

Intelligent Emergency-Auto-Adjustment for Faculty

END TERM REPORT

By

NAME OF THE CANDIDATES

(section: **K18RD**)

(Roll Numbers: 24,31,37,38)



LOVELY
PROFESSIONAL
UNIVERSITY

Department of intelligent systems

School of Computer Science Engineering

Lovely Professional University, Jalandhar

10-04-2020

Student Declaration

This is to declare that this report has been written by us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be copied, we shall take full responsibility for it.

K. Sai kumar reddy

Roll number: 24

Kilaru Deekshita

Roll number:31

Naravade Yash Sandeep

Roll number: 37

Abhishek Singh

Roll number: 38

Lovely professional university

10-04-2020

TABLE OF CONTENTS

1. INTRODUCTION

2.ABSTRACT

3.PROBLEMS

4.COLLISIONS

5.CONTRAINS

6.CODE

7.TESTING AND RESULT

8.USE

INTRODUCTION

It is basically a Timetable generator for university schedules when few faculty members are on leave. This program is implemented in Python using *genetic algorithms*.

The class timetabling problem is a scheduling algorithm with great interest and implications in the fields of operational research and artificial intelligence. The problem was first studied by Gotlieb, who formulated a class-teacher timetabling problem by considering that each lecture contained one group of students and one teacher, such that the combination of teacher and students can be chosen freely. Dynamic changes in the context of timetabling problems, had started to be studied at. A survey of existing approaches to dynamic scheduling can be found in. Because of the size of the real problem, almost all effective solutions are heuristic in nature, and do not guarantee optimality. Among the well-known results there are that deal with various cases of the problem settings. While setting a timetable, importance is given to effective utilization of resources such as the classroom, the teacher, etc.` This becomes a very tedious task which needs to be addressed at least once a year by every academic institute. Most institutes deal with this problem manually, i.e. a trial and error method are used to set a timetable.

ABSTRACT

This project implements one of possible solutions for generating university schedules. The proposed solution is based on methods of evolutionary computing, uses *(1+1) evolutionary strategy* and *simulated hardening*.

The success of a solution is estimated on fulfillment of given constraints and criteria. Results of testing the algorithm show that all hard constraints are satisfied, while additional criteria are optimized to a certain extent.

This paper proposes a general solution for the School timetabling problem. Most heuristic proposed earlier approaches the problem from the students' point of view.

This solution, however, works from the teachers' point of view i.e. teacher availability for a given time slot. While all the hard constraints (e.g. the availability of teachers, etc.) are resolved rigorously, the scheduling solution presented in this paper is an adaptive one, with a primary aim to solve the issue of clashes of lectures and subjects, pertaining to teachers. Index Terms— timetabling, scheduling, operational research, artificial intelligence, heuristic.

PROBLEMS

The assignment is to find a generic solution that will facilitate generating schedules for the university.

Each class on faculty is represented as a block (lasts an arbitrary number of hours, mostly from 1 to 4). For conducting every class required are the teacher, *classroom*, *start time*, *duration* and *groups* which attend the class. It is also known in advance which groups attend which class and all classrooms are the same size (each group can fit to a classroom). Teaching is done on faculty from 9AM until 5PM on each workday.

Input data is the classroom and time for each class. Time is determined by day (Monday to Friday) and the start of the class.

K18RD TIME-TABLE								
DAYS	9-10	10-11	11-12	12-1	1-2	2-3	3-4	4-5
MONDAY	Break	Teacher_C	Teacher_A	Teacher_B	Teacher_C	Break	Break	Teacher_D
TUESDAY	Teacher_A	Teacher_B	Teacher_D	Break	Teacher_E	Teacher_A	Teacher_C	Teacher_E
WEDNESDAY	Teacher_B	Teacher_A	Teacher_D	Teacher_E	Teacher_C	Break	Teacher_B	Break
THURSDAY	Teacher_A	Teacher_B	Teacher_C	Teacher_E	Teacher_D	Break	Teacher_A	Break
FRIDAY	Break	Teacher_A	Teacher_C	Break	Teacher_B	Teacher_A	Teacher_E	Teacher_E

COLLISION

There is a possibility that teacher availability for a subject may be at a slot where another subject s_j is allocated. Under such a situation, if s_j is not present in the output data structure, s_j is moved into a Clash data (i.e. no more free time slots are available), the Clash data structure is revisited and an effort is made to allocate the subjects in it to an available time slot in the day. If, however, it is not possible to allocate any/all the subjects in the Clash data structure, these subjects are moved to the Day_Clash data structure. When the sequence for the next day is generated preference is given to the subjects under Day_Clash.

CONSTRAINTS

1. Resources cannot overlap timewise
 - No teacher can hold two classes at the same time
 - No group can listen for two classes at the same time
 - No classroom can receive two classes at the same time

Note: under the term "same time" is not meant only at the beginning of the class, it should be considered the duration of the class. If the resource is busy at the moment T_1 and the class lasts t_1 , then the resource can only be re-occupied at the moment $T_2 = T_1 + t_1$.

2. Class should take place in one of the allowed classrooms
3. If the subject has several forms of teaching, the preferred order for each group is the *lectures*, *exercises*, and *laboratory exercises*.

Constraints 1 and 2 must be met, while the 3rd limit is "soft" and allowed to be violated.

Additional criteria for estimating solution (used also for cost function):

- Fulfill hard constraints (1 & 2)
- Fulfill soft constraints (3)

- Minimize total "idle" for each group (eliminating pause between classes)
- Minimize total "idle time" for each teacher (elimination of pause between classes)
- Provide one hour on a teaching week where no one has classe

CODE

1. Loading and processing data

The classes are loaded from the file using file handling in python and are allocated to the appropriate structures. Also, random data mixing is done.

2. Model setting

Creation of schedule matrix and dictionaries representing empty fields, filled fields in a matrix, order of classes for each type of class and group, an empty group idle, and an empty teacher's idle.

3. (1 + 1) evolution strategy for hard constraints

The selection of classes that change the place in the schedule is based on the cost of the classes calculating only the hard constraints. Classes with the highest cost are chosen and with a certain probability mutated. Mutation is changing the field in the matrix by looking for a free field that meets all the rigid constraints. Schwefel's notation is also used.

4. Simulated hardening for additional criteria

5. Representing schedule

Below is the code implementation

```
import os  
  
import sys
```

```
import string

def loginCall(username,Pass):

    Password=open("Main//pass.txt",'r')

    Login=False

    for i in Password:

        st=i.rstrip()

        tempPass=st.split()

        if len(tempPass)==0:

            continue

        if(username==tempPass[1] and Pass==tempPass[3]):

            Login=True

    return Login

#Starting interface

print("\n\n\n\t\t\t\t\t\t\t\t\t\tWELCOME TO FACULTY LEAVE REQUEST SYSTEM\n\n\n\n\n")

trialLeft=3

while(trialLeft):

    username=input("\t\t\t\t\t\t\t\t\tEnter username: ")

    Pass=input("\t\t\t\t\t\t\t\t\tEnter password: ")
```



```
if loginCall(username,Pass):

    os.system('cls' if os.name == 'nt' else 'clear')

    break

else:

    print("Wrong username or password")

    trialLeft-=1

    print("trialLeft: ",trialLeft)

if(trialLeft==0):

    print("All attempt wrong contact system administrator")

    sys.exit()


#Main interface

print("\n\n\t\t\t\t\tFACULTY LEAVE REQUEST INTERFACE\n\n")

print("Welcome! {}".format(username))


schedule=open("Student_Timetable//K18RD.txt",'r')

timeTable=[]

for i in schedule:

    print(i)

    tempTimeTable=i.split()

    if len(tempTimeTable)==0 or tempTimeTable[0] not in
```

[illegible]

```
    if tempArr[0]==Day:

        TimeSlots=[z for z,val in enumerate(tempArr) if val==username]

fh.close()

#finding available teachers for adjustment

Teachers=['Teacher_A','Teacher_B','Teacher_C','Teacher_D','Teacher_E']

TeacherAvailable=[]

for i in Teachers:

    if i==username:

        continue

    fh=open("Faculty_Timetable//{}.txt".format(i),'r')

    for k in fh:

        tempArr=k.split()

        if len(tempArr)==0:

            continue

        if tempArr[0]==Day:

            for p in TimeSlots:

                if tempArr[p]=='B':

                    TeacherAvailable.append([tempArr.count('O'),p,i])

    fh.close()
```

```

#Check for any left timeSlots

count=0

for x in TimeSlots:

    for i in TeacherAvailable:

        if i[1]==x:

            count+=1

            break

if len(TeacherAvailable)==0:

    print("No Teachers available for adjustment")

    print("Cannot apply for leave")

    sys.exit()

elif count!=len(TimeSlots):

    print("No Teachers available for adjustment for some lectures")

    print("Cannot apply for leave")

    sys.exit()

#Choose Teacher for adjustment according to less workload on teacher that day

TimeSlotsDic={1:'9-10',2:'10-11',3:'11-12',4:'12-1',5:'1-2',6:'2-3',7:'3-4',8:'4-5'}

TeacherAssigned=[]

for i in TimeSlots:

    workload=[]

```

```

for t in TeacherAvailable:

    if t[1]==i:

        workload.append([t[0],t[1],t[2]])

workload.sort(reverse=True)

TeacherAssigned.append([workload[0][1],workload[0][2]])

    print("\nTeacher for Adjustments for {0} slot from {1} with least Workload:
".format(Day,TimeSlotsDic[i]),workload[0][2])

#change the timetable

TeacherAssigned.sort()

schedule=open("Student_Timetable//K18RD.txt",'r')

leave=open("Student_Timetable//LEAVE.txt", 'w')

for i in schedule:

    if Day in i:

        if len(TeacherAssigned)==1:

            line=i.replace(username,TeacherAssigned[0][1],1)

            leave.write(line)

            continue

        elif len(TeacherAssigned)==2:

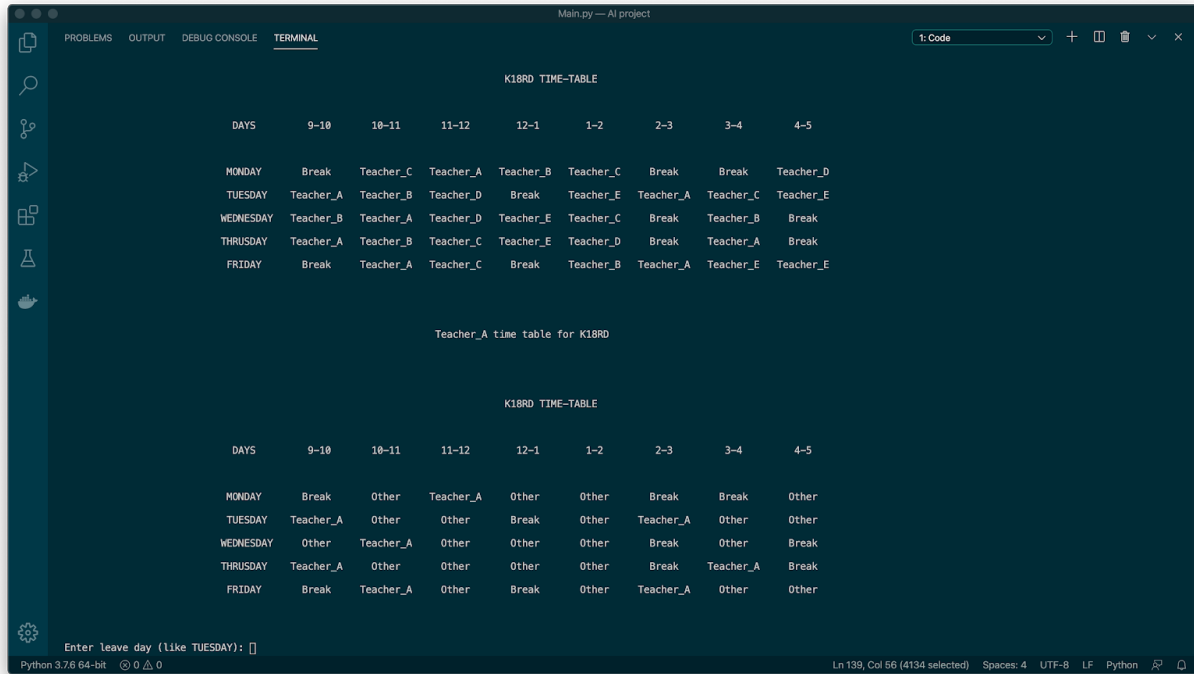
line=i.replace(username,TeacherAssigned[0][1],1).replace(username,TeacherAssigned[1][1],1)

            leave.write(line)

```


4.Auto adjustment is made using the rule-based system.

Output for TEST CASE:



```
Main.py — AI project
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Code + - ×

K18RD TIME-TABLE

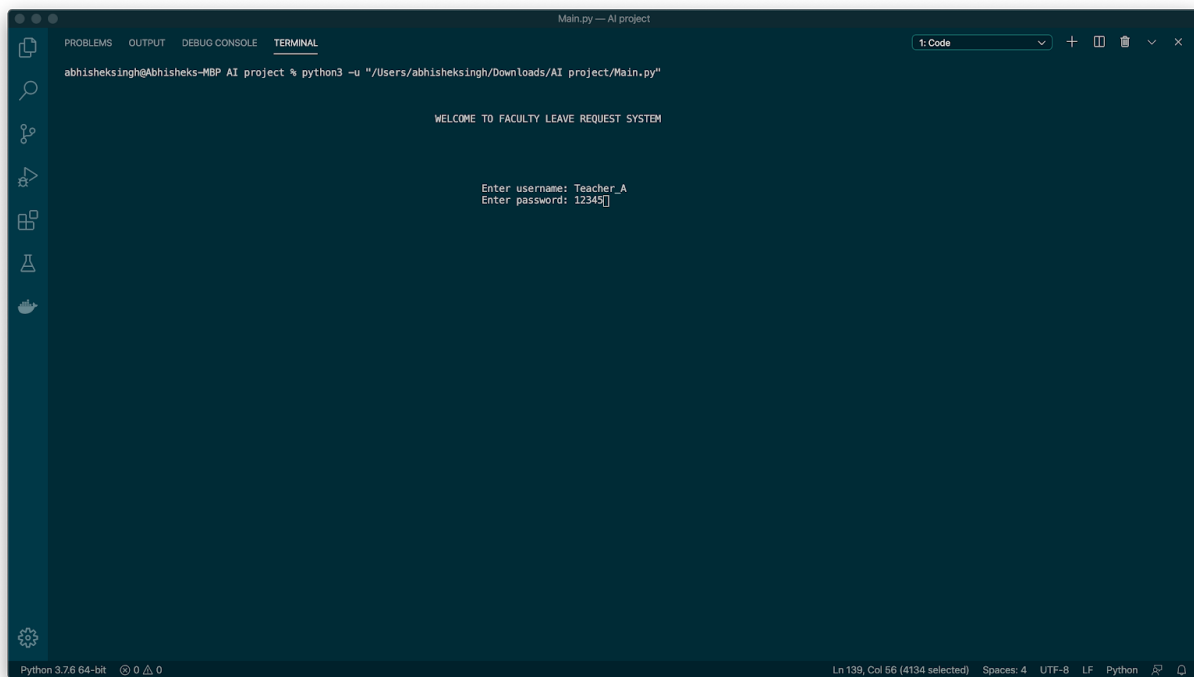
DAYS      9-10    10-11    11-12    12-1     1-2      2-3      3-4      4-5
MONDAY     Break    Teacher_C Teacher_A Teacher_B Teacher_C Break     Break    Teacher_D
TUESDAY    Teacher_A Teacher_B Teacher_D Break     Teacher_E Teacher_A Teacher_C Teacher_E
WEDNESDAY  Teacher_B Teacher_A Teacher_D Teacher_E Teacher_C Break     Teacher_B Break
THURSDAY   Teacher_A Teacher_B Teacher_C Teacher_E Teacher_D Break     Teacher_A Break
FRIDAY     Break    Teacher_A Teacher_C Break     Teacher_B Teacher_A Teacher_E Teacher_E

Teacher_A time table for K18RD

K18RD TIME-TABLE

DAYS      9-10    10-11    11-12    12-1     1-2      2-3      3-4      4-5
MONDAY     Break    Other    Teacher_A Other    Other    Break     Break    Other
TUESDAY    Teacher_A Other    Other    Break    Other    Teacher_A Other    Other
WEDNESDAY  Other    Teacher_A Other    Other    Other    Break     Other    Break
THURSDAY   Teacher_A Other    Other    Other    Other    Break     Teacher_A Break
FRIDAY     Break    Teacher_A Other    Break    Other    Teacher_A Other    Other

Enter leave day (Like TUESDAY):
```

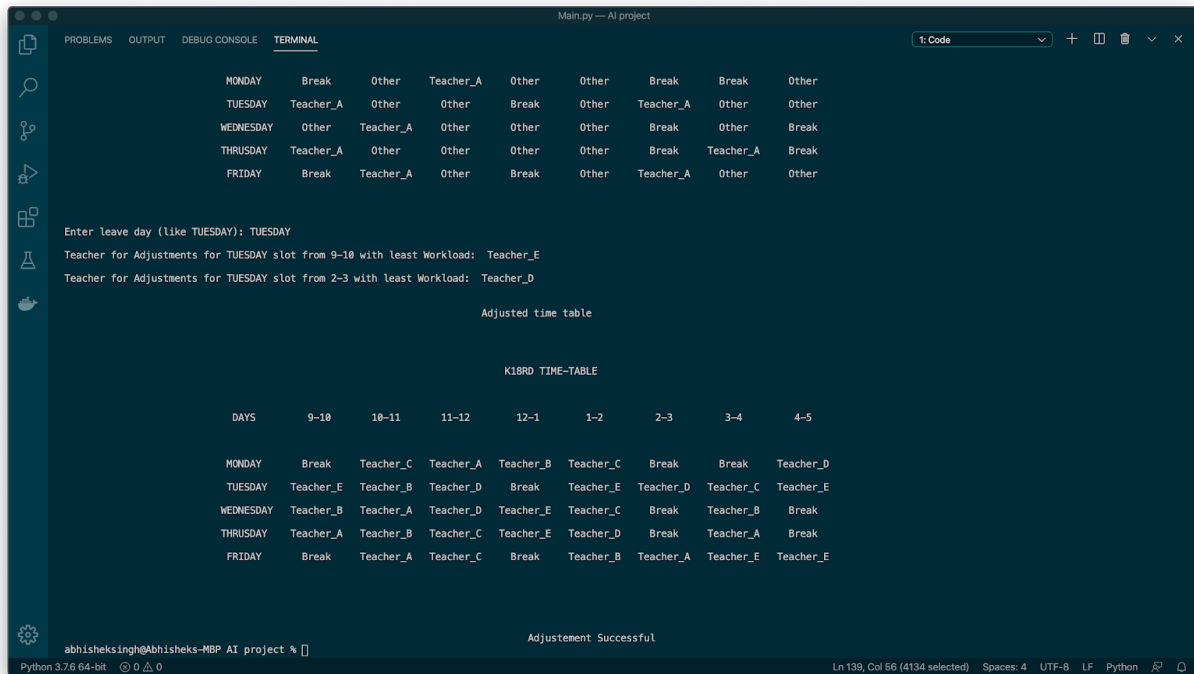


```
Main.py — AI project
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Code + - ×

abhisheksingh@abhisheks-MBP AI project % python3 -u "/Users/abhisheksingh/Downloads/AI project/Main.py"

WELCOME TO FACULTY LEAVE REQUEST SYSTEM

Enter username: Teacher_A
Enter password: 12345
```



```

Main.py — AI project
1: Code

MONDAY    Break    Other    Teacher_A    Other    Other    Break    Break    Other
TUESDAY   Teacher_A    Other    Other    Break    Other    Teacher_A    Other    Other
WEDNESDAY    Other    Teacher_A    Other    Other    Other    Break    Other    Break
THURSDAY   Teacher_A    Other    Other    Other    Other    Break    Teacher_A    Break
FRIDAY     Break    Teacher_A    Other    Break    Other    Teacher_A    Other    Other

Enter leave day (like TUESDAY): TUESDAY
Teacher for Adjustments for TUESDAY slot from 9-10 with least Workload: Teacher_E
Teacher for Adjustments for TUESDAY slot from 2-3 with least Workload: Teacher_D

Adjusted time table

K18RD TIME-TABLE

DAYS      9-10    10-11    11-12    12-1    1-2    2-3    3-4    4-5

MONDAY    Break    Teacher_C    Teacher_A    Teacher_B    Teacher_C    Break    Break    Teacher_D
TUESDAY   Teacher_E    Teacher_B    Teacher_D    Break    Teacher_E    Teacher_D    Teacher_C    Teacher_E
WEDNESDAY    Teacher_B    Teacher_A    Teacher_D    Teacher_E    Teacher_C    Break    Teacher_B    Break
THURSDAY   Teacher_A    Teacher_B    Teacher_C    Teacher_E    Teacher_D    Break    Teacher_A    Break
FRIDAY     Break    Teacher_A    Teacher_C    Break    Teacher_B    Teacher_A    Teacher_E    Teacher_E

Adjustment Successful

abhisheksingh@Abhisheks-MBP AI project %
Python 3.7.6 64-bit 0 0 0 Ln 139, Col 56 (4134 selected) Spaces: 4 UTF-8 LF Python
```

USE

Running `python3main.py` runs the code executing the algorithm for the data that is loaded from the file whose path is in the main function in the same file.

GitHub Link: -<https://github.com/AbhishekSingh1180/Artificial-Intelligence-Project>

BONAFIDE CERTIFICATE

Certified that this project report “Intelligent Emergency-Auto-Adjustment for Faculty” is the bonafide work of “ Abhishek singh,Kaluvakolu Sai kumar reddy, Narvade Yash Sandeep, Kilaru Deekshita” who carried out the project work under my supervision.

<<Signature of the Supervisor>>

Mr. Ashish Shrivastava

Assistant Professor

Department of intelligent systems

School of Computer science engineering