_{ij} \right)_{opt} \right| \tag{17}$$

The total cost (equation 16) is computed by summing up the product between each element in the cost and [Hungarian] assignment matrices. The mismatch extent (μ) in is the number of assignments made in the optimization that don't match the manual assignment. The total cost in each run are shown in Figures 7(a) to (c).

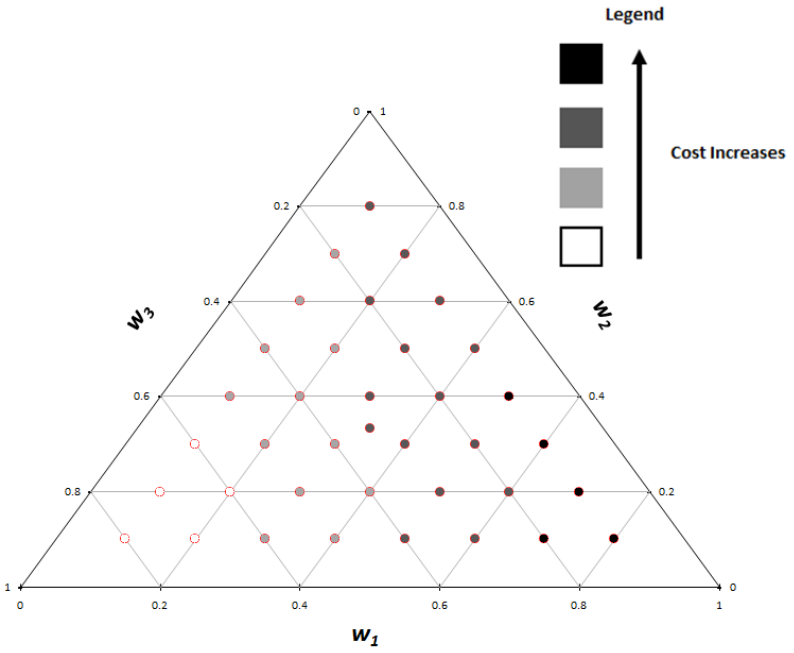

Figure 7 (a) – Ternary plot exhibiting the change in optimization cost under various weight vectors represented as Barycentric coordinates (Semester 1)

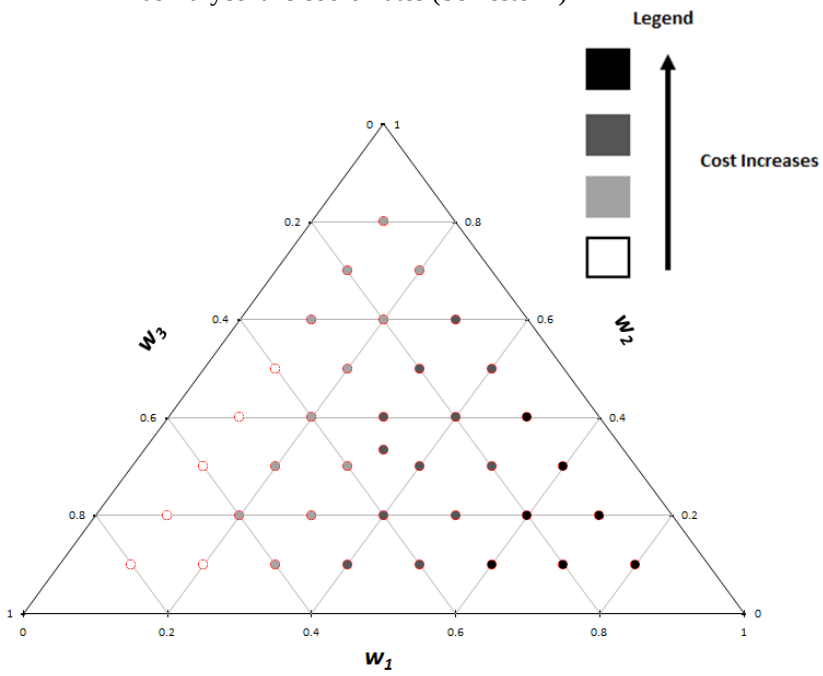

Figure 7 (b) – Ternary plot exhibiting the change in optimization cost under various weight vectors represented as Barycentric coordinates (Semester 2)

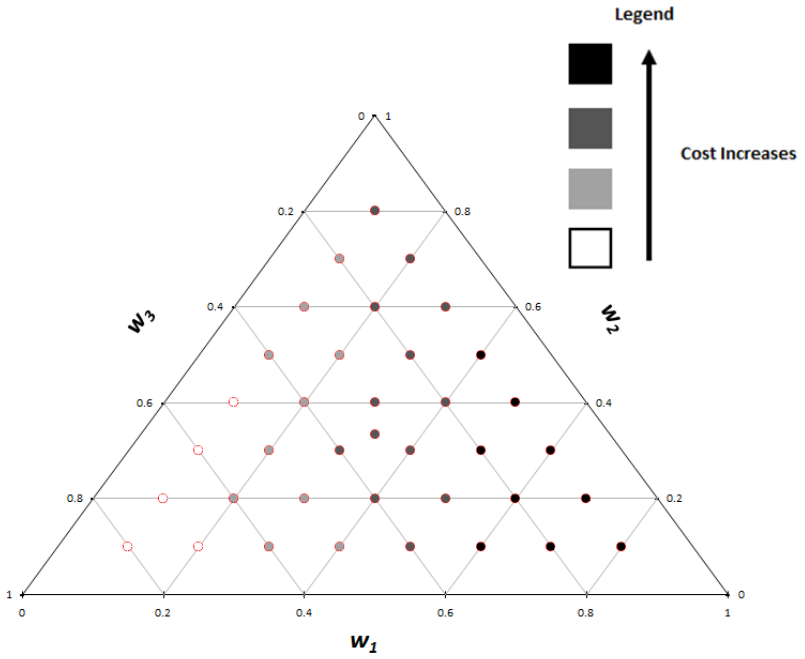

Figure 7 (c) – Ternary plot exhibiting the change in optimization cost under various weight vectors represented as Barycentric coordinates (Semester 3)

All semesters exhibit the same pattern where cost is lowest when the weight of the income fairness sub-objective function (w_3) is at a maximum. However, this is by no means indicative of a favorable designation for the weights, since the preference-based objective functions (f_1, f_2) are more prone to biases due to their discontinuous nature compared to their more continuous counterpart (f_3). Also, considering the optimizer automatically sets a maximal (least favorable) value to CF_{ij} for candidates who had no preference ascribed back to them by the instructor, this should therefore explain the high cost towards the course preference weight (w_1) in the ternary plots. Caramia et al. (2008) outlined that there is no a-priori correspondence between a weight vector and a solution. It is up to the developer to choose appropriate weights, noting that weighting coefficients do not necessarily correspond directly to the relative importance of the objective functions [11]. However, in the case where the objective functions are scalarized with similar ranges in magnitude ($0 < f_p < 1$ in our case), the relative importance may indeed prevail. It should be noted that the manual assignment (within equation 17) for each semester do not necessitate an ideal scenario, but is merely used as a reference point since the optimizer aims to improve upon it, and not just act as a time saving alternative. Figures 8 (a) and (b) show the mismatch extent recorded for each weight vector run on all three semesters, where no real trend or consistent set of weight vectors are observed. This is indicative that no set of weights would necessarily provide the lowest mismatch, especially when considering the uncertainty involved with the manual assignment due to human error.

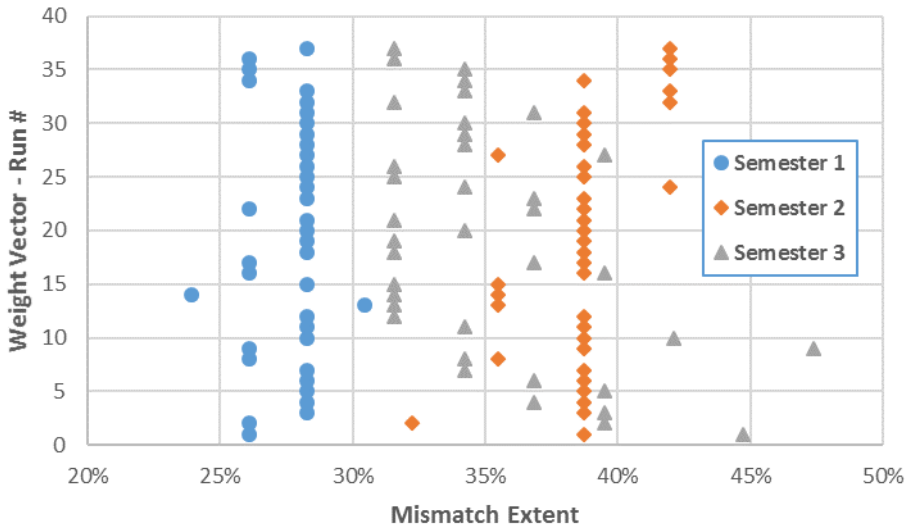

Figure 8 (a) – Dispersion of the mismatch extent across various sub-objective weight vectors, on all three semesters. Each weight vector run # corresponds to a specific set of weights represented as Barycentric coordinates.

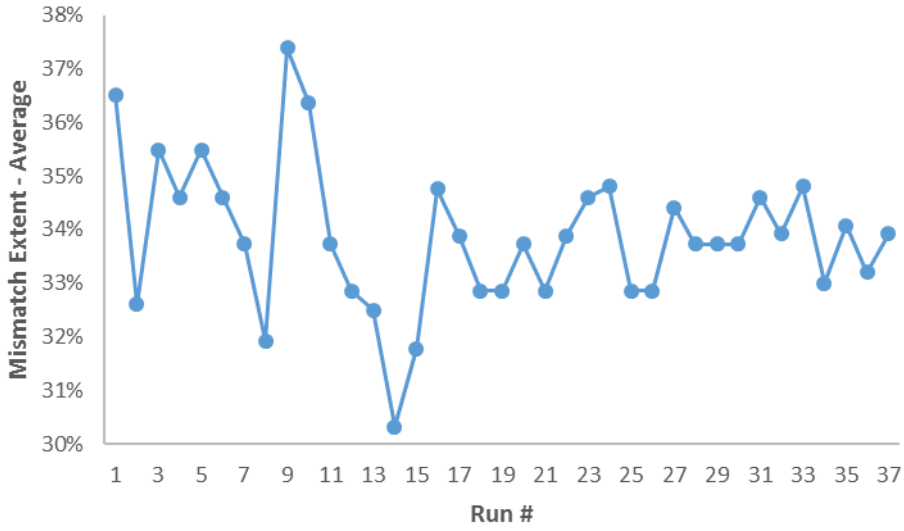

Figure 8 (b) – Dispersion of the three-semester averaged mismatch across various sub-objective weight vectors. Each weight vector run # corresponds to a specific set of weights represented as Barycentric coordinates.

One notable observation demonstrated in Figure 8 (a) is the varying degree of data dispersion among the 3 semesters. This is further elaborated in Table 2 where semester 3 exhibits the highest standard deviation, followed by semester 2, and semester 1. The distribution of tasks among courses in juxtaposition with the allocation odds may be used to interpret these deviations.

Table 2: Statistical dispersions on the costs and mismatch extents under all 37 weight vectors.

Semester	**1**	**2**	**3**
Mismatch Extent Median *(μ)*	**13** of 46 (28.3%)	**12** of 31 (38.7%)	**13** of 38 (34.2%)
Mismatch Extent	0.571	0.645	1.501

Standard Deviation (σ)			
Maximum Deviation (+)	14	13	18
Minimum Deviation (-)	11	10	12
Average Cost (C_{avg})	0.491	0.472	0.362
Allocation Odds	77.93	91.00	116.6

Figures 9 (a) and (b) illustrate these distributions, and it seems that certain courses such as course #2 in semester 1 and course #17 in semester 2 possess a much higher clustering of tasks compared to the rest of the courses in those respective semesters (leptokurtic distribution).

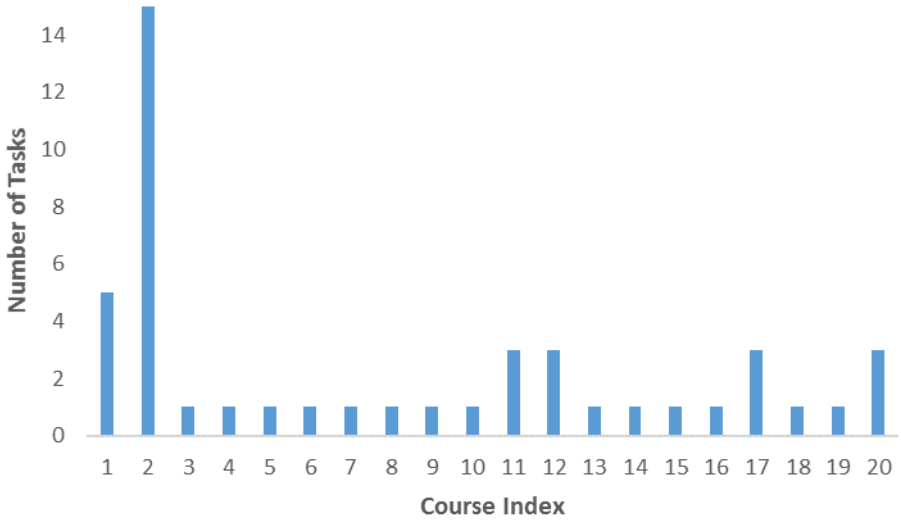

Figure 9 (a) – Distribution of tasks among courses in semester 1

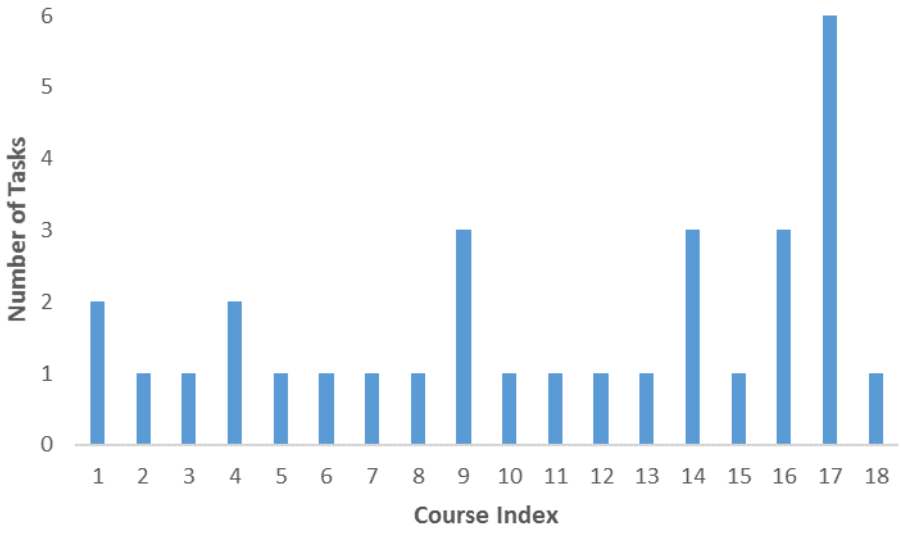

Figure 9 (b) – Distribution of tasks among courses in semester 2

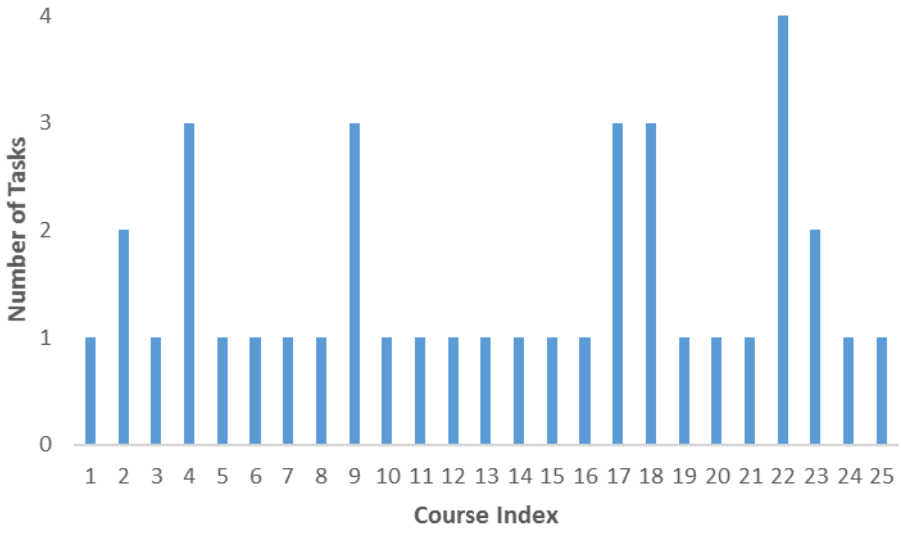

Figure 9 (c) – Distribution of tasks among courses in semester 3

This may be indicative of lesser availability in assigning TAs towards other courses when varying the set of weights. This is further reinforced when analyzing the allocation odds (given by equation (18)) provided in Table 2.

$$\tau = \sum_{q=1}^{n}\left(\frac{1}{t_q}\sum_{i=1}^{k}\frac{SP_{iq}}{SP_{iq}}\right) \qquad \forall\ SP_{iq} > 0 \tag{18}$$

The allocation odds in Eq. (18) is the number of registered preferences by students towards a specific course divided by the number of available tasks in that course, summed up over all courses within a semester. As shown in the equation, τ is contingent on the provided raw data but not on the optimization. The higher the allocation odds, the higher will be the combination of available assignment solutions. As a result, it serves as an overview to the rigidity of a particular assignment problem. From the analysis of the three semesters, the mismatch deviations (σ) when tested against multiple weight vectors increased with an increase in allocation odds. It should be noted that computing the allocation odds a-priori can serve to test the optimizer's susceptibility to variations in the sub-objective weights. This perhaps also explains why the *average cost* tends to be lowest with semester 3 (as shown in Figure 9 (c) containing the highest τ). When more possibilities are obtainable and the formulated assignment problem is flexible, the statistical advantage from expanding the Pareto domain would yield more favorable costs.

6. Conclusion

The project attempted to replicate the TA assignment computationally via the Hungarian algorithm and was successful in achieving the desired objectives. The developed VBA script was able to successfully extract all necessary candidate and preference information from the TA application metadata, and parse the income history of each candidate, then carry out the optimization procedure. Optimization involving minimizing the objective function was carried out on the developed model that had three individual sub-objective functions accounting for the income deviation, student-course preferences, and vice versa. The three objectives were averaged into a single function using a vector of three weights represented as Barycentric coordinates. The optimizer was tested on candidates applying for teaching assistantships in three separate semesters, yielding an average of 60-70% matching extent from the manual assignments previously carried out by the department. This was done across 37 different combinations of varying weight vectors.

The statistical dispersion in optimization cost and matching extent towards the manual assignment was analyzed using different set of weights. If the number of available tasks and registered student preferences were more balanced and not clustered towards a few courses (termed allocation odds), the more susceptible the model was towards assignment deviations. Higher assignment deviations would extend the Pareto domain, resulting also in

more favorable optimization costs. The constructed model was able to successfully carry out feasible assignments on all weight vector combinations, and achieve its primary purpose of eliminating the grunt work required by the department TA committee every semester. Also, by being able to adjust the weight vector, the client is given the flexibility to prioritize the model's decision-making process.

With VBA's programming paradigm being object-oriented (mostly!), it provided the developer with plentiful possibilities of updating and adding additional features in any future iterations without major alterations to the base code. In the latter stages of the model's development, additional features were added such as relaxing the conditions of disqualification (for not meeting certain pre-requisites), and forcing the assignment of certain students towards a course. Other options may seamlessly be added in the future, such as allowing a student to be assigned to more than one course, or if the client would like to reduce the parity between students based on other secondary attributes such as gender or immigration status. The optimizer may also undergo more extensive modifications for usage in other problems, such as optimizing administrator and faculty member tasks under different sub-objective criteria.

References

[1] He, J. Teaching Assistant Assignment Planner. Master's Thesis, Concordia University, *Retrieved from ProQuest Dissertations and Theses*, 2002.

[2] Üney-Yüksektepe, F.; Karabulut, İ. Mathematical Programming Approach to Course-Teaching Assistant Assignment Problem, *41st Int. Conf. Comp. Ind. Eng.*, Los Angeles. 2011.

[3] Ozdemir, M. S.; Gasimov, R. N. The analytic hierarchy process and multiobjective 0–1 faculty course assignment, *Eur. J. Op. Res.* 157, 2004, 398-408

[4] Ismayilova N. A.; Sagir M.; Gasimov R. N. A multiobjective faculty–course–time slot assignment problem with preferences, *Mat. Comp. Mod*. 46, 2007, 1017-1029

[5] Daskalaki S.; Birbas T.; Housos E. An integer programming formulation for a case study in university timetabling, *Eur. J. Op. Res.* 153, 2004, 117-135

[6] Maya, P.; Sorensen, K.; Goos, P. A metaheuristic for a teaching assistant assignment routing problem, *Comp. Op. Res.* 39, 2012, 249-258

[7] Glaubius, R. A Constraint Processing Approach to Assigning Graduate Teaching Assistants to Courses. Honor's Thesis, University of Nebraska-Lincoln, *Retrieved from ProQuest Dissertations and Theses*, 2001.

[8] Fuentes, V. K. Assigning Teaching Assistants to Courses: Mathematical Models. Honor's Thesis, University of California - Davis, *Retrieved from ProQuest Dissertations and Theses*, 2014.

[9] Dinkel, J. J.; Mote, J.; Venkataramanan, M. A. An Efficient Decision Support System for Academic Course Scheduling, *Op. Res*. 37 (6), 1989, 853-864

[10] Güler, M. G.; Keskin, M. E.; Döyen, A.; Akyer, H. On teaching assistant-task assignment problem: A case study, *Comp. Ind. Eng*. 79, 2015, 18-26

[11] Caramia, M.; Dell'Olomo, P. Multi-Objective Management in Freight Logistics Increasing Capacity, Service Level and Safety with Optimization Algorithms, Springer. 2008, 13-32.

[12] Schrijver, A. On the History of Combinatorial Optimization (Till 1960), Discr. Opt. 2005, 12, 1-68.

[13] Kuhn, H. W. A tale of three eras: The discovery and rediscovery of the Hungarian Method, Eur. J. Op. Res. 219, 2012, 641-651

Chandra Mouli R. Madhuranthakam is an assistant professor in the department of chemical engineering at the Abu Dhabi University, Abu Dhabi, United Arab Emirates. He earned B.Tech in chemical engineering from Sri Venkateswara University, India Master's in chemical engineering at the Indian Institute of Science, Bangalore, India and PhD in chemical engineering University of Waterloo, Canada. He is a member of the American Institute of Chemical Engineers and the Institute for Operations Research and Management Sciences (INFORMS).

Mukhtar Al-Ismaily is a graduate student in chemical engineering at the University of Waterloo, Canada.

Ali Elkamel is a professor in the department of chemical engineering at University of Waterloo, Waterloo, Canada. He earned his B.S. and M.S. in chemical engineering at Colorado School of Mines and PhD in chemical engineering at Purdue University. Ali Elkamel is also a visiting Professor in the Department of Chemical Engineering at Khalifa University, Abu Dhabi, UAE. He is a member of the Canadian Society for Chemical Engineering, the American Institute of Chemical Engineers, the Canadian Operational Research Society, and the Institute for Operations Research and Management Sciences (INFORMS).