The TA Assignment Problem with Open Tutorials

Hans Aisake, Issac He, Mark Strange
Department of Mathematics, Simon Fraser University, Surrey BC

Abstract

We consider the problem of scheduling teaching assistants (TAs) to an open tutorial lab. Open tutorials provide an integrated framework of offering tutorials for multiple courses. Hence, assignment of teaching assistants raises several optimization challenges. We formulate the TA assignment problem as a minimum-cost network flow model with additional restrictions to distribute the tutorial workload among several TAs. Factors such as TA availability, course schedules, and expected student turnout are taken into account. The model takes advantage of certain jobs that do not require specific time scheduling. The output of the model yields a solution that can reach the most students possible. Furthermore, since these are generally small problems, solutions can be found in a timely manner using a general purpose integer programming solver.

1 Introduction

Open Tutorials are a type of tutorial format where students and tutorials are not assigned to each other. Instead, tutorials are held in an open room where students can arrive at any time. The problem with this structure is scheduling teaching assistants (TAs) in a way such that they can help all students that arrive. With a large number of TAs, a TA can always be present in the room. However, with a small number of TAs this is not possible. Moreover, not all TAs will handle all the courses. Thus efficient allocation of TAs to group of courses and to appropriate time slots is a difficult problem. To the best of our knowledge, this problem has not yet been studied in literature although, university course scheduling is studied extensively in literature.

The purpose of this research project is to investigate if the *Open Tutorial TA assignment problem* (OTTA) can be modeled and solved with an integer linear program. Üney-Yüksektepe and Karabulut [3] used a mixed integer linear programming model to assign TAs to tutorials, but their tutorials are not open tutorials. Daskalaki, Birbas, and Housos [2] use a binary integer program to solve a university course timetable problem. Al-Husain, Hasan, and Al-Qaheri [6] implement a multi-stage integer goal program to solve a similar university course timetable problem. And Osman, Balola, Ali Yahya, and Ali Abdelrahman [1] use a genetic algorithm for solving a university course timetable problem. University course scheduling is a well studied area with a large amount of literature. These models however is not applicable to our problem.

2 The Problem

We seek to derive a schedule for TAs to work in the Open Tutorial room (open lab). In addition, TAs have other tasks they need to perform, such as: marking assignments, marking exams, and exam proctoring. Several factors must be considered: student availability, TA availability, TA requested workload, and open lab operating time.

The ultimate goal of any tutorial is to help students, and the Open Tutorial is no different. With this in mind, student availability is a very important factor. There are various ways to gather information on student availability. For example, looking at each student's course schedule or conducting a survey, can yield student availability information. This information can be used to construct a weight function which correlates student demand with specific time slots. We assumed that an appropriate weight function can be found, where large weights are "good" times, and "small" weights are bad times. A further restriction on the weight function is that is must be positive. In addition, it is preferable to be integer-valued.

In the open lab, there are teaching assistants providing student help, as well as professors providing office hours. With a limited number of personnel, it is not preferable to have many people in the open lab at the same time. However, if student demand is high enough, overlap between schedules is allowed. We do not consider room size limitations: this is justified since the problem only considers a small number of TAs.

3 Formulation of the Model

We formulate the OTTA problem as a minimum-cost network flow. Each TA is represented by a source node, with supply equal to the number of half-hours of requested workload in a semester. The different jobs that must be assigned, (assignment and exam marking, exam proctoring, etc.), are represented by sink nodes with demand equivalent to the number of half-hours needed to complete the job. These sink nodes are classified as *fixed-time* jobs, since their demand must be completely satisfied and is invariant. Some TAs can be designated *special*, and assigned to only do marking tasks; whereas other TAs receive no special designation, and can be assigned to any task.

The arcs from each TA node to the fixed-time job nodes have cost 0 and capacity related to the number of TAs available to do that job, taking into account TAs with *special* designation. The capacity is bounded this way in order to provide a fair distribution of fixed-time job workload. Assignment marking arcs are not given a capacity, since these are handled by TAs in their own time.

The open lab is represented by a single sink node with infinite demand, so it can take all left over supply from TAs. Each TA has a series of arcs to the open lab node with unit capacity. Each arc represents a specific half-hour time block available to work in the open lab. These arcs are assigned corresponding values from the weight function, which represent the availability of students in the open lab.

Below is shown a simplified version of the network, with one TA visible. Arc labels are (cost, capacity).

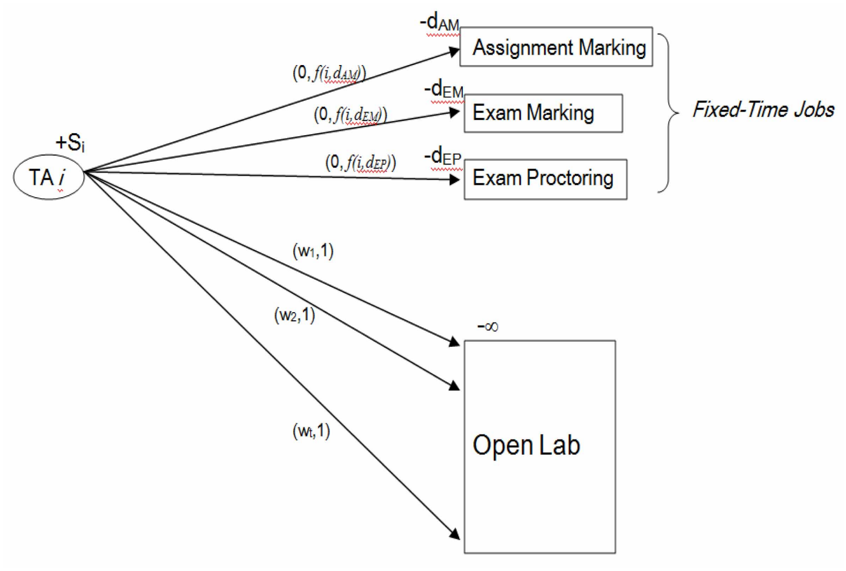

Figure 1: Simple graphic depiction of the network

3.1 Assumptions

- **Every TA is equally competent at every job**
  We can assume this because TAs are generally graduate students, and should be able to competently tutor first-year mathematics students.

- **Every TA has already taken into account travel time, etc. in their given availability schedule**
  Solving for each TA's individual travel time is needlessly complicated and unnecessary. TAs can be informed to take travel time into account in their availability schedules.

- **The time needed to mark assignments, midterms, and finals is fixed**
  Even though it is possible that the time needed to mark midterm and final examinations will overflow, this is assumed to be an exceptional occurrence.

- **If midterm and final marking must be done in a specific room, we assume the booking and staffing of these rooms is handled externally**
  The department supervisor is assumed to be able to find and book rooms for marking midterm and final exams, since adding room availability schedules to the model is very complicated.

- **All assignment marking is assumed to fit into a TA's schedule**
  TAs are generally required to mark assignments on their own time, and we do not need to schedule the exact time that they will do this work. We need only allot a sufficient number of hours worth of pay to account for this job. This is in part because assignments are easy and quick to mark, and TAs have all evenings and weekends as possible work time.

- **The number of assignments and midterms in a class does not change over the semester**
  Professors are assumed not to add extra midterms and assignments during the semester, since they will have a set course plan for the term.

The following assumptions are implemented for the sake of simplification, and can be removed for an updated model

- **All exam marking and proctoring is handled separately, and we don't worry about fitting these events into a schedule**
  Accounting for room scheduling and professor availability in the model adds a lot of complexity, and does not add much to improving the schedule solution. Midterms and exams are rare events, and room booking can be done externally without much difficulty. In the event that one knows the day and time midterms will be marked

beforehand, the model can account for this by giving these times 0 or negative weight.

- **TAs will be present for all assigned open lab hours**
  We do not account for sudden sickness or other unscheduled reasons for absence in the assigned TA schedules.

- **Midterm marking for each course will be done by all TAs and at least one professor**
  There should be assigned an equal number of midterm marking hours to each TA in a workshop. Since personnel is limited, this allows exams to be marked quickly.

3.2 Notation, Parameters, and Variables

t = index for time: counting in order all half-hour time slots to allocate.

g = specific values of index t corresponding to the start of a day.

k = index for the days of the week. $k = \{0, 1, \ldots K-1\}$, where $K \leq 7$ is the number of days in a week where the open lab is open.

h_0 = the number of half-hour time slots in one day.

i = index for each T.A. $i = \{0, 1, \ldots, I\}$, where I is the total number of TAs.

c = index for each course. $c = \{0, 1, \ldots, C\}$, where C is the total number of courses serviced by the Open Tutorial.

l = index for a type of fixed-time TA job (a job other than tutoring in the open lab).

S_i = the number of half-hour blocks that TA i can work in a semester, based on their requested workload.

λ_c = the number of students in course c.

α_c = the number of assignments course c will have in the semester.

β_c = the number of midterms class c will have in the semester.

$\vec{m} = (m_A, m_M, m_F)$ a vector of alloted minutes for a TA to mark one assignment, midterm, or final.

ϕ = the number of *special* TAs, only able to do marking tasks.

γ = the number of all other TAs, able to do all possible tasks.

d_l = the number of half-hours needed to complete all instances of fixed-time job l over the entire semester.

w_t = the weight function, proportional to the number of potential students that can be helped in the open lab at time t.

μ_i^l = the upper time bound for TA i to give to job l, to enforce a fair distribution of work.

$$y_{it} = \begin{cases} 1 & \text{if TA } i \text{ is available at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$z_t = \begin{cases} 1 & \text{if a professor has office hours at time } t \\ 0 & \text{otherwise} \end{cases}$$

Variables:

f_i^l = the number of half-hours that TA i is assigned to doing a fixed-time job l.

$$x_{it} = \begin{cases} 1 & \text{if TA } i \text{ is assigned to the open lab at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$V_{ig} = \begin{cases} 1 & \text{if TA } i \text{ works day starting at } g \\ 0 & \text{otherwise} \end{cases}$$

$$R_{ig} = \begin{cases} 1 & \text{if TA } i \text{ is assigned to work all} \\ & \text{occurrences of the day of the week } g \\ 0 & \text{otherwise} \end{cases}$$

3.3 Constraints

1. All fixed-time jobs must be complete.

$$\sum_i f_i^l \geq d_l \qquad \forall l$$

2. Use as many available TA half-hours as possible.

$$\sum_l f_i^l + \sum_t x_{it} \leq S_i \qquad \forall i$$

3. There should be a TA or a professor in the open lab every day.

$$\sum_{t=g}^{g+h_0} \left(\sum_i x_{it} + z_t \right) \geq 1 \qquad \forall g$$

4. Each TA's availability must be satisfied when assigning them to the open lab.

$$x_{it} \leq y_{it} \qquad \forall i, t$$

5. No TA should be in the open lab at the same time as another TA, or when a professor is holding office hours there.

$$\sum_i x_{it} \leq 1 - z_t \qquad \forall t$$

6. The number of open lab hours for each TA in a day, should be equal for the same day in the following week.

$$\sum_{t=g}^{g+h_0} x_{it} = \sum_{t=g+(K)h_0}^{g+(K+1)h_0} x_{it} \qquad \forall i, g$$

7. If TA i is set to work on day starting at g, then they must work between one and six half-hour segments over that day.

$$V_{ig} \leq \sum_{t=g}^{g+h_0} x_{it}$$

$$\sum_{t=g}^{g+h_0} x_{it} \leq 6V_{ig} \qquad \forall i, g$$

8. Ensure TA i works the same days in consecutive weeks.

$$\sum_{g \text{ lies on day } k} V_{ig} = LR_{ik} \qquad \forall i, k$$

   Where L is the number of weeks in a semester.

9. The variables x_{it} are binary.

$$x_{it} \in \{0, 1\} \qquad \forall i, t$$

10. The variables f_i^l are integer.

$$f_i^l \in \{0, 1, 2, \dots\} \qquad \forall i, l$$

4 Model

A summary of the network flow model in mathematical notation. The objective is to maximize the number of potential students helped in the open lab.

$$\max \sum_i \sum_t w_t x_{it}$$

Subject to:

$$\sum_i f_i^l \geq d_l \qquad \forall l$$

$$\sum_l f_i^l + \sum_t x_{it} \leq S_i \qquad \forall i$$

$$\sum_{t=g}^{g+h_0} \left(\sum_i x_{it} + z_t \right) \geq 1 \qquad \forall g$$

$$x_{it} \leq y_{it} \qquad \forall i, t$$

$$\sum_i x_{it} \leq 1 - z_t \qquad \forall t$$

$$\sum_{t=g}^{g+h_0} x_{it} = \sum_{t=g+(K)h_0}^{g+(K+1)h_0} x_{it} \qquad \forall i, g$$

$$V_{ig} \leq \sum_{t=g}^{g+h_0} x_{it}$$

$$\sum_{t=g}^{g+h_0} x_{it} \leq 6V_{ig} \qquad \forall i, g$$

$$\sum_{g \text{ lies on day } k} V_{ig} = LR_{ik} \qquad \forall i, k$$

Where L is the number of weeks in a semester.

$$x_{it} \in \{0, 1\} \qquad \forall i, t$$

$$f_i^l \in \{0, 1, 2, \ldots\} \qquad \forall i, l$$

5 Case Study

One of the motivations for this project was to develop a model to solve the OTTA problem at Simon Fraser University's Surrey campus. To test the derived model, we obtained a set of sample data for the Spring 2012 semester at SFU Surrey. This study shows that a reasonable solution is obtained using this real data.

The mathematics department at SFU Surrey subdivides the open lab into three distinct workshops: Intro Math, Pure Calculus, and Applied Calculus. Each workshop is dedicated to a specific and independent subset of first- and second-year math courses. TAs for the open lab are preassigned to one of the three workshops, and are not allowed to cross between the different workshops. Because of this, the entire model network can be partitioned into three independent and structurally identical networks. These networks are then solved separately. In addition to open lab tutoring, TAs at SFU Surrey are also required to do assignment, midterm, and final exam marking, as well as midterm and final exam proctoring. These fixed-time jobs are indexed by $l \in \{AM, MM, FM, MP, FP\}$, respectively. We assume that all TAs in a given workshop will be present for midterm and final exam proctoring for any courses covered in their workshop. This is a pattern we observed in already existing schedules for the open lab.

The open lab at SFU Surrey operates from 9:00AM to 4:30PM, Monday through Friday, for all 16 weeks during a semester. This gives the possible values of the time index, $t = \{0, 1, \ldots, 1199\}$. The interval $[15n, 15(n+1))$ for $n = 0, 1, \ldots, 79$ is one day, while $[75n, 75(n+1))$ for $n = 0, 1, \ldots, 15$ is one week. This also shows that g indicates the start of a day whenever $t \bmod 15 = 0$.

$C \leq 4$ is an upper bound on the total courses in one workshop. The workload of each TA is provided in Basic Units (BUs), which is converted to half-hour time blocks: $S_i = \lfloor (\text{BUs of TA } i)(41)(2) \rfloor$. Alloted minutes for a TA marking are $\vec{m} = (2, 6, 12)$, for assignment, midterm, and final, respectively.

5.1 Input

We provide the data only for the Applied Calculus workshop, for simplicity.

Table 1: Course Information

Course Name (c)	Students (λ_c)	Assignments (α_c)	Midterms (β_c)
Math 155	97	10	2
Math 157	68	10	2
Math 232	83	10	2

Table 2: Anonymized TA Data

TA Name	Designation	Requested BUs	Half-hours
SL	Normal	5.3	434
BB	Normal	3.8	311
FD	Normal	3.02	247
T	Normal	2.3	188
H	*Special*	2	164
Y	*Special*	1.3	106

Consequently, the number of half-hours needed to complete the fixed-time job l is (taking into account *special* TAs):

$$d_l = \begin{cases} \frac{2}{30}\sum_c \alpha_c\lambda_c = 166 & l = AM \\ \frac{6}{30}\sum_c \beta_c\lambda_c\frac{\gamma+\phi}{1+\gamma+\phi} = 86 & l = MM \\ \frac{12}{30}\sum_c \lambda_c\frac{\gamma+\phi}{1+\gamma+\phi} = 86 & l = FM \\ \frac{60}{30}\sum_c \beta_c\gamma = 48 & l = MP \\ \frac{180}{30}C\gamma = 72 & l = FP \end{cases}$$

And upper time bounds for TA i to give to job l are given by:

$$\mu_i^l = \begin{cases} \frac{d_l}{\gamma+\phi} = 15 & l \in \{MM, FM\} \\ \frac{d_l}{\gamma} = 12 & l = MP \\ \frac{d_l}{\gamma} = 18 & l = FP \end{cases}$$

The model is implemented using Microsoft Excel, and the OpenSolver plugin [7].

When running the model, we encountered a twofold issue: a TA schedule would either have several time gaps in a day's work, or several TAs would alternate working in one contiguous segment of time (Note the Tuesday, Thursday, and Friday timetables in Figure 2). Since every TA is assumed to be equally competent, the only difference should be their availability. These issues can then be manually adjusted with some difficulty after the solution is found. However, we sought an adjustment to the model that would force these issues to resolve themselves during processing. The solution we found was to introduce another constraint: limit the number of TAs present in the open lab in a single day.

Setting the upper bound on number of TAs per day to either 4 or 3 did not affect the schedule issues significantly. The resulting schedules still proved difficult to manually adjust after the solution was found.

When the upper bound is lowered to 2 TAs per day, an excess of unassigned time occurs for some TAs in the solution:

The unassigned time for these three TAs is not necessarily an unwanted effect. This extra time can be treated as a sort of slack, and manually reassigned for special events such as midterm and final extra tutoring. The weekly schedules also show reduced gaps, and the limited number of TAs in a day prevents excessive alternating TAs in a schedule.

Figure 2: Sample of a Poor Schedule

Time	Monday	Tuesday	Wednesday	Thursday	Friday
9:00-9:30	BB	FD	SL	SL	SL
9:30-10:00		SL		SL	
10:00-10:30		T		T	BB
10:30-11:00	SL	T	BB	T	FD
11:00-11:30	BB	SL	BB	T	SL
11:30-12:00		SL		SL	BB
12:00-12:30		T		SL	BB
12:30-1:00	BB	FD	SL	SL	SL
1:00-1:30	BB	SL	SL	T	FD
1:30-2:00	BB	SL	SL	T	BB
2:00-2:30		FD	BB	FD	BB
2:30-3:00		FD	BB	FD	BB
3:00-3:30		SL	BB	FD	SL
3:30-4:00		FD		SL	SL
4:00-4:30		T		FD	SL

Table 3: Unassigned Time for 2 TAs per Day

TA Name	Unassigned Half-Hours
SL	24
BB	0
FD	8
T	36

What issues do remain are small enough that they can be manually adjusted.

When the upper bound is lowered to only one TA per day, a significant excess of unassigned time occurs:

These excesses are so large that they defeat the point of solving the problem using an integer program, since clearly the solution cannot be close to optimal. In addition, due to the lack of assigned time, gaps in TA schedules appear again.

We conclude from this case study that any increase beyond 2 TAs per day will increase the occurrence of gaps, and allow more alternating TA schedules in a day. In addition, the magnitude of this adverse effect increases with more TAs added. We also determined that dropping the

Table 4: Unassigned Time for One TA per Day

TA Name	Unassigned Half-Hours
SL	182
BB	155
FD	91
T	36

number of TAs per day to one is unacceptable, since it fails to assign all available TA time to tasks. Shown below in Figure 3 is the output schedule from limiting the number of TAs in one day to 2.

Figure 3: Output Schedule for 2 TAs per Day

Time	Monday	Tuesday	Wednesday	Thursday	Friday
9:00-9:30	BB	SL	BB	SL	BB
9:30-10:00		SL	FD	T	SL
10:00-10:30	BB		BB	T	BB
10:30-11:00	BB		FD	T	BB
11:00-11:30	SL	SL	FD	T	BB
11:30-12:00		FD		SL	BB
12:00-12:30		FD		SL	SL
12:30-1:00	SL	FD	FD		SL
1:00-1:30	BB	SL	BB		
1:30-2:00	SL	SL	BB	T	
2:00-2:30	SL	FD	BB	SL	
2:30-3:00		FD	FD		SL
3:00-3:30		SL	FD	T	SL
3:30-4:00		FD		SL	BB
4:00-4:30				SL	SL

6 Future Development

The model developed above gives a framework for solving the OTTA problem, but many assumptions were made to reduce the overall complexity of the model. Future versions of the model will replace these assumptions with new model components which take into account their effects. In addition, there are other subproblems which can be included to expand the breadth of the solution.

Solutions produced by this model tend to produce TA schedules with disjoint segments of time, as discovered in the case study. That is, for a given TA on a single day there are several distinct and separated time blocks scheduled. Unfortunately, these schedules are not convenient for TAs or students, even though the schedule gives an optimal assignment to student availability. To correct for this, either an additional set of constraints or a reformulation of the model are necessary.

Fixed-time jobs, such as exam marking and proctoring, are assumed to fit into TA schedules when assigned. However, this can possibly conflict with other assigned tasks or TA availability. The exam proctoring assignment is a problem that can be addressed easily in an update to the model: by adding constraints for each proctoring assignment associating each one with a specific time slot.

The model as presented gives a full schedule for the whole term. This does not account for unexpected events such as sickness. To adjust for this in the model, a post-processing improvement heuristic can be used. This sort of improvement heuristic has already been implemented in a similar integer programming problem for airline schedule recovery [5].

With these additional constraints and modifications, the model may become prohibitively complex. If this is the case, then there is precedent for applying an improvement heuristic: Gunawan et al. [4] modeled a timetabling problem as an integer program that was very complex. They developed an improvement heuristic that solves their problem in a reasonable time. Their heuristic splits the problem into two sub-problems and solves them iteratively, sacrificing optimality in exchange for speed.

7 Conclusion

At the start of this project, we intended to investigate whether the OTTA problem could be solved by a linear program. Though the model presented does not completely solve all the issues present in the problem definition, it does provide a starting point for further development. The OpenSolver-based spreadsheet we developed is able to find a solution that is optimal for the conditions set up in the model.

Improvements to the solution would be in the form of more cohesive schedules, not a more optimal solution. This can be achieved either by altering the model, or performing some improvement heuristics. An alternate improvement would be converting the model into a piece of custom software with a better user interface, with no dependence on Microsoft Excel.

8 Acknowledgments

We wish to thank Dr. Natalia Kouzniak for providing us the details of the Open Tutorial problem at SFU's Surrey campus. She also provided for us anonymized TA and course data, that we were able to use for the case study.
This project is the culmination of an SFU MATH 402 project course, taught by Dr. Abraham Punnen. We would also like to thank Dr. Punnen, for guiding us and providing criticism on our progress developing the network flow model.

References

[1] Addin Osman, Adlan Balola, Anwar Ali Yahya, Yahya Ali Abdelrahman; A Survey of University Courses Timetable Scheduling Problem. *Journal of Computing* 3(2011) 85-90

[2] S. Daskalaki, T. Birbas, E. Housos; An integer programming formulation for a case study in university timetabling. *European Journal of Operations Research* 153(2004) 117-135

[3] Fadime Üney-Yüksektepe, İlayda Karabulut; Mathematical Programming Approach to Course-Teaching Assistant Assignment Problem. *Proceedings of the 41st International Conference on Computers & Industrial Engineering* pg. 878-883

[4] Aldy Gunawan, Kien Ming Ng, Kim Leng Poh; An Improvement Heuristic for the Timetabling Problem. *International Journal of Computational Science* 1(2007) 162-178

[5] Benjamin G. Thengvall, Jonathan F. Bard, Gang Yu; Balancing user preferences for aircraft schedule recovery during irregular operations. *IIE Transactions* 32(2000) 181-193

[6] Raed Al-Husain, Mohamad K. Hasan, Hameed Al-Qaheri A Sequential Three-Stage Integer Goal Programming (IGP) Model for Faculty-Course-Time-Classroom Assignments. *Informatica* 35(2011) 157-164

[7] *OpenSolver - The Open Source Optimization Solver for Excel.* Retrieved from,

`http://opensolver.org/`

.