Carleton University

Creating Stable Matching with Shift Scheduling

COMP 4905

Carleton University

Dr. John Oommen

Fall 2013

Abstract

Creating effective schedules for a workplace with regards to the availability of resources and the jobs that need to be completed can be desirable for large and small corporations. This project considers the problem of having resources with incomplete lists of availability which need to be matched to the jobs. This project tests and evaluates the effectiveness of using the Gale-Shapley algorithm to acquire a stable matching with incomplete lists and indifference between the many-to-many relationship between resources and jobs. The algorithm is modified to fit the specifications of the jobs being done and the constraints placed on the resources in order to create the highest percentage of preferred stable matchings.

Table of Contents

Table of Figures

Table of Tables

Introduction

Motivation

Creating a productive and accommodating schedule for large companies that are offering shift work can become a challenging process when employees have different availabilities and skill levels. Employers could spend hours trying to create an effective schedule which could still end up with employees not getting what they want. In this case a system that is able to allocate employees to jobs in an efficient and effective way can be a very useful tool for the employer. This project explores the idea of comparing this scheduling problem to the Stable-Marriage problem, as well as tests and analyzes the use of a modified version of the famous Gale-Shapley algorithm to satisfy the constraints of employees and jobs. The algorithm used as a solution to this problem is able to assess the availability and preferred shift times for *x* number of employees and then create a schedule that allows as many employees as possible to work their preferred shifts. In addition, if there are multiple tasks for employees to work on, this modification to the Gale-Shapley algorithm allows the user to select a task that they wish to optimize, and the schedule will then be made to have as many employees as possible working in the task selected while still matching employees to their preferred shifts. The first objective of this project is to randomly assign *x* number of employees with a list of availability and preferred shifts, as well as a list of the tasks that the employee has the knowledge to complete. The second objective is to modify the Gale-Shapley algorithm to schedule all employees an appropriate number of shifts while allowing them to work as many of their preferred shifts as possible. The next objective is to adjust the tasks employees are working on in order to optimize a specific task, if necessary. The final objective of this project is to create a user interface that allows the user to view information about the schedules that are created.

Similar Work

The Stable Marriage problem has been the focus of much attention in literature in the past few decades as it has many real-world applications. One instance of this problem in the real world is the task of matching medical students to hospitals in order for the students to complete their residency. In this case, each student has an ordered list of preferred hospitals and each hospital has an ordered list of preferred students. This information is used to create stable matchings between students and hospitals. In the past, there was no coordinated system to match the students with hospitals, so the hospitals would extend their offers to students while giving very short response times for fear of losing the student to a more desirable hospital. Now, the National Resident Matching Program in the United States, the Canadian Resident Matching Service, and the Scottish Foundation Allocation Scheme all automate the Hospitals/Residents problem by using variations of the Gale-Shapely algorithm (Manlove, et al. 2007).

Background

For this project, employees need to be matched to shifts in order to create a complete schedule. Each shift has a minimum and maximum number of employees that can be working for it. In addition, each shift has multiple tasks that need to be performed by the employees and require that employees have the specific skills needed to complete the task. The user has the option to select specific tasks that he or she wishes to optimize. Allocating more employees to a specific task allows for minimizing the time it takes for completion of the task.

The employees have a list of available shifts that they are able to work with some of those shifts being marked as preferred shifts. If an employee marks a shift as preferred, it means they would rather work that shift than a different, non-preferred shift that is also in their availability. Employees also have a list of skills which correlate to which tasks they are able to work on. Each employee must work between a minimum and maximum number of shifts and must have more than the minimum number of shifts in their availability.

The formulation of the lists of availability and preferred shifts for employees results in this being a Stable Marriage problem with indifference and incomplete lists, also known as an SMTI problem. An incomplete list with regards to the stable matching problem means that not all of the possible matches appear on an element's preference list (Kiraly 2013). In this case, employees may not have all of the shifts for the week in their list of availability, which results in many employees having incomplete lists. The shifts, however, have complete lists of employees as they do not care which employees are working, as long as there are enough. This Scheduling problem consists of a set *S* of shifts and a set *E* of Employees. A matching of (s, e) where $s \in S$ and $e \in E$, is only an acceptable matching if *s* appears in the availability list of *e* and *e* is in the list of employees for *s.* The indifference component of the problem implies that this list of preferences may contain ties, where some elements are ranked at the same level as other elements (Kiraly 2013). In the case of the scheduling problem, the shifts rank all employees at the same level. The employees rank some shifts as preferred, so all preferred shifts are tied at the highest position, and all of the non-preferred shifts are tied at the lowest position.

Methodology

Methods used

To determine the reliability of a modified Gale-Shapely algorithm for the scheduling problem, the problem and solution were implemented using the java programming language for its simplicity and readability. The employees are randomly assigned lists of availability of different length as well as varying numbers of preferred shifts.

The objects are set up as follows:

Each shift contains its identification number, an indicator of whether or not the shift is free to have more employees, and a list of the tasks being performed during that shift.

Each task contains its identification number and name, the minimum and maximum number of employees able to work on that task per shift, and a list of the employees that will be working on the task.

Each employee contains an identification number, a list indicating the tasks they are able to perform, a list containing the times they are available to work, and a list containing the times that they are scheduling to work. A time object contains what shift it belongs to, what task the employee will be performing, and whether or not the time is preferred by the employee.

The model, containing all of the core information for this problem, contains two hashmaps containing the data. Two hashmaps are necessary as each task that is selected to optimize results in a different possible schedule, so each key in the hashmaps is the key for the employee and shift data for that specific schedule. The first hashmap uses the name of the task to be optimized as the key, and a list of the shifts as the values. This data structure is illustrated below.

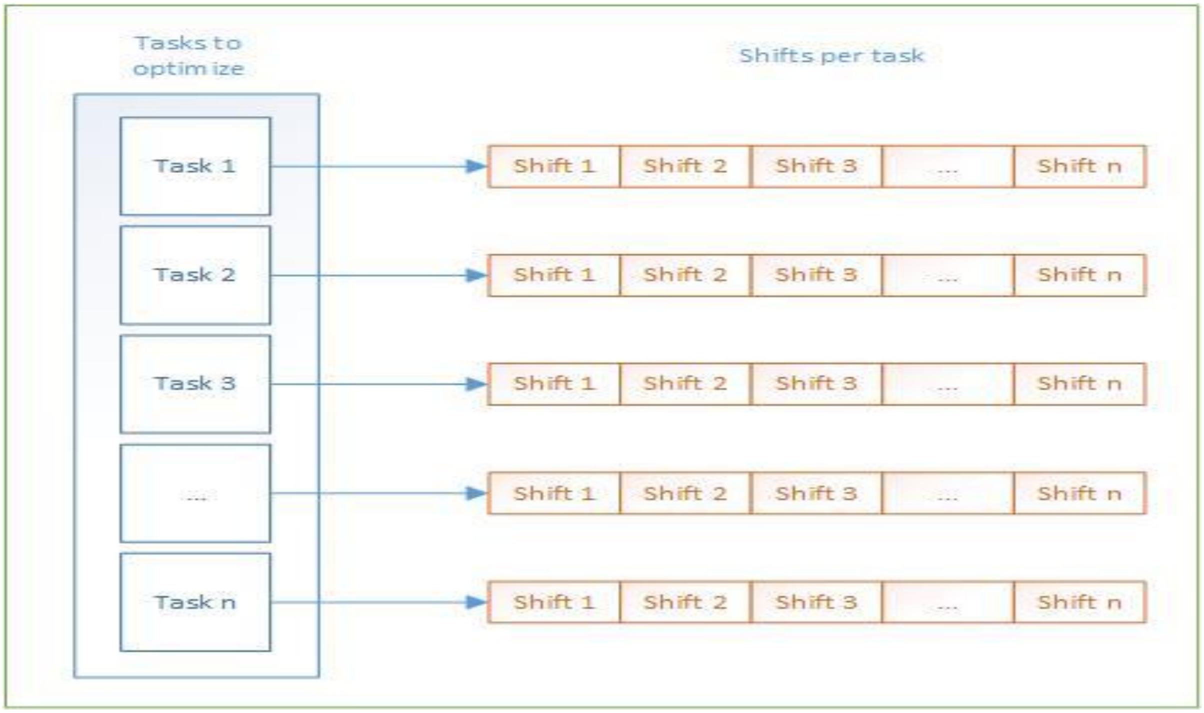


Figure 1 - Hashmap for Tasks and Shifts

The second hashmap also uses the name of the task to be optimized as the key, and a list of the employees available to work as the value. It is necessary to have a separate list of employees for each schedule, as the shifts and tasks they will be working on can vary. An illustration of the task and employee hashmap is shown below.

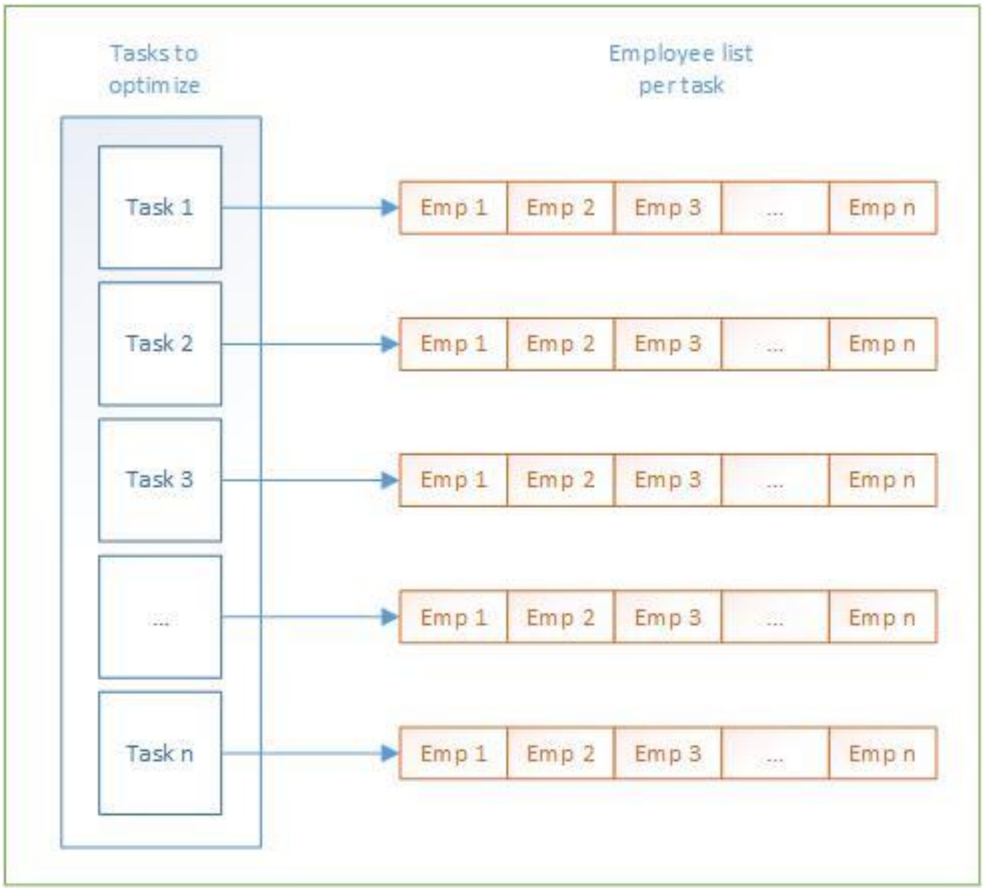


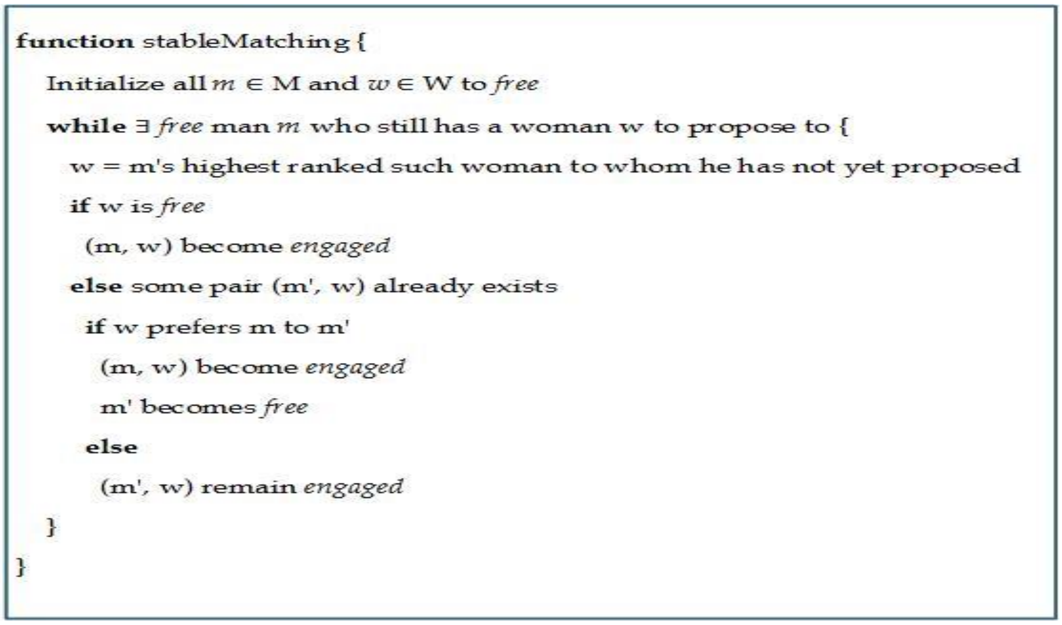
Figure 2 - Hash map of Tasks and Employees

The quality of this algorithm will be tested based on the percentage of preferred shifts that employees are scheduled to work after the algorithm has been performed. This percentage is to be determined using the number of preferred shifts that the employee provided and the number of preferred shifts that the employee received. In addition, when the user selects an option to optimize the completion of certain tasks, the algorithm is also tested by ensuring that as many employees as possible are scheduled to work on that task.

To view the results created by executing the algorithm, a calendar GUI was created that shows the total number of employees working on a shift throughout the week, the percentage of preferred shifts given to employees, and allows the user to view a detailed list for each shift showing the employees working for that shift and the task they will be working on during that shift.

Relevant Ideas

As discussed earlier, the Scheduling Problem presented in this report is being explored as a variation of the Stable Marriage problem with incomplete lists and indifference. The Stable Marriage problem consists of trying to find a stable matching between two sets of elements where both sets have a list of preferences. A matching between two sets *A* and *B* becomes stable when there is no element *a* ∈ *A* that prefers an element *b* ∈ *B* over the element it is already matched with, and *b* also prefers the element *a* over the element with witch it is already paired (Gale and Shapley 1962). The Gale-Shapley algorithm is able to find a solution to the Stable Marriage problem in linear time. Pseudo code for the Gale-Shapley Stable Marriage algorithm is shown below.



```
function stableMatching {
   Initialize all $m \in M$ and $w \in W$ to free
   while $\exists$ free man $m$ who still has a woman w to propose to {
     w = m's highest ranked such woman to whom he has not yet proposed
     if w is free
       (m, w) become engaged
     else some pair (m', w) already exists
       if w prefers m to m'
         (m, w) become engaged
         m' becomes free
       else
         (m', w) remain engaged
   }
}
```

Figure 3 - Gale-Shapley Algorithm

In this algorithm, each man proposes to his highest ranked woman to whom he has not yet proposed. If the woman is not engaged, she will accept the proposal of the man. Otherwise, if she is engaged, she will determine if she prefers the man proposing over the man she is already engaged to. It the woman prefers the proposing man, she accepts his proposal and they become engaged, while the other man is set to be no longer engaged. The algorithm executes until all men and all women are engaged and all matches are stable.

The similarities between the Stable Marriage problem and the Scheduling Problem at hand are the following:

1. There are two sets needed for a Stable Marriage problem and there are two sets involved in this Scheduling Problem, which are the shifts and the employees.
2. The list of preferences for an employee consists of the shifts listed in their availability and which of those shifts are ones that they would prefer.
3. The list of preferences for a task during a certain shift consists of the skills needed in order to complete the task. Although the task doesn't have an ordered list of preferred employees, the list contains employees that possess the skill needed to accomplish the task.

This scheduling problem being addressed in this report can be more closely related to the Hospitals/Residents problem, which is a variation of the Stable Marriage problem. The main difference between the simple Stable Marriage problem and the Hospitals/Residents problem is that the stable marriage problem is one-to-one while the hospital/residents algorithm is one-to-many. A modification is made to the original Gale-Shapley algorithm to allow for hospitals to accept more than one student as a match (Gale and Shapley 1962). While this Scheduling Problem is more closely related to the Hospitals/Residents Problem, the Scheduling Problem is many-to-many instead of many-to-one. This scheduling problem is many-to-many because the shifts must have multiple employees working for them and each employee must work multiple shifts in order to be employed.

Another relevant problem to the scheduling problem presented here is the constraint satisfaction problem. The Constraint Satisfaction Problem involves having objects whose state must satisfy a number of constraints. A CSP must contain a finite set of variables, a function which maps the variables to a specified domain, and a finite set of constraints on a subset of the variables. (Borrett and Tsang 2001). This relates to the scheduling problem as the tasks must have between j and k employees working on them, where $0 < j < k < n$, where n is the total number of employees. Additional constraints are the list of available shifts that employees are able to work, as well as the fact that employees must work a certain number of shifts per week. Additionally, there can be different variants of constraint satisfaction problems. These include dynamic constraint satisfaction problems and flexible constraint satisfaction problems. The scheduling problem being solved for this project is a variant of a flexible constraint satisfaction problem as shifts and tasks are able to have a range of number of employees working them and still be satisfied (Miguel and Shen 1999).

Furthermore, with the tasks involved and the option to optimize certain tasks, this scheduling problem also contains some components that are similar to the Job Shop Scheduling problem. The Job Shop Scheduling problem is an optimization problem where jobs are assigned to resources in order to have everything completed in the most time efficient way. There are many variations of the Job Shop Problem, these variations can involve having the resources be related or independent, having a sequence that the resources need to be used in, and jobs and resources may have their own constraints. This Scheduling Problem is related to the Job Shop Problem because of the allocation of resources to the tasks that need to be completed. In this case the employees are the resources and the tasks are the jobs that need to be completed. The tasks can all be performed simultaneously and do not rely on the completion of a different task in order to be completed itself. A variation to the job shop problem that is being applied in this case is that the time for completion of a task can be varied depending on how many employees are assigned to that task. So the more resources being used by a job, the quicker the job can be completed. As there are enough resources to be allocated to all jobs and all jobs are worked on simultaneously, Job Shop Scheduling algorithms are not explored further in this report.

Possible approaches

For this problem an approach that can be used is a modification of the Gale-Shapely algorithm. For this modification, the tasks would take on the role of the *men* in the algorithm and the employees would take the role of the *women.* So in this case, the tasks for each shift are sending requests to the employees. If all of the relevant constraints are satisfied, the shift and task is added to the employees list of working shifts. However, if the employee has already reached their maximum number matches, then a check is done to see if the current shift is preferred over any shifts in the employees list of working shifts. If there is a non-preferred shift in the employee’s list of working shifts, then the non-preferred shift is removed from the employee’s list of shifts and the current, preferred shift is added.

Additional modifications need to be made to the algorithm due to the relationship between the two sets of objects being a many-to-many relationship while the Gale-Shapely algorithm is made for a one-to-one or a one-to-many relationship. In order to account for these differences, the algorithm used for the scheduling problem must perform additional iterations through the list of shifts and employees.

Supplementary approaches may be chosen from the different techniques of Constraint Satisfaction Problem resolutions. The local search resolution is done by iteratively improving the assignments of the variables. On each iteration, a subset of variables are changed in order to increase the number of constraints satisfied by the assignment (Miguel and Shen 1999). Another resolution technique is that of constrain propagation, which contains methods that can be used to obtain a form of local consistency between variables (Miguel and Shen 1999).

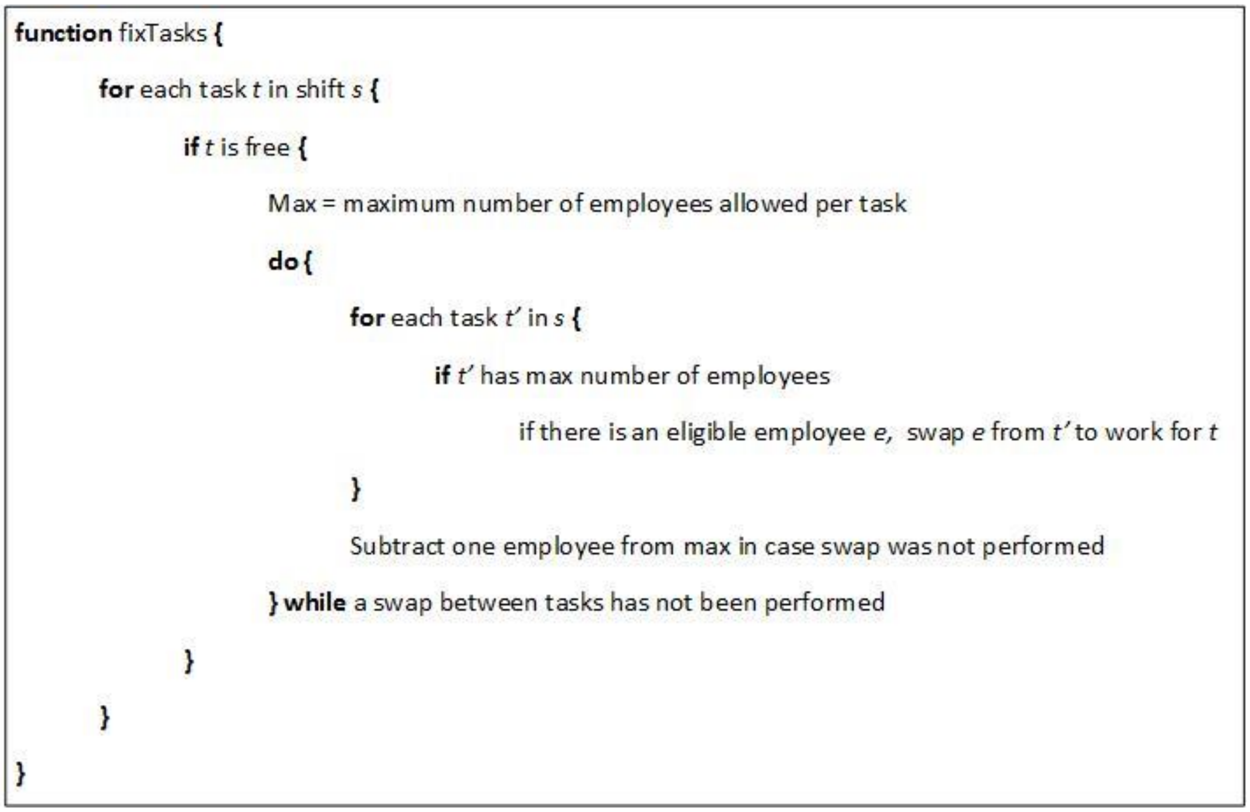
Approaches chosen

The approach chosen was to use a modified version of the Gale-Shapley algorithm, while also incorporating some techniques of constraint propagation. This modified algorithm was chosen to be shift-optimal as the shifts send out requests to the employees in order to ensure that all shifts have an acceptable number of employees working for them. The algorithm runs until all employees are scheduled for *x* shifts and until each task has at a minimum of *y* employees working at it for each shift. In addition, the tasks go through a search for an employee to add to its list of workers for a shift one by one by doing a check to see if the employee has the skill required to work on that task and if the employee has that shift in his or her list of availability. If both of these conditions are verified as true, the employee does a check to see if he or she is already working on a different task for that shift. If not, the employee adds this shift and task to their list of shifts and the task adds the employee to its list of working employees. The pseudo code for the modified algorithm is shown below.

```
function stableMatching {
    for each $s \in S$ {
        initialize all $t \in T$ and $e \in E$ as free
        while there exists a free task or a free employee {
            $e$ = the next employee for $t$ that $t$ has not yet sent a matching request to
            if $e$ is free and not already working for shift $s$,
                ($t$,$e$) are matched
          else e is not fr
              if $e$ has $t$ in its preferred list and there exists a ($t'$, $e$) matching where $t'$
              is not a preferred shift

                ($t$, $e$) become matched

                $t'$ removes $e$ from its list of employees

            else ($t'$, $e$) remain matched

        }

        if $s$ contains at least the minimum number of workers necessary

            swap employees between tasks so all tasks have at least the minimum number
            of workers necessary

    }
    if a task $t$ has been selected to be optimized

     adjust workers within tasks so $t$ has maximum number of workers possible

}
```

Figure 4 - Modified Gale-Shapley algorithm to solve Scheduling Problem

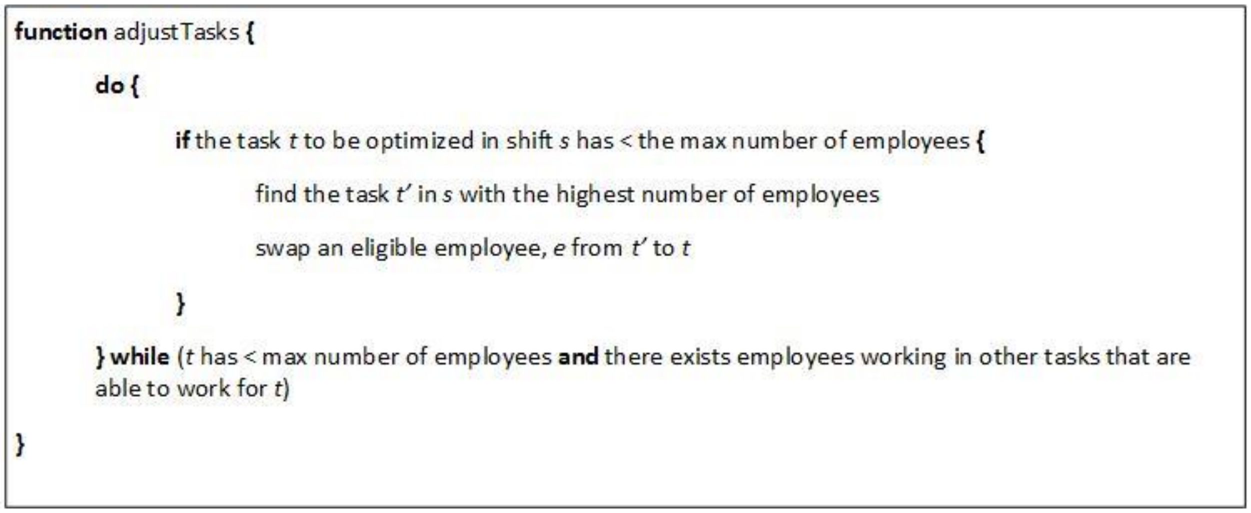
This modified algorithm is performed for each task that is selected to be optimized in order to ensure that each schedule has the maximum number of people working for the optimized task.



```
function fixTasks {
    for each task t in shift s {
        if t is free {
            Max = maximum number of employees allowed per task
            do {
                for each task t' in s {
                    if t' has max number of employees
                        if there is an eligible employee e, swap e from t' to work for t
                }
                Subtract one employee from max in case swap was not performed
            } while a swap between tasks has not been performed
        }
    }
}
```

Figure 5 - Algorithm to balance employees working for each task

Figure 5 shows the algorithm used to ensure that all tasks have at least the minimum number of employees when the shift containing the tasks has the combined minimum allowed amount. This algorithm prevents the case of having some tasks in a shift containing the maximum number of employees while other shifts have less than the minimum number of employees allowed. For this algorithm, if a task is free, it implies that the task has less than the minimum number of employees allowed.



```
function adjustTasks {
    do {
        if the task t to be optimized in shift s has < the max number of employees {
            find the task t' in s with the highest number of employees
            swap an eligible employee, e from t' to t
        }
    } while (t has < max number of employees and there exists employees working in other tasks that are
    able to work for t)
}
```

Figure 6 - Algorithm to adjust employees to optimize a task

Figure 6 describes the algorithm used to adjust the tasks in the case of having a task selected to optimize. As shown in Figure 4, this function runs after all shifts and tasks have been assigned an appropriate number of employees and all employees have been matched with the correct number of shifts. This function determines if the task to be optimized can accept additional employees, and if it can, then employees from different tasks are to be switched to the optimized task, assuming the employee has the correct skill needed to work on that task.

What was accomplished

The execution of the scheduling algorithm allows for all employees to be scheduled for the correct number of shifts and has an acceptable number of employees working for all shifts and tasks. The solution for this problem ensures that all employees are matched with all or as many of their preferred shifts as possible. The ability of the algorithm to adjust tasks from within the shift allows for easy modifications to each schedule to optimize a specific task. The table below shows the total number of shifts that were selected by employees as their preferred shifts, and the percentage of those shifts that were given to the employees.

Table 1 - Preferred Shifts for Employees

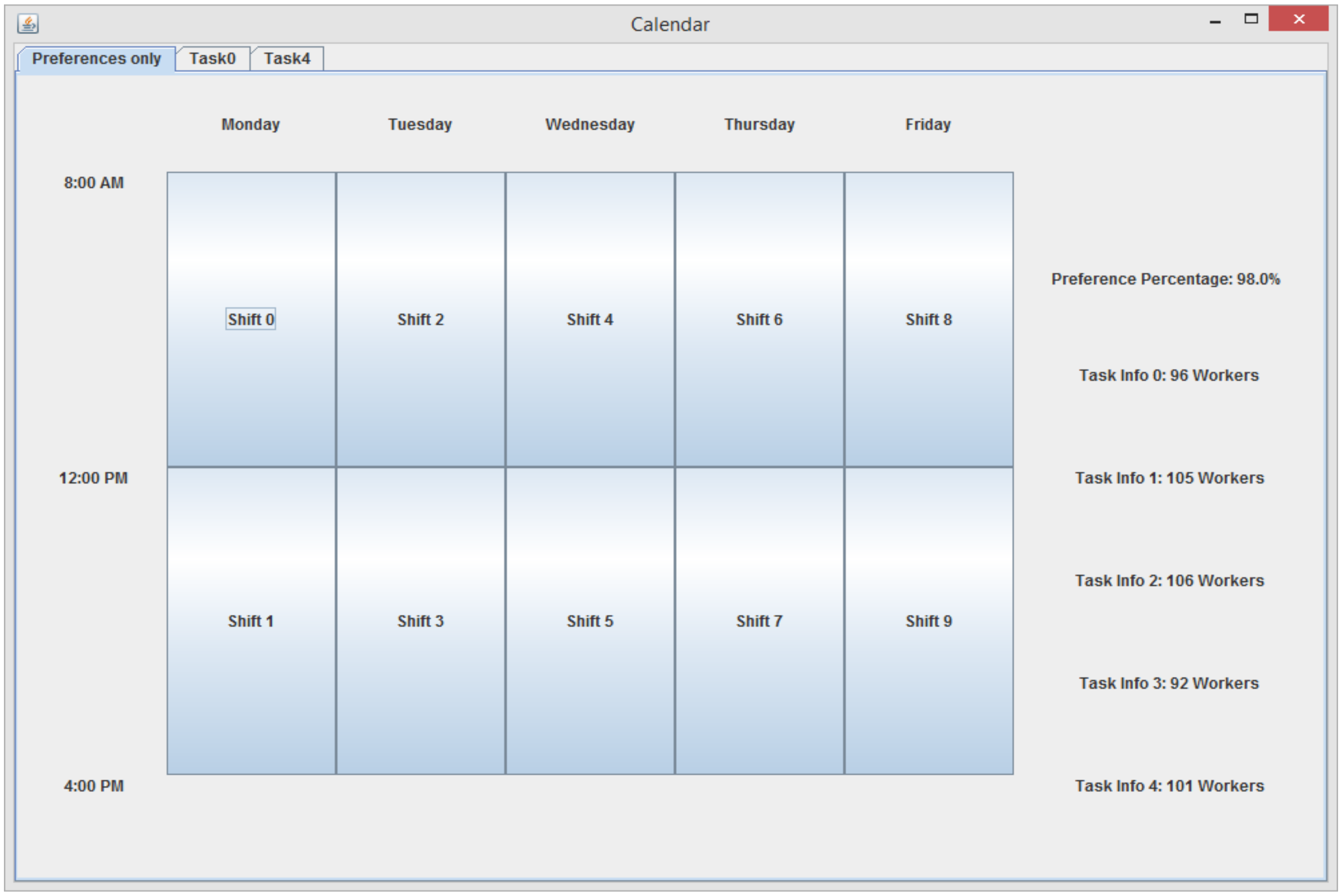
Number of Employees	Total number of shifts available	Number of Preferred shifts selected by employees	Percentage of preferred shifts being worked by employees
100	500	334	95%
100	500	321	94%
100	500	324	94%
100	500	324	91%
100	500	312	95%
100	500	329	91%
100	500	338	98%
100	500	344	97%
100	500	317	95%
100	500	334	97%
Average:		328	95%

Table 2 - Number of employees working each shift

Shift:	0	1	2	3	4	5	6	7	8	9	Total
Employees:	52	52	45	53	53	43	55	41	59	47	500

These tables show that the schedule was completed with a full list of stable matchings between shifts and employees. We know this because on average 95% of preferred shifts are given to employees, and all tasks and shifts have between the minimum and maximum number of employees working for them. As tasks have a completely indifferent list of employees to send requests to, this leaves no situation where an employee *e* would prefer a shift *s* over a current one and *s* would prefer *e* over one of its current employees that are working for it.

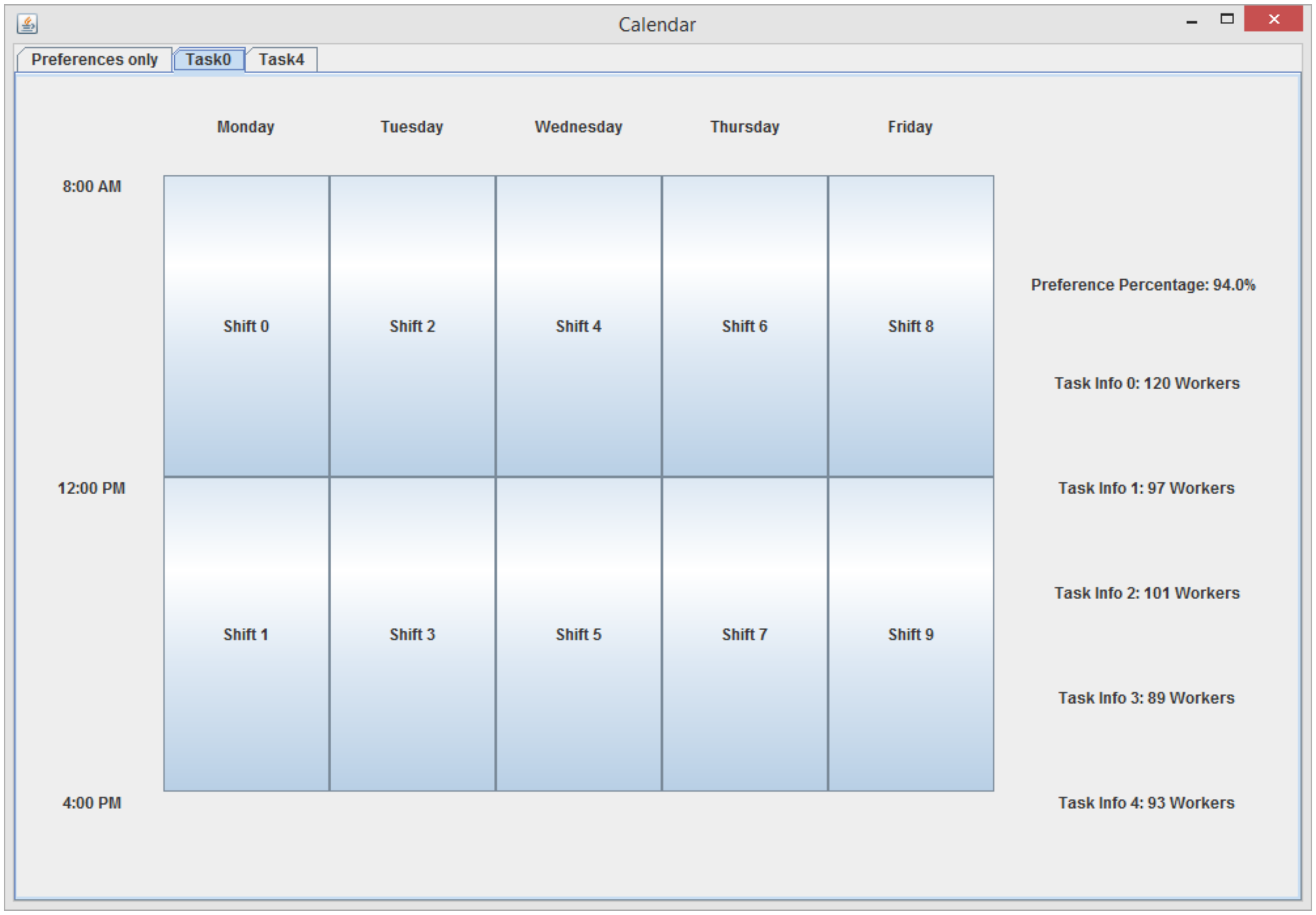
The user interface created with this algorithm allows for a user to view multiple schedules after one execution and allows for the comparing of the number of employees working on each task per week. Shown below are screenshots showing how the user can select the tasks to optimize and how the final results of the scheduling algorithm are displayed.



```
Preference Percentage: 98.0%
Task Info 0: 96 Workers
Task Info 1: 105 Workers
Task Info 2: 106 Workers
Task Info 3: 92 Workers
Task Info 4: 101 Workers
```

Figure 7 - Screenshot of Preferences-Only Calendar

Figure 7 shows a general view of the calendar when the preferences only option is selected. On the right it shows the number of workers working on each task throughout the week as well as the percentage of preferred shifts received by employees. As you can see from the tabs at the top of the window, multiple options were selected for the tasks to optimize.



Preference Percentage: 94.0%

Task Info 0: 120 Workers

Task Info 1: 97 Workers

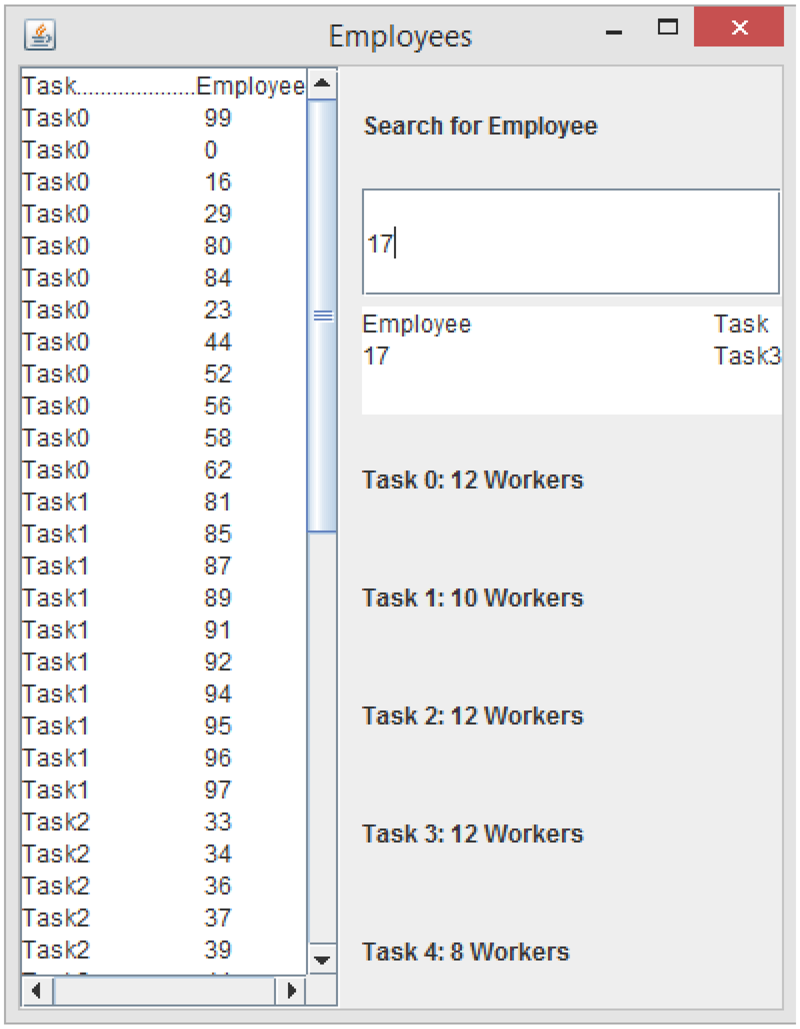
Task Info 2: 101 Workers

Task Info 3: 89 Workers

Task Info 4: 93 Workers

Figure 8 - Screenshot of Task 0 Calendar

Figure 8 shows the general view of a calendar created to optimize Task 0, as that is the tab selected at the top of the window. The layout is the same as that of Figure 5; the only difference is in the number of workers for each task. As Task 0 has been chosen to be optimized, it has the maximum number of employees working on it, in the preferences only calendar, the employees were spread out more evenly between all of the tasks.



Employees

Task..................	Employee
Task0	99
Task0	0
Task0	16
Task0	29
Task0	80
Task0	84
Task0	23
Task0	44
Task0	52
Task0	56
Task0	58
Task0	62
Task1	81
Task1	85
Task1	87
Task1	89
Task1	91
Task1	92
Task1	94
Task1	95
Task1	96
Task1	97
Task2	33
Task2	34
Task2	36
Task2	37
Task2	39

Search for Employee

17

Employee	Task
17	Task3

Task 0: 12 Workers

Task 1: 10 Workers

Task 2: 12 Workers

Task 3: 12 Workers

Task 4: 8 Workers

Figure 9 - Detailed View of Shift

Figure 9 shows the detailed view that the user sees when he or she clicks on one of the tasks in the general calendar. This pop-up window shows a list of the employees working and what tasks the employees will be working on. On the right it also shows the number of employees working on each task and gives the user the option to search for a specific employee in the list.

Conclusion

Future Work

There are multiple modifications that could be made to the scheduling algorithm used for this project. One modification that could be made to the algorithm would be to have the employees play the role of the men in the algorithm and the shift play the role of the women. In this case, the employees would be proposing to the shifts that they find most preferable to work at. The shift would accept any employee that requests to work with them as long as there are positions available working on the tasks that the employee is available to work on, or as long as the people already working could be rearranged to allow for a spot for the employee sending the request. This would result in the algorithm being employee optimal which would focus on getting all of the employees their preferred shifts. This would most likely increase the percentage of preferred shifts received by employees, but it may be more difficult to achieve a balanced number of employees working for each shift.

Additionally, constraints may be added employees which involve the employees having preferences of tasks as well as shifts. This would require a change to the algorithm as swapping the tasks between employees would no longer be as simple. With these constraints the percentage of preferred shifts given to employees would be lower, especially in cases of optimizing specific tasks, as employees who prefer other tasks may need to be switched to the optimized task.

Finally, supplementary methods may be added into the rearrangement of employees to tasks from within the shift for situations where there are not enough employees working that shift for the optimized task to have the maximum number of employees working on it. The changes made would be to find possible employees who would be able to switch from different shifts in order to fulfill the requirements needed by the optimized task.

Summary

This project has examined the ways to modify and implement an algorithm to schedule employees to shifts by creating stable matches between the two sets. Components of the Gale-Shapley algorithm were modified to account for the many-to-many relationship between the employees and shifts, as well as the incompleteness and indifference within the preference lists of the objects. By creating randomized data for employees, effectively modifying and adding to the Gale-Shapley algorithm for use with this Scheduling problem, and creating a graphical user interface for a user to view details of the schedule, I was able to meet all of the stated objectives of this project.

Works Cited

Gale, D., and L. S. Shapley. "College Admissions and the Stability of Marriage." *The American Mathematical Monthly*, 1962: 9-15.

Kiraly, Zoltan. "Linear Time Local Approximation Algorithm for Maximum Stable Marriage." *Algorithms*, 2013: 471-484.

Manlove, David F., Gregg O'Malley, Patrick Prosser, and Chris Unsworth. "A Constraint Programming Approach to the Hospitals/Residents Problem." *Lecture Notes in Computer Science 4510*, 2007: 155-170.