

# Mathematical Modeling of Common Scheduling

Morgan Redfield, Ryan Timmons

April 9, 2007

## Abstract

We have created a scheduling model that will provide the “most optimal” sequencing of workshops subject to various constraints on time, location, and instructor availability. We modeled this problem using binary integer programming and solved it with MATLAB’s BINTPROG function. The constraints on this function ensured that no room or instructor would be scheduled for two things at the same time, as well as making sure that no workshop could take place at the same time as any other workshop.

This problem can be modeled using the binary-integer problem and is thus considered to be  $\mathcal{NP}$ -hard. In light of this, we had to focus our model on a problem subspace that available technology can solve.

We used our model to schedule six workshops taught by three instructors over a four day period in up to two rooms. This five-day period was taken from the current schedule for Spring quarter, 2007. The schedule created by our method satisfies all of the constraints we have given and is a proof-of-concept for our overall algorithm.

## 1 Full description of Problem

UW Catalyst currently offers a variety of workshops to University staff and students. These workshops are taught by seven instructors who have various other responsibilities. Generally, each instructor teaches between ten and twenty distinct workshops. Each workshop is usually offered twice in a quarter leading to over 100 separate workshop offerings. Before each quarter, the schedule of workshops must be created. The creation of a schedule is generally given a week and a half to be completed.

In the past, the schedule was created at random as each instructor chose a time and place to hold each of the workshops that they were responsible for teaching. This has led to strife as potential workshop students have had to choose between two or more workshops to attend at one particular time.

The current method of scheduling can also lead to the problem of having long periods of time during which there are no workshops offered. There are few classes that need to be offered compared to the possible times that we can offer them, and in past quarters there have been long periods of time (two or more school days) with no workshops followed by many workshops in a very short span of time. This form of schedule also leads to many overlapping classes.

This scheduling problem has elements that make it much easier than a

normal scheduling problem. For example, no class is ever filled to capacity. This means that room capacity is immaterial for our model.

This scheduling problem also has elements that make it much more difficult than a normal scheduling problem. For many applications (especially in academics), the events being scheduled repeat each week. This being the case, only one week must be scheduled. For our problem, we do not have a repeating schedule. This fact will dramatically increase the number of variables that we have to deal with.

## 1.1 Current models of scheduling problems

There are three main methods that are generally used to solve scheduling problems. These are the brute force method, heuristics, and linear programming. The problem is generally agreed to be equivalent to the graph coloring problem.[2]

The majority of scheduling issues currently being tackled in industry and academia seem to be an attempt to schedule a dense selection of classes for a limited number of rooms. In general, many organizations seek to create an optimal schedule by manually brute forcing the problem. Often the organization will simply try many different combinations of classes in different rooms by arranging the classes in a grid like pattern. This is the method that has been used by Catalyst in the past.

Although less common than the brute force method in practice, a far more interesting and efficient approach seems to be to use one of a variety of heuristics developed from a number of natural processes, such as annealing. Simulated Annealing is a form of Monte Carlo method in which a sample solution is used to derive more optimal solutions by comparing the sample and random changes to the sample with some global parameter. Graph theory is also used by putting the problem into the form of a graph coloring problem.

Integer programming has also been used in the past, though we were unable to find an algorithm that had been made by a third party that could be used to solve our specific problem.

## 1.2 Motivation for our Model

Currently there are many different ways of scheduling classes. We have found that many of these ways will not work for our purposes because they work from the assumption that classes happen consecutively. Since we will not have consecutive classes, it will be difficult to use graph theory or one of

the many other heuristics to solve our problem. There will be many times when no classes at all are offered, which conflicts with many current models of scheduling.

We will use integer programming in the solution of this problem because it offers the most flexibility in the organization of our workshops with the fewest changes to the existing models.

Furthermore, according to Cheng, Kang, Leung, and White, instructor availability is far easier to model using linear programming than other available methods[3]. This form of model also allows changes to the constraints to be made much more easily. Since our constraints will change quarterly, it is important that these constraint changes are taken care of quickly so that our schedule can be produced in the 9 days allotted for schedule production.

## 2 Evaluation of Our Model

The analysis of our model is fairly straight forward. We can say for certain that our model works correctly and is accurate if it manages to create a schedule that conforms to the given constraints. Further, our model should consistently give us a schedule that has no overlaps if such a schedule is possible.

This scheduling model does not take into account instructor preference, and such things will be ignored in the evaluation of our model.

## 3 Simplifications

We are making a variety of simplifications to aid in the modeling of this complex problem. These include:

1. We will assume that the instructors do not have a preference in the times that they teach.
2. We will assume that no workshops are offered twice. If we want to remove this simplification, we will represent different instances of the same workshop with unique workshop subscripts.
3. We will assume that all workshops take 2 hours.
4. There is no set workshop order. Any workshop can be offered before any other.

## 4 Overview of the Solving Technique

The Simplex Algorithm[1] provides a mathematical framework onto which a finite number of linear constraints can be applied to a finite number of decision variables in order to maximize a certain linear "value" function. That is,

$$\begin{aligned} \mathcal{P} : \quad & \text{Minimize} && \vec{c} \cdot \vec{a} \\ & \text{Subject To} && A\vec{a} \leq \vec{b} \\ & && A \in \mathbb{R}^{m \times n} \\ & && \vec{c}, \vec{a} \in \mathbb{R}^n \\ & && \vec{b} \in \mathbb{R}^m \end{aligned}$$

### 4.1 A Note on Linear Versus Integer Programming

We start with a state vector  $\vec{a}$  with one entry for every possible workshop/time/instructor/room combination (i.e. an entry for every point in the possibilities subspace). Each entry  $\vec{a}_j$  in  $\vec{a}$  indicates the number of workshops being offered at that entry's corresponding workshop/time/instructor/room combination. The most exigent and obvious constraint is that no instructor can teach more than one workshop at one time; and certainly a fraction of a workshop cannot be offered.

With each entry in  $\vec{a}$  limited to being either a 1 or a 0, our problem reduces to what is informally known as a "zero-one" program. This is a subclass of integer programming problems. These problems are known to be  $\mathcal{NP}$ -hard and, in general, are not applicable to MATLAB's LINPROG function, whose output is not guaranteed to be integers.

MATLAB 7.0's optimization toolbox includes a function, BINTPROG, that is designed to handle binary integer programming. Unfortunately, it is limited by the computer's architecture and cannot handle a large number of variables. With this in mind, we will add as many constraints as possible to make the branch and bound performed by BINTPROG easier.

## 5 Basic Sets and Definitions

1.  $W \subset \mathbb{N}$  is the index set of all **classes** offered. All workshops are one class; we have 56 workshops.
2.  $H \subset \mathbb{N}$  is the index set of all **time periods** in which workshops can be scheduled. Our model states that workshops may start no earlier

than noon and end no later than 8:00pm. Thus, the workshops must start no later than 6:00pm. Workshops are not offered during the first week of classes or during finals week. Workshops are not offered on weekends or holidays.

We will begin by numbering each allowable two-hour block, starting at 0 to indicate noon of the first day during which workshops can be offered. Noon on the second day will be denoted as time block 4 and so on. We can then use integer division and modular arithmetic to establish a bijection between  $\mathbf{h} \in \mathbf{H}$  and the hour  $h_i$  of day  $d_i$  by  $\text{Time}(\mathbf{h}) = (d_i, h_i)$ . i.e.

$$\text{Time} : \mathbf{H} \rightarrow \mathbb{N} \times \{0, 1, 2, 3\} \quad (1a)$$

$$\text{Time } n = (\lfloor n/4 \rfloor, n \bmod 4) \quad (1b)$$

$$\text{Time}^{-1} : \mathbb{N} \times \{0, 1, 2, 3\} \rightarrow \mathbf{H} \quad (2a)$$

$$\text{Time}^{-1}(n, m) = (n)(4) + m \quad (2b)$$

We have 10 weeks of 5 days with 4 time periods each. Thus, we have 200 time periods during which we will consider offering workshops.

3.  $\mathbf{I} \subset \mathbb{N}$  is the index set of all **instructors** who can teach workshops. We currently have 7 instructors.
4.  $\mathbf{R} \subset \mathbb{N}$  is the index set of all the **classrooms** in which a workshop can be taught. There are currently 5 rooms.

We can therefore represent combinations of workshops, times, instructors, and rooms as integer points in a subspace of  $\mathbb{N}^4$ . We will refer to this subspace as the *possibilities subspace*. To be exact, there are  $56 \times 200 \times 7 \times 5 = 392000$  discrete points in this subspace.

We will enumerate the times of the quarter with  $a_{whir}$  being workshop  $w \in \mathbf{W}$  offered at half-hour  $h \in \mathbf{H}$  by instructor  $i \in \mathbf{I}$  in room  $r \in \mathbf{R}$

## 6 Modeling the Objective Mathematically

We originally attempted to minimize the number of overlaps of workshops during a certain hour, but modeling this as an equation cannot be done linearly. If we used some program that can model an objective function non-linearly, we could use this. However, since we can't do that, we will model

the overlaps as a constraint. Precisely, we will state that no workshops may start on the same hour.

By modeling overlaps as a constraint, we find that we have nothing left to minimize and any feasible solution is considered optimal. For this reason, we will simply minimize the summation of all variables. This is something of a “dummy” optimization. Our vector of coefficients would be made up entirely of ones or zeroes.

## 7 Modeling the Constraints Mathematically

Certain entries of  $\vec{a}$  are limited in their possible combinations. These constraints include the obvious fact that an instructor cannot teach more than one workshop at one time. That is to say, our constraints limit what one (or more) entry in  $\vec{a}$  must be when compared to another entry.

We use the following guidelines in determining whether a workshop can or can not take place.

1. An instructor’s university class schedule has top priority. No workshops may be scheduled during an instructor’s class time. If instructor  $\hat{i}$  is unavailable at any time during the two-hour block  $\hat{h}$ , then

$$\sum_{r \in \mathbf{R}, w \in \mathbf{W}} a_{w\hat{h}ir} = 0 \quad (3)$$

2. Each instructor can teach only 6 hours per week. Thus,

$$\sum_{\substack{h=20(n+1) \\ h=20n \\ r \in \mathbf{R}, w \in \mathbf{W}}} a_{whir} \leq 3 \quad n = 0, 1, \dots, 9; \quad i \in \mathbf{I} \quad (4)$$

Here  $n$  represents a given week. Since there are 40 hours during which a workshop can be offered every week, we don’t want any instructor’s teaching hours in any 20-hour period to be more than 6.

3. The rooms are not available at all times. The classrooms in which workshops are taught are also reserved for other classes, which must be accounted for in our model. If room  $\hat{r}$  is unavailable at time block  $\hat{h}$  then

$$\sum_{w \in \mathbf{W}, i \in \mathbf{I}} a_{w\hat{h}ir} = 0 \quad (5)$$

4. Not all instructors are able to teach all workshops. If an instructor  $\hat{i}$  is unable to teach workshop  $\hat{w}$ , then

$$\sum_{h \in \mathbf{H}, r \in \mathbf{R}} a_{\hat{w}\hat{h}\hat{i}r} = 0 \quad (6)$$

5. Each room can contain only one class during any given time period. This constraint will be modeled with

$$\sum_{w \in \mathbf{W}, i \in \mathbf{I}} a_{w\hat{h}ir} \leq 1 \quad (7)$$

6. No two workshops can take place at the same time.

$$\sum_{w \in \mathbf{W}, i \in \mathbf{I}, r \in \mathbf{R}} a_{w\hat{h}ir} \leq 1 \quad (8)$$

7. An instructor can teach only one workshop at a time. This is given by the equation

$$\sum_{w \in \mathbf{W}, r \in \mathbf{R}} a_{w\hat{h}ir} \leq 1 \quad (9)$$

8. Only three workshops can be taught in one day. This is given by the equation

$$\sum_{w \in \mathbf{W}, h = \hat{h}, i \in \mathbf{I}, r \in \mathbf{R}} a_{w\hat{h}ir} \leq 3 \quad (10)$$

9. All workshops must be offered once. For a given workshop:

$$\sum_{h \in \mathbf{H}, i \in \mathbf{I}, r \in \mathbf{R}} a_{\hat{w}hir} = 1 \quad (11)$$

## 8 Generating and Inputting the Constraints

We have equality constraints and inequality constraints. We broke these two types of constraints into two matrices and two vectors. We have  $A_{eq}$  the matrix of equality constraints,  $b_{eq}$  the vector corresponding to  $A_{eq}$ ,  $A_{leq}$  the matrix of inequality constraints, and  $b_{leq}$  the vector corresponding to  $A_{leq}$ .

We created a PHP script<sup>1</sup> to create the MATLAB code for the rows of the constraints matrices  $A$ . Once created, we can concatenate these rows to create the matrix. We create our corresponding  $\vec{b}$  at the same time that we concatenate these rows. We also used the PHP script to create our objective function.

Once we have all of our constraints and objective function made, we can plug them into MATLAB and compute an optimal schedule.

Entering the scheduling constraints for a given quarter involves modification to the PHP script that translates the scheduling constraints into the mathematical constraints required by MATLAB.

## 9 Results

Originally we were planning to use MATLAB to compute the most optimal results, but this proved to be too costly for us. MATLAB only has the capability to solve binary programming problems in version 7.0, which was only available to us at for short periods of time in the Math Sciences Center. We then decided to use GNU Linear Programming Kit, GLPK, which is a free and open source program for solving linear programming problems. This method seemed like it could have been effective, but since GLPK is still relatively new in its lifecycle, we did not implement our model using its syntax. In the end, we decided to deal with MATLAB, in spite of its limitations.

If we had more time and funding, we could find a program that will solve an integer programming problem that does not have a linear optimization function. GLPK, for example, can do non-linear optimizations.

The next stumbling block we ran into was the large number of variables required for this problem. Scheduling is notorious for the enormous possibilities subspace that it has regardless of the model used to solve it. It would take far too much time and computational power to compute an optimal schedule for an entire quarter using BINTPROG. For this reason we decided to schedule only a small sample of our workshops and constraints as testament to the effectiveness of our model.<sup>2</sup>

This simplification dramatically reduces the number of variables (and thus the possibilities subspace) that we have to deal with. However, making a smaller schedule still requires the same model. If we can make an optimal

---

<sup>1</sup>See Appendix 2

<sup>2</sup>See Appendix 1

schedule that only has a few workshops, the extension of the model to the full schedule is feasible given more computing resources.

One benefit of the method we created is that it is easily customizable. Customizability was extremely important for our model, as we will have to create a new schedule each quarter. Each of our instructors will have wildly differing schedules from quarter to quarter, not to mention that the rooms will have different availabilities as well. Appendix 2 (specifically, the readme file) contains information about how to do this.

We found that our method of solving this problem provided accurate results, but it was constrained by the enormous number of variables for an entire quarter. There are a variety of ways that we could simplify our schedule to reduce the number of variables, but we would have to seek approval from Catalyst before going ahead with these changes.

The first change is to restrict class to Monday through Thursday. We currently offer classes on Fridays, but the attendance of these tends to be low. We also consider the schedule to be a repeated event. Since most of the instructors' time constraints are due to classes, they are repeated each week. This means that the schedule for the first iteration of classes may very well work for the second iteration of classes. If we modeled only the first half of the quarter and offered each class once for this time period, we could simply repeat this schedule for the last half of the quarter.

Despite our massive simplifications to the scheduling algorithm, our model is still useful for many applications. Specifically, it can be used for scheduling classes that repeat weekly, as in high school and elementary schools. One elementary school has already contacted us about using our model to schedule gym, music, and library sessions for their students.

## 9.1 Breakdown of work by group member

Each member of our group did approximately the same amount of work, with some focus on different areas. Mr. Timmons was responsible for the majority of the programming, while Mr. Redfield was responsible for much of the writing. Both contributed about equally to the creation of the model and the constraints.

## 10 Appendices

Our appendices contain the schedule we created with our model, the code used to create said schedule, and instructions for modifying that code to use it for longer schedules.

**Appendix 1** This is the schedule that we created with our model for six workshops taught by three instructors over a five day period in up to five rooms.

**Appendix 2** This is the “readme” document for our project code. It includes a detailed description of the other files in the folder space, information on how to run the programs, and information on how to customize the schedule for a new quarter’s constraints.

**Appendix 3** This contains a listing of all the code used by this model. In particular, the listings that are “customizable” from quarter-to-quarter have been customized to the schedule mentioned in the earlier appendices.

## References

- [1] Vašek Chvátal, *Linear Programming*, 1983 W.H. Freeman and Company.
- [2] Anthony Wren, “Scheduling, Timetabling and Rostering - A Special Relationship?” *Practice and Theory of Automated Timetabling*, 1995 Springer-Verlag.
- [3] Czarina Cheng, Le Kang, Norrus Leung, George M. White, “Investigations of a Constraint Logic Programming Approach to University Timetabling,” *Practice and Theory of Automated Timetabling*, 1995 Springer-Verlag.