DESIGN OF AN AUTOMATED EMPLOYEE SCHEDULING SYSTEM

A Senior Project submitted

In Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Science in Industrial Engineering

to the Faculty of California Polytechnic State University,

San Luis Obispo

by

Matthew Cameron and Yuriy Goldman

March 2013

Graded by:_____ Date of Submission_____

Checked by:_____ Approved by:_____

ABSTRACT

Many retail stores, as well as other organizations that employ a multitude of part-time employees, rely on developing schedules frequently, since the availabilities of the employees as well as the needs of the business change often. This process is often performed on a weekly basis, and is complex and time-consuming. The schedule must typically satisfy numerous requirements, including business needs, legal restrictions, and employee availability constraints. As a result, errors are common, and employee time that could have been spent on improving sales or operations is instead consumed by the scheduling task.

This project explores two solution methods for this problem. One method is the use of linear programming (LP) to develop an optimal schedule weekly. The other is the design and implementation of a scheduling system that uses a heuristic method. After developing both methods it was determined that while the LP approach may lead to optimal solutions, it was impractical due to high costs and complexity. The heuristic approach resulted in an automatic scheduling system that is easy to use, low cost, and flexible. The new scheduling system was evaluated and approved by future users.

ACKNOWLEDGMENTS

A special thank you to Michael Ochi for his time and assistance in the formulation and design of the approaches used to solve the problem addressed in this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Introduction

The subject of this project is the exploration and design of a method to reduce the time that is spent writing a weekly schedule in an organization such as a retail store. Typically, the previously mentioned organizations operate with a large number of part-time employees and therefore do not have a set schedule and have to deal with changing employee availabilities and business needs. A high level of complexity in scheduling results from attempting to address these needs. As a result, scheduling becomes a time-consuming process that has the potential for a multitude of errors. The fact that this schedule is usually written by one person, usually a manager who has several other concurrent responsibilities, adds to the likelihood of human error occurring. The idea originated from one of us working at a Staples store in San Luis Obispo and seeing (and being directly impacted by) the consequences of the previously mentioned errors and the time consumed in writing the schedule. We began this project with the goal to provide a solution that reduces the time spent writing a weekly schedule and eliminates errors due to availability conflicts. Here is a list of objectives that must be completed for this project:

- Investigate the current scheduling situation at the subject store
- Formulate an approach (or multiple approaches) to solving the issues found in the investigation
- Design a system that would efficiently and consistently satisfy the scheduling requirements and reduce or completely eliminate the issues found in the current scheduling method
- Create and test a prototype of the new scheduling system
- Compare newly designed approaches to the current scheduling approach
- Recommend the best solution to the problem

These objectives will be achieved by implementing the use of several engineering tools and skills to design a user-friendly system that will consistently create a feasible schedule. Tools that will be used include and are not limited to: database design and management, human factors, and linear programming (LP). There will be two methods used in an attempt to solve this problem. The first will be the use of LP to create an optimal weekly schedule using variables and sets of constraints. The second will be the design and use of an automated scheduling system that schedules employees based on availabilities, needed shifts, and labor constraints. Each alternative will be presented with its methodology, followed by a discussion of advantages and disadvantages, comparisons, and a recommendation of the best method, as well as suggested improvements.

Background

This case involves evaluating and attempting to improve the scheduling method at a Staples store in San Luis Obispo, California. Patricia Carr, as the operations manager of the store, is responsible for scheduling the entire staff's shifts for the week. When asked about how long it takes her to write a schedule for one week, Patricia Carr replied "usually it takes 2-3 hours, but at times can take significantly longer, sometimes even several days if the store becomes very busy and her attention is needed elsewhere". She said that there are also several challenges she has to consider while writing the week's schedule, many of which result in constantly having to revise the schedule. Figure 1 demonstrates the manual changes that are made to the schedule after it is written and posted. In this particular schedule, one employee's entire weekly schedule was changed completely after the schedule had already been posted due to miscommunication of time off requests. Other changes were the result of availability conflicts and time off requests.

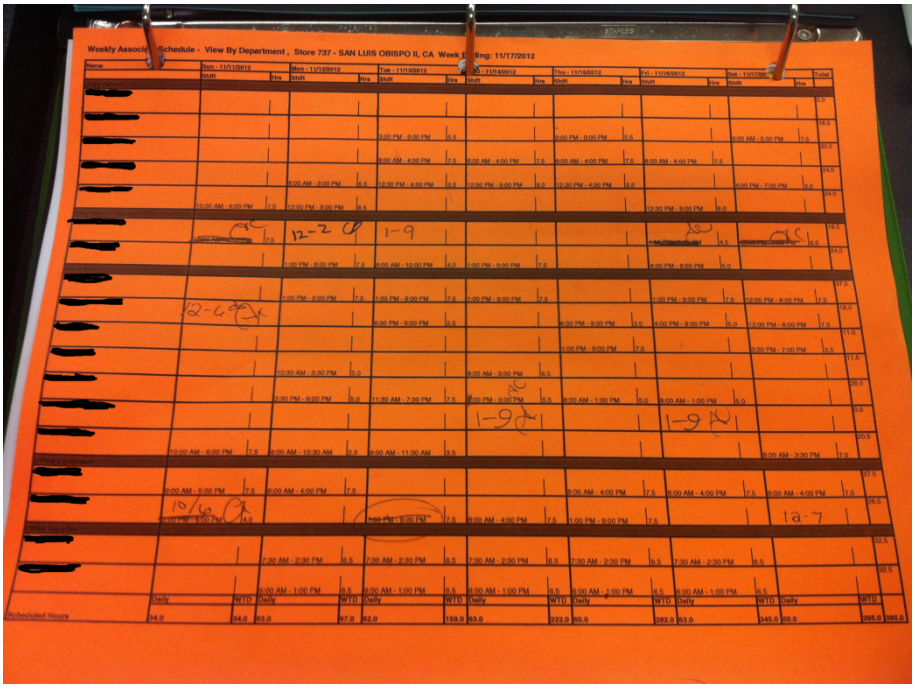


Figure 1: A weekly schedule for the Staples store

Department/Skill-Based Limitations

An important factor to consider is the issue that there are multiple departments to be covered in a given store. In the Staples store in question, there are four distinct departments: office supplies,

office equipment/tech, front end (cashiers), and copy center. There is also a price auditor, which is a full time employee, who will be considered as a separate department for scheduling purposes. While practically every employee in the store is trained as a cashier and can provide relief for breaks or in case of absenteeism, most associates are not cross-trained in other departments, therefore requiring associates with specific training in each department at all times. Failure to have a person with department-specific training present can result in vital operations functions not being performed to standard or neglected completely. Therefore, in the design of a scheduling system to fit the needs of an organization, it becomes very important to take into account the training and skill levels of individual associates, a task that the operations manager undertakes every time while writing a weekly schedule.

Availability-Based Limitations

Most of the employees at the Staples store in question are part-time employees with limited availabilities. This presents another scheduling challenge, since many times the availability of the employees prevents them from being able to work full shifts. This is a challenge that is universal to many businesses in the area, since much of the workforce for local retail consists of college students whose availabilities are limited by their school schedules. This sometimes results in the store having inadequate coverage at peak times and having one employee cover multiple departments, resulting in poorer customer satisfaction and less tasks being accomplished. Table 1 below shows the availabilities of the store's employees, with names excluded due to privacy reasons.

Table 1: Associate availabilities by department

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Copy Center							
Associate 1		8:00 - 4:00	8:00 - 4:00	8:00 - 4:00	8:00 - 4:00	8:00 - 4:00	
Associate 2	Any	N/A	3:00 - Close	N/A	3:00 - Close	Any	Any
Associate 3	Any	12 - Clsoe	N/A	12:00 - 5:30	N/A	1:30 - Close	Any
Front End							
Associate 4	2:00 - Close	Any	Any	Any	Any	Any	Any
Associate 5	12:00 - Close	10:30 - 3:30	N/A	N/A	N/A	N/A	N/A
Associate 6	Any	Any	N/A	Any	N/A	Any	Any
Associate 7	Any	Open - 11:00	Open - 12:00	Open - 11:00	Open - 12:00	Open - 11:00	Any
Office Equipment							
Associate 8	Any	8:00 - 5:30	8:00 - 5:30	8:00 - 5:30	8:00 - 5:30	Any	Any
Associate 9	Any	Any	Any	Any	Any	Any	Any
Associate 10	Any	1:00 - Close	N/A	1:00 - Close	N/A	4:00 - Close	Any
Associate 11	Any	Any	N/A	Any	N/A	Any	Any
Office Supplies							
Associate 12	Any	Any	Any	Any	Any	Any	Any
Associate 13	Any	N/A	12:00 - 3:00	N/A	12:00 - 3:00	Any	Any
Associate 14	Any	Any	N/A	Any	N/A	Any	Any
Price Checker							
Associate 15	Any	Any	Any	Any	Any	Any	Any

It is worth noting that at times, it becomes necessary to "borrow" an associate from another department in order to cover a time slot where there is simply no one available in that department. For example, on Tuesdays, there is only one person available to work a closing shift for the front end. If that person has already been scheduled for the maximum amount of hours, has already worked a shift during that day, or is simply unable to make it that day, there is no one else in that department available to cover the shift. Therefore, the only course of action is to schedule someone from another department from the closing cashiering shift. According to Patricia Carr, cases like this are not a common occurrence, but are far from being considered irregularities. Therefore, a consideration in this design is the potential to enable a user to handle

these occasional exceptional cases efficiently and easily, for example by means of a prompt in the event of such a condition arising.

Another way to handle potential availability conflicts is by scheduling split-shifts. These are not commonly utilized, since most employees prefer to work a straight shift, or only have the availability to work one small period of the day. However, there are some student associates who may have only one class in the middle of the day (for example, from 1 until 2 PM), and with the store's hours of 8 AM – 9 PM, can easily work half of their shift in the morning, and come back for a second half at night. At the Staples store in question, there are currently no employees who get scheduled for split-shifts at the time that the schedule is initially written, but this type if shift is not out of the ordinary among the changes that are made to the schedule manually after it has been published. Therefore, it is worth considering that scheduling employees for split-shifts from the start can help resolve some availability and preference conflicts.

Laws and Regulations

Another important portion of scheduling employees is the need to account for breaks and maximum shift lengths. In most states, there is a law in place that limits the amount of hours a person can work in a given day without having to be paid overtime. All companies make every effort to cut costs, and for most companies, overtime is a huge problem when it comes to incurring high labor costs. Therefore, every scheduler is forced to make an effort to schedule employees in such a way that overtime is not necessary. It has already been established that there will be several constraints involved in the design of the scheduling system, and compliance with labor laws and regulations will likely account for a large portion of these constraints.

Literature Review

This literature review details the research performed on the major topics pertinent to this project. These include: linear programming (LP), Microsoft Access, comparable systems, data requirements, and user interface design.

Linear Programming

According to a paper written by Fred Glover and Claude McMillan, "… most efforts to automate scheduling have relied chiefly on one or another variant of linear programming" (Glover, 34). A linear programming problem may be defined as a problem of obtaining an optimal solution for a linear function subject to linear constraints, which can be equalities of inequalities (Ferguson, 3). This sort of approach is useful for the application of designing an automatic scheduling system because it allows expressing many of the real-life constraints of the work environment as algebraic formulas. The problem of creating an optimal schedule will involve creating an objective function based on minimizing cost, maximizing coverage, or a combination of the two, with one of the factors possibly becoming a constraint as opposed to an objective function. A Prentice Hall module on linear programming discusses the application to employee scheduling, mentioning that the LP approach is useful when scheduling for tasks or jobs that require overlapping or interchangeable talents. "..the most common type of application involves the general problem of allocating limited resources among competing activities in a best possible way (i.e., optimal) way" (Hillier, 23). Since at least one of the applications that is within the scope of this project will require multiple talent sets that will often overlap (i.e. coverage for breaks and relief), this further proves that LP is a viable method for tackling scheduling and creating an automated scheduling system that will address the needs of the end-users.

Microsoft Access

Microsoft Access, a fully functional RDBMS (Relational Database Management System), was introduced in the early 1990s. Since its launch, Microsoft Access has been one of the most powerful and sought-after programs in the Microsoft Office suite of applications. Microsoft Access 2010 allows database developers to build dynamic and easily portable databases. Access comes with many easy-to-use features such as graphical forms, database templates, SQL query builders, as well as a subset of the Visual Basic language known as VBA for building data-driven applications (Vine, 136). VBA will allow the scheduling algorithms to be coded. A database is a collection of data stored in a file for future retrieval and analysis. This data can be combined, modified, updated, and stored in other reports or forms. Access has many features that have been previously listed, as well as application interfaces to connect to other programs such as cloud systems on the web or other intermediate programs. Access has been utilized in many different industries as well as colleges. With the growth of the computers, the growth of computer programs followed. Microsoft Access became popular because of its relatively low cost and user interface.

Comparable Systems

Previous projects have been done in an attempt to create similar automatic scheduling systems. One such project was done by Mark Peter Smith. His design is called the Lemming Scheduler, and it is a Java-based desktop application, which also allows for employees to check their schedules on the internet (Smith, 8). One problem noted by Smith in his report is the absence of automatic schedule generation, which was missing from several of the systems he had examined (Smith, 9). With the goal of creating a system that reduces the time to come up with a functional schedule as much as possible, a feature such as automatic schedule generation is essential.

Another important aspect of creating a scheduling system that would yield the most favorable results is data management. In order for scheduling to be more efficient, the method for entering and/or importing data must be efficient as well. According to Smith, there are systems that "… [take] data management out of the hands of the employer" (Smith, 12). This leads to the conclusion that ideally, employers need to spend as little time as possible on managing employee data such as availability and time off information. According to Smith, one way of taking data management out of the hands of the employer is using a web-based interface; however, the tradeoff to this is a system that is less powerful and offers a less robust feature set.

Required Data

As previously mentioned, there are multiple employee availabilities and constraints that must be taken into account. For this application, this data includes employee information, such as name and availability, defined departments and shifts for each department, and labor constraints. The data must be entered in such a way that a programmed scheduling algorithm can efficiently convert it into a user-friendly format. However, the user interface used to enter the data must be simple to operate, as discussed in the next section.

User Interface

Computers profoundly impact all aspects of life, whether at work or at home. They have revolutionized the way people perform office tasks such as writing, communicating with coworkers, analyzing data, keeping databases and searching for documents (Wickens, Lee & Liu, 136). While computers are being used in almost every aspect of life, the user interface of computers should be becoming more user-friendly so that even the most novice user will be able to navigate through the system. Computing systems, however, are no longer the province of the specialist user. In August 2000, 51% of households in the United States had access to one or more home computer (U.S. Census Bureau, 2001). "UI development should be user- centered.

This means that an understanding of the users, the task that they wish to carry out, and the environments in which they will be working must central to the development of the process" (Stone, 29). Therefore, it is necessary to design and develop a system or program that is both user friendly and has the ability to support the task that the user intends to perform.

A good user interface must follow a set of General Interface Design Principles. See Appendix A for a full description of these principles. The principles can also be used as the basis for a heuristic evaluation of prototypes. The terms "good" and "poor" are often used to describe the quality of an interface. However, those terms are relative to the skill level of the user. A more appropriate term is usability. Usability is defined in Part 11 of the ISO 9241 standard (BSI, 1998) as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use." Ensuring that a system has appropriate usability will avoid the user's frustration and dissatisfaction. There are many scenarios and situations in which the user becomes frustrated with the system. As a result of this frustration, the user becomes less motivated, and the level of productivity decreases substantially. In a company environment, this problem may become costly, and has the potential to affect the morale of other workers. Since the success of a computer program is directly related to the usability of the user interface, it is important that one takes the correct steps in creating a well-designed interface.

Design

Scheduling is a task required at all levels of industries, from employee scheduling to equipment and material scheduling. This task may be time consuming as well as inefficient, keeping managers from other tasks in which their involvement is more necessary. This project focuses on developing a solution that will improve the current process. A successful project is defined by: incurring little to no cost to implement, showing a decrease in process time, and will have high stakeholder satisfaction, the primary stakeholders being the scheduler and employees. Specifically, this project will be evaluated based on processing time, cost, flexibility, ease of implementation, required knowledge on the user's part, and the amount of scheduling errors. The evaluation criteria are described in the Methodology section of this report.

Definition of Requirements

Before design could begin, the current scheduling situation has to be explicitly defined. As previously mentioned, the manager in charge of scheduling at the Staples store said that scheduling took her at least 2-3 hours a week. Currently, the employee availabilities are stored on paper, making them easy to lose track of, and forcing the scheduler to constantly refer to the paper while making the schedule. Because of this, the setup time is often negligible. Below is a list of some of the factors that the scheduler currently has to take into account weekly:

- Employee availabilities
- Labor law constraints:
    - In California, an employee cannot work more than 40 hours a week without being paid overtime
    - For this scenario, it was assumed that an employee could not be scheduled for more than five days per week
    - In California, an employee cannot work more than 8 hours per week without being paid overtime
- Department needs

For this case, the pertinent details are as follows:

- 15 employees; 10 part-time and 5 full-time
- Managers are scheduled separately, and therefore not considered
- There are five departments

    - Copy Center
    - Front End (cashiers)
    - EasyTech (or Office Equipment)
    - Office Supplies
    - Price Auditor (one employee; considered as a separate department for scheduling purposes)
- Hours of operation:
    - 8 AM – 9 PM Monday – Friday
    - 9 AM – 7 PM Saturday
    - 10 AM – 6 PM Sunday
- Must always have at least one cashier, one Copy Center employee, and one tech present during store hours

The following two sections discuss in detail the two methods that were used in an attempt to create a solution for this scheduling problem: a linear programming method and a heuristic method.

Linear Programing

Linear Programing (LP) has been successfully implemented in many scheduling applications. LP works by minimizing, maximizing, or optimizing an objective function given a set of constraints. Compared to several scheduling methods, LP has two main advantages. The first advantage is that given the correct formulation, LP always produces an optimal solution. Another advantage is that computer-based programs that solve LP problems have already been developed and exist on the market. For example, Microsoft Excel has its own solver imbedded in its programing. However, it is worth considering that for a complex scheduling case such as this one, the LP formulations can get complex and may require industrial-level solvers, which are complicated and can be very expensive. See the Results and Discussion section for a discussion of the advantages and disadvantages of this method.

The first step in formulating this LP is to declare the variables. There are three indices in the variables this formulation: i representing employees, j representing the different hour blocks, and k representing each department. The index i ranges from 1 to 15 because in this case, 15

employees are considered. The index j ranges from 1 to 83, hour 1 being the first hour on Monday, and hour 83 being the last hour on Sunday. Below are a list of variables, the objective function, and the constraints needed for the LP formulation.

- Variables
  - Indices
    - i = employee (1-15)
    - j = hour (1-83 given previously mentioned hours of operation)
    - k = department (1-5)
  - X_{ijk} represents employee i working hour j in department k
  - X_{ijk} = 1 if the employee is scheduled; 0 otherwise (binary)
  - R_{jk} = people required to cover department k during hour j

- Objective Function (OF):

$$Min\ z = \sum_{i=1}^{15}\sum_{j=1}^{83}\sum_{k=1}^{5} x_{ijk}$$

- Constraints
  - Non-negativity: $x_{ijk} \geq 0$ for all i, j, and k
  - Coverage constraints:

$$\sum_{i=1}^{15} x_{ijk} \geq R_{jk} \text{ for all j and k}$$

  - Availability constraints

$$\sum_{k=1}^{5} x_{ijk} \leq 1 \text{ for j = 1 to 83 and k = 1 to 5}$$

  - Labor law constraints, which involve multiple combinations of several variables

Feasibility

With the given ranges of i, j, and k, it is possible to have up to 15 x 5 x 83 = 1,245 variables for a weekly schedule in this scenario. This also has the potential to result in tens of thousands of different constraints. An LP formulation this large will require an industrial size solver. In summary, while solving this linear programing formulation is not necessarily impossible, it is not a practical method to solve this scheduling problem.

Heuristic Method (Automatic Scheduling Program Design)

Another alternative considered is the design of an automated scheduling system that schedules employees based on *heuristics* such as availability, labor constraints, etc. In contrast to LP, this method does not guarantee the optimality of the produced schedule. However, it is possible to design a system that is flexible and would be easier to implement than an LP system.

Interface

The first step is to develop a method for the user to enter data efficiently and for the system to store data in the desired format. The main screen of the system contains a button that leads to the Departments form pictured in Figure 2. This form allows the scheduler to define the departments; create, view, and delete shifts for each department; and also to delete departments. It should be noted that it is impossible for the user to enter shifts with errors (i.e. duplicate shifts, or shifts where the start time is after the end time), with the system prompting the user if such an error occurs.

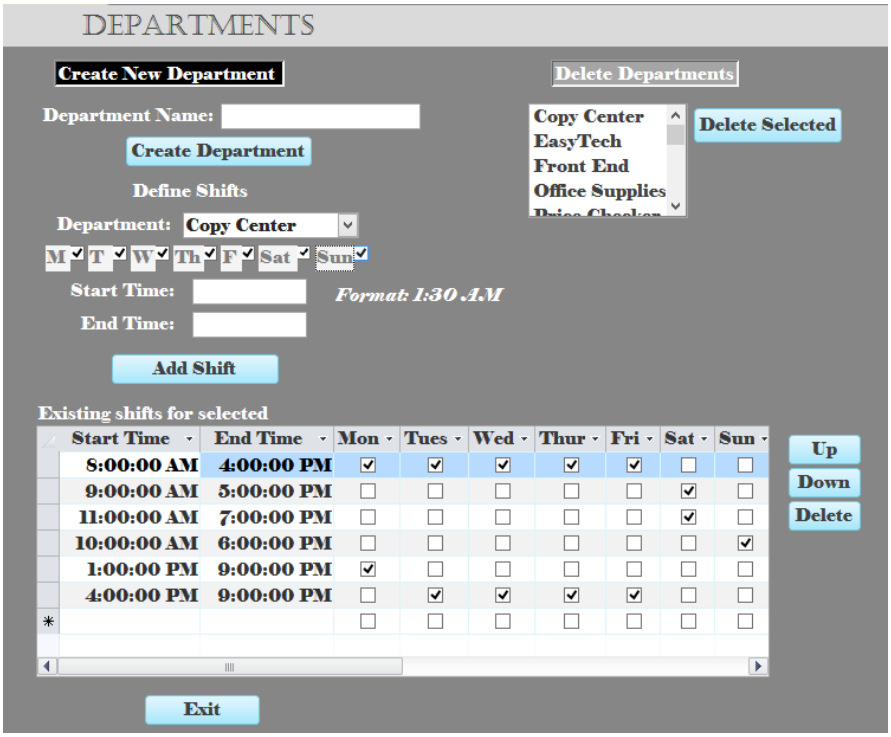


Figure 2: Screenshot of Departments form

A similar form exists for defining employees and their availabilities, pictured in Figure 3. This form contains similar functionality and similar error-proofing methods as the Departments form. The data is displayed in a format that is easy for the user to follow. With a click of a button, the user may see every shift for a certain department along with the availabilities of a selected employee.

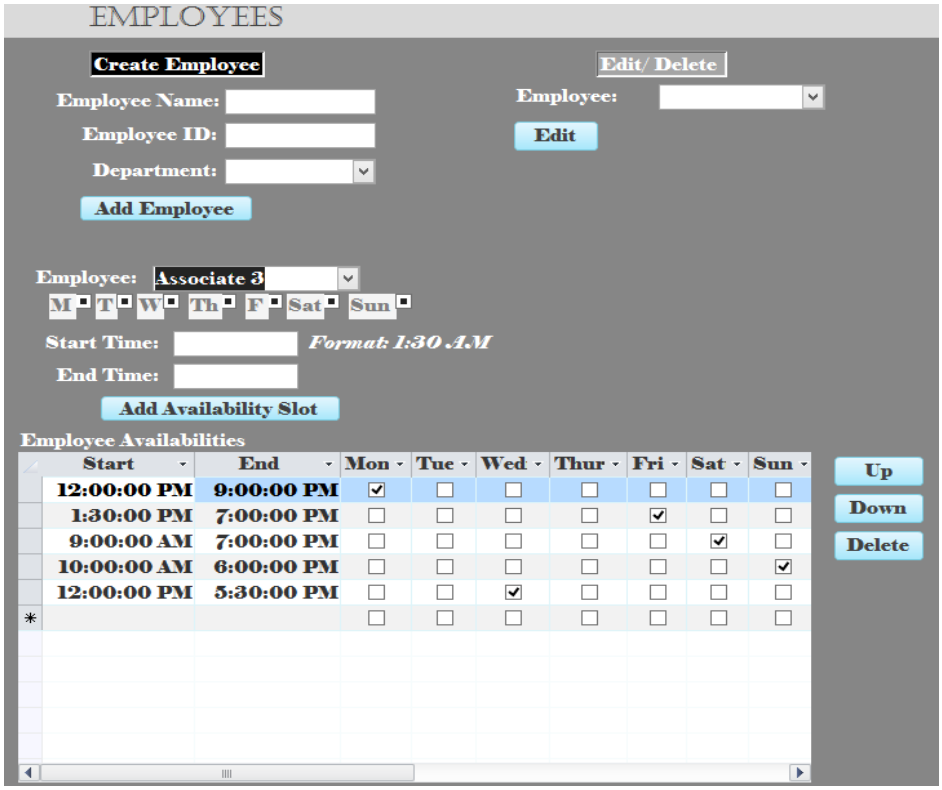


Figure 3: Screenshot of the Employees form

Another form developed is the Constraints form, pictured in Figure 4. This form allows the user to define and change the previously mentioned labor law and shift length constraints. This form is very simple, but is used for the storage of data that is used in the constraints formulated by the scheduling algorithm, which will be discussed next.

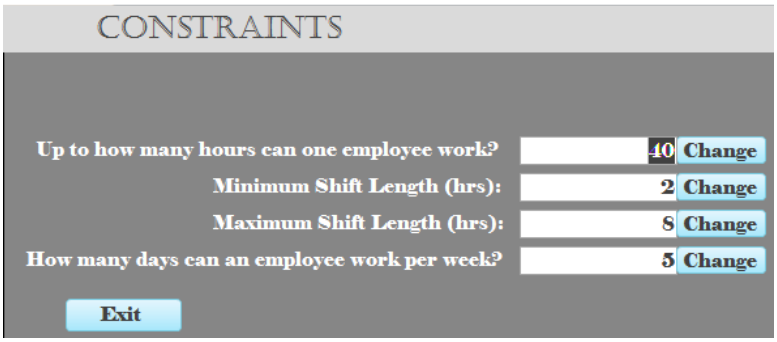


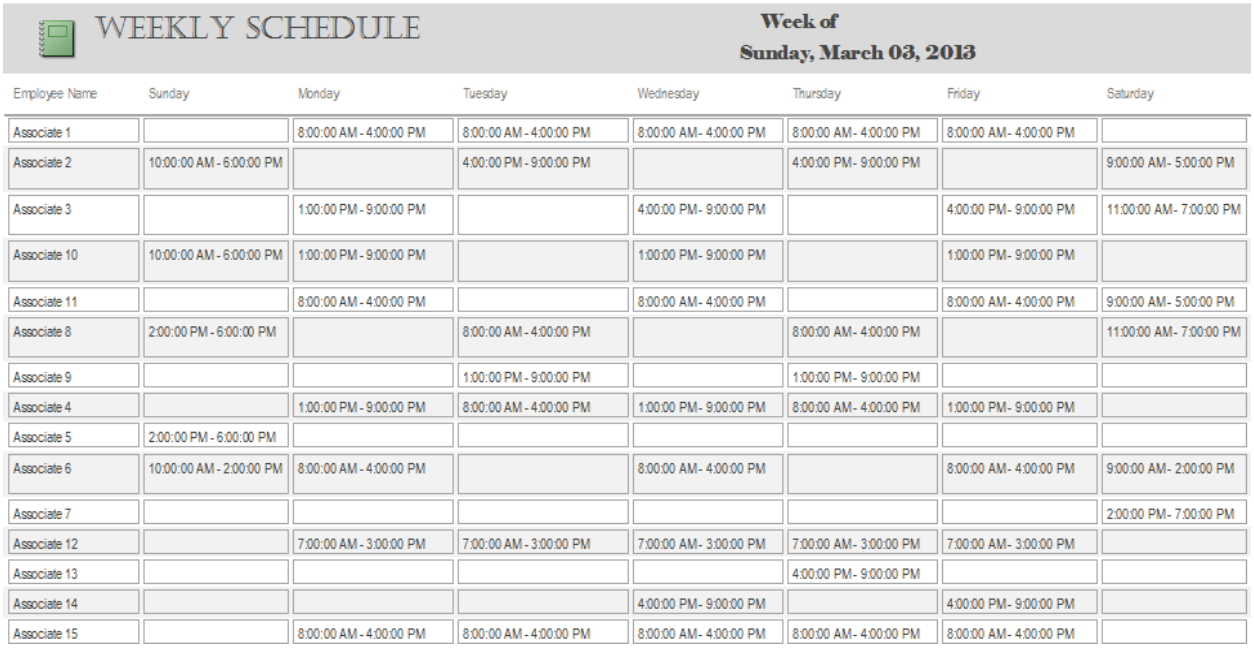
Figure 4: Screenshot of the Constraints form

Scheduling Algorithm

The last step in the process of creating this scheduling system is the creation of the scheduling algorithm that would use the previously mentioned heuristics in order to select an employee to assign to a shift. The algorithm used in this system was created through the use of database queries and VBA. The scheduling process begins when the "Create Schedule" button on the main form is pressed, as seen in Figure 5. This algorithm operates like a series of filters. First, it selects all the shifts that need to be covered that day (the algorithm begins scheduling on a Monday). Then, all the employees that can work on Monday are selected. The algorithm goes through a series of queries, selecting an employee that can work in the desired department, whose availabilities do not conflict with the shift, and who do not violate any other constraints. The system uses these queries as a funnel for the data, until one employee remains, at which point the system matches the selected employee to the current shift. Finally, the system uses queries to remove the employee from the pool of workers that can work that day, and proceeds to repeat the process for the next shift. This process is repeated for each day, until a full week's schedule is written. Figure 6 shows a

Figure 5: Screenshot of the main form

schedule output for the case of the Staples store considered in this project. Note that no employee is scheduled to work more than 40 hours per week or 5 shifts per week.



WEEKLY SCHEDULE

Week of
Sunday, March 03, 2013

Employee Name	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Associate 1		8:00:00 AM - 4:00:00 PM	8:00:00 AM - 4:00:00 PM	8:00:00 AM- 4:00:00 PM	8:00:00 AM- 4:00:00 PM	8:00:00 AM- 4:00:00 PM	
Associate 2	10:00:00 AM - 6:00:00 PM		4:00:00 PM - 9:00:00 PM		4:00:00 PM- 9:00:00 PM		9:00:00 AM- 5:00:00 PM
Associate 3		1:00:00 PM - 9:00:00 PM		4:00:00 PM- 9:00:00 PM		4:00:00 PM- 9:00:00 PM	11:00:00 AM- 7:00:00 PM
Associate 10	10:00:00 AM - 6:00:00 PM	1:00:00 PM - 9:00:00 PM		1:00:00 PM- 9:00:00 PM		1:00:00 PM- 9:00:00 PM	
Associate 11		8:00:00 AM - 4:00:00 PM		8:00:00 AM- 4:00:00 PM		8:00:00 AM- 4:00:00 PM	9:00:00 AM- 5:00:00 PM
Associate 8	2:00:00 PM - 6:00:00 PM		8:00:00 AM - 4:00:00 PM		8:00:00 AM- 4:00:00 PM		11:00:00 AM- 7:00:00 PM
Associate 9			1:00:00 PM - 9:00:00 PM		1:00:00 PM- 9:00:00 PM		
Associate 4		1:00:00 PM - 9:00:00 PM	8:00:00 AM - 4:00:00 PM	1:00:00 PM- 9:00:00 PM	8:00:00 AM- 4:00:00 PM	1:00:00 PM- 9:00:00 PM	
Associate 5	2:00:00 PM - 6:00:00 PM						
Associate 6	10:00:00 AM - 2:00:00 PM	8:00:00 AM - 4:00:00 PM		8:00:00 AM- 4:00:00 PM		8:00:00 AM- 4:00:00 PM	9:00:00 AM- 2:00:00 PM
Associate 7							2:00:00 PM- 7:00:00 PM
Associate 12		7:00:00 AM - 3:00:00 PM	7:00:00 AM - 3:00:00 PM	7:00:00 AM- 3:00:00 PM	7:00:00 AM- 3:00:00 PM	7:00:00 AM- 3:00:00 PM	
Associate 13					4:00:00 PM- 9:00:00 PM		
Associate 14				4:00:00 PM- 9:00:00 PM		4:00:00 PM- 9:00:00 PM	
Associate 15		8:00:00 AM - 4:00:00 PM	8:00:00 AM - 4:00:00 PM	8:00:00 AM- 4:00:00 PM	8:00:00 AM- 4:00:00 PM	8:00:00 AM- 4:00:00 PM	

Figure 6: Weekly employee schedule output

Methodology

Once the development phase of the system is complete, the next step is to properly test the interface and scheduling algorithm to make sure that they operate consistently. The system was populated with employee names and availability data found in Table 1. Next, the departments and shifts were defined in accordance to the store's real needs. Finally, the scheduling algorithm was allowed to run, and the schedule, whose output is shown in the previous section, was generated. This was repeated twenty times with the same data on a PC with an Intel Core i5 2.27 GHz processor and 8 GB of RAM.

Next, the system was tested to make sure that it would operate even if the data were changed. Some of the changes tested included the changing of certain shifts, adding and deleting employees, changing employee availability, and adjusting some labor law constraints. Most of these changes were tested separately, so the impact wasn't major. This testing was done in order to determine whether or not the scheduler works correctly when data is changed, and to see whether small, regular changes would have a significant effect on how long the scheduling algorithm takes to run. The results of testing, as well as a discussion of the strength and weaknesses of the scheduling system, are presented in the next section.

Results and Discussion

Automated Scheduling System Tests

Testing the system using the method defined in the Methodology section of this report yielded that the mean run time of the scheduling algorithm is 5.58 seconds with a standard deviation of 0.1760 seconds. For a graphical summary of the data used, see Figure 7 below. Making minor changes, such as adding or deleting a single employee, changing an employee's availability, or changing a small amount of shifts did not produce a significant difference in the time it took the algorithm to run. However, it should be noted that if a sizable change is made, such as the addition or removal or multiple departments, a large increase or decrease in the amount of shifts, and the presence of a much larger or smaller amount of employees, the run time is likely to change significantly due to the difference in the amount of data to be processed. With all the changes tested, the scheduling system consistently produced a schedule that did not violate any constraints and had no employees scheduled outside of their availabilities. The results were not perfect, as the system does not currently have a way of dealing with a situation where there are no employee matches for a certain shift, and therefore currently leaves said shift unscheduled. Alerting the user to which shifts do not have matches is the first step to improving this issue. This problem could be resolved in the future by implementing a method that allows the algorithm to split such a shift between two or more available employees. Aside from that issue, the designed system functions as predicted, generating a consistent weekly schedule in a consistently small amount of time.

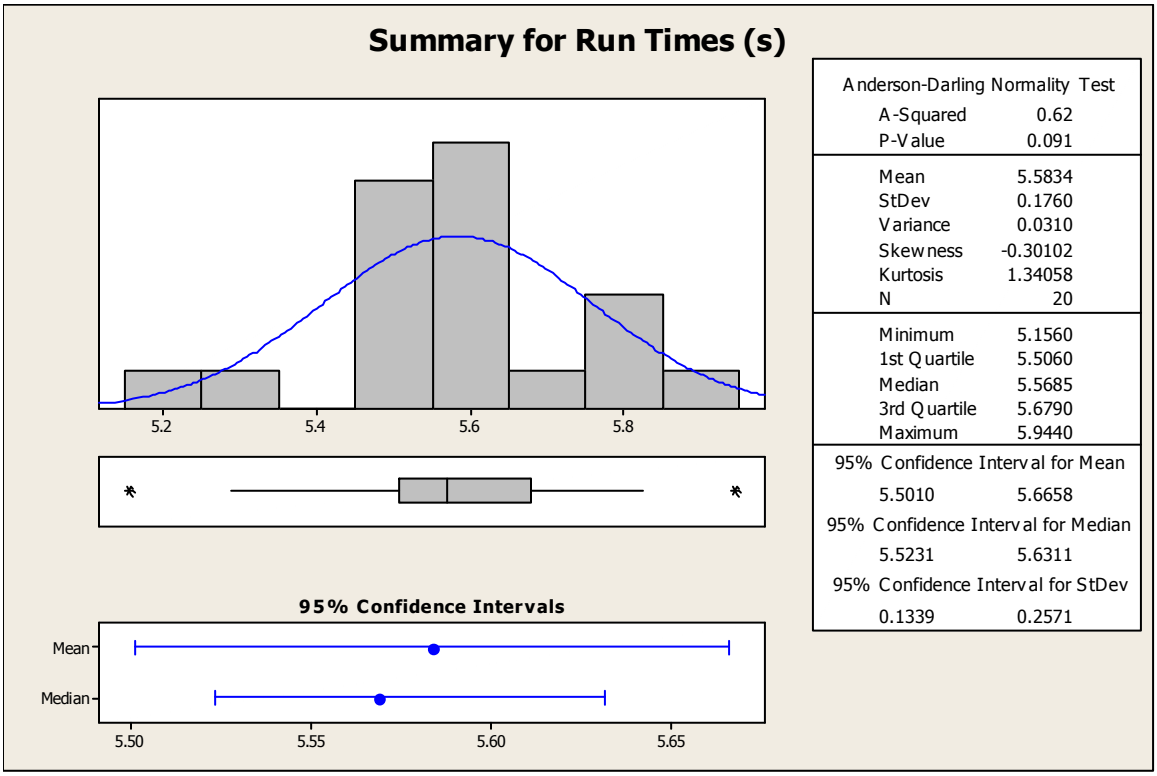


Figure 7: Graphical summary of scheduling system run-time analysis

Comparison of Scheduling Methods

Selecting the best scheduling method for this situation involves considering several distinct criteria, since each method has its own advantages and drawbacks. For this project these attributes consist of processing time, cost, flexibility, ease of implementation, required knowledge on the part of the user, and the efficiency of eliminating scheduling errors. A multi-attribute analysis was performed in order to determine which method is the most appropriate solution to this problem. The different criteria were weighted based on their importance to this problem, and the different methods were scored by assigning a value from 0 to 10 for each of the solution methods, 10 meaning that the method was superior or ideal in the given criterion, and 0 meaning that it was completely unsatisfactory in the given criterion. Table 2 below shows the results of the multi-attribute analysis.

Table 2: Muti-Attribute Analysis of Solution Methods

Method Alternatives				
Attribute	**Weight**	**Original**	**LP**	**Heuristic**
Processing time	20%	3	8	10
Cost	15%	8	3	8
Flexibility	10%	7	2	9
Ease of Implementation	15%	10	3	7
Required Knowledge	15%	9	4	10
Scheduling Errors	25%	2	10	9
Sum	100%	39	30	53
Score	10	5.85	5.8	8.9

The LP method received the lowest overall score, because while in theory it would eliminate all scheduling errors and consistently provide an optimal schedule, it is costly, difficult to implement, requires specialized knowledge on the scheduler's behalf, and would only be flexible with the creation of a comprehensive shell program that defines variables and constraints based on user input. Therefore, while much more accurate and possibly faster than the current scheduling method, the impracticality of LP renders it the least favored method to solving this scheduling problem.

In contrast, while the designed automatic scheduling system (heuristic approach) does not guarantee an optimal solution, it possesses the following advantages to the current scheduling method:

- Quick, consistent, and efficient operation (5.58 seconds on average to create a schedule)
- Requires no additional knowledge on the part of the user, aside from the system's learning curve
- A high degree of flexibility, since employee data, department data, and constraints can be easily changed

- Low cost, since the system currently functions entirely in Microsoft Access, which many organizations already have on their computers. Also saves significant labor cost, since it takes less than ten seconds to write a schedule, as compared to the 2-3 hours it takes using the current manual scheduling method
- Relatively easy to implement

All of these advantages give the heuristic method a higher score and make it the favorable method for solving this scheduling problem.

Conclusion

Two new methods for scheduling employees in a retail store were explored and compared to the current scheduling method in an attempt to make the process more efficient and to eliminate the common errors that occur. One method involves the use of linear programming (LP) to produce an optimal schedule, while the other involves the use of a designed scheduling system and a heuristic method for scheduling employees. Below is a summary of the key findings of this project:

- The current scheduling method is inefficient due to the amount of time consumed and the vast potential for errors
- Adding a degree of automation to the scheduling process would simplify the process and make it more efficient
- Using LP would, in theory, provide an optimal weekly schedule; however, it would be very costly, time-consuming, and difficult to implement
- The designed automated scheduling system which uses a heuristic method for scheduling does not guarantee an optimal solution, but will quickly and consistently generate a feasible schedule with no violations of constraints or employee availabilities

Based on the findings detailed in this report, and the analysis and comparison of the three different scheduling methods in this report, the implementation and use of the designed automatic scheduling system is the recommended approach to solving the scheduling problem presented in this project.

Potential Design Improvements

As previously mentioned, there are several potential improvements that can be made to the designed system that would make it even more ideal for solving this scheduling problem. These improvements include, but are not limited to:

- The ability of the system to handle shifts with no perfect matches, possibly by splitting the shift between multiple employees
- Accounting for time-off requests
- Integration with employer's current employee database and online services
- Accounting for budgeted payroll hours

Expansion of Scope and Other Potential Applications

This project began with the goal of improving the scheduling process for one retail store. However, during the design phase of this project, the scope was inadvertently expanded to designing a universal scheduling solution for similar department-based organizations that need to write weekly schedules for their employees. Given the success of this prototype and with the implementation of some of the previously mentioned improvements, it is possible to modify and apply this heuristic scheduling method to other scheduling situations such as creating a schedule of events, production scheduling, departure/arrival scheduling for flights, and many more. With adequate time and resources, a heuristic scheduling method like the one created in this project has the potential to change the way that scheduling works in multiple industry fields.

Works Cited

Carr, Patricia J., and Stacey L. Miller. "Scheduling Needs and Challenges." Personal interview. 13 Nov. 2012. (pg. 5)

Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) = Exigences Ergonomiques Pour Travail De Bureau Avec Terminaux À Écrans De Visualisation (TEV). Geneva, Switzerland: International Organization for Standardization, 1992. Print. (pg.12)

Ferguson, Thomas S. Linear Programming: A Concise Introduction. Rep. N.p., n.d. Web. 10 Oct. 2012. <http://www.math.ucla.edu/~tom/LP.pdf>. (pg. 9)

Glover, Fred, and Claude McMillan. The General Employee Scheduling Problem: An Integration of MS and AI. Tech. N.p.: Pergamon Journals, 1986. University of Colorado, Boulder. Web. 1 Nov. 2012. <http://leeds-faculty.colorado.edu >. (pg. 9)

Hillier, Frederick S., and Gerald J. Lieberman. Introduction to Operations Research. San Francisco: Holden-Day, 1980. Print. (pg. 9)

Neilson, J. Enhancing the explanatory power of visibility heuristics. Chi ’94 Proceedings New York: Association for computing Machinery. (pg. 11 & 28)

Smith, Mark P. *Employee Time Scheduling*. *Cal Poly Digital Commons*. California Polytechnic State University, June 2010. Web. 27 Oct. 2012. (pg. 10)

Stone, Debbie, Caroline Jarrett, Mark Woodroffe, and Shailey Minocha. User Interface Design and Evaluation. San Francisco, CA: Elsevier, 2005. Print. (pg. 11)

Wickens, Christopher D., John D. Dee, Yili Liu, and Sallie E. Gordon Becker. An Introduction To Human Factors Engineering. 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2004. Print. (pg. 11)

United States. Congress. House Reports. Washington, D.C.: U.S. G.P.O., 2000. Print. (pg. 11)

Vine, Michael A. *Microsoft Access VBA Programming for the Absolute Beginner*. Boston, MA: Thomson Course Technology PTR, 2005. Print (pg. 10)

APPENDIX A: General Interface Design Principles

General Interface Design Principles
Match Between system and real world
Speak the user's language.
Use familiar conceptual models and/or metaphors.
Follow real-world conventions.
Map cues onto user goals.
Consistency and standards
Express the same thing the same way throughout the interface.
Use color coding uniformly.
Use a uniformed input syntax (e.g., require the same actions to perform the same functions.)
Functions should be logically grouped and consistent from screen to screen.
Conform to platform interface conventions.
Visibility of the system status
Keep user informed about what goes on.
Show the input has been received.
Provide timely feedback for all actions.
Indicate progress in task performance.
Use direct manipulation: visible objects, visual results
Use control and freedom
Forgiveness: Obvious way to undo, cancel and redo actions.
Clearly marked exits.
Allow user to initiate/control actions.
Avoid modes when possible.
Error prevention, recognition and recovery
Prevent errors occurring in the first place.
Help user recognize, diagnose, and recover from errors.
Use clear, explicit error messages.
Memory
Use see-and-point instead of remember and type.
Make the repertoire of available actions salient.
Provide lists of choices and picking from the list.
Direct manipulation: visible objects, visible choices.
Flexibility and efficiency of use
Provide shortcuts and accelerators.
User has options to speed up frequent actions.
Systems should be efficient to use (also, the ability to initiate, reorder, or cancel task).
Simplicity and aesthetic integrity
Things should look good with a simple graphic design.
Use simple and natural dialog; eliminate extraneous words or graphics.
All information should appear in a natural and logical order.

Source: Neilson, J. Enhancing the explanatory power of visibility heuristics. Chi ’94 Proceedings New York: Association for computing Machinery.