0 Index

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 organization 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) by J. M. Redondo.

1.2 Purpose

This project aims to create a program that automates the process of assigning classrooms to all the groups of a given semester.

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 **??**.

Along with the system, the user and programmer manuals are submitted. These have the purpose of teaching how to use, maintain and extend the system.

1.3 Standards and references

1.3.1 Standards

UNE 157801. General criteria for the design of information systems projects. This standard is used for defining the structure and contents of this document. However, some sections have been added, removed or modified to fit the format of a final degree project. A conclusions chapter has been introduced to the general structure as well.

1.3.2 Licenses

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

1.3.3 Other references

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

1.4 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.
- **Program:** The software described in this document.
- **System:** The software described in this document.
- **Technical reader:** used in this document as someone who knows about software engineering, but not necessarily about genetic or greedy algorithms.

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.

1.5 Initial requirements

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 **??**.

1.5.1 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.

1.5.2 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).

1.5.3 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.

1.5.4 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).

1.6 Assumptions and restrictions

The assumptions and constraints that directly or indirectly affect the project are listed as follows.

- **Duration of the project.** The project started on January 28th, 2022, and must be completed before July 6th, 2022.
- **Failure to find an optimal solution.** It could be the case that the genetic algorithm failed to find the optimal solution in a reasonable time. This implies a reconsideration of the constraints of the problem, or in any case a change in the evaluation of the algorithm's solutions.
- **Unbearable computational cost.** Related to the previous assumption, if the optimal solution requires a machine with unreasonable high specs in order to solve the problem optimally, a change in the objectives of the project could be in place.

1.7 Study of alternatives and feasibility

1.7.1 Programming Language

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.

1.7.2 Format of data files

There were two file formats taken into consideration, JSON and CSV.

Bearing in mind that:

- Java has no native support for processing either format.
- JSON is easier to manage by web services.
- CSV is easier to read and write from Java than JSON.
- CSV can be imported into a spreadsheet editor like LibreOffice Calc or Excel for its manipulation.
- It is preferred not to use external code libraries to avoid licensing issues.

The decision was to choose CSV as the file format for most of the input and output files of the system.

1.8 Description of the proposed solution

The software solution proposed in this document consists of a Java command line application that takes as input the CSVs of the classrooms, subjects, groups, group schedule and group academic weeks and transforms them into a CSV containing the allocation of one class for each group, as well as a TXT file with the details of the executions in a human-readable style.

This transformation is achieved by executing a genetic algorithm that generates possible solutions to the problem and a greedy algorithm that carries out the assignments and ranks the solution. At the end of the execution the solution ranked with the highest value will be returned.

Optionally, the program accepts as input the output CSV of a previous execution and also a CSV of classroom preferences for the assignments, which assists the algorithm in ranking the solutions.

Finally, there is also a configuration file that controls the parameters of the genetic algorithm, the version of the program, and the folder path for the log and output of the system.

The main advantage of the system is its great flexibility. By focusing more on the input data than on the code, it has been possible to create a tool whose functionality adapts to the CSVs built by the user.

Use cases in the middle of the semester, like adding new groups and designating their classrooms, removing laboratories due to renovation works, finding classrooms for specific courses that only last one month, finding classrooms for exams that only happen in a week, etc. are all very easy to carry out. The user just needs to change the input CSVs in the correct way and then introduce as input the output of the execution conducted at the beginning of the semester, and the program will take the current assignments into account when allocating classsrooms for these new groups.

1.9 Risk analysis

1.10 Time planning

1.11 Budget summary

1.12 Prioritisation of project documents

2 Introduction

2.1 Project justification

2.2 Project goals

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 some 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 tool, the author of this project was given the task of automating the assignment of classrooms to subject groups.

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

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

9.2 Programming languages

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

TODO

16 Annexes

16.1 Definitions and abbreviations

16.2 Submission contents

17 Source code