An Automated Employee Timetabling System for Small Businesses

Richard Hoshino, Aaron Slobodin, and William Bernoudy

Quest University Canada, Squamish, British Columbia, Canada

Abstract

Employee scheduling is one of the most difficult challenges facing any small business owner. The problem becomes more complex when employees with different levels of seniority indicate preferences for specific roles in certain shifts and request flexible work hours outside of the standard eight-hour block. Many business owners and managers, who cannot afford (or choose not to use) commercially-available timetabling apps, spend numerous hours creating sub-optimal schedules by hand, leading to low staff morale.

In this paper, we explain how two undergraduate students generalized the Nurse Scheduling Problem to take into account multiple roles and flexible work hours, and implemented a user-friendly automated timetabler based on a four-dimensional integer linear program. This system has been successfully deployed at two businesses in our community, each with 20+ employees: a coffee shop and a health clinic.

Introduction

The Nurse Scheduling Problem (NSP) seeks to assign shifts to nurses at a hospital. Each nurse has hard constraints on days they must have off, as well as soft constraints on desired and undesired shifts during each 24-hour day. The challenge is to ensure that all of the nurses have a feasible schedule while simultaneously satisfying the hospital's requirement that a certain number of nurses be working at all times. This well-known operations research problem is NP-hard.

Given the applications of the NSP to fields other than health care, numerous researchers have tackled this problem over many years (Ernst et al. 2004), developing sophisticated algorithms that involve methods in constraint programming, including simulated annealing and Tabu search. Many of these methods have found their way into commercially-available scheduling applications that are used by organizations, especially large businesses with 100+ employees.

The lead author is a former employee of the Government of Canada, who helped implement an Integer Linear Program (ILP) solver to optimize the allocation of shifts to customs officers at airports. The rotating 56-day schedules were created by applying the NSP to this specific context, where each 30-minute block had a required number of officers to reduce waiting times for passengers clearing customs.

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Solving the problem for Canadian customs officers was a relatively simple problem given that every individual had the exact same job description with only two possible shift patterns: either 8-hour shifts with five days on and two days off, or 11.5-hour shifts with four days on and four days off. The final posted schedule was an $N \times 56$ matrix, with N being the number of staff. The Customs Union adopted the approach known as the Serial Dictatorship (Hoshino and Raible-Clark 2014), where the officers picked their row (i.e. their shifts for the next eight weeks) based on their seniority.

This process is sub-optimal since employees do not have the opportunity to indicate their preferences for working certain hours on specific days, causing an enormous gap between satisfied veteran employees who received the shifts they wanted and unsatisfied rookies who did not.

A much better solution would have been to create an ILP to maximize preference points, where the objective function is $\Sigma_e \Sigma_d \Sigma_s S_e \cdot P_{e,d,s} \cdot X_{e,d,s}$, with S_e being the seniority of employee e, $P_{e,d,s}$ indicating e's preference to work on day d in shift s, and $X_{e,d,s}$ being the binary variable indicating whether e is assigned to work on that specified day and shift.

However, this improved solution would not have been sufficient for small businesses: it is unreasonable to ask staff to commit to a rotating 56-day schedule; employees want schedules that are more flexible than the two shift patterns described above; and unlike customs officers, employees have different roles and responsibilities.

For example, at the most popular cafe in our community, the 30 staff fill multiple roles (barista, line worker, and back-of-kitchen baker), with numerous employees qualified for more than one role. Employees want morning shifts on some days and evening shifts on others. They have many hard constraints on which shifts and days they are unavailable while expressing strong preferences for certain roles, such as being a barista on a busy shift and a line worker on a slow shift. Moreover, they have different levels of seniority. The cafe manager spent up to six hours a week constructing weekly schedules for her staff by hand, a process that was time-consuming and frustrating to both her and her team.

The other two authors of this paper are undergraduates. As frequent customers at this cafe, we were familiar with the manager's weekly scheduling conundrum and were eager to tackle a real-world problem. Last year, we successfully deployed our AI-based automated timetabler for this business.

After implementing the employee scheduling system for the cafe, we were contacted by the owner of a local health clinic, a thriving small business in our community that offers treatments including acupuncture, massage therapy, and chiropractic care. Despite the different contexts, we developed and deployed a single scalable solution for both organizations. Realizing the potential impact of our product, an automated preference-maximizing employee timetabling system, the two soon-to-be-graduating student authors recently founded a scheduling consulting company.

In the next section, we present a brief literature review, after which we define the specific optimization problem that falls under the broad AI field of constraint satisfaction. We present our generalization of the NSP, a four-dimensional integer linear program (ILP) that successfully solves the optimization problem for both businesses and can be extended to numerous other contexts beyond cafes and health clinics. We then illustrate how this automated timetabler was successfully deployed, present screenshots, and conclude the paper with some open problems and ideas for future research.

Background and Related Work

In the literature, there are two broad categories of Nurse Scheduling Problems: cyclical and non-cyclical. The latter NSP is much harder as the business must create a new schedule for each period, usually every 7 or 14 days. There are many AI-based methods of solving non-cyclical staff scheduling problems (Cheang et al. 2003), including game-theoretic models of strategic behaviour, where we view the process of allocating shifts to employees as a finite repeated game with perfect information (Kalinowski et al. 2013).

Some businesses construct their employee schedule using a preferential bidding system (Gamache et al. 1998) and others have employed genetic algorithms to modify existing schedules to incorporate small changes while maintaining most of the original schedule (Moz and Pato 2007). An evolutionary algorithm, combining scatter search with hill-climbing, successfully solved a real-world NSP by creating solutions from multiple parents and using new methods for measuring similarity between solutions (Burke et al. 2010).

The same authors created a hybrid multi-objective model that used a two-step process to solve highly-constrained nurse rostering problems. They began with an ILP to solve a sub-problem, consisting of all hard constraints and a subset of the soft constraints, and then a variable neighbourhood search to improve the ILP results (Burke, Li, and Qu 2010).

A different team of researchers applied an integer solver to solve a three-dimensional nurse scheduling problem in a French public hospital. Their solution was to create variables to express equally-qualified anesthesiology nurses working during five different shifts throughout the schedule period (Trilling, Guinet, and Magny 2006).

Another paper increases the complexity of the NSP by considering two types of nurses and adding constraints to model staffing requirements around these types. They used a two-step approach, applying an ILP solver to optimize a different objective function at each step. This allowed them to improve employee satisfaction and reduce cost and overtime at two hospitals (Wright and Mahar 2013).

Though there is much previous work on the NSP, none of the solutions applied to our two problems due to the unique staffing constraints imposed by our clients' businesses.

Problem Definition

The standard NSP is an example of a constraint satisfaction problem, which asks whether there exists a feasible assignment of employees to days and shifts satisfying the hard constraints of the nurses and the hard constraints of the hospital.

The more general version of the NSP is a combinatorial optimization problem, which asks for the best assignment of employees to days and shifts satisfying all of the hard constraints while maximizing the preferences of the nurses.

Both versions of the NSP can be set up as a 0-1 ILP, an NP-complete problem in which each unknown variable $X_{e,d,s}$ represents whether employee e works on day d in shift s. The goal is to maximize $c^T x$, subject to $Ax \leq b$ and $x \in \{0,1\}^n$, where c and b are $n \times 1$ vectors and A is an $n \times n$ matrix.

In the NSP, we have $n = |E||D||S|$, where $|E|$ is the number of employees, $|D|$ is the number of days, and $|S|$ is the number of shift types. Thus, each element in vector c represents some coefficient $C_{e,d,s}$ that is a function of employee e's seniority and his/her preference for working on day d in shift s.

The NSP does not solve the scheduling problem for either of our two clients due to each business assigning employees to specific non-interchangeable "roles". Furthermore, each of our clients had a unique staffing constraint that was not addressed in the NSP, explained as follows:

(a) The cafe employs individuals who prefer certain roles on different days. For example, one employee wants to be a barista on Saturdays and a line worker on other days. If $P_{e,d,r}$ is the "preference score" of employee e working on day d in role r, then $P_{e,d_1,r}$ might not equal $P_{e,d_2,r}$ if $d_1 \neq d_2$. Furthermore, employees have a different level of seniority for each role in the business. Thus, if $S_{e,r}$ is the "seniority score" of employee e in role r, then S_{e,r_1} might not equal S_{e,r_2} if $r_1 \neq r_2$.

(b) The health clinic employs individuals who want flexible work hours outside of the standard eight-hour shift. For example, one therapist wants to work 2 hours some days and 8 hours on other days. Thus, instead of partitioning each day into fixed-length shifts, we must consider variable-length "blocks" that begin and end at any hour.

At both businesses, the owners did their best to assign work to maximize "employee satisfaction" while ensuring they had the correct number of staff members in their roles for each hour of operation. These owners, however, naturally found it challenging to solve these combinatorial optimization problems each week using pencil and paper, leading to poor employee retention and disgruntled staff members who felt that their bosses favoured certain individuals over others.

In the next section, we present how both scheduling problems can be reduced to a single scalable four-dimensional model, where the four dimensions are the sets of employees, days, roles, and shift blocks (e.g. 8AM to 12PM).

Mathematical Model

Let E be the set of employees, D be the set of days, R be the set of roles, H be the set of hours, and B be the set of shift blocks. For example, the cafe has $E = \{1, 2, \ldots, 30\}$, $D = \{1, 2, \ldots, 7\}$, $R = \{\text{Barista}, \text{Line}, \text{Kitchen}\}$, $H = \{1, 2, \ldots, 19\}$, and $B = \left\{\bigcup_{1 \le i \le j \le 19} [i, j]\right\}$.

The cafe staff work in shifts that span a 19-hour time period each day, with the first employees arriving at 5AM to set up and the last employees leaving at midnight, after finishing all of the baking for the following day's service. We let 5AM-6AM refer to hour 1, 6AM-7AM refer to hour 2, and so on.

Each block is an ordered pair $[i, j]$ marking the start and end of the shift. For example, $[1, 19]$ corresponds to the entire 19-hour shift and $[8, 8]$ is the shift that begins at noon and ends at 1PM. If there are $|H| = n$ values in set H, then there are $\binom{n+1}{2}$ possible pairs in set B.

For each $e \in E$ and $r \in R$, let $S_{e,r}$ be the seniority of employee e in role r, i.e. more experienced employees are assigned higher points.

For each $e \in E, d \in D, r \in R, b \in B$, define $P_{e,d,r,b}$ to be the preference of employee e working on day d in role r in block b. This coefficient is positive if e is available in this capacity, with higher scores for more desired shifts. This coefficient is 0 if the employee does not wish to work this day (a soft constraint) and negative if the employee cannot work on this day (a hard constraint).

Let $X_{e,d,r,b}$ be the binary variable that is equal to 1 if employee e works on day d in role r in block b, and 0 otherwise. Finally, let $f(S_{e,r}, P_{e,d,r,b})$ be a function of the previous two coefficients, representing the overall suitability of assigning this specific day/role/block to employee e.

Therefore, the objective function of our ILP is:

$$\text{maximize} \quad \sum_{e \in E} \sum_{d \in D} \sum_{r \in R} \sum_{b \in B} f(S_{e,r}, P_{e,d,r,b}) \cdot X_{e,d,r,b}$$

Note that $f(S_{e,r}, P_{e,d,r,b}) = S_{e,r} \cdot P_{e,d,r,b}$ is the natural way to combine our two coefficients; however, f does not have to be this simple multiplicative function, especially if the business wishes to assign different weights for preference and seniority.

Each employee is assigned at most one block a day in a single role. Thus, an employee is not allowed to be a barista from 12PM to 3PM and then work in the kitchen from 3PM to 6PM. Furthermore, an employee's shift must be a continuous block of hours, i.e. on the same day she cannot work from 12PM to 3PM and then from 7PM to 10PM. Thus,

$$X_{e,d,r,b} \in \{0, 1\} \quad \forall e \in E, d \in D, r \in R, b \in B \tag{1}$$

$$\sum_{r \in R} \sum_{b \in B} X_{e,d,r,b} \le 1 \quad \forall e \in E, d \in D \tag{2}$$

For each $b \in B$, define $len(b)$ to be the length, in hours, of block b. Thus, if $b = [h_1, h_2]$, then $len(b) = h_2 - h_1 + 1$.

Let $MAXHR_{e,d}$ and $MINHR_{e,d}$ be the maximum and minimum number of hours employee e wishes to work if s/he is assigned a shift on day d. Then, for all $e \in E, d \in D, r \in R$, we have

$$X_{e,d,r,b} = 0 \quad \forall b \in B \text{ with } len(b) > MAXHR_{e,d} \tag{3}$$

$$X_{e,d,r,b} = 0 \quad \forall b \in B \text{ with } 0 < len(b) < MINHR_{e,d} \tag{4}$$

Let $HOURS_e$ and $DAYS_e$ be the maximum number of hours and days that employee e is able to work over the $|D|$ days. Then,

$$\sum_{d \in D} \sum_{r \in R} \sum_{b \in B} len(b) X_{e,d,r,b} \le HOURS_e \quad \forall e \in E \tag{5}$$

$$\sum_{d \in D} \sum_{r \in R} \sum_{b \in B} X_{e,d,r,b} \le DAYS_e \quad \forall e \in E \tag{6}$$

An employee who closes a shift cannot open the next day. And so, for all $e \in E, d \in D$, we have

$$\sum_{r \in R} \sum_{b \ni |H|} X_{e,d,r,b} + \sum_{r \in R} \sum_{b \ni 1} X_{e,d+1,r,b} \le 1 \tag{7}$$

This covers all of the constraints from the perspective of the employees. From the perspective of the business, let $MAXNUM_{d,r,h}$ and $MINNUM_{d,r,h}$ be the maximum and minimum number of employees that need to be working on day d in role r in hour h. Then, for all $d \in D, r \in R, h \in H$, we have

$$\sum_{e \in E} \sum_{b \ni h} X_{e,d,r,b} \le MAXNUM_{d,r,h} \tag{8}$$

$$\sum_{e \in E} \sum_{b \ni h} X_{e,d,r,b} \ge MINNUM_{d,r,h} \tag{9}$$

Thus, our ILP maximizes our objective function, subject to the nine constraints described above.

This is the general solution to our scheduling problem. Both businesses can optimally staff their employees by solving a specific version of the above ILP:

(a) The cafe has $|E| = 30$, $|D| = 7$, and $|H| = 19$. Each shift is exactly 8 hours in length, and so we only consider blocks $b \in B$ with $len(B) = 8$. Thus, $|B| = 12$. Finally, $|R| = 3$, as each employee has one of three roles: a barista, a line worker, or a back-of-kitchen baker.

(b) The health clinic has $|E| = 15$, $|D| = 14$, and $|H| = 15$. The 15 massage therapists want to work in blocks that are between 2 and 8 hours. Thus, $|B| = 14 + 13 + 12 + 11 + 10 + 9 + 8 = 77$. Each therapist has a preferred room based on various factors, including lighting, size, and proximity to the main entrance. Therapists stay in the same room within each block. In this context, the roles are the rooms; as there are five massage rooms, we have $|R| = 5$ and $MAXNUM_{d,r,h} = 1$ for all $d \in D, r \in R, h \in H$.

Proof of Concept

Many business owners view employee scheduling as an "art" rather than as a "science" – or perhaps, more appropriately, as an AI-optimization problem. To convince the cafe manager of the merits of our ILP model, we asked her to provide us with an Excel sheet listing the seniority scores $S_{e,r}$ of each employee as well as their preference scores $P_{e,d,r,b}$ for shifts in the upcoming week. Conveniently, much of this was already documented by this manager on an Excel sheet that she used to construct her weekly schedules. In our objective function, we set $f(S_{e,r}, P_{e,d,r,b}) = S_{e,r} \cdot P_{e,d,r,b}$.

In order to compare our two approaches, we asked for the rest of the constraints, from both the perspective of the employees (e.g. $DAYS_e$) as well as the management (e.g. $MINNUM_{d,r,h}$). The following table compares the results of the manager's pencil-and-paper solution and our ILP solver that is guaranteed to return a solution that maximizes the objective function.

System	Objective Function	Time Spent
By Hand	932 points	5.5 hours
Our ILP	1068 points	8.5 seconds

With $|E| = 30$ and $|D| = 7$, our ILP runs in under 9 seconds on an old laptop purchased in 2012, specifically a 4GB MacBook Pro with a 2.30 Ghz Intel i7 processor. To demonstrate the robustness of our program, we increased these two variables in case the cafe wanted to hire additional staff (especially during the peak summer months) and change to a 14-day schedule. The results were as follows:

$\|E\|$	$\|D\|$	Time Spent
30	7	8.5 seconds
40	7	17 seconds
30	14	131 seconds
40	14	358 seconds

After demonstrating that we could rapidly generate a provably-optimal schedule, the manager and owner realized the benefit of our product: not only would this program save them time, it would also relieve their stress and produce weekly schedules that would boost employee morale.

We had our first client.

System Design and Deployment (Part 1)

Our ILP solver was coded in Maple (www.maplesoft.com), a computer algebra system that was well-known to the authors. The cafe manager purchased a copy of Maplesoft and the Maplesoft worksheet containing our ILP was installed on her laptop.

Since the manager was already using Excel to keep track of her employees' preferences, it was only a small change to run the Maplesoft program, which used Excel for both the inputted ILP variables and the outputted final schedule.

Our goal was to create an easy-to-use Excel-based GUI that would allow the cafe to generate an automated schedule each week, without the manager having to modify the Maplesoft worksheet in any way.

In the Excel tab below, we see that each shift must be covered by a fixed number of employees in a specific role. Thus, for the cafe, $MAXNUM_{d,r,h} = MINNUM_{d,r,h}$ for all d, r, h. The other tabs indicate the values of the other variables, including the coefficient variables $S_{e,r}$ and $P_{e,d,r,b}$.

Number of Employees Required Per Day Per Role Per Hour

Position	Hours	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Barista	5AM to 1PM	2	2	2	2	2	2	2
Barista	1PM to 9PM	1	1	1	1	1	1	1
Barista	9PM to 12AM	0	0	0	0	0	0	0
Line	5AM to 1PM	1	1	1	1	1	1	1
Line	1PM to 9PM	1	1	1	1	1	1	1
Line	9PM to 12AM	0	0	0	0	0	0	0
Kitchen	5AM to 1PM	1	1	1	1	1	1	1
Kitchen	1PM to 4PM	0	0	0	0	0	0	0
Kitchen	4PM to 12AM	1	1	1	1	3	2	3

As each shift is 8 hours long, we have $MAXHR_{e,d} = MINHR_{e,d} = 8$ for all $e \in E, d \in D$. Thus, for each employee e, we must have $HOURS_e = 8 \times DAYS_e$.

The manager would then open our Maplesoft worksheet. At the click of the "Execute the Entire Worksheet" button, the program would import the Excel sheet, read the data, solve the ILP, and output the optimal solution in Excel. Below is one week's output of the first 15 employees:

Name	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Employee 1	Barista 5AM	Barista 5AM	Barista 5AM	Barista 5AM			
Employee 2							
Employee 3	Barista 1PM			Barista 1PM	Barista 1PM	Barista 1PM	
Employee 4		Line 5AM	Line 5AM	Line 5AM			
Employee 5	Line 5AM		Barista 1PM				
Employee 6	Kitchen 4PM						Kitchen 4PM
Employee 7					Kitchen 4PM	Kitchen 4PM	
Employee 8	Barista 5AM	Barista 5AM					Barista 5AM
Employee 9					Barista 5AM	Barista 5AM	
Employee 10	Kitchen 5AM	Kitchen 5AM					
Employee 11					Line 5AM	Line 5AM	Barista 5AM
Employee 12							Barista 1PM
Employee 13	Line 1PM	Line 1PM	Line 1PM	Line 1PM	Line 1PM		
Employee 14			Barista 5AM	Barista 5AM	Barista 5AM	Barista 5AM	
Employee 15				Kitchen 4PM	Kitchen 4PM	Kitchen 4PM	

If employee e could not work day d in role r in block b, then our Maple program automatically set $P_{e,d,r,b} = -10000$. This large negative coefficient was used for all hard constraints. As the Excel output listed the value of the objective function, the manager could check whether a hard constraint had been violated just by verifying that the objective value was positive.

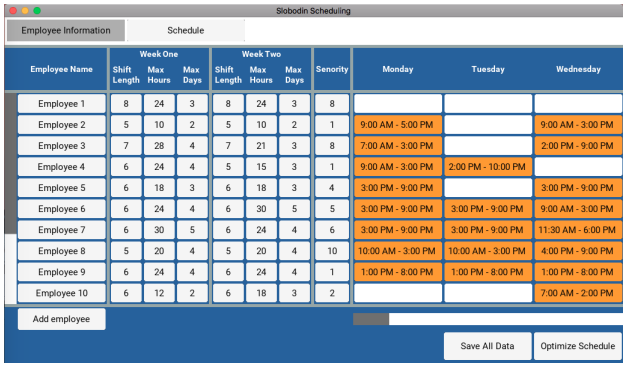
The cafe purchased our automated timetabler in July 2016 and has been using it ever since to produce weekly schedules for its employees. Depending on the season, $|E|$ is at least 30 and at most 40, with many of the employees being part-time staff who work two or three shifts each week.

Given that there are $n = |E||D||R||B|$ variables in our ILP, we have $7560 \leq n = |E| \times 7 \times 3 \times 12 \leq 10080$. As demonstrated in our proof of concept, the runtimes for each week are at most twenty seconds, from start to finish.

Alas, our Excel/Maplesoft solution was inadequate for the integrated health clinic as the clinic owner asked us to create two-week schedules, where therapists could vary their shifts between 2 and 8 hours. With $(|D|, |B|) = (14, 77)$ compared to $(|D|, |B|) = (7, 12)$, our runtime was too slow. As a result, we required a new solution, whose successful deployment is explained in the next section.

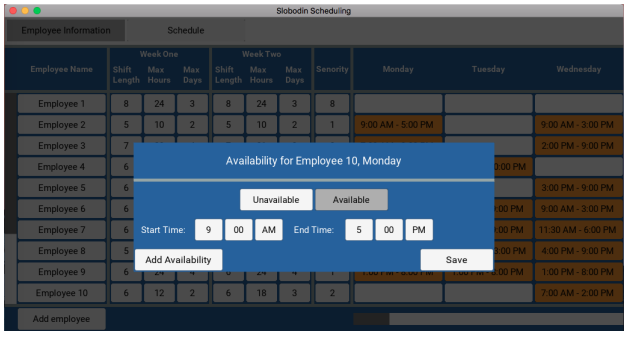
System Design and Deployment (Part 2)

Our improved solution is a local desktop application written in Python using Kivy, the open-source GUI library. For the health clinic, we designed a GUI that allows the owner to input each therapist's desired length of shift, maximum days and hours they can work over the next 14 days, and their seniority (i.e. number of years at the clinic).



Therapist e indicates a desired shift length for each week, which equals $MAXHR_{e,d}$ for $1 \leq d \leq 7$ and $8 \leq d \leq 14$. At the request of the owner, we set $MINHR_{e,d} = 0$, for all e and d, to ensure that senior therapists would get (close to) their desired number of hours. In the above table, marked in orange, are the times of availability for each therapist.

These times of availability can be modified via a modal popup window for each day in the 14-day period.

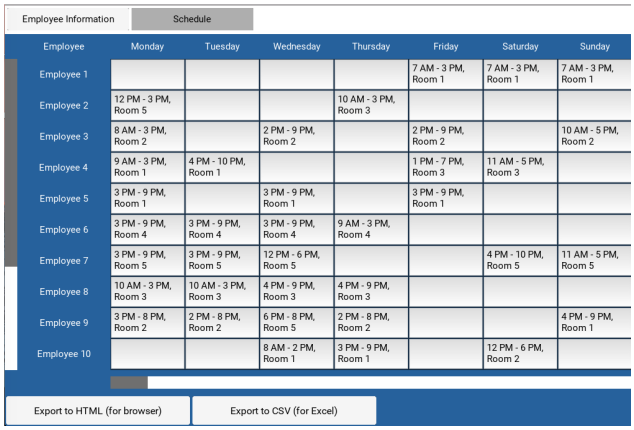


This information determines the value of the preference coefficient $P_{e,d,r,b}$, which is maximized whenever the length of a feasible block matches the employee's desired shift length. In the above table, Employee 3 desires seven-hour shifts, and is available between 7AM and 3PM on Monday. Thus, the Monday block from 7AM to 2PM, as well as the Monday block from 8AM to 3PM, are most ideal, and are given the maximum coefficient of 10.

We use the formula $P_{e,d,r,b} = 10\left(\frac{len(b)}{MAXHR_{e,d}}\right)$ whenever block b is a subset of the hours in which employee e is available on day d. If block b is not a subset of these hours or $len(b) > MAXHR_{e,d}$ then $P_{e,d,r,b}$ is set to -10000 and treated as a hard constraint. For example, the Monday block from 7AM to 12PM is given a coefficient of $10\left(\frac{5}{7}\right) \approx 7.14$. On the other hand, the blocks from 7AM to 3PM and 9AM to 4PM each have a coefficient of -10000.

The 20-employee clinic has seven dedicated rooms: one for chiropractic, one for nutrition and dietetics, and five for acupuncture and massage therapy. While the first two are simple to coordinate without any optimization software, the last one is most challenging as there are fifteen therapists who are competing for the use of five rooms.

The goal of the ILP is to maximize the objective function, and so the optimal solution will fill up the five rooms as much as possible. When the "Optimize Schedule" button is clicked, all of the input is aggregated and sent to a script that builds and solves the corresponding ILP. Below is one week's schedule produced for the health clinic, showing the first ten therapists.



The ILP is solved using the open-source COIN-OR-CBC solver using PULP, a Python interface (COIN-OR 2017). With $|E| = 15, |D| = 14, |B| = 77$, the ILP is solved for the acupuncturists and massage therapists in just 9.0 seconds, using the same stand-alone laptop previously mentioned.

This same solver reduces the time taken for the Proof of Concept problem for the cafe from 8.5 seconds to just 0.38. Therefore, this new system is significantly faster than our initial Excel/Maplesoft solution.

To optimize the schedule for the acupuncturists and massage therapists, we intentionally chose to set $|R| = 1$ and $MAXNUM_{d,r,h} = 5$, rather than $|R| = 5$ and $MAXNUM_{d,r,h} = 1$. This is equivalent to setting $P_{e,d,r,b}$ constant for all values of r. We did this because we realized that the ILP could be manipulated by therapists who could intentionally mark all five rooms as "most-desired", to maximize the possibility of being assigned work. Not only does our system run faster (since n is reduced by a factor of 5), our final solution is fairer.

A greedy algorithm is then used to assign the therapists to rooms, with the most senior employee being assigned their most-preferred available room. In the schedule view pictured above, the clinic owner can export the final product to HTML for easy printing.

The application is compiled with Cython and cx_Freeze and runs on Mac, Linux, and Windows. The two undergraduate authors maintain these applications. Each application is specifically customized to the needs of the business and updates are mostly simple bug fixes with the GUI.

Results

The Excel/Maplesoft system was deployed at the cafe in July 2016. The Python program was deployed at the health clinic in June 2017 and at the cafe immediately thereafter due to the new system's increased functionality and speed.

Both businesses report tremendous satisfaction with the final deliverable, noting increased convenience and efficiency as well as better satisfaction among employees who prefer the impartial "mathematical black box" over the manual pencil and paper schedule that was perceived to be biased in favour of certain individuals.

In particular, the therapists at the health clinic are appreciative of the increased hours they are able to work, with the owner delighted that nearly every room is being used each hour of each day. In the past, the clinic had divided each day into the A-block (7AM to 3PM) and the B-block (3PM to 10PM), where each therapist could only have a subset of hours within their given block.

As a result of our solution, this simplification is no longer necessary. On any given day, each room could be used by three or four therapists; furthermore, the therapists report great satisfaction in being able to work shifts that straddle the two blocks, especially one therapist with a young child who specifically requests to work between 12PM and 4PM.

The cafe manager recently e-mailed us a testimonial, reporting: "The amount of personal time you have spent developing your program specifically to my schedule-making needs is invaluable, and I am so grateful for the ongoing support that you have provided".

Both businesses are able to use the product to test viable future schedules based on predicted demand and better anticipate needed changes to employee hiring. And most importantly, the clinic owner and cafe manager no longer need to endure the weekly stress of creating the schedules themselves; instead, they can rely on our program to instantly generate provably-optimal timetables for their employees.

Maintenance

Our deployed solution can easily be maintained whenever the business hires or fires employees, and has changes in employee preferences, seniority, as well as operating hours.

The Python program has an "Add employee" button in the bottom-left corner of the screen. When the ILP is solved, the program first determines $|E|$, the number of employees, based on the number of rows that contain information. Similarly, our application's simple GUI allows the user to make rapid changes to the data. For example, the health clinic recently had two therapists resign their positions, and hired three therapists to replace them. These new therapists were given the lowest seniority rating, and the remaining employees had their ratings bumped up accordingly.

Modifying existing features is easily done. To illustrate, each of the following fixes is simple, given the scalability of our four-dimensional ILP: the cafe creating a new role beyond the existing three jobs (barista, line, kitchen), the health clinic building an additional room, and either business changing their operating hours.

Additional features do require additional work, but the amount of effort is not substantial. For example, the clinic owner requested that each therapist's room preference be taken into consideration when creating the schedules. This required one of the student authors to modify the Python code as well as the GUI, which took just one full day.

Finally, our application can be applied to numerous other businesses beyond this specific cafe and health clinic, as many businesses have the same general features (e.g. employee roles, seniority, operating hours, employee preferences, and hours of availability), with their objective function customized to suit each business.

To provide just one example, we implemented an automated scheduler for our local library shortly after we submitted the initial version of this paper. It took less than a week from the initial contact (the head librarian contacting the lead author) to the time we trained a library staff member to use the implemented application.

The solution was particularly simple as there were only two roles, shelving and circulation, with a dozen employees and no consideration of seniority. After the program was implemented, our community newspaper ran a feature article on our work, in which the head librarian noted how our "willingness to jump in on this project and use [our] expertise to work with the library staff to create a scheduling solution has improved the scheduling process tremendously".

Next Steps

Our deployed solution has a niche in that it maximizes the total preferences of the employees, and allows for flexible work hours. We have examined many commercial scheduling programs and have found this combination to be the unique selling point of our automated timetabler. We recognize the impact and value of this flexible preference-optimizing scheduling program, especially for family-run businesses that place such a premium on employee satisfaction to ensure retention.

Realizing the potential of this four-dimensional ILP beyond cafes and health clinics, the two student authors recently founded a scheduling consulting company and are in the initial stages of establishing the business.

Before signing any new clients, we plan to improve the functionality of our software in the following two ways:

(a) Instead of working from the paradigm where managers gather preferences from all of their employees every scheduling period and enter it into the program, a web-based app could allow for employees to enter this data themselves on their own devices. Managers would then be presented with the same data view that our application now provides and generate the schedules directly.

(b) Employees are sometimes unable to come to work due to illness, family emergencies, transit problems, or other legitimate reasons. In these last-minute circumstances, the manager/owner has to scramble to find someone to cover their shift or fill the shift themselves. Our plan is to improve the software by outputting the employees that could be "on call" for a specific shift on a given day, without violating any hard constraints.

Open Problems and Future Ideas

Here are some open problems and ideas for future research.

(i) By adding a randomization component to the ILP, we can reduce the possibility that senior employees always get their desired shifts. This would occur by decreasing the preference coefficient $P_{e,d,r,b}$ by suitably small random numbers. Another approach is that employees who were assigned less desirable shifts one week get a slight bump in their preference coefficients the following week. These two approaches would create even greater buy-in among the employees, who would realize that the system is not biased in favour of a particular individual.

(ii) Some business prefer to have schedules with variety, i.e. employees should not work the same shift and role multiple times within a single schedule period. A simple solution would be to add hard constraints limiting the number of identical shifts and roles, but perhaps there are other methods that naturally take advantage of our general ILP model. For example, perhaps the key is to change the multiplicative coefficient function from $f(S_{e,r}, P_{e,d,r,b}) = S_{e,r} \cdot P_{e,d,r,b}$ to something else.

(iii) Another improvement would be for businesses to enter both a seniority level as well as a skill level for each employee and role. Constraints could then take this variable into account by ensuring a minimum total skill level for all the employees working in a certain role and shift, or perhaps ensuring that two junior employees are never working at the same time.

(iv) A clever employee can lie about their preferences in order to land their desired shifts. For example, suppose 3PM to 9PM is a popular block for the health clinic. A massage therapist who actually prefers 8-hour shifts can dishonestly set $MAXHR_{e,d} = 6$ to ensure that $P_{e,d,r,b} = 10$ for this popular six-hour block, rather than the true coefficient of $P_{e,d,r,b} = 10 \times 6/8 = 7.5$. We wonder how this ILP can be made less-manipulable, so that employees have the incentive to report their preferences truthfully.

Conclusion

We are delighted to report that our work has inspired a culture of "mathematical activism" at our small 700-person liberal arts university. Though we have no defined math or computer science majors, students are seeking to apply what they learn in these courses to solve real-life problems.

This journey started when the faculty author teamed up with an undergraduate student to automate the process of assigning students to courses via a selection "draft" similar to the process used in professional sports leagues (Hoshino and Raible-Clark 2014). The success of this project led the faculty author to pursue other opportunities for his students, most notably the research presented in this paper.

We were able to teach several undergraduates how to modify our automated employee timetabling system to optimally schedule the campus Emergency Medical Services team as well as the on-call shifts of the student-leaders serving as Floor Representatives in our residences.

Most ambitiously, a fourth-year student was inspired to create his own ILP to build a university-wide course timetable. This project became his Honours Thesis, and his optimal assignment of 40 professors to 240 courses across 8 months was accepted by our senior administration, as well as the entire faculty, and has been implemented for the 2018-19 academic year.

Acknowledgments

We gratefully acknowledge Amanda Desjardins and Adrian Blachut, the manager and owner of the Zephyr Cafe, as well as Susan Chapelle, the owner of Squamish Integrated Health. We would also like to thank Kyla Slobodin for providing the initial motivation for this work.

References

Burke, E. K.; Curtois, T.; Qu, R.; and Berghe, G. V. 2010. A scatter search methodology for the nurse rostering problem. *Journal of the Operational Research Society* 61(11):1667–1679.

Burke, E. K.; Li, J.; and Qu, R. 2010. A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. *European Journal of Operational Research* 203(2):484–493.

Cheang, B.; Li, H.; Lim, A.; and Rodrigues, B. 2003. Nurse rostering problems – a bibliographic survey. *European Journal of Operational Research* 151(3):447–460.

COIN-OR. 2017. CBC: A COIN-OR integer programming solver. `https://projects.coin-or.org/Cbc`. [Online; accessed 10-November-2017].

Ernst, A.; Jiang, H.; Krishnamoorthy, M.; and Sier, D. 2004. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research* 153:3–27.

Gamache, M.; Soumis, F.; Villeneuve, D.; Desrosiers, J.; and Gelina, E. 1998. The preferential bidding system at Air Canada. *Transportation Science* 32(3):246–255.

Hoshino, R., and Raible-Clark, C. 2014. The Quest draft: an automated course allocation algorithm. *Proceedings of the 26th IAAI Conference on Artificial Intelligence (IAAI-14)* 2906–2913.

Kalinowski, T.; Narodytska, N.; Walsh, T.; and Xia, L. 2013. Strategic behavior when allocating indivisible goods sequentially. *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI-13)* 452–458.

Moz, M., and Pato, M. V. 2007. A genetic algorithm approach to a nurse rerostering problem. *Computers & Operations Research* 34(3):667–691.

Trilling, L.; Guinet, A.; and Magny, D. L. 2006. Nurse scheduling using integer linear programming and constraint programming. *12th IFAC Symposium on Information Control Problems in Manufacturing* 39(3):671–676.

Wright, P. D., and Mahar, S. 2013. Centralized nurse scheduling to simultaneously improve schedule cost and nurse satisfaction. *Omega* 41(6):1042–1052.