Washington University in St. Louis

Scheduling Algorithm with Optimization of Employee Satisfaction

by

Philip I. Thomas

Senior Design Project

http : *//students.cec.wustl.edu/* ∼ *pit*1*/*

Advised By

Associate Professor Heinz Schaettler

Department of Electrical and Systems Engineering

May 2013

Abstract

An algorithm for weekly workforce scheduling with 4-hour discrete resolution that optimizes for employee satisfaction is formulated. Parameters of employee availability, employee preference, required employees per shift, and employee weekly hours are considered in a binary integer programming model designed for automated schedule generation.

Contents

Chapter 1

Introduction and Background

1.1 Background

Recent healthcare legislation in the United States is affecting workforce management. The Patient Protection and Affordable Care Act defines full-time employees as those who work 30 or more hours per week, on average. Under the legislation, businesses with 50 or more employees are required to provide healthcare benefits to full-time employees, or pay a $2000 fine [1].

This legislation is disrupting workforce management. Specifically, the Wall Street Journal notes that these changes are driving employers to schedule more employees with more strict hour requirements [2]. Rather than employing a workforce of full-time employees, employers are increasingly hiring a larger workforce of part-time employees for 29-hour or less workweeks to avoid the penalty.

Scheduling more employees with additional constraints increases the difficulty of the scheduling problem, and current analog scheduling methods are proving inadequate.

Currently, managers of small workforces schedule employees by hand. In general, their goal is meeting feasibility constraints - scheduling employees when they are available to work. This is a largely subjective process, and there are many possible solutions, though these solutions may be difficult to obtain.

With the decreasing cost of computers and the rise of software as a service products, business owners are becoming more receptive to technological solutions to

common problems. In addition, with the recent *big data* trend, operations research and applied statistics are becoming mainstream. Tools like Mapquest and Google Search use complex mathematical models, yet have become integrated into consumers' everyday lives.

Scheduling is a classic operations research problem. By minimizing employee labor cost, a large system optimization is often used to place employees in shifts. Constraints such as shift lengths, employee weekly hours, and minimum number of shifts are often considered.

Corporations such as Taco Bell use binary or mixed integer programing models [3] with success to schedule thousands of employees every week. Recent trends show the propagation of optimized scheduling techniques to more businesses and organizations.

1.2 Existing Models

Current scheduling models are based on minimizing labor costs. Specifically, every time slot is assigned a minimum number of employees needed to work that shift, and the number of employees working may exceed that minimum in order ot fulfill minimum shift length constraints. On a large scale, minimizing employee labor costs provides a quantifiable financial benefit.

1.3 Motivating Example

This project was motivated by a coffee shop that operates 24-hours per day. It already schedules employees in four-hour shifts throughout the week, and employees consist of a mixture of full-time and part-time workers.

Currently, scheduling is done manually by the manager. Because of the long hours of the shop, employees availability on a weekly basis must be considered. In addition, employees have a strong preference for shift, specifically between night, morning, and afternoon shifts.

Current scheduling software does not meet the needs of the shop because it aims to minimize employee working hours, and because it treats an employee's preferred

shift as a hard constraint. In addition, because the shop has few employees and already schedules in 4-hour shifts, the software package's minimization of labor costs provides little benefit.

The coffee shop seeks new scheduling software that treats employee preference as a soft constraint, in addition to availability as a hard constraint. When provided with the number of employees required for each shift, and with the availability and preferred shifts of each employee, it should return a suitable work schedule. Constraints such as shifts per week and number of shifts per day should be considered. Furthermore, employees who choose not to prioritize shifts should not be penalized during scheduling.

1.4 Problem Statement

Design an algorithm that automates workforce scheduling in a way that provides a recognizeable benefit for small workforces of less than 100 employees where current cost-saving optimizations provide less utility.

Chapter 2

Approach

2.1 Proposed Model

A modified scheduling algorithm is proposed that optimizes for employee satisfaction instead of minimizing cost. In the setting of small and medium-sized businesses, improvements in morale may more tangible than the labor costs saved.

Current workforce scheduling algorithms gain these cost savings by scheduling employees down to 15-minute discrete resolution, including breaks. Per the motivating example, the workforces targeted by this algorithm do not require such precise scheduling. Instead, they seek a more basic interface that schedules employees in 4-hour blocks.

In addition to the minimum employees working per hour that current models provide, the proposed new model sets a maximum constraint of number of workers per shift. After specifying availability, employees select priority shifts every week. Each shift is assigned a weight on a per-employee basis, according to employee prioritization.

2.2 Project Goals

The goal of this project is to design a scheduling algorithm that takes a 7-day work week and analyzes it as 42 discrete shifts each 4 hours in length.

The manager sets how many employees are required at each shift throughout the week. If the business is closed, then 0 employees are required. Note, however, that the model should function for a business that never closes.

The manager also specifies the minimum and maximum shifts each employee must work each week. For instance, if a manager wish to keep someone from exceeding 30 hours of work per week, they specify a maximum of 6 shifts per week (28 hours).

Each employee may set their availability by specifying a boolean 'available to work' or 'unavailable to work' for every shift throughout the week. Then, they are provided with the option to set boolean 'preferred shift' or 'no preference' for every shift where they are available to work.

To comply with basic labor practices, the model also implements a limit on the number of shifts an employee may work per day. This is may be specifed by the manager, but in general may be treated as 2 shifts (8 hours) per day.

A feasible solution of the model treats employee availability, work limits, number of employees working per shift, and shifts per 24-hour segment as hard constraints. The objective function seeks to maximize the placement of employees in shifts based on preference.

2.3 Model Design

A binary integer programming (BIP) model was selected because of the discrete and boolean nature of the decision variable, which applies well to shift scheduling. Other scheduling algorithms have been implemented as mixed integer programs, but those models have focuses on complex shift overlaps between employees. Because employees in this model work in integral numbers of shifts with no overlap, such an model does not suitably apply.

Due to the nondeterministic time complexity of a BIP algorithm, heuristics serve an important role in providing quick runtimes and reaching feasibility. By making the decision variable have the same indices as the availability or preference inputs, straightforward heuristics for the first iteration based on these inputs are possible.

While the BIP model is nonlinear, removing the binary constraint on the decision variable and treating it into a continuous 0-1 interval constraint makes the problem

linear. Thus, a linear programming relaxation model may be used to generate the first BIP iteration. Specically, using the Simplex algorithm to solve the linear program, then rounding the continouous variables to discrete binary variables for the first interval provides a solution that is a reasonable first approximation for a feasible solution of the BIP model. The marginal effect on overall runtime by adding the linear programming relaxation is trivial due to the polynomial runtime of the Simplex algorithm.

A variety of algorithms for solving BIP models have been commercially implemented in a variety of software packages. In general, the NP-Complete classification of BIP problems means that runtimes have nondeterministic polynomial runtime and are computationally difficult.

2.4 Implementation

This model is best implemented in a Software as a Service web platform, where employers are able to manage parameters and employees are able to provide the necessary input remotely.

First, employers configure the workweek and anticipated load in the system. The minimum and maximum employees per shift are set, with a maximum of 0 meaning that the business is closed.

Separate models are needed for each role. For instance, separate models are needed for cooks and dishwashers because they are non-overlapping sets.

Finally, employers configure employees and working standards. Specifically, constraints on per-employee hours are set, and minimum and maximum shift lengths are specified.

2.5 Usage

Constraints in the model are designed so that the schedule is to be repeated every week to comply with labor parameters. When considering that the number of shifts an employee may work in a 24-hour period is limited, the first shift and the last shift of the week are considered to share a boundary, such that if the

schedule were repeated every week, work limit parameters would satisfied across this boundary.

If employee availability and preferences changes on a week-by-week basis, the employee availability matrix may be used to manually enforce worktime parameters without additional modifications to the model. For instance, if the prior workweek ended with three consecutive shifts for a particular employee, they could be marked as 'unavailable' for the first 3 shifts of the proceeding week such that they do not exceed the number of allowed shifts in a 24-hour period.

Chapter 3

Model

3.0.1 Indices

i: Employee
j: Shift

3.0.2 Parameters

$\text{Employee}_{min,j} \in \mathbb{Z}$: Minimum employees working at time
$\text{Employee}_{max,j} \in \mathbb{Z}$: Maximum employees working at time
$\text{Preference}_{i,j} \in \{0, 1\}$: Boolean Employee Shift Preference
$\text{WeightedPreference}_{i,j} \in \mathbb{R}$: Weighted Employee Shift Preference
$\text{Availability}_{i,j} \in \{0, 1\}$: Boolean employee availability at time
$\text{Shift}_{min,i} \in \mathbb{Z}$: Minimum employee shifts per week
$\text{Shift}_{max,i} \in \mathbb{Z}$: Maximum employee shifts per week

3.0.3 Decision Variable

1 if employee i is scheduled to work shift j; 0 otherwise.

$$x_{i,j} \in \{0, 1\}$$

3.1 Constraints

$$\sum_{i} Availability_{i,j} \geq Employee_{min,j} \tag{3.1}$$

For each i:

$$\sum_{j=k}^{k+5} x_{i,(j \bmod count(j))} \leq 2 \tag{3.2}$$

$$Shift_{min,i} \leq \sum_{j} x_{i,j} \leq Shift_{max,i} \tag{3.3}$$

$$Preference_{i,j} = Preference_{i,j} \cdot Availability_{i,j} \tag{3.4}$$

3.2 Employee Shift Weighting

The employee preference parameter is created to allow employees to designate shifts they prefer. The model then optimizes to give employees the shifts they desire.

The preferences parameter is calculated such that:

$$\sum_{j} Preference_{i,j} = \sum_{j} Availability_{i,j}$$

Should an employee choose not to specify priority shifts, or if an employee chooses to prioritize all shifts, the preference matrix should thus be equally weighted, and equal to the availability parameter:

$$Preference_{i,j} = Availability_{i,j}$$

Based on the objective function, priority shifts are upweighted, and based on the specified constraints the non-priority shifts must be downweighted. The upweighting factor is designated as α and the downweighting factor is designated as β.

Considering j = 4 with one specified priority shift:

$$(1+\alpha) + (1-\beta) + (1-\beta) + (1-\beta) = 4$$

Thus, in this case:

$$\alpha = 3\beta$$

Clearly, β must always be less than one. However, we also consider that employees who specify only a single priority shift have more weight placed on that shift than someone who prioritizes all but one shift. We specify that a prioritized shift may be no more than twice as weighted as a non-weighted shift, thus $\alpha < 1$.

$$\alpha_i = \frac{\sum_j Availability_{i,j} - \sum_j Preference_{i,j}}{\sum_j Availability_{i,j}}$$

Thus,

$$\beta_i = \alpha \frac{\sum_j Preference_{i,j}}{\sum_j Availability_{i,j} - \sum_j Preference_{i,j}}$$

In addition, if $\sum_j Availability_{i,j} = 0$ or if $\sum_j Availability_{i,j} = \sum_j Preference_{i,j}$,

$$\beta_i = \alpha_i = 0$$

The weighted preference matrix is this computed for each i:

$$WeightedPreference_{i,j} \begin{cases} 0, Availability_{i,j} = 0 \\ 1 + \alpha, Availability_{i,j} = 1 \& Preference_{i,j} = 1 \\ 1 - \beta, Availability_{i,j} = 1 \& Preference_{i,j} = 0 \end{cases}$$

An implementation of the weighting formula in Matlab is available in Chapter 5.

3.3 Objective Function

$$max\ Z = \sum_{i,j} x_{i,j} \cdot WeightedPreference_{i,j}$$

Chapter 4

Implementation and Results

4.1 Constraints

All of the constraints described formulated as hard constraints.

Equation (3.1) ensures that the employee availability spans the required employees.

Equation (3.2) limits every employee to 2 shifts per 24-hour period. This constraint formulates the boundaries of the beginning and end of the week as cyclical, such that if the schedule were repeated every week, an employee would not exceed 2 shifts in a 24-hour period.

Equation (3.3) ensures that every employee works between their minimum and maximum shifts per week, inclusive.

Equation (3.4) ensures that an employee may only have a "preferred" shift if they are available to work that shift.

4.2 Excel Implementation

A working implementation in Excel using the Solver package is available for download at the project website:

$$http://students.cec.wustl.edu/\sim pit1/$$

The implementation is designed so that it may be calculated using the standard Excel Solver package. It schedules 4 employees over a 72-hour period with cyclical constraints.

With 72 decision variables and 98 constraints, the problem successfully optimizes the model using the package's GRG Nonlinear engine.

Based on tests, the scheduler normally takes between 1 and 120 minutes to optimize this model.

4.3 Runtime

A BIP model is classified as an NP-Complete problem, which is among the most computationally-difficult models to optimize. The runtimes encountered in the above implementation had such varying runtimes because BIP models run in non-deterministic polynomial time.

Scaling the model for more employees requires significant computational power and an improved solver library. For an implementation with 13 employees and 42 shifts, there are 2^{546} possible boolean matrices, which is over 10^{164} solutions. Hence, an efficient solving package is required.

4.4 Heuristic

Based on tests with the model, the most efficient way to decrease runtime was to use a version of a greedy heuristic. This was implemented by using the availability matrix as the first iteration of the BIP model. By starting the model as assuming that every employee is scheduled for every shift when they are available, the availability constraint is immediately satisfied, and the algorithm seems to decrease runtime by moving toward feasibility more efficiently.

Tests with the linear programming relaxation technique to determine a first iteration for the BIP model had varying levels of effectiveness. When the α value was zero, the model often create an initial solution similar to the greedy heuristic. However, at high α values, such high weight placed on particular shifts made the

initial solution by the linear programming relaxation highly infeasible due to so few shifts being assigned.

In a commercial implementation, the greedy heuristic seems to provide the best first iteration solution in terms of runtime due to the meeting the availability constraint and providing a near-feasible first iteration.

4.5 Alterations

One main alteration that is possible in this model is making the employee availability constraint, equation (3.1), a soft constraint instead of a hard one. Thus, employees could be scheduled when they are unavailable.

By making this a soft constraint, employees would be unlikely to be scheduled when they are unavailable because the shift's weight is zero in the objective function calculation. However, in edge cases where not enough employees are available to work at a particular shift, the model would still return a schedule instead of encountering a feasibility error.

In a real-world implementation, minimizing errors and forcing the return of a non-ideal schedule would allow managers to make hand corrections after the creation of the schedule, instead of being forced to have all constraints met prior to returning a solution.

4.6 Model Improvements

The model would benefit from the addition of a constaint limiting time between shifts. The current model limits employees to two 4-hour shifts during every 24-hour period. However, it does not limit that those shifts be contiguous - hence, an employee could conceivably have 8-hour breaks between 4-hour shifts for multiple days in a row. This is undesirable because it interrupts sleep.

The solution to this is to add a constraint that enforces a minimum time between non-contiguous shifts. However, adding this constraint would be highly inefficient in a BIP model.

Including this constraint would be best implemented by re-formulating the model as a mixed integer program where the decision variable contains both a start time and a shift length.

4.7 Implementation

In a real-world implementation where there exist multiple roles in a workforce, for example having both a cook and a dishwasher who cannot fill each others roles, this algorithm should be implemented to treat every role in a company as an independent model with independent solutions.

Due to the low cost and high speed of modern computing, converting this algorithm into a Software as a Service (SaaS) product would be the ideal user interface, particularly because it makes collecting employee availability and preferencees from the remote workforce easier for the manager. The difficulty in a real-world implementation is handling infeasible inputs - for instance, not having enough employees who may work at a particular shift.

Basic scheduling needs are met by the model, and it excels at speed relative to other workforce management algorithms because the only nonlinearity is the binary aspect of the decision variables. However, the next step in a more advanced model is adding contiguity constraints such that there is a minimum break between non-continuous shifts. Such a nonlinear constraint adds significant computational difficulty.

Finally, due to the nondeterministic polynomial runtime of the algorithm and due to the nature of scheduling, the model does not need to be run to completion. Returning a non-optimal but feasible solution still satisfies the original problem statement because it automatically generates a schedule that meets all hard constraints, and any improvement over the worst-case scenario still provides recognizeable benefit to management and employees.

Chapter 5

Matlab Implementation of Weighted Preference Matrix

```
function [ weighted_preferences ] = weighted_shifts( availability, preference )
%WEIGHTED_SHIFTS Returns a weighted shift matrix

% Get sizes for loops
[num_employees num_shifts ] = size( availability );
% Initialize matrix by copying availability
weighted_preference = availability;

% Loop through each employee
for i = 1:num_employees

    % Count how many shifts they are available, and how many
    % they prefer
num_available = sum( availability(i,:) );
num_preferred = sum( preference( i,: ) );

if ( num_preferred == 0 ) || ( num_preferred == num_available )
% If they do not prefer any shifts, or if they prefer every shift
% The availability matrix weighting is correct (all 1s)
else
% we upweights and downweights based on number of preferred shifts
```

```
% upweight calculation
alpha = ( num_available - num_preferred ) / num_available;

% downweight calculation
beta = alpha * num_preferred / ( num_available - num_preferred );

% Loop through shifts and set weight
for j = 1:num_shifts
if availability( i , j ) == 1 && preference( i , j ) == 1
% upweight shift
weighted_preference( i , j ) = 1 + alpha;
elseif availability( i , j ) == 1 && preference( i , j ) == 0
% downweight shift
weighted_preference( i , j ) = 1 - beta;
else
% Already zero weight due to availability matrix
% weighted_preference( i , j ) = 0;
end
end
end
end


end
```

Bibliography

[1] http://www.ncsl.org/documents/health/ppaca-consolidated.pdf

[2] http://online.wsj.com/article/SB10001424127887324616604578304072420873666.html

[3] An Efficient Integer Programming Algorithm with Network Cuts for Solving Resource-Constrained Scheduling Problems http://mansci.journal.informs.org/content/24/11/1163.full.pdf+html

[4] A genetic algorithm approach for a constrained employee scheduling problem as applied to employees at mall type shops http://www.sersc.org/journals/IJAST/vol14/1.pdf

[5] An integer programming approach to reference staff scheduling http://www.sciencedirect.com/science/article/pii/0306457385900913