A Heuristic Algorithm for Solving the Faculty Assignment Problem

Manar I. Hosny

1College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia
mifawzi@ksu.edu.sa

Abstract—In this paper we describe a heuristic-based algorithm for the faculty assignment problem. The algorithm was designed and implemented as part of a graduation project in the IT department of King Saud University, Riyadh, Saudi Arabia. The purpose of the project was to automate the manual tedious process of assigning TAs (Teaching Assistants) to suitable lab sessions; a task that needs to be done every semester in the IT department. The algorithm was designed to fulfill the assignment task while taking into consideration a number of constraints pertaining to both the TAs (e.g., personal preference and available hours) and the IT department (e.g., maximum allowed workload). The problem definition, the algorithm, and the experimental results are thoroughly explained in this paper. The computational experimentation indicate the efficiency of the proposed algorithm in assigning courses that conform to the preferences of the TAs and the rules set forth by the IT department.

Keywords: The Faculty Assignment Problem, Scheduling, Timetabling, Heuristics, Meta-heuristics

1. Introduction

The faculty assignment problem is one type of scheduling problems that almost every educational institution needs to handle periodically. The problem is concerned with assigning courses to staff members, while a number of constraints are adhered with. For example, the assignment should take into consideration the preferences of staff members, their availability, and their permitted workload. Many institutions still perform this task manually, which often requires a lot of time and effort. In addition, the resulting staff schedules are usually inefficient in terms of not satisfying the preferences of all staff members, or the uneven distribution of workload among them. Automating the planning and scheduling tasks has great benefits in reducing the workload of the planner. In addition, creating balanced and fair schedules help to satisfy faculty members and reduce the stress among them, which may encourage them to stay in the academic career.

The faculty assignment problem is a special case of a wider class of scheduling problems that take many shapes and forms. For example, there are exams scheduling, courses scheduling, nurse rostering, scheduling of sports and business events, etc [1]. Scheduling problems belong to the class of NP-hard problems [2] for which there is no polynomial time algorithm that exists to solve the problem to optimality. In addition, the presence of a number of constraints makes the problem even more difficult to solve. In general constraints are of two types: hard constraints and soft constraints [3] [4]. Hard constraints are those that must be strictly enforced in the solution, otherwise the solution will be infeasible. For instance, the condition that a faculty member cannot teach two sessions at the same time is a hard constraint. Some hard constraints may be specific to the institution. For example, one institution may probably require that a staff member cannot teach more than three sections a day. Soft constraints, on the other hand, are desirable but they do not make the solution infeasible. For example, giving staff members courses within their wish list is a soft constraint. Given the difficulty of solving this problem, due to the often conflicting constraints, heuristic and meta-heuristic algorithms are usually used to provide a good problem solution in a reasonable amount of time.

In this research we consider the faculty assignment problem that is specific to the Information Technology (IT) department, of the College of Computer and Information Sciences in King Saud University (KSU), Riyadh, Saudi Arabia. We developed a heuristic algorithm to solve this problem based on the constraints specified by the IT department. The algorithm was integrated with a web-based tool having a centralize database. The system as a whole was a 2-semester graduation project of five students in the IT department. The preliminary system analysis and design phases of the project, prior to the implementation phase, were described in a virtual brief paper [5]. In this paper, we mainly focus on the heuristic algorithm, which is the core of the implemented tool, describing and analyzing its results and performance after the end of the implementation phase of this project.

The rest of this paper is organized as follows: Section 2 is a brief description of some related work. Section 3 explains the problem that we are trying to handle in this research, by describing the context in which it is applied and the problem constraints that should be adhered with. Section 4 describes in detail the heuristic algorithm used in solving the underlying problem. Section 5 presents and discusses the experimental results of the algorithm. Finally, Section 6 provides some concluding remarks and some thoughts for future directions.

2. Related Work

Solutions to the faculty assignment problem are based on the specific problem under consideration. Every institution
has certain constraints that should be fulfilled when assigning faculty members to courses. Thus, the quality of the solution is very much subjective: a good assignment for one institution may be a poor one relative to another institution. Accordingly, techniques for solving this problem were mostly developed to fit the needs of specific institutions, under clearly pre-determined conditions.

[6] presents a preference-based decision making approach for the faculty assignment problem, which takes into consideration faculty preferences, students preferences and administration regarding instructor-course assignment, and other organizational constraints, such as available facilities and classroom resources are also considered in this model. The optimization process is divided into two stages: class-course scheduling and class-faculty assignment. Two fitness functions are designed to evaluate the satisfaction of the teachers and the students. The heuristic approach is based on iterative mutation, which either changes the time slot assigned to a certain course, or changes the teacher assigned to a certain course.

In the following sections, we explain in detail the research conducted in the project under consideration.

3. Problem Definition

As previously mentioned, the problem handled in this research was specifically defined according to the needs of the IT department at KSU. Specifically, we were concerned with assigning TAs (Teaching Assistants) to suitable lab sessions, taking into consideration a number of constraints to be explained shortly. In the IT department, there are approximately 12 courses which are associated with lab sessions. Each course has a number of sections (usually ranging from 1-6), and each section has its own lab session, whose time and date has been pre-determined by the department. Weekdays are from Saturday to Wednesday. Each lab sessions runs for 2 hours, and there are 3 possible lab time slots per day: 8-10, 10-12, and 1-3. Due to a large number of students enrolled, each lab session is assigned to two TAs, who should both attend the lab session concurrently. The TAs in the department are of two types:

1) **Full-Time TAs**: whose maximum permitted workload is 16 hours per week.
2) **Masters Students TAs**: whose maximum permitted workload is 8 hours per week. In addition, those TAs have to attend their Masters lectures, i.e., they have a limited availability that should be considered when lab sessions are assigned to them.

The TA Assignment Committee (TAAC) of the IT department is responsible for assigning suitable TAs to all lab sessions. To do this, the TAAC first asks all TAs in the department to fill in a "wish list", in which they indicate three courses that they like to teach in order of preference. In addition, Masters students TAs should indicate their busy hours (i.e. Masters lectures), during which they cannot teach lab session.

We can define the constraints of the above problem as follows:

- **Hard Constraints**:
  1) To assign a lab session to a certain TA, the lab’s date and time must not conflict with the TA's availability, i.e., she should not be busy during the time slot of the lab (e.g. taking a Masters lecture or teaching another lab).
2) Each lab session should be assigned a maximum of two TAs.
3) The maximum permitted workload of the TA cannot be exceeded.

- **Soft Constraints:**
  1) The preferred courses for the TA should be taken into consideration, in order.
  2) Each lab session should be assigned to at least one TA.
  3) The distribution of workload should be as fair as possible.

The algorithm developed in this research will help facilitate the task of the manual generation of schedules for the TAAC in the IT department. The algorithm will generate schedules for the TAs automatically, based on the information and the constraints provided by both the TAs and the TAAC. The steps of the algorithm are described in detail in the following section.

4. The Faculty Assignment Algorithm

The faculty assignment heuristic algorithm will try to assign courses that conform to the preferences of the TAs, their available times, and their permitted workloads. The assignment process will be as fair as possible in terms of the distribution of the workload. Specifically, older Masters students (based on the year they are currently enrolled in) are given higher priority in the assignment of their preferred courses. Second priority is given to Masters students with higher GPA. For the full-time TAs, the older ones (according to their employment date) are given priority in assignment. However, during the processing of the algorithm, priorities of assignment change adaptively, depending on the currently assigned working hours of each TA. This is intended to increase the number of assigned working hours for underutilized TAs.

The input to the algorithm will be information related to a number of TAs and a number of lab sessions:

- For each TA: a list of 3 preferred courses, available working hours, maximum allowed working hours, and the number of assigned working hours (initially 0).
- For each course: course name, section number, day, time, number of assigned teachers (maximum 2, initially 0), and the list of assigned teachers (initially empty).

The output of the algorithm will be the automatically generated labs schedule after the assignment of each lab session to two TAs, if possible, in addition to the total working hours assigned to each TA by the algorithm. If no solution can be found, i.e., the algorithm cannot assign all lab session to suitable TAs, the algorithm will display a message informing the TAAC that some labs were not assigned to two TAs, in which case the schedule must be adjusted manually. Thus the main objective of the algorithm is to maximize the number of assignments of TAs to courses. The objective is fulfilled when each lab section is assigned to two TAs, i.e., the maximum number of assignments = \(2 \times NumberOfSections\).

The algorithm will first sort the TAs according to their studentship status, i.e., Masters students are placed at the top of the list to give them priority in assignment. Within the list of Masters students, older students are preferred, followed by high GPA students. Regarding full-time TAs, they are sorted according to employment date; such that older TAs in the department are given priority.

The algorithm then proceeds in two phases:

**Phase 1**: will iterate 3 times (once for each preference). In each iteration, the algorithm will try to assign to every TA in the sorted list the current preference under consideration, subject to the following:

- Her preferred lab is not already assigned to two TAs;
- The lab day and time does not conflict with her availability;
- And her workload will not be exceeded after assigning the lab to her.

At the end of each iteration (i.e., before progressing to the next preference in order), the sorted list of TAs will be reordered by giving priority to TAs with the least number of assigned working hours. If there are still lab sessions that are not assigned to two TAs each, the second phase will start.

**Phase 2**: This phase will start by first initializing the best so far schedule to the schedule generated in phase 1. It will then use a Hill Climbing (HC) optimization approach to improve this schedule. First, the list of TAs will be reordered, according to the ratio of assigned working hours to the permitted workload. Thus, priority is given to underutilized TAs. Then, the algorithm will proceed similar to phase 1, but instead of assigning each TA to her preferred lab session, it will try to assign to each TA a random non-assigned lab session, subject to the same conditions mentioned above. This process will be repeated until one of the two conditions is satisfied: (1) all courses have been assigned to two TAs; (2) a maximum number of trials (e.g., 10) has been attempted.

During the HC process, each new schedule generated will be compared with the best so far schedule. If the new schedule is better, in terms of reducing the number of unassigned courses, it will replace the best so far schedule. The final best schedule will be returned as the output of the algorithm. In case there are still lab sessions that could not

---

\(^1\)We will use the terms courses and labs interchangeably since we only consider assignment of lab sessions associated with courses in this research.

\(^2\)Manual adjustment is allowed through the interface of the web-based tool.
be assigned after the end of the optimization process, the algorithm will display a warning message so that the TAAC will take an appropriate action.

The steps of Phase 1 of the algorithm is shown in Algorithm 1, while the steps of Phase 2 are shown in Algorithm 2.

Algorithm 1 Faculty Assignment Algorithm - Phase 1

1: create \textit{TAList}, the sorted list of all TAs
2: totalAssignments = numSections * 2
3: numAssignments = 0
4: prefIndex = 0
5: currentSchedule = empty
6: while (numAssignments < totalAssignments) AND (prefIndex < 3) do
7: prefIndex += 1
8: for (curT A \in TAList) do
9: prefLab \leftarrow Labs[prefIndex]
10: if (prefLab not already assigned to two TAs) AND (prefLab time slot does not conflict with curT A’s availability) AND (curT A’s workload will not be exceeded after assigning prefLab to her) then
11: Assign curT A to prefLab
12: numAssignments += 1
13: Increment number of TAs assigned to prefLab
14: Update availability of curT A to reflect the new assignment
15: Increase assigned working hours of curT A
16: Update currentSchedule to reflect the assignment
17: Re-sort TAList in ascending order of assigned working hours

Algorithm 2 Faculty Assignment Algorithm - Phase 2

1: if (numAssignments < totalAssignments) then
2: numAttempts = 0
3: MAXATTEMPTS = 10
4: bestSchedule = currentSchedule
5: repeat
6: numAttempts ++
7: create newTAList, the new sorted list of all TAs {sort TAs according to the ratio workingHours/workLoad, lowest ratio first}
8: Stop = False [The loop will stop if no successful assignments can be found for all TAs in the last pass]
9: while (numAssignments < totalAssignments) AND (Stop == false) do
10: for (curT A \in newTAList) do
11: randomLab \leftarrow Labs[randomIndex]
12: if (randomLab not already assigned to two TAs) AND (randomLab time slot does not conflict with curT A’s availability) AND (curT A’s workload will not be exceeded after assigning randomLab to her) then
13: Assign curT A to randomLab
14: numAssignments += 1
15: Increment number of TAs assigned to randomLab
16: Update availability of curT A to reflect the new assignment
17: Increase assigned working hours of curT A
18: Update currentSchedule to reflect the assignment
19: Stop = False [found a successful assignment this iteration]
20: else
21: Stop = true
22: if (currentSchedule is better than bestSchedule) then
23: bestSchedule = currentSchedule
24: until (numAssignments == MAXATTEMPTS) OR (numAssignments == totalAssignments)
25: Return bestSchedule
26: if (numAssignments < totalAssignments) then
27: display warning message

5. Results and Discussion

The algorithm was implemented using the Ruby language on an Intel(R) 1.73 GHz processor, and a 2.00 GB RAM. It was tested on 10 test cases using data obtained from the IT department. In 8 out of the 10 test cases there are 30 lab sections and 15 TAs (10 Masters and 5 full-time). To test the performance of the algorithm in some special circumstances, where there is a fewer number of sections or a fewer number of TAs, test case 9 had only 10 lab sections, and test case 10 had only 10 TAs (6 Masters and 4 full-time). In addition, in test cases 2 and 3, we tried the case where some TAs (4 out of 15) did not choose any preferences.

To measure the performance of the algorithm, we recorded the number of times the algorithm was able to assign to each TA a course among her wish list, in addition to the total number of assigned working hours. We also calculated the total number of assignments of TAs to sections (remember that our goal is to assign each section to two TAs under the given constraints). Table 1 shows a summary of the results of the 10 test cases. The first column indicates the number of the test case. Columns 2-5 show the percentage of preference 1, preference 2, preference 3, and non-preference (preference 0) assignments among all TAs in the corresponding test case. Columns 6-7 show the average assigned working hours for Masters and full-time (regular) TAs respectively. Finally column 8 shows the average processing time of each test case. The final row of the table calculates the overall average of each corresponding column. Test cases 2, 3, 9 and 10, which have some special conditions as explained above, are shown in italic.

Fig. 1 shows the percentage of assignments achieved by the algorithms compared to the total number of assignments required for the 10 test cases. For example in case 1, there

\[ \text{Algorithm 1 Faculty Assignment Algorithm - Phase 1} \]

\[ \text{Algorithm 2 Faculty Assignment Algorithm - Phase 2} \]
Table 1: Algorithm Results

<table>
<thead>
<tr>
<th>Case</th>
<th>Pref 1</th>
<th>Pref 2</th>
<th>Pref 3</th>
<th>Pref 0</th>
<th>Avg MSc Hrs</th>
<th>Avg Reg Hrs</th>
<th>Proc Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>73%</td>
<td>53%</td>
<td>60%</td>
<td>27%</td>
<td>7.2</td>
<td>8.4</td>
<td>0.76</td>
</tr>
<tr>
<td>2</td>
<td>67%</td>
<td>33%</td>
<td>47%</td>
<td>87%</td>
<td>6.6</td>
<td>9.2</td>
<td>0.98</td>
</tr>
<tr>
<td>3</td>
<td>60%</td>
<td>40%</td>
<td>53%</td>
<td>73%</td>
<td>7</td>
<td>9.2</td>
<td>0.74</td>
</tr>
<tr>
<td>4</td>
<td>67%</td>
<td>47%</td>
<td>53%</td>
<td>13%</td>
<td>6.6</td>
<td>9.6</td>
<td>0.9</td>
</tr>
<tr>
<td>5</td>
<td>60%</td>
<td>40%</td>
<td>53%</td>
<td>73%</td>
<td>7</td>
<td>9.2</td>
<td>0.74</td>
</tr>
<tr>
<td>6</td>
<td>67%</td>
<td>53%</td>
<td>33%</td>
<td>47%</td>
<td>6.6</td>
<td>9.6</td>
<td>0.9</td>
</tr>
<tr>
<td>7</td>
<td>67%</td>
<td>67%</td>
<td>33%</td>
<td>47%</td>
<td>7</td>
<td>8.4</td>
<td>0.67</td>
</tr>
<tr>
<td>8</td>
<td>67%</td>
<td>53%</td>
<td>40%</td>
<td>40%</td>
<td>6.6</td>
<td>9.6</td>
<td>0.93</td>
</tr>
<tr>
<td>9</td>
<td>53%</td>
<td>40%</td>
<td>27%</td>
<td>13%</td>
<td>2.8</td>
<td>2.4</td>
<td>0.38</td>
</tr>
<tr>
<td>10</td>
<td>70%</td>
<td>50%</td>
<td>40%</td>
<td>0%</td>
<td>7</td>
<td>12.5</td>
<td>0.97</td>
</tr>
<tr>
<td>Average</td>
<td>66%</td>
<td>47%</td>
<td>43%</td>
<td>37%</td>
<td>6.38</td>
<td>8.89</td>
<td>0.81</td>
</tr>
</tbody>
</table>

The following can be deduced from Table 1 and Fig. 1:

1. The preferred courses may be conflicting with her availability.
2. She may have been assigned one or more of her preferred courses, but her total number of assigned hours is still small compared to other TAs.
3. Her preferred course(s) may have been already assigned to other TAs who have a higher priority in the TAs’ list (e.g., a graduating Masters student, or an older TA).

With the exception of the special cases 9 and 10, those having a fewer number of sections and a fewer number of TAs respectively, the algorithm was very stable in the number of hours assigned to the TAs. For Masters students TAs, the average assigned hours ranged from 6.2 to 7.2. For full-time TAs, the average assigned hours ranged from 8.4 to 10.

Again, with the exception of cases 9 and 10, full-time TAs were assigned, on average, approximately 27% more working hours than Masters students TA. This is a reasonable assignment, given that the number of sections tried in this experiment is not very large. We expect that if there was a larger number of sections, the assigned working hours of full-time TAs could be as much as 50% more compared to Masters students TAs. This assignment would then conform to the conditions of the maximum working hours mentioned in Sect. 3.

The processing time of the algorithm was very fast. In all test cases the resulting schedule was produced in less than 1 second. The average processing time is 0.8 seconds.

We can observe from Fig. 1 that the algorithm was able to achieve a very high percentage of fulfilling the required sections assignments. With the exception of test case 10, in which there was only 10 available TAs, the percentage of assignments ranged from 95% to 100%. The low percentage of assignment in test case 10 is expected, since there is a shortage in the number of TAs that can be assigned to the available courses. On the other hand, the 100% assignment rate in test case 9 was also expected since the number of courses in this test case was fewer than the other test cases. Having said this, it is our conjecture that the algorithm can even achieve higher assignment rates, if we increase the maximum number of attempts (currently set at only 10). This can be easily done since the algorithm is quite fast and a small increase in processing time will not degrade its performance.

6. Conclusions

This research is based on a BSc graduation project in the IT department of King Saud University, Riyadh, Saudi Arabia. The project is a web-based tool with an integrated database designed to help the TA assignment committee in the IT department to automatically assign TAs to lab...
sessions, while fulfilling a number of difficult constraints. The constraints included the preferences of the TAs, their availability and their maximum allowed workload. The core of the tool, which is the theme of this paper, is a 2-phase heuristic algorithm that deals with the above constraints and tries to achieve balanced schedules by dynamically changing the priorities in assignment, based on the current progress of the algorithm.

The experimental results indicate that the algorithm was very successful in producing the anticipated results in terms of satisfying the preferences of the TAs and balancing the workload among them. In addition, the algorithm was very fast, since the schedule was always generated in less than 1 second of time.

Although the algorithm in this project was specifically designed to meet the requirements of the IT department, it is a robust algorithm that can be applied in several other domains. For example, with just minor modifications in the problem settings, the algorithm can be used to assign courses to different faculty members, not only TAs. The framework of the algorithm is also general enough to allow the modification of constraints, based on the specific needs of a department or institution, without changing the whole algorithm.

It remains to test the tool in realistic settings to measure the satisfaction of users and decide whether it can completely or partially replace the tedious manual planning process.

Acknowledgment

Many thanks to Wijdan Al-Dawood, Ashwaq Al-Rudyan, Hind Al-Harbi, Sara Al Hayash and Noorh Al-Sfyan for their effort in designing the web-based tool and implementing and testing the algorithm described in this paper.

References