UNIVERSIDAD DE OVIEDO

ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

"MEJORAS EN LA ASIGNACIÓN DE ESTUDIANTES A GRUPOS DE CLASE EN LA ESCUELA DE INGENIERÍA INFORMÁTICA MEDIANTE MÉTODOS DE INTELIGENCIA ARTIFICIAL"

TUTORES: CARLOS MENCÍA CASCALLANA
FERNANDO ÁLVAREZ GARCÍA

AUTORA: INÉS ANDRÉS SANTOS

Agradecimientos

Me gustaría agradecer en primer lugar, tanto a mis tutores Carlos y Fernando, como al grupo de investigación iScOp, ya que han sido la base para que este trabajo se haya podido realizar y gracias a ellos he aprendido mucho.

Agradecer a la Universidad por todo. Me ha puesto en el camino a Elmer y Sara, que habéis conseguido que lo bueno haya sido aún más bueno y lo malo lo lleve mejor. Algún día será en un brunch. Además de ellos, la Universidad me ha regalado la mejor persona que podía imaginar, Fernando, que ha sido mi mayor apoyo siempre. Simplemente por ti los 5 años de carrera han merecido la pena.

Gracias a mis amigas y amigos, que llevan al lado toda la vida y han sufrido todas mis subidas y bajadas en la carrera. A vosotras os digo que algún día nos tatuaremos 33008, y a vosotros que ojala sigamos jugando a juegos de mesa muchos años más.

Por último, agradecer a mi familia, que me ha ayudado a crecer. Jovita, Alba y Carmen, me entendéis hasta cuando ni yo me entiendo. A mis padres, que ha sido un año realmente malo pero siempre han tenido una buena cara. Espero algún día devolveros todo lo bueno que hacéis por mi, pero creo que necesitaría otra vida más.

Resumen

El proyecto de investigación tiene como objetivo encontrar una solución para las mejoras requeridas en el sistema que se usa actualmente en la Escuela de Ingeniería Informática de la Universidad de Oviedo para asignar a los alumnos a los grupos de clase.

El problema a resolver plantea ciertas consideraciones con las que debe contar el sistema. Se requiere que el sistema permita respetar asignaciones iniciales introducidas por el usuario, así como indicar asignaciones que no es necesario realizar. Además, debe manejar restricciones y preferencias horarias de los alumnos. Asimismo, el sistema debe contar con una funcionalidad de asignación individual para un único alumno, que proporcione varias posibles soluciones a su asignación.

En primer lugar, se describe el sistema actual, para después definir el problema a resolver. Se propone una división del sistema en dos modos, uno para la asignación de todos los alumnos matriculados, y otro para la asignación de un único alumno. El primer modo se integra con el sistema actual, modificando el algoritmo genético y el algoritmo voraz implementados en la versión anterior de la herramienta, y en el segundo se propone un algoritmo de backtracking. Asimismo, el modo individual va a contar con dos modos: modo determinista, en el que las soluciones son parecidas y modo aleatorio, en el que las soluciones son diferentes.

Se han realizado experimentos con datos de matrículas reales pero anonimizadas, proporcionados por la escuela, y con archivos de entrada generados de forma aleatoria para analizar cómo afecta el uso de las nuevas funcionalidades a la calidad de la solución.

Palabras clave

Metaheurísticos, Algoritmo Voraz, Algoritmo Genético, Algoritmo de Backtracking, asignaciones iniciales, asignaciones que no son necesarias, restricciones horarias, preferencias horarias, modo grupal, modo individual.

Abstract

The research project aims to find a solution for the improvements required in the system currently used by the School of Computer Engineering of the University of Oviedo to assign students to class groups.

The problem to be solved raises certain considerations that the system must take into account. the system is required to allow the fulfillment of initial assignments introduced by the user, as well as to indicate assignments that do not need to be made. In addition, it must handle time restrictions and preferences of students. Also, the system must have an individual assignment functionality for a single student, which gives several possible solutions to his or her assignment.

First, the current system is described, and then the problem to be solved is defined. A division of the system into two modes is proposed, one for the assignment of all the students, and one for the assignment of a single student. The first mode is integrated with the current system, modifying the genetic algorithm and the greedy algorithm implemented in the previous version of the tool, and in the second a backtracking algorithm is proposed. Likewise, the individual mode will have two modes: deterministic mode, in which the solutions are similar, and random mode, in which the solutions are different.

Experiments have been carried out with real but anonymised enrolment data provided by the school and with randomly generated input files to analyse how the use of these new functionalities affects the quality of the solution.

Keywords

Metaheuristics, Greedy Algorithms, Genetic Algorithms, Backtracking Algorithms, initial assignments, assignments that are not required, temporal restrictions, temporal preferences, individual mode, group mode.

Índice general

1. Memoria del proyecto **12**
- 1.1. Motivación . . . 12
- 1.2. Objetivos . . . 12
- 1.3. Alcance . . . 13
- 1.4. Resumen de todos los aspectos . . . 13

2. Introducción **14**
- 2.1. Justificación del proyecto . . . 14
- 2.2. Objetivos del proyecto . . . 15
- 2.3. Estudio de la situación actual . . . 15

3. Aspectos teóricos **17**
- 3.1. Algoritmos Voraces . . . 17
- 3.2. Metaheurísticas . . . 18
- 3.3. Algoritmos genéticos . . . 19
  - 3.3.1. Operadores . . . 20
- 3.4. Backtracking . . . 22

4. Definición del problema **24**
- 4.1. Motivación . . . 24
- 4.2. Descripción del problema . . . 25
- 4.3. Definición formal del problema . . . 26
  - 4.3.1. Subproblema 1 . . . 26
  - 4.3.2. Subproblema 2 . . . 29

5. Solución propuesta **30**
- 5.1. Espacio de búsqueda . . . 30
  - 5.1.1. Concepto de asignación . . . 30
  - 5.1.2. Concepto de restricción horaria . . . 31
  - 5.1.3. Concepto de preferencias horarias . . . 31
  - 5.1.4. Asignaciones a todos los alumnos (Subproblema 1) . . . 31
  - 5.1.5. Asignación a un único alumno (Subproblema 2) . . . 33
- 5.2. Matriz de colisiones . . . 33
  - 5.2.1. Excepción de colisión . . . 34

5.2.2. Colisión obligatoria 34
5.3. Algoritmo Voraz 34
5.3.1. Asignación inicial 35
5.3.2. Preprocesamiento 36
5.3.3. Heurístico 36
5.3.4. Proceso de reparación 37
5.3.5. Preferencias de asignación 39
5.4. Algoritmo Genético 40
5.4.1. Codificación - Decodificación 42
5.4.2. Operadores 43
5.4.3. Fitness 44
5.5. Backtracking 46
5.5.1. Creación del individuo 47
5.5.2. Ejecución backtracking 47

6. Planificación del proyecto y resumen de presupuesto 51
6.1. Planificación temporal 51
6.2. Resumen del presupuesto 58

7. Análisis 59
7.1. Definición del sistema 59
7.1.1. Alcance del sistema 59
7.2. Requisitos del sistema 60
7.2.1. Requisitos funcionales 60
7.2.2. Requisitos no funcionales 78
7.2.3. Actores del sistema 78
7.3. Identificación de los subsistemas en la fase de análisis 78
7.3.1. Descripción de los Subsistemas 78
7.4. Diagrama de clases preliminar del análisis 79
7.4.1. Diagrama de Clases 79
7.4.2. Descripción de las Clases 81
7.5. Análisis de casos de uso y escenarios 88
7.5.1. Modo Grupal 88
7.5.2. Modo Individual 94
7.6. Análisis de interfaces de usuario 99

8. Diseño del sistema 100
8.1. Arquitectura del Sistema 100
8.1.1. Diagrama de paquetes 100
8.1.2. Diagrama de componentes 102
8.2. Diseño de las clases 103
8.2.1. Diagrama de las clases del sistema 105
8.3. Diseño Interfaz de Usuario 110
8.3.1. Descripción de la Interfaz 110
8.3.2. Descripción del comportamiento de la interfaz 111

9. Implementación del sistema **118**
9.1. Lenguaje de programación 118
9.2. Herramientas de desarrollo 119
9.2.1. IntelliJ IDEA 119
9.2.2. Git 119
9.2.3. Gradle 119
9.3. Creación del sistema 119
9.3.1. Problemas encontrados 119
9.3.2. Descripción detallada de las clases 120

10. Pruebas del sistema **121**
10.1. Pruebas de validación de los archivos de entrada y de la salida . 121
10.2. Pruebas de integración 126
10.3. Diseño de los experimentos 128

11. Resultados experimentales **129**
11.1. Modo grupal 129
11.1.1. Instancias del problema 129
11.1.2. Mejoras del modo grupal 130
11.1.3. Peso de las preferencias 139
11.2. Modo individual 144
11.2.1. Matricula simple 144
11.2.2. Matrícula compleja 144
11.2.3. Matrícula imposible 145

12. Manuales del sistema **146**
12.1. Manual de ejecución 146
12.1.1. Ejecución 146
12.1.2. Detención 146
12.2. Manual de usuario 147
12.2.1. Requisitos del sistema 147
12.2.2. Ejecutar el sistema 147
12.2.3. Ficheros del sistema 148
12.2.4. Ficheros de salida 155

13. Conclusiones y ampliaciones **159**
13.1. Conclusiones 159
13.2. Ampliaciones 159

14. Presupuesto **161**
14.1. Definición de la empresa 161
14.1.1. Personal 161
14.1.2. Productividad del personal 161
14.1.3. Costes indirectos 162
14.1.4. Costes de los medios de producción 162
14.1.5. Horas productivas y no productivas 163

14.1.6. Precio hora 163
14.1.7. Resumen 163
14.2. Descripción del proyecto 164
14.3. Presupuesto de costes 164
14.3.1. Partida 1: Planificación 165
14.3.2. Partida 2: Revisión bibliográfica 165
14.3.3. Partida 3: Análisis 165
14.3.4. Partida 4: Implementación 166
14.3.5. Partida 5: Pruebas 166
14.3.6. Partida 6: Validación del cliente 167
14.3.7. Partida 7: Experimentos 167
14.3.8. Partida 8: Cierre del proyecto 167
14.3.9. Agregación final del presupuesto de costes 167
14.4. Presupuesto del cliente 168
14.4.1. Presupuesto del cliente resumido 169

A. Contenido como anexo a la memoria **171**
A.1. Javadoc (javadoc.zip) 171
A.2. Instancias (Instancias.zip) 171
A.3. Código fuente (assignStudents.zip) 171
A.4. Ejecutable (Ejecutable.zip) 172

Índice de figuras

3.1. Cruce en un punto 21
3.2. Operador de cruce OX 21
3.3. Mutación simple 22

6.1. Planificación de la primera parte del proyecto. 55
6.2. Planificación de la segunda parte del proyecto. 56
6.3. Planificación de la tercera parte del proyecto. 57
6.4. Resumen presupuesto. 58

7.1. Diagrama preliminar de clases. 80

8.1. Diagrama de paquetes. 101
8.2. Diagrama de componentes. 103
8.3. Diagrama de clases del sistema 104
8.4. Diagrama de clases del paquete Model 105
8.5. Diagrama de clases del paquete Parser 106
8.6. Diagrama de clases del paquete Reporter 106
8.7. Diagrama de clases del paquete Backtracking 107
8.8. Diagrama de clases del paquete Greedy 108
8.9. Diagrama de clases del paquete Genetic 109
8.10. Diagrama de clases del paquete Program 110
8.11. Modo grupal: gráfica del fitness. 112
8.12. Modo grupal: gráfica de las colisiones. 113
8.13. Modo grupal: gráfica de la varianza. 114
8.14. Modo grupal: gráfica de las preferencias horarias. 115
8.15. Modo grupal: información de la generación 116
8.16. Modo individual: información de la solución 117

11.1. Asignaciones iniciales: asignaciones sin resolver 132
11.2. Asignaciones iniciales: homogeneidad 132
11.3. Asignaciones iniciales: calidad del horario del alumno 133
11.4. Restricciones horarias: asignaciones sin resolver. 134
11.5. Restricciones horarias: homogeneidad. 135
11.6. Restricciones horarias: calidad del horario. 135

11.7. Restricciones horarias: fitness. 136
11.8. Preferencias horarias: asignaciones sin resolver. 137
11.9. Preferencias horarias: homogeneidad. 137
11.10Preferencias horarias: calidad del horario. 138
11.115 % alumnos con preferencias horarias: asignaciones sin resolver. . 140
11.125 % alumnos con preferencias horarias: homogeneidad. 140
11.135 % alumnos con preferencias horarias: calidad del horario. 141
11.1420 % alumnos con preferencias horarias: asignaciones sin resolver. 142
11.1520 % alumnos con preferencias horarias: homogeneidad. 142
11.1620 % alumnos con preferencias horarias: calidad del horario. . . . 143

12.1. Fichero de matrícula de los alumnos. 149
12.2. Fichero del horario escolar. 150
12.3. Fichero de asignaciones iniciales. 150
12.4. Fichero de asignaciones que no son necesarias 151
12.5. Fichero de las restricciones horarias de los alumnos. 151
12.6. Fichero de excepciones de colisiones. 152
12.7. Fichero de colisiones obligatorias. 152
12.8. Fichero de preferencias de asignación. 153
12.9. Fichero de las abreviaturas de las asignaturas. 153
12.10Fichero de las propiedades del Algoritmo Genético. 154
12.11Fichero de las propiedades del Algoritmo de Backtracking. 155
12.12Fichero de resumen de la solución generada. 156
12.13Fichero con las asignaciones que no se han podido realizar. . . . 156
12.14Fichero con el horario del alumno. 157
12.15Fichero con las asignaciones del alumno. 157
12.16Fichero con la distribución de los grupos de la asignatura. 158

14.1. Personal. 161
14.2. Productividad del personal. 162
14.3. Costes indirectos. 162
14.4. Medios de producción. 162
14.5. Horas productivas y no productivas 163
14.6. Precio hora. 163
14.7. Resumen empresa. 164
14.8. Partida 1: Planificación 165
14.9. Partida 2: Revisión bibliográfica 165
14.10Partida 3: Análisis 165
14.11Partida 4: Implementación 166
14.12Partida 5: Pruebas 166
14.13Partida 6: Validación del cliente 167
14.14Partida 7: Experimentos 167
14.15Partida 8: Cierre del proyecto 167
14.16Resumen. 168
14.17Presupuesto cliente detallado. 169
14.18Presupuesto cliente resumido. 170

Capítulo 1

Memoria del proyecto

1.1. Motivación

Desde pequeña siempre me ha interesado la investigación. Mis primeras opciones cuando pensaba en un futuro siempre eran enfocadas en la investigación en matemáticas, física o informática pero todo en dirección a la investigación, ya que era algo que me atraía mucho. Recuerdo la primera charla que asistí de la facultad, organizada por la Delegación de Estudiantes, en la que se introducía a la investigación de forma general, y cómo ese año tuve mi primer contacto con los algoritmos. Fue ahí, cuando, además, en una asignatura, se nos explicó además la inteligencia artificial y mi interés aumentó. Por esa razón, en el momento en el que había que pensar en el trabajo de fin de grado, supe que quería que fuera enfocado en ese campo, ya que era demás algo que me iba a enseñar mucho.

En 2018, en la Escuela de Ingeniería Informática de Oviedo, Gonzalo De La Cruz Fernández realizó un Trabajo de Fin de Grado [1] sobre la asignación de los alumnos a sus clases, que ha sido puesto en práctica a lo largo de los años de forma efectiva. Durante este uso, se han identificado una serie de mejoras que ayudarían mucho para facilitar el trabajo, y fue por esto por lo que se presentó y aprobó una propuesta de proyecto de fin de carrera.

1.2. Objetivos

Como objetivo principal de este proyecto tenemos el estudio de la herramienta utilizada actualmente en la asignación y realización de unas extensiones y mejoras ajustándose a los algoritmos elegidos anteriormente. Las mejoras son definidas por el cliente, que es la persona que está usando la propuesta en [1], y que cuenta con un conocimiento más profundo sobre éste.

Una vez realizado lo anterior, el siguiente paso consiste en estudiar las soluciones a cada una de las mejoras propuestas anteriormente e integrar las solu-

ciones en la versión anterior de la herramienta, de modo que conserve todas las funcionalidades anteriores.

1.3. Alcance

Como se ha explicado en el apartado anterior, el objetivo es realizar un estudio de las funcionalidades a añadir y solucionar el problema. Se ha realizado con la aplicación realizada por Gonzalo de aplicación de consola, para ser utilizado por la Escuela de Ingeniería Informática de Oviedo, como ha sido usada anteriormente. Esta aplicación será configurada mediante ficheros de texto.

1.4. Resumen de todos los aspectos

Inicialmente se realiza un análisis de las funcionalidades a añadir, que nos ha descrito el cliente. Es necesario en este proceso, además, realizar una revisión bibliográfica exhaustiva del sistema inicial, para entender cómo funciona y poder integrar las funcionalidad de manera eficiente, para que el sistema se comporte igual si no se utilizan.

Posteriormente se realizará un diseño de cómo se integran, y una vez que se haya diseñado, se implementará modificando el software diseñado en el sistema inicial, de forma que si en un futuro se necesita introducir nuevas funcionalidades se puede realizar más fácilmente.

Se realizarán experimentos con datos de matrículas de alumnos reales y horario de clases que han sido dados por la Escuela de Ingeniería Informática. Además, será necesario crear instancias para poder experimentar con las nuevas funcionalidades, y así comprobar si su uso es conveniente y cómo afecta a la calidad de la solución.

Capítulo 2

Introducción

2.1. Justificación del proyecto

Desde hace unos años, la asignación de los alumnos a grupos de clase se ha realizado con una herramienta diseñada e implementada en el Trabajo de Fin de Grado de Gonzalo De La Cruz Fernández [1], consiguiendo así que este proceso se realice de forma automatizada de tal forma que, el responsable de utilizar este programa, podía ejecutarlo y encontrarse con el problema prácticamente resuelto, teniendo que revisarlo y posiblemente, teniendo que realizar algunos cambios a mano. Para llevar a cabo esta herramienta se tuvieron en cuenta dos puntos importantes: que se realicen el mayor número de asignaciones posibles y la homogeneidad en el tamaño de los grupos. Además de esto, un punto que también tiene peso, es la calidad del horario desde el punto de vista de los estudiantes. Es decir, con calidad se hace referencia a que el alumno tenga un horario, con las menos horas libres posibles y los menores días con pocas horas de clase, mientras que la homogeneidad hace referencia a que los grupos estén equilibrados en referencia al número de alumnos.

Como se dijo anteriormente, se tuvieron dos puntos en cuenta importantes, pero se ha dejado de lado otros que a lo largo del tiempo el responsable ha visto que son necesarios, como por ejemplo que permita partir de asignaciones iniciales que hay que respetar. Además de esto, en algunas asignaturas, los alumnos que repiten tienen la posibilidad de guardar la parte aprobada para el siguiente curso, no teniendo que realizar una asignación a ese tipo de clase de dicha asignatura. Por otra parte, puede haber alumnos con trabajo u otras actividades justificadas, por lo que asignarles a un grupo al que no van a poder asistir es un problema para ellos, y para el responsable, que tendría que realizar cambios de grupo. Por último, sería conveniente tener en cuenta la opinión del alumno respecto a preferencias horarias en las que se les asignen grupos de clase, si las tuviera, para intentar agradar al máximo posible de alumnos en sus asignaciones.

El programa desarrollado en [1] además funciona con un gran número de alumnos y asignaturas, utilizando algoritmos evolutivos para conseguir la mejor solución posible. Este es un problema cuando se quiere asignar asignaturas a un único alumno, ya sea porque su matrícula se hizo más tarde, Erasmus, etc… Sería útil que la herramienta diera varias soluciones posibles para que el responsable vea las posibilidades y lo pueda discutir con el alumnado.

Este proyecto nace con el objetivo de intentar ampliar el programa de Gonzalo De La Cruz Fernández [1], para así conseguir una solución que se adapte a los requisitos dados y además que sea eficiente, conservando las funcionalidades existentes.

2.2. Objetivos del proyecto

El objetivo de este proyecto es investigar cómo se podría mejorar la herramienta que actualmente se está usando en la Escuela de Ingeniería Informática de la Universidad de Oviedo. Para ello, durante este proyecto se realiza un estudio exhaustivo de la herramienta actual. Es necesario llevar a cabo un estudio de la solución que se ha implementado en la herramienta, teniendo en cuenta qué algoritmos se han utilizado para poder realizar una estrategia inteligente para extender esa herramienta.

Tras realizar el estudio de la herramienta y de los algoritmos utilizados, en el siguiente paso es primordial la ayuda del encargado en la escuela para que exponga los problemas a resolver. Tras esto, se diseñará una solución para cada uno de ellos, de forma que se integre con el diseño actual de la herramienta.

Se implementará el prototipo de las mejoras hechas, para así poder evaluar el resultado y su calidad. Ese prototipo será implementado teniendo en cuenta los factores que se han tenido para la herramienta actual, tanto los horarios de las asignaturas y matrículas de alumnos como la homogeneidad de los grupos.

El objetivo es que este prototipo pueda ser utilizado por la Escuela de Ingeniería Informática de Oviedo, facilitando en la mayor medida posible la tarea de asignar grupos de clase a los alumnos matriculados.

2.3. Estudio de la situación actual

Actualmente, como se ha mencionado, la Escuela de Ingeniería Informática de la Universidad de Oviedo dispone de una herramienta, diseñada en un trabajo de fin de grado [1], para la asignación de los alumnos a los grupos de clase. Esta herramienta recibe del usuario los horarios de las clases, los alumnos matriculados y unos ficheros de configuración que contienen información sobre colisiones (o solapamientos en el tiempo) entre asignaturas o preferencias.

La herramienta realiza las asignaciones empleando un algoritmo genético teniendo en cuenta que se desea tener grupos homogéneos, que existen colisiones entre clases, y que el alumno tenga las menores horas libres y los menores días con pocas horas posibles. El algoritmo tarda unas horas aunque depende mucho de la cantidad de alumnos que haya.

Como la herramienta solo tiene en cuenta las clases, los alumnos, y ciertas configuraciones, se aleja de algunas cuestiones que afectan al horario de forma indirecta, como las preferencias horarias o restricciones que puede tener un alumno, o la necesidad de no tener que realizarle la asignación en un tipo de asignatura, ya sea por tener una asignación ya hecha anteriormente o porque, por ejemplo, se le guarda la nota del año anterior.

El objetivo de este proyecto es intentar integrar en la herramienta actual las cuestiones nombradas anteriormente, ya que la herramienta trabaja de manera eficiente, dando buenos resultados pero, después de ejecutarlo, hay que realizar modificaciones porque no se tienen en cuenta estos factores. Además, esta herramienta no tiene la posibilidad de dar varias posibilidades de horario para un único alumno.

Capítulo 3

Aspectos teóricos

3.1. Algoritmos Voraces

El Algoritmo Voraz o Greedy Algorithm es un algoritmo fácil de diseñar y normalmente tiene una menor complejidad en comparación con algoritmos exactos aunque suelen proporcionar soluciones de peor calidad. Este algoritmo empieza por una solución vacía y se va construyendo la solución asignándole un valor a cada variable de decisión en cada momento, hasta que la solución completa esté generada.

Como viene descrito en el libro [2], una solución puede venir definida por la presencia o ausencia de un conjunto finito de elementos, que se puede definir como $E = \{e_1, e_2, ..., e_n\}$. La función objetivo se puede definir como $f : 2^E \to \mathbb{R}$ y el espacio de búsqueda se define como $F \subset 2^E$. Una solución parcial s puede verse como un subconjunto $\{e_1, e_2, ..., e_k\}$ de elementos e_i del conjunto de elementos E.

Como se dijo anteriormente, parte de una solución vacía y en cada paso un heurístico local selecciona un nuevo elemento para ser incluido en la solución y no debería ser sustituido por otro, ya que este tipo de algoritmos son típicamente deterministas. Para el diseño de estos algoritmos hay que tener en cuenta dos elementos principales.

- El conjunto de elementos: para un problema dado, es necesario identificar una solución como un conjunto de elementos para que así, las soluciones parciales manipuladas puedan verse como subconjuntos de elementos.
- El heurístico de selección de elementos: En cada paso del algoritmo, se utiliza un heurístico para seleccionar el siguiente elemento que formará parte de la solución, de tal forma que se minimiza localmente la función objetivo. De este modo, el heurístico calculará el *profit* para cada elemento y seleccionará el mejor. Sin embargo, la optimización local no implica una optimización global.

Un pseudocódigo se encuentra en el algoritmo 1.

Algorithm 1 Algoritmo Voraz

```
1: s = ∅
2: while s=∅ and !solution() do
3:     e_i= Heuristic(E) ▷ Siguiente elemento del conjunto E que no haya sido seleccionado
4:     if s ∪e_i then
5:         s = s ∪e_i
6:     end if
7: end while
8: if s ∪e_i then
9:     return s
10: else
11:     return ∅
12: end if
```

3.2. Metaheurísticas

Las metaheurísticas representan una familia de técnicas de optimización aproximada que ha ganado mucha popularidad en las últimas décadas ya que proporcionan soluciones aceptables en un tiempo razonable para resolver problemas difíciles y complejos en ciencia e ingeniería. El concepto de heurística para resolver problemas de optimización fue introducido por Polya en 1945.

A diferencia de los algoritmos exactos, las metaheurísticas no garantiza que las soluciones encontradas son óptimas ni lo cerca que están las soluciones obtenidas de las óptimas, pero a diferencia de los métodos exactos, permiten abordar casos de problemas de gran tamaño ofreciendo soluciones satisfactorias en un tiempo razonable.

Las metaheurísticas han recibido cada vez más popularidad en los últimos años. Su uso en muchas aplicaciones demuestra su eficiencia y eficacia para resolver problemas grandes y complejos. La aplicación de las metaheurísticas abarca un gran número de áreas. Algunas de ellas son: planificación, optimización de topologías u optimización estructural en aerodinámica entre otras. Se puede encontrar información detallada sobre las metaheurísticas en [2].

Como los problemas de optimización suelen ser complejos y además, la optimización se encuentra en numerosos ámbitos, es normal que en consecuencia las metaheurísticas sea ampliamente utilizadas, por lo que es entendible que el número de sesiones, talleres y conferencias que tratan sobre las metaheurísticas haya crecido considerablemente.

3.3. Algoritmos genéticos

Los Algoritmos Genéticos, o Genetic Algorithms fueron introducidos por John Holland y sus colaboradores de la Universidad de Michigan a principios de los años 70 [3], y son los algoritmos más populares dentro de los Algoritmos Evolutivos.

Son algoritmos de optimización y búsqueda que se basan en los mecanismos de la evolución natural, en particular la selección natural y la herencia genética y se clasifican dentro de las denominadas metaheurísticas. Se puede ver más información en [2], y además se encuentra un tutorial en [4]. En la literatura se han propuesto algoritmo genético para resolver problemas de planificación en el ámbito académico relacionados con el que se resuelve en este trabajo de fin de grado, como en los artículos [5], [6], [7] y [8].

En los Algoritmos Genéticos, el término cromosoma se utiliza normalmente para referirse a una solución candidata a un problema, que suele estar codificada. Un cromosoma está compuesto por genes que son los parámetros individuales que componen una solución.

Un pseudocódigo de la estructura general de los Algoritmo Genético se puede ver en el algoritmo 2. Como se puede observar, el algoritmo genético maneja

Algorithm 2 Algoritmo Genético

1: $Poblacion = inicializarPoblacion(Poblacion_{size})$
2: $EvaluarPoblacion(Poblacion)$
3: $S_{best} = conseguirLaMejorSolucion(Poblacion)$
4: **while** not $CriterioTerminar()$ **do**
5: $Padres = seleccionarPadres(Poblacion, Poblacion_{size})$
6: $Hijos =$
7: **for** $Padre_1, Padre_2 \in Padres$ **do**
8: $\{Hijo_1, Hijo_2\} = cruce(Padre_1, Padre_2, p_c)$
9: $mutacion(Hijo_1, p_m)$
10: $mutacion(Hijo_2, p_m)$
11: $Hijo = Hijo \cup \{Hijo_1, Hijo_2\}$
12: **end for**
13: $EvaluarPoblacion(Hijo)$
14: $Poblacion = remplazar(Poblacion, Hijo)$
15: $S_{best} = conseguirLaMejorSolucion(Poblacion)$
16: **end while** **Output:**S_{best}

una población que representan soluciones del probelma. Este recibe como parámetros el tamaño de la población ($Poblacion_{size}$), la probabilidad de cruce (p_c) y la probabilidad de mutación (p_m). Inicialmente, genera una población inicial y la evalúa, asignando a cada individuo una medida de fitness, que mide la calidad

de la solución que representa. Posteriormente, de un modo iterativo la población evoluciona mediante los operadores genéticos de selección, cruce, mutación y remplazamiento. El proceso finaliza cuando se cumple el criterio de terminación que suele ser completar un número de generaciones introducido como parámetro, retornando la mejor solución.

3.3.1. Operadores

Selección

El operador de selección es uno de los principales componentes de los algoritmos evolutivos. Se encarga de determinar qué individuos serán seleccionados para hacer el rol de padres en la siguiente generación y cuántos descendientes genera cada individuo. En general, cuanto más apto sea el individuo, mayor será la probabilidad de ser seleccionado como padre. Este principio trata de conducir a la población hacia mejores soluciones, pero esto no significa que los individuos de menor calidad deban ser descartados, ya que los individuos de menor calidad podrían aportar material genético útil a la población.

A continuación se describen algunos de los operadores de selección más comunes:

- **Selección aleatoria:** Los individuos de la población se seleccionan se realiza al azar. Cada individuo tiene la misma probabilidad de ser seleccionado para el cruce.
- **Selección por ruleta:** A cada uno de los individuos de la población se le asigna una probabilidad de selección, que va en función del fitness. Si todos los individuos tienen el mismo fitness, esta estrategia no introduce suficiente presión para seleccionar a los mejores individuos.
- **Selección por torneo:** Consiste en seleccionar k individuos al azar para luego aplicar un torneo sobre ellos y seleccionar el mejor. Por ejemplo para seleccionar 6 individuos, es necesario realizar 6 torneos.

Cruce

El operador de cruce es un operador comúnmente binario, aunque a veces es n-ario. El cruce se encarga de generar descendientes que hereden algunas características de los padres y debe garantizar que los nuevos individuos hereden el material genético de ambos padres y que representen soluciones válidas.

Se cuenta con una *probabilidad de cruce pc* que determina la probabilidad de que esta operación se realice, que suele encontrarse en el rango $[0{,}45, 0{,}95]$.

El operador de cruce básico es el de cruce en un punto, y consiste en seleccionar un punto de forma aleatoria y se generan dos hijos intercambiando los segmentos de los padres en ese punto. Un ejemplo se puede ver en la figura 3.1.

En este ejemplo, podemos ver que los padres tienen tamaño 10. Se seleccionó de forma aleatoria el punto después del cuarto dígito, quedando divididas dos secciones: [0,3] y [4,9]. El primer hijo heredará la primera sección del padre1 y la segunda sección del padre2, mientras que el segundo hijo heredará la primera sección del padre2 y la segunda sección del padre1.

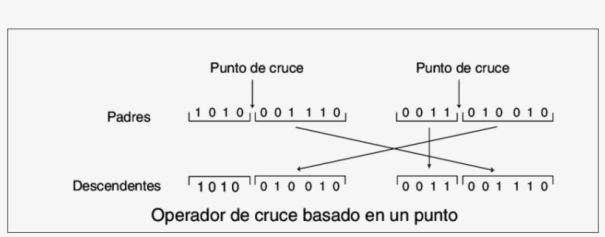

Figura 3.1: Cruce en un punto

Otro operador de cruce, que es muy conocido, es el operador Order Crossover (OX) [9], que es muy útil cuando le codificación está basada en permutaciones. En este operador, se copia una subsecuencia de símbolos del primer padre al hijo manteniendo el orden y la posición. El resto de símbolos ocupan las posiciones restantes manteniéndose el orden que tienen en el segundo padre, de forma que cada hijo hereda el orden y la posición de algunos genes de un padre, y el orden relativo de los restantes genes del otro padre. Un ejemplo gráfico lo podemos ver en la figura 3.2.

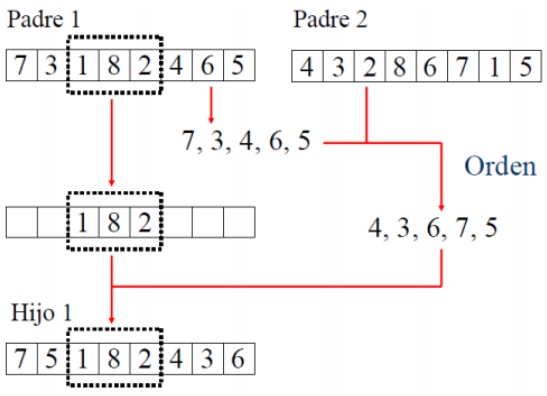

Figura 3.2: Operador de cruce OX

En el ejemplo podemos ver que los padres tienen tamaño 8 y que los dos puntos los dividen en tres secciones: [0,1], [2,4] y [5,7]. Del primer padre se escoge la sección entre los puntos, manteniendo el orden y la posición. Del segundo padre

cogen los símbolos que quedan (los de la primera y última sección), y después del segundo punto, se miran que símbolos no se repiten y en el mismo orden se añaden en el hijo. El segundo hijo se genera del mismo modo, intercambiando los roles de los padres.

Mutación

El operador de mutación se aplica a un solo individuo. Este operador introduce pequeños cambios en los individuos dependiendo de la *probabilidad de mutación pm*. La probabilidad de mutación suele ser muy baja para no perder cadenas de genes de buena calidad.

Un operador de mutación común es el operador de mutación simple, que modifica un único gen. Introduce características aleatorias en la estructura del cromosoma. Un ejemplo se puede ver en la figura 3.3.

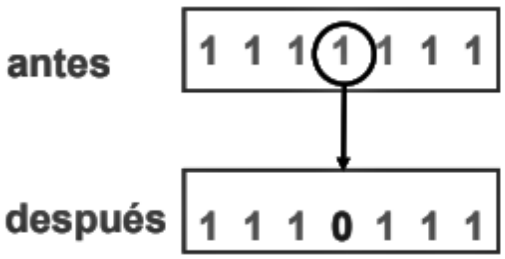

Figura 3.3: Mutación simple

Existe el caso de la mutación simple por permutaciones en el que se intercambian dos genes aleatoriamente.

3.4. Backtracking

El Algoritmo de Backtracking o Vuelta Atrás fue nombrado por primera vez en la década de 1950 por D. H. Lehmer. Este algoritmo es muy útil en los problemas en los que no disponemos de cierta información a la hora de tomar una decisión, y además esa decisión nos lleva a otras decisiones hasta llegar a una solución final. La idea de Backtracking se asemeja a un recorrido en profundidad dentro de un grafo dirigido.

Se llama Vuelta Atrás, ya que si el recorrido no tiene éxito en alguna etapa, este vuelve atrás, de tal forma que elimina los elementos que se han generado en ese nodo, y en el caso de que vuelva a un nodo que tiene uno o más hijos,

prosigue con ellos.

Como se puede ver en [10], el algoritmo llega a una solución que se va construyendo de forma progresiva y que se puede expresar en un vector $(x_1, x_2, ..., x_n)$, en el que n es el nivel final. En cada nivel el algoritmo buscará un estado para solución parcial, y en el caso de encontrarla, se añadiría y se pasaría al siguiente nivel. Un pseudocódigo de este algoritmo se muestra en el algoritmo 3.

Algorithm 3 Backtracking

```
1: procedure BACKTRACKING(j)
2:     if j == n then
3:         Añadir solución
4:         haySolucion = true
5:     else
6:         for i ← 0, n do
7:             if !haySolucion then
8:                 j=j+1
9:                 Backtracking(j)              ▷ Call Backtracking again
10:            end if
11:        end for
12:    end if
13: end procedure
```

Como se puede observar, el algoritmo de Backtracking recibe como parámetro el nivel en el árbol de estados. Si el nivel es igual a n, que es el tamaño la altura del árbol, es que el algoritmo ha llegado a una solución por lo que se añade. En caso contrario, el algoritmo recorre las ramas, y si todavía no se encontró la solución, vuelve a llamarse a backtracking aumentando el nivel del árbol.

Existe un algoritmo de Backtracking mejorado, en el que se realiza una poda de los nodos, y establece condiciones que no permite desarrollar nodos que no son válidos, de tal forma que podamos encontrar una solución más eficientemente.

Capítulo 4

Definición del problema

4.1. Motivación

Antes de comenzar las clases en cada curso académico, la Escuela de Ingeniería Informática de la Universidad de Oviedo realiza la asignación de los alumnos a los diferentes grupos de clases en las que están matriculados, determinado de esta forma a qué clase va a asistir y el horario completo de cada uno de los alumnos.

Desde hace unos años, para realizar esta asignación se ha utilizado una herramienta diseñada por Gonzalo De La Cruz Fernández en 2018 [1], fruto de un Trabajo de Fin de Grado. Esta herramienta es capaz de realizar un gran número de asignaciones de forma eficiente, teniendo en cuenta ciertos factores y configuraciones. Como configuraciones principales se puede nombrar la afinidad entre asignaturas y preferencias de asignaciones. Además, se debe añadir tanto las asignaturas con sus horarios, como los alumnos matriculados. Teniendo en cuenta estas configuraciones, se realiza el mayor número de asignaciones sin colisiones teniendo en cuenta ciertos factores: la calidad del horario y la homogeneidad entre los grupos. Con calidad del horario se hace referencia a que no tenga colisiones, que queden las menores horas posibles libres y los menores días libres.

Atendiendo a lo anterior, existen otros factores importantes que no se tienen en cuenta y que el responsable tiene que modificar después de ejecutar la herramienta. No se considera que pueda existir una asignación inicial o que un alumno no necesite ser asignado en tipo de clase de una asignatura. Por otro lado, también hay que tener en cuenta que si el alumno tiene un justificante por el que no puede acudir en una franja horaria, no se le puede asignar a un grupo al que no puede asistir, y también hay que tener en cuenta sus preferencias a la hora de realizar los horarios, para que estén conformes el mayor número de alumnos.

La motivación de este proyecto es extender la herramienta para ayudar al respon-

sable de realizar las asignaciones de los alumnos a los grupos de clases, pudiendo realizar una configuración inicial que tenga en cuenta factores que suelen afectar a la hora de realizar el horario, por lo que así el responsable solo tendrá que asegurarse que las configuraciones estén bien, ejecutarlo y seguidamente realizar alguna mejora manualmente que sea necesaria. Además, se desea contar con una funcionalidad que realice asignación de únicamente un alumno a sus asignaturas matriculadas, dando al usuario distintas posibilidades de asignaciones.

4.2. Descripción del problema

Cada año, en la Escuela de Informática de la Universidad de Oviedo se necesita realizar la asignación a los alumnos a grupos de clase en cada una de las asignaturas a las que está matriculado.

Esta asignación se realiza mediante una herramienta que contiene cierta información. En primer lugar, se cuenta tanto con los alumnos matriculados y a las asignaturas en las que lo está, así como el horario de cada asignatura. Además, se tiene en cuenta las asignaturas que colisionan entre sí (se solapan en el tiempo) y algunas preferencias de asignación..

La herramienta actual realiza asignaciones intentando maximizar la homogeneidad en el número de estudiantes en los grupos, así como algunos criterios de calidad de las soluciones. Sin embargo, a pesar de que es capaz de calcular buenas soluciones, hay algunos aspectos que no tiene en cuenta y que facilitarían en gran medidas la resolución de esta tarea por parte del responsable.

- Se puede contar con unas asignaciones iniciales que fueran satisfechas a la hora de utilizar la herramienta.
- Como no es obligatorio cursar todas las asignaturas de un curso ni superarlas todas para continuar los estudios, es esperable que haya gente que repite asignaturas que ha suspendido previamente. Existen casos en los que se les permite guardar la nota en ciertos tipos de clases de una asignatura que han sido superadas, por lo que no es necesario realizarle la asignación a ese tipo de clase.
- Los alumnos de la carrera normalmente tienen más de 18 años de que muchos pueden trabajar, o tienen algún tipo de justificante por lo que no pueden asistir en ciertas franjas horarias en días específicos de la semana, por lo que sería deseable no asignarle a los grupos que se encuentren en esa franja horaria restringida.
- Si realmente se pretende que el horario del estudiante sea de calidad, se deberían también tener en cuenta sus preferencias horarias.
- El responsable de realizar las asignaciones puede tener que hacer el horario de ciertos alumnos que se matriculan después del inicio de curso, o por

ejemplo, estudiantes de Erasmus, por lo que sería interesante contar con una herramienta que tenga en cuenta los factores o configuraciones dichos anteriormente que ofrezca varias soluciones a su horario.

4.3. Definición formal del problema

El problema se puede subdividir en dos subproblemas. Por un lado, la necesidad de asignar a todos los alumnos a sus asignaturas matriculadas, que a partir de ahora llamaremos *subproblema 1*, y por otro lado, la necesidad de ofrecer varias posibilidades de horario a un único alumno, que será *subproblema 2*.

4.3.1. Subproblema 1

Para la primera cuestión, que sería la asignación de todos los alumnos a grupos de sus asignaturas matriculadas, vamos a contar con varias entradas: los estudiantes, las asignaturas, las asignaciones iniciales, las asignaciones que no son necesarias, las restricciones y las preferencias.

En primer lugar, un conjunto de n *estudiantes* $S = \{S_1, S_2, ..., S_n\}$. Además contaremos también con un conjunto de m *asignaturas* $U = \{U_1, U_2, ..., U_m\}$. En este problema tenemos que tener en cuenta que un alumno S_i va a estar matriculado en un conjunto de asignaturas pertenecientes al conjunto U.

En la Escuela de Ingeniería Informática disponen de dos opciones de idiomas para cursar algunas asignaturas. Estos idiomas son: Inglés y Español. Para poder realizar la asignación de manera correcta ya que un alumno matriculado en una asignatura de Inglés, tiene que cursarla en Inglés, se va a contar como dos asignaturas diferentes. Es decir, Álgebra Español y Álgebra Inglés contarán como dos asignaturas diferentes en el sistema.

Cada asignatura está compuesta por t *tipos de clase* $T_i = \{T_{i_1}, T_{i_2}, ..., T_{i_t}\}$, que representa los tipos de clase en la asignatura. Por ejemplo, para la asignatura Álgebra, consideraríamos sus tipos de clase Laboratorio, Seminario y Teoría.

A su vez cada tipo de clase tendrá a su vez con un conjunto de grupos de clases $G_{ij} = \{G_{ij_1}, G_{ij_2}, ..., G_{ij_k}\}$, que sería por ejemplo en Álgebra teoría grupos 1 y 2. Esos grupos de clases tienen un número máximo de alumnos que pueden tener asignados $nmax_{ijk}$. Cada grupo de clase tiene a su vez un conjunto c_{ij} de clases $C_{ijk} = \{C_{ijk_1}, C_{ijk_2}, ..., C_{ijk_t}\}$. Las *clases* son eventos que se repiten todas las semanas y duran un determinado tiempo, definido por su hora inicio y hora final. Una clase la podemos representar de esta forma (díaDeLaSemana, hh:mm). Por ejemplo, la asignatura de Álgebra teoría, su grupo 1 tiene clase todas las semanas los martes de 15:00 a 17:00.

Además, debemos de contar con las restricciones y preferencias horarias, que

van a representarse de la misma forma aunque se refieran cosas diferentes. Las *restricciones* o *preferencias* horarias son limitaciones a la hora de realizar asignaciones. Estas restricciones o preferencias son en un día específico, y viene definido por una hora inicio y hora final, por lo que se puede representar de esta manera: (díaDeLaSemana,hh:mm). La diferencia principal es que las restricciones se tienen que cumplir obligatoriamente y la preferencias no. Es decir, un alumno tiene justificación para no tener clase de 15:00 a 17:00 los martes, por lo que no se le puede asignar a un grupo que tenga clase a esa hora ese día, pero si su preferencia es no tener clase a esa hora ese día, podría tenerla pero se tendrá en cuenta a la hora de elegir el mejor horario.

El objetivo del problema es que para cada alumno matriculado en asignaturas, se le asigne a un grupo de cada tipo de esas asignaturas. Para garantizar la validez y calidad de esta solución se deben satisfacer ciertas restricciones que se describen a continuación.

Restricciones del problema

Es necesario especificar ciertas restricciones para poder resolver el problema. Estas restricciones las podemos dividir en restricciones duras, y blandas.

Restricciones duras Las restricciones duras deben ser satisfechas obligatoriamente para poder garantizar que las soluciones generadas sean válidas. En este problema podemos identificar:

1. Ningún estudiante debe tener dos clases asignadas que colisionen entre ellas, es decir que se hagan en el mismo intervalo de tiempo. Para describirlo se tendrá en cuenta que dos clases A y B colisionan si se cumple que:

$$((inicioA < finalB) \wedge (finalA > inicioB))$$

2. Un alumno sólo puede ser asignado a grupos de clase en el idioma en el que está matriculado en esa asignatura.

3. Existen alumnos que no es necesario realizar asignación ya que están asignados previamente a un grupo en alguna asignatura, por lo que tienen una asignación inicial.

4. Un estudiante puede estar matriculado en una asignatura pero que no sea necesario asignarle a un grupo de un tipo de clase tipo de clase de esa asignatura.

5. Pueden existir restricciones del alumno a ciertas franjas horarias en días específicos de la semana.

Las restricciones 1, y 2 son restricciones contempladas en la herramienta actual, por lo que se mantienen y las restricciones 3, 4 y 5 nuevas respecto a la versión inicial.

El cuarto punto hace referencia a una asignación inicial ya que existen casos en los que hay alumnos que obligatoriamente tienen que estar asignados a un grupo concreto en alguna asignatura, por lo que la herramienta se debe asegurar que esa asignación se cumple.

Existen momentos en los que un alumno está matriculado en una asignatura, por ejemplo Álgebra, pero no es necesario asignarle a un tipo de asignatura, por ejemplo Laboratorio. Esto puede pasar, entre otros motivos, porque un alumno repite asignatura y se le guarda la nota de ese tipo de asignatura. A esto es a lo que se hace referencia en el punto quinto.

Existen franjas horarias en los que los alumnos no pueden asistir a clase por causas justificadas. En estos casos es necesario asignarle a grupos que no estén dentro de esa franja horaria.

Restricciones blandas Las restricciones blandas no son obligatorias ya que su incumplimiento no afecta a la validez de las soluciones, pero sí a su calidad, y suelen ser evaluadas en la función objetivo. Podemos identificar:

1. Todos los grupos de clase de un tipo de clase deben tener un número homogéneo de alumnos en cada asignatura.
2. Un alumno debe tener el mínimo número posible de horas libres entre clases.
3. Si es posible, un alumno no debe tener días con un número muy bajo de clases. Las horas de clase que un alumno tiene en una semana deben estar equilibradas entre todos los días de la semana evitando así que los estudiantes tengan días llenos de clases y días sin clase.
4. Si es posible, se deben tener en cuenta las preferencias horarias que tiene un alumno sobre las franjas horarias en las que desearía no tener clase.

La restricción número 4 es nueva respecto la versión inicial de la herramienta.

La primera restricción blanda busca tener clases igualadas en el número de alumnos, para que así sea más fácil organizar las clases y que estén en las mismas condiciones tanto por parte de los alumnos como del profesorado.

La segunda y la tercera restricciones blandas persiguen la calidad desde el punto de vista de los alumnos, ya que sería incomodo para ellos tener muchas horas libres porque muchos no viven cerca, y prefieren tener todas las clases seguidas y no perder tiempo. Además, esto equilibraría las clases entre todos los días de la semana, para así no tener un día mucha carga de trabajo y otros días nada.

La última restricción blanda garantiza la calidad desde el punto de vista del

alumno. Los alumnos suelen tener preferencias por franjas horarias, es decir, horas en las que prefieren no tener clase pero sin justificante, sino para organizarse mejor por lo que sería conveniente tenerlo en cuenta para que tengan un horario con el que van a estar más a gusto.

4.3.2. Subproblema 2

Para el segundo subproblema, que sería la asignación de un único alumno a sus asignaturas matriculadas, vamos a contar con las mismas entradas descritas anteriormente: los estudiantes, las asignaturas, las asignaciones iniciales, las asignaciones que no son necesarias, las restricciones y las preferencias horarias.

El objetivo del problema es que para el alumno, se ofrezcan varias soluciones a su problema de asignación a cada grupo de clase. El problema es parecido al anterior, pero cuenta con restricciones de problema diferentes.

Restricciones del problema

Es necesario especificar ciertas restricciones para poder resolver el problema. Estas restricciones las podemos dividir en restricciones duras, y blandas.

Restricciones duras En este problema podemos identificar:

1. El estudiante no debe tener dos clases asignadas que colisionen entre ellas, es decir que se hagan en el mismo intervalo de tiempo. Para describirlo vamos a tener en cuenta que dos clases A y B colisionan si se cumple que:

$$((inicioA < finalB) \wedge (finalA > inicioB))$$

2. El alumno sólo puede ser asignado a grupos de clase en el idioma en el que está matriculado en esa asignatura.
3. El alumno puede contar con una asignación inicial.
4. El estudiante puede no tener que ser asignado en tipos de clase de una asignatura.
5. Pueden existir restricciones del alumno a ciertas franjas horarias en días específicos de la semana.

Estas restricciones duras son parecidas a las del subproblema 1, con la diferencia de que son para un único alumno.

Restricciones blandas Podemos identificar:

1. Si es posible, se deben tener en cuenta las preferencias horarias que tiene un alumno sobre las franjas horarias en las que desearía no tener clase.

En el subproblema 2, se informa del número de preferencias horarias insatisfechas de cada solución.

Capítulo 5

Solución propuesta

Parte del problema que se ha descrito de las asignaciones a alumnos (subproblema 1) ha sido resuelto mediante un Algoritmo Genético para explorar el espacio de búsqueda y un Algoritmo Voraz para evaluar las soluciones y guiar la búsqueda hacia soluciones de buena calidad.

Para resolver el resto del problema y ampliar el diseño inicial, se necesita modificar ciertos factores de estos algoritmos para que tenga en cuenta las asignaciones iniciales, la necesidad de no realizar asignación en un tipo de asignatura, las preferencias y restricciones horarias. En la solución propuesta del subproblema1 se va a hablar de lo ya resuelto y de lo nuevo.

Por otro lado, el subproblema 2 es nuevo, por lo que es necesario a añadir una funcionalidad a la herramienta para que cuente con la posibilidad de mostrar un número dado de posibilidades de asignaciones a asignaturas. Para ello se emplea un algoritmo de Backtracking.

5.1. Espacio de búsqueda

5.1.1. Concepto de asignación

En el problema actual, una asignación esta compuesta por tres elementos. En primer lugar, el estudiante al que se le realiza. En segundo lugar, el tipo de clase y por último el grupo al que es asignado. Teniendo esto en cuenta, una asignación se puede representar de la siguiente forma

$$(U_i, T_{jk}, G_{jkz})$$

Para entenderlo mejor, se podría decir que una asignación viene descrita por el estudiante Fernando, y se le asigna en Álgebra Laboratorio el grupo 1, y se representaría: ($Fernando, Alg.L, Alg.L$,1).

El total de asignaciones que se tienen que realizar para todos los estudiantes se puede calcular de esta forma, siendo n el número de estudiantes y E_i el conjunto de asignaturas a las que está matriculado el alumno:

$$TotalAsignaciones = \sum_{i=1}^{n} \sum_{j=1}^{|E_i|} |T_j|$$

El proceso empieza con una asignación vacía, de tal forma que está incompleta

$$AsignacionIncompleta = (S_i, T_{jk}, -)$$

En una solución completa todas las asignaciones que se han podido realizar tendrán los tres valores fijando un grupo de clase concreto en el tercer elemento de la tupla..

5.1.2. Concepto de restricción horaria

Una restricción horaria en este problema es una franja horaria en un día de la semana en la que el alumno no puede asistir a clase compuesta por cuatro elementos: el estudiante, el día de la semana, hora inicio y hora final y se puede representar de la siguiente forma.

$$(U_i, diaSemana, horaInicial, horaFinal)$$

Un ejemplo sería: Fernando no puede asistir los lunes de 16:00 a 18:00.

5.1.3. Concepto de preferencias horarias

Cuando se piensa en preferencias horarias, normalmente se piensa en las horas que el alumno preferiría tener clase, pero como ese marco es muy amplio, se restringe a las horas que el alumno no quiere tener clase. Viene definido por un alumno, un día de la semana, hora inicio y hora final, y significaría la horas que el alumno preferiría no tener clase. Se puede representar de la misma forma que las restricciones horarias mencionada anteriormente.

$$(U_i, diaSemana, horaInicial, horaFinal)$$

5.1.4. Asignaciones a todos los alumnos (Subproblema 1)

Definición de estado

Un estado es una solución parcial en la que no se han realizado todas las asignaciones. Inicialmente el problema cuenta con una solución compuesta por asignaciones incompletas. Durante el proceso, se evaluarán varios estados intermedios representados como soluciones parciales y cuando se han realizado todas las asignaciones se puede decir que se ha generado una solución completa.

Expansión de un estado

La expansión de un estado se realiza asignando un grupo de clase a una asignación que se encuentre incompleta dentro del subconjunto de asignaciones de la solución parcial que no se ha completado aún, es decir, asignarle a un alumno un grupo que no se haya hecho anteriormente.

Para una asignación incompleta $A = (S_i, T_{kk}, -)$, el número de nuevos estados que se pueden generar para un alumno y tipo de asignatura se puede calcular como

$$NumeroEstadosDirectos = |G_{jk}|$$

Como el número de estados posibles es muy grande, la búsqueda es realizada por un Algoritmo Voraz que utiliza algunos heurísticos para llegar a soluciones prometedoras. Lo que hace el Algoritmo Voraz es tomar decisiones que no considera todos los estados posibles y toma decisiones deterministas en función del orden en que se presentan las asignaciones a realizar.

En el Algoritmo Voraz las soluciones dependen del orden en el que se realicen las asignaciones, por lo que se usa un Algoritmo Genético que genera nuevas ordenaciones que serán evaluadas por el Algoritmo Voraz.

Dificultad de las instancias

La dificultad de las instancias depende de varios factores. Evidentemente uno de los factores es el número de alumnos, pero no es el único. Las características de la matrícula de cada alumno aumentan o disminuyen la dificultad de resolver el problema. En la Escuela de Ingeniería Informática, existen cursos en los que las asignaturas se cursan en la misma franja horaria, por lo que por ejemplo, si un alumno tiene asignaturas de primero y de tercero (que en ambos cursos son por la mañana), realizar las asignaciones podría complicarse.

Otro factor importante es el número de grupos de clase disponibles para cada tipo de clase de una asignatura, ya que cuantos más grupos haya, más fácil será encontrar un grupo a un alumno que no se solape con su asignación parcial.

Las asignaciones iniciales podrían dificultar la instancia ya que pueden haberse configurado asignaciones iniciales que hagan imposible o dificulten al algoritmo encontrar buenas soluciones.

Si el alumno tiene restricciones horarias en la mayoría de las horas disponibles o en su totalidad, dificultará la instancia, pudiendo hacer imposible realizar una asignación completa por lo que hay que ser muy estrictos con lo que se considera restricción, solo en casos debidamente justificados.

Un factor muy importante es el diseño del horario de la Escuela de Ingeniería

Informática de la Universidad de Oviedo, ya que, como se dijo anteriormente, se tiene las clases de primero y tercero por la mañana y segundo y cuarto curso por la tarde. Pero esto podría cambiar y podría aumentar las posibilidades de que varias asignaturas se solapen, aumentando así la dificultad.

5.1.5. Asignación a un único alumno (Subproblema 2)

Definición de estado

El estado definido en este problema, es parecido al descrito en la sección 5.1.4, pero en ese el estado cuenta con todas las asignaciones necesarias de los alumnos matriculados, y en este caso se cuenta con las asignaciones de un solo alumno.

Expansión de un estado

La expansión de un estado se realiza asignando un grupo de clase a una asignación que se encuentre incompleta dentro del subconjunto de asignaciones de la solución parcial que no se ha completado aún, es decir, asignarle al alumno un grupo que no se haya hecho anteriormente.

Para una asignación incompleta $A = (S_i, T_{kk}, -)$, el número de nuevos estados a los que se puede ir directamente se puede calcular como

$$NumeroEstadosDirectos = |G_{jk}|$$

El número de estados posibles son aquellos que cumplen las restricciones duras definidas, y el orden que se va a seguir va a ser el orden en el que se encuentre.

Dificultad de estancia

La dificultad de la instancia dependerá de las característica de la matrícula, las asignaciones iniciales y las restricciones horarias del alumno. Estas características determinarán el número de soluciones factibles, pudiendo ocurrir que no haya ninguna que cumpla todas las restricciones, y por tanto aumentando la dificultad de la instancia.

5.2. Matriz de colisiones

La matriz de colisiones nos permite detectar solapamientos entre grupos de clase facilitando la identificación si dos grupos de clase colisionan entre sí. Se define como una matriz booleana de forma que las filas y las columnas representan los grupos de clase.

Para ejemplificar la matriz de colisiones se puede considerar un ejemplo en el

que tenemos dos asignaturas, por ejemplo laboratorio y con dos grupos de clase para cada laboratorio.

$$G_{Alg.L,1}, G_{Alg.L,2}, G_{Comp.L,1}, G_{Comp.L,1}$$

Las colisiones se representan como una matriz de 4×4 con el mismo orden descrito anteriormente

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{5.1}$$

Los 0's representan la falta de colisión y los 1's la existencia de colisión. Por tanto, según la matriz anterior, Alg.L.1 colisiona con Comp.L.1, es decir, los horarios de estos dos grupos de clase se solapan en el tiempo.

La matriz de colisiones se calcula antes de ejecutar el algoritmo, que será empleada por el Algoritmo Voraz y como se puede ver, su uso es muy sencillo y nos ayuda a garantizar la eficiencia del algoritmo.

5.2.1. Excepción de colisión

Existen asignaturas que se cursan a la misma hora pero en semanas alternas por lo que realmente no colisionan. Esta situación no es poco común, ya que ocurre con algunos seminarios.

Estos casos se tienen que tener en cuenta, ya que es información necesaria para la inicialización de la matriz de colisiones. Para ello se utiliza un archivo de configuración.

5.2.2. Colisión obligatoria

En algunas situaciones es necesario forzar la colisión de grupos que realmente no colisionas. Esto pasa, por ejemplo, cuando los alumnos del grupo de teoria 1 y solamente estos alumnos, deben estar asignados a los grupos de seminario 2 o 3. Para que esto ocurra, se fuerza colisión del grupo 2 de teoría con los seminarios citados anteriormente, y además del grupo de teoría 1 con el resto de seminarios disponibles, de tal forma que va a estar obligado a que si asiste a la teoría 1, también asista a los seminarios 1 o 2.

Esta información se da a la herramienta mediante un archivo de configuración.

5.3. Algoritmo Voraz

El Algoritmo Voraz es un algoritmo determinista que recibe una lista de asignaciones incompletas como entrada e itera sobre ella asignando un grupo de clase

a cada una. El algoritmo utiliza algunos heurísticos para determinar el grupo que se asigna, y ha sido diseñado para lograr unos objetivos principales:

1. El número de alumnos asignados a un grupo de un tipo de clase de una asignatura debe ser homogéneo con el resto de grupos de ese tipo de clase.
2. Los alumnos no deben tener asignaturas que colisionen entre sí.
3. El alumno debe estar asignado a un grupo si se encuentra en las asignaciones iniciales (a no ser que esas asignaciones se encuentren dentro de sus restricciones horarias).
4. Las asignaciones de un alumno deben cumplir sus restricciones horarias.

El algoritmo recibirá como entrada una lista de asignaciones con un orden determinado, por lo que al ser un algoritmo determinista, si esa lista contiene las asignaciones en el mismo orden, la salida del algoritmo será la misma. Esa entrada será creada por el Algoritmo Genético, que es el encargado de crear nuevos órdenes de asignación que será evaluado por el Algoritmo Voraz

A continuación se explican cada uno de los componentes del Algoritmo Voraz.

5.3.1. Asignación inicial

Se pueden realizar asignaciones iniciales antes de la ejecución del algoritmo sin necesidad de un heurístico. Estas asignaciones iniciales se administran mediante un archivo de entrada.

En las asignaciones iniciales se da por hecho que no colisionan entre ellas. El procedimiento de asignación inicial del Algoritmo Voraz se muestra en el algoritmo 4.

Algorithm 4 Asignación inicial

```
1:  procedure initialAssignmentsDone(assignments)
2:      for all a ∈ initialAssignment do
3:          if preference(a.student, a.group) then
4:              a.preference=1
5:          else
6:              a.preference=0
7:          end if
8:          a.type=INICIAL
9:          assignments.add(a)
10:         a.group.addStudent(a.student)
11:         a.student.addAssignment(a)
12:     end for
13: end procedure
```

Las asignaciones iniciales que maneja el algoritmo llevan un preprocesamiento en el que si una asignación inicial se encuentra dentro de las restricciones horarias del alumno, no se añade, y se tratará como una asignación normal sin tener en cuenta que tiene ese grupo como asignación inicial. Es por esto que en el algoritmo no se comprueba si la asignación afecta a las restricciones, pero si que tiene en cuenta si las asignaciones iniciales cumplen las preferencias horarias del alumno.

5.3.2. Preprocesamiento

Antes de la ejecución del algoritmo, se pueden realizar unas operaciones de preprocesamiento que nos ayudan a mejorar la eficiencia del algoritmo.

Se realizan asignaciones directas sin necesidad de un heurístico ya que existen tipos de clases que solo tienen un grupo, por lo que se realiza esa asignación automáticamente al no existir otra posibilidad. Solo existe tres casos en los que no se realice esta asignación, y es en el caso de que colisione con una asignación inicial previamente hecha, que incumpla las restricciones horarias del alumno, si las hubiera o que ya se le haya realizado una asignación (en el caso de ser una asignación inicial). Este procedimiento se puede ver en el algoritmo 5.

Algorithm 5 Preprocesamiento

```
1:  procedure PREPROCESSING(assignments)
2:      for all a ∈ assignments do
3:          if a.group==null and len a.typeOfClass.groups==1 then
4:              selectedGroup=a.typeOfClass.groups[0]
5:              if !constraint(a.student, selectedGroup) then
6:                  if preference(a.student, selectedGroup) then
7:                      a.preference=1
8:                  else
9:                      a.preference=0
10:                 end if
11:                 assignments.add(a)
12:                 a.group=selectedGroup
13:                 selectedGroup.addStudent(a.student)
14:             end if
15:         end if
16:     end for
17: end procedure
```

5.3.3. Heurístico

Para determinar qué grupo de clase se debe asignar al alumno cuando se evalúa una asignación incompleta se utiliza un heurístico que se define como:

Definición 5.3.1 *El grupo de clase que se debe asignar al alumno en un tipo de clase de una asignatura debe ser el que menor cantidad de alumnos asignados tenga, que no colisione con las asignaciones ya realizadas al alumno, que no haya alcanzado su número máximo de estudiantes y que no se encuentre dentro de las franjas horarias restringidas por el alumno.*

El heurístico definido nos ayuda a satisfacer los objetivos principales del Algoritmo Voraz descritos anteriormente. La función que decide qué grupo se asigna en una asignación en el Algoritmo Voraz se muestra en el algoritmo 6.

Evitando que el grupo que se le va a asignar a un alumno colisione con los grupos que tiene asignados, nos impide que tenga dos grupos a la misma hora. Además, seleccionando el grupo con menos alumnos y a su vez, no asignando un grupo que haya alcanzado a su máximo, nos ayuda a que los grupos de un tipo de clase de una asignatura sean homogéneos. Por último, evitando que un alumno sea asignado a un grupo que su horario se encuentre entre las horas restringidas, nos ayuda a que el alumno pueda asistir a su clase asignada sin necesidad de hacer un cambio de clase.

Algorithm 6 Heurístico de selección de grupos

```
1:  procedure getBestGroup(assignment)
2:      collidedAssignments = ∅
3:      minNumberOfStudents = MAX_INTEGER_VALUE
4:      selectedGroup = ∅
5:      for all group ∈ assignment.typeOfClass.groups do
6:          assignmentCollided=getAssignmentCollided(assignment.stu-
    dent,group)
7:          if assignmentCollided == ∅ then
8:              if !constraint(a.student, selectedGroup) then
9:                  selectedGroup=group
10:                 minNumberOfStudents = group.numberOfStudents
11:             end if
12:         else
13:             collidedAssignments.insert(assignmentCollided)
14:         end if
15:     end for
16:     return selectedGroup, collidedAssignments
17: end procedure
```

5.3.4. Proceso de reparación

Existen ocasiones en las que el heurístico no puede realizar la asignación a un grupo ya que todos los grupos de ese tipo en esa asignatura colisionan con asignaciones del alumno ya hechas. Cuando ocurre esto, se realiza un proceso de reparación.

El proceso de reparación recibe la lista de todas las asignaciones que colisionan con la asignación que estamos intentando realizar e itera sobre ellas. La asignación sobre la que se está iterando puede ser de dos tipos:

- Asignación inicial dada por el usuario.
- Asignación no-inicial realizada por el algoritmo voraz en una iteración anterior.

En el caso de que esa asignación sea inicial, no se hace nada ya que ese tipo de asignaciones *no* se deben cambiar.

En el caso de que la asignación no sea inicial, se intenta realizar un cambio de grupo de clase. El grupo que será seleccionado para cambiar será el que determine el heurístico y además que no debe colisionar con el grupo de la asignación que se estaba intentando realizar. Después de llevar a cabo el cambio, se intenta asignar el grupo y si es posible en este momento realizar la asignación, se detiene el proceso. En el caso de que no sea posible, se continúa con la siguiente asignación de la iteración, de manera que se realizan cambios en los grupos que colisionan hasta que se pueda realizar. Existen ocasiones en las que termina el proceso y no se ha podido realizar la asignación, por lo que esta asignación entra dentro de la lista de asignaciones no resultas que tendría que realizar el responsable de forma manual. Este proceso se encuentra en el algoritmo 7.

Algorithm 7 Reparación

```
1: procedure REPAIRING(newAssignment, collidedAssignments, unsolvedCollisions)
2:     solvedCollision = false
3:     student = newAssignment.student
4:     for all assignmentCollided ∈ collidedAssignments do
5:         if assignmentCollided.type!=INICIAL then
6:             newCollisionAssignedGroup = getBestGroup(assignmentCollided)
7:             if newCollisionAssignedGroup != ∅ then
8:                 assignmentCollided.group.removeStudent(student)
9:                 assignmentCollided.group = newCollisionAssignedGroup
10:                if preference(student, assignmentCollided.group) then
11:                    assignmentCollided.preference=1
12:                else
13:                    assignmentCollided.preference=0
14:                end if
15:                newCollisionAssignedGroup.addStudent(student)
16:                newAssignmentGroup = getBestGroup(NewAssignment)
17:                if newAssignmentGroup != ∅ then
18:                    newAssignment.group = newAssignmentGroup
19:                    newAssignmentGroup.addStudent(student)
20:                    solvedCollision = true
21:                    break
22:                end if
23:            end if
24:        end if
25:    end for
26:    if !solvedCollision ∧ newAssignment.type!=INICIAL then
27:        unsolvedCollisions.add(newAssignment)
28:    end if
29: end procedure
```

5.3.5. Preferencias de asignación

Existen preferencias a la hora de realizar una asignación a los estudiantes. El horario de la Escuela de Ingeniería Informática de la Universidad de Oviedo está pensando para que si un estudiante es asignado al mismo número de grupo de clase para todas las asignaturas del mismo curso, no se producen colisiones. Por ejemplo, si un alumno es asignado al grupo 1 de laboratorio de Álgebra, si solo tiene asignaturas de primero, se le podría asignar al grupo 1 de laboratorio del resto de asignaturas.

El Algoritmo Voraz cuando realiza la asignación de un grupo, mira si hay otra asignatura del mismo curso y del mismo tipo para realizar la asignación. Esta asignación de preferencias es no determinista ya que al existir un proceso de

reparación puede ser modificado. Podemos definir preferencias como:

Definición 5.3.2 *Conjunto de tipos de clase que cuando a un alumno se le asigna a una de ellas, si es posible debe ser asignado al mismo número de grupo en el resto.*

Las preferencias se definen mediante archivos de configuración. La asignación de preferencias de asignación en el Algoritmo Voraz se muestra en el algoritmo 8.

Algorithm 8 Preferencias de asignación

```
1:  procedure assignPreferences(assignment, selectedGroup)
2:      student = assignationMade.student
3:      preferences = getPreferences(assignmentMade.group)
4:      if preferences != ∅ then
5:          for all group ∈ preferences do
6:              for all studentAssignment ∈ student.assignments do
7:                  if studentAssignment.typeOfClass = group.typeOfClass then
8:                      assignmentCollided=getAssignmentCollided(student,
    group)
9:                      if !constraint(studentAssignment.student, group) then
10:                         if preferences(studentAssignment, group) then
11:                             studentAssignment.preference=1
12:                         else
13:                             studentAssignment.preference=0
14:                         end if
15:                         studentAssignment.group = group
16:                         group.addStudent(student)
17:                     end if
18:                 end if
19:             end for
20:         end for
21:     end if
22: end procedure
```

Como se ve, se tiene en cuenta además que no haya restricciones horarias que afecten al alumno para ese grupo, y en el caso de que no haya problema, si mira si cumple sus preferencias horarias.

5.4. Algoritmo Genético

El Algoritmo Genético realizará secuencias de asignaciones en diferentes órdenes que luego el Algoritmo Voraz evaluará. Una vez que el Algoritmo Voraz evalúa una secuencia, se tienen en cuenta algunos factores para determinar la calidad de la solución generada. Este algoritmo como es utilizado en este proyecto se puede ver en el algoritmo 9. El Algoritmo Genético recibe como parámetros la

Algorithm 9 Algoritmo genético

```
1: procedure GENETICALGORITHM(p_m, p_c, NGen, PopSize)
2:     population = generateInitialPopulation(PopSize)
3:     generation = 0
4:     while generation ≤ NGen do
5:         newPopulation = ∅
6:         pairs = selectPairs(population)
7:         for all pair ∈ pairs do
8:             if generateRandomFloat < p_c then
9:                 descendantsEncoded = crossover(pair)
10:                descendantsDecoded = ∅
11:                for all descendantEncoded ∈ descendantsEncoded do
12:                    if generateRandomFloat < p_m then
13:                        mutate(descendantEncoded)
14:                    end if
15:                    descendantDecoded = decode(descendantEncoded)
16:                    greedyAlgorithm(descendantDecoded)
17:                    fitnessFunction(descendantDecoded)
18:                    descendantsEncoded.add(descendantEncoded)
19:                end for
20:            end if
21:            bestTwoIndividuals = tournamentPD(pair, descendantsDecoded)
22:            newPopulation.add(bestTwoIndividuals)
23:        end for
24:        population = newPopulation
25:        generation = generation + 1
26:    end while
27:    return population
28: end procedure
```

probabilidad de mutación *pm*, la probabilidad de cruce *pc*, el número máximo de generaciones que el Algoritmo Genético simulará *NGen* y el tamaño de la población *PopSize*. Estos valores son modificables, de modo que si se quiere cambiar para una instancia específica es posible.

El Algoritmo Genético empieza generando una población inicial con un tamaño *PopSize* de individuos. Cada individuo contará con las cromosomas que son necesario. Es decir, no contemplará los cromosomas que representen las asignaciones iniciales (que no se ven afectadas por las restricciones horarias) ni las asignaciones que no son necesarias. A continuación, hasta que no se alcanza el criterio de generación máxima, en cada generación el algoritmo selecciona *n/2 pares* de individuos, que cada uno tiene una probabilidad *pc* de cruzarse. Si un par se cruza, generará dos hijos que tendrán una probabilidad *pm* de ser mutados.

Los operadores de cruce y mutación se aplican sobre la codificación de los individuos, y el algoritmo voraz se encarga de decodificarlos, transformando una secuencia de asignaciones a una solución. Los individuos se representan en una lista de asignaciones a realizar con un orden determinado, que será lo que reciba el Algoritmo Voraz, de forma que realizará las asignaciones en ese orden. Por último, una función evaluará la calidad de la solución generada por el Algoritmo Voraz.

Los individuos que compondrán la población de la siguiente generación se seleccionan en un torneo entre padres y descendientes que tiene lugar para cada pareja. Sólo los dos individuos más aptos de cada torneo sobrevivirán y formarán parte de la siguiente generación.

5.4.1. Codificación - Decodificación

La codificación y decodificación de este algoritmo mantiene el realizado en la versión anterior de la herramienta [1].

Los individuos se codifican como permutaciones de enteros cuya longitud es igual al número de asignaciones que se deben hacer para esa instancia específica del problema. Cada individuo codificado representa un orden en el que las asignaciones deben ser procesadas por el Algoritmo Voraz. Las diferentes posiciones del array contienen un número que representa un índice en una lista maestra que contiene todas las asignaciones a realizar. Este índice indica qué asignación debe ser evaluada por el Algoritmo Voraz en esa posición.

Para entender esta idea, se considera el siguiente ejemplo. Podemos tener una lista de asignaciones que serían el problema a resolver

$$[(U_1, T_{11}, -), (U_1, T_{12}, -), (U_2, T_{11}, -), (U_2, T_{12}, -)]$$

En concreto, serían dos alumnos U_1 y U_2, que a cada uno se le tiene que asignar a dos tipos de asignatura, T_{11} y T_{12}. Una codificación a este problema podría

ser

$$2, 3, 4, 1$$

, que hace referencia al orden en el que serán evaluados, siendo el primero el alumno U_2 en el tipo de asignatura T_{12} $(U_2, T_{12}, -)$, y el segundo el alumno U_1 en el tipo de asignatura T_{11} $((U_1, T_{11}, -))$.

El decodificador primero genera una nueva lista de asignaciones a partir de la lista inicial pero en el orden determinado por la codificación. Si aplicamos la decodificación en el ejemplo anterior, como resultado quedará

$$[(U_2, T_{12}, -), (U_1, T_{11}, -), (U_1, T_{12}, -), (U_2, T_{11}, -)]$$

siendo esta la lista que recibe el Algoritmo Voraz, produciendo una solución.

5.4.2. Operadores

En los operadores se mantiene la solución propuesta por Gonzalo De La Cruz Fernández [1] ya que funciona de forma correcta y en las ampliaciones no se ha necesitado realizar ningún cambio.

Operador de selección

El operador de selección que se propone selecciona siempre todos los individuos de la población para el cruce: se generan $n/2$ pares aleatorios y aproximadamente el $100 \times pc$ por ciento de ellos se cruzan y generan hijos.

Para cada una de las parejas, el operador de selección realiza un torneo entre padres e hijos y sólo los dos individuos con mejor fitness pasan a la siguiente generación. Este proceso introduce elitismo en el algoritmo, ya que se asegura que la información genética de buena calidad se mantiene ya que le mejor individuo de cada generación siempre formará parte de la siguiente.

Operador de cruce

El operador de cruce utilizado es el operador de cruce Order Crossover (OX) que ha sido explicado previamente en la sección 3.3.1. Es uno de los operadores clásicos utilizados en los Algoritmos Genéticos cuando los cromosomas están representados por permutaciones de elementos y es ampliamente utilizado. Se ha demostrado que el operador OX obtiene buenos resultados cuando se trabaja con permutaciones.

En la solución propuesta el primer punto del cruce debe estar entre la primera posición y el 80 % de la longitud del individuo. El segundo punto estará entre el primero y la longitud del individuo. Un pseudocódigo se puede ver en el algoritmo 10.

Algorithm 10 Operador de cruce

```
1: procedure CROSSOVER(pair)
2:     parent1 = pair.parent1
3:     parent2 = pair.parent2
4:     startingPoint = randomPosition(0, len parent1 * 0.8)
5:     endPoint = randomPosition(startingPoint, len parent1)
6:     descendant1 = generateIndividual(parent1, parent2, startingPoint, endPoint)
7:     descendant2 = generateIndividual(parent2, parent1, startingPoint, endPoint)
8:     return descendant1, descendant2
9: end procedure
```

Operador de mutación

Recibe un individuo codificado y realiza pequeñas mutaciones sobre él, intercambiando el orden de dos elementos. El número de intercambios que realiza sobre el individuo pasado como parámetro varía entre 0 y 5 % de la longitud del individuo. Un pseudocódigo se puede ver en el algoritmo 11.

Algorithm 11 Operador de mutación

```
1: procedure MUTATION(individual)
2:     numberOfMutations = generateRandomInt(0, len individual * 0.05)
3:     i = 0
4:     while i <numberOfMutations do
5:         positionGene1 = generateRandomInt(0, len individual)
6:         positionGene2 = generateRandomInt(0, len individual)
7:         swap(positionGene1, positionGene2)
8:         i = i + 1
9:     end while
10: end procedure
```

5.4.3. Fitness

La función objetivo extiende la especificada en [1] teniendo en cuenta las preferencias horarias de los estudiantes, y a la vez manteniendo los criterios considerados en la versión anterior de la herramienta. Los criterios se describen a continuación.

Homogeneidad en el número de alumnos por grupo de clase

- Varianza total entre grupos: es la suma de todas las varianzas del número de alumnos que tiene cada grupo de clase de todo tipo de asignaturas. Se utiliza para intentar que el número de alumnos en los grupos de clase de un tipo de asignatura sea homogéneo.

- Diferencia máxima entre el número de alumnos de dos grupos de clase del mismo tipo de asignatura: es la máxima diferencia que existe entre el grupo de clase con menor número de alumnos y el de mayor número de estudiantes del mismo tipo de clase de una asignatura. Se utiliza junto con el anterior para evitar que haya un grupo o un conjunto de grupos con una gran diferencia en el número de alumnos entre los grupos de clase con menor y mayor número de estudiantes. Utilizando sólo la varianza total entre grupos esto no se puede asegurar.

Colisión entre grupos de clases

- Número de colisiones: Es el número total de asignaciones que el algoritmo no pudo completar. Este es el factor más importante porque evitar las colisiones entre clases es una restricción dura del problema y minimizar el número de asignaciones no realizadas facilita la resolución del problema por parte del responsable.

Calidad del horario de los estudiantes

- Número total de franjas libres: Es la suma de todas las franjas de media hora libres que los alumnos tienen durante la semana. Solo se tienen en cuenta las franjas entre clases, no las franjas antes del comienzo de la primera clase y después del final de la última. Se utiliza para intentar que los estudiantes tengan el mínimo de tiempo libre entre clases para que sus horarios sean más compactos.

- Número total de días con bajo número de clases: Es la suma de todos los días que los alumnos tienen con menos de dos horas de clase. Se utiliza para evitar que los estudiantes tengan que ir a la Universidad sólo para una pequeña cantidad de clases.

- Número total de preferencias horarias que no se cumplen: Es la suma de todas las asignaciones que no cumplen las preferencias del alumno. Se utiliza para intentar que los estudiantes tengan el mayor número de preferencias satisfechas.

Se podrían haber tenido en cuenta otros criterios de optimización en el diseño de la función de fitness. Gonzalo De La Cruz Fernández [1] pensó en considerar las preferencias de las asignaturas como otro criterio para el cálculo de la función fitness pero lo descartó ya que con los experimentos se vio que no era necesario considerarlo y supone un aumento del tiempo a la hora de evaluar la calidad del individuo.

Hay que tener en cuenta que las preferencias del alumno a la hora de su horario, es un factor que se debería tener en cuenta más que los factores de que haya menos horas libres y menos días libres, ya que podemos considerar que el

alumno está mejor teniendo menos horas libres pero si cumpliendo eso, no cumplimos sus preferencias horarias, al final va a disminuir la calidad de su horario para él.

Para definir la fórmula de cálculo del fitness de un individuo se dan diferentes pesos a las medidas explicadas anteriormente. Los pesos se pueden modificar para dar más prioridad a unos factores sobre otros en el fichero de configuración. Los pesos wc, wv, wd, wf y wl se encontraban en la herramienta base, pero el peso wp es nuevo.

$$\begin{aligned} fitness = wc * numeroColisiones \\ +wv * totalDiferencia \\ +wd * maxDifEntreGrupos \\ +wf * totalHorasLibres \\ +wl * diasClasesBajas \\ +wp * totalPrefSinCumplir \end{aligned} \tag{5.2}$$

Para determinar la mejor función fitness se han realizado diferentes experimentos que se puede ver en la sección 11.1.3 y como resultado se ha obtenido que los valores recomendados vienen descritos en la tabla 5.4.3.

[H]

wc	100
wv	1
wd	10
wf	0.08
wl	0.2
wp	25

Hay que aclarar que cuando se hace referencia al fitness, no es el fitness convencional, ya que en la solución propuesta se resuelve un problema de minimización en el que el mejor individuo es el que tenga menor valor de la función descrita. Se mantiene así de la herramienta base, ya que es más visual para el usuario. Es por esto que cuanto mayor sea el "fitness", peor es la solución.

5.5. Backtracking

El Algoritmo de Backtracking ha sido seleccionado para ofrecer distintas soluciones sobre un alumno en el subproblema 2.

El algoritmo tiene como entrada la matricula del alumno, el horario de la Escuela, las asignaciones iniciales del alumno, sus restricciones y preferencias. En el archivo de configuración se especificará el número de soluciones que se desea, así como el semestre y el número de intentos que se desea que haga.

Este algoritmo se puede dividir en dos modos:

1. Modo determinista.

2. Modo aleatorio.

5.5.1. Creación del individuo

Se cuenta con una lista por cada alumno de las asignaciones que no son necesarias de realizar y las asignaciones iniciales, para no realizarle ninguna asignación. En el momento de crear el individuo, se desechan las asignaciones que se encuentran en la lista. Se puede ver un pseudocódigo en el algoritmo 12:

Algorithm 12 Creación de individuo

```
1: procedure createIndividual(planification, notDoAssignment)
2:     for all student ∈ planification.students do
3:         for all subject ∈ student.subjects do
4:             for all subjectclass ∈ subject.subjectclasses do
5:                 if subjectclass ∉ notDoAssignment(student) then
6:                     assignment = new Assignment(student, subjectclass)
7:                     assignments.add(assignment)
8:                     student.addAssignment(assignment)
9:                 end if
10:            end for
11:        end for
12:    end for
13:    return new Individual(assignments)
14: end procedure
```

Posteriormente, en las asignaciones del individuo se añade al principio las asignaciones iniciales que no se ven afectadas por las restricciones horarias del alumno. Esto es así, porque en la llamada inicial de backtracking se le pasa como parámetro el tamaño de las asignaciones iniciales, para que comienza en el nivel del árbol donde solo se encuentran asignaciones sin realizar.

5.5.2. Ejecución backtracking

La ejecución del algoritmo de backtracking depende del modo que se haya seleccionado en la configuración.

Modo determinista

En el modo determinista, el algoritmo de Backtracking no para al encontrar una solución, sino que acaba cuando se lleguen a las soluciones deseadas, que se encuentran en el fichero de configuración. De esta forma, cuando encuentra una solución, vuelve a atrás y sigue asignando otro grupo diferente, por lo que soluciones consecutivas suelen ser muy parecidas. La del programa se puede ver en el algoritmo 13, y su invocación en el algoritmo 14.

Algorithm 13 Ejecución programa

```
1: procedure START
2:     backtracking(len initialAssignments)
3: end procedure
```

Algorithm 14 Backtracking determinista

```
1: procedure BACKTRACKING(j)
2:     if j = len assignments then
3:         if individual ∉ solutions then
4:             soluciones.add(individual)
5:             sol++
6:             if sol == NUMERO_SOLUCIONES then
7:                 return
8:             end if
9:         end if
10:    else
11:        assignment = individual.getAssignment(j)
12:        groups = assignment.groups
13:        for all group ∈ groups do
14:            if !constraint(assignment.student, group) then
15:                if !notCollided(group, j) ∧ sol < solutionSize then
16:                    if !ifPreferences(assignment.student, group) then
17:                        individual.preferences = 1
18:                    else
19:                        individual.preferences = 0
20:                    end if
21:                    assignment.group = group
22:                    doAssignment(j)
23:                    backtracking(j + 1)
24:                    removeAssignnment(j)
25:                end if
26:            end if
27:        end for
28:    end if
29: end procedure
```

En el caso de que no encuentre solución porque no existe o encuentre menos soluciones de las establecidas en *solutionSize*, el algoritmo no parará hasta mirar todas las posibilidades. Es por esto, que este modo sirve también para saber si existe alguna solución.

Modo aleatorio

Al contrario que en el anterior modo, backtracking termina cuando encuentra una solución. Se realiza así, porque el fin de este modo es que las soluciones sean diferentes, por lo que cuando encuentra una solución no vuelve atrás, sino que empieza otra vez, consiguiendo así, junto con la baraja de los grupos, que las soluciones sean diferentes. Para conseguir el número de soluciones seleccionado x, se ejecuta el modo x veces. De esta forma, al barajar los grupos de cada tipo de clase de las asignaturas, las soluciones serán diferentes. la ejecución del programa se puede ver en el algoritmo 15, y su ejecución en el algoritmo 16.

Algorithm 15 Ejecución programa

```
1: procedure START
2:     while (len solutions < solutionsSize) ∧ attempts < attempsMax do
3:         backtrackingAleatorio(len initialAssignments)
4:     end while
5: end procedure
```

[H] Existe la posibilidad en este modo en el que no se pueda encontrar ninguna solución o menos de las deseadas en *solutionsSize*. Para ello se introduce el número de intentos *attempts*, ya que cada vez que se ejecuta el algoritmo de backtrackingAleatorio, se aumenta el número de intentos, y si llega al máximo de intentos para el algoritmo. Este algoritmo no será capaz de decir si existe alguna solución, a diferencia del modo determinista.

Algorithm 16 Backtracking aleatorio

```
1: procedure BACKTRACKINGALEATORIO(j)
2:     if j = len assignments then
3:         if individual ∉ solutions then
4:             soluciones.add(individual)
5:             sol++
6:         end if
7:     else
8:         assignment = individual.getAssignment(j)
9:         groups = assignment.groups
10:        shuffle groups
11:        for all group ∈ groups do
12:            if !constraint(assignment.student, group) ∧ sol = 0 then
13:                if !notCollided(group,j) then
14:                    if !preferences(assignment.student, group) then
15:                        individual.preferences = 1
16:                    else
17:                        individual.preferences = 0
18:                    end if
19:                    assignment.group = group
20:                    doAssignation(j)
21:                    backtracking(j + 1)
22:                    removeAssignation(j)
23:                end if
24:            end if
25:        end for
26:    end if
27: end procedure
```

Capítulo 6

Planificación del proyecto y resumen de presupuesto

6.1. Planificación temporal

A continuación se muestra la planificación temporal seguida en la ejecución del proyecto, que ha durado 115 días. El proyecto comenzó en la primera reunión con el tutor, el 27 de Enero, y terminó la primera semana de Julio.

El proyecto se ha dividido en distintas fases:

1. **Planificación (27/01/21 - 29/01/21):** Fase inicial en la que el alumno tiene una reunión con el tutor para iniciar el proyecto y su organización.
   a) **Reunión con el tutor (27/01/21 - 27/01/21)**
   b) **Definición de los pasos del proyecto (28/01/21 - 29/01/21)**
2. **Revisión bibliográfica (27/01/21 - 28/02/21):** En esta fase se realiza la lectura y revisión de ciertos materiales recomendados u obligatorios para poder realizar el proyecto. Estos materiales sirven para entender el funcionamiento del sistema inicial y de las nuevas mejoras.
   a) **Revisión del Trabajo de Fin de Grado de Gonzalo De La Cruz Fernández (27/01/21 - 12/02/21):** Esta fase es muy importante ya que encamina el proyecto, leyendo y entendiendo la herramienta en su versión previa.
      1) **Lectura (27/01/21 - 04/02/21)**
      2) **Esquema de funcionamiento (04/02/21 - 12/02/21)**
   b) **Investigación de algoritmos genéticos, voraces y de backtracking (12/02/21 - 28/02/21)**

3. **Análisis (28/02/21 - 16/03/21):** En esta fase se define el problema para así poder formalizarlo y facilitar la siguiente fase.
   - *a*) **Reunión con el cliente y el tutor sobre las mejoras y restricciones (28/02/21 - 28/02/21):** En esta reunión se define las mejoras que se implementan sobre el sistema inicial.
   - *b*) **Familiarización con el código (28/02/21 - 16/03/21):** Se recibe el código, con lo que se pretende que se entienda su funcionamiento.
   - *c*) **Definición de las mejoras (28/02/21 - 05/03/21)**
     1) **Asignaciones iniciales (28/02/21 - 05/03/21)**
     2) **Asignaciones que no son necesarias (28/02/21 - 05/03/21)**
     3) **Restricciones horarias(28/02/21 - 05/03/21)**
     4) **Preferencias horarias(28/02/21 - 05/03/21)**
     5) **Modo individual (28/02/21 - 05/03/21)**
   - *d*) **Esquema de cómo se integran las mejoras (05/03/21 - 14/03/21)**
4. **Implementación (16/03/21 - 17/06/21):** En esta fase se implementan las soluciones definidas en el apartado anterior, tanto las lecturas de los ficheros de entrada como su integración en el sistema.
   - *a*) **Modo grupal (16/03/21 - 25/05/21)**
     1) **Asignaciones iniciales (16/03/21 - 11/04/21)**
        - *a′ Lectura del fichero de entrada(16/03/21 - 26/03/21)*
        - *b′ Integración con el sistema actual (26/03/21 - 11/04/21)*
     2) **Asignaciones que no son necesarias (11/04/21 - 25/04/21)**
        - *a′ Lectura del fichero de entrada(11/04/21 - 17/04/21)*
        - *b′ Integración con el sistema actual (17/04/21 - 25/04/21)*
     3) **Restricciones horarias(25/04/21 - 09/05/21)**
        - *a′ Lectura del fichero de entrada(25/04/21 - 01/05/21)*
        - *b′ Integración con el sistema actual (02/05/21 - 09/05/21)*
     4) **Preferencias horarias(10/05/21 - 25/05/21)**
        - *a′ Integración con el sistema actual (10/05/21 - 25/05/21)*
   - *b*) **Modo individual (06/05/21 - 17/06/21)**
     1) **Backtracking (06/05/21 - 17/06/21)**
        - *a′ Determinista (06/05/21 - 06/06/21)*
        - *b′ Aleatorio (06/06/21 - 17/06/21)*
5. **Pruebas (17/06/21 - 19/06/21)**
   - *a*) **Pruebas unitarias (17/06/21 - 18/06/21)**
     1) **Modo grupal (17/06/21 - 18/06/21)**
        - *a′ Asignaciones iniciales (17/06/21 - 18/06/21)*

- *b′ Asignaciones que no son necesarias (17/06/21 - 18/06/21)*
- *c′ Restricciones horarias (17/06/21 - 18/06/21)*
- *d′ Preferencias horarias (17/06/21 - 18/06/21)*

2) **Modo individual (17/06/21 - 18/06/21)**

- *a′ Backtracking (17/06/21 - 18/06/21)*
  - *Determinista (17/06/21 - 18/06/21)*
  - *Aleatorio (17/06/21 - 18/06/21)*

b) **Pruebas de integración (18/06/21 - 19/06/21)**

1) **Modo grupal (18/06/21 - 19/06/21)**

- *a′ Asignaciones iniciales (18/06/21 - 19/06/21)*
- *b′ Asignaciones que no son necesarias (18/06/21 - 19/06/21)*
- *c′ Restricciones horarias (18/06/21 - 19/06/21)*
- *d′ Preferencias horarias (18/06/21 - 19/06/21)*

2) **Modo individual (18/06/21 - 19/06/21)**

- *a′ Backtracking (18/06/21 - 19/06/21)*
  - *Determinista (18/06/21 - 19/06/21)*
  - *Aleatorio (18/06/21 - 19/06/21)*

6. **Validación del cliente (19/06/21 - 20/06/21):** En esta fase se enviará el resultado al cliente para que valide si el funcionamiento del sistema es el esperado y si se requiere algún cambio.

a) **Reunión con el cliente para que valide los resultados (19/06/21 - 20/06/21)**

7. **Experimentos (20/06/21 - 03/07/21):** Durante esta fase se va a realizar los experimentos de las soluciones implementadas. Para cada una de las mejoras se realizan los experimentos con diferentes instancias para indicar como afecta a su calidad.

a) **Creación de las instancias (20/06/21 - 23/06/21):** En esta fase se van a crear las instancias dependiendo del modo que se vaya a probar.

1) **Modo grupal (20/06/21 - 23/06/21)**
2) **Modo individual (20/06/21 - 21/06/21)**

b) **Experimentos (23/06/21 - 03/07/21)**

1) **Pesos función fitness (23/06/21 - 29/06/21):** Durante esta fase se va a probar pesos para el criterio de cumplimiento de preferencias horarias para determinar qué peso es el más adecuado.
2) **Modo grupal (23/07/21 - 03/07/21):** En esta fase se va a experimentar con las distintas mejoras implementadas en el modo grupal, para así saber como varía la calidad de la solución dependiendo del uso de cada una de ellas en distintas instancias.

3) **Modo individual (23/07/21 - 26/07/21):** Durante esta fase se va a experimentar con distintos estudiantes y distintas matrículas para probar en el modo individual.

8. **Cierre del proyecto (03/07/21 - 13/07/21):** En esta fase final, se define los pasos finales para realizar el cierre del proyecto. Como pasos finales, se revisa tanto el proyecto como su documentación.

   a) **Definición de las ampliaciones (03/07/21 - 04/07/21)**

   b) **Conclusiones (04/07/21 - 04/07/21)**

   c) **Revisión (05/07/21 - 13/07/21)**

      1) **Proyecto (05/07/21 - 08/07/21)**
      2) **Documentación (09/07/21 - 13/07/21)**

Además de esto, periódicamente se realizan reuniones con el tutor para analizar el proyecto, y encaminar las mejoras a realizar.

En las figuras (6.1, 6.2 y 6.3) posteriores se especifican de forma visual la planificación de las fases.

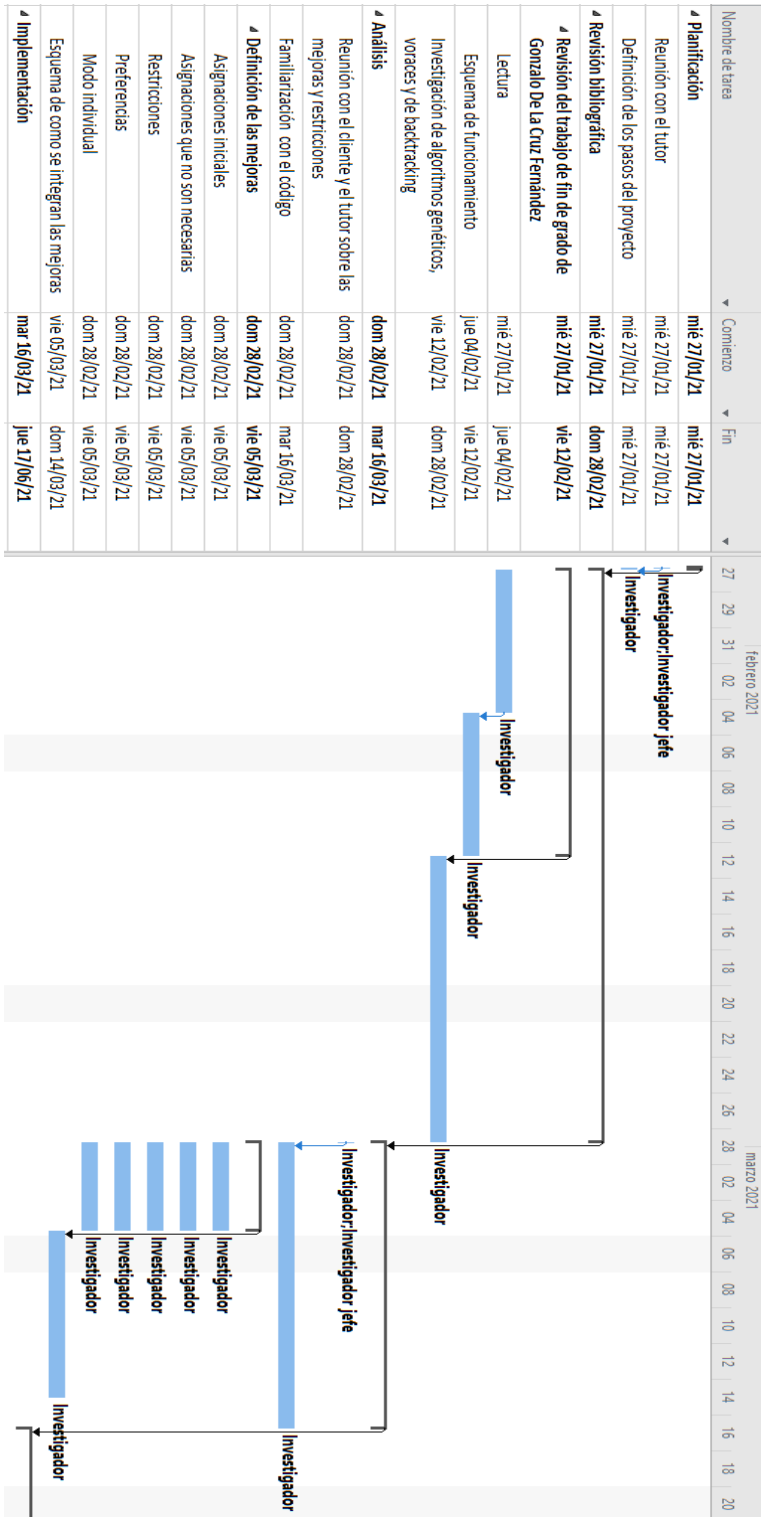

Figura 6.1: Planificación de la primera parte del proyecto.

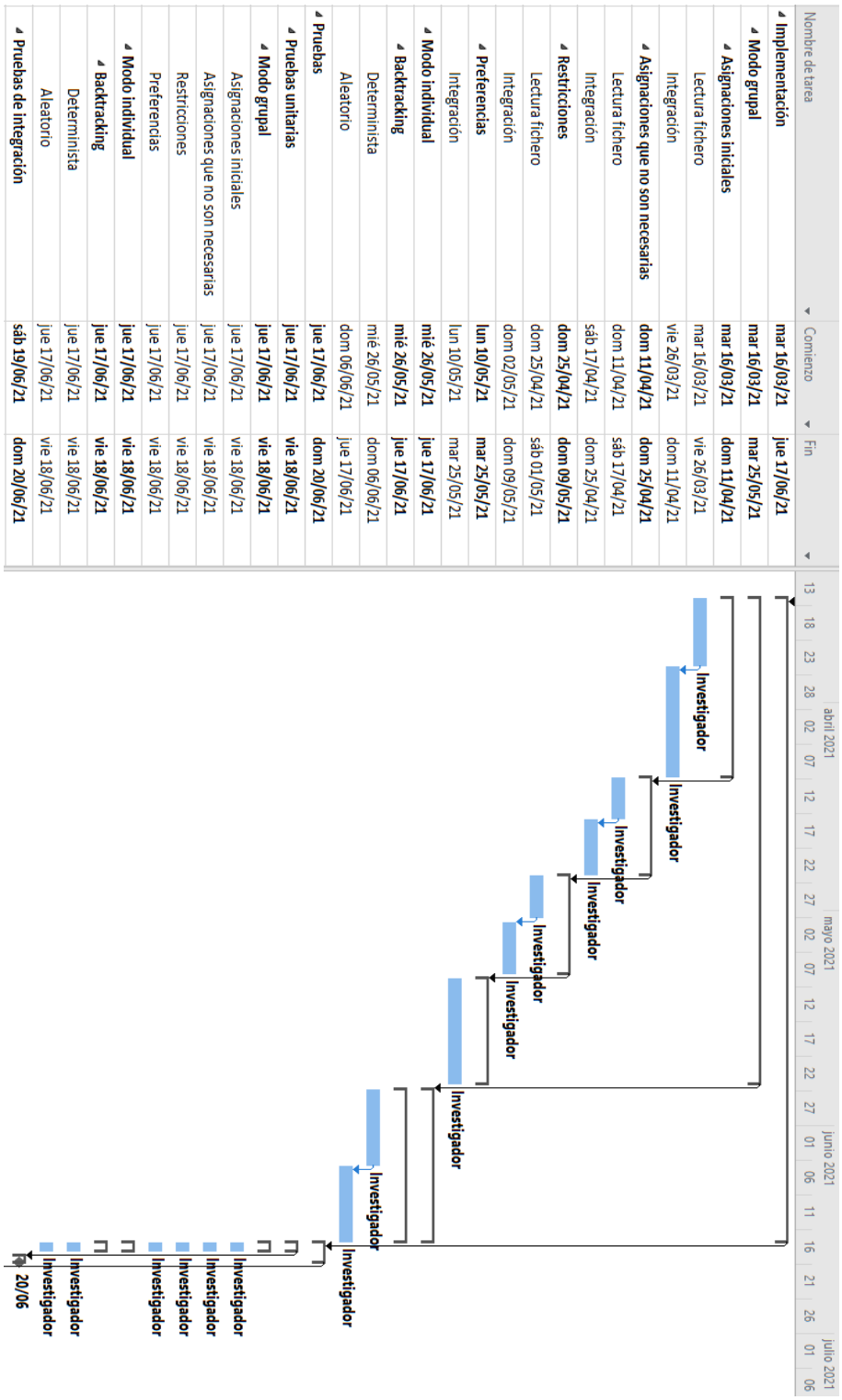

Figura 6.2: Planificación de la segunda parte del proyecto.

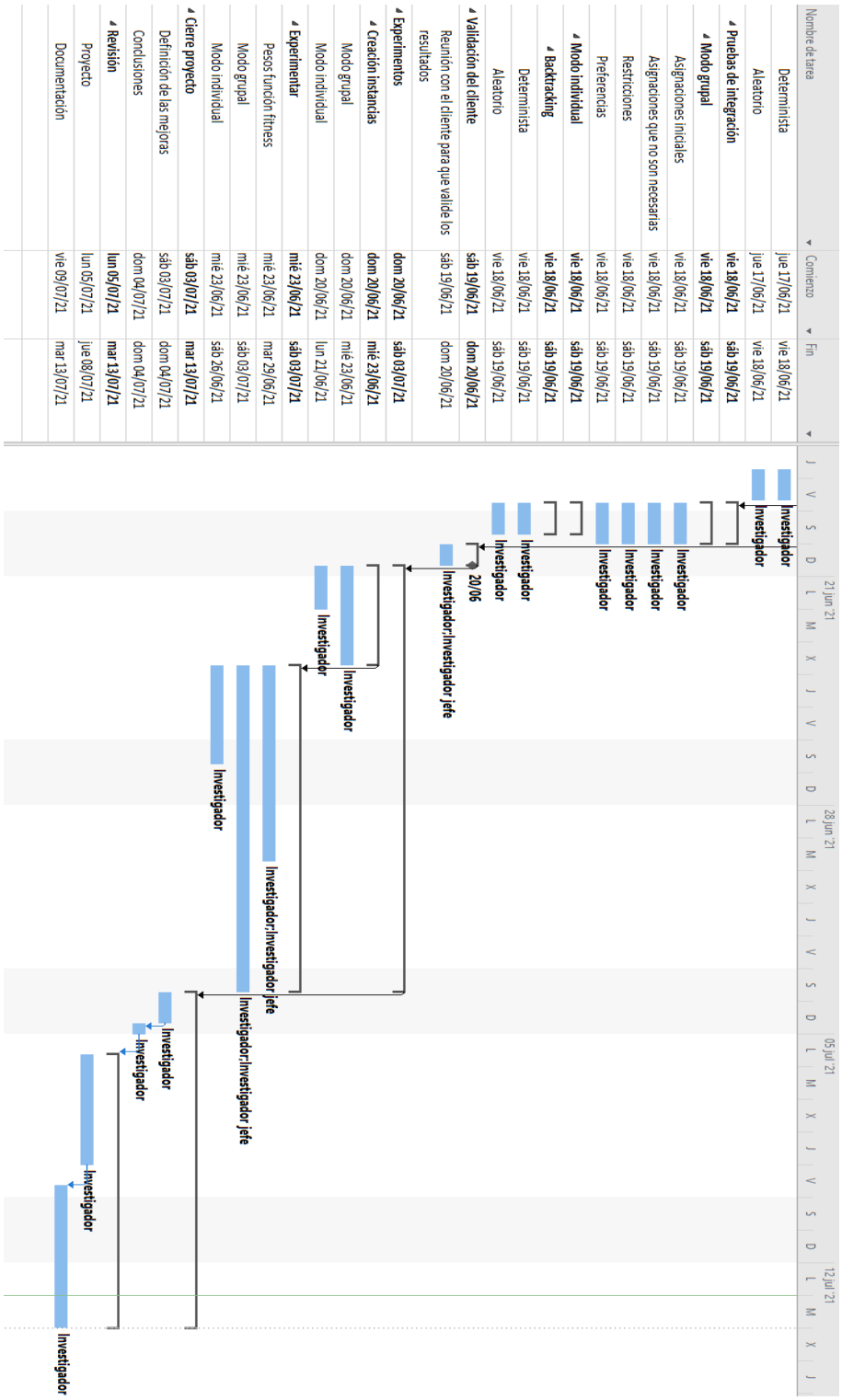

Figura 6.3: Planificación de la tercera parte del proyecto.

6.2. Resumen del presupuesto

Se ha realizado un presupuesto en base a la planificación descrita anteriormente, a pesar de que el fin del proyecto no es el dinero.

El resumen del presupuesto lo podemos encontrar en la figura 6.4, que se divide en 8 partidas. El desarrollo del presupuesto se puede ver en el capítulo 14.

Partidas	Nombre	Importe
1	Planificación	55.88 €
2	Revisión bibliográfica	2,880.00 €
3	Análisis	4,483.00 €
4	Implementación	7,344.00 €
5	Pruebas	1,152.00 €
6	Validación del cliente	39.75 €
7	Experimentos	3,264.00 €
8	Cierre del proyecto	951.00 €
TOTAL		**20,169.63 €**

Figura 6.4: Resumen presupuesto.

Capítulo 7

Análisis

En este capitulo se describe la especificación de los requisitos, además de toda la documentación relativa al análisis del sistema, a partir de la cual se diseñará el sistema.

7.1. Definición del sistema

7.1.1. Alcance del sistema

El objetivo del proyecto es ampliar la herramienta diseñada por Gonzalo De La Cruz Fernández [1], que llamaremos herramienta base, de la forma que ha sido diseñada y descrita en el capitulo 5 para resolver las mejoras propuestas en la herramienta utilizada en la Escuela de Ingeniería Informática de la Universidad de Oviedo para la asignación de alumnos a grupos de clases.

La herramienta base es una aplicación de línea de comandos, que recibe como entrada información las matrículas de los alumnos, el horario de las asignaturas de ese año y unos archivos de configuración. Como lo que se quiere es ampliar la herramienta, la nueva continuará siendo una aplicación de línea de comandos, pero recibirá como información más datos además de los anteriores: el modo que de ejecución (modo grupal o individual), las asignaciones iniciales, las asignaciones que no se deben realizar, las restricciones y las preferencias horarias. El sistema procesará esta información y ejecutará el algoritmo.

Los distintos modos nombrados anteriormente, grupal e individual, se han descrito anteriormente.

Se mantiene también de la herramienta base ciertas cuestiones. En el modo grupal, se proporciona información del estado actual de la ejecución y alguna información sobre la mejor solución encontrada hasta ese momento. Además, al final de la ejecución del algoritmo, el sistema debe generar un informe en el

formato que la escuela utiliza para gestionar las asignaciones y otro informe en un formato legible para la persona encargada de realizar las tareas para comprobar la calidad de la solución generada. En el modo individual, se facilitará información de las soluciones encontradas, generando un informe legible por cada una de ellas para que sea más fácil visualizar el horario, y otro informe con las soluciones juntas e información sobre ellas.

7.2. Requisitos del sistema

Los requisitos que se especifican a continuación, tanto los funcionales como los no funcionales, contienen los considerados en la versión actual de la herramienta, ya que su ampliación requiere mantenerlos.

7.2.1. Requisitos funcionales

Tabla 7.1: Requisitos del sistema

ID	**Nombre requisito**	**Descripción**
RF.1	Modo de la herramienta	El sistema debe permitir al usuario seleccionar el modo que desea de la herramienta
RF.1.1	Modo grupal	El sistema debe tener un modo de asignaciones a estudiantes a sus asignaturas.
RF.1.1.1	Archivos de entrada **(Modo grupal)**	El sistema debe permitir al usuario seleccionar archivos de entrada.
RF.1.1.1.1	Seleccionar archivo de matrícula **(Modo grupal)**	El sistema debe permitir al usuario seleccionar un archivo que contenga la información sobre las inscripciones de los estudiantes.
RF.1.1.1.1.1	Validación del fichero de matrícula **(Modo grupal)**	El sistema debe validar que el formato del archivo de matrícula seleccionado es correcto. Si no lo es, debe mostrarse un mensaje de error.
RF.1.1.1.2	Seleccionar archivo de horarios **(Modo grupal)**	El sistema debe permitir al usuario seleccionar un archivo que contenga la información sobre el horario escolar.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	**Nombre requisito**	**Descripción**
RF.1.1.1.2.1	Validación del archivo de horario **(Modo grupal)**	El sistema debe validar que el formato del archivo de horario seleccionado es correcto. Si no lo es, debe mostrarse un mensaje de error.
RF.1.1.1.3	Seleccionar archivo de asignaciones iniciales **(Modo grupal)**	El sistema debe permitir al usuario seleccionar un archivo que contenga la información sobre las asignaciones iniciales.
RF.1.1.1.3.1	Validación del archivo de asignaciones iniciales **(Modo grupal)**	El sistema debe validar que el formato del archivo de las asignaciones iniciales seleccionado es correcto. Si no lo es, debe mostrarse un mensaje de error.
RF.1.1.1.4	Seleccionar archivo de las asignaciones que no se deben hacer **(Modo grupal)**	El sistema debe permitir al usuario seleccionar un archivo que contenga la información sobre las asignaciones que no son necesarias.
RF.1.1.1.4.1	Validación del archivo de las asignaciones que no se deben hacer **(Modo grupal)**	El sistema debe validar que el formato del archivo de las asignaciones que no se deben hacer seleccionado es correcto. Si no lo es, debe mostrarse un mensaje de error.
RF.1.1.1.5	Seleccionar archivo de las restricciones horarias **(Modo grupal)**	El sistema debe permitir al usuario seleccionar un archivo que contenga la información sobre las restricciones horarias del alumnado.
RF.1.1.1.5.1	Validación del archivo de las restricciones horarias **(Modo grupal)**	El sistema debe validar que el formato del archivo de restricciones horarias seleccionado es correcto. Si no lo es, debe mostrarse un mensaje de error.
RF.1.1.1.6	Seleccionar archivo de las preferencias horarias **(Modo grupal)**	El sistema debe permitir al usuario seleccionar un archivo que contenga la información sobre las preferencias del alumnado.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	**Nombre requisito**	**Descripción**
RF.1.1.1.6.1	Validación del archivo de las preferencias horarias **(Modo grupal)**	El sistema debe validar que el formato del archivo de preferencias horarias seleccionado es correcto. Si no lo es, debe mostrarse un mensaje de error.
RF.1.1.2	Archivos de configuración **(Modo grupal)**	El sistema debe permitir al usuario modificar la configuración del sistema.
RF.1.1.2.1	Configuración de las excepciones de colisión **(Modo grupal)**	El sistema debe permitir al usuario introducir excepciones de colisión entre grupos de clases.
RF.1.1.2.1.1	Formato del archivo de configuración de excepciones de colisión **(Modo grupal)**	Los archivos de configuración de las excepciones de colisión deben ser un archivo de texto plano en el que cada fila estará compuesta por una serie de grupos de clases separados por símbolos de punto y coma. Un ejemplo de fila: $ED.S.I-1; AC.S.I-1$ Los grupos de clase que componen una fila no colisionarán entre sí en la ejecución del algoritmo.
RF.1.1.2.1.2	Archivo de excepciones de colisión validación **(Modo grupal)**	El sistema debe validar que el formato del archivo de configuración de excepción de colisiones es correcto.
RF.1.1.2.2	Configuración de las colisiones obligatorias **(Modo grupal)**	El sistema debe permitir al usuario introducir colisiones obligatorias entre grupos de clases.
RF.1.1.2.2.1	Formato del archivo de configuración de colisiones obligatorias **(Modo grupal)**	Los archivos de configuración de colisiones obligatorias deben ser un archivo texto plano en el que cada fila estará compuesta por una serie de grupos de clase separados por símbolos de punto y coma. Un ejemplo de fila: $AL.T$,1; $AL.S$,3 Los grupos de clases que componen una fila siempre colisionan entre sí en la ejecución del algoritmo.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	Nombre requisito	Descripción
RF.1.1.2.2.2	Validación del fichero de colisiones obligatorias **(Modo grupal)**	El sistema debe validar que el archivo de configuración de colisiones obligatorias es correcto.
RF.1.1.2.3	Configuración de preferencias de asignación **(Modo grupal)**	El sistema debe permitir al usuario introducir preferencias de asignación para diferentes grupos de clases.
RF.1.1.2.3.1	Formato de los archivos de configuración de preferencias de asignación **(Modo grupal)**	Los archivos de configuración de preferencias deben ser un archivo de texto plano en el que cada fila estará compuesta por una serie de grupos de clase separados por símbolos de punto y coma. Un ejemplo de fila: $Cal.S$,1; $AL.S$,1; $IP.S$,1; $Emp.S$,1 Durante la ejecución del algoritmo, cuando el primer grupo de clase que compone una fila se asigna, los demás grupos de clase de la fila también lo hacen.
RF.1.1.2.3.2	Validación del fichero de preferencias de asignación **(Modo grupal)**	El sistema debe validar que el formato del fichero de sea correcto.
RF.1.1.3	Asignación de alumnos **(Modo grupal)**	Para cada alumno, el sistema debe asignar grupos de clase para cada una de las asignaturas en las que está matriculado el alumno.
RF.1.1.3.1	Colisiones en la asignación de alumnos **(Modo grupal)**	Un estudiante no debe tener dos asignaciones que colisionen entre sí.
RF.1.1.3.2	Lenguaje de matriculación **(Modo grupal)**	El sistema sólo debe asignar a un estudiante grupos de clase grupos en los que las lecciones se imparten en el idioma en el que se haya matriculado en esa asignatura.
RF.1.1.3.3	Asignaciones de alumnos no resueltas **(Modo grupal)**	El sistema debe mostrar al usuario las asignaciones que el algoritmo no ha podido realizar.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	**Nombre requisito**	**Descripción**
RF.1.1.4	Grupos de clase **(Modo grupal)**	El sistema debe intentar que los alumnos se distribuyan entre todos los posibles grupos de clase de una asignatura.
RF.1.1.4.1	Número homogéneo de estudiantes en grupos de clase **(Modo grupal)**	El sistema debe intentar que el número de estudiantes en los grupos de clase de un tipo de asignatura de una asignatura sea homogéneo.
RF.1.1.4.2	Número máximo de estudiantes en un grupo de clase **(Modo grupal)**	El sistema no puede asignar a un alumno a un grupo de clase grupo de clase que ya ha alcanzado el límite de estudiantes permitidos.
RF.1.1.5	Cumplimiento de las asignaciones iniciales **(Modo grupal)**	El sistema debe garantizar que se cumpla las asignaciones iniciales.
RF.1.1.6	Asignaciones que no se deben realizar **(Modo grupal)**	El sistema debe garantizar que no se le asigne a los estudiantes en grupos de clase que no sean necesarios.
RF.1.1.7	Cumplimiento de las restricciones horarias **(Modo grupal)**	El sistema debe garantizar que no se le asigne a los estudiantes en grupos de clase que se encuentren dentro de sus restricciones horarias.
RF.1.1.8	Calidad del horario de los estudiantes **(Modo grupal)**	El sistema debe buscar una buena calidad de horario de los estudiantes.
RF.1.1.8.1	Horario de estudiantes libre franjas horarias **(Modo grupal)**	El sistema debe minimizar el número de espacios de tiempo libres que tiene un estudiante entre clases.
RF.1.1.8.2	Días con bajo número de clases en el horario de los estudiantes **(Modo grupal)**	El sistema debe intentar que los estudiantes tengan el número mínimo de días con menos de dos horas de clase durante la semana.
RF.1.1.8.3	Horario de estudiantes con preferencias horarias **(Modo grupal)**	El sistema debe garantizar que los estudiantes tengan el número mayor de preferencias horarias satisfechas.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	Nombre requisito	Descripción
RF.1.1.9	Información durante la ejecución **(Modo grupal)**	El sistema debe mostrar información sobre el estado de la ejecución mientras se ejecuta el algoritmo.
RF.1.1.9.1	Información de la generación **(Modo grupal)**	El sistema debe proporcionar información del estado del algoritmo genético en cada generación.
RF.1.1.9.1.1	Información sobre las asignaciones no resueltas **(Modo grupal)**	El sistema debe proporcionar la lista de asignaciones no resueltas del mejor individuo. Para cada una de ellas debe proporcionarse la siguiente información: ▪ Nombre del alumno. ▪ Nombre del tipo de clase que no pudo ser asignada.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	**Nombre requisito**	**Descripción**
RF.1.1.9.1.2	Información sobre la población **(Modo grupal)**	El sistema debe proporcionar información sobre la población actual. En cada generación, el algoritmo debe proporcionar la siguiente información sobre la población actual: ▪ Número de generación de la población. ▪ Aptitud del mejor individuo de la población. ▪ Aptitud media de los individuos de la población. ▪ Número medio de asignaciones no resueltas. ▪ Media de la diferencia máxima entre los grupos de clase que tienen el máximo y el mínimo número de alumnos en un tipo de clase de los individuos de la población. ▪ Media de la suma de las diferencias entre los grupos de clase que tienen el número máximo y el número mínimo de alumnos en un tipo de clase de los individuos de la población. ▪ Media del número total de huecos libres entre lecciones de los individuos de la población. ▪ Media del número total de días con 2 horas de lecciones o menos de los individuos de la población. ▪ Media del número total de preferencias horarias no satisfechas.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	Nombre requisito	Descripción
RF.1.1.9.1.3	Información sobre la mejor solución encontrada **(Modo grupal)**	El sistema debe proporcionar información sobre las características del mejor individuo encontrado hasta este momento. En cada generación, el algoritmo debe proporcionar la siguiente información sobre la mejor solución: ▪ Número de asignaciones no resueltas. ▪ Diferencia máxima entre los grupos de clases que tienen el máximo y el mínimo número de estudiantes en un tipo de clase. ▪ Suma de las diferencias entre los grupos de clase grupos de clase que tienen el máximo y el mínimo número de estudiantes en un tipo de clase de los individuos de la población. ▪ Varianza máxima entre los grupos de clase de un tipo de clase. ▪ Varianza total entre los grupos de clase de tipo de clases. ▪ Número total de huecos libres entre clases. ▪ Número total de días con dos horas de lecciones o menos. ▪ Número total de preferencias horarias no satisfechas.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	**Nombre requisito**	**Descripción**
RF.1.1.9.2	Información de la ejecución **(Modo grupal)**	El sistema debe mostrar información sobre la evolución de la ejecución mientras se ejecuta el algoritmo.
RF.1.1.9.2.1	Gráfico de evolución de la aptitud **(Modo grupal)**	El sistema debe mostrar un gráfico que muestre la evolución de la mejor aptitud y la aptitud media de la población en cada generación. El gráfico debe ser actualizado en tiempo real al final de cada generación.
RF.1.1.9.2.2	Gráfico de la evolución de las evolución de las asignaciones **(Modo grupal)**	El sistema debe mostrar un gráfico que muestre la evolución del número medio de asignaciones no resueltas de la población en cada generación. El gráfico debe actualizarse en tiempo real al final de cada generación.
RF.1.1.9.2.3	Gráfico de evolución de la varianza **(Modo grupal)**	El sistema debe mostrar un gráfico que muestre la evolución de la varianza entre grupos del mejor individuo de la población en cada generación. El gráfico debe actualizarse en tiempo real al final de cada generación.
RF.1.1.9.2.4	Gráfico de evolución de la asignaciones que no satisfacen las preferencias horarias **(Modo grupal)**	El sistema debe mostrar un gráfico que muestre la evolución de las preferencias horarias no satisfechas del mejor individuo de la población en cada generación. El gráfico debe actualizarse en tiempo real al final de cada generación.
RF.1.1.10	Informes **(Modo grupal)**	El sistema debe generar informes con los resultados de la ejecución cuando el algoritmo haya finalizado.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	Nombre requisito	Descripción
RF.1.1.10.1	Informe de Excel **(Modo grupal)**	El sistema debe generar un informe de Excel en el formato utilizado por la Escuela de Ingeniería Informática de la Universidad de Oviedo. Cada fila del excel representará a un alumno. Para un alumno se debe proporcionar la siguiente información en el siguiente orden. 1. ID del estudiante. 2. Nombre del estudiante. 3. Apellido del alumno. 4. Asignación del alumno en cada una de las asignaturas. Si el alumno está matriculado en la asignatura, el número del grupo asignado se muestra en la celda. En caso contrario, se muestra el símbolo ".
RF.1.1.10.2	Orden de las asignaciones de los alumnos **(Modo grupal)**	En el informe de Excel, las asignaciones de los alumnos deben aparecer ordenadas por asignaturas. Las asignaturas deben aparecer en el siguiente orden: ▪ **Primer semestre**: AL, Cal, Emp, FI, IP, AC, Com, CPM, ED, TEC, DS, IPS, RI, SEW,CVVS, IR, SI, IA, IFA, IAE, RAA, SIW, SEV, SDM, SRF. ▪ **Segundo semestre**: AMD, Est, FCR, MP, OyE, Alg, BD, CN, SO, TPP, ASR, ASW, DLP, SSI, SDI, ASLEPI, DPPI.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	**Nombre requisito**	**Descripción**
RF.1.1.10.3	Informe textual **(Modo grupal)**	El sistema debe generar un informe mediante archivos de texto plano que muestre las mejores asignaciones de soluciones.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	Nombre requisito	Descripción
RF.1.1.10.3.1	Informe resumido **(Modo grupal)**	El sistema debe generar un archivo texto plano que muestre información de la mejor solución encontrada durante la ejecución del algoritmo. Este informe debe contener la siguiente información ▪ Número de asignaciones no resueltas. ▪ Diferencia máxima entre los grupos de clases que tienen el máximo y el mínimo número de alumnos en un tipo de clase. ▪ Suma de las diferencias entre los grupos de clase que tienen el máximo y el mínimo número de alumnos en un tipo de clase de los individuos de la población. ▪ Varianza máxima entre los grupos de clase de un tipo de clase. ▪ Varianza total entre los grupos de clase de tipo de clases. ▪ Número total de huecos libres entre clases. ▪ Número total de días con dos horas de lecciones o menos. ▪ Número total de días preferencias horarias no satisfechas.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	**Nombre requisito**	**Descripción**
RF.1.1.10.2.1	Informes de los alumnos **(Modo grupal)**	El sistema debe generar un archivo de texto plano para cada alumno que muestre la información personal del mismo, sus asignaciones de grupo en las asignaturas en las que está matriculado y su horario semanal.
RF.1.1.10.2.2	Informes de asignaturas **(Modo grupal)**	El sistema debe generar un archivo de texto plano para cada una de las asignaturas, mostrando el número de alumnos que están asignados a cada uno de los grupos de clase de los diferentes tipo de clases de la asignatura.
RF.1.1.10.2.3	Informe de asignaciones no resueltas **(Modo grupal)**	El sistema debe generar un archivo de texto plano que muestre las asignaciones que no han podido ser completadas. Para cada tarea no resuelta debe proporcionarse la siguiente información: ▪ Nombre del alumno. ▪ Nombre del tipo de clase que no pudo ser asignada.
RF.1.2	Modo asignación a un estudiante	El sistema debe tener un modo de asignación, que por un estudiante se den varias soluciones posibles a sus asignaciones.
RF.1.2.1	Archivos de entrada **(Modo individual)**	El sistema debe permitir al usuario seleccionar archivos como de entrada.
RF.1.2.1.1	Seleccionar archivo de matrícula **(Modo individual)**	El sistema debe permitir al usuario seleccionar un archivo que contenga la información sobre las matriculas del estudiante.
RF.1.2.1.1.1	Validación del fichero de matrícula **(Modo individual)**	El sistema debe validar que el formato del archivo de matrícula seleccionado es correcto. Si no lo es, debe mostrarse un mensaje de error.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	Nombre requisito	Descripción
RF.1.2.1.2	Seleccionar archivo de horarios **(Modo individual)**	El sistema debe permitir al usuario seleccionar un archivo que contenga la información sobre el calendario universitario.
RF.1.2.1.2.1	Validación del archivo de horarios **(Modo individual)**	El sistema debe validar que el formato del archivo de horarios seleccionado es correcto. Si no lo es, debe mostrarse un mensaje de error.
RF.1.2.1.3	Seleccionar archivo de asignaciones iniciales **(Modo individual)**	El sistema debe permitir al usuario seleccionar un archivo que contenga la información sobre las asignaciones iniciales.
RF.1.2.1.3.1	Validación del archivo de asignaciones iniciales **(Modo individual)**	El sistema debe validar que el formato del archivo de las asignaciones iniciales seleccionado es correcto. Si no lo es, debe mostrarse un mensaje de error.
RF.1.2.1.4	Seleccionar archivo de las asignaciones que no se deben hacer **(Modo individual)**	El sistema debe permitir al usuario seleccionar un archivo que contenga la información sobre las asignaciones que no son necesarias.
RF.1.2.1.4.1	Validación del archivo de las asignaciones que no se deben hacer **(Modo individual)**	El sistema debe validar que el formato del archivo de las asignaciones que no se deben hacer seleccionado es correcto. Si no lo es, debe mostrarse un mensaje de error.
RF.1.2.1.5	Seleccionar archivo de las restricciones **(Modo individual)**	El sistema debe permitir al usuario seleccionar un archivo que contenga la información sobre las restricciones del alumnado.
RF.1.2.1.5.1	Validación del archivo de las restricciones **(Modo individual)**	El sistema debe validar que el formato del archivo de restricciones seleccionado es correcto. Si no lo es, debe mostrarse un mensaje de error.
RF.1.2.1.6	Seleccionar archivo de las preferencias horarias **(Modo individual)**	El sistema debe permitir al usuario seleccionar un archivo que contenga la información sobre las preferencias del alumnado.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	**Nombre requisito**	**Descripción**
RF.1.2.1.6.1	Validación del archivo de las preferencias horarias **(Modo individual)**	El sistema debe validar que el formato del archivo de preferencias horarias seleccionado es correcto. Si no lo es, debe mostrarse un mensaje de error.
RF.1.2.2	Archivos de configuración **(Modo individual)**	El sistema debe permitir al usuario modificar la configuración del sistema.
RF.1.2.2.1	Configuración de las excepciones de colisión **(Modo individual)**	El sistema debe permitir al usuario introducir excepciones de colisiones entre grupos de clases.
RF.1.2.2.1.1	Formato del archivo de configuración de excepciones de colisión **(Modo individual)**	Los archivos de configuración de las excepciones de colisiones deben ser un archivo de texto plano en el que cada fila estará compuesta por una serie de grupos de clases separados por símbolos de punto y coma. Un ejemplo de fila: $ED.S.I-1; AC.S.I-1$ Los grupos de clase que componen una fila no colisionarán entre sí en la ejecución del algoritmo.
RF.1.2.2.1.2	Validación del archivo de excepciones de colisiones **(Modo individual)**	El sistema debe validar que el formato del archivo de configuración de excepciones de colisiones es correcto.
RF.1.2.2.2	Configuración del archivo de colisiones obligatorias **(Modo individual)**	El sistema debe permitir al usuario introducir colisiones obligatorias entre grupos de clases.
RF.1.2.2.2.1	Formato del archivo de configuración de colisiones obligatorias **(Modo individual)**	Los archivos de configuración de colisiones obligatorias deben ser un archivo de texto plano en el que cada fila estará compuesta por una serie de grupos de clase separados por símbolos de punto y coma. Un ejemplo de fila: $AL.T$,1; $AL.S$,3 Los grupos de clases que componen una fila siempre colisionan entre sí en la ejecución del algoritmo.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	Nombre requisito	Descripción
RF.1.2.2.2.2	Validación del fichero de colisiones obligatorias **(Modo individual)**	El sistema debe validar que el archivo de configuración de colisiones obligatorias es correcto.
RF.1.2.2.3	Configuración de preferencias de asignación **(Modo individual)**	El sistema debe permitir al usuario introducir preferencias de asignación para diferentes grupos de clases.
RF.1.2.2.3.1	Formato del archivo de configuración de preferencias de asignación **(Modo individual)**	Los archivos de configuración de preferencias deben ser un archivo de texto plano en el que cada fila estará compuesta por una serie de grupos de clase separados por símbolos de punto y coma. Un ejemplo de fila: $Cal.S$,1; $AL.S$,1; $IP.S$,1; $Emp.S$,1 Durante la ejecución del algoritmo, cuando el primer grupo de clase que compone una fila se asigna, los demás grupos de clase de la fila también lo hacen.
RF.1.2.2.3.2	Validación del fichero de preferencias de asignación **(Modo individual)**	El sistema debe validar que el formato del fichero de sea correcto.
RF.1.2.2.4	Configuración del algoritmo de backtracking	El sistema debe permitir al usuario introducir configuraciones en el algoritmo.
RF.1.2.2.4.1	Formato del archivo de configuración de backtracking	El archivo de configuración de backtracking debe ser un archivo properties en el que se introducen las configuraciones que se deben realizar. Esas configuraciones pueden ser: número de soluciones que se desea, modo que se desea ejecutar del algoritmo (determinista o aleatorio), número de intentos y semestre. En el caso de que no existan se utilizarán los valores predeterminados.
RF.1.2.3	Asignaciones del alumno **(Modo individual)**	El sistema debe asignar grupos de clase para cada una de las asignaturas en las que está matriculado el alumno.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	**Nombre requisito**	**Descripción**
RF.1.2.3.1	Colisiones en la asignación del alumno **(Modo individual)**	El estudiante no debe tener dos asignaciones que colisionen entre sí.
RF.1.2.3.2	Lenguaje de matriculación **(Modo individual)**	El sistema sólo debe asignar al estudiante grupos de clase en los que las lecciones se imparten en el idioma en el que se haya matriculado en esa asignatura.
RF.1.2.4	Cumplimiento de las asignaciones iniciales **(Modo individual)**	El sistema debe garantizar que se cumpla las asignaciones iniciales.
RF.1.2.5	Asignaciones que no se deben realizar **(Modo individual)**	El sistema debe garantizar que no se le asigne al estudiante en grupos de clase que no sean necesarios.
RF.1.2.6	Cumplimiento de las restricciones horarias **(Modo individual)**	El sistema debe garantizar que no se le asigne al estudiante en grupos de clase que se encuentren dentro de sus restricciones horarias.
RF.1.2.7	Informes **(Modo individual)**	El sistema debe generar informes con los resultados de la ejecución cuando el algoritmo haya finalizado.

Continua en la siguiente página

Tabla 7.1 – *Continua de la anterior página*

ID	Nombre requisito	Descripción
RF.1.2.7.1	Informe de Excel **(Modo individual)**	El sistema debe generar un informe de Excel en el que en la primera fila se encontrará el alumno y se debe proporcionar la siguiente información en el siguiente orden. 1. ID del estudiante. 2. Nombre del estudiante. 3. Apellido del alumno. 4. Asignación del alumno en cada una de las asignaturas. Si el alumno está matriculado en la asignatura.- Debajo se va a encontrar cierta de información de la solución 1. Número de horas libres. 2. Días de la semana con dos horas o menos de clases. 3. Número total de preferencias que no se satisfacen. 4. Asignación del alumno en cada una de las asignaturas. En cada hoja del excel se encuentra una de las opciones.
RF.1.2.7.2	Informe textual **(Modo individual)**	El sistema debe generar un informe mediante archivos de texto plano que muestre las soluciones.
RF.1.2.7.2.1	Informes del alumno **(Modo individual)**	El sistema debe generar un archivo texto plano para el alumno que muestre la información personal del mismo en cada solución encontrada, sus asignaciones de grupo en las asignaturas en las que está matriculado y su horario semanal.

7.2.2. Requisitos no funcionales

ID	Nombre requisito	Descripción
NFR.1	Usabilidad - UI	El sistema debe ser fácil de usar para las personas que están acostumbradas a trabajar con interfaces de usuario basadas en texto.
NFR.2	Rendimiento	El sistema debe dar soluciones de buena calidad en menos de un día.

7.2.3. Actores del sistema

En este sistema podemos identificar únicamente un actor, que es la persona encargado de hacer la asignación de los alumnos a los grupos de clase en la Escuela de Ingeniería Informática de la Universidad de Oviedo. Será, además, el encargado de configurar el sistema y lo ejecutará. Una vez que finalice el proceso, deberá comprobar los resultados obtenidos y resolver manualmente las asignaciones que no se han podido realizar.

7.3. Identificación de los subsistemas en la fase de análisis

7.3.1. Descripción de los Subsistemas

A continuación se describen los diferentes subsistemas en los que podemos descomponer el sistema, de forma que nos facilitará su análisis.

- **Subsistema de Parser (parser)**: Este subsistema se encarga de leer los ficheros que recibe el algoritmo como entrada. Analiza ciertos archivos:
  - Información sobre el horario.
  - Información de las matriculas de los alumnos.
  - Información sobre las asignaciones iniciales.
  - Información sobre las asignaciones que no se deben realizar.
  - Información sobre las restricciones horarias.
  - Información sobre las preferencias horarias.
  - Configuración del sistema

  También se encarga de detectar errores en el formato de todos los ficheros analizados.

- **Subsistema de Algoritmo Genético (genetic)**: Este subsistema contiene toda la funcionalidad relacionada con el Algoritmo Genético que se describe en el capitulo 5. A su vez está compuesto por otros subsistemas:
  - **Subsistema de Codificación (encoder)**: Este subsistema se encarga de la codificación y decodificación de los individuos en el Algoritmo Genético.
  - **Subsistema de Selección (selection)**: Este subsistema contiene el operador de selección que se utiliza en el Algoritmo Genético.
  - **Subsistema de Cruce (crossover)**: Este subsistema contiene el operador de cruce que se utiliza en el Algoritmo Genético.
  - **Subsistema de Mutación (mutation)**: Este subsistema contiene el operador de mutación que se utiliza en el Algoritmo Genético.
  - **Subsistema de Fitness (fitness)**: Este subsistema contiene la función fitness que se utiliza en el Algoritmo Genético.
- **Subsistema de Algoritmo Voraz (greedy)**: Este subsistema contiene toda la lógica de negocio del Algoritmo Voraz utilizado para realizar las asignaciones y también otros elementos que son necesarios para esta tarea como la Matriz de Colisiones.
- **Subsistema de Infomres (reports)**: Este subsistema se encarga de la generación de informes mientras se ejecuta el algoritmo y al final de la ejecución. Genera una salida de consola y muestra gráficos sobre la evolución del algoritmo en tiempo real en el caso de que sea el modo grupal y también genera archivos de salida con los resultados al final de la ejecución.
- **Subsistema de Algoritmo de Backtracking (backtracking)**: Este subsistema contiene toda la lógica del Algoritmo de Backtracking utilizado para realizar las asignaciones. Además cuenta con otros elementos que son necesarios para poder realizarlas.

7.4. Diagrama de clases preliminar del análisis

A continuación se describen las clases identificadas en la fase de análisis y son una aproximación de las clases finales.

7.4.1. Diagrama de Clases

El diagrama de clases preliminar lo podemos encontrar en la figura 7.1.

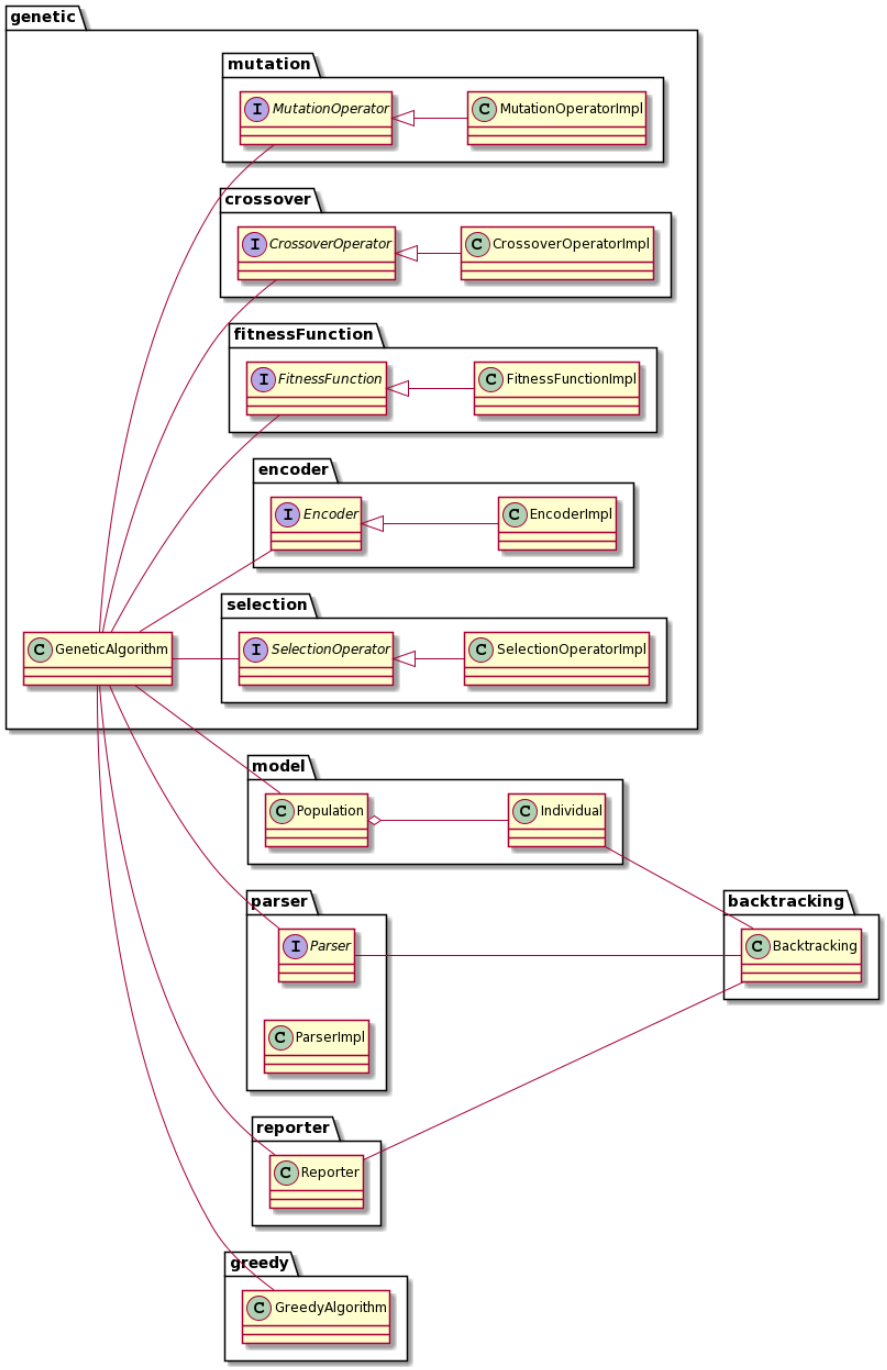

Figura 7.1: Diagrama preliminar de clases.

7.4.2. Descripción de las Clases

En esta sección se describen las clases más importantes de los subsistemas.

Subsistema de Algoritmo Voraz (greedy)

Algoritmo Voraz
Descripción
Clase que se encarga de evaluar las soluciones proporcionadas por el Algoritmo Genético.
Responsabilidades
Realiza las asignaciones de los alumnos a los grupos de clase tomando como entrada la lista de asignaciones proporcionada por el Algoritmo Genético. Por último, después de realizar las asignaciones, evalúa ciertos factores de calidad de la solución que ha generado, que van a ser usados en la función fitness del Algoritmo Genético.
Atributos propuestos
▪ **Matriz de colisiones**: Matriz booleana que representa las colisiones entre los grupos de clases. ▪ **Asignaciones iniciales**: Representación las asignaciones iniciales de cada alumno que se deben cumplir. ▪ **Restricciones horarias**: Representación los grupos por los que un alumno no puede estar asignado al ser una restricción. ▪ **Preferencias horarias**: Representación los grupos en los que un alumno es preferible que no esté asignado al ser una preferencia horaria.
Métodos propuestos
▪ **execute(individual)**: ejecuta el Algoritmo Voraz sobre el individuo que ha sido pasado como parámetro.

Subsistema de Informes (reporter)

Reporter
Descripción
Genera informes del estado de la ejecución mientras el algoritmo se ejecuta y además, muestra los resultados finales obtenidos cuando finaliza.
Responsabilidades
Generar informes sobre la situación actual del algoritmo y sobre las soluciones encontradas en tiempo real en cada generación. Además, cuando se termina la ejecución, genera un informe en excel y otro en txt que muestra la mejor solución encontrada.
Atributos propuestos
-
Métodos propuestos
▪ **showGenerationReport(population)**: Muestra la información sobre la población evaluada en la última generación, incluyendo información sobre el mejor individuo y la calidad media de los individuos de la población. ▪ **generateExcelReport(bestIndividual)**: Genera un informe excel siguiendo el formato descrito. ▪ **generateTxtReport(bestIndividual)**: Genera un informe txt.

Subsistema de Algoritmo Genético (genetic)

Algoritmo Genético
Descripción
Se encarga de ejecutar el algoritmo genético con las entradas cargadas siguiendo el procedimiento descrito en la sección 5.4.
Responsabilidades
Coordinar las llamadas a los diferentes operadores y componentes del Algoritmo Genético. Además, cuando termine su ejecución, debe llamar a los componentes que realizan los informes finales.
Atributos propuestos
▪ **Codificador**: Codificador y decoficador que será utilizado. ▪ **Operador de selección**: Operador de selección que será utilizado. ▪ **Operador de mutación**: Operador de mutación que será utilizado. ▪ **Operador de cruce**: Operador de cruce que será utilizado. ▪ **Función fitness**: Función fitness que va a ser utilizada en la ejecución. ▪ **Población**: Individuos que el algoritmo maneja en cada generación. ▪ **Reporter**: Componente que genera informes durante el proceso y al terminar.
Métodos propuestos
▪ **execute()**: Lanza la ejecución del Algoritmo Genético

Subsistema de Función Fitness (fitness)

Función Fitness
Descripción
Calcula el fitness de cada individuo.
Responsabilidades
Evaluar el fitness de las soluciones generadas por el Algoritmo Voraz.
Atributos propuestos
-
Métodos propuestos
▪ **calculateFitness(individual)**: Calcula el fitness del individuo pasado por parámetro.

Subsistema de Operador de Selección (selection)

Operador de Selección
Descripción
Proporciona el operador de selección del Algoritmo Genético.
Responsabilidades
Selecciona entre los individuos de la población que se pasa como parámetro.
Atributos propuestos
-
Métodos propuestos
▪ **selectPairs(population)**: Selecciona los pares de individuos que van a ser cruzados dentro de la población.

Subsistema de Operador de Cruce (crossover)

Operador de Cruce
Descripción
Proporciona el operador de cruce del Algoritmo Genético.
Responsabilidades
Cruza los cromosomas que recibe como parámetro.
Atributos propuestos
-
Métodos propuestos
▪ **crossover(individual1, individual2)**: Cruza los individuos que recibe como parámetro, generando dos nuevos individuos.

Subsistema de Operador de Mutación (mutation)

Operador de Mutación
Descripción
Proporciona el operador de mutación del Algoritmo Genético.
Responsabilidades
Mutar el cromosoma que recibe como parámetro.
Atributos propuestos
-
Métodos propuestos
▪ **mutate(individual)**: Muta el individuo que recibe como parámetro.

Subsistema de codificación (encoder)

Codificador
Descripción
Proporciona las funciones de codificación y decodificación que usa el Algoritmo Genético.
Responsabilidades
Codificar y decodificar individuos como 5.4.1.
Atributos propuestos
▪ **allAssignments**: lista que contiene todas las asignaciones que el algoritmo debe realizar y que se utiliza como referencia para la codificación y decodificación de las soluciones.
Métodos propuestos
▪ **encode(chromosomeDecoded)**: Codifica la lista de asignaciones como una matriz de enteros que representa la posición de cada asignación en la lista allAssignments. ▪ **decode(chromosomeEncoded)**: Decodifica una matriz de enteros que representa la posición de cada asignación en la lista allAssignments que se pasa como parámetro en una lista de asignaciones siguiendo el orden indicado en la matriz de enteros.

Subsistema de parseo (parser)

Parser
Descripción
Procesa la entrada y los archivos de configuración, proporcionando esta información al sistema. Además, detecta si existe algún tipo de error en los ficheros.
Responsabilidades
Procesar la entrada y los archivos de configuración, detectando además lo máximo posible los errores de formato.
Atributos propuestos
-
Métodos propuestos
▪ **parsePlanification(pathToFile)**: Analiza los ficheros que contienen la información de los estudiantes. ▪ **parseGroupTimetable(pathToFile)**: Analiza el fichero que contiene la información de los horarios de los grupos. ▪ **parseMandatoryCollisions(pathToFile)**: Analiza el fichero que tiene información de las colisiones obligatorias. ▪ **parseCollisionExceptions(pathToFile)**: Analiza el fichero que tiene información de las colisiones. ▪ **parsePreferences(pathToFile)**: Analiza el fichero que tiene información de las preferencias. ▪ **parseGeneticAlgorithmParams(pathToFile)**: Analiza el fichero que tiene información de la configuración de los parámetros del Algoritmo Genético. ▪ **parseInitialAssignment(pathToFile)**: Analiza el fichero que tiene las asignaciones iniciales. ▪ **parseNotAssignment(pathToFile)**: Analiza el fichero que tiene información de los tipos de clase que no se debe realizar asignación de una asignatura para cada alumno. ▪ **parseRestrictionsOrPreferences**: Analiza el fichero que tiene información de las restricciones o preferencias horarias.

Subsistema de Backtracking (backtracking)

Backtracking
Descripción
Clase que se encarga de calcular las distintas soluciones posibles para un alumno. Su procedimiento se puede ver en la sección 5.5.
Responsabilidades
Calcular las distintas soluciones de asignaciones para un alumno. Después de que se realice una solución, se calcula las preferencias cumplidas.
Atributos propuestos
▪ **Matriz de colisión**: Matriz booleana que representa las colisiones entre los grupos de clases. ▪ **Asignaciones iniciales**: Representa las asignaciones iniciales de cada alumno que se deben cumplir. ▪ **Restricciones horarias**: Representa los grupos en los que un alumno no puede estar asignado al ser una restricción. ▪ **Preferencias horarias**: Representa los grupos por los que un alumno no es preferente que esté asignado al ser una preferencia horaria.
Métodos propuestos
▪ **execute(j)**: ejecuta el algoritmo de Backtracking en el nivel j pasado como parámetro.

7.5. Análisis de casos de uso y escenarios

Podemos diferenciar dos modos de ejecución de la herramienta. En primer lugar, el algoritmo que busca la mejor solución para un grupo de alumnos y asignaturas (modo grupal), y por último el modelo que busca una o varias soluciones para un alumno (modo individual). Es por esto que se identifican dos casos de uso ya que existe muy poca interacción del usuario con la herramienta.

7.5.1. Modo Grupal

Ejecución del modo grupal

Tabla 7.3: Ejecución del modo grupal

Tabla 7.3 – *Continua de la anterior página*

Continua en la siguiente página

Precondiciones	Para ejecutar el algoritmo se debe cumplir estas precondiciones: ▪ El modo grupal con el texto 'grupal' debe ser pasado como parámetro. ▪ Un archivo sobre las matriculas debe ser pasado como parámetro con un formato correcto. ▪ Un archivo del horario de clases debe ser pasado como parámetro con un formato correcto. ▪ Un archivo de las asignaciones iniciales debe ser pasado como parámetro con un formato correcto o no existir. ▪ Un archivo de las asignaciones que no se deben hacer debe ser pasado como parámetro con un formato correcto o no existir. ▪ Un archivo de las restricciones horarias que se deben cumplir debe ser pasado como parámetro con un formato correcto o no existir. ▪ Un archivo de las preferencias horarias de los alumnos que se deben cumplir debe ser pasado como parámetro con un formato correcto o no existir. ▪ Un archivo con las excepciones de colisiones con el nombre collisionExceptions debe existir en el directorio en el que el sistema se encuentra con un formato correcto. ▪ Un archivo con las colisiones obligatorias con el nombre mandatoryCollisions debe existir en el directorio en el que se encuentre el sistema, con un formato correcto. ▪ Un archivo con la configuración de las preferencias de asignación con el nombre de preferences debe existir en el directorio en el que se encuentre el sistema con un formato correcto. ▪ Un archivo properties con la configuración del algoritmo genético con el nombre de geneticParameters debe existir en el directorio.

Tabla 7.3: Ejecución del modo grupal

Tabla 7.3 – *Continua de la anterior página*

Continua en la siguiente página

Postcondiciones	Cuando se acaba de ejecutar, el archivo debe haber generado los siguientes ficheros: ▪ Un informe en excel con la información de las asignaciones. ▪ Un informe en .txt por cada alumno y por cada asignatura con la información de las asignaciones.
Actores	El único actor que interactúa con este sistema es el responsable encargado de realizar las asignaciones, que tendrá que ejecutar el algoritmo y revisar los resultados.
Descripción	El usuario que lanza la aplicación indicará los ficheros necesarios descritos anteriormente. El algoritmo buscará la solución hasta que se satisfaga el criterio de terminación. Además, se puede pulsar la letra 'Q' con lo que el sistema generará los ficheros finales con mejor solución encontrada hasta el momento. Al acabar, el algoritmo generará los informes en un archivo excel y archivos de texto plano con los resultados obtenidos.

Tabla 7.3: Ejecución del modo grupal

Tabla 7.3 – *Continua de la anterior página*

Continua en la siguiente página

Excepciones	▪ El modo del programa es inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el modo, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero con las matriculas no existe o es inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero del horario de clases no existe o tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de las asignaciones iniciales tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de las asignaciones que no se deben hacer tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de las restricciones horarias de los alumnos tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo.

Tabla 7.3: Ejecución del modo grupal

Tabla 7.3 – *Continua de la anterior página*

Continua en la siguiente página

Excepciones	▪ El fichero de las preferencias horarias de los alumnos que se deben cumplir tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de configuración de las excepciones de colisiones no existe o tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de configuración de colisiones obligatorias no existe o tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de configuración de las preferencias de asignación no existe o tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de configuración de los parámetros del algoritmo no existe o tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo.

Tabla 7.3: Ejecución del modo grupal

Tabla 7.3 – *Continua de la anterior página*

Continua en la siguiente página

7.5.2. Modo Individual

Ejecución del modo individual

Tabla 7.4: Ejecución del modelo individual

Tabla 7.4 – *Continua de la anterior página*

Continua en la siguiente página

Precondiciones	Para ejecutar el algoritmo se debe cumplir estas precondiciones: ▪ El modo individual con el texto 'individual' debe ser pasado como parámetro. ▪ Un archivo sobre la matricula del alumno debe ser pasado como parámetro con un formato correcto. ▪ Un archivo del horario escolar debe ser pasado como parámetro con un formato correcto. ▪ Un archivo de las asignaciones inicial debe ser pasado como parámetro con un formato correcto. ▪ Un archivo de las asignaciones que no se deben hacer debe ser pasado como parámetro con un formato correcto. ▪ Un archivo de las restricciones que se deben cumplir debe ser pasado como parámetro con un formato correcto. ▪ Un archivo de las preferencias de los alumnos que se deben cumplir debe ser pasado como parámetro con un formato correcto. ▪ Un archivo con las excepciones de colisiones con el nombre collisionExceptions debe existir en el directorio en el que el sistema se encuentra con un formato correcto. ▪ Un archivo con las colisiones obligatorias con el nombre mandatoryCollisions debe existir en el directorio en el que se encuentre el sistema, con un formato correcto. ▪ Un archivo properties con la configuración del algoritmo de backtracking con el nombre de backtracking debe existir en el directorio. En el se van a encontrar el número de soluciones que se quiere y el número máximo de intentos.

Tabla 7.4: Ejecución del modelo individual

Tabla 7.4 – *Continua de la anterior página*

Continua en la siguiente página

Postcondiciones	Cuando se acaba de ejecutar, el programa debe haber generado los siguientes ficheros si encuentra soluciones: ▪ Un informe en excel en el que en cada hoja se da la información de las asignaciones. ▪ Un informe en .txt para cada soluciones de las asignaciones. Si no encuentra soluciones, porque no existen, backtracking determinista lo detecta y avisa. En el caso de backtracking aleatorio, lo intenta hasta el número máximo de intentos.
Actores	Los únicos actores que interactúan con este sistema es el responsable encargado de realizar las asignaciones, y tendrá que ejecutar el algoritmo y revisar los resultados.
Descripción	El usuario que lanza la aplicación indicará los ficheros necesarios descritos anteriormente. Al acabar, el algoritmo generará los informes en excel y txt de los resultados obtenidos.

Tabla 7.4: Ejecución del modelo individual

Tabla 7.4 – *Continua de la anterior página*

Continua en la siguiente página

Excepciones	▪ El modo pasado por parámetro es inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el modo, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero con las matriculas no existe, es inválido o trae más de un alumno. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero del horario escolar no existe o tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de las asignaciones iniciales tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de las asignaciones que no se deben hacer tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo.

Tabla 7.4: Ejecución del modelo individual

Tabla 7.4 – *Continua de la anterior página*

Continua en la siguiente página

Excepciones	▪ El fichero de las restricciones del alumno tiene un formato inválido. • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de las preferencias de los alumnos que se deben cumplir tiene un formato inválido • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de configuración de las excepciones de colisiones no existe o tiene un formato inválido • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de configuración de colisiones obligatorias no existe o tiene un formato inválido • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de configuración de las preferencias no existe o tiene un formato inválido • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo. ▪ El fichero de configuración de los parámetros del algoritmo no existe o tiene un formato inválido • El algoritmo no se ejecutará. En su lugar, mostrará un error que nos indicará el problema que ha habido con el fichero, y que debe ser solucionado para ejecutar el algoritmo.

Tabla 7.4: Ejecución del modelo individual

7.6. Análisis de interfaces de usuario

El sistema es una ampliación del sistema diseñado por Gonzalo De La Cruz Fernández [1]. Ese sistema inicial se basaba en una interfaz de usuario simple, más específico en una aplicación de consola que interactúa muy poco con el usuario, ya que únicamente se va a interactuar al principio. En la inicialización, el modo y los archivos que se necesita procesar son pasados como parámetros.

El sistema dará información al usuario durante la ejecución del algoritmo. Esa información dependerá del modo que pase como parámetro:

- Modo grupal: Informes sobre el estado de la población y su mejor individuo en la consola. Además mostrará algunos gráficos que representan la evolución de la población durante la ejecución del algoritmo, viendo así como evolucionan los individuos a través de las generaciones.
- Modo individual: Informe sobre la solución que se haya encontrado en ese momento en la consola.

Capítulo 8

Diseño del sistema

En este capítulo se describe el diseño del sistema, definiendo la arquitectura del sistema, el diseño de las clases y su interfaz de usuario.

8.1. Arquitectura del Sistema

La arquitectura del sistema se presenta a continuación en forma de diagramas.

8.1.1. Diagrama de paquetes

El diagrama de paquetes se encuentra en la figura 8.1. A continuación se describen brevemente.

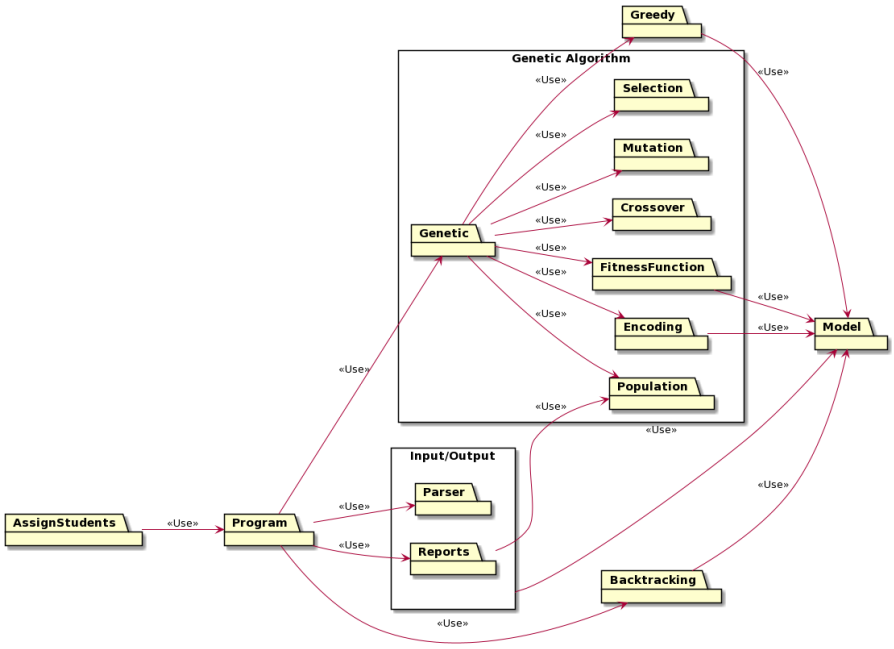

Figura 8.1: Diagrama de paquetes.

Program

El paquete Program contiene los diferentes modos que se van a poder ejecutar, delegando la ejecución en ellos dependiendo del parámetro de entrada. Se encuentra el modo grupal, y el individual.

Model

El paquete Model contiene las clases que se utilizan para modelar el problema. Podemos encontrar: las asignaciones, los estudiantes, las asignaturas, restricciones y otras clases que son necesarias para poder representarlo.

Parser

El paquete Parser contiene las clases que nos van a ayudar a analizar la información necesarias para ejecutar el algoritmo: la configuración y los ficheros de entrada. Nos permite transformar la información a las clases que se encuentran en el paquete Model o en otras representaciones.

Reporter

El paquete Reporter contiene las clases que nos ayudan a realizar informes durante la ejecución del algoritmo y los ficheros excel y txt al acabar de ejecutarse

el algoritmo.

Backtracking

El paquete Backtracking contiene toda la funcionalidad necesaria para realizar varias soluciones a la asignación de un alumno a grupos.

Greedy

El parquete Greedy contiene toda la funcionalidad necesarias para realizar las asignaciones de alumnos a grupos, siguiendo el orden que nos da el Algoritmo Genético.

Genetic

El paquete Genetic contiene la implementación del Algoritmo Genético y delega la ejecución de los operadores a otros paquetes.

Population El paquete Population contiene las clases necesarias para representar a los individuos y a la población.

Encoding El paquete Encoding contiene las clases necesarias para realizar la codificación y decodificación de los individuos.

FitnessFunction El paquete FitnessFunction contiene todas las clases relativas al cálculo del fitness de los individuos.

Selection El paquete Selection contiene todas las clases relativas al operador de selección que se aplica.

Crossover El paquete Crossover contiene las clases relacionadas con el operador de cruce que se aplica.

Mutation El paquete Mutation contiene todas las clases relacionadas con el operador de mutación que se aplica.

8.1.2. Diagrama de componentes

El diagrama de componentes se muestra en la figura 8.2.

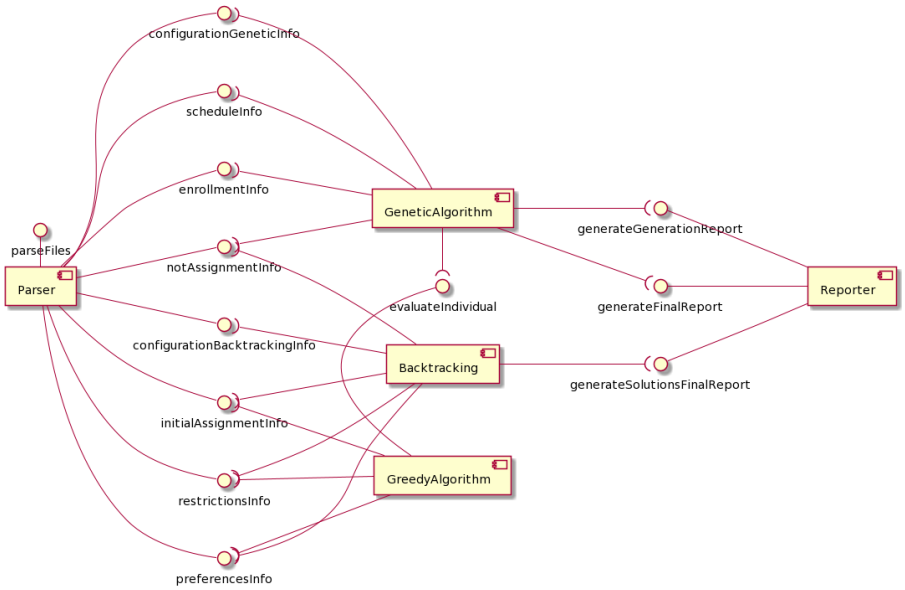

Figura 8.2: Diagrama de componentes.

8.2. Diseño de las clases

Para describir el diseño de clases, primero se muestra un diagrama completo del sistema en la figura 8.3.

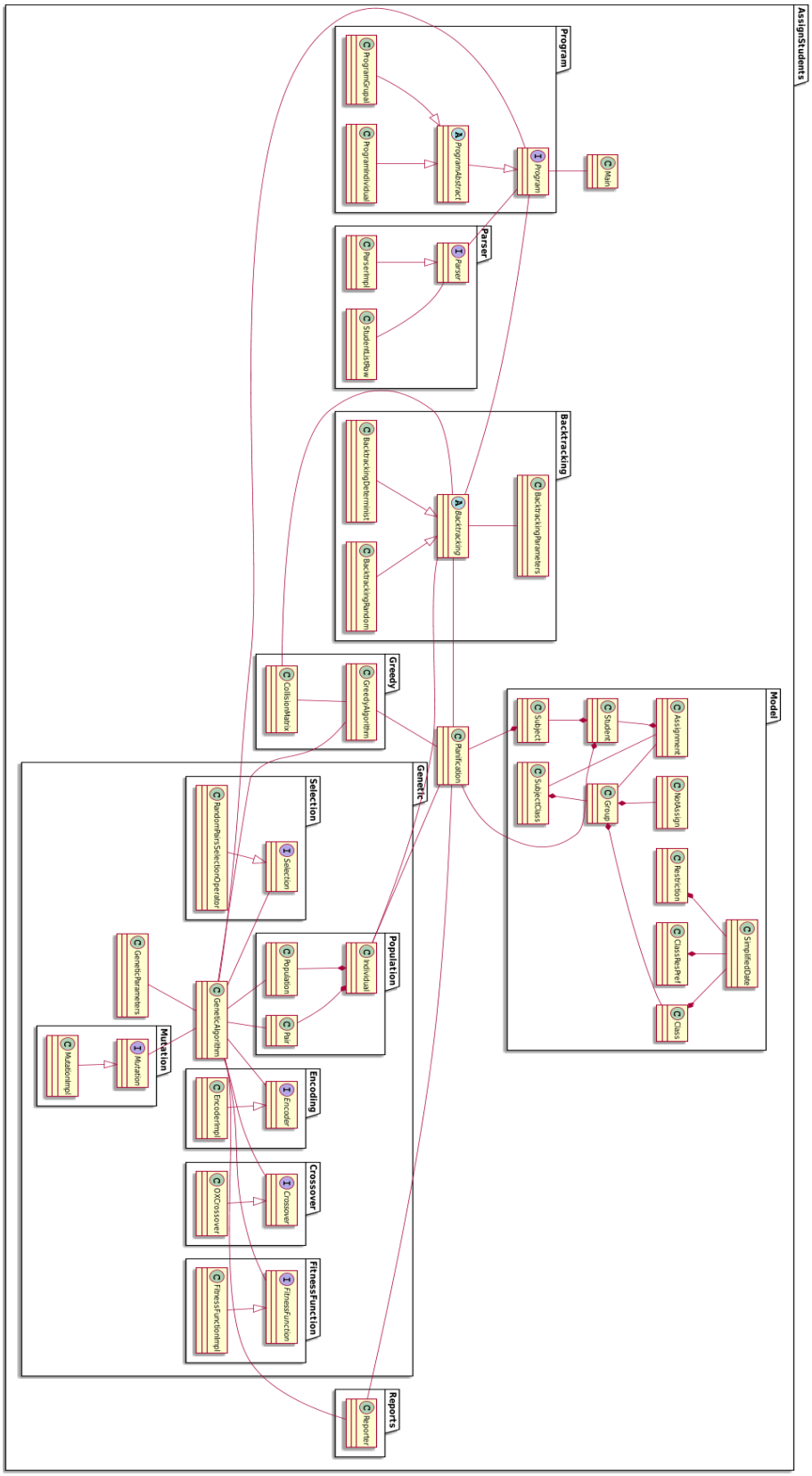

Figura 8.3: Diagrama de clases del sistema

Para poder entenderlo mejor, a continuación se realiza un diagrama de clases de cada paquete.

8.2.1. Diagrama de las clases del sistema

Diagrama de clases del paquete Model

Se encuentra en la figura 8.4.

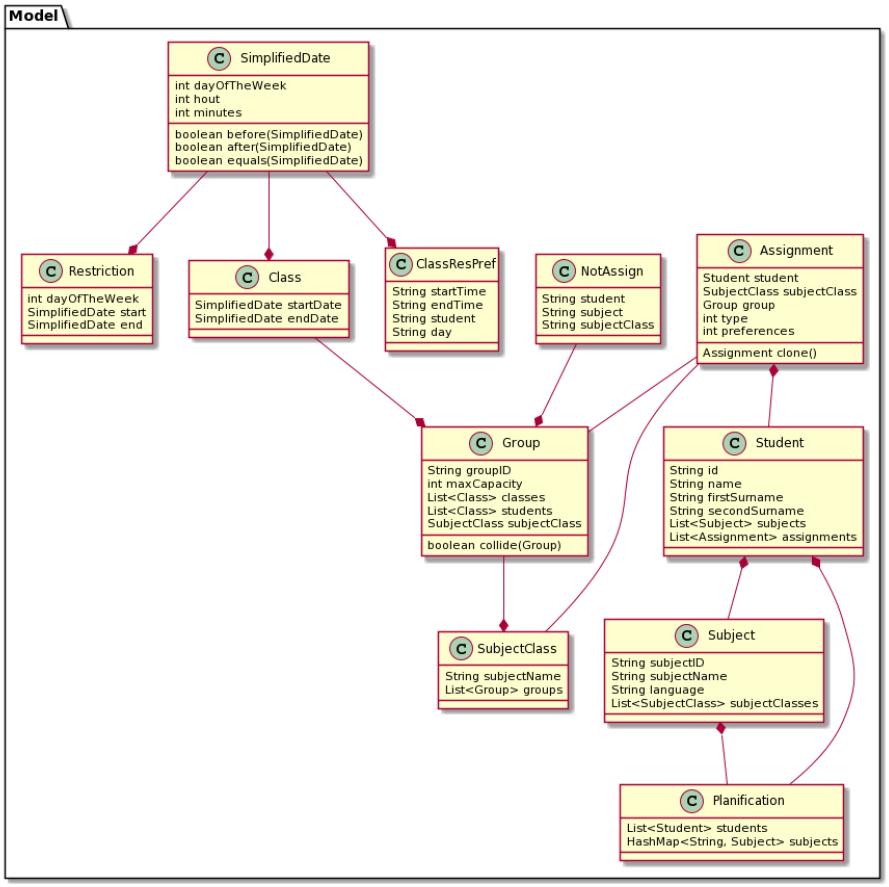

Figura 8.4: Diagrama de clases del paquete Model

Diagrama de clases del paquete Parser

Se encuentra en la figura 8.5.

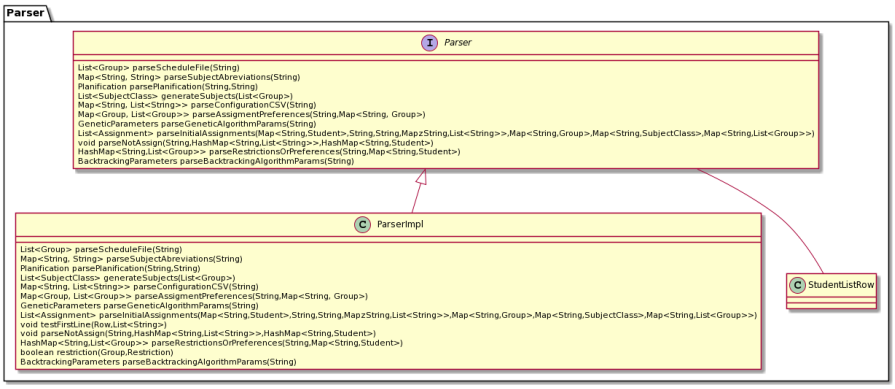

Figura 8.5: Diagrama de clases del paquete Parser

Diagrama de clases del paquete Reporter

Se encuentra en la figura 8.6.

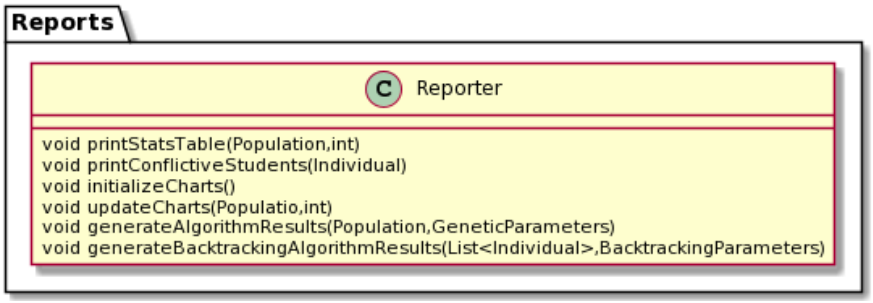

Figura 8.6: Diagrama de clases del paquete Reporter

Diagrama de clases del paquete Backtracking

Se encuentra en la figura 8.7.

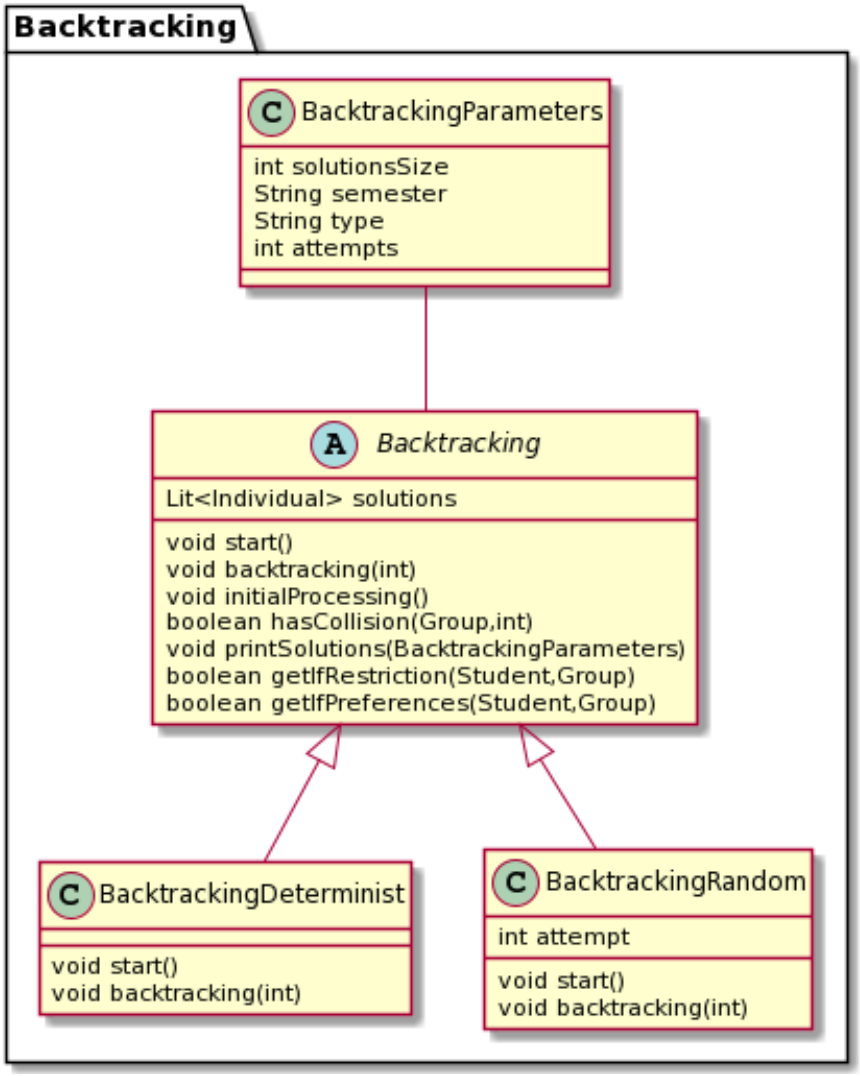

Figura 8.7: Diagrama de clases del paquete Backtracking

Diagrama de clases del paquete Greedy

Se encuentra en la figura 8.8.

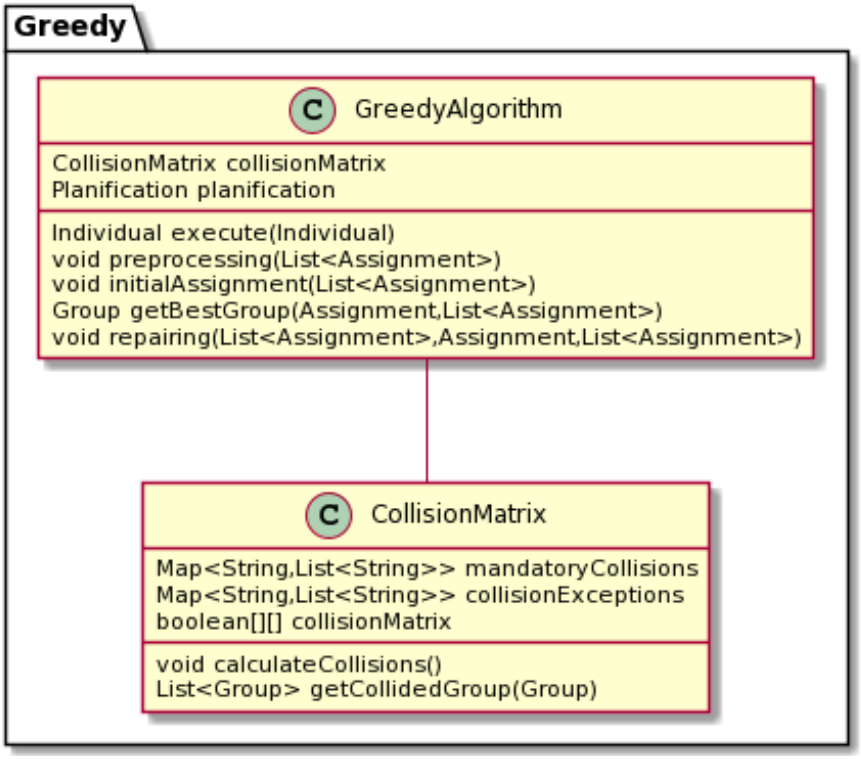

Figura 8.8: Diagrama de clases del paquete Greedy

Diagrama de clases del paquete Genetic

Se encuentra en la figura 8.9.

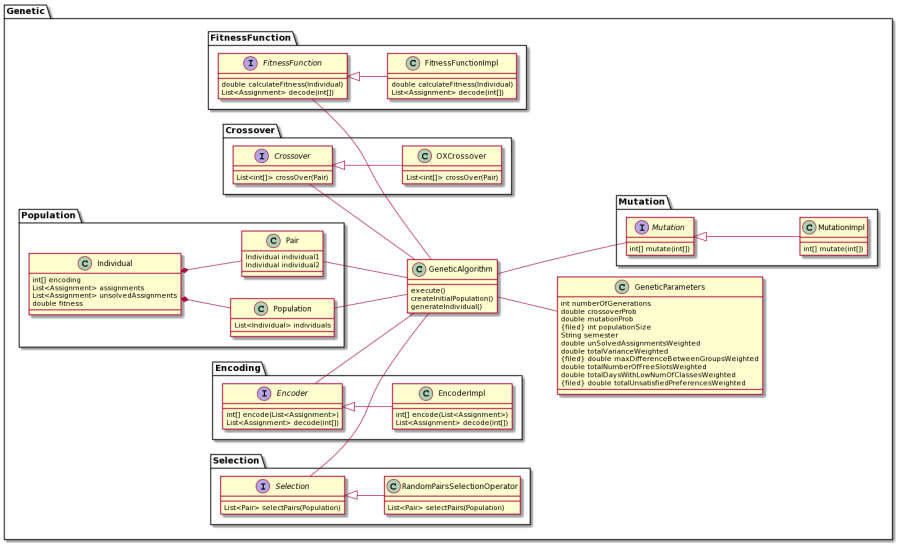

Figura 8.9: Diagrama de clases del paquete Genetic

Diagrama de clases del paquete Program

Se encuentra en la figura 8.10.

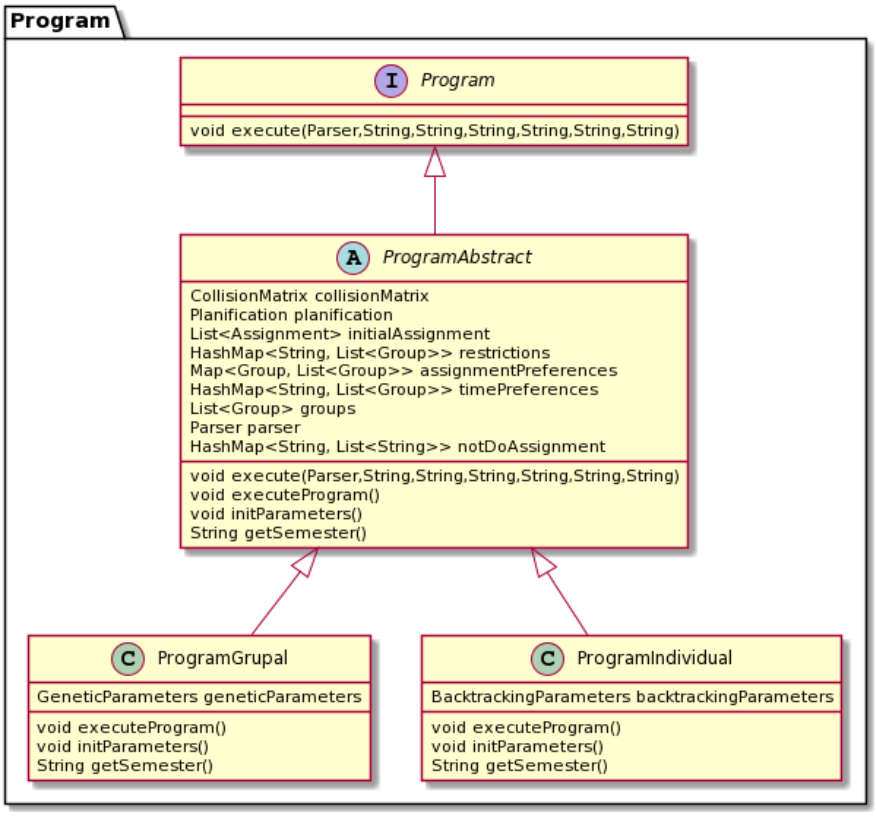

Figura 8.10: Diagrama de clases del paquete Program

8.3. Diseño Interfaz de Usuario

8.3.1. Descripción de la Interfaz

La interacción del usuario con el sistema es bastante limitado, ya que es por línea de comandos que mostrará al usuario la información necesaria durante la ejecución.

La única interacción que debe realizar el usuario es en el momento de iniciar el programa, ya que debe invocarlo pasándole una serie de parámetros, dependiendo del modo que quiera seleccionar. El primer parámetro es el modo que se desea ejecutar. Los siguientes parámetros son: la matrícula de los estudiantes, el horario de las clases, las asignaciones iniciales, las asignaciones que no son necesarias, las restricciones y las preferencias horarias. En el caso del que modo sea individual, la matrícula debe ser únicamente de un alumno. y dependiendo de cuál sea, los siguientes parámetros serán diferentes.

8.3.2. Descripción del comportamiento de la interfaz

La primera y única interacción del usuario con el sistema se realiza en la inicialización, en el que pasara en primer lugar el modo a realizar, y a continuación los archivos de matrícula, de horario de las clases, de asignaciones iniciales, asignaciones que no se deben realizar, restricciones y preferencias horarias. Para ejecutar el programa, es necesario hacer la siguiente invocación por línea de comandos:

java -jar assignStudents <programmeMode><enrollmentFile><scheduleFile><initialAssingment><notAssign><restrictions><preferences>

En el caso de que no se realice la anterior invocación y que falte algún parámetro, el siguiente error aparecerá. También existe la posibilidad de que algún archivo tenga un formato que no es necesario o que falte alguno. Los errores se muestran de esta forma: En el caso de que las entradas y los archivos estén bien, se ejecutará el programa. Dependiendo de qué modo se haya seleccionado como primer parámetro, el sistema mostrará una información u otra.

Modo grupal

El sistema mostrará la información de la configuración del algoritmo genético en el caso de existir, o usará los valores por defecto. Además, en el caso de que una de las variables no se encuentre configurada, se seleccionará el valor por defecto únicamente de esa variable.
Mutation Probability:0.05
Crossover Probability:0.95
Number of generations:1500
Population Size:200
WARNING: semester not provided. Default value S1 will be used
UnsolvedAssignments Weighted:100.0
Total variance weighted:1.0
Max difference between groups weighted:10.0
Total number of free slots weighted:0.08
Total days with low num of classes weighted:10.0
Total unsatisfied preferences weighted:25.0

En la ejecución del programa, unos gráficos se mostrarán como visualización del estado del algoritmo en ciertos factores en tiempo real.

- Gráfico del fitness medio y del fitness del mejor individuo, como se puede ver en la figura 8.11.
- Gráfico de la media de colisiones como se puede ver en la figura 8.12.
- Gráfico de la varianza del mejor individuo como se puede ver en la figura 8.13.

- Gráfico de la media preferencias horarias sin satisfacer como se puede ver en la figura 8.14.

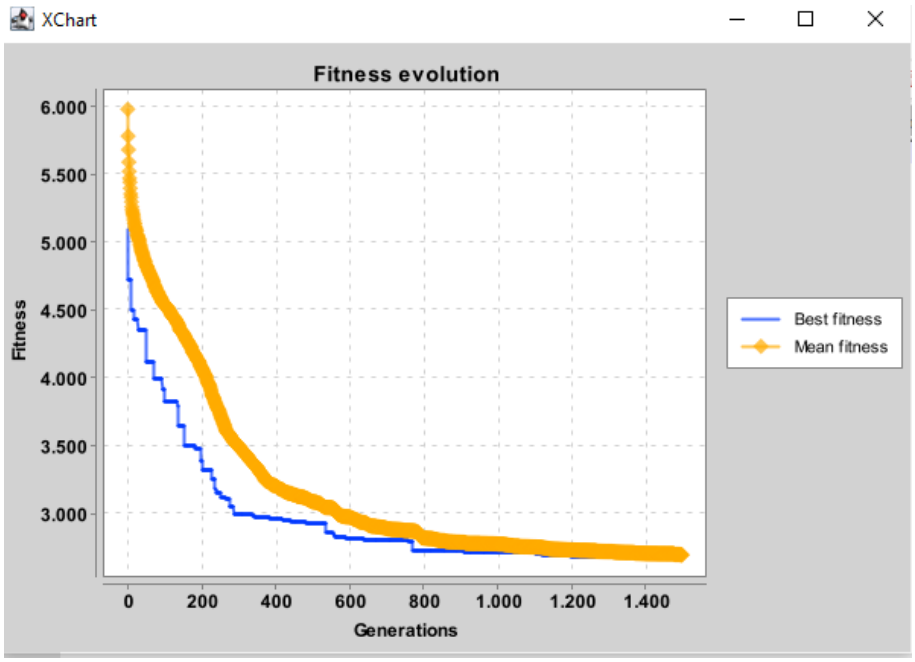

Figura 8.11: Modo grupal: gráfica del fitness.

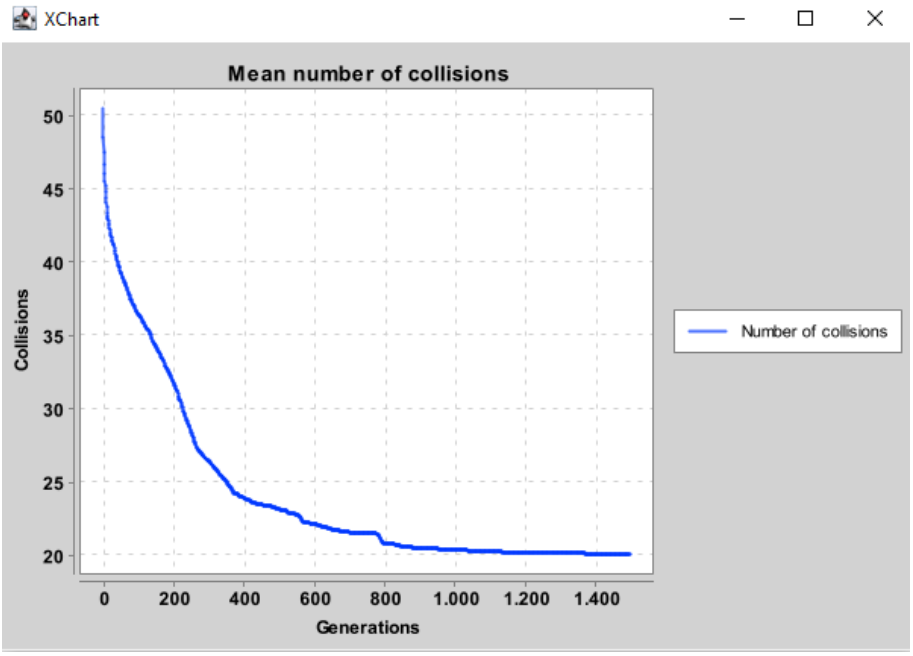

Figura 8.12: Modo grupal: gráfica de las colisiones.

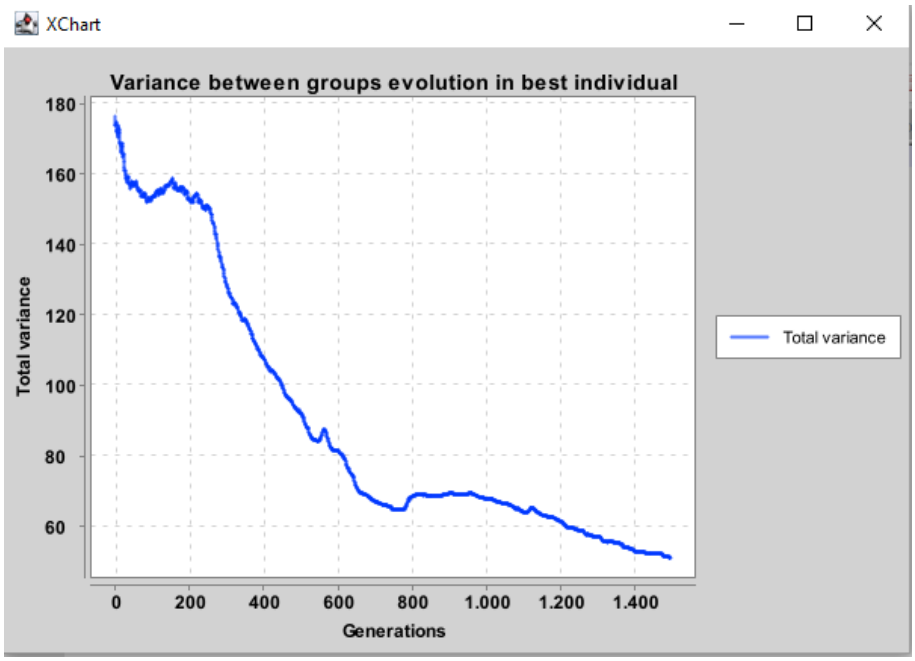

Figura 8.13: Modo grupal: gráfica de la varianza.

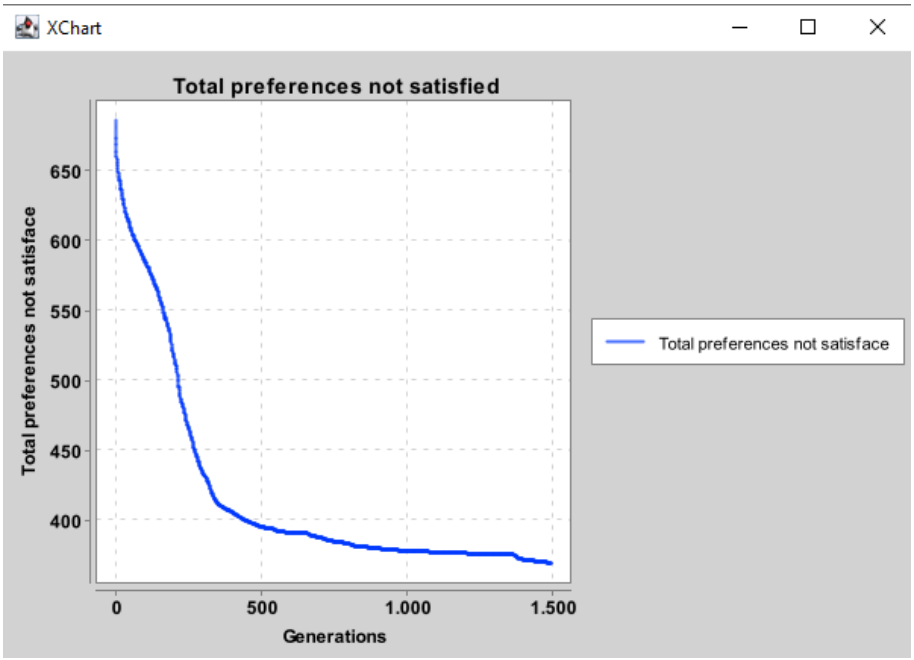

Figura 8.14: Modo grupal: gráfica de las preferencias horarias.

En cualquier momento se puede parar el programa y que genere los resultados pulsando la letra G. Además, cada 20 generaciones, el sistema mostrará información de la población y del mejor individuo hasta el momento, como se puede ver en la figura 8.15

Generation	0
Best fitness	9194.325499341227
Average fitness	10849.618105356922
Mean number of collisions	51.21
Mean max difference between groups	11.545
Mean sum of difference between groups	197.06
Mean total number of free slots	6927.445
Mean total days with 2 hours or less	423.355
Mean total preferes not satisfied	40.0

BEST INDIVIDUAL	
Number of collisions	32
Max difference in group	9.0
Sum of differences between groups	193
Max variance	20.25
Total variance	160.60549934122716
Total number of free slots	6734.0
Total days with 2 hours or less	443.0
Total preferences not satisface	31.0

CONFLICTIVE STUDENT	CONFLICTIVE SUBJECT
Alejo Andrés Castro	TEC.S
Armando Almeda Manzano	SIW.L

Figura 8.15: Modo grupal: información de la generación

Modo individual

El sistema mostrará la información de la configuración del algoritmo backtracking en el caso de existir, o usará los valores por defecto. Además, en el caso de que una de las variables no se encuentre configurada, se seleccionará el valor por defecto únicamente de esa variable.
Maximum Number of Solutions: 10
Semester evaluated: S1
WARNING: Type not provided. Default value determinista will be used
Number of attempts selected: 1000

El algoritmo se ejecutará y la única información que mostrará será la calidad de las asignaciones que se le realiza al alumno en la solución encontrada en ese momento. Esa información se muestra en una tabla ASCII, como se puede ver en la figura 8.16.

Solution 4	
Total number of free slots	8.0
Total days with 2 hours or less	1.0
Total preferences not satisfied	0.0

Figura 8.16: Modo individual: información de la solución

Capítulo 9

Implementación del sistema

9.1. Lenguaje de programación

Como el sistema que se va a realizar es una ampliación de la herramienta actual, se ha continuado con el mismo lenguaje de programación con el que se ha realizado.

Anteriormente, en la herramienta base [1] se eligió Java como lenguaje por varias razones:

1. El rendimiento del sistema no es un elemento clave, ya que al cliente no le importa si el sistema tarda varias horas en ejecutarse.
2. Al tratarse de un prototipo para probar el comportamiento del algoritmo, la velocidad de desarrollo es más importante que el rendimiento.
3. El programador tiene mucha más experiencia usando Java que C++, por lo que se sentirá más cómodo y eficiente escribiendo el código en Java que en C++.

Si se hubiera podido elegir el lenguaje, también hubiera escogido Java, primordialmente por lo familiarizada que estoy con el, ya que se lleva trabajando con este lenguaje desde primer curso del grado.

Además otros lenguajes han sido seleccionados:

- ***LaTeX*:** Es un sistema de preparación de documentos ampliamente utilizado en el intercambio y publicación de documentos científicos en muchos campos. En proyectos de investigación, LaTeX es más popular que Word porque proporciona funcionalidades para insertar ecuaciones, fórmulas o pseudocódigo.
- **PlantUML:** PlantUML es un lenguaje de código abierto que permite a los usuarios crear diagramas UML a partir de un lenguaje simple e intuitivo,

por lo que es de gran ayuda ya que escribir es más rápido que dibujar, sobretodo si hay que realizar modificaciones.

9.2. Herramientas de desarrollo

9.2.1. IntelliJ IDEA

IntelliJ IDEA es un entorno de desarrollo integrado de Java desarrollado por JetBrains. Proporciona muchas funciones, lo que lo convierte en el IDE de Java más inteligente. Tiene herramientas útiles: finalización de código mediante análisis de contexto, navegación rápida de código y refactorización de código. También proporciona integración con herramientas de compilación / paquete como Maven o Gradle, para controlar la versión, como Git y bases de datos, como PostgreSQL o MySQL.

9.2.2. Git

Git es un sistema de control de versiones para administrar cambios en el código fuente que se produjo durante el desarrollo del proyecto. Es el sistema de control más utilizado.

Es un sistema de control de versiones distribuido donde cada desarrollador mantiene su propio repositorio local. En realidad, se trata de un clon del repositorio central de su disco duro. De esta manera, pueden confirmar y actualizar su repositorio local sin interferir con el repositorio central.

Git permite volver a la versión anterior del código y comparar su la evolución a través de diferentes versiones. También permite la creación de sucursales.

9.2.3. Gradle

Gradle es un sistema de automatización de compilación de código abierto, similar a otros sistemas como Apache Ant o Apache Maven, pero introdujo un DSL basado en Groovy en lugar de XML para declarar la configuración del proyecto.

Se utiliza principalmente para el desarrollo y la implementación en Java o Scala, también están disponibles otros lenguajes. Es el sistema de compilación estándar utilizado en proyectos de Android, por lo que se usa ampliamente en la industria.

9.3. Creación del sistema

9.3.1. Problemas encontrados

Uno de los principales problemas encontrados al principio del desarrollo fue la familizariación con un código que estaba escrito por otra persona, de forma que

había que estudiarlo detenidamente para que las mejoras se puedan añadir sin que afecte a la funcionalidad ya hecha. Lo primero que se ha realizado es un estudio detenido tanto de la herramienta base, como de los algoritmos utilizados, haciendo una revisión del Trabajo de Fin de Grado de Gonzalo De La Cruz Fernández [1].

Otro problema que me encontré ha sido decidir qué estructuras de datos posteriormente es necesario utilizar para la nueva información dada, ya que luego se va a acceder a ellas repetidamente. La solución fue realizar un preprocesamiento de las estructuras de datos para así almacenarlas en HashMaps, de forma que cuando se acceda a ellas se va a tener una complejidad de O(1). De esta forma, se mejora el rendimiento y el tiempo de ejecución se reduce.

9.3.2. Descripción detallada de las clases

La descripción detallada se encuentra en uno de los ficheros (javadoc) como anexo a la memoria, como se puede ver en la sección A.1 del apéndice.

Capítulo 10

Pruebas del sistema

Se van a tener en cuenta tres aspectos para probar la implementación del sistema:

- Validación de los archivos de entrada: Se debe comprobar que el sistema valida los archivos de entrada y de configuración del algoritmo que pueden ser localizados y tienen el formato adecuado.
- Resultados del algoritmo: Se debe validar que el algoritmo trabaja como se espera y que produce los resultados esperados y de calidad.
- Validación de la salida: Se debe comprobar que el sistema genera los archivos de salida con los resultados de la ejecución del algoritmo. Además, estos ficheros deben tener el formato especificado.

Las pruebas del primero y segundo punto se basarán en los casos de uso especificados en la sección 7.5.

Además, es necesario realizar una serie de pruebas para analizar si el comportamiento del algoritmo es el esperado, y si las soluciones que proporciona son adecuadas.

10.1. Pruebas de validación de los archivos de entrada y de la salida

Las pruebas de validación que se han realizado son:

Caso de Uso: Ejecución del algoritmo en modo grupal			
ID	**Test**	**Resultado esperado**	**Resultado obtenido**

Tabla 10.1: Validación del modo grupal

Tabla 10.1 – *Continua de la anterior página*

Continua en la siguiente página

PU-G.1	Ejecutar el sistema con todos los archivos de configuración correctos	El sistema se ejecuta de forma correcta y al final genera un excel y algunos ficheros de texto con la información del resultado	OK
PU-G.2	Ejecutar el sistema con modo de programa no válido	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.3	Ejecutar el sistema con una ruta que no es válida del fichero de matrículas	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.4	Ejecutar el sistema con un formato que no es válido del fichero de matrículas	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.5	Ejecutar el sistema con una ruta que no es válida del fichero de horario escolar	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.6	Ejecutar el sistema con un formato que no es válido del fichero de horario escolar	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.7	Ejecutar el sistema con un formato que no es válido del fichero con las asignaciones iniciales	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.8	Ejecutar el sistema con un formato que no es válido del fichero con las asignaciones que no son necesarias	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK

Tabla 10.1: Validación del modo grupal

Tabla 10.1 – *Continua de la anterior página*

Continua en la siguiente página

PU-G.9	Ejecutar el sistema con un formato que no es válido del fichero con las restricciones horarias de cada alumno	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.10	Ejecutar el sistema con un formato que no es válido del fichero con las preferencias horarias de cada alumno	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.11	Ejecutar el sistema con una ruta que no es válida del fichero con las colisiones obligatorias	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.12	Ejecutar el sistema con un formato que no es válido del fichero las colisiones obligatorias	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.13	Ejecutar el sistema con una ruta que no es válida del fichero con las excepciones de colisiones	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.14	Ejecutar el sistema con un formato que no es válido del fichero con las excepciones de colisiones	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.15	Ejecutar el sistema con una ruta que no es válida del fichero de preferencias de asignación	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK

Tabla 10.1: Validación del modo grupal

Tabla 10.1 – *Continua de la anterior página*

Continua en la siguiente página

PU-G.16	Ejecutar el sistema con un formato que no es válido del fichero de preferencias de asignación	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.17	Ejecutar el sistema con una ruta que no es válida del fichero con los parámetros del algoritmo genético	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-G.18	Ejecutar el sistema con un formato que no es válido del fichero con los parámetros del algoritmo genético	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK

Caso de Uso: Ejecución del algoritmo en modo individual			
ID	**Test**	**Resultado esperado**	**Resultado obtenido**
PU-I.1	Ejecutar el sistema con todos los archivos de configuración correctos	El sistema se ejecuta de forma correcta y al final genera un excel y algunos ficheros de texto con la información de los resultados	OK
PU-I.2	Ejecutar el sistema con un modo de programa no válido	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.3	Ejecutar el sistema con una ruta que no es válida del fichero de matrículas	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.4	Ejecutar el sistema con más de un alumno en el fichero de matrículas	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK

Tabla 10.2: Validación del modo individual

Tabla 10.2 – *Continua de la anterior página*

Continua en la siguiente página

PU-I.5	Ejecutar el sistema con un formato que no es válido del fichero de matrículas	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.6	Ejecutar el sistema con una ruta que no es válida del fichero de horario escolar	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.7	Ejecutar el sistema con un formato que no es válido del fichero de horario escolar	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.8	Ejecutar el sistema con un formato que no es válido del fichero con las asignaciones iniciales	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.9	Ejecutar el sistema con un formato que no es válido del fichero con las asignaciones que no son necesarias	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.10	Ejecutar el sistema con un formato que no es válido del fichero con las restricciones horarias del alumno	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.11	Ejecutar el sistema con un formato que no es válido del fichero con las preferencias horarias del alumno	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.12	Ejecutar el sistema con una ruta que no es válida del fichero con las colisiones obligatorias	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK

Tabla 10.2: Validación del modo individual

Tabla 10.2 – *Continua de la anterior página*

Continua en la siguiente página

PU-I.13	Ejecutar el sistema con un formato que no es válido del fichero las colisiones obligatorias	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.14	Ejecutar el sistema con una ruta que no es válida del fichero con las excepciones de colisiones	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.15	Ejecutar el sistema con un formato que no es válido del fichero con las excepciones de colisiones	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.16	Ejecutar el sistema con una ruta que no es válida del fichero de preferencias de asignación	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.17	Ejecutar el sistema con un formato que no es válido del fichero de preferencias	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.18	Ejecutar el sistema con una ruta que no es válida del fichero con los parámetros del algoritmo genético	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK
PU-I.19	Ejecutar el sistema con un formato que no es válido del fichero con los parámetros del algoritmo de backtracking	El sistema no se ejecutará y mostrará un mensaje de error en la consola	OK

10.2. Pruebas de integración

Algoritmo en modo grupal			
ID	**Test**	**Resultado esperado**	**Resultado obtenido**
PI-G.1	Asignaciones iniciales se mantienen	Las asignaciones se mantienen excepto si hay una restricción horaria del alumno que la restrinja	OK
PI-G.2	No se realizan asignaciones iniciales que afecten a restricciones horarias	No se realizan asignaciones iniciales en grupos que su horario entre dentro de una restricción horaria del alumno al que se le va a asignar	OK
PI-G.3	No se realizan asignaciones iniciales que no son necesarias	No se realizan asignaciones que no son necesarias	OK
PI-G.4	Se cumplen las restricciones horarias	No se realizan asignaciones en grupos que su horario entre dentro de una restricción horaria del alumno al que se le va a asignar	OK

Algoritmo en modo individual			
ID	**Test**	**Resultado esperado**	**Resultado obtenido**
PI-I.1	Asignaciones iniciales se mantienen	Las asignaciones se mantienen excepto si hay una restricción horaria del alumno que la restrinja	OK

Tabla 10.4: Pruebas de integración del algoritmo individual

Tabla 10.4 – *Continua de la anterior página*

Continua en la siguiente página

PI-I.2	No se realizan asignaciones iniciales que afecten a restricciones horarias	No se realizan asignaciones iniciales en grupos que su horario entre dentro de una restricción horaria del alumno al que se le va a asignar	OK
PI-I.3	No se realizan asignaciones iniciales que no son necesarias	No se realizan asignaciones que no son necesarias	OK
PI-I.4	Se cumplen las restricciones horarias	No se realizan asignaciones en grupos que su horario entre dentro de una restricción horaria del alumno	OK

10.3. Diseño de los experimentos

Se van a llevar unos experimentos para evaluar tanto la calidad de las soluciones, como el funcionamiento de las mejoras añadidas. El objetivo de estos experimentos es comprobar que el algoritmo se comporta como se espera.

Para ello se ejecutarán algunos experimentos, que dependerán del modo de algoritmo seleccionado, y están descritos y analizados en el siguiente capítulo.

Capítulo 11

Resultados experimentales

Es necesario probar el algoritmo en situaciones reales por lo que se han realizado algunos experimentos.

En primer lugar, en el modo grupal, se probará ciertos pesos en la función fitness en el algoritmo genético y se comparará para encontrar la mejor solución. Posteriormente, se probará ciertos puntos de cada mejora para ver como afecta a la calidad. En segundo lugar, en el modo individual, se va a probar con distintas instancias para así poder ver como afecta.

11.1. Modo grupal

En los experimentos, se han realizado 20 ejecuciones independientes de cada instancia empleada en el estudio experimental. Estas instancias se describen a continuación.

11.1.1. Instancias del problema

La Escuela de Ingeniería Informática de la Universidad de Oviedo ha proporcionado datos reales de matrícula del primer semestre, anonimizados, en el curso 2020-2021. Estos datos serán usados en todas las instancias para la matrícula de los alumnos y el horario de las clases.

A partir de los datos anteriores, se han generado instancias para probar las distintas mejoras realizadas en el trabajo. Para ello, se ha ejecutado el algoritmo inicial [1], y con la solución obtenida se han creado las siguientes instancias:.

- Asignaciones iniciales: se mantienen, como asignaciones iniciales, un porcentaje de las asignaciones obtenidas en la ejecución anterior. Con cada porcentaje se han creado 5 instancias aleatorias.
    - 10 % de las asignaciones obtenidas.

  - 50 % de las asignaciones obtenidas.
  - 90 % de las asignaciones obtenidas.

- Restricciones horarias: considerando diferentes porcentajes de los alumnos, se asignará a cada alumno una restricción horaria de forma aleatoria que se solape en el tiempo con algún grupo asignado en la solución obtenida en la ejecución anterior. Por ejemplo, si un alumno ha sido asignado al seminario 1 de Algoritmia que es los miércoles de 10 a 11, se puede añadir una restricción los miércoles de 10 a 11. Con cada porcentaje considerado se han creado 5 instancias aleatorias.
  - 5 % de los alumnos.
  - 10 % de los alumnos.
  - 20 % de los alumnos.
- Preferencias horarias: se han empelado las mismas instancias que las relativas a las restricciones horarias pero entendiéndolas como preferencias.

Se han tenido en cuenta estos porcentajes, ya que es normal exista una asignación inicial con pocas o muchas asignaciones, pero las restricciones o preferencias horarias no se espera que sean muchas.

11.1.2. Mejoras del modo grupal

Se analizará la medida en que afectas las mejoras en el algoritmo en los factores de calidad. Estos experimentos se han realizado en un clúster de Linux (64 bits) con procesadores Intel Xeon 2.26 GHz y 128 GB RAM, y se ha empleado la versión 10.0.1. de Java.

Los criterios de calidad se describen a continuación. Con respecto a la herramienta base, el único criterio nuevo añadido es la medida de satisfacción de las preferencias horarias, considerado como una medida de calidad del horario de los alumnos.

- **Número de asignaciones sin resolver:** Este factor era, y es el más importante, ya que representa las asignaciones que no se han podido realizar.
- **Homogeneidad de los grupos:** Representa la uniformidad de los grupos de cada tipo de clase de una asignatura. Se tienen en cuenta ciertas medidas: la varianza total entre el número de alumnos de los grupos de clase de una asignatura y la diferencia máxima entre los grupos con menos y con más alumnos de la misma clase de asignatura.
- **Calidad del horario de los alumnos:** Con calidad del horario se hace referencia al número total de huecos libres y número de días con dos o menos horas de clase. Además de esto, se añade las preferencias horarias del alumno, por lo que se tendrá en cuenta el número de preferencias horarias que no se satisfacen.

Las instancias empeladas en los experimentos son las descritas en la subsección anterior. los parámetros del Algoritmo Genético son los recomendados en un estudio paramétrico realizado en la versión de la herramienta [1]:

- Tamaño de la población: 200
- Número de generaciones: 1500
- Probabilidad de cruce: 0.5
- Probabilidad de mutación 0.05.

Por otra parte, los pesos en la función objetivo son los descritos en la sección 11.1.3.

Asignaciones iniciales

Para las asignaciones iniciales, se tendrán en cuenta 15 instancias, que se agrupan en 3 grupos. Esos grupos dependen del porcentaje de asignaciones que consideremos iniciales. De cada ejecución nos quedaremos con el mejor individuo, del que se analizarán los distintos factores nombrados anteriormente.

En las asignaciones iniciales nos podemos encontrar con que una asignación inicial nos ayude y nos facilite la ejecución pero también se puede encontrar lo contrario, ya que puede ser una asignación inicial que no nos permite realizar otras asignaciones, dejando algunas sin resolver.

En la tabla 11.1 se puede encontrar un resumen de los resultados experimentales. Se han realizado 20 ejecuciones en cada instancia ya que el algoritmo tarda en ejecutarse bastante tiempo, y cada ejecución se llevará acabo 1500 generaciones. Cada grupo (10 %, 50 % y 90 % de las asignaciones iniciales) cuenta con 5 instancias, por lo que en total son 15 instancias. Se muestran las medias de los valores de los criterios de calidad de las mejores soluciones obtenidas en las 20 ejecuciones del algoritmo, agrupando las instancias en los tres grupos mencionados anteriormente.

%	10 %	50 %	90 %
Número de instancias	5	5	5
Número de ejecuciones por instancia	20	20	20
Número asignaciones sin resolver	22,241	19,6	20
Varianza total	85,777	45,7235	61,948
Diferencia máxima total	6,944	5,532	5,81
Horas libres	6864,518	6788,963	6530,343
Días con pocas horas de clase	266,638	264,941	292,67
Fitness	5580,747	5252,969	5590,971

Tabla 11.1: Asignaciones iniciales

El número de asignaciones sin resolver, como se puede ver en la figura 11.1, es similar en los tres casos considerados, siendo ligeramente mejor en las instancias en las que las asignaciones iniciales representan un 90 % y especialmente un 50 % del total de las asignaciones. Según este experimento, podemos esperar que una buena asignación inicial ayude a que existan un menor número de asignaciones sin resolver.

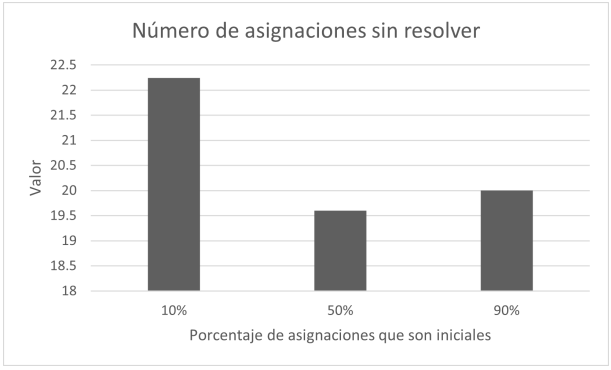

Figura 11.1: Asignaciones iniciales: asignaciones sin resolver

Para analizar la homogeneidad, se tienen en cuenta dos factores, la varianza total y la diferencia máxima en el número de alumnos entre grupos de la misma asignatura. Como se puede ver en la gráfica en la figura 11.2, se vuelve a mantener que una asignación inicial del 50 % de las asignaciones nos da una mejor respuesta en estos factores. La mejora es más clara en la varianza total, estando muy próximo al caso de 90 %.

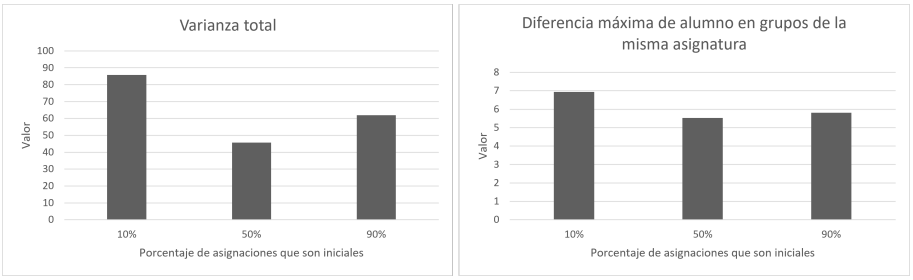

Figura 11.2: Asignaciones iniciales: homogeneidad

Los factores a tener en cuenta para que el horario del alumno sea de calidad son: número de horas libres entre clases, número de días con menos de dos horas de clase, y número de preferencias horarias sin satisfacer. Como en este experimento no se tienen en cuenta las preferencias, vamos a obviar este factor. Como se puede ver en la gráfica de la figura 11.3, el único que da buen resultado a la calidad es el caso de el 50 %, ya que en el caso de número de horas libres entre clases, el

valor que nos da el caso de el 10 % de asignaciones está muy cerca del 50 % pero sigue siendo el peor de todos, mientras que en el número de días con menos de dos horas de clase, es el 90 % la peor opción. Cuantas más asignaciones iniciales haya, más difícil va a ser para el sistema mantener un horario de calidad, ya que pueden ser asignaciones iniciales que nos impida esto.

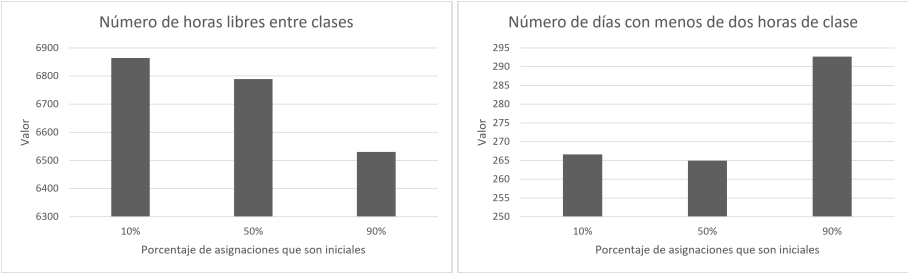

Figura 11.3: Asignaciones iniciales: calidad del horario del alumno

Hay que tener en cuenta los tiempos de ejecución, como se puede ver en la tabla 11.2. Como se puede ver, cuantas más asignaciones iniciales haya, menor tiempo de ejecución necesita el algoritmo genético.

%	10 %	50 %	90 %
Tiempo ejecución	7166152	6917051	4708968

Tabla 11.2: Asignaciones iniciales: tiempo de ejecución en milisegundos

Este experimento se ha realizado con instancias aleatorias de asignaciones que nos ha dado como resultado el algoritmo, por lo que son asignaciones de calidad. Es por esto, que el uso de asignaciones iniciales se tiene que tratar con cuidado, ya que puede facilitar o dificultar mucho el trabajo, consiguiendo más/menos asignaciones sin resolver o mejor/peor calidad del horario. Es por esto, que una buena asignación inicial (de calidad) puede dar muy buenos resultados cuanto más porcentaje de asignaciones iniciales haya.

Restricciones horarias

En los experimentos de las restricciones horarias, las instancias se dividen en grupo dependiendo del porcentaje de alumnos que tiene restricciones:

- 5 % de los alumnos.
- 10 % de los alumnos.
- 20 % de los alumnos.

Estas instancias contienen restricciones que afectan directamente a la matrícula del alumno. Asimismo, en estos experimentos no se consideran asignaciones iniciales. El resumen del experimento lo podemos ver en la tabla 11.3.

%	5 %	10 %	20 %
Número de instancias	5	5	5
Número de ejecuciones por instancia	20	20	20
Número asignaciones sin resolver	26,094	32,451	42,379
Varianza total	81,604	86,343	86,140
Diferencia máxima total	7,311	7,158	7,191
Horas libres	6924,178	6921,249	6885,88
Días con pocas horas de clase	314,9564	324,187	327,477
Fitness	6589,55	7141,018	8179,17

Tabla 11.3: Restricciones horarias.

Como se puede ver en la figura 11.4, según se añaden más restricciones horarias, más asignaciones habrá sin resolver. Esto es esperable, ya que las restricciones horarias limitan las posibilidad de que un alumno tenga grupos suficientes a los que poder asistir en todas las asignaturas de su matrícula.

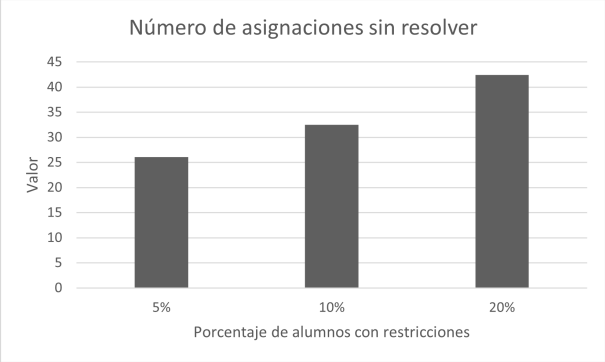

Figura 11.4: Restricciones horarias: asignaciones sin resolver.

Además, el factor de la homogeneidad no se ve afectado directamente, como se puede ver en la figura 11.5, ya que las asignaciones que se vayan a realizar van a mantener la homogeneidad en el grupo. Mientras que en la varianza total el mejor resultado se obtiene en el caso del 5 % de alumnos con restricciones, en el factor de máxima diferencia de alumnos entre grupos de la misma asignatura pasa lo contrario. El factor de homogeneidad se ve afectado por el anterior, el número de asignaciones sin resolver.

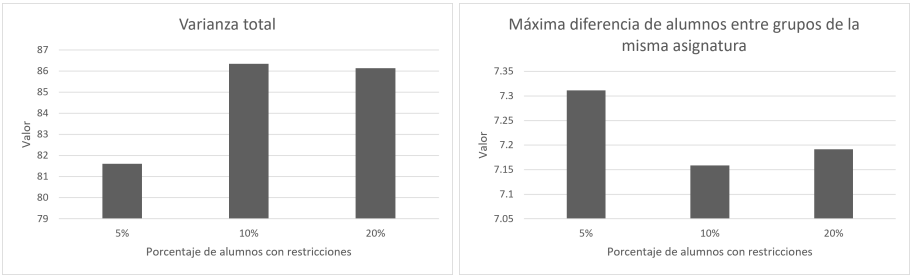

Figura 11.5: Restricciones horarias: homogeneidad.

Por último se analiza la calidad del horario del alumno, que se puede visualizar en la gráfica de la figura 11.6. Se puede observar que las restricciones afectan de forma diferente a cada factor. Esto es así principalmente por el número de asignaciones sin resolver, ya que el alumno tendrá días con menos clases porque le faltarán grupos por asignar, pero a su vez podría tener menos horas libres por la misma razón.

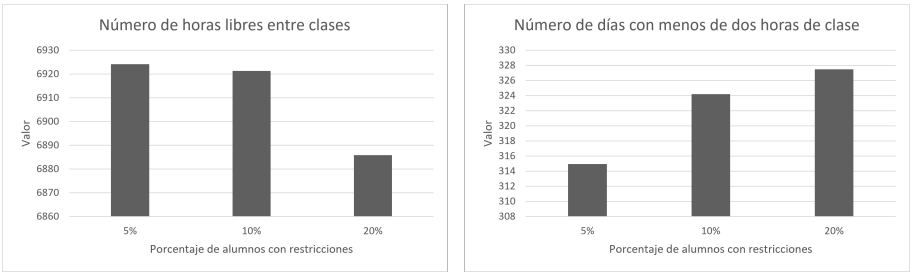

Figura 11.6: Restricciones horarias: calidad del horario.

El número de restricciones afectan a ciertos factores, pero como se puede ver en la tabla 11.4, no afecta al tiempo de ejecución, siendo pequeñas las diferencias en los tres casos considerados.

%	5 %	10 %	20 %
Tiempo ejecución	7475159	7823768	7634526

Tabla 11.4: Restricciones horarias: tiempo de ejecución en milisegundos

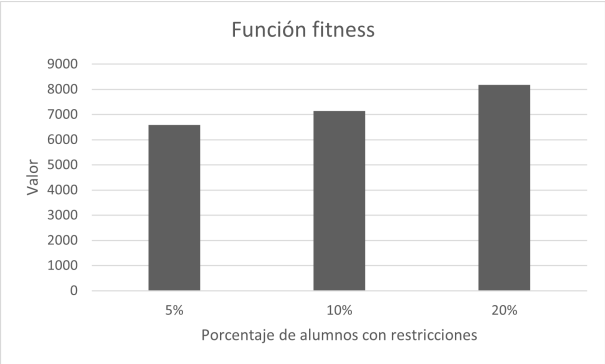

Figura 11.7: Restricciones horarias: fitness.

En conclusión, el uso de restricciones va a afectar la calidad de la solución. Cuantas más restricciones, más asignaciones sin resolver, por lo que el fitness será peor, como se puede ver en la gráfica de 11.7. El resto de factores no se verán tan afectados, ya que el algoritmo siempre buscará encontrar una solución con horario de calidad para el alumno y homogéneo. Pero el valor que se ve afectado es la base del algoritmo, que es buscar el mayor número de asignaciones. En resumen, para usar esta funcionalidad, es conveniente que las restricciones no se añadan inconscientemente.

Preferencias horarias

Para las preferencias horarias de los alumnos, las instancias se van a dividir en tres grupos, dependiendo del porcentaje de alumnos tienen alguna preferencia horaria.

- 5 % de los alumnos.
- 10 % de los alumnos.
- 20 % de los alumnos.

%	10 %	50 %	90 %
Número de instancias	5	5	5
Número de ejecuciones por instancia	20	20	20
Número asignaciones sin resolver	21,664	21,923	22,845
Varianza total	88,507	89,064	90.62
Diferencia máxima total	7,264	7,249	7,354
Horas libres	6919,22	6938,149	6910,8206
Días con pocas horas de clase	327,45	330,382	333,02
Preferencias horarias sin satisfacer	13,11	24,720	53,61

Tabla 11.5: Preferencias horarias.

Un resumen del experimento se puede ver en la tabla 11.5. A continuación se discute de cada factor de calidad.

Como se puede ver en la gráfica de la figura 11.8, según aumentamos los alumnos con preferencias horarias, las asignaciones sin resolver aumentan. A pesar de que en la gráfica la diferencia es muy grande, la realidad es que el 20 % de alumnos con preferencias aumenta en uno las asignaciones sin resolver comparándolo con el 5 % de alumnos, por lo que es una desventaja pero no es un cambio muy grande.

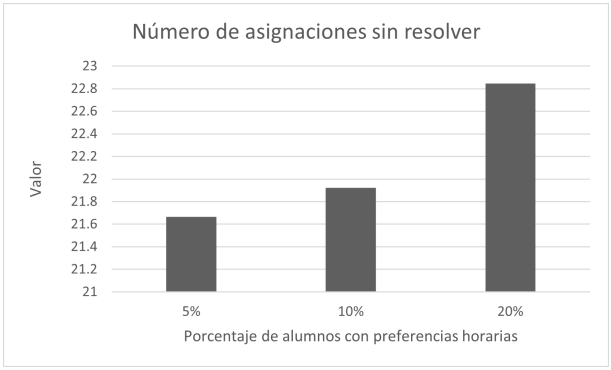

Figura 11.8: Preferencias horarias: asignaciones sin resolver.

La homogeneidad de grupos se ve afectada por las preferencias, ya que a más preferencias horarias, más peso se le da a que se satisfagan y menos a otros puntos, como se puede ver en la figura 11.9. En los dos casos, el mayor uso de preferencias horarias afecta directamente a la homogeneidad, pero si su porcentaje es más bajo, el 10 %, la solución se encuentra muy cercana a la menor, 5 %. Es por esto, que la homogeneidad se ve afectada cuantas más preferencias hay, pero si se realiza un uso de esta funcionalidad de forma moderada, puede ayudar a la calidad del horario del alumno sin afectar mucho a la homogeneidad.

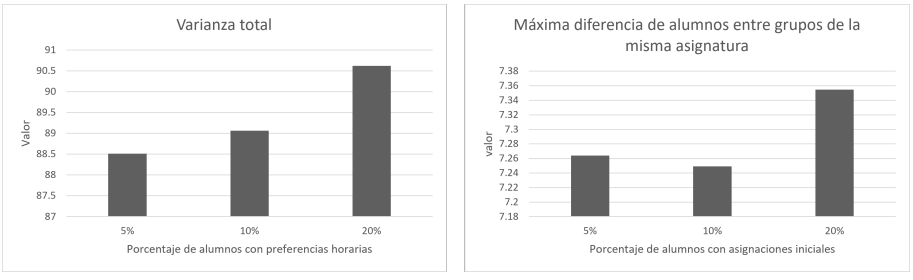

Figura 11.9: Preferencias horarias: homogeneidad.

El factor que se va a ver más afectado es la calidad del horario del alumno. A

esto hay que tener en cuenta que el alumno puede preferir tener clase un día entero pero no tener otro día, por lo que el factor de días con pocas horas de clase se va a ver afectado, pero es una preferencia del alumno por lo que tiene más peso. Los experimentos realizados dan los resultados que podemos ver en la gráfica 11.10

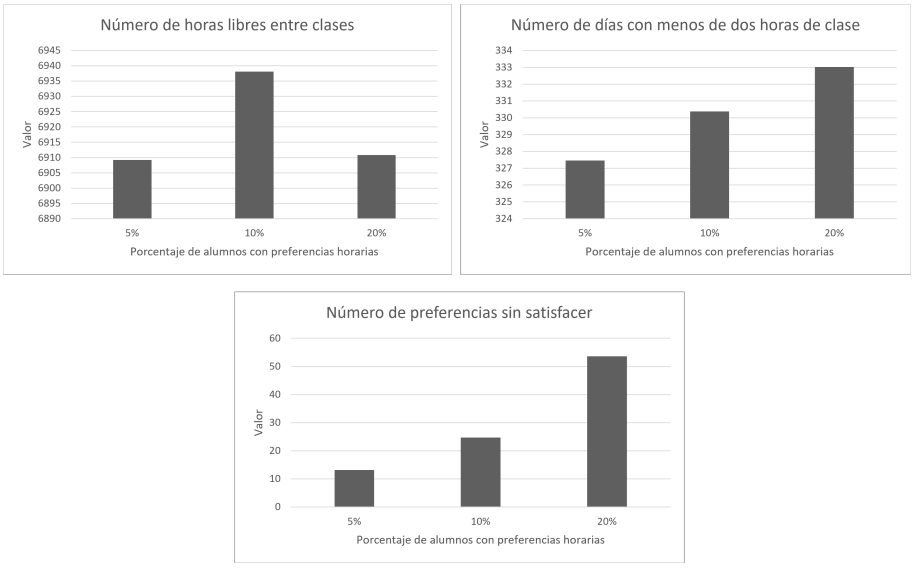

Figura 11.10: Preferencias horarias: calidad del horario.

Es obvio que, al ser instancias que contienen preferencias que afectan directamente a la matrícula del alumno, cuantas más preferencias haya, más preferencias sin satisfacer habrá. Además, se puede ver cómo afecta las preferencias al número de horas libres entre clases y al número de días con menos de dos horas de clase, pero son factores que al existir preferencias, es normal que cambien.

El tiempo de ejecución no varía con el porcentaje de alumnos con preferencias, siendo muy parecidos como se puede ver en la tabla 11.6.

%	5 %	10 %	20 %
Tiempo ejecución	6377352	6776285	6634439

Tabla 11.6: Preferencias horarias: tiempo de ejecución en milisegundos

Como se ha observado, el número de preferencias horarias que haya va a afectar a la solución, pudiendo producir cambios en el número de asignaciones sin resolver, y en la homogeneidad. Además, si las soluciones entran dentro de un horario que obligatoriamente va a tener clase, va a empeorar la solución. Es por esto que hay

que tener cuidado con esta mejora.

11.1.3. Peso de las preferencias

El objetivo de este apartado es encontrar el valor adecuado para el peso de las preferencias en la función fitness en el Algoritmo Genético.

Para los diferentes pesos considerados en la versión anterior de la herramienta, Gonzalo De La Cruz [1] realizó un experimento por lo que se mantienen esos pesos, que son los siguientes:

- **Asignaciones sin resolver (wc):** 100
- **Varianza total (wv):** 1
- **Diferencia máxima entre grupos de la misma asignatura (wd):** 10
- **Total de horas libres entre clases (wf):** 0.08
- **Total de días con menos de dos horas de clase (wl):** 0.2

En este experimento se prueban diferentes pesos para el *total de preferencias sin satisfacer*, que serán: 0, 10, 25 y 50.

Las instancias empleadas son las mismas que las utilizadas anteriormente como restricciones horarias, pero considerándolas como preferencias horarias. Además, el estudio se restringe a los casos de 5 % y 20 % de los alumnos con preferencia. Para cada grupo de porcentaje, se cuenta con 5 instancias y se realizan 20 ejecuciones de cada instancia.

5 % de alumnos con preferencias

Para los distintos pesos considerados en los experimentos se han obtenidos los resultados que se muestra en la tabla 11.7.

Pesos	0	10	25	50
Número de instancias	5	5	5	5
Número de ejecuciones por instancia	20	20	20	20
Número asignaciones sin resolver	21.4765	21.25545	21.64955	21.8235
Varianza total	80.147	85.738	88.320	91.707
Diferencia máxima total	6.630	7.220	7.230	7.244
Horas libres	6892.59	6902.894	6896.253	6910.123
Días con pocas horas de clase	325.509	325.467	327.386	329.190
Número de preferencias horarias no satisfechas	26.637	16.681	12.609	8.903

Tabla 11.7: Pesos para 5 % alumnos con preferencias.

Como se puede observar en la figura 11.11, los mejores resultados con respecto al número de asignaciones sin resolver son producidos con pesos 0 y 10, y lo mismo ocurre con la homogeneidad, como se puede ver en la figura 11.12. Sin embargo, estos pesos dan poca prioridad a las preferencias horarias, lo que puede contribuir a reducir la calidad del horario de los alumnos.

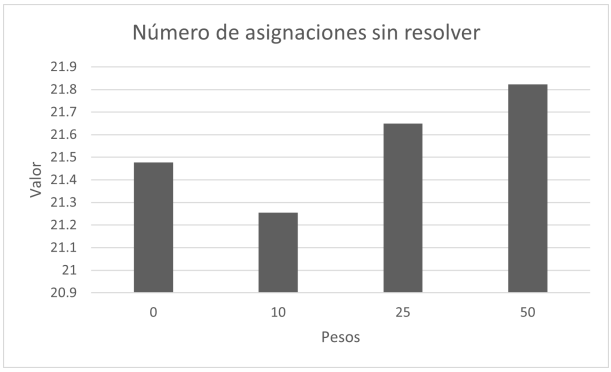

Figura 11.11: 5 % alumnos con preferencias horarias: asignaciones sin resolver.

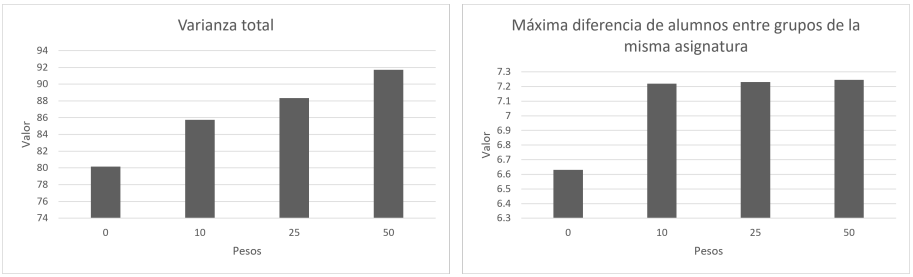

Figura 11.12: 5 % alumnos con preferencias horarias: homogeneidad.

Es lógico que el mayor peso va a obtener mejor resultado en cuanto el número de preferencias sin satisfacer, pero da muy malos resultados en el resto de factores. El peso 25 se acerca a mantener el resto de factores de calidad, satisfaciendo las preferencias horarias en buena medida.

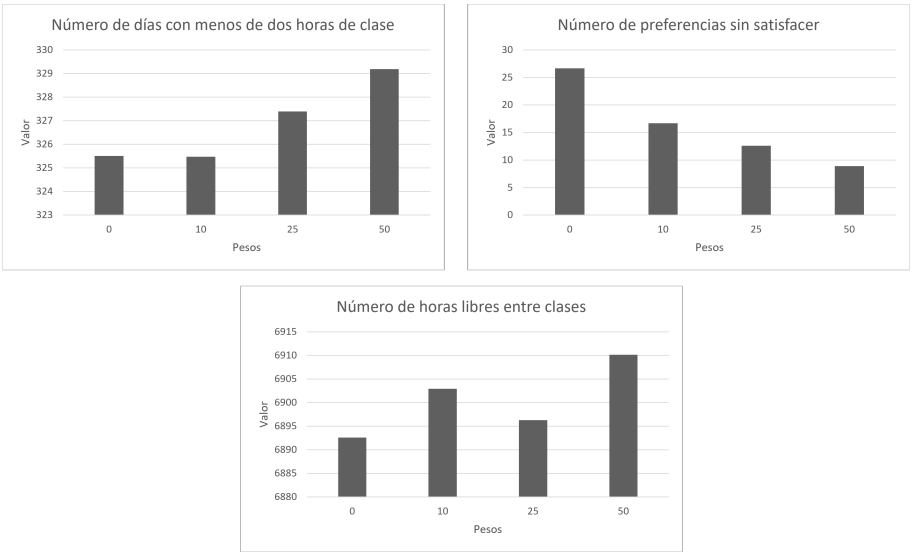

Figura 11.13: 5 % alumnos con preferencias horarias: calidad del horario.

20 % de alumnos con preferencias

El resumen de los resultados con un 20 % de alumnos con preferencias según el peso que se le de a las preferencias horarias, se puede encontrar en la tabla 11.8.

Pesos	0	10	25	50
Número de instancias por porcentaje	5	5	5	5
Número de ejecuciones por instancia	20	20	20	20
Número asignaciones sin resolver	21.427	22.153	22.749	24.657
Varianza total	84.630	94.247	92.921	96.991
Diferencia máxima total	6.898	7.268	7.458	7.488
Horas libres	6930.287	6918.983	6901.343	6926.654
Días con pocas horas de clase	324.905	328.538	330.5006	342.520
Número de preferencias horarias no satisfechas	117.440	67.826	52.335	44.173

Tabla 11.8: Pesos para 20 % alumnos con preferencias.

En este caso, según aumenta el número de preferencias, la diferencia de asignaciones sin resolver entre el peso 0, 10 y 25 es cada vez menor. El peso mayor (50) es el que se ve más afectado.

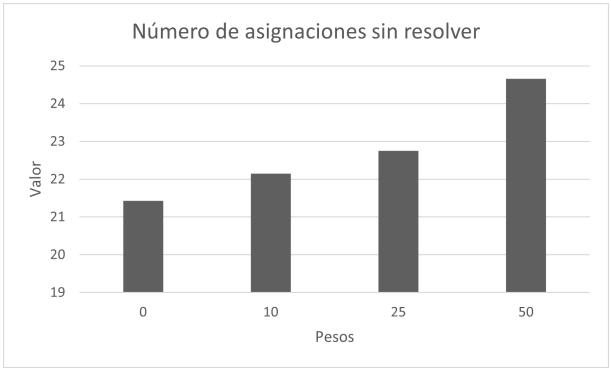

Figura 11.14: 20 % alumnos con preferencias horarias: asignaciones sin resolver.

La homogeneidad es el factor que se ve más afectado en este experimento como se puede ver en la figura 11.15. El sistema intentará mantener la homogeneidad, pero las preferencias deben recibir más peso. Es así, que en ninguno de los dos factores que se contemplan para la homogeneidad, se ve que un peso mayor o igual que 10 pueda afectar positivamente. Lo contrario pasa con la calidad del horario del alumno, que se ve afectada positivamente cuanto más peso para las preferencias horarias, ya que más se serán satisfechas. El peso que mejor satisface los distintos factores de calidad es el 25, como se puede ver en 11.16.

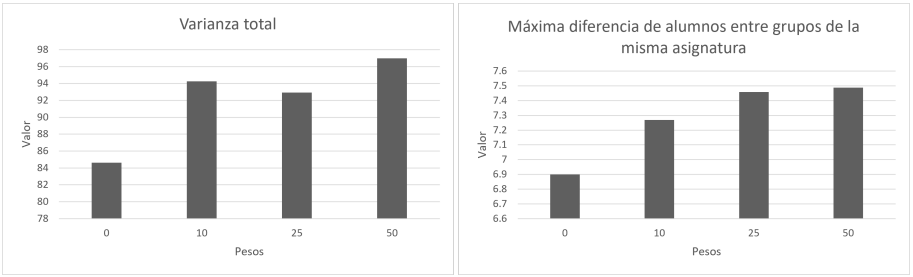

Figura 11.15: 20 % alumnos con preferencias horarias: homogeneidad.

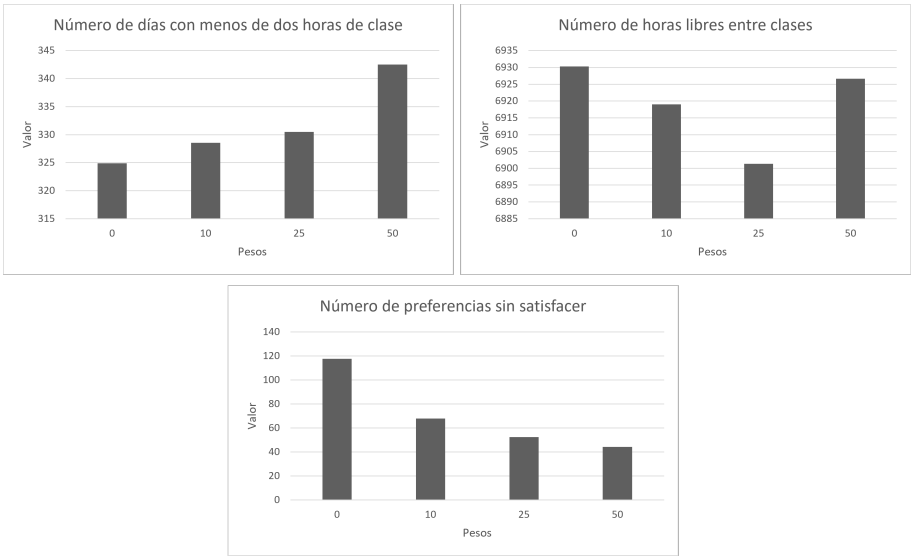

Figura 11.16: 20 % alumnos con preferencias horarias: calidad del horario.

En resumen, cualquier peso mayor que 0 va a afectar al número de asignaciones sin resolver, y es un factor que hay que tener en cuenta. Además, la homogeneidad empeora cuanto mayor sea el peso, pero no afecta drásticamente, ya que el sistema siempre va a intentar mantenerla. Se puede ver en los experimentos cómo un peso de 25 afecta muy positivamente a la calidad del horario del alumno, bajando a la mitad el número de preferencias horarias sin satisfacer, pero sin afectar en gran medida al resto de factores. Un peso de 10 también da buenos resultados, pero no mejora mucho la calidad del horario de alumno, y es una de las prioridades.

Parámetros recomendados

Como resultado de este experimento y del realizado para la versión anterior de la herramienta [1], los parámetros recomendados para el Algoritmo Genético son:

- **Asignaciones sin resolver** (wc)**:** 100
- **Varianza total** (wv)**:** 1
- **Diferencia máxima entre grupos de la misma asignatura** (wd)**:** 10
- **Total de horas libres entre clases** (wf)**:** 0.08
- **Total de días con menos de dos horas de clase** (wl)**:** 0.2
- **Total de preferencias sin satisfacer** (wp)**:** 25

11.2. Modo individual

Se han realizado estos experimentos en un portatil (64 bits) con procesador Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz y 12 GB RAM.

En los experimentos se han realizado 100 ejecuciones por instancia. Se cuenta con tres instancias, en las que la primera va a ser una matricula de alumno normal, sin asignaciones iniciales, asignaciones que no son necesarias, restricciones horarias ni preferencias horarias. En segundo lugar, se contará con la misma matrícula, pero con asignaciones iniciales y restricciones horarias que potencialmente reduzcan el número de soluciones. Y por último, un alumno con una asignación imposible de realizar.
Cada ejecución se hará con la misma configuración. Se especifican 10 soluciones, y 1000 intentos para el modo aleatorio.

11.2.1. Matricula simple

Para la matrícula simple, se ha escogido una matrícula de un alumno básica, sin asignaciones iniciales, ni preferencias horarias.

Modo	Determinista	Aleatorio
Número de ejecuciones	1	100
Número de soluciones conseguidas	10	10
Horas libres	2.812	9.029
Días con pocas horas de clase	0.409	0.02
Tiempo en milisegundos	6.51	5.84

Tabla 11.9: Modo individual: matrícula de alumno simple

Como se puede ver en la tabla 11.9, la media de la calidad de las soluciones según el modo puede verse alterada. En el modo determinista, al intentar ir siempre por el primer grupo, es menos probable que si no cuenta con restricciones, se intente realizar una asignación que colisione. Es por esto, que con el modo aleatorio existen más posibilidades de que el horario no sea de calidad para el al muestrear distintas partes del espacio de búsqueda, aunque en ninguno de los dos casos se escogen las soluciones con ese criterio.

El tiempo es muy corto en los dos casos, pero algo más en el aleatorio, ya que realiza menos vueltas atrás. Además, al ser una matrícula simple, los dos llegan al número de soluciones esperadas

11.2.2. Matrícula compleja

Se cuenta con la misma matrícula, pero con a restricción que afectará en las soluciones. El alumno está matriculado en una asignatura, que para el seminario

cuenta con dos grupos, por lo se realiza una asignación inicial para el resto, y se restringe uno de los dos grupos. En este caso, el programa no podrá encontrar el número de soluciones establecido, por lo que la calidad del horario será la misma para las dos, pero se hará una comparación de tiempo y de intentos que han realizado. Cada vez que se llamada al método de backtracking, se considera un intento.

Modo	Determinista	Aleatorio
Número de ejecuciones	1	100
Número de soluciones conseguidas	1	1
Horas libres	6	6
Días con pocas horas de clase	1	1
Intentos	2	1000
Tiempo en milisegundos	0.87	1.7

Tabla 11.10: Modo individual: matrícula de alumno compleja

La calidad de la solución en los dos casos es igual, ya que solo existe una solución posible. Es por esto, que se nota en el tiempo y en los intentos, como el modo determinista recorre todo el árbol, y para. En el caso del modo aleatorio, al realizarse una mezcla de grupos, para cuando llega al número de soluciones esperadas o al llegar al número de intentos, como pasa en este caso.

11.2.3. Matrícula imposible

Junto con lo anterior, se añade una restricción horaria al otro grupo que quedaba libre. De esta forma, con una asignación inicial completa menos un tipo de asignatura, en el que no se le puede asignar a ninguno de los dos grupos, será una matrícula imposible de realizar.

En este caso, no hay calidad de la solución porque no existe y en los dos modos, si no existe solución, el sistema avisa y acaba la ejecución sin generar ningún fichero. El modo determinista únicamente realiza un intento, y termina, por lo que el tiempo es casi 0. El modo aleatorio realiza los 1000 intentos, y tarda un poco más.

Es por esto, que el modo aleatorio hay que tener cuidado con los intentos que se configuran, ya que si son muy altos podría estar buscando cuando no hay solución, pero si son muy bajos parará antes de encontrar todas o algunas de las soluciones. Este problema no lo tiene el modo determinista, que es capaz de determinar si existe una o varias soluciones posibles en la misma ejecución.

Capítulo 12

Manuales del sistema

12.1. Manual de ejecución

12.1.1. Ejecución

Para la correcta ejecución de la aplicación lo primero es necesario encontrarse en el directorio en el que se encuentre el archivo assignStudents.jar y tener instalado Java 8 o una versión más actualizada. Por último, por línea de comandos, es necesario ejecutar el siguiente comando:
java -jar assignStudents <programmeMode><enrollmentFile><scheduleFile> <initialAssignment><notAssignment><restriction><preferencesStudent>

- **programmeMode**: modo que se desea ejecutar, y puede ser 'grupal' o 'individual'.
- **enrollmentFile**: matrícula de los alumnos.
- **scheduleFile**: horario de las clases.
- **initialAssignment**: asignaciones iniciales de el/los alumno/s.
- **notAssignment**: asignaciones que no se deben realizar.
- **restriction**: restricciones horarias de los alumnos.
- **preferencesStudent**: preferencias horarias de los alumnos.

Con

12.1.2. Detención

En el caso del modo grupal (Asignación a alumnos), el programa finaliza cuando se acaba de ejecutarse el algoritmo, escribiendo los ficheros de salida que se

explican en la sección 12.2.4. Existe una forma de que se guarden los ficheros de la mejor solución encontrada hasta ese momento, y sería pulsando la letra *G*.

En el caso de modo individual (Asignaciones a un alumno), el programa finaliza cuando termina el algoritmo, creando un Excel y un fichero .txt para cada solución encontrada por el algoritmo.

12.2. Manual de usuario

12.2.1. Requisitos del sistema

Para poder ejecutar el programa, es necesario que se tenga instalado Java 8 o una versión superior, ya que el sistema no funcionará con versiones anteriores. Esto es así porque en la versión 8 se introdujo programación funcional y expresiones lambda.

12.2.2. Ejecutar el sistema

El sistema está encapsulado en un archivo .jar con el nombre assignStudents.jar. El sistema debe recibir parámetros que dependen de la opción que queramos ejecutar. En el caso de querer ejecutar el modo grupal, como primer parámetro se debe pasar "grupal" y en el caso del modo individual, se debe pasar "individual".

Para ejecutar el programa, se debe ejecutar el siguiente comando dependiendo de la opción:
java -jar assignStudents <ProgrammeMode><enrollmentFile><scheduleFile>
<initialAssignment><notAssignment><restriction><timePreferences>

Los tres primeros parámetros son obligatorios, mientras que los últimos cuatro pueden no existir, pudiendo rellenar ese parámetro con cualquier valor si no se quiere o no se necesita para el funcionamiento. En el caso de que los ficheros no obligatorios se rellenen con un nombre de fichero existente, debe de cumplir el formato que se va a describir próximamente.

Además, el sistema necesita una serie de archivos de configuración que deben colocarse en el mismo directorio en el que se encuentra el archivo assingStudents.jar y deben tener el nombre exacto con el que nos vamos a referir a ellos a continuación:

- **collisionExceptions**: fichero que contiene la información de los grupos que no pueden colisionar entre ellos.
- **mandatoryCollisions**: fichero que contiene la información de los grupos que no colisionan sus horarios pero forzamos su colisión.
- **preferences**: fichero que contiene la información de los grupos que queremos asignar al mismo tiempo.

- **subjectAbv**: Fichero que contiene las abreviaturas de las asignaturas.

En el caso de que se haya elegido como parámetro de opción la opción grupal, también es necesaria la existencia de este fichero:

- **geneticParameters.properties**: fichero que contiene los parámetros necesarios en el Algoritmo Genético.

Por otro lado, en el caso de la opción individual necesita de este fichero:

- **backtracking.properties**: fichero que contiene los datos necesarios para el Algoritmo de Backtracking.

Cuando se lanza la ejecución del algoritmo se mostrará un mensaje de error informativo para el usuario en caso de que la entrada proporcionada no sea correcta. Los errores pueden producirse por distintas causas, ya sea la no existencia del fichero o que su formato no es válido.

Una vez que los ficheros sean correctos y se lance la aplicación, el algoritmo imprimirá por consola información sobre la ejecución. Dependiendo del modo pasado por parámetro, la información será una u otra.

- En el caso del modo grupal, se imprimirá cada 20 generaciones la información, y al final de la ejecución, se generará un directorio llamado *results* con el resultado de la mejor solución encontrada, donde podremos encontrar los resultados del algoritmo en archivo de texto plano para cada alumno y asignatura, además de un archivo Excel con los resultados.
- En el caso del modo individual, se imprimirá la información cada vez que se encuentre una solución, y al final de la ejecución se formará un directorio llamado *resultsStudent* con la hora, donde dentro saldrán todas las soluciones en un archivo Excel con una hoja por cada solución, y un archivo de texto plano por cada solución.

12.2.3. Ficheros del sistema

En esta sección se muestra el contenido de cada archivo de configuración y su formato, además de su necesidad en la ejecución.

Matrículas

El fichero de matriculas (enrollmentFile) tiene un formato excel con extensión .xlsx que contiene la información sobre las matriculas. Cada fila representa un alumno y su matrícula a una asignatura, ya que es la forma en la que se utiliza en la Escuela de Ingeniería Informática de la Universidad de Oviedo, como se puede ver en la figura 12.1.

1								
2	ALUMNOS CON MATRICULA EN ESTADO ACTIVA /ANULADA /BORRADA /PROVISIONAL /TRÁMITE DE ANULACIÓN POR PLAN DE ESTUDIOS							
3								
4	**Universidad de Oviedo**							
5	**Curso Académico:**	2017-2018						
6	**Centro:**	Escuela de Ingeniería Informática						
7	**Localidad:**	Oviedo						
8								
9	**Núm.Doc.**	**Primer Apellido**	**Segundo Apellido**	**Nombre**	**Núm. Asig**	**Asig. en Inglés**	**Asignaturas**	**Curso**	**Temporalidad**
10	PalSotPed	Palacios	Soto	Pedro José	8	N	Repositorios de Información	Tercero	S1

Figura 12.1: Fichero de matrícula de los alumnos.

Las primeras 8 líneas están reservadas para la información del fichero y son ignoradas por el programa. La línea número 9 está reservada para encabezados de la tabla, por lo que la información tiene que comenzar en la línea 10.

Para cada matrícula se debe indicar los siguientes datos:

1. NumDoc: ID del estudiante.
2. Primer apellido del estudiante.
3. Segundo apellido del estudiante.
4. Nombre del estudiante.
5. Num Subj: número de asignaturas en las que está matriculado.
6. English: determina si el alumno está matriculado en la asignatura en Español ("N") o en Inglés ("S").
7. Subject: Nombre de la asignatura a la que está matriculado.
8. Course: Curso en el que importate da esa asignatura, que puede ser "Primero", "Segundo", "Tercero" o "Cuarto".
9. Semester: Semestre en el que se cursa la asignatura. Puede ser "S1" si es del primer semestre, y "S2" si es del segundo semestre.

En el caso del modo individual, en este archivo solo se encontrará la matrícula de un alumno, para el que se deseen enumerar algunas soluciones posibles.

Horario escolar

El fichero del horario escolar (scheduleFile), es un Excel con extensión .csv con el formato de Google Calendar y lo proporciona la Universidad. Contiene el horario de cada grupo que hay en la Escuela de Ingeniería Informática para todo el semestre, y debe encontrarse obligatoriamente.

Cada fila tiene datos separados por comas, que especifican el grupo al que pertenece la información, la fecha de inicio y fin de la clase, una descripción y dónde se va a realizar. Un ejemplo se puede ver en la figura 12.2.

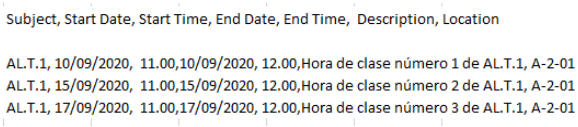

Figura 12.2: Fichero del horario escolar.

Asignaciones iniciales

El fichero de las asignaciones iniciales (initialAssignments), es un excel con extensión .xlsx que contiene las asignaciones iniciales siguiendo el formato que se usa en la escuela para hacer las asignaciones. Este fichero no es de carácter obligatorio. Cada fila representa un alumno y su asignación inicial a una asignatura, en el que tiene esta información

1. ID.
2. FIRST SURNAME.
3. SECOND SURNAME.
4. NAME.

Después, aparecen las asignaciones de las asignaturas por cada tipo de clase, siendo primero la teoría *THEORY*, luego el seminario *SEMINARY* y por último el laboratorio *LABORATORY*.

Un ejemplo de este fichero se puede encontrar en la figura 12.3.

	ID	FIRST SURNAME	SECOND SURNAME	NAME	THEORY	AL	Cal	Emp	FI	IP
2	PalSotPed	Palacios	Soto	Pedro José						
3	ValCabAli	Valle	Cabrero	Alicia		1	1			

Figura 12.3: Fichero de asignaciones iniciales.

Asignaciones que no son necesarias

El fichero de las asignaciones que no son necesarias (notAssign), es un Excel con extensión .csv que contiene las asignaciones que no son necesarias para un alumno para un tipo de clase de una asignatura. Cada fila tiene datos separados por comas. El primer dato es el ID del alumno, el segundo es la abreviatura de la asignatura y el tercero es el tipo de clase al que no se le quiere que se le asigne *Teoría (T), Seminario (S) o Laboratorio (L)*. Un ejemplo se puede ver en la figura 12.4.

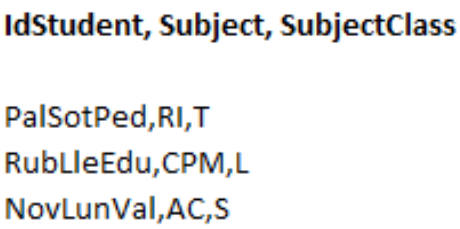

Figura 12.4: Fichero de asignaciones que no son necesarias

Restricciones horarias

El fichero de las restricciones (restrictions), es un Excel con extensión .csv que contiene las restricciones de los alumnos a ciertas franjas horarias en días específicos de la semana. Cada fila tiene datos separados por comas. El primer dato es el ID del alumno, el segundo es el día de la semana en inglés, el tercero es la hora de inicio y la cuarta es la hora fin. En las horas, las horas y los minutos se separan por un punto ".". Un ejemplo se puede ver en la figura 12.5.

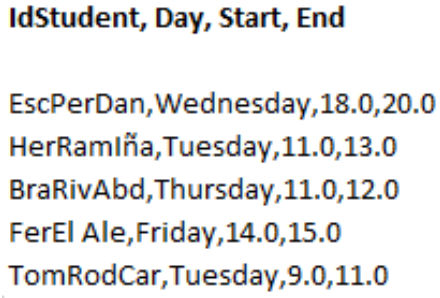

Figura 12.5: Fichero de las restricciones horarias de los alumnos.

Preferencias horarias

Este fichero es igual que el anterior 12.2.3, pero el fin es otro, ya que son horas en las que los estudiantes prefieren no tener clase.

Excepciones de colisiones

El fichero de las excepciones de colisiones (collisionExceptions) nos da información sobre grupos que colisionan pero que no se debe tener en cuenta a la hora de ejecutar el algoritmo. Nos ayuda con grupos que se cursan a la misma hora, el mismo día de la semana pero en semanas distintas.
Cada fila contiene una lista de grupos separados por ";", que queremos que el algoritmo los trate como si no colisionaran. Un ejemplo se puede ver en la figura 12.6.

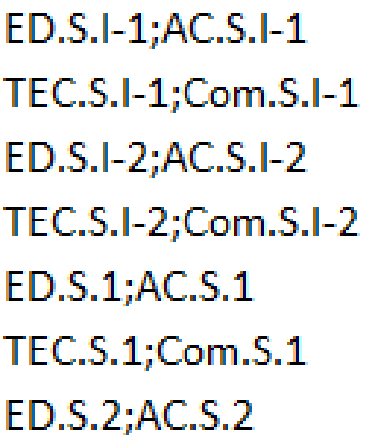

Figura 12.6: Fichero de excepciones de colisiones.

Colisiones obligatorias

El fichero de colisiones obligatorias (mandatoryCollisions) nos da la información de grupos que queremos que colisionen aunque realmente no lo hagan. Esto ocurre en el caso en el que queremos que si un estudiante es del grupo de teoría 1, vaya al grupo de seminario 1 o 2, pero nunca al 3 o 4.

Cada fila contiene una lista de grupos separados por ";", que queremos que el algoritmo los trate como si colisionaran. Un ejemplo se puede ver en la figura 12.7.

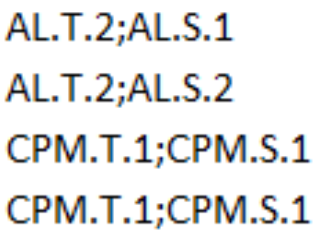

Figura 12.7: Fichero de colisiones obligatorias.

Preferencias de asignación

El fichero de preferencias de asignación (preferences) nos da información sobre grupos, que cuando se asigna uno, el algoritmo intenta asignarle el resto de grupos de la fila en el caso de que esté matriculado en ellas.

Cada fila del fichero contiene una lista de grupos separados por ";", que queremos que el algoritmo asigne en el momento que se asigne uno de la fila. Un ejemplo del fichero se puede ver en la figura 12.8.

```
Cal.S.1;AL.S.1;IP.S.1;Emp.S.1
Cal.S.2;AL.S.2;IP.S.2;Emp.S.2
Cal.S.3;AL.S.3;IP.S.3;Emp.S.3
Cal.S.I-1;AL.S.I-1;IP.S.I-1
Cal.S.I-2;AL.S.I-2;IP.S.I-2
```

Figura 12.8: Fichero de preferencias de asignación.

Abreviaturas de las asignaturas

El fichero de las abreviaturas de las asignaturas (subjectAbv) que asigna las abreviaturas a los nombres completos de las asignaturas. Esto se tiene que realizar ya que las matriculas están hechas con el nombre completo de las asignaturas pero el fichero de Google Calendar con el horario de las asignaturas contiene las asignaturas con abreviaturas.

En cada fila se representan <NombreCompleto>;<Abreviatura>. Un ejemplo de esto se puede ver en la figura 12.9.

```
Metodología de la Programación;MP
Computabilidad;Com
Ondas y Electromagnetismo;OyE
Tecnología y Paradigmas de Programación;TPP
Tecnología Electrónica de Computadores;TEC
Comunicación Persona-Máquina;CPM
Fundamentos de Informática;FI
```

Figura 12.9: Fichero de las abreviaturas de las asignaturas.

Propiedades del Algoritmo Genético

El fichero de propiedades del Algoritmo Genético (geneticParameters.properties) contiene las propiedades que contienen pares clave-valor separados por el símbolo "=". Se configuran los siguiente parámetros:

- **mutationProb**: Probabilidad de mutación que se aplica en el Algoritmo Genético.
- **crossoverProb**: Probabilidad de cruce que se aplica en el Algoritmo Genético.

- **populationSize**: Número de individuos que tendrá la población en cada generación.
- **semester**: Semestre que se tendrá en cuenta para el fichero de matriculas.
- **unSolvedAssignmentsWeighted**: Peso en la función fitness al número de asignaciones sin resolver.
- **totalVarianceWeighted**: Peso en la función fitness a la varianza total.
- **maxDifferenceBetweenGroupsWeighted**: Peso en la función fitness a la diferencia máxima en el número de alumnos entre grupos de la misma asignatura.
- **totalNumberOfFreeSlotsWeighted**: Peso en la función fitness al número de horas libres entre clases.
- **totalDaysWithLowNumOfClassesWeighted**: Peso en la función fitness al número de días con menos de dos horas de clase.
- **totalUnsatisfiedPreferencesWeighted**: Peso en la función fitness al número de preferencias horarias no satisfechas.

Los valores asignados deben ser números excepto el semestre que puede ser "S1" o "S2". Un ejemplo del fichero es el que se muestra en la figura 12.10.

```
mutationProb=0.05
crossoverProb=0.95
generations=1500
populationSize=200
semester=S1
unSolvedAssignmentsWeighted=100
totalVarianceWeighted=1
maxDifferenceBetweenGroupsWeighted=10
totalNumberOfFreeSlotsWeighted=0.08
totalDaysWithLowNumOfClassesWeighted=0.2
totalUnsatisfiedPreferencesWeighted=25
```

Figura 12.10: Fichero de las propiedades del Algoritmo Genético.

Propiedades de Backtracking

El fichero de propiedades del Algoritmo de Backtracking (backtracking.properties) contiene las propiedades que contienen pares clave-valor separados por el símbolo "=". Se configuran los siguiente parámetros:

- **solutionSize**: Número de soluciones que buscará el algoritmo de Backtracking.
- **semester**: Semestre que se tendrá en cuenta para el fichero de matriculas.

- **attemps**: Número de intentos que se desea que haga el algoritmo de Backtracking.
- **type**: Tipo de algoritmo de Backtracking que queremos ejecutar, ya puede ser determinista o aleatorio.

El primer valor debe ser un valor numérico, mientras que el segundo valor, el semestre, puede ser "S1" o "S2". El tipo de algoritmo *"type"* puede ser "determinista" o "aleatorio" y el número de intentos *"attemps"* debe ser un valor numérico. Un ejemplo del fichero es el que se muestra en la figura 12.11.

```
solutionSize=10
semester=S1
attempts=1000
type=aleatorio
```

Figura 12.11: Fichero de las propiedades del Algoritmo de Backtracking.

12.2.4. Ficheros de salida

El sistema genera diferentes archivos de salida dependiendo del modo que se haya seleccionado como entrada en el mismo directorio en el que se encuentra el archivo assignStudents.jar.

- Modo grupal: archivos de texto para los alumnos y las asignaturas y un archivo excel en el formato utilizado por la Escuela de Ingeniería Informática para la realización de las asignaciones.
- Modo individual: archivos de texto para las distintas soluciones y un archivo excel con una hoja por cada solución con las asignaciones y resumen de la calidad de la solución.

Modo grupal

summary.txt *summary.txt* es un fichero que contiene un resumen de la calidad de la solución generada. Un ejemplo del archivo se puede ver en la figura 12.12.

BEST INDIVIDUAL	
Number of collisions	20
Max difference in group	5.0
Sum of differences between groups	120
Max variance	6.25
Total variance	47.481377167104974
Total number of free slots	6265.0
Total days with 2 hours or less	423.0
Total preferences don't satisface	0.0

Figura 12.12: Fichero de resumen de la solución generada.

unsolvedAssignments.txt *unsolvedAssignments.txt* es un fichero que contiene una lista de las asignaciones que no se pudieron realizar por el algoritmo. Un ejemplo de este fichero se puede ver en la figura 12.13.

STUDENT	SUBJECT CLASS
Nourdine Freije Morán	IAE.S
Alejo Andrés Castro	TEC.S
Fernando de la Tuñón Melendi	CVVS.S
Alejo Andrés Castro	AC.S
Roberto de la Rubio Diaz	SDM.L
Elena Urrutia Muñiz	TEC.S
Nuria Flórez Lewis	IP.T

Figura 12.13: Fichero con las asignaciones que no se han podido realizar.

excelResults.xlsx *excelResults.xlsx* es un fichero Excel que contiene los resultados de todas las asignaciones para todos los estudiantes. Para cada alumno matriculado se genera una fila, en la que en las primeras columnas se encuentra información sobre el estudiante. El formato de este fichero se puede ver en los requisitos.

<StudentID>.txt Se genera un fichero por cada estudiante en un directorio llamado *students* que se encuentra dentro de la carpeta *results*, y contiene el horario y las asignaciones del alumno, como se puede ver tanto en la figura 12.14 como en la 12.15.

NAME: Luis Abad Cano
ID: AbaCanLui

TIMETABLE:

TIMETABLE OF ID: AbaCanLui | Luis Abad Cano							
9:0							
9:30							
10:0							
10:30							
11:0							
11:30							
12:0							
12:30							
13:0							
13:30							
14:0					CPM.S.I-2		
14:30					CPM.S.I-2		
15:0		AC.T.I-1	CPM.L.I-1	ED.T.I-1	AC.S.I-2		
15:30		AC.T.I-1	CPM.L.I-1	ED.T.I-1	AC.S.I-2		
16:0		AC.T.I-1	CPM.L.I-1	ED.T.I-1			
16:30		AC.T.I-1	CPM.L.I-1	ED.T.I-1			
17:0	CPM.T.I-1		CPM.T.I-1				
17:30	CPM.T.I-1		CPM.T.I-1				
18:0		ED.L.I-1	AC.L.I-1				
18:30		ED.L.I-1	AC.L.I-1				
19:0		ED.L.I-1	AC.L.I-1				
19:30		ED.L.I-1	AC.L.I-1				
20:0							
20:30							

Figura 12.14: Fichero con el horario del alumno.

STUDENT ASSIGNATIONS

SUBJECT CLASS	GROUP
CPM.S.I	CPM.S.I-2
CPM.L.I	CPM.L.I-1
CPM.T.I	CPM.T.I-1
AC.L.I	AC.L.I-1
AC.S.I	AC.S.I-2
AC.T.I	AC.T.I-1
ED.S.I	ED.S.I-2
ED.T.I	ED.T.I-1
ED.L.I	ED.L.I-1

Figura 12.15: Fichero con las asignaciones del alumno.

<subjectAbbreviation>.txt Se genera un fichero por cada asignatura en un directorio llamado *subjects* que se encuentra dentro de la carpeta *results*. En este fichero se encuentra la distribución de los alumnos en los grupos de la asignatura para así poder visualizar información sobre la homogeneidad, como se puede ver en la figura 12.16.

SUBJECT: Arquitectura de Computadores
COURSE: Segundo

**********SUBJECT CLASSES**********

CLASS: AC.L.I
TOTAL NUMBER OF STUDENTS: 47
STUDENT DISTRIBUTION IN GROUPS:

GROUP ID	NUMBER OF STUDENTS
AC.L.I-1	12
AC.L.I-3	12
AC.L.I-2	12
AC.L.I-4	11

Figura 12.16: Fichero con la distribución de los grupos de la asignatura.

Modo individual

excelResults.xlsx *excelResults.xlsx* es un fichero Excel que contiene las soluciones que se han encontrado. Por cada hoja, se encuentra una de las soluciones en el que se muestra sus asignaciones y un resumen de la calidad de la solución. El formato de este fichero se puede ver en los requisitos.

<StudentID><numSolution>.txt Se genera un fichero por cada solución de la asignación del estudiante en un directorio llamado *solutions* que se encuentra dentro de la carpeta *results*, y contiene el horario y las asignaciones del alumno para esa solución. El formato del fichero es el mismo que se encuentra en el modo grupal.

Capítulo 13

Conclusiones y ampliaciones

13.1. Conclusiones

Se ha diseñado un sistema que integra el actual con las nuevas funcionalidades para resolver las mejoras establecidas en el sistema actual de la Escuela de Ingeniería Informática de la Universidad de Oviedo. Debido a la existencia de un sistema inicial, ha sido necesario analizarlo minuciosamente y desarrollar una solución específica.

Las pruebas de integración han demostrado que las funcionalidades se han introducido de forma eficaz. Los experimentos se han realizado con matrículas reales, proporcionadas por la Escuela de Ingeniería Informática y han dado como resultado que el uso de las funcionalidades facilita el trabajo al usuario sin alterar en gran medida la calidad de la solución. Además, los experimentos nos presenta cual es el mejor uso de estas funcionalidades.

El prototipo ha sido validado por el antiguo responsable de realizar las asignaciones de alumnos en la escuela, y será posteriormente entregado a la Escuela de Ingeniería Informática, y así poder realizaren un futuro alguna mejora.

13.2. Ampliaciones

Algunas de las ampliaciones que se van a nombrar se han tenido en cuenta en la herramienta base, y se puede ver en [1].

- Método exacto para determinar si el horario de un alumno puede ser resuelto: En este momento no hay ningún algoritmo implementado en el sistema para comprobar si todas las tareas de un alumno pueden ser resueltas sin colisiones. Esta información podría ser útil para el usuario para saber qué tareas no se han resuelto porque son imposibles de resolver y qué tareas no fueron resueltas porque el algoritmo no fue capaz de encontrar una solu-

ción por sí mismo. Se podría diseñar otro algoritmo para comprobar esto y luego notificar al usuario con las conclusiones.

- Interfaz de usuario: El prototipo desarrollado sólo tiene una sencilla interfaz de línea de comandos con la que el usuario sólo interactúa para inicializar el algoritmo y poco más. Sería útil que el sistema tuviera una interfaz de usuario más completa y atractiva en la que el usuario pudiera echar un vistazo a las soluciones generadas hasta el momento, compararlas y en general tener una mejor experiencia de usuario.

- Pesos normalizados: la homogeneidad del grupo y la calidad del horario están establecidos ya, teniendo mucha importancia la homogeneidad. Es posible que se prefiera jugar con la importancia que se le da a los factores, por lo que una buena idea sería normalizar los pesos y así se podría pasar como parámetro la importancia que se le quiera dar de tal forma que el algoritmo sigue funcionando eficientemente.

Capítulo 14

Presupuesto

El presupuesto se basa en la planificación del proyecto que se ha redactado anteriormente, y estima el coste real del desarrollo.

Se tiene en cuenta que el desarrollo del proyecto se lleva a cabo por el estudiante en su casa. Se van a presentar dos presupuestos: el de costes y el del cliente.

14.1. Definición de la empresa

14.1.1. Personal

Se tiene dos personas para el proyecto, y su sueldo bruto en el periodo de tiempo que dura el proyecto, según se encuentra en *glassdoor*.

Personal

Personal	Núm.	Sueldo bruto seis meses	Coste Salarial Proyecto	TOTAL
Investigador	1	8211	11,730.00 €	11,730.00 €
Investigador Jefe	1	12500	17,857.14 €	17,857.14 €

Figura 14.1: Personal.

14.1.2. Productividad del personal

Hay que tener en cuenta la implicación que tiene el personal en el desarrollo del proyecto.

Productividad del personal

Personal	TOTAL	Prod(%)	Coste Directo	CI(%)	Coste Indirecto
Investigador	11,730.00 €	90.00%	10,557.00 €	10.00%	1,173.00 €
Investigador Jefe	17,857.14 €	20.00%	3,571.43 €	80.00%	14,285.71 €
	29,587.14 €		14,128.43 €		15,458.71 €

Figura 14.2: Productividad del personal.

14.1.3. Costes indirectos

Hay un servicio que hay que tener en cuenta en el proyecto ya que se realiza en casa y es el coste de electricidad en los meses del proyecto.

Costes indirectos

Servicio	Coste mes	Coste proyecto
Consumos de electricidad	30.00 €	180.00 €
		180.00 €

Figura 14.3: Costes indirectos.

14.1.4. Costes de los medios de producción

Para poder realizar el proyecto es necesario diversos medios tecnológicos que vienen descritos a continuación.

Coste de los medios de producción

Equipo / Licencia	Medida	Unid.	Precio	Coste Total	Coste proyecto	Tipo	Plazo
Portatil	Unidad	1	800.00 €	800.00 €	800.00 €	Amort.	1
Licencia Microsoft Project	Mensual	6	46.50 €	279.00 €	279.00 €	Alquil.	-
Licencia Microsoft Office	Mensual	6	10.00 €	60.00 €	60.00 €	Alquil.	-
Soporte GitHub	Mensual	6	21.00 €	126.00 €	126.00 €	Alquil.	-
Lincencia IntelliJ	Mensual	6	49.90 €	299.40 €	299.40 €	Alquil.	-
					1,564.40 €		

Figura 14.4: Medios de producción.

14.1.5. Horas productivas y no productivas

Las horas de trabajo deben de ser también asignadas a horas productivas y no productivas, teniendo en cuenta los porcentajes anteriores y las horas del proyecto.

Horas productivas y no productivas

Personal	Prod(%)	Horas/projecto	Horas prod./projecto (por persona)	Horas prod. (total empresa)
Investigador	90.00%	1,456	1310.4	1310.4
Investigador Jefe	20.00%	1,456	291.2	291.2
				1,601.60 €

Figura 14.5: Horas productivas y no productivas

14.1.6. Precio hora

A continuación se establecen los precios/hora, que incluyen todos los costes indirectos. Pero hay que tener en cuenta el precio/hora sin beneficio, que será utilizado después para el cálculo de costes.

Precio hora

Personal	Precio/hora	Horas prod. (total empresa)	Facturación	Precio/hora (sin beneficio)
Investigador	24.00 €	1310.4	31,449.60 €	18.00 €
Investigador Jefe	29.00 €	291.2	8,444.80 €	21.75 €
		1601.6	39,894.40 €	

Figura 14.6: Precio hora.

14.1.7. Resumen

Un resumen de los cálculos anteriores lo podemos ver en la siguiente tabla.

RESUMEN

Nº	Concepto	IMPORTE
1	Total de los costes directos	14,128.43 €
2	Total de los costes indirectos	17,203.11 €
3	Suma de los costes directos e indirectos	31,332 €
4	Beneficio deseado (25%)	7,832.89 €
5	Coste total (sumo de los costes directos, indirectos y beneficios)	39,164.43 €
6	Facturación posible en función de las horas de producción y de los precios por hora calculados	39,894.40 €
7	Margen entre el coste total y la facturación (relación entre 5 y 6)	1.83%

Figura 14.7: Resumen empresa.

14.2. Descripción del proyecto

El proyecto se divide en 8 partidas:

1. Partida 1: Planificación.
2. Partida 2: Revisión bibliográfica.
3. Partida 3: Análisis.
4. Partida 4: Implementación.
5. Partida 5: Pruebas.
6. Partida 6: Experimentos.
7. Partida 7: Validación del cliente.
8. Partida 8: Cierre del proyecto.

14.3. Presupuesto de costes

14.3.1. Partida 1: Planificación

Planificación					Precio:	55.88 €
Descripción	Cantidad	Uds	Precio	Subtotal(2)	Subtotal(1)	Total
Planificación						55.88 €
Reunión con el tutor					19.88 €	
Investigador	0.5	Horas	18.00 €	9		
Investigador Jefe	0.5	Horas	21.75 €	10.875		
Definición de los pasos del proyecto					36.00 €	
Investigador	2	Horas	18.00 €	36		

Figura 14.8: Partida 1: Planificación

14.3.2. Partida 2: Revisión bibliográfica

Revisión bibliográfica									PRECIO	2,880.00 €
I1	I2	I3	Descripción	Núm.	Uds.	Precio	Subtotal (3)	Subtotal (2)	Subtotal (1)	Total
1			Revisión del Trabajo de Fin de Grado de Gonzalo De La Cruz Fernández						1,440.00 €	2,880.00 €
	1		Lectura					720.00 €		
		1	*Investigador*	40	horas	18.00 €	720.00 €			
	2		Esquema de funcionamiento					720.00 €		
		1	*Investigador*	40	horas	18.00 €	720.00 €			
2			Investigación de algoritmos genéticos, voraces y de backtracking						1,440.00 €	
	1		*Investigador*	80	horas	18.00 €	1,440.00 €			

Figura 14.9: Partida 2: Revisión bibliográfica

14.3.3. Partida 3: Análisis

Análisis								PRECIO	4,483.88 €
I1	I2	I3	I4	Descripción	Núm.	Uds.	Precio	Subtotal (2)	Subtotal (1)	Total
1				Reunión con el cliente y el tutor sobre las mejoras y restricciones					19.88 €	4,483.88 €
	1			*Investigador*	0.5	horas	18.00 €	9.00 €		
	2			*Investigador Jefe*	0.5	horas	21.75 €	10.88 €		
2				Familiarización con el código					1,440.00 €	
	1			*Investigador*	80	horas	18.00 €	1,440.00 €		
3				Definición de las mejoras					2,160.00 €	
	1			Asignaciones iniciales						
		1		*Investigador*	24	horas	18.00 €	432.00 €		
	2			Asignaciones que no son necesarias						
		1		*Investigador*	24	horas	18.00 €	432.00 €		
	3			Restricciones						
		1		*Investigador*	24	horas	18.00 €	432.00 €		
	4			Preferencias						
		1		*Investigador*	24	horas	18.00 €	432.00 €		
	5			Modo individual						
		1		*Investigador*	24	horas	18.00 €	432.00 €		
4				Esquema de como se integran las mejoras					864.00 €	
	1			*Investigador*	48	horas	18.00 €	864.00 €		

Figura 14.10: Partida 3: Análisis

14.3.4. Partida 4: Implementación

Implementación										PRECIO	7,344.00 €
I1	I2	I3	I4	Descripción	Núm.	Uds.	Precio	Subtotal (4)	Subtotal (3)	Subtotal (2)	Subtotal (1)	Total
1				Modo grupal							5,328.00 €	7,344.00 €
	1			Asignaciones iniciales						1,296.00 €		
		1		Lectura fichero					576.00 €			
			1	Investigador	32	horas	18.00 €	576.00 €				
		2		Integración					720.00 €			
			1	Investigador	40	horas	18.00 €	720.00 €				
	2			Asignaciones que no son necesarias						1,296.00 €		
		1		Lectura fichero					576.00 €			
			1	Investigador	32	horas	18.00 €	576.00 €				
		2		Integración					720.00 €			
			1	Investigador	40	horas	18.00 €	720.00 €				
	3			Restricciones						1,296.00 €		
		1		Lectura fichero					576.00 €			
			1	Investigador	32	horas	18.00 €	576.00 €				
		2		Integración					720.00 €			
			1	Investigador	40	horas	18.00 €	720.00 €				
	4			Preferencias						1,440.00 €		
		1		Integración					1,440.00 €			
			1	Investigador	80	horas	18.00 €	1,440.00 €				
1				Modo individual							2,016.00 €	
	1			Backtracking						2,016.00 €		
		1		Determinista					1,008.00 €			
			1	Investigador	56	horas	18.00 €	1,008.00 €				
		2		Aleatorio					1,008.00 €			
			1	Investigador	56	horas	18.00 €	1,008.00 €				

Figura 14.11: Partida 4: Implementación

14.3.5. Partida 5: Pruebas

Pruebas										PRECIO	1,152.00 €
I1	I2	I3	I4	Descripción	Núm.	Uds.	Precio	Subtotal (4)	Subtotal (3)	Subtotal (2)	Subtotal (1)	Total
1				Pruebas unitarias							576.00 €	1,152.00 €
	1			Modo grupal						432.00 €		
		1		Asignaciones iniciales					108.00 €			
				Investigador	*6*	horas	18.00 €	108.00 €				
		2		Asignaciones que no son necesarias					108.00 €			
				Investigador	*6*	horas	18.00 €	108.00 €				
		3		Restricciones					108.00 €			
				Investigador	*6*	horas	18.00 €	108.00 €				
		4		Preferencias					108.00 €			
				Investigador	*6*	horas	18.00 €	108.00 €				
	2			Modo individual						144.00 €		
		1		Backtracking								
			1	Determinista					72.00 €			
				Investigador	*4*	horas	18.00 €	72.00 €				
			2	Aleatorio					72.00 €			
				Investigador	*4*	horas	18.00 €	72.00 €				
2				Pruebas de integración							576.00 €	
	1			Modo grupal						432.00 €		
		1		Asignaciones iniciales					108.00 €			
				Investigador	*6*	horas	18.00 €	108.00 €				
		2		Asignaciones que no son necesarias					108.00 €			
				Investigador	*6*	horas	18.00 €	108.00 €				
		3		Restricciones					108.00 €			
				Investigador	*6*	horas	18.00 €	108.00 €				
		4		Preferencias					108.00 €			
				Investigador	*6*	horas	18.00 €	108.00 €				
	2			Modo individual						144.00 €		
		1		Backtracking								
			1	Determinista					72.00 €			
				Investigador	*4*	horas	18.00 €	72.00 €				
			2	*Aleatorio*					72.00 €			
				Investigador	*4*	horas	18.00 €	72.00 €				

Figura 14.12: Partida 5: Pruebas

14.3.6. Partida 6: Validación del cliente

				Validación del cliente						PRECIO	39.75 €
I1	I2	I3	I4	Descripción	Núm.	Uds.	Precio	Subtotal (4)	Subtotal (1)	Subtotal (1)	Total
1				Reunión con el cliente para que valide los resultados						39.75 €	39.75 €
	1			*Investigador*	1	horas	18.00 €	18.00 €			
	2			*Investigador jefe*	1	horas	21.75 €	21.75 €			

Figura 14.13: Partida 6: Validación del cliente

14.3.7. Partida 7: Experimentos

				Experimentos							PRECIO	3,264.00 €
I1	I2	I3	I4	Descripción	Núm.	Uds.	Precio	Subtotal (4)	Subtotal (3)	Subtotal (2)	Subtotal (1)	Total
1				Creación instancias						432.00 €	3,264.00 €	3,264.00 €
	1			Modo grupal					288.00 €			
		1		*Investigador*	16	horas	18.00 €	288.00 €				
	2			Modo individual					144.00 €			
		1		*Investigador*	8	horas	18.00 €	144.00 €				
2				Experimentar						2,832.00 €		
	1	3		Pesos función fitness					954.00 €			
				Investigador	*24*	horas	18.00 €	432.00 €				
		4		*Investigador jefe*	24	horas	21.75 €	522.00 €				
	2			Modo grupal					1,590.00 €			
		1		*Investigador*	40	horas	18.00 €	720.00 €				
		2		*Investigador jefe*	40	horas	21.75 €	870.00 €				
	3			Modo individual					288.00 €			
		1		*Investigador*	*16*	horas	18.00 €	288.00 €				

Figura 14.14: Partida 7: Experimentos

14.3.8. Partida 8: Cierre del proyecto

				Cierre del proyecto						PRECIO	951.00 €
I1	I2	I3	I4	Descripción	Núm.	Uds.	Precio	Subtotal (3)	Subtotal (2)	Subtotal (1)	Total
1				Definición de las mejoras					72.00 €	951.00 €	951.00 €
	1			*Investigador*	4	horas	18.00 €	72.00 €			
2				Conclusiones					87.00 €		
	1			*Investigador*	*4*	horas	21.75 €	87.00 €			
3				Revisión					792.00 €		
	1			Proyecto							
		1		*Investigador*	20	horas	18.00 €	360.00 €			
	2			Documentación							
		1		*Investigador*	24	horas	18.00 €	432.00 €			

Figura 14.15: Partida 8: Cierre del proyecto

14.3.9. Agregación final del presupuesto de costes

Agregando todos los costes de las 8 partidas desglosadas anteriormente, se obtiene el siguiente resumen

Partidas	Nombre	Costo
1	**Planificación**	55.88 €
	Reunión con el tutor	19.88 €
	Definición de los pasos del proyecto	36.00 €
2	**Revisión bibliográfica**	2,880.00 €
	Revisión del Trabajo de Fin de Grado de Gonzalo De La Cruz Fernández	1,440.00 €
	Investigación de algoritmos genéticos, voraces y de backtracking	1,400.00 €
3	**Análisis**	4,483.00 €
	Reunión con el cliente y el tutor sobre las mejoras y restricciones	19.88 €
	Familiarización con el código	1,440.00 €
	Definición de las mejoras	2,160.00 €
	Esquema de como integrar las mejoras	864.00 €
4	**Implementación**	7,344.00 €
	Modo grupal	5,328.00 €
	Modo individual	2,016.00 €
5	**Pruebas**	1,152.00 €
	Pruebas unitarias	576.00 €
	Pruebas de integración	576.00 €
6	**Validación del cliente**	39.75 €
	Reunión con el cliente para que valide los resultados	39.75 €
7	**Experimentos**	3,264.00 €
	Creación de las instancias	432.00 €
	Experimentar	2,832.00 €
8	**Cierre del proyecto**	951.00 €
	Definición de las amplicaciones	72.00 €
	Conclusiones	87.00 €
	Revisión	732.00 €
TOTAL		**20,169.63 €**

Figura 14.16: Resumen.

14.4. Presupuesto del cliente

Para la elaboración del presupuesto del cliente se ha decidido tomar un porcentaje de beneficio del 25 %. Además, hay costes que se considera mejor no enviarse al cliente, como la partida 1 y 2, por lo que el coste que conlleva esas partidas debe de ser distribuido entre las partidas del cliente de manera porcentual.

- Partida 1: Planificación: 55,88€.
- Partida 2: Revisión bibliográfica: 2.880€.
- Beneficio 25 %: 5.045,41€.

En total supone un valor de 7.978,28€para promediar entre las partidas elegidas para el presupuesto del cliente.

La suma de todos los items de las partidas que se van a dar al cliente es de 17.233,75€, por lo que hay que compensar los 7.978,28€en esta cantidad, lo que es casi el doble.

El presupuesto del cliente detallado lo podemos encontrar en la figura

Partidas	Nombre	Costo	Incremento	Coste Final
3	**Análisis**	4,483.00 €	2,075.38 €	6,558.38 €
	Reunión con el cliente y el tutor sobre las mejoras y restricciones	19.88 €	9.20 €	29.08 €
	Familiarización con el código	1,440.00 €	666.64 €	2,106.64 €
	Definición de las mejoras	2,160.00 €	999.96 €	3,159.96 €
	Esquema de como integrar las mejoras	864.00 €	399.98 €	1,263.98 €
4	**Implementación**	7,344.00 €	3,399.87 €	10,743.87 €
	Modo grupal	5,328.00 €	2,466.57 €	7,794.57 €
	Modo individual	2,016.00 €	933.30 €	2,949.30 €
5	**Pruebas**	1,152.00 €	533.31 €	1,685.31 €
	Pruebas unitarias	576.00 €	266.66 €	842.66 €
	Pruebas de integración	576.00 €	266.66 €	842.66 €
6	**Validación del cliente**	39.75 €	18.40 €	58.15 €
	Reunión con el cliente para que valide los resultados	39.75 €	18.40 €	58.15 €
7	**Experimentos**	3,264.00 €	1,511.05 €	4,775.05 €
	Creación de las instancias	432.00 €	199.99 €	631.99 €
	Experimentar	2,832.00 €	1,311.06 €	4,143.06 €
8	**Cierre del proyecto**	951.00 €	440.26 €	1,391.26 €
	Definición de las amplicaciones	72.00 €	33.33 €	105.33 €
	Conclusiones	87.00 €	40.28 €	127.28 €
	Revisión	732.00 €	338.88 €	1,070.88 €
TOTAL		**17,233.75 €**		**25,212.03 €**

Figura 14.17: Presupuesto cliente detallado.

14.4.1. Presupuesto del cliente resumido

Un resumen de las partidas del presupuesto de cliente se encuentra en la siguiente figura.

Partidas	Nombre	Importe
3	Análisis	6,558.38 €
4	Implementación	10,743.87 €
5	Pruebas	1,685.31 €
6	Validación del cliente	58.15 €
7	Experimentos	4,775.05 €
8	Cierre del proyecto	1,391.26 €
TOTAL		**25,212.03 €**

Figura 14.18: Presupuesto cliente resumido.

Apéndice A

Contenido como anexo a la memoria

A.1. Javadoc (javadoc.zip)

En este directorio se encuentran los html que representan el javadoc. El principal que engloba a todos es el index.html.

A.2. Instancias (Instancias.zip)

Aquí se puede encontrar distintas instancias para probar. En la raiz del directorio se encuentran los archivos que son obligatorios para los dos modos y dos carpetas: una carpeta 'Grupal' con instancias para ejecutar el modo grupal, y una carpeta 'Individual' con instancias para ejecutar el modo individual.

A.3. Código fuente (assignStudents.zip)

Código fuente en el que se encuentra implementado el sistema. En el directorio src, se encuentra un paquete uniovi.assign, donde se encuentra la clase Main y InputThread y separa el resto de paquetes:

- backtracking
- genetic
- greedy
- model
- parser
- programs

- reports

A.4. Ejecutable (Ejecutable.zip)

Ejecutable del sistema. Para saber como ejecutarlo, mirar el manual de usuario 12.2.

Bibliografía

[1] Gonzalo De La Cruz Fernández. "Metaheuristics for the assignment of students to class groups".

[2] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation.*

[3] John Henry Holland. *Adaptation in Natural and Artificial Systems.* 1975.

[4] Miguel Angel González y Ramiro Varela. "Tutorial sobre Algoritmo Genéticos".

[5] Emilio G Ortiz-García Antonio Portilla-Figueras Luis E Agustín-Blas Sancho Salcedo-Sanz y Ángel M Pérez-Bellido. "A hybrid grouping genetic algorithm for assigning students to preferred laboratory groups". En: (2009), Expert Systems with Applications, 36(3):7234–7241.

[6] Rhydian Lewis y Ben Paechter. "Application of the grouping genetic algorithm to university course timetabling. In European Conference on Evolutionary Computation in Combinatorial Optimization". En: (Springer, 2005), pages 144–153.

[7] Dave Corne Peter Ross y Hsiao-Lan Fang. "Successful lecture timetabling with evolutionary algorithms. In Proceedings of the ECAI, volume 94". En: (1994).

[8] Panagiotis Adamidis y Panagiotis Arapakis. "Evolutionary algorithms in lecture timetabling. In Evolutionary Computation". En: (1999. CEC 99. Proceedings of the 1999 Congress on, volume 2, pages 1145–1151. IEEE, 1999.).

[9] Lawrence Davis. "Applying adaptive algorithms to epistatic domains". En: (1985), n IJCAI, pages 162–164.

[10] Juan Ramón Pérez Pérez. "Backtracking".