0 Index

1 Overview

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.

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 Redondo. However, some additional chapters were introduced in order to capture the particularity of the work carried out, inspired by the research of de la Cruz [2].

Report. This is the chapter where I explain the personal motivation to take on this project, as well as give a brief explanation on the nature of the project and what it covers.

Introduction. Here we explain in a simple way the problem we want to solve, what reasons are behind the development of the project and give a description of the current situation of the School with regard to this and other similar problems.

Theoretical aspects. The first chapter delving into the theory supporting the developed system. One example of an assignment problem is presented and solved using a greedy algorithm and a genetic algorithm, which helps to better internalize the concepts.

Problem definition. Here the formulation of the problem as an assignment problem is elaborated.

Proposed solution. This chapter lays down in detail the proposed solution to the previously defined assignment problem by means of a genetic and greedy algorithms.

Project planning and budget overview. For the planning, the Gantt chart of the project is shown, as well as the work breakdown structure. The internal and client budgets are presented, but not how they were calculated. That goes in the budget section.

Analysis. An analysis of the system, with the system requirements, draft diagrams, use cases and the test plan specification.

System design. The technical details of the system analysed in the previous section. Here we include the finalised diagrams, as well as the arquitecture of the system and the in-depth test plan.

System implementation. Details of the development of the software. The programming languages, standards and tools used to code the system, and all the relevant information gathered in the process of creating the system.

Test development. A rundown of all the testing done for the system, with explanations for every test and the obtained results.

Experimental results. The conclusions reached after experimenting with different input data and the optimal configuration for the genetic algorithm's parameters.

System manuals. All the manuals for the system, with screenshots and guided steps meant to help the target audience for each document.

Conclusions and future work. The conclusions after the implementation of the project are given, as well as a list of possible improvements and new functionalities for the prototype.

Budget. Here we give the full details for the elaboration of the internal and client budgets, with all the intermediate steps that led us to that result.

Annexes. The glossary of definitions and abbreviations and a small commentary on the submitted files.

Source code. The *javadoc* of the software developed.

2 Report

2.1 Motivation

> First, solve the problem. Then, write the code.
>
> *John Johnson*

One of the things I learned in my university years is that when you are presented with a coding problem, be it an exam or a project, is good to step back and try to solve it on paper. This project embodies that sentiment.

This is a problem you need to solve *before* writing the code, first of all, because the code is a *prototype* of the proposed solution, and all its elements *depend* on the algorithm in the first place. Input data? Data structures? Good luck with those if you start typing away without thinking.

However, even if you do solve the problem, there is no guarantee that it will work in practice. Maybe the computations are so complex that the time the program takes to solve the problem makes it unusable, or you simply cannot give useful solutions without creating a lot of conflicting assignments. This uncertainty factor is undoubtedly a characteristic of the project that sets it apart from others I have previously undertaken.

Lastly, it is important to mention that the success of this project will make someone's job easier, so prototype or not, the system needs to work and must have as the primary goal the experience of the user.

2.2 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 4. 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.

2.3 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 2.2 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.

3 Introduction

3.1 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.

3.2 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.

3.3 Situation overview

3.3.1 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, I was given the task of automating the assignment of classrooms to subject groups by similar means.

3.3.2 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.

4 Theoretical aspects

4.1 Assignment problem

Imagine that a digital magazine tasked you with the following problem. They need to assign book reviews to freelance writers, and they want the assignments to result in the maximum profit. Therefore, we have:

A set of n freelancers f

A set of m book reviews r

A set of $n \times m$ assignments a_{fr} such that $a_{fr} = 0$ when freelancer f is not assigned to book review r and $a_{fr} = 0$ when freelancer f is assigned to book review r.

A set of $n \times m$ profits p_{fr} which indicate the profit obtained when assigning freelancer f to book review r and that $p_{fr} > 0$.

A valid solution is defined as a set of assignments where all the book reviews have a freelancer assigned to them and no book review has more than one associated freelancer.

The profit for all the assignments will then be:

$$\sum_{f=1}^{n} \sum_{r=1}^{m} a_{fr} p_{fr} \tag{4.1}$$

The optimal solution consists on having a set of assignments such that the sum of all the profits for the current assignments is maximised.

For example, let us say that we have two freelancers $f1$ and $f2$ and three book reviews $r1$, $r2$ and $r3$. We can represent this data with two sets F and R, respectively.

$$F = \{f1, f2\} \tag{4.2}$$

$$R = \{r1, r2, r3\} \tag{4.3}$$

Then, our assignments and profits will be represented by the sets A and P.

$$A = \{a11, a12, a13, a21, a22, a23\} \tag{4.4}$$
$$P = \{p11, p12, p13, p21, p22, p23\} \tag{4.5}$$

Now, we are going to study valid and non-valid solutions. As we explained before, a solution is valid if every book review has a freelancer assigned to it, and no more than one.

We will analyse four sets of values for the A set:

$$A1 = \{0, 0, 0, 1, 1, 1\} \tag{4.6}$$
$$A2 = \{0, 1, 0, 1, 0, 1\} \tag{4.7}$$
$$A3 = \{0, 0, 0, 1, 0, 1\} \tag{4.8}$$
$$A4 = \{0, 1, 0, 1, 1, 1\} \tag{4.9}$$

From these sets, we can deduce that $A1$ and $A2$ are valid solutions, because they have one freelancer for each book review. We are not concerned with a freelancer having no book reviews assigned. However, a book review without an associated freelancer represents a non-valid solution. That is precisely the case for $A3$, the book review $r2$ has not an assigned freelancer. In the case of $A4$, the fact that $r2$ has two freelancers assigned to it makes it a non-valid solution.

Now, we will give values to the P set in order to discuss possible optimal solutions. We will compare them with the assignment sets $A1$ and $A2$

$$P1 = \{3, 1, 6, 10, 2, 15\} \tag{4.10}$$
$$P2 = \{3, 7, 6, 10, 2, 15\} \tag{4.11}$$

$P1$ y $A1$:

$$Profit = \sum_{f=1}^{n} \sum_{r=1}^{m} a_{fr} p_{fr} = 1 \times 10 + 1 \times 2 + 1 \times 15 = 27 \quad (4.12)$$

$P1$ y $A2$:

$$Profit = \sum_{f=1}^{n} \sum_{r=1}^{m} a_{fr} p_{fr} = 1 \times 1 + 1 \times 10 + 1 \times 15 = 26 \quad (4.13)$$

We can observe that for $P1$, the set of assignments $A1$ is better than $A2$, because it results in a better profit. Another important thing about $A1$ is that it is the optimal solution to the problem, because it assigns the book reviews to the freelancers with the better profit value. Now $P2$ will be evaluated.

$P2$ y $A1$:

$$Profit = \sum_{f=1}^{n} \sum_{r=1}^{m} a_{fr} p_{fr} = 1 \times 10 + 1 \times 2 + 1 \times 15 = 27 \quad (4.14)$$

$P2$ y $A2$:

$$Profit = \sum_{f=1}^{n} \sum_{r=1}^{m} a_{fr} p_{fr} = 1 \times 7 + 1 \times 10 + 1 \times 15 = 32 \quad (4.15)$$

In the case of the profits values of $P2$, the situation is reversed. $A2$ is now the optimal solution and therefore better than $A1$.

The important thing to notice here is that the values for the P set right now may appear as having no meaning whatsoever. But if we think of P as a profict funtion, maybe we can interpret $P1$ as values of profit in a context where freelancer $f1$ has just tweeted an opinion on the book related with the review $r2$ and caused a massive controversy. We can then say that $P1$ is a function which gives more importance to public relations and so the profit p_{12} is very low, whereas $P2$ gives more importance to clickbait and so the profit p_{12} is higher! Of course in a real problem you know what the function is calculating, but for this example it is left to the imagination of the reader.

With this example of an assignment problem we have seen its main characteristics and worked around with it to some extent. We will return to it time and time again as we follow through this chapter.

4.2 Heuristics

4.3 Greedy algorithm

4.4 Metaheuristics

4.5 Genetic algorithm

5 Problem definition

6 Proposed solution

7 Project planning and budget overview

7.1 Planning

7.2 Budget summary

8 Analysis

8.1 System definition

8.2 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.

8.2.1 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.

8.2.2 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).

8.2.3 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.

8.2.4 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).

8.3 Subsystem mapping

8.4 Preliminary class diagram

8.5 Analysis of use cases

8.6 Analysis of user interfaces

8.7 Test plan specification

9 System design

9.1 System architecture

9.2 Class design

9.3 Interaction and state diagrams

9.4 Activity diagram

9.5 Interface design

9.6 Technical specification of the test plan

10 System implementation

10.1 Standards and references

10.1.1 Standards

10.1.2 Licenses

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

10.1.3 Other references

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

10.2 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.

10.3 Tools and programs used in development

10.4 System development

11 Test development

11.1 Unit tests

11.2 Integration and system tests

11.3 Usability and accessibility tests

11.4 Performance tests

12 Experimental results

13 System manuals

13.1 Installation manual

13.2 Execution manual

13.3 User manual

13.4 Programmer manual

14 Conclusions and future work

14.1 Final conclusions

14.2 Future work

15 Budget

15.1 Internal budget

15.2 Client budget

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

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.