0 Index

1 Report

1.1 Summary of the document

TODO: EXPLAIN MORE DETAILS ABOUT THE SECTIONS

This document presents all the important information regarding the *Classroom management at the School of Computer Engineering using Artificial Intelligence methods* end-of-degree thesis.

First, a report of the project is given. It functions as a general summary of the work done. This is followed by the introduction to the project, which gives a justification for the project, lays down the objectives and analyses the past and present situation.

After that, the theoretical sections of the project are presented. They consist of a number of definitions of key concepts, a formal definition of the problem to solve and a detailed explanation of the proposed solution.

Then the project planning and budget overview are shown.

Subsequent to the project planning rundown follow the software engineering chapters, which include the system analysis, design and implementation and the test development.

After outlining the specifications of the system, the results of the experiments conducted are listed.

The system manuals for the programmer and the user are handed.

Finally, the conclusions and future work are listed, along with the final budgets, the bibliography, annexes and the source code.

It is important to note that the structure of the contents for this document is done following the criteria and recommendations of the template document for Degree's and Master's Thesis of the School of Computing Engineering of Oviedo (version 1.4) [1] by J. M. Redondo. However, some additional chapters were introduced by the author in order to capture the particularity of the work carried out, inspired by the research of G. de la Cruz [2].

1.2 Purpose

This project aims to help the personnel of the School of Computing Engineering at the University of Oviedo manage their classrooms. It will address two main functionalities, the automation of the process of assigning classrooms to all the groups of a given semester (starting from scratch or using a previous partial or total assignment), and a tool that searches for gaps in a previous set of assignments for single or multi-day events in one or more classes.

The implementation of this system is intended to assist in the work of the supervisor for this process, and provide an efficient and flexible tool that expands the possibilities of such work. To do so, the program executes two algorithms, a genetic algorithm guided by a greedy algorithm. For a more detailed view on these algorithms the reader might refer to 3. Once the assignments have been calculated, the system will allow the users to find classrooms to hold specific events in the middle of the semester.

Along with the system, the system manuals are submitted. These have the purpose of teaching how to install, use, maintain and extend the system. Apart from the manuals, another tool to generate the necessary files for the program is handed.

1.3 Scope

The project needs to formally define the problem of assigning classrooms of the School to all the groups of the semester, conduct a study on the problem and propose a solution.

A development of a software prototype that solves the problem is planned, designed, implemented and tested. This prototype will solve the two main functionalities indicated in 1.2 and will consist of a command line application that takes input data in plain text files and outputs the solution to plain text files. The program is configured by different configuration files depending on the functionality being executed. An experimental study on the results of the software system is carried out, finding the most fitting default values for the configuration files. The project also contains the system manuals of the application, which consist of the installation, usage, user and programmer manuals.

Finally, an additional tool for automating the creation of the input files of the software is given. It uses a format agreed with the client and will use the same technical specifications of the main prototype, like the programming language and the development environment.

2 Introduction

2.1 Project justification

The School of Computing Engineering of the University of Oviedo has more than twenty classrooms, including theory classrooms and laboratory classrooms. Each semester there are over three hundred groups, each with their type (theory, seminar or laboratory), subject and schedule. The timetable of the groups varies on a weekly basis, this means that not all groups have to attend classes all weeks, and some of them do not even have repeating patterns.

This makes assigning classrooms to groups a complicated task, since there can be no temporal collisions. When various other constraints enter the equation, such as minimising the number of labs used by a subject or assigning classrooms to Spanish groups that are different from English groups, things become much more complex.

All this assignments are done *manually* by one person. Because groups can change once the semester has already started, more assignments usually made, checking once again all the restrictions.

This project provides the supervisor of this process with a tool to help them calculating the assignments, reducing their workload. Not only does it generate assignments for all the groups of the semester, but can use previous assignments, total or partial, to calculate a subset of assignments (for example, the assignments for the new groups created in the middle of the semester). On top of that, the prototype developed in this thesis makes finding a set of free classrooms to hold events easy and fast, using the assignments generated previously by the system itself.

2.2 Project goals

The project seeks to achieve the following objectives:

1. Formally define the problem of assigning classrooms to the groups of the School.
2. Study the problem and the means to solve it.
3. Define the proposed solution.

4. Build a prototype that solves the problem using the algorithms described in the proposed solution.
   (a) It will receive plain text input files with the required data.
   (b) It will output the solution to plain text files.
   (c) It will be able to make the assignments starting from scratch or from a total or partial set of assignments.
   (d) It will be able to search a set of free classrooms for a specific event in one or more days.
5. Make a set of experiments to find the best default values for the configuration files.
6. Write a set of manuals to cover the essentials of the system.
7. Create another software tool that will automate the creation of the input plain text files for the main system.
8. Validate solution with the users.

2.3 Situation overview

2.3.1 Background

At the beginning of each semester, the School of Computing Engineering of the University of Oviedo opens a process in which the person in charge takes the list of groups for the semester, their schedules and the list of classrooms, and performs a manual compilation of all the assignments.

There are a number of other similar procedures, like the creation of the exam timetable or the assignments of enrolled students to subject groups. However, some are not manual, but automated by a system, like the previously mentioned procedure of assigning students to groups. Seeing the potential of such tools, the author of this project was given the task of automating the assignment of classrooms to subject groups by similar means.

2.3.2 Description of the current situation

As explained in the previous section, the school has a supervisor for the process of assigning classrooms to groups. This procedure is done after configuring the student groups for the

semester and knowing their schedules.

Even though it is a manual process, the supervisor does not start making the assignments from scratch. First, they have the knowledge of previous years, and then they have a list of preferences or premade assignments. For example, certain laboratories can only be assigned to specific groups, like the ones from the Electronic Technology of Computers subject.

The system described in this document preserves these sources of information and builds on top of them.

3 Theoretical aspects

4 Problem definition

5 Proposed solution

6 Project planning and budget overview

6.1 Planning

6.2 Budget summary

7 Analysis

7.1 System definition

7.2 System requirements

TODO: THESE ARE THE INITIAL REQUIREMENTS, THEY WILL CHANGE

The requirements listed here are a basic overview of the fundamental functionality covered by the project. For the complete list of in-depth requirements the reader might refer to NOPE.

7.2.1 Interface

- The program must implement a CLI.
  - The CLI must show basic or complete information to the user depending on the given option flag.
  - The CLI must show the encountered errors to the user before terminating the execution.
  - The CLI must have help, license and version options.

7.2.2 Input

- The progam receives as input the classrooms, groups, group schedule and the academic weeks of each group.
- The program might optionally receive as input a subset of assignments already performed.
- The program might optionally receive as input a previous complete list of assignments but without some of the classrooms/laboratories used in it.
- The program might optionally receive as input a previous complete list of assignments but with more or less groups.
- The program might optionally recieve as input a list of classroom preferences for the groups of a particular subject, given their type (theory or laboratory) and language (english or spanish).

7.2.3 Configuration

- Program configuration must allow the user to control the parameters of the genetic algorithm.
- Program configuration must allow the user to change the version of the program.
- Program configuration must allow the user to specify the folder paths for the log and output files.
- Program configuration can change in the middle of the course.

7.2.4 Algorithm

- The program must use a genetic algorithm guided by a greedy algorithm.
- Language group requirements:
  - English groups should go to different classrooms/laboratories from the spanish groups.
- Classroom requirements:
  - Some initial classroom assignments can be specified before the execution of the program and they must remain the same.
  - The program must be able to find a gap in the current list of assignments to include a (mono/multi)-(classroom/laboratory).
  - The number of groups of the same number and course assigned to the same theory classroom must be maximised.
  - The number of groups of the same subject assigned to the same laboratory must be maximised.
  - In each time slot there must be a minimum number of free laboratories.
  - Some big laboratories must be empty for emergency reasons.
  - The program must penalise assignments where the number of students is far below the number of computers.
  - The laboratories must have some free space defined by the user.
  - In small laboratories (of 16 computers) there must be at least two free computers.
  - The program must be able to handle a split in two of a laboratory group with only one professor (for emergencies).

7.3 Subsystem mapping

7.4 Preliminary class diagram

7.5 Analysis of use cases

7.6 Analysis of user interfaces

7.7 Test plan specification

8 System design

8.1 System architecture

8.2 Class design

8.3 Interaction and state diagrams

8.4 Activity diagram

8.5 Interface design

8.6 Technical specification of the test plan

9 System implementation

9.1 Standards and references

9.1.1 Standards

9.1.2 Licenses

The software of this project is licensed under the GNU General Public License v2.0.

9.1.3 Other references

Java Code Conventions. Set of guidelines and conventions for programmers to consider when using the Java programming language.

9.2 Programming languages

There were two programming languages considered for the implementation of the system, **C** and **Java**.

Considering that:

- The author and only developer of the system has worked with Java throughout his university studies, but only used C in one subject and in some of his personal projects.
- Java is probably less efficient than C when executing the genetic and greedy algorithms.
- Java code is more easy to run in other systems than C code.
- The program is going to be executed only a few times a year.

For this reasons, even if C would be faster in execution, because the program will not be running every day, and taking into account the other two advantages, Java was the language of choice for implementing the system.

9.3 Tools and programs used in development

9.4 System development

10 Test development

10.1 Unit tests

10.2 Integration and system tests

10.3 Usability and accessibility tests

10.4 Performance tests

11 Experimental results

12 System manuals

12.1 Installation manual

12.2 Execution manual

12.3 User manual

12.4 Programmer manual

13 Conclusions and future work

13.1 Final conclusions

13.2 Future work

14 Budget

14.1 Internal budget

14.2 Client budget

15 Bibliography

Bibliography

[1] J. M. Redondo: Documentos-modelo para Trabajos de Fin de Grado/Master de la Escuela de Informática de Oviedo. Version 1.4. University of Oviedo.

[2] G. de la Cruz, C. Mencía, F. Álvarez: Metaheuristics for the assignment of students to class groups. University of Oviedo. 2018.

16 Annexes

16.1 Definitions and abbreviations

Listed below is a glossary of definitions and abbreviations used in the document whose meaning may not be obvious.

Glossary of definitions:

- **Genetic algorithm:** metaheuristic search and optimization algorithm.
- **Greedy algorithm:** algorithm that builds the solution in successive steps, always trying to take the optimal solution for each step
- **Heuristic:** function that gives value to each path in a search algorithm, based on current information.
- **Java:** general-purpose, high-level, object-oriented programming language.
- **Metaheuristic:** high-level heuristic that guides the search in a combinatorial optimization problem.

Glossary of abbreviations:

- **CSV:** Comma-Separated Values. Refers to a text file format.
- **CLI:** Command Line Interface.
- **GNU:** GNU is not Unix (recursive acronym). Refers to the free software project announced by Richard Stallman.
- **TXT:** Text. Refers to the text file format.
- **UNE:** in spanish, *Una Norma Española*. Refers to the Spanish Association for Standardisation.

16.2 Submission contents

17 Source code