Report on Project 3: Developing Optimized Employee Schedules for Malwart

by

Md Hishamur Rahman

Student ID: 1400077

Umail: hisham.rahman@utah.edu

University of Utah

28th November, 2022

Contents

Executive Summary

Malwart spends on average 500 hours of overtime to meet their prescribed workloads, which incurs additional cost of about $9,000 a week. This report provides an optimized work schedule for all the employees at Malwart based on their requested minimum and maximum weekly working hours, employee availability, and the company’s staff requirements. To produce the required employee working schedules for an entire week, a mathematical model based on integer programming is developed that can be solved optimally using open-source software. Experiment based on employee data for a week revealed that the model developed in this report can fulfill the employee requirements of Malwart based on employees requested weekly working hours without any overtime, which saves the additional overtime cost of $9,000.

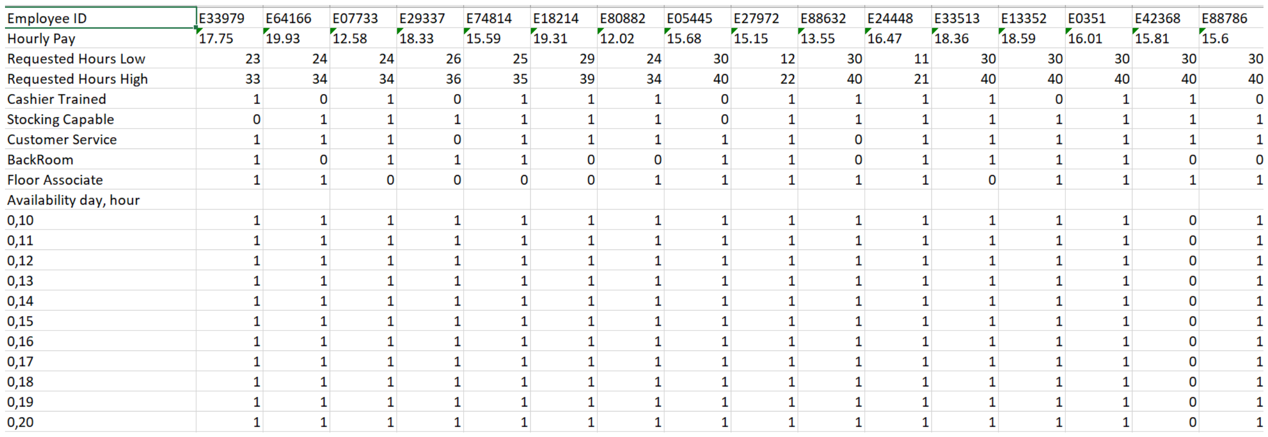
Introduction

To meet the prescribed workloads, Malwart spends on average 500 hours of overtime. At 1.5 times the employee cost and benefits, about $9,000 a week extra money is spent on overtime. As a result, Malwart requires an optimized working schedule for their employees in order to minimize their total employee payments. The optimized employee scheduling methodology is expected to minimize employee payments and overtime costs.

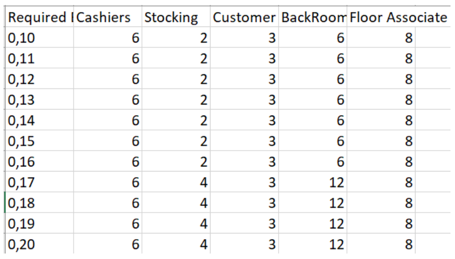
The abovementioned problem is addressed by modeling the employee working schedule mathematically using an integer programming approach. The model considered various constraints based on requested minimum and maximum weekly working hours, employee availability, and the company's staff requirements. The objective of the mathematical model is to minimize the total payments to the employees. The resulting mathematical model is solved optimally using Pulp, a Python package for solving linear programs. Experiments based on employee data for a week shows that the proposed mathematical model can yield cost savings of at least $9000 per week, diminishing the overtime cost altogether.

Data and Assumptions

The computer scientists from our company company developed a webportal where employees submit their available work times and managers submit the required number of employees (minimum) of each time for each hour of the week. The week-long employee data are not shown in the report for conciseness; however, couple of rows from the database are shown below as an example.



Employee ID	E33979	E64166	E07733	E29337	E74814	E18214	E80882	E05445	E27972	E88632	E24448	E33513	E13352	E0351	E42368	E88786
Hourly Pay	17.75	19.93	12.58	18.33	15.59	19.31	12.02	15.68	15.15	13.55	16.47	18.36	18.59	16.01	15.81	15.6
Requested Hours Low	23	24	24	26	25	29	24	30	12	30	11	30	30	30	30	30
Requested Hours High	33	34	34	36	35	39	34	40	22	40	21	40	40	40	40	40
Cashier Trained	1	0	1	0	1	1	1	0	1	1	1	1	0	1	1	0
Stocking Capable	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
Customer Service	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1
BackRoom	1	0	1	1	1	0	0	1	1	0	1	1	1	1	0	0
Floor Associate	1	1	0	0	0	0	1	1	1	1	1	0	1	1	1	1
Availability day, hour																
0,10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0,11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0,12	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0,13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0,14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0,15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0,16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0,17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0,18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0,19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0,20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1



Required I	Cashiers	Stocking	Customer	BackRoom	Floor Associate
0,10	6	2	3	6	8
0,11	6	2	3	6	8
0,12	6	2	3	6	8
0,13	6	2	3	6	8
0,14	6	2	3	6	8
0,15	6	2	3	6	8
0,16	6	2	3	6	8
0,17	6	4	3	12	8
0,18	6	4	3	12	8
0,19	6	4	3	12	8
0,20	6	4	3	12	8

The employee requested hours and hourly staff requirements are shown in Figure 1. From the employees' requested minimum and maximum working hours as shown in Figure 1 (left), it is clearly evident that no worker has requested overtime (requested hour high is maximum 40 hours). This indicates that if we constrain our model between the employee requested minimum and maximum working hours, there would be no overtime in the schedule.

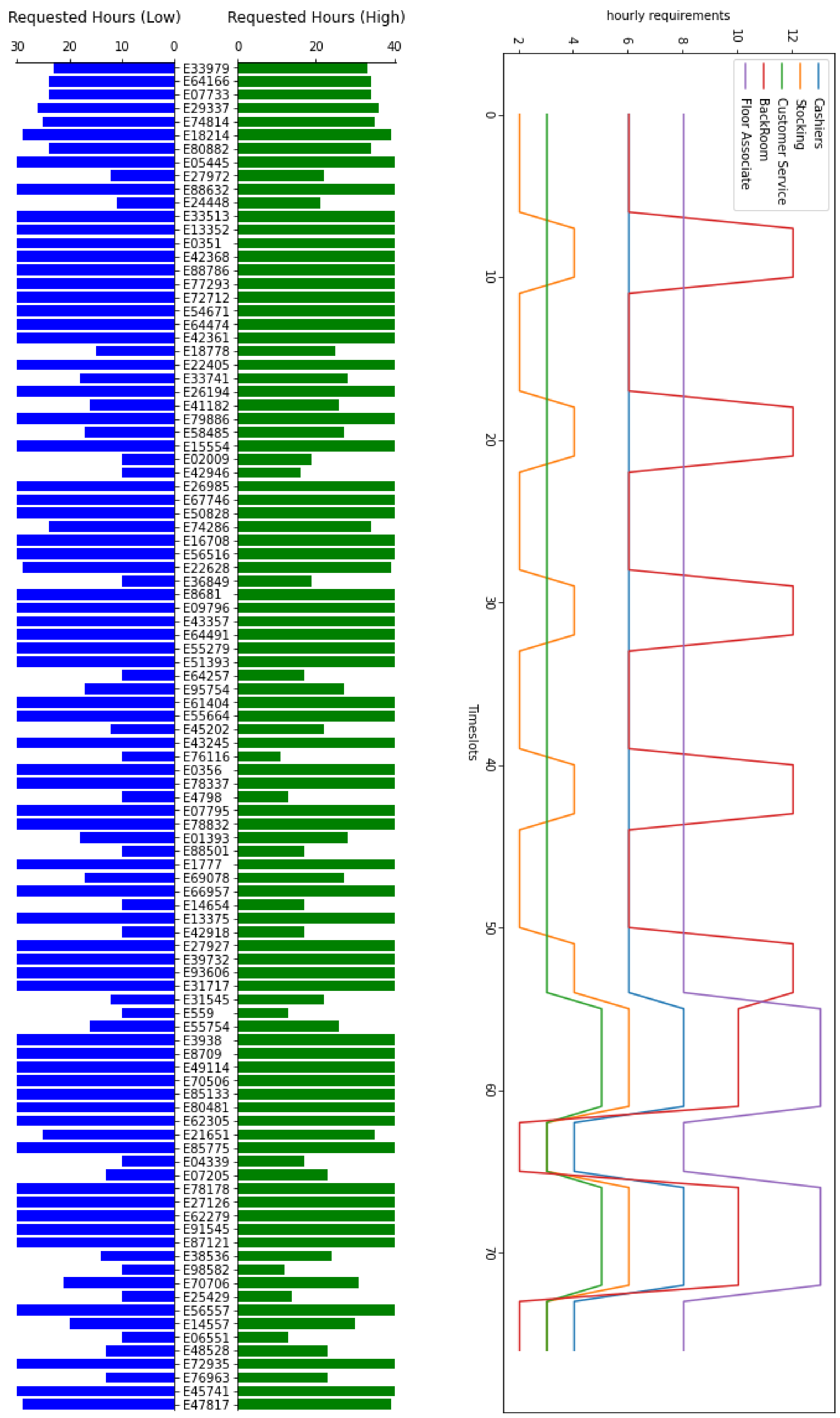


Figure 1. Requested employee hours (left) and company hourly staff requirements (right)

Mathematical Model

The mathematical model for this project is developed based on integer programming approach. The model includes a binary decision variable that needs to be solved, inequality constraints to meet the required staff limits and employee minimum and maximum working hours in the week, assigning employees based on availability and training, and an objective function that involves minimizing the total payments to the employees. They are described below with equation along with the Python code snippets.

Decision Variable(s)

The decision variable for this model are as follows:

Employee i working for work category j (cashier, stocking, etc.) in timeslot k → Sijk
The above binary decision variables are coded in Pulp as follows:

```
sched = pulp.LpVariable.dicts("schedule", ((i, j, k) for i in range(len(employees))
                                           for j in range(categories) for k in range(time)), lowBound=0, cat = 'Binary')
```

Constraints

For each timeslot and work category, the total number of employees working in that time slot must be greater than the required employees (R). This can be expressed as follows:

$$\sum_i Sijk \geq Rjk \qquad \forall j,k \qquad (1)$$

```
for k in range(time):
    for j in range(categories):
        my_lp_problem += pulp.lpSum([sched[i, j, k] for i in range(len(employees))]) >= requirement[j][k]
```

For each employee i, the total working hours must be within the minimum requested working hours (L) and maximum requested working hours (H), which can be expressed as follows:

$$Li \leq \sum_{jk} Sijk \leq Hi \qquad \forall i \qquad (2)$$

```
for i in range(len(employees)):
    my_lp_problem += pulp.lpSum([sched[i, j, k] for j in range(categories) for k in range(time)]) <= employees[i].highhours

for i in range(len(employees)):
    my_lp_problem += pulp.lpSum([sched[i, j, k] for j in range(categories) for k in range(time)]) >= employees[i].lowhours
```

Each employee i has to be scheduled in every timeslot based on their availability (A) and training (T), where A and T are binary variables. An employee can be assigned to a timeslot k only when both A and T is equal to 1, i.e., employee available for that timeslot k and trained for category j. to This can be expressed as follows:

$$Sijk \leq Aik \times Tij \qquad \forall i,j,k \qquad (3)$$

```
for k in range(time):
    for j in range(categories):
        for i in range(len(employees)):
            my_lp_problem += sched[i, j, k] <= employees[i].availability[k]*training[i][j]
```

Each employee i can work in only one work category for a timeslot k, which can be expressed as follows:

$$\sum_j Sijk \leq 1 \qquad \forall i,k \qquad (4)$$

Objective Function

The objective of the mathematical model is to minimize the total payment to the employees based on their individual payments (P), which can be expressed as follows:

$$\min \sum_{ijk} Sijk \times Pi \qquad (5)$$

```
my_lp_problem = pulp.LpProblem("My LP Problem", pulp.LpMinimize)
```

```
my_lp_problem += pulp.lpSum([employees[i].pay*sched[i, j, k] for i in range(len(employees))
                                                              for j in range(categories) for k in range(time)])
```

The Python code for the mathematical model is provided in the Appendix.

Analysis

The developed mathematical model is optimized and solved using Pulp package in Python. For conciseness, example of solution format generated by our code is shown below:

Employee costs	38592.18							
Timeslot	E33979	E64166	E07733	E29337	E74814	E18214	E80882	E05445
0,10	0	1	0	1	0	1	0	0
0,11	1	0	1	0	1	0	1	0
0,12	1	1	1	1	1	0	0	1
0,13	1	0	1	1	0	0	1	0
0,14	0	0	1	0	1	1	0	1
0,15	0	0	0	0	0	0	1	1
0,16	0	1	0	0	1	0	0	0
0,17	1	1	1	0	1	1	0	0
0,18	0	0	1	1	0	0	1	1
0,19	1	1	1	0	0	0	0	0

The comparison of model results against the company requirements are shown in Figure 2. The top figure shows the assigned hours vs the requested hours for each employee. It is clearly evident that our model assigned each of employees within their maximum requested time, which also shows that there is no overtime from our model. The bottom figure shows the comparison between total employees assigned in each timeslot vs the total requirement. This further illustrates that the model has always fulfilled the employee requirement of each timeslot.

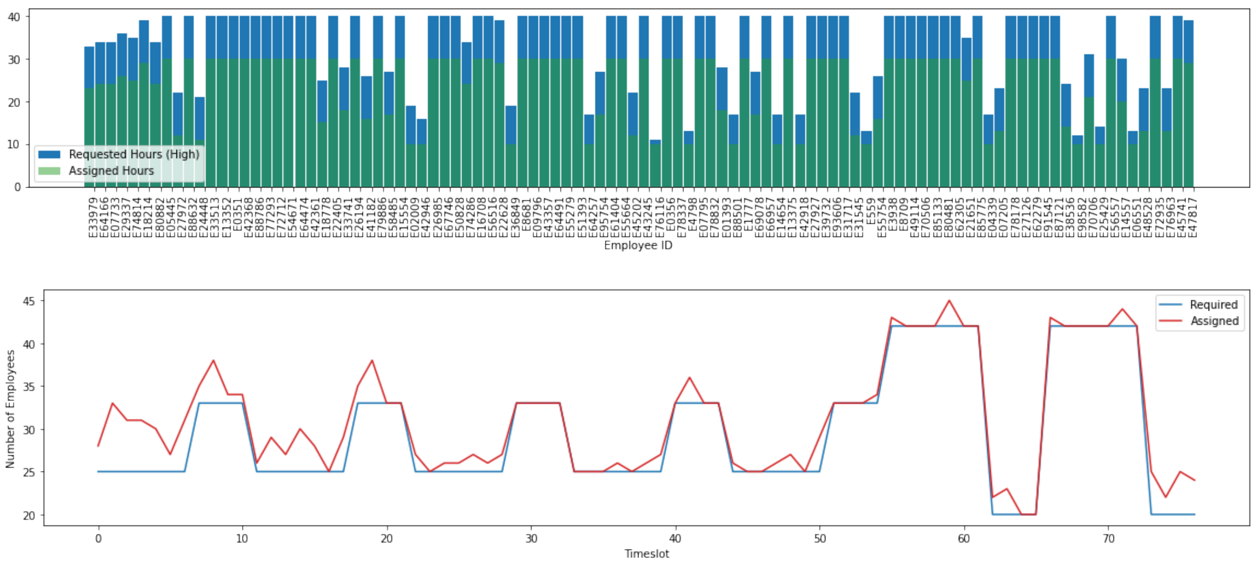


Figure 2. Model assignment vs company requirements

Conclusion

This report provided a mathematical model for optimizing the employee working schedule at Malwart. The mathematical model used a integer programming approach that was coded and solved using the Pulp package in Python. The analysis based on employee data for a week showed that the mathematical model is capable of assigning employees to the timeslots based on company requirements without any overtime, which can create cost savings of at least $9000 per week.

In this report, we utilized only one week of data to test our mathematical model. More analysis can be done on a dataset over couple of weeks to assess the cost savings trend. A sensitivity analysis can also be provided based on the model if requested. Additionally, the model can be further improved by imposing additional constraints to make employees scheduled in blocks.

Appendix

```
import openpyxl
import pulp
import openpyxl


class Employee (object):
  def __init__(self,eid, pay, lowhours, highhours, cashier, stocking, 

customerservice, backroom, floorassociate, availability):
                self.eid = eid
                self.pay = pay
                self.lowhours = lowhours
                self.highhours = highhours
                self.cashier = cashier
                self.stocking = stocking
                self.customerservice = customerservice
                self.backroom = backroom
                self.floorassociate = floorassociate
                self.availability = availability


def readexcelfile(path, employees, cashier, stocking, customerservice, 
backroom, floorassociate):

    wb = openpyxl.load_workbook(filename = path)
    sheetnames = wb.sheetnames
    wsheet =wb[sheetnames[0]]
    print (wsheet.max_column, wsheet.max_row)

    for j in range (1,wsheet.max_column):
        eid = wsheet[1][j].value
        pay = float(wsheet[2][j].value)
        #print (pay)
        lowhours = int(wsheet[3][j].value)
        highhours = int(wsheet[4][j].value)
        cashier = int(wsheet[5][j].value)
        stocking = int(wsheet[6][j].value)
        customerservice = int(wsheet[7][j].value)
        backroom = int(wsheet[8][j].value)
        floorassociate= int(wsheet[9][j].value)
        availability=[]
        for i in range (11, wsheet.max_row+1):
            availability.append(int(wsheet[i][j].value))
        employees.append(Employee(eid, pay, lowhours, highhours, cashier, 
stocking, customerservice, backroom, floorassociate, availability))

    wsheet =wb[sheetnames[1]]
    for i in range (2,wsheet.max_row+1):
         rcashier.append(int(wsheet[i][1].value))
         rcustomerservice.append(int(wsheet[i][2].value))
         rstocking.append(int(wsheet[i][3].value))
         rbackroom.append(int(wsheet[i][4].value))
         rfloorassociate.append(int(wsheet[i][5].value))
path ="Project3data.xlsx"
employees=[]
rcashier=[]
rstocking=[]
```

```
rcustomerservice=[]
rbackroom=[]
rfloorassociate=[]

readexcelfile(path, employees, rcashier, rstocking, rcustomerservice,
rbackroom,rfloorassociate)

requirement = [rcashier, rstocking, rcustomerservice, rbackroom,
rfloorassociate]
categories= len(requirement)
time =  len(employees[0].availability)
training = [[employees[i].cashier, employees[i].stocking,
employees[i].customerservice, employees[i].backroom,
employees[i].floorassociate] for i in range(len(employees))]

print('The demands for cashiers by the company are ')
for i in range (0,len(rcashier)):
    print (rcashier[i], rstocking[i])

print('Information on the employees ')
for i in range (0,len(employees)):
    print (employees[i].eid, employees[i].pay)
    for j in range (0,len(employees[i].availability)):
        if employees[i].availability[j]==0:
            print (j, 'Not available at that time')
        if employees[i].availability[j]==1:
            print (j, 'Available at that time')


#Make your LP.
my_lp_problem = pulp.LpProblem("My LP Problem", pulp.LpMinimize)

sched = pulp.LpVariable.dicts("schedule", ((i, j, k) for i in
range(len(employees)) for j in range(categories) for k in range(time)),
lowBound=0, cat = 'Binary')

# obj and constr are added using += to the my_lp_problem class.
#Pulp assumes that the the objective function is alsyws given first.
my_lp_problem += pulp.lpSum([employees[i].pay*sched[i, j, k] for i in
range(len(employees)) for j in range(categories) for k in range(time)])

#constraints
for k in range(time):
    for j in range(categories):
        my_lp_problem += pulp.lpSum([sched[i, j, k] for i in
range(len(employees))]) >= requirement[j][k]

for i in range(len(employees)):
    my_lp_problem += pulp.lpSum([sched[i, j, k] for j in
range(categories) for k in range(time)]) <= employees[i].highhours

for i in range(len(employees)):
    my_lp_problem += pulp.lpSum([sched[i, j, k] for j in
range(categories) for k in range(time)]) >= employees[i].lowhours

for k in range(time):
    for j in range(categories):
        for i in range(len(employees)):
            my_lp_problem += sched[i, j, k] <=
employees[i].availability[k]*training[i][j]
```

```
for k in range(time):
    for i in range(len(employees)):
        my_lp_problem += pulp.lpSum([sched[i, j, k] for j in
range(categories)]) <= 1

# print (my_lp_problem)

status=my_lp_problem.solve()
if pulp.LpStatus[my_lp_problem.status] =='Infeasible':
    print ('INFEASIBLE ****************************************')


#To help you write out the solution. else:
    wbr = openpyxl.Workbook()

    #Sheets start at 0

    wbr.create_sheet(index = 0, title = "solution") sheetnames
    = wbr.sheetnames

    wsheet = wbr[sheetnames[0]]

    #Writing to individual cells
    #Cells start at 1, 1

    row =1
    wsheet.cell(row,1).value ='Employee costs' wsheet.cell(row,2).value
    = pulp.value(my_lp_problem.objective) row+=2


    for i in range (0, len(employees)):
        wsheet.cell(row , i+2).value = employees[i].eid

    beginrow = row
    for k in range(time):
        row+=1
        for i in range (0, len(employees)):
            avail = 0
            for j in range(categories):
                avail = avail + sched[i,j,k].varValue if
            avail>1:
                print("model not correct") wsheet.cell(row
            , i+2).value = avail


    days=0 hours=10
    row=beginrow
    for i in range (0, len(rcashier)): row+=1
        wsheet.cell(row , 1).value = str(days)+','+str(hours) hours+=1
        if hours==21:
            hours=10 days+=1
```