The University Class Scheduling Problem Programming Report

(A1714470)

Once the requirements document for the assignment to be developed is available, my software design phase begins. When the requirements specification activity completely addresses the class scheduling problem domain, the design is the first phase of turning the problem into a solution. In the design phase, problem partitioning and software structure design needs, as well as technical considerations, together form the class scheduling system. In fact, several main aspects in the programming process can solve class scheduling problems. I will report on several aspects, including preparing the programming environment, basic approach, algorithms, results in testing, and reflection.

To begin with, the programming environment has a shallow impact on the programming process in this case for a variety of reasons. Because I mainly use the sublime text before. This software is not allowing me to debug. So I have to find another way to set a programming environment for doing this assignment. Because start to set up the environment. I started reading the Programming Assignment description. I have to set up an svn directory in my pc and the server first, copy the provided files into it, and then try to run the test script with add some a little code for testing. So I can understand the program logic of the assignment. And then download the files and then try to import them into Xcode. I am trying to set up a Programming Assignment makefile project in Xcode: syntax highlighting, auto-completion, jump to definition and more! Get the tools to build my Programming Assignment project in Xcode. In this case, the crazyflie project requires the GNU Embedded Toolchain for Arm. They provide several installation alternatives (virtual machine, docker, source code build, download precompiled binaries from ARM), but the easiest way I found is to use this brew formula, which gets the last precompiled from ARM. Binary files and configure them. And the get the Programming Assignment EvalUCS source code. To import my Programming Assignment to Xcode and test it. In this case, get the crazyflie firmware. Because this specific project makes use of git submodules, I need to also pass the --recursive argument. Later, I finished setting up a programming environment.

Besides, finishing the basic approach is very important as my solutions are developed incrementally. So I try to use the basic approach. I need to generate an initial timetable allocation, ignoring the lecturer's preferences, and treating all non-busy periods the same. I will also ignore the lecture theatre limit. This will make it easier to find an initial solution. And then I want to handle the complexity of the design process effectively. Effectively design of the complexity will not only reduce the effort needed for design but can also reduce the scope of introducing errors during

design. I go-ahead to construct the ScheduleSystem class. I need to generate an initial timetable allocation. Hence, I need just need to revise three functions Cell, ArrangeTimetable, CreateTimetable, and OutputSolution. In these classes not included any constraint violation rule. To do the basic approach, "Cell" just return -1, ArrangeTimetable just assigns -1 to each time. And then CreateTimetable is used to construct the Timetable matrix. Lastly, the OutputSolution function is used to create a solution file.

Furthermore, greedy algorithms and brute force are mainly used to find out the solution in this class scheduling problem for a variety of reasons. It is a course schedule that is a large problem. So I have to consider to break down the large problem into several small problems. Moreover, to cater to this small program. I have to construct several classes to handle the entire problem at once but the significant problem, divide the problems and conquer the problem it means that I have to divide the course scheduling problem into smaller pieces so that each piece can be captured separately. So for my programming design, the goal is to divide the problem into manageable pieces. There have several classes that I would like to create. The first is the Utility. This class is used to contain the helper function. Below is a list of helper functions:

1) It is used to convert the value to the string
```
template <typename T>
std::string to_string(T value);
```

2) It is the right way to split a string into a vector of strings
```
static std::vector<std::string> split(std::string s, std::string t);
```

3) Checks whether the input string is a decimal digit character.
```
static bool is_digit(std::string str);
```

4)Checks whether the input string is a positive number.
```
static bool is_positive_number(std::string str);
```

5)Trim all the whitespace from a string
```
static std::string trim(std::string str);
```

6)Find integer from the vector
```
static bool FindInt(std::vector<int> v,int val);
```

7)Selection sort
```
static std::vector<int> sel_sort(std::vector<int> v);
```

8)Swap two value
```
static void swap_int(int *x, int *y);
```

9) The maximum value of the integer
```
static int _INT_MAX;
```

10)Create a file and write
```
static void file_w(std::string file,std::string s);
```

11)Check whether the file exists
```
static bool is_file_e(char *fileName);
```

The next is ScheduleSystem which is used to create the solution. And then, I should create CourseManagement and CourseData classes. This class is for processing constraint violation of course scheduling. And then the last thing is the Teacher and TeacherManagement classes. These classes are used for processing constraint violation of lecturer allocating. Therefore, I have to use these classes to solve a few minor problems shown in the following list.

1) Creating the solution
2) Course Management
3) Handle Course Data.
4) Processing constraint violation of course scheduling
5) No more than two hours of course.
6) Prevent to hold two of the same courses on the same day.
7) Processing constraint violation of lecturer allocating.
8) Catering the TL binary mapping.
9) Catering the LP preferences.

As I mentioned before my greedy algorithm is to make the course scheduling problem break into several small problems. Then my program is easy to understand, becomes simple, easy to test, easy to modify, easy to maintain, and easy to expand. Hence, I will create several functions for these classes to help me to cater to several small problems.

Teacher(class):

After rest, the remain hours will back to default hours.
void Rest(int time, bool must)

This function is used to record daily working hours.
int UsedHour(int time)

This function is used to find remain working hours of the lecturer.
int FindRemainHours(int d)

TeacherManagment(class):

Checking whether the lecturer is available on that session
bool IsLecturerAvaliable(int l,int time);

Set all teachers rest and reset the remain hours.
void AllRest(int time,bool must);

Allocated lecturer at that time.
bool Allocated(int l,int time);

Checking whether the teacher is allocated at that time.
bool IsTeacherLocated(int c,int l);

Checking the lecturer's preferences.
bool IsLecturerPreferences(int l, int t, int p);

These pieces cannot be entirely independent of each other as they together form the system. They have to cooperate and communicate to solve the problem. This communication adds complexity. And I use the greedy algorithm because it is a simple and intuitive algorithm used to optimize problems. When the algorithm tries to find an overall optimal method for solving the whole problem, it makes an optimal choice in each step.

```
Procedure CreateTimetable(){
   Timetable.clear();
   Initialization_of_the_Timetable();
   Arrange_Timetable();
   Assign_the_lecturer_to_the_course();
   std::vector<int> corderlist;
   Repeat{
      corderlist=AssignCourseHoursLeft(corderlist);
      if(corderlist.size()>getMC())
         corderlist.clear();
   }
   while(Course.TotalHours()>0);
   PrintCourseHours();
EndProcedure.
```

```
Procedure Cell(int c,int t,int torder, int p){
   if(LunchHour!=(t+1)%8 &&
      IsCourseAvaliable(c) &&
      !IsSeparateSession(c,t) &&
      !IsWrongAssignedCourse(c,t,2) &&
      Course.TotalHours()>0){
       if(IsRoomAvailable(t)){
          std::vector<int> LecturerOrder=Lecturer.getLectureAssignOrders()[torder];
          Repeat(int l=0;l<LecturerOrder.size();l++){
             int lect=LecturerOrder[l];
             if(c==0 && lect==3){
                c=0;
             }
             if(IsTeacherLocated(c,lect)){
                if(p>0){
                   if(IsLecturerPreferences(lect,t,p)){
                      if(Lecturer.Allocated(lect,t)){
                         RoomReservation(t);
                         CourseHeld(c);
                         return lect;
                      }
                   }
                }
                else{
                   if(IsLecturerPreferences(lect,t,1) ||
                     IsLecturerPreferences(lect,t,2) ||
                     IsLecturerPreferences(lect,t,5)){
                      if(Lecturer.Allocated(lect,t)){
                         RoomReservation(t);
                         CourseHeld(c);
                         return lect;
                      }
                   }

                }
             }
          }
       }
   }
   Lecturer.AllRest(t,false);
EndProcedure
```

As you can see, During arranging the time table. The program will attempt to assign the lecturer to the course of the course. If the program finds that it has not assigned all the time, of course, the program will record the courses and then my program will make the best choice in the next step. The program will process these courses first and then the remaining courses.

Moreover, I have to test my program to make a final solution. I run below a list of commands and I got the result from EvalUCS.

./schedule simple1.ucs.txt fnl_soln.sch

rooms: 2 mCourses: 5
hours read, course names read
lecturers: 4 lecture names read
LP read
calling readSolution
Timetable read
calling constrains
your solution is feasible
Fitness value 2.30769

./schedule medium1.ucs.txt fnl_soln.sch
rooms: 2 mCourses: 10
hours read, course names read
lecturers: 6 lecture names read
LP read
calling readSolution
Timetable read
calling constrains
your solution is feasible
Fitness value 1.96875

./schedule medium2.ucs.txt fnl_soln.sch
rooms: 1 mCourses: 10
hours read, course names read
lecturers: 6 lecture names read
LP read
calling readSolution
Timetable read

calling constrains
your solution is feasible
Fitness value 2.46875

Finally, in this assignment, there is almost a need for conciseness - when the code is compiled, the OOP concept will be important. Therefore, readability and scalability are key. By indenting the unique shape of the code block, I can quickly find my class statement, then figure out which function to satisfy which type of issues and quickly determine the position of the clause. Some parts it too complex so it doesn't look like normal code. Also, if I decide to add additional helper functions (Utility) to make it easier for me to figure out programming problems, I will find it much easier if I start with the first method. As you can see my program is a success to find out the solution. So my approach is an algorithmic paradigm that builds solutions step by step and always chooses the next solution that offers the most obvious and direct benefits. Therefore, choosing local optimums will also lead to problems in the overall solution that are best for me to use a greedy algorithm for this course scheduling problem.