An efficient and flexible preference based binary staff scheduler for small businesses

Jaedin Davasligil

November 15, 2020

Small businesses come with an assortment of difficult problems. One of the major challenges is scheduling employees with a variety of policies, regulations and a little chaos. In this paper we propose a model which effectively schedules employees for one workweek at 1-hour resolution. We take into account the practical requirements of a manager: the model needs to be flexible to sudden changes and business closures; the model must give both the manager and the employees the ability to adjust the schedule to their needs; and the model must yield a serviceable schedule even when conflicts arise. Our model addresses each of these concerns efficiently. Our target demographic is daytime businesses since the model does not address night-shifts, however, it can be adapted to work with 24-hour businesses with careful consideration. Schedule conflicts are avoided at all costs but are easily traceable when they occur. Flexibility and manual control is obtained by utilizing large parameter matrices which indicate employee availability, preferred hours, and lower / upper bounds on total employees scheduled. The problem of scheduling to volume is delegated to the manager and is left as a separate problem entirely. The manager is also given the power to bound weekly hours for each employee and total daily hours for all employees. The purpose of this model is to provide a tool for providing a baseline schedule while detecting potential conflicts.

1 Contact List

1. Zoe Coffee & Kitchen (Restaurant)
   (a) Small, family owned, and local. The owners are active with the community.
   (b) Restaurant management entails complex optimization problems which include scheduling and profit maximization with consideration for changes in demand.
   (c) I am interested in restaurant management.
2. Paradise Creek Bicycles (Bike Shop)
   (a) Owner is usually on site and very friendly / approachable. Local.
   (b) Operating a bicycle store includes managing rentals, repairs, scheduling staff, and stocking the store for various seasons. There are many important decisions for an owner to make and it may lead to interesting sub-problems.
   (c) I am interested in the bicycle industry.
3. Palouse-Clearwater Environmental Institute (Non-Profit)
   (a) Local non-profit and easy to contact.
   (b) There are many possible optimization problems due to the large scope of the organization. How do you decide whether or not to greenlight a project? Budget management and optimization are necessary for the survival of a nonprofit.
   (c) I am curious about this organization and their operations.

2 Contact Plan

Two businesses will be contacted at a time. If the contacts fail then I will move down the list which will grow as needed. Contact will be initiated as follows:

1. Prepare and send emails to the respective public email addresses for each business. For example,

   To whom it may concern,

   My name is Jaedin Davasligil and I am a senior studying Mathematics at Washington State University. I am a student in Dr. Asaki's Mathematical Optimization course and I am researching how Mathematics can be used to assist decision-making processes in business situations as a part of my class project. One of my interests is in scheduling events and personnel.

   Would it be possible to schedule a short meeting (20-30 minutes) to discuss scheduling decisions and business related challenges with an appropriate representative of the company? If so, I am available Tuesdays and Thursdays between 12:00 and 3:00PM, Fridays after 3:00PM, and on Saturday or Sunday at any reasonable hour.

   If you have any questions you may contact my professor at tasaki@wsu.edu.

   Sincerely,

   Jaedin Davasligil

   I may not have access to the contact information of specific persons. If a specific contact exists then the email will be tailored for that contact.

2. If the person of interest can only be contacted directly then I may have to initiate contact in person. I will introduce myself, explain the situation as in the first paragraph of the email example, and offer to schedule a meeting. Our contact information will be printed out including both the professor's email and mine. This information will be available for them at the conclusion of the initial meeting.

   This approach will only be taken if the first fails and if it is appropriate.

2.1 Prepared Questions

- Zoe Coffee & Kitchen (Restaurant)
  1. What challenges do you face scheduling staff?
  2. How do you decide how many people you need to hire?
  3. If some people work part time and others work full time, how do you distribute their hours to meet everyone's needs, including the customer's?
  4. How do these needs change from season to season? What are the major factors? Is being situated right outside campus a large part?
- Paradise Creek Bicycles (Bike Shop)
  1. As a bike store owner, what kinds of decisions do you need to make?
  2. For example, how do you decide what days to open and for how long?
  3. What are some factors you need to take into consideration?
  4. Would you say the goal is to minimize operating costs while selling the most products and services?
  5. What are some other important goals?
- Palouse-Clearwater Environmental Institute (Non-Profit)
  1. This institute has an impressive amount of ongoing projects and services. With a limited budget, how do you decide what to prioritize?
  2. What kinds of decisions and considerations are there for approving or cutting a project / service?
  3. What factors are out of the Non-Profit's control?
  4. How do you manage those factors? How do you adapt?

3 Meeting Synopsis

I met with Mike, the owner of Zoe Coffee & Kitchen, on tuesday (02/17/2020) from 13:00 to 14:00. Our discussion primarily revolved around the complexities of scheduling restaurant staff. The restaurant can be divided into two primary operations: the front and the back. The front is involved with customer service: waiting tables; operating the register; making drinks; et cetera. The back is concerned with running the kitchen. There is usually an experienced head chef and a few less experienced cooks. The less experienced staff may be assigned to wash dishes or prepare ingredients for the next day. These jobs only need to be assigned on certain days. Naturally, the more people that come, the more frequent dishes need to be cleaned.

The primary problem with scheduling staff is preparing to meet volume. Volume is defined as the number of customers who need to be served at a particular time. Since the goal of every business is to maximize profits, we need to minimize operating costs by scheduling the minimum amount of staff necessary to meet the demand at a given time. This demand can be predicted in advance by past data and common sense. The demand is always changing with multiple variables, but for the sake of our model we can assume this demand can be predicated in advance.

The head chef must begin prepping the kitchen an hour before opening. A barista must begin preparing the front (espresso machines, etc.) thirty minutes before opening. There must be a third waiter scheduled upon opening. This forms the bare minimum skeleton crew for the restaurant. There must always be at least two operating the front and one in the back. Only one person (typically the chef) needs to stay thirty minutes to close the restaurant.

There are several complications. Federal and state regulations impose constraints on how long employees can work, how much time they need for lunch, and minimum requirements for breaks. Schedules must be prepared for employees at least two weeks in advance. For flexibility, employees may be on call. However, with new regulations, it may be required to pay them while waiting on call. Further, if the employer wishes to reschedule someone against their will after these two weeks, then they are required to be paid for their time. Employees can volunteer their flexibility freely, however, it can not be expected.

There are two models to consider: an adaptive model with employees on call to meet unexpected demand and a rigid model with a set schedule.

In summary, the objective is to minimize the number of people to schedule at each time partition. The model can be broken into two parts: front and back. If a day is a dishwashing day, then one person must be scheduled on top of the skeleton crew. If the demand is not zero in the evening, then an additional person must be scheduled to prepare food for the next day. There are two kinds of positions for both front and back. In general, there are senior and junior employees. We may assume senior employees are paid more. We must consider opening and closing requirements. We must also consider legal regulations. We can either consider a realistic adaptive model or a simple rigid model.

4 Problem Statement

Pugs n' Mugs Coffee and Kitchen is a new pet friendly restaurant. You have been hired to formulate a linear program to help their manager schedule staff for one workweek. A workweek is defined as a periodic cycle of 168 hours.

The manager has decided that it is more important to optimize the schedule for availability and shift preference than to minimize operating costs. Preference should be weighted towards those who select fewer preferences.

As a small restaurant, there are only three distinctive positions which must be scheduled. Workers at the "front" are servers but they also handle the register and prepare drinks. This is why there must always be at least two front workers during working hours. In the "back" is the head chef and some less experienced cooks. There must always be at least one head chef. The model must be flexible enough to account for these details.

The manager will decide both a minimum and maximum on the number of workers for any shift in the form of workweek tables (see the example below). They will always require at least two front staff and one chef. An additional cook is needed on Mon., Wed., Fri., and Sat. to wash dishes. Another cook is needed every evening to prepare ingredients for the following day. This will be reflected in the manager's parameters given to you.

The program should always return a schedule despite worker availability. Such conflicts should be made apparent to the manager so they can either make arrangements or hire more staff for the unavailable positions.

The restaurant has hours from 06:00 to 21:00 but opens at 07:00 on weekends. The schedule should enforce a minimum and maximum on the total hours per employee as well as a max on the daily hours.

	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	24:00
Sun	0	0	0	0	0	1	2	2	3	3	3	3	3	2	2	2	2	3	3	3	1	0	0	0
Mon	0	0	0	0	1	2	2	2	2	2	3	3	3	2	2	2	2	2	2	2	1	0	0	0
Tue	0	0	0	0	1	2	2	2	2	2	3	3	3	2	2	2	2	2	2	2	1	0	0	0
Wed	0	0	0	0	1	2	2	2	2	2	3	3	3	2	2	2	2	2	2	2	1	0	0	0
Thu	0	0	0	0	1	2	2	2	2	2	3	3	3	2	2	2	2	2	2	2	1	0	0	0
Fri	0	0	0	0	1	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	1	0	0	0
Sat	0	0	0	0	0	1	2	2	3	3	3	3	3	3	3	3	3	3	3	3	1	0	0	0

Table 1: Minimum number of front workers.

5 Optimization Model

5.1 Decision Variables

Let x_{ij} be one if employee i is scheduled to work the jth hour and zero otherwise where $i = 1,2,\ldots,m$ and $j = 1,2,\ldots,n$ for m employees and n workweek-hours. There should be $n = 7*24 = 168$ workweek-hours.

5.2 Parameters

Let a_{ij} be one if employee i is available to work the jth hour and zero otherwise.

Let $\alpha = \max_i \sum_j a_{ij}$ denote the largest number of hours available to any employee.

Let $p_{ij} = a_{ij} p_{ij}$ be one if employee i prefers to work hour j and zero otherwise.

Let $w_{ij} = \frac{\alpha}{\sum_j p_{ij}} p_{ij}$ be the weighted preference. (Zero if $\sum_j p_{ij} = 0$.)

Let $M = n\alpha$ be a reasonable upper bound for total weight.

Let t_i and T_i be the minimum and maximum hours per week for each employee.

Let r_j and R_j be the minimum and maximum number of employees required at time j.

Let D be the maximum hours per day any employee is allowed to work.

5.3 Constraints

1. $t_i \le \sum_j x_{ij} \le T_i$ for $i = 1,2,\ldots m$.
2. $r_j \le \sum_i x_{ij} \le R_j$ for $j = 1,2,\ldots n$.
3. $\sum_{j \in \mathcal{I}_d} x_{ij} \le D$ where $d = 1,2,\ldots 7$ for $i = 1,2,\ldots m$.

The first constraint ensures that each employee gets their required hours.

The second constraint guarantees that there are enough employees scheduled for each hour.

The final constraint makes sure that the daily hours of each employee is bounded so that an employee can not be scheduled to work all day long. We define the (discrete) day intervals

$$\mathcal{I}_d = [1 + 24(d-1),\ 24d],\ d = 1,2,\ldots,7.$$

Note: The business is closed if and only if $r_j = R_j = 0$. We can use this fact to ensure no employee is scheduled during off hours. We can also ensure each employee has 1 day of rest by setting an entire day of their availability to zero. This tactic also enables emergency schedule changes.

5.4 Objective

$$\max_x \; z = \sum_{i,j} \big(w_{ij} - M(1 - a_{ij})\big) x_{ij}.$$

We include an overwhelming penalty for assigning an unavailable worker. The model will avoid this if at all possible without being infeasible.

5.5 Model

$$\begin{aligned}
\max_x \quad & z = \sum_{i,j} \big(w_{ij} - M(1 - a_{ij})\big) x_{ij} \\
\text{s.t.} \quad & t_i \le \sum_j x_{ij} \le T_i \\
& r_j \le \sum_i x_{ij} \le R_j \\
& \sum_{j \in \mathcal{I}_d} x_{ij} \le D \\
& x_{ij} \in \{0, 1\}
\end{aligned}$$

We can check for schedule conflicts by testing if $x^*_{ij} = (a_{ij}) x^*_{ij}$. A zero indicates a schedule conflict. An infeasible schedule indicates that there are not enough workers to meet volume.

6 Solution

Suppose Pugs n' Mugs has a company policy requiring that front staff work a minimum of 15 and a maximum of 35 hours per week and are limited to 8 hours per day. The coffeehouse currently has 12 front-staff available. A work-week output from the scheduler function [Appendix I] will try to form a schedule according to the employee's preferred hours given an arbitrary but reasonable pair of availability and preference matrices. Since the inputs and outputs are too large to fit on this page we refer to the code in the appendices.

Schedule conflicts seem extremely rare. With a realistic example it can certainly happen, however, it is unlikely with a sufficient number of workers. I have observed that the upper bound on workweek hours, T, can lead to an infeasible problem. The smaller this bound, the more staff you need to hire. Notice that $\sum T \leq (m)\max(T)$ so to be feasible we require that

$$\sum r \leq m\mu \quad \text{where} \quad \mu = \min\big(\max(T), 6D\big),$$

which states that the total hours employees can work must be greater than the total hours required to work. This gives a lower bound on the number of employees:

$$\left\lceil \frac{1}{\mu} \sum r \right\rceil + 2 \leq m.$$

We add two for flexibility. For our problem we obtain $10 \leq m$. With twelve employees we are in very good shape! This lower bound assumes total availability and so it should be used only as a first estimate to guarantee feasibility.

To test the robustness of the scheduler we performed $N = 1000$ experiments starting with full availability and randomly removing available hours with $p = .5$ so that on average employees are only available half of the time with uniform randomness. The schedule had conflicts 65.6% of the time with a median of 1 conflict and standard deviation of 1.2158. This is a fairly extreme scenario yet there is only one conflict on average with a fair chance of no conflicts.

Reducing the probability to $p = 0.25$ yields conflicts 0.2% of the time with a median of 0 conflicts and standard deviation of 0.0447. This is a more reasonable scenario with only 25% availability loss and there are practically no schedule conflicts.

The average run time on a 3.4GHz i5 processor machine is about 0.04 seconds in Octave.

We can also test how well preferences are being met by taking the ratio of total preference conflicts divided by total number of preferences. Experiments indicate that, with the parameters in the test code in appendix II, if the preference matrix is sparse then the proportion of marked preferences, e.g., 10%, is fairly to the proportion of conflicts.

We find that the model works exceedingly well under typical circumstances where we assume employees have been hired with a sufficient variety of available times. The model will work as intended and forms an optimal schedule even if schedule conflicts occur. It will also report the number of conflicts with the exact time and employee being fully traceable. The model does not micro-manage and instead allows the manager to influence the schedule manually with the availability matrix. There are simple guidelines for how to guarantee things like mandatory rest periods. This flexibility is highly desirable for small businesses which frequently require schedule changes.

Below is an example schedule for the first employee where the availability and preference matrices are generated randomly. The code can be found in appendices I–III. Schedules appear slightly scattered with random parameters, however, the outcome is quite reasonable otherwise. Notice how most preferred hours are scheduled.

	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00
Sun	0	0	0	0	0	0	0	1	1	1	1	1
Mon	0	0	0	0	1	1	1	1	1	1	0	0
Tue	0	0	0	0	0	0	0	0	0	0	0	0
Wed	0	0	0	0	0	0	0	0	0	0	0	0
Thu	0	0	0	0	0	0	0	0	0	0	0	0
Fri	0	0	0	0	0	0	1	0	0	1	1	0
Sat	0	0	0	0	0	0	1	0	1	1	0	0

	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	24:00
Sun	0	0	0	0	0	1	0	1	0	0	0	0
Mon	0	0	0	0	1	0	0	0	0	0	0	0
Tue	0	0	0	0	0	1	0	0	0	0	0	0
Wed	0	0	0	0	0	1	0	0	0	0	0	0
Thu	1	1	0	1	1	1	0	1	0	0	0	0
Fri	1	1	0	0	1	1	1	0	0	0	0	0
Sat	1	0	0	0	0	0	0	1	0	0	0	0

Table 2: Employee 1 Hours Scheduled (Random)

	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00
Sun	0	0	0	0	0	0	0	0	0	0	1	0
Mon	0	0	0	0	0	0	0	0	0	0	0	0
Tue	0	0	0	0	0	0	0	0	0	0	0	0
Wed	0	0	0	0	0	0	0	0	0	0	0	0
Thu	0	0	0	0	0	0	0	0	0	0	0	0
Fri	0	0	0	0	0	0	1	0	0	0	0	0
Sat	0	0	0	0	0	0	0	0	0	0	0	0

	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	24:00
Sun	0	0	0	0	0	0	0	0	0	0	0	0
Mon	0	0	0	0	1	0	0	0	0	0	0	0
Tue	0	0	0	0	0	1	0	0	0	0	0	0
Wed	0	0	0	0	0	1	0	0	0	0	0	0
Thu	0	1	0	0	0	1	1	0	0	0	0	0
Fri	0	0	0	0	0	1	1	0	0	0	0	0
Sat	0	0	0	0	0	0	0	1	0	0	0	0

Table 3: Employee 1 Hours Preferred

For an example with no randomness and complete availability, the first employee's schedule looks like the following:

	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00
Sun	0	0	0	0	0	1	1	1	1	1	1	1
Mon	0	0	0	0	1	1	1	1	1	1	1	0
Tue	0	0	0	0	0	0	0	0	0	0	0	0
Wed	0	0	0	0	0	0	0	0	0	0	0	0
Thu	0	0	0	0	0	0	0	0	0	0	1	1
Fri	0	0	0	0	0	0	0	0	0	1	1	1
Sat	0	0	0	0	0	1	1	1	1	0	0	0

	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	24:00
Sun	1	0	0	0	0	0	0	0	0	0	0	0
Mon	0	0	0	0	0	0	0	0	0	0	0	0
Tue	0	0	0	0	0	0	0	0	0	0	0	0
Wed	0	0	0	0	0	0	0	0	0	0	0	0
Thu	1	0	0	0	1	1	1	1	1	0	0	0
Fri	1	1	1	1	1	0	0	0	0	0	0	0
Sat	0	0	0	0	0	0	0	0	0	0	0	0

Table 4: Employee 1 Hours Scheduled

This indicates that the GLPK's simplex algorithm naturally forms nice shifts in contiguous blocks. This is a desirable feature which is not apparent when testing examples with randomness. It appears that as long as the availability and preference matrices are formed nicely then the schedule will be nice as well. Nice, in this case, meaning contiguous hours with few gaps.

In any case, more testing with real world scenarios is necessary before giving this model the stamp of approval. The Pugs n' Mugs canonical daytime restaurant example is formulated after Zoe Coffee and Kitchen and, although the model works with random data, we may have interesting results with real data. Schedule conflicts may be more common in reality since unavailability blocks can often overlap. Hopefully the model provides enough information for managers to detect and deal with said conflicts.

There are several shortcomings of this model. First, the 1-hour resolution imposes an obvious limitation. In reality, employees may need to be scheduled to the minute for certain operations. It would not be too difficult to reformulate the model at another resolution, however, it does become unwieldy. Second, the optional daily hour limitation does not work for overnight shifts. We can let $D = 24$ to effectively remove the constraint and then leave such details up to the manager. They need only enforce the rule upon the availability matrix before running the scheduler. Third, the model makes no effort to enforce contiguous blocks of time. We can not speak for other algorithms but the GLPK algorithm does a good job on its own. The preference matrix should be formulated carefully with contiguous blocks to prevent chaos.

Overall, the goal of this model is to provide a useful, flexible tool for developing a baseline schedule while detecting conflicts. It may not be suitable to the needs of every business.

7 Reflection

The most difficult part of this project was obtaining a meeting and communicating with my contact. Small business owners have notoriously busy schedules and I am extremely grateful that Mike, the owner of Zoe Coffee and Kitchen, found the time to meet with me. I could have done a better job leading the interview and getting more useful information. The main take away from this meeting was how important flexibility was to the scheduling process. The manager needed to have the power to change the schedule on short notice. It can be very difficult interviewing someone for mathematical information but I have learned from this experience and will be much better prepared in the future. This process was a little out of my comfort zone but I believe it was a valuable experience. In the future I will have a better idea of what questions to ask. I would have liked to discuss the model with my contact during development but unfortunately circumstances have limited us greatly.

The modelling process went surprisingly well. I tried to keep it clean and simple without micro managing in order to develop a useful tool. It was still difficult to program despite the deceptive simplicity. It is not too difficult to understand the model and its behavior due to the clarity of the parameters and model design. The results can be read in a simple table format and patterns are readily apparent. If the lower bound for employees is met then the model is practically guaranteed to be feasible regardless of conflicts. I am very happy with the behavior of the model so far. Most of my ideas for the model came from my experience working for a tutoring center where my boss was in charge of scheduling. We were required to tell her our available hours and preferred hours. This seems like a very natural method of building an agreeable schedule for both the manager and employees.

The only unexpected results I was having had to do with infeasibility which I solved by formulating the lower bound. The model will yield a feasible solution as long as the number of employees meets that requirement. Also, I ran quite a number of experiments with both random and hand crafted data. The model works predictably with hand crafted data and is quite uninteresting. It is easy to construct a conflict or prevent a conflict. I was primarily interested in randomized data and how the proportion of unavailability is related to the probability of a conflict. The results are promising for uniform randomness but there is a rich vein of statistics to be investigated with realistically distributed (clustered) data. I do not have the time right now but it will be fun to investigate in the future.

My mathematical writing abilities have improved greatly in the two years I have been at Washington State University. I feel quite comfortable writing mathematically and I hope it shows. As always, I am glad to accept feedback and improve.

Overall I felt comfortable with the majority of this project and the given timeline. I have five upper division classes this semester and was still able to fit this project into my schedule despite the challenges involved.

Appendix

I Scheduler Program

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Math 464 Linear Programming Project: Pugs n' Mugs Coffee and Kitchen.     %
%                                                                           %
% This code is for educational purposes only and is not for any other use,  %
% quotation, or distribution without written consent of the author.         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Function 'scheduler' creates an availability and preference based staff
% schedule in the form of an employee by workweek-hour matrix. This matrix
% can be transformed into a daily schedule format using the given interval I.
%
% Author: J.U. Davasligil
% Version: 2020-04-21 (ISO 8601)
%
% INPUTS:
%
%    A    An m employee by n workweek-hour binary availability matrix.
%
%    P    An m employee by n workweek-hour binary preference matrix.
%
%    t    A column vector of the min hours/week for each employee.
%
%    T    A column vector of the max hours/week for each employee.
%
%    r    A column vector of the min employees for each workweek-hour.
%
%    R    A column vector of the max employees for each workweek-hour.
%
%    D    A constant enforcing a maximum on the hours/day for each employee.
%
% OUTPUT:
%
%    S    An m employee by n workweek-hour binary staff schedule.
%
% NOTES: A workweek is a fixed, regularly-recurring period of 168 hours.
%        The model does not account for rest periods since we assume the
%        business is not open 24 hours. This requirement can, however,
%        be met manually with 8 hour rest periods in the A matrix.
%
%        This model assumes a workweek starts Sunday at 01:00.
%
%        We know a business is closed if r_j = R_j = 0.

function S = scheduler(A,P,t,T,r,R,D)

%% Default values

m = rows(A);
n = columns(A);
S = [];

%% Input Validation

if n != 168
        disp('Error: Not a valid workweek. Must be 7 * 24 = 168 hours long.')
        return
end

if m < 1
        disp('Error: There must be at least one employee.')
        return
end
```

```
if floor(D) != D || D < 1
        disp('Error: D must be a positive integer')
        return
end

if any(size(A) != size(P))
        disp('Error: Availability and preference matrices are not the same size.')
        return
end

if   m != rows(t)...
  || m != rows(T)...
  || n != rows(r)...
  || n != rows(R)
        disp('Error: Dimension mismatch. Check vectors t, T, r, and R.')
        return
end

if any(P != (A .* P))
        disp('Error: Preference marked when unavailable.')
        return
end

mu = min(max(T), 6*D);
lb_m = ceil(sum(r) / mu);
lb_warning = sprintf('Warning: %d employees minimum.', lb_m);

if m < lb_m
    disp(lb_warning)
end

%% Calculated parameters and constants

% Largest number of hours available to any employee (for scale).
alpha = max(sum(A'));

% Big M method: Upper bound on total weight possible.
M = n*alpha;

% How each employee's preferences get weighted. Division by zero = 0.
weights = alpha ./ sum(P');
weights = min(weights, M + 1).*(weights ~= Inf);

% Building the weight matrix:
W = P;
for i = 1:m
        W(i,:) = W(i,:) .* weights(i);
end

%% Optimization Model: glpk (c, B, b, lb, ub, ctype, vartype, sense, param)

c = vec(W) - M * (1 - vec(A));

b = [(-1)*t ; T ; (-1)*r ; R ; D*ones(7*m,1)];

B = [ (-1)*repmat(eye(m),1,n) ;
           repmat(eye(m),1,n) ;
      (-1)*kron(eye(n),ones(1,m)) ;
           kron(eye(n),ones(1,m)) ;
           kron(eye(7),repmat(eye(m),1,24)) ];

lb = zeros(m*n,1);

ub = ones(m*n,1);

ctype = repmat('U',1,rows(b));
```

```
vtype = repmat('I',1,m*n);

sense = -1;

[dv,ov] = glpk(c, B, b, lb, ub, ctype, vtype, sense);

S = reshape(dv, m, n);
```

II Pugs n' Mugs Example

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Math 464 Linear Programming Project: Pugs n' Mugs Coffee and Kitchen.      %
% By Jaedin Davasligil (2020/04/21).                                         %
%                                                                            %
% Description: An example of a staff schedule for a small coffee shop.       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% First visible column represents 01:00, last is 00:00 [24:00].
% Weekday pre-open hour is 05:00; weekend is 06:00.
% All days have closing (afterhour) at 21:00.
%
% Rules of Thumb: 1 FTE can work 5 shifts, hire 2 more than necessary.
% Currently we are scheduling 12.
%
% The following table presentation is called workweek (ww) format.

% Number of employees.
m = 12;

% Expected unavailability proportion:
 p_unavail = 0.25;

% Expected preference proportion:
 p_pref = 0.10;

% Every employee is fully available by default:
A = ...
[ 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 ... % Sun
  0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 ... % Mon
  0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 ... % Tue
  0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 ... % Wed
  0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 ... % Thu
  0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 ... % Fri
  0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 ];  % Sat

A = repmat(A, [m,1]);

% Randomized Unavailability Option
A = A .* discrete_rnd([0,1],[p_unavail, 1 - p_unavail], size(A));

% For custom preference:
P = ...
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ];

% Same for each employee. For full customization,
% we need to manually fill out P for each worker.
P = repmat(P, [m,1]);

% Randomized Preference Option
P = A .* discrete_rnd([0,1],[1 - p_pref, p_pref],size(P));

 t = 15*ones(m,1);

T = 35*ones(m,1);

% Minimum employees required for each workweek-hour.
 r =...
[ 0 0 0 0 0 1 2 2 3 3 3 3 3 2 2 2 2 3 3 3 1 0 0 0 ... % Sun
  0 0 0 0 1 2 2 2 2 2 3 3 3 2 2 2 2 2 2 2 1 0 0 0 ... % Mon
```

```
   0 0 0 0 1 2 2 2 2 2 2 3 3 3 2 2 2 2 2 2 2 2 1 0 0 0 ... % Tue
   0 0 0 0 1 2 2 2 2 2 2 3 3 3 2 2 2 2 2 2 2 2 1 0 0 0 ... % Wed
   0 0 0 0 1 2 2 2 2 2 2 3 3 3 2 2 2 2 2 2 2 2 1 0 0 0 ... % Thu
   0 0 0 0 1 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 1 0 0 0 ... % Fri
   0 0 0 0 0 1 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 0 0 0 ]'; % Sat

% Maximum employees required for each workweek-hour.
R =...
[ 0 0 0 0 0 1 2 2 3 4 4 4 4 3 3 3 4 4 4 3 1 0 0 0 ... % Sun
  0 0 0 0 1 2 2 2 2 2 2 3 3 3 2 2 2 2 2 2 2 2 1 0 0 0 ... % Mon
  0 0 0 0 1 2 2 2 2 2 2 3 3 3 2 2 2 2 2 2 2 2 1 0 0 0 ... % Tue
  0 0 0 0 1 2 2 2 2 2 2 3 3 3 2 2 2 2 2 2 2 2 1 0 0 0 ... % Wed
  0 0 0 0 1 2 2 2 2 2 2 3 3 3 2 2 2 2 2 2 2 2 1 0 0 0 ... % Thu
  0 0 0 0 1 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 1 0 0 0 ... % Fri
  0 0 0 0 0 1 3 3 3 4 4 4 4 3 3 3 4 4 4 3 1 0 0 0 ]'; % Sat

% Maximum hours an employee may work per day.
D = 8;

% Lower bound on number of employees.
mu = min(max(T), 6*D);
m_lb = ceil(sum(r) / mu) + 2;

tic();
S = scheduler(A,P,t,T,r,R,D);
runtime = toc()

At = ww_format(A);
Pt = ww_format(P);
St = ww_format(S)

% To check for schedule conflicts
schedule_conflict_total = sum(sum(S.*A ~= S))

% To check for preference conflicts
sum_pref = sum(sum(P));
preferences_met_ratio = (sum_pref - sum(sum(P .* S ~= P))) / sum_pref
```

III Workweek Formatting Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Math 464 Linear Programming Project: Pugs n' Mugs Coffee and Kitchen.   %
%                                                                          %
% This code is for educational purposes only and is not for any other use, %
% quotation, or distribution without written consent of the author.        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Function 'ww_format' transforms a given m employee by n workweek-hour matrix
% into a standard workweek (ww) table form with 24 columns ranging from hour
% 01:00 to 24:00. Every 7 rows represents an employee's daily schedule.
%
% Author: J.U. Davasligil
% Version: 2020-04-13 (ISO 8601)
%
% INPUTS:
%
%    A    An m employee by n workweek-hour matrix.
%
% OUTPUT:
%
%    At   A workweek formatted table.

function At = ww_format(A)

m = rows(A);
n = columns(A);

if n != 168
        return
end

At = zeros(7*m,24);

k = 0;
for i = 1:(7*m)
        j = mod(i-1,7) + 1;
        if j == 1
                k = k + 1;
        end
        At(i,:) = A(k,(1+(j-1)*24):(j*24));
end
```