



Universidad de Castilla-La Mancha
Escuela Superior de Ingeniería Informática

Trabajo Fin de Grado
Grado en Ingeniería Informática
Computación

**Optimización Multiobjetivo de la asignación
docente en el Grado de Ingeniería Informática
para evitar superposiciones y satisfacer
preferencias de horarios de los estudiantes**

José Ángel Serrano Pardo

Junio, 2024



TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Computación

Optimización Multiobjetivo de la asignación docente en el Grado de Ingeniería Informática para evitar superposiciones y satisfacer preferencias de horarios de los estudiantes

Autor: José Ángel Serrano Pardo

Tutor: José Miguel Puerta Callejón

Junio, 2024

*Aquí va la dedicatoria que cada cual
quiera escribir. El ancho se controla
manualmente*

Declaración de autoría

Yo, José Ángel Serrano Pardo , con DNI 49800390P, declaro que soy el único autor del trabajo fin de grado titulado “Optimización Multiobjetivo de la asignación docente en el Grado de Ingeniería Informática para evitar superposiciones y satisfacer preferencias de horarios de los estudiantes”, que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual, y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a ... de ... de 20 ...

Fdo.: José Ángel Serrano Pardo

Resumen

Los algoritmos genéticos son una rama de la inteligencia artificial donde se simula el proceso de la evolución biológica, se usan métodos de búsqueda heurística que utilizan procesos de selección, cruzamiento y mutación para generar soluciones óptimas o cercanas a las óptimas a problemas complejos que son difíciles de resolver mediante técnicas convencionales. A través de iteraciones sucesivas, una población inicial de soluciones potenciales evoluciona hacia una solución óptima o cercana, adaptándose al entorno definido por una función fitness específica. En este estudio, se adapta un algoritmo genético para abordar la optimización multiobjetivo en la asignación de grupos para las matriculas en el Grado de Ingeniería Informática de la Universidad de Castilla-La Mancha. El desafío incluye múltiples objetivos, como maximizar la satisfacción de los alumnos, minimizar el número de solapes entre los horarios de las matriculas de un alumno, maximizar la cohesión de grupos de teoría y práctica para asignaturas del mismo curso en los alumnos y maximizar el equilibrio del número de alumnos en los grupos de teoría y practicas en cada asignatura. El algoritmo genético diseñado utiliza una codificación específica para representar las variables del problema y operadores genéticos adaptados para mantener la diversidad genética en la población, permitiendo así una exploración efectiva del espacio de soluciones.

Agradecimientos

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Índice general

1	Introducción	1
1.1	Motivación	1
1.2	Objetivos	2
1.3	Competencias	2
1.4	Estructura del documento	3
2	Estado del arte	5
2.1	Escuela superior de ingeniería informática de Albacete	5
2.2	Metaheurísticas evolutivas	11
2.3	Algoritmos Genéticos	12
2.4	Algoritmos Genéticos Multiobjetivo	15
2.4.1	NSGA-III	16
3	Desarrollo	19
3.1	Modelización del problema	19
3.1.1	Representación de los individuos	19
3.1.2	Objetivos	20
3.2	Algoritmo genético	24
3.2.1	Función de evaluación	24
3.2.2	Población inicial	25
3.2.3	Selección	26
3.2.4	Cruce	27
3.2.5	Mutación	29
3.2.6	Sustitución de la población	29

3.2.7	Condición de parada	30
3.2.8	Ficheros de salida	31
3.3	NSGA-III	32
3.3.1	Población Inicial	32
3.3.2	Puntos de referencia	33
3.3.3	Clasificación no dominada	33
3.3.4	Asignación a puntos de referencia	34
3.3.5	Selección	35
3.3.6	Cruce y Mutación	36
3.3.7	Reemplazo	37
3.3.8	Condición de parada	37
3.3.9	Ficheros de salida	37
4	Experimentación	39
4.1	Introducción	39
4.2	Entorno de ejecución	40
4.3	Evaluación de métodos del algoritmo genético	40
4.4	Evaluación de hiperparámetros del algoritmo genético	45
4.5	Evaluación del algoritmo NSGA-III	48
5	Desarrollo de la aplicación web	53
5.1	Introducción	53
5.2	Diseño de la aplicación	53
5.3	Flask	54
5.4	Arquitectura del proyecto	54
5.5	Aplicación web	55
5.6	Conclusiones y futuras mejoras	61
6	Conclusiones	63
6.1	Conclusiones	63
6.2	Trabajo futuro	64
A	Anexos	65
A.1	Repositorio del código y scripts	65
	Referencia bibliográfica	67

Índice de figuras

2.1	Módulos que forman el grado de Ingeniería Informática	6
2.2	Asignaturas módulo Formación Básica para la Ingeniería	7
2.3	Asignaturas módulo Formación Común a la rama de ingeniería Informática	8
2.4	Horario 1º cuatrimestre Grupo 1ºA	9
2.5	Horario 1º cuatrimestre Grupo 2ºA	10
2.6	Flujo básico de un algoritmo genético. Fuente: https://www.cs.us.es/~fsancho/Blog/posts/Algoritmos_Geneticos.md.html	14
2.7	Flujo básico de NSGA-III. Fuente: https://www.researchgate.net/figure/Flowchart-of-NSGA-III_fig3_329986923	18
3.1	Representación de un individuo en el problema de asignación de matrículas.	20
5.1	Estructura del proyecto	55
5.2	Página principal de la aplicación web	57
5.3	Pantalla de carga de la aplicación web	58
5.4	Parte superior de la pantalla de resultados del algoritmo clásico	59
5.5	Parte inferior de la pantalla de resultados del algoritmo clásico	59
5.6	Parte superior de la pantalla de resultados NSGA-III	60
5.7	Parte inferior de la pantalla de resultados NSGA-III	61

Índice de tablas

4.1	Resultados primer script de experimentacion	42
4.2	Evolución torneo varios_puntos reemplazo	43
4.3	Resultados segundo script de experimentacion	47
4.4	Primeras 10 combinaciones del tercer script de experimentación	51

1. Introducción

1.1. Motivación

La gestión de matriculas para asignar grupos es uno de los procesos esenciales y más complejos en cualquier universidad ya que actualmente existe una gran cantidad de alumnos y no todos los grupos son iguales, ya que tienen distintos horarios, siendo algunos por la mañana y otros por la tarde, además de que cada alumno tiene preferencias distintas ya sea porque prefiere un horario temprano o porque le complementa mejor con las asignaturas que tiene u otras actividades.

Actualmente los estudiantes analizan muy bien los horarios de las asignaturas antes del proceso de matriculación para calcular cual sería su configuración optima y tener el menor numero de solapes de horarios y menor perdida de tiempo entre clases, aunque no todos consiguen matricularse en sus grupos ideales debido a que los alumnos no realizan la matrícula a la vez, sino que se hace en distintos días y primero la realizan los estudiantes con mayor nota(referencia), esto puede causar frustración en los alumnos que no consiguen matricularse en el grupo que querían debido a que no quedaban plazas pudiendo afectar a su rendimiento académico u organización personal.

Después de realizar las matriculas se tiene que seleccionar grupo de prácticas, en este caso si se realiza al mismo tiempo para todos los alumnos por lo que suele ser complicado obtener el grupo que se desea añadiendo más dificultad a los alumnos en su tarea de conseguir un horario optimo. Teniendo en cuenta que los grupos tienen que tener una cantidad de alumnos limitada y es imposible satisfacer todas las preferencias de los alumnos de forma individual. Por eso nace la idea de desarrollar un sistema con inteligencia artificial que recibiendo las matriculas que han realizado los alumnos, usando procesos de optimización se haga una asignación de grupos de forma que se intente favorecer a los alumnos teniendo el menor número de solapes de horarios en asignaturas de un alumno y reducir el tiempo muerto entre clases para cada alumno, todo esto manteniendo el equilibrio de alumnos por grupo en las asignaturas e intentado mantener cierto parecido con la configuración original.

1.2. Objetivos

1. Analizar el problema de la asignación docente:
 - Identificar y analizar los principales factores y restricciones que afectan a la asignación docente, como las preferencias de los alumnos, las restricciones horarias, las plazas limitadas en los grupos de teoría y prácticas.
2. Diseñar un modelo de optimización multiobjetivo:
 - Desarrollar un modelo matemático que represente el problema de la asignación docente, donde este explicados todos los parámetros a optimizar y sus restricciones.
3. Implementar un algoritmo genético adaptado:
 - Configurar e implementar un algoritmo genético específico para resolver el modelo de optimización, incluyendo la codificación de soluciones, la función de fitness, y los operadores de selección, cruzamiento y mutación.
4. Realizar experimentos y evaluar el rendimiento del algoritmo:
 - Diseñar y llevar a cabo una batería de experimentos para probar el algoritmo genético con distintas configuraciones y en escenarios diferentes comparando el rendimientos y la evolución del tiempo de ejecución.
5. Analizar los resultados y validar el modelo:
 - Evaluar los resultados obtenidos, analizando la eficacia y eficiencia del algoritmo genético en la optimización de la asignación docente usando varias métricas para determinar cual es la mejor configuración que puede tener este algoritmo genético para obtener la mejor solución.
6. Desarrollar una aplicación web para la ejecución de los algoritmos genéticos:
 - Implementar una interfaz gráfica sencilla utilizando que permita a los usuarios cargar los ficheros de entrada, para usar los algoritmos genéticos y poder escoger entre una configuración manual o la óptima determinada durante la fase de experimentación, y visualizar los resultados finales incluyendo la evolución del algoritmo y la ubicación de los ficheros generados.

1.3. Competencias

En este proyecto se han trabajado las siguientes competencias.

- Comunes a la rama de la informática

- **[CO6]** Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.
 - **[CO7]** Conocimiento, diseño y utilización de forma eficiente de los tipos y estructuras de datos más adecuados para la resolución de un problema.
 - **[CO14]** Conocimiento y aplicación de los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real.
 - **[CO15]** Conocimiento y aplicación de los principios fundamentales y técnicas básicas de los sistemas inteligentes y su aplicación práctica.
- Tecnología específica. Computación.
- **[CM3]** Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.
 - **[CM4]** Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación.
 - **[CM5]** Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes o entornos inteligentes.

1.4. Estructura del documento

La estructura de los capítulos de la memoria es la siguiente.

- Introducción
- Presenta el contexto y la motivación del estudio, describe los desafíos de la asignación docente y establece los objetivos generales y específicos del trabajo.
- Estado del arte
- Revisa el contexto del Grado Universitario en Ingeniería Informática, el funcionamiento del sistema de matrículas y la organización de los grupos. Proporciona una visión general de las metaheurísticas evolutivas y una revisión detallada de los algoritmos genéticos y su aplicación en problemas similares.
- Desarrollo

-
- Detalla el diseño del algoritmo genético, incluyendo la representación de soluciones, la función de aptitud y los operadores genéticos utilizados. Describe la implementación del algoritmo en un entorno de desarrollo específico.
 - Experimentación
 - Describe la configuración experimental, los escenarios de prueba, los métodos de evaluación y presenta los resultados obtenidos. Incluye una discusión sobre la eficacia del algoritmo
 - Desarrollo de la aplicación web
 - Explica el diseño, arquitectura e implementación de una interfaz web desarrollada para facilitar la interacción con los algoritmos genéticos implementados. Describe las tecnologías utilizadas, la estructura del proyecto, y cómo los usuarios pueden configurar y ejecutar los algoritmos desde la web.
 - Conclusiones
 - Resume los hallazgos principales del estudio, discute la contribución del trabajo y su impacto en la mejora de la asignación docente. Propone posibles mejoras al algoritmo y sugiere direcciones para futuras investigaciones.

2. Estado del arte

En este capítulo se describe el funcionamiento de las matriculas en la Escuela Superior de Ingeniería Informática de Albacete y el tipo de algoritmos genéticos que se pueden usar para optimizar la asignación de grupos en las matriculas de los estudiantes.

2.1. Escuela superior de ingeniería informática de Albacete

Las titulaciones en la Escuela Superior de Ingeniería Informática de Albacete se inician en el año 1985 con los estudios de Diplomado en Informática. Posteriormente, en el curso académico 1992-93, se implantan los títulos de Ingeniería Técnica en Sistemas e Ingeniería Técnica en Gestión, ambos con una duración de tres años. En el curso 1999-2000 se incorpora la titulación de Ingeniería Informática de cinco años. Estas tres titulaciones permanecen vigentes hasta el curso 2010-11, momento en el que se introduce el actual Grado en Ingeniería Informática, adaptado al Espacio Europeo de Educación Superior [European Commission, 2024], sustituyendo así a las titulaciones anteriores.

Actualmente, además del Grado en Ingeniería Informática, la Escuela ha ampliado su oferta formativa con nuevas titulaciones. Entre ellas se encuentra el doble grado en Ingeniería Informática y Administración y Dirección de Empresas, que combina competencias en el ámbito de la informática y la gestión empresarial. En cuanto a estudios de posgrado, se imparten el Máster Universitario en Ingeniería Informática y el programa de Doctorado en Tecnologías Informáticas Avanzadas, junto con diversos títulos propios como el Máster en Sistemas Informáticos Embarcados y el Máster en Ciberseguridad y Seguridad de la Información. A esta oferta se suman recientemente nuevos másteres oficiales como el Máster en Big Data y Computación en la Nube y el Máster en Inteligencia Artificial, los cuales refuerzan la formación en áreas estratégicas con gran proyección profesional.

El presente trabajo se desarrolla en el contexto del actual Grado en Ingeniería Informática. Por ello, se describe la organización del plan de estudios, los tipos de asignaturas que lo componen, así como el funcionamiento del sistema de matrícula y su relación con el problema planteado. Toda la información relativa a este grado se encuentra disponible en la página web oficial de la Escuela

[Escuela Superior de Ingeniería Informática de Albacete, 2024]

Dentro de dicha página, la información relevante se encuentra en la sección dedicada al Grado en Ingeniería Informática, en particular en los apartados del plan de estudios y los horarios. El primero ofrece un desglose detallado de los módulos que conforman el grado, así como los créditos asignados a cada uno de ellos.

Módulo I - Formación Básica (60 ECTS)

El Módulo I es el módulo de Formación Básica comprende 60 ECTS. Está integrado por 9 asignaturas de 6 ECTS asociadas a la rama de Arquitectura e Ingeniería y 1 asignatura de 6 ECTS asociada a la rama de Ciencias Sociales y Jurídicas, que se ofrecen dentro de la primera mitad del Plan de Estudios (concretamente 8 asignaturas en el primer curso, y 2 asignaturas en el segundo curso).

Módulo II - Común a la Rama de Informática (96 ECTS)

El Módulo II es el módulo común a la Rama Informática y está compuesto por 96 créditos ECTS.

Módulo III - Tecnologías Específicas (48 ECTS)

El Módulo III es el módulo de Tecnología Específica, tiene 48 ECTS y se realizan cuatro implementaciones del mismo (para cubrir las cuatro intensificaciones ofertadas y por tanto sólo se puede cursar uno). En el suplemento europeo al título se anotará la intensificación que ha realizado el estudiante en función de la implementación del módulo de tecnología específica realizado. Una descripción de los conocimientos y competencias que se adquieren en cada una de las 4 tecnologías específicas ofertadas, así como sus posibles salidas profesionales, puede consultarse [aquí](#).

Módulo IV - Trabajo Fin de Grado (12 ECTS)

El Módulo IV es el correspondiente al Trabajo Fin de Grado y comprende los 12 ECTS dispuestos para el trabajo fin de grado.

Módulo V - Optativas

El Módulo V comprende una oferta de 72 ECTS de optativas de los cuales se deben cursar 24 ECTS, para que el alumno seleccione y configure su currículum para reforzar algunas de las competencias adquiridas. Las optativas se ofrecen en cuarto curso.

Figura 2.1: Módulos que forman el grado de Ingeniería Informática

El grado se organiza en los siguientes módulos:

- **Formación Básica para la Ingeniería y Formación Común a la Rama de Ingeniería Informática:** Son obligatorios para todos los alumnos.
- **Tecnologías Específicas:** Dividida en cuatro especialidades, de las que el estudiante debe cursar todas las asignaturas de una para completar el grado.
- **Optatividad:** Incluye asignaturas de libre elección.

■ Trabajo Fin de Grado

De estos módulos, los dos primeros tienen una relevancia especial, ya que incluyen asignaturas con más de un grupo, lo cual resulta fundamental para este análisis. Las asignaturas con un solo grupo no pueden ser optimizadas en términos de asignación de matrícula, aunque sí influyen en el proceso general.

Módulo I - Formación Básica (60 ECTS)		
ÁLGEBRA Y MATEMÁTICA DISCRETA Créditos: 6.0 Idioma: Español Tipo: Formación Básica Módulo: I	CÁLCULO Y MÉTODOS NUMÉRICOS Créditos: 6.0 Idioma: Español - Inglés Tipo: Formación Básica Módulo: I	ESTADÍSTICA Créditos: 6.0 Idioma: Español - Inglés Tipo: Formación Básica Módulo: I
ESTRUCTURA DE COMPUTADORES Créditos: 6.0 Idioma: Español - Inglés Tipo: Formación Básica Módulo: I	FUNDAMENTOS DE GESTIÓN EMPRESARIAL Créditos: 6.0 Idioma: Español - Inglés Tipo: Formación Básica Módulo: I	FUNDAMENTOS DE PROGRAMACIÓN I Créditos: 6.0 Idioma: Español - Inglés Tipo: Formación Básica Módulo: I
FUNDAMENTOS DE PROGRAMACIÓN II Créditos: 6.0 Idioma: Español - Inglés Tipo: Formación Básica Módulo: I	FUNDAMENTOS FÍSICOS DE LA INFORMÁTICA Créditos: 6.0 Idioma: Español - Inglés Tipo: Formación Básica Módulo: I	LÓGICA Créditos: 6.0 Idioma: Español Tipo: Formación Básica Módulo: I
TECNOLOGÍA DE COMPUTADORES Créditos: 6.0 Idioma: Español Tipo: Formación Básica Módulo: I		

Figura 2.2: Asignaturas módulo Formación Básica para la Ingeniería

El Módulo II es el módulo común a la Rama Informática y está compuesto por 96 créditos ECTS.

ARQUITECTURA DE COMPUTADORES Créditos: 6.0 Idioma: Español - Inglés Tipo: Obligatoria Módulo: II	ASPECTOS PROFESIONALES DE LA INFORMÁTICA Créditos: 6.0 Idioma: Español Tipo: Obligatoria Módulo: II	BASES DE DATOS Créditos: 6.0 Idioma: Español Tipo: Obligatoria Módulo: II
ESTRUCTURA DE DATOS Créditos: 6.0 Idioma: Español - Inglés Tipo: Obligatoria Módulo: II	INGENIERÍA DEL SOFTWARE I Créditos: 6.0 Idioma: Español - Inglés Tipo: Obligatoria Módulo: II	INGENIERÍA DEL SOFTWARE II Créditos: 6.0 Idioma: Español - Inglés Tipo: Obligatoria Módulo: II
INTERACCIÓN PERSONA-ORDENADOR I Créditos: 6.0 Idioma: Español - Inglés Tipo: Obligatoria Módulo: II	METODOLOGÍA DE LA PROGRAMACIÓN Créditos: 6.0 Idioma: Español - Inglés Tipo: Obligatoria Módulo: II	ORGANIZACIÓN DE COMPUTADORES Créditos: 6.0 Idioma: Español - Inglés Tipo: Obligatoria Módulo: II
PROGRAMACIÓN CONCURRENTES Y TIEMPO REAL Créditos: 6.0 Idioma: Español - Inglés Tipo: Obligatoria Módulo: II	REDES DE COMPUTADORES I Créditos: 6.0 Idioma: Español - Inglés Tipo: Obligatoria Módulo: II	REDES DE COMPUTADORES II Créditos: 6.0 Idioma: Español Tipo: Obligatoria Módulo: II
SISTEMAS DE INFORMACIÓN Créditos: 6.0 Idioma: Español - Inglés Tipo: Obligatoria Módulo: II	SISTEMAS DISTRIBUIDOS Créditos: 6.0 Idioma: Español Tipo: Obligatoria Módulo: II	SISTEMAS INTELIGENTES Créditos: 6.0 Idioma: Español - Inglés Tipo: Obligatoria Módulo: II
SISTEMAS OPERATIVOS I Créditos: 6.0 Idioma: Español - Inglés Tipo: Obligatoria Módulo: II		

Figura 2.3: Asignaturas módulo Formación Común a la rama de Ingeniería Informática

Las Figuras incluidas muestran las asignaturas que componen los dos bloques obligatorios del Grado en Ingeniería Informática. Estos bloques contienen todas las materias correspondientes al primer y segundo curso, así como las asignaturas del primer cuatrimestre del tercer curso y la asignatura “Aspectos Profesionales de la Informática”.

Por otro lado, la sección de horarios disponible en la web oficial de la Escuela [Escuela Superior de Ingeniería Informática de Albacete, 2024]

proporciona información relevante para el desarrollo de este trabajo. Los horarios están estructurados por cuatrimestres e indican las franjas horarias asignadas a cada asignatura y grupo, lo cual resulta fundamental para la detección de posibles solapamientos entre asignaturas, aspecto que se verá en secciones posteriores.

El sistema de matriculación asigna a cada estudiante un día y una hora específicos para realizar el proceso a través de la plataforma de Secretaría Virtual de la universidad [?]. Durante dicho proceso, el alumnado debe seleccionar las asignaturas en las que desea matricularse y, en caso de que una asignatura cuente con varios grupos, se requiere elegir un grupo de teoría. La asignación a los subgrupos de prácticas se gestiona posteriormente a través de la página correspondiente de cada asignatura en el campus virtual [?].

En situaciones en las que el estudiante se matricula únicamente en asignaturas correspondientes a un mismo curso, la selección de grupos suele realizarse sin mayores complicaciones. Sin embargo, cuando se combinan asignaturas de distintos cursos en el mismo cuatrimestre, el proceso de selección se vuelve más complejo. Un diseño inadecuado de la matrícula puede dar lugar a conflictos de horarios, impidiendo la asistencia regular a todas las clases.

Por ejemplo, un estudiante matriculado simultáneamente en la asignatura “Fundamentos de Programación I” (perteneciente al primer curso) y “Organización de Computadores” (del segundo curso) debe tener en cuenta tanto los grupos de teoría como los subgrupos de prácticas para evitar solapamientos. En este caso concreto, no sería viable seleccionar el mismo grupo (por ejemplo, el grupo A) para ambas asignaturas si se desea asegurar la asistencia completa a todas las clases.

1º GRUPO A (10) - 1 Semestre 2023-24 Primer Cuatrimestre					
	Lunes	Martes	Miércoles	Jueves	Viernes
8:15 9:45	Fundamentos de Gestión Empresarial AULA 1.11	Fundamentos de Programación I AULA 1.11	Fundamentos Físicos de la Informática AULA 1.11	Tecnología de Computadores AULA 1.11	Cálculo y Métodos Numéricos AULA 1.11
9:45 11:15	Cálculo y Métodos Numéricos AULA 1.11	Fundamentos de Gestión Empresarial AULA 1.11	Fundamentos de Programación I AULA 1.11	Fundamentos Físicos de la Informática AULA 1.11	Tecnología de Computadores AULA 1.11
11:35 13:05	Fundamentos Físicos de la Informática Lab. Física 2	Tecnología de Computadores Elect.Digit.1	Cálculo y Métodos Numéricos SOFTW. 4	Fundamentos de Gestión Empresarial Aula Multiusos	Fundamentos de Programación I SOFTW. 7
16:00 17:30	Fundamentos Físicos de la Informática Lab. Física 2	Tecnología de Computadores Elect.Digit.1	Cálculo y Métodos Numéricos SOFTW. 4	Fundamentos de Gestión Empresarial Aula Multiusos	Fundamentos de Programación I SOFTW. 7

Figura 2.4: Horario 1º cuatrimestre Grupo 1ºA

2º GRUPO A (10) - 1 Semestre 2023-24 Primer Cuatrimestre					
	Lunes	Martes	Miércoles	Jueves	Viernes
9:45 11:15	Estructura de Datos SOFTW. 4	Lógica Aula Multiusos	Organización de Computadores SOFTW. 7	Ingeniería del Software I SOFTW. 7	Sistemas Operativos I SOFTW. 4
11:35 13:05	Ingeniería del Software I AULA 1.11	Sistemas Operativos I AULA 1.11	Estructura de Datos AULA 1.11	Lógica AULA 1.11	Organización de Computadores AULA 1.11
13:05 14:35	Organización de Computadores AULA 1.11	Ingeniería del Software I AULA 1.11	Sistemas Operativos I AULA 1.11	Estructura de Datos AULA 1.11	Lógica AULA 1.11
17:30 19:00	Estructura de Datos SOFTW. 4	Lógica Aula Multiusos	Organización de Computadores AULA 1.11	Ingeniería del Software I SOFTW. 7	Sistemas Operativos I SOFTW. 4

Figura 2.5: Horario 1º cuatrimestre Grupo 2ºA

Una vez completado el proceso de matrícula por parte de todos los estudiantes, considerando sus preferencias y las plazas disponibles en cada grupo, se obtiene una configuración inicial que será utilizada como punto de partida para el algoritmo genético propuesto en este trabajo.

A partir de la información obtenida de la web oficial de la escuela, se han generado dos ficheros clave que servirán como base para el desarrollo del algoritmo genético. El primero es *asignaturas.xlsx*, que tienen un listado completo de asignaturas del grado y su clasificación, incluyendo:

- **COD.ASIG:** Identificador único de la asignatura.
- **NOMBRE ASIGNATURA:** Nombre completo de la asignatura.
- **TECNOLOGÍA:** Indica que tipo de asignatura es, obligatoria, optativa o si pertenece a alguna intensificación.
- **CUATRIMESTRE:** Cuatrimestre en el que se imparte la asignatura.

El segundo fichero, *horarios.xlsx*, tiene todos los horarios de los grupos de teoría y práctica de las asignaturas, extraídos de la sección de horarios. Este fichero incluye la siguiente información sobre los horarios de cada asignatura:

- **CUATRIMESTRE:** Cuatrimestre en el que se imparte la asignatura.
- **CURSO:** Curso en el que se imparte la asignatura.
- **GRUPO:** Grupo al que pertenece este horario.
- **ID GRUPO:** Identificador numérico del curso.
- **DÍA:** Día en el que se imparte la asignatura.

- **HORARIO:** Horario en el que se imparte la asignatura.
- **BLOQUE:** Indica al bloque al que pertenece la asignatura.
- **ASIGNATURA:** Nombre completo de la asignatura.
- **CODIGO:** Identificador único de la asignatura.
- **TEORÍA/PRÁCTICA:** Indica si el horario es de un grupo de teoría o practicas.

Por último esta el fichero *matriculas.xlsx*, que tiene los registros anonimizados de las matrículas de los estudiantes del curso 2023-2024. La información que hay sobre cada matrícula es:

- **AÑO:** Año lectivo de la matrícula.
- **PLAN:** Plan docente al que pertenece la matrícula.
- **CODIGO:** Identificador único de la asignatura.
- **ASIGNATURA:** Nombre completo de la asignatura.
- **DNI:** Dni anonimizado del estudiante al que pertenece la matrícula.
- **ALUMNO:** Nombre anonimizado del estudiante al que pertenece la matrícula.
- **GRUPO:** Grupo de teoría al que pertenece la matrícula.
- **GP:** Subgrupo de prácticas al que pertenece la matrícula.

2.2. Metaheurísticas evolutivas

En el ámbito de la optimización, las metaheurísticas son consideradas herramientas fundamentales para resolver problemas complejos, especialmente aquellos que no pueden abordarse mediante métodos exactos debido a su alta dimensionalidad, no linealidad o la falta de modelos matemáticos explícitos. Una metaheurística es un enfoque general de optimización que se basa en heurísticas flexibles y adaptativas, diseñadas para explorar eficientemente grandes espacios de búsqueda en busca de soluciones cercanas al óptimo global.

El término "metaheurística" proviene del griego "meta", que significa "más allá", y "heurística", que se refiere a un método de descubrimiento o aprendizaje. Esto destaca su objetivo de ir más allá de las técnicas heurísticas convencionales para encontrar soluciones de calidad en un marco más amplio y general.

Las principales características que poseen las metaheurísticas son:

- **Flexibilidad:** Son aplicables a una amplia gama de problemas, desde la planificación logística hasta la inteligencia artificial.
- **Efectividad:** Debe ser capaz de encontrar soluciones de alta calidad, cercanas al óptimo, incluso en problemas complejos y de gran escala.
- **Eficiencia:** Es crucial que haga un uso equilibrado de los recursos disponibles, minimizando el tiempo de ejecución y el consumo de memoria.

-
- **Generalidad:** La metaheurística debe ser versátil y funcionar bien en distintos tipos de problemas, manteniendo un rendimiento consistente.
 - **Robustez:** Su desempeño debe mantenerse estable incluso cuando el problema o el entorno presentan pequeñas variaciones.
 - **Autonomía:** La metaheurística debe requerir la menor parametrización posible, o ser capaz de ajustar sus parámetros automáticamente para facilitar su uso.

Las metaheurísticas evolutivas son una categoría dentro de las metaheurísticas, se caracterizan por su inspiración en procesos biológicos y evolutivos. Estas técnicas se fundamentan en principios de selección natural, reproducción, mutación y supervivencia del más apto, adaptando estos conceptos para tratar problemas más complejos de optimización. Imitan la evolución natural como un modelo de búsqueda y optimización. En lugar de centrarse en un único candidato a solución, trabajan con una población de posibles soluciones, que evoluciona a través de iteraciones o generaciones. Cada generación se construye aplicando operadores inspirados en la biología, como la selección, el cruce y la mutación. Estos mecanismos permiten la exploración de nuevos puntos en el espacio de búsqueda y el refinamiento progresivo de las soluciones.

Las metaheurísticas evolutivas poseen todas las propiedades generales de las Metaheurísticas, pero también presentan características propias que aumentan su aplicabilidad en problemas específicos:

- **Exploración y explotación equilibradas:** Gracias al uso de operadores de cruce y mutación, las metaheurísticas evolutivas logran un balance entre explorar nuevas áreas del espacio de búsqueda (exploración) y refinar las soluciones existentes (explotación). Esto ayuda a evitar óptimos locales y mejorar la calidad de las soluciones.
- **Adaptabilidad:** Se pueden adaptar a múltiples dominios, ya que los operadores genéticos pueden ser modificados o personalizados según las características del problema a resolver.
- **Soluciones poblacionales:** El uso de una población de soluciones permite trabajar en paralelo con múltiples candidatos, esto incrementa la diversidad y reduce una convergencia prematura en óptimos locales.
- **Robustez frente a incertidumbre:** Las metaheurísticas evolutivas son menos susceptibles a variaciones en las condiciones del problema debido a su enfoque probabilístico, lo que las hace robustas en escenarios inciertos o dinámicos.

2.3. Algoritmos Genéticos

Los algoritmos genéticos (AG) son una implementación concreta y la más estudiada dentro de las metaheurísticas evolutivas. Estas técnicas están basadas en los procesos de selección natural descritos por Charles Darwin, utilizando principios biológicos como la reproducción, mutación y selección para resolver problemas de optimización. Fueron introducidos

por John Holland en la década de 1970 [Holland, 1975], los AG han evolucionado hasta convertirse en una herramienta fundamental para abordar problemas complejos en diversos campos, desde la ingeniería hasta la biología computacional.

Un algoritmo genético opera sobre una población de soluciones potenciales, representadas como *individuos*. A través de iteraciones o *generaciones*, estas soluciones se someten a transformaciones utilizando operadores inspirados en la genética, como el cruce y la mutación. Cada individuo se evalúa utilizando una función de aptitud (*fitness*), que determina su calidad en relación con el problema a resolver.

La representación de los individuos es un aspecto crucial en los AG, ya que define cómo se estructuran las soluciones dentro del algoritmo. Las representaciones más comunes son las siguientes:

- **Codificación binaria:** Utiliza cadenas de bits (0 y 1) para representar las soluciones. Es especialmente adecuada para problemas combinatorios [Goldberg, 1989].
- **Codificación real:** Utiliza valores numéricos reales, lo que resulta más eficiente en problemas de optimización continua.
- **Codificación por permutación:** Diseñada para problemas en los que el orden de los elementos es importante, como el problema del viajante de comercio (*TSP*) [Deb, 2001].

El flujo básico de un algoritmo genético se estructura en los siguientes pasos [Goldberg, 1989]:

1. **Inicialización:** Se genera una población inicial de individuos de forma aleatoria o utilizando heurísticas específicas. Esta población representa un conjunto diverso de posibles soluciones al problema.
2. **Evaluación:** Cada individuo es evaluado mediante la función fitness, que proporciona una medida de su calidad. La función de aptitud es clave para guiar la evolución hacia mejores soluciones.
3. **Selección:** Los individuos más aptos tienen una mayor probabilidad de ser seleccionados para el cruce. Existen distintos métodos.
4. **Cruce:** Se combinan pares de individuos seleccionados para producir descendencia. Este operador permite explorar nuevas regiones del espacio de búsqueda al intercambiar información genética entre los individuos.
5. **Mutación:** Se introducen pequeñas modificaciones en los individuos para mantener la diversidad genética y evitar la convergencia prematura.
6. **Reemplazo:** La nueva población resultante reemplaza total o parcialmente a la población anterior, dependiendo de la estrategia de reemplazo utilizada.
7. **Condición de parada:** El algoritmo genético continúa hasta que se cumple una condición, como alcanzar un número máximo de generaciones, encontrar una solución con un valor de fitness suficiente o ver que la población no ha cambiado en varias generaciones (convergencia).

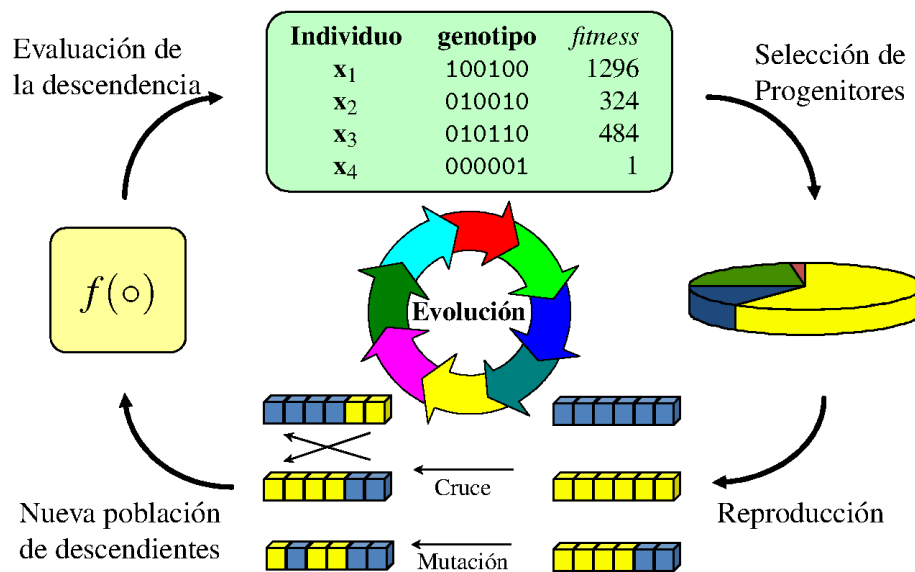


Figura 2.6: Flujo básico de un algoritmo genético. Fuente: https://www.cs.us.es/~fsancho/Blog/posts/Algoritmos_Geneticos.md.html.

Aunque los algoritmos genéticos son herramientas poderosas y muy utilizadas, tienen algunas desventajas a considerar. La primera es que estos algoritmos pueden requerir un número elevado de generaciones para converger a soluciones óptimas, sobre todo en problemas de alta dimensionalidad o cuando el espacio de búsqueda contiene numerosos óptimos locales. Este comportamiento puede traducirse en un tiempo de ejecución considerablemente alto en escenarios complejos.

También son altamente sensibles a la configuración de sus parámetros. El tamaño de la población, las probabilidades de cruce y mutación, y la condición de parada son factores críticos que afectan directamente en el rendimiento del algoritmo. Una parametrización inadecuada puede llevar a una exploración insuficiente del espacio de búsqueda o a un aumento innecesario del coste computacional sin mejoras significativas en las soluciones obtenidas.

Otra limitación importante recae en el coste computacional asociado a la evaluación de la función objetivo. En cada generación la calidad de los individuos debe ser calculada, lo que puede ser muy costoso en problemas donde la evaluación conlleva cálculos complejos.

Por último, los algoritmos genéticos presentan desafíos específicos en el contexto de problemas multiobjetivo. En estos casos, se busca optimizar simultáneamente múltiples objetivos conflictivos, lo que introduce una mayor complejidad en la búsqueda de soluciones. Sin un diseño adecuado, puede llevar a soluciones que favorecen únicamente algunos objetivos, perdiendo la diversidad. Estos desafíos hacen necesario el desarrollo de estrategias específicas para abordar eficazmente problemas multiobjetivo.

2.4. Algoritmos Genéticos Multiobjetivo

En la mayoría de problemas de optimización del mundo real, es común enfrentarse a múltiples objetivos conflictivos que tienen que ser optimizados simultáneamente. Los algoritmos genéticos multiobjetivo (MOGA, por sus siglas en inglés) se han desarrollado como una extensión de los algoritmos genéticos convencionales para tratar este tipo de problemas. En los algoritmos genéticos clásicos se tiende a combinar los objetivos en una única función escalar, en cambio, en los MOGA se busca un conjunto de soluciones conocidas como frente de pareto, que representa el equilibrio entre los distintos objetivos, mostrando soluciones en las que mejorar un objetivo implica empeorar otro, esto proporciona una variedad de opciones igual de buenas para elegir según las necesidades del problema.

La optimización multiobjetivo esta basada en el concepto Pareto-optimidad introducido por Vilfredo Pareto en el contexto de la economía [?]. Por ejemplo, en un problema donde hay que minimizar 5 objetivos, una solución $x_1 \in S$ domina a otra solución $x_2 \in S$ si se cumplen las siguientes condiciones:

- $f_i(x_1) \leq f_i(x_2)$ para todos los objetivos $i = 1, \dots, 5$.
- $f_i(x_1) < f_i(x_2)$ para al menos un objetivo i

El conjunto de todas las soluciones no dominadas forman el frente de pareto. El objetivo principal es aproximar este frente de pareto de forma eficiente, generando soluciones que tengan en cuenta dos características: una alta convergencia hacia el frente verdadero y una buena diversidad dentro de este. Esto permite que además de tener soluciones de alta calidad en términos de optimización, tenemos soluciones que permiten una gran variedad a la hora de tomar una decisión para elegir una solución final.

Para cumplir estos objetivos, los MOGA usan técnicas que extienden las ideas clásicas de los algoritmos genéticos. En la evaluación de aptitud de una solución no se basa en un fitness calculado mediante una función escalar, sino que se tiene en cuenta el nivel de no-dominancia de cada solución respecto a las demás. Esto se realiza clasificando las soluciones en frentes jerárquicos, donde las soluciones no dominadas forman el primer nivel, el siguiente frente lo formarán las soluciones dominadas solo por el frente anterior y así sucesivamente. Con esto se asegura que las mejores soluciones tengan una mayor prioridad.

Para mantener la diversidad entre las soluciones, se emplean estrategias como el calculo de distancias entre las soluciones del frente. Este tipo de estrategias se usan para evitar que las soluciones converjan hacia una única región en el frente de pareto. Los MOGA son especialmente relevantes en aplicaciones prácticas gracias a su capacidad para manejar objetivos conflictivos de manera simultánea y eficiente. Al contrario que los algoritmos genéticos clásicos que ofrecen una única solución optima, los MOGA proporcionan un conjunto diverso de soluciones que pueden adaptarse a distintas necesidades y restricciones.

2.4.1. NSGA-III

NSGA-III es un algoritmo evolutivo diseñado específicamente para abordar problemas de optimización multiobjetivo con un alto número de objetivos. En este trabajo se ha elegido este algoritmo de optimización multiobjetivo para usarlo en el problema planteado por su capacidad de manejar múltiples objetivos de manera eficiente. Su enfoque basado en puntos de referencia garantiza una distribución uniforme de las soluciones a lo largo del frente de Pareto, esto es clave para obtener una visión clara y equilibrada de las posibles compensaciones entre los objetivos en conflicto.

Este algoritmo fue propuesto por Deb y Jain en 2014 [Deb and Jain, 2014], se introduce un marco avanzado que amplía los principios de clasificación no dominada del NSG-II, algoritmo que fue desarrollado previamente por el propio Deb junto a Pratap, Agarwal y Meyarivan en el año 2002 [?]. La evolución se centra en el uso de un conjunto predefinido de puntos de referencia en el espacio de objetivos, que guía el proceso evolutivo hacia una distribución uniforme de soluciones en el frente de Pareto. Estos puntos ayudan al algoritmo a mantener una buena diversidad incluso en problemas con más de tres objetivos, donde otros algoritmos tienden a fallar.

El proceso de selección en NSGA-III no solo considera el nivel de no-dominancia de las soluciones, sino también su proximidad a los puntos de referencia, lo que garantiza una exploración equilibrada del espacio de objetivos.

El flujo básico del algoritmo NSGA-III es el siguiente [Deb and Jain, 2014]:

1. **Inicialización:** Se genera una población inicial aleatoria de soluciones, a partir de esa generación usando cruce y mutación se genera otra generación, la unión de esas dos servirá como población inicial. Se define un conjunto de puntos de referencia equidistantes en el espacio objetivo. Estos puntos sirven como guía para asegurar una distribución uniforme de las soluciones en el frente de Pareto.
2. **Evaluación:** Cada individuo es evaluado en todos los objetivos del problema. No existe una función objetivo para valorar la solución.
3. **Clasificación no dominada:** Se clasifica la población en frentes según el principio de Pareto. El primer frente contiene las soluciones no dominadas por ninguna otra, el segundo incluye aquellas dominadas solo por las del primer frente, y así sucesivamente.
4. **Asignación a puntos de referencia:** Las soluciones de los frentes se asignan a uno de los puntos de referencia. Esta asignación permite medir qué tan representativa es cada solución respecto a la diversidad deseada en el frente.
5. **Selección:** Cuando el último frente no cabe completamente en la nueva población, se seleccionan las soluciones más cercanas (o menos representadas) en relación con los puntos de referencia. Este proceso asegura un equilibrio entre convergencia y diversidad.
6. **Cruce y mutación:** Se generan nuevos individuos mediante los operadores genéticos de cruce y mutación.
7. **Reemplazo:** Se crea una nueva población combinando padres e hijos. A continuación,

se repite la clasificación no dominada y la selección basada en los puntos de referencia.

8. **Condición de parada:** El algoritmo continúa iterando hasta alcanzar una condición de parada, como un número máximo de generaciones o una estabilidad en el frente de soluciones obtenidas.

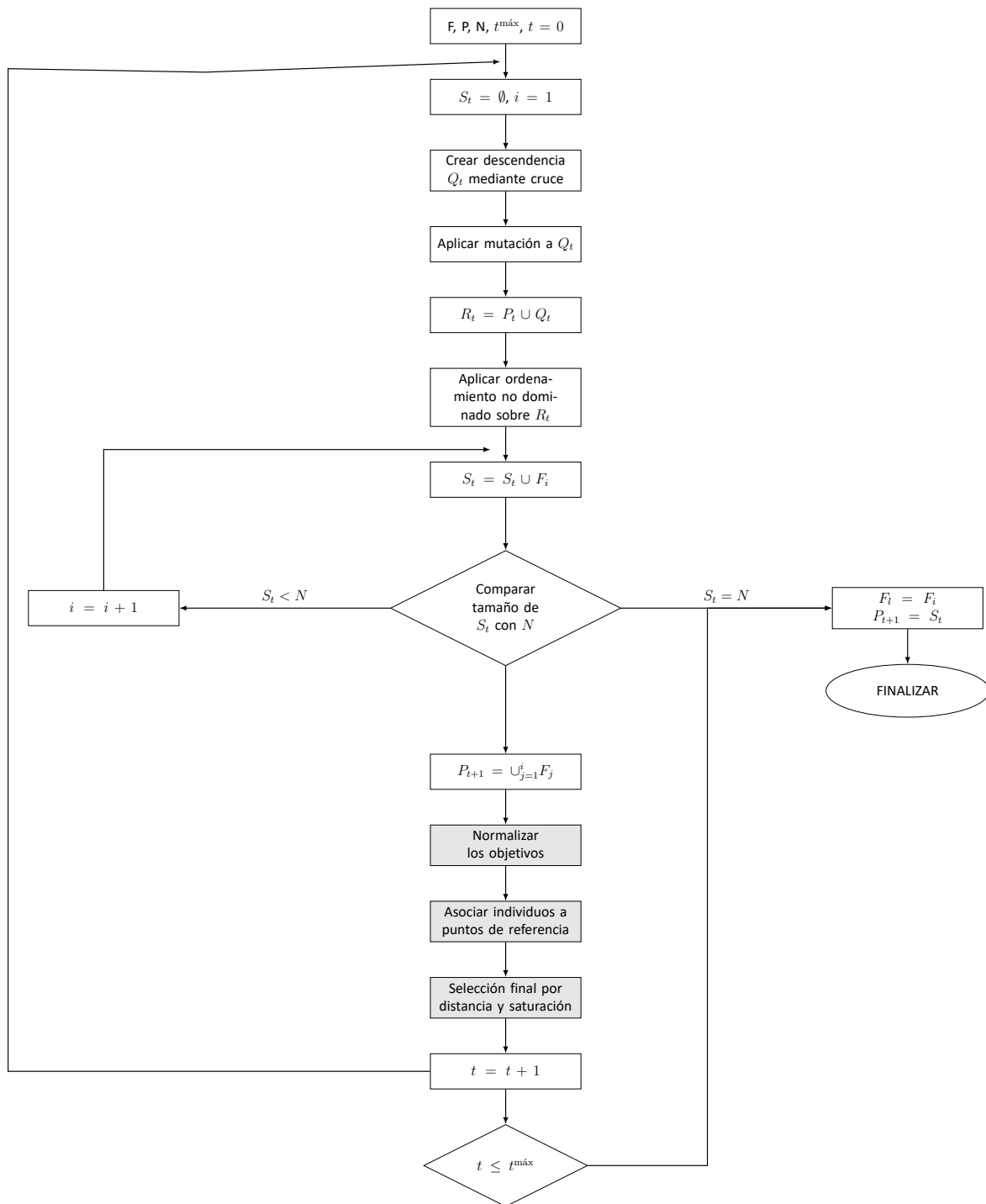


Figura 2.7: Flujo básico de NSGA-III. Fuente: https://www.researchgate.net/figure/Flowchart-of-NSGA-III_fig3_329986923

3. Desarrollo

En este capítulo se explica el proceso de desarrollo del algoritmo genético, los ficheros iniciales, la modelización matemática del problema, ficheros de salida y las distintas implementaciones que se han hecho.

3.1. Modelización del problema

Una vez planteado el problema, es necesario establecer una modelización adecuada que permita su resolución mediante un algoritmo genético. Para ello, deben definirse de forma precisa los componentes clave del modelo evolutivo como la representación de los individuos que conforman cada población, los objetivos que se desean optimizar, también los métodos para su cálculo e integración dentro de una función de evaluación.

Además, es imprescindible especificar el funcionamiento de los principales operadores genéticos, como la selección, el cruce y la mutación, junto con la condición de parada que determinará el fin del proceso evolutivo.

3.1.1. Representación de los individuos

En este problema cada individuo representara una configuración para todas las matriculas de cada alumno, cada individuo tendrá una lista de alumnos, cada alumno tendrá dos listas, una de matriculas fijas, las cuales son de grupo único y no pueden ser modificadas y otra de matriculas variables, que son de asignaturas que poseen más de un grupo y pueden ser modificadas en cada generación, en cada matricula estará reflejado el grupo de teoría, subgrupo de práctica y su horario. En esta representación los genes que podrán ser modificados serán el grupo de teoría y subgrupo de práctica de las matrículas variables, el grupo de teoría podrá tomar los valores de 10, 11 y 12 en asignaturas del 1º curso, 10 u 11 en asignaturas de los demás cursos y el subgrupo de prácticas podrá ser 1 o 2.

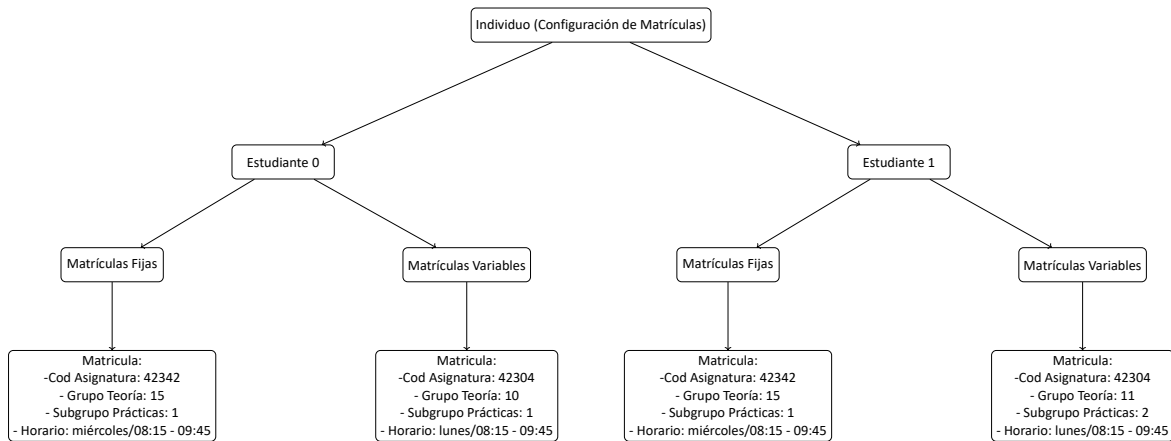


Figura 3.1: Representación de un individuo en el problema de asignación de matrículas.

3.1.2. Objetivos

Una vez definida la representación de los individuos, es necesario establecer los objetivos que guiarán el proceso de optimización. Para este problema se consideran cinco objetivos, seleccionados por su relevancia. A continuación, se describe cada uno de ellos junto con el criterio utilizado para su evaluación.

1. Número de solapes (minimizar):

Representan la suma de solapamientos en los horarios de distintas asignaturas para cada alumno, constituye uno de los objetivos a minimizar. Para calcular este valor, se analiza individualmente cada alumno, identificando los solapamientos existentes entre las asignaturas matriculadas que pertenecen al mismo cuatrimestre. Esta distinción es fundamental, ya que los solapamientos entre asignaturas de distintos cuatrimestres no interfieren en la asistencia y, por tanto, no se consideran relevantes para este criterio de evaluación.

$$\text{Solapes}_{\text{cuatrimestre}} = \sum_{\substack{i=1 \\ c_i \in Q}}^{n-1} \sum_{\substack{j=i+1 \\ c_j \in Q}}^n \text{intersección}(C_i, C_j)$$

Donde:

- C_i y C_j son asignaturas específicas en las que está matriculado el estudiante.
- Q representa el conjunto de asignaturas de ese cuatrimestre en las que está matriculado el estudiante.
- n es el número total de asignaturas en las que está matriculado el estudiante en ese cuatrimestre.
- La función $\text{intersección}(C_i, C_j)$ devuelve 1 si las asignaturas C_i y C_j se solapan en el tiempo y 0 en caso contrario.

Una vez que se tiene la definición para calcular los solapes totales por cuatrimestre de cada estudiante, se realiza la suma de ambos para obtener el total de solapes de un estudiante:

$$\text{Solapes}_{\text{estudiante}} = \text{Solapes}_{\text{cuatrimestre}_1} + \text{Solapes}_{\text{cuatrimestre}_2}$$

El número total de solapes en un individuo se obtiene sumando los solapes de cada uno de los estudiantes:

$$\text{Solapes Totales} = \sum_{i=1}^n \text{Solapes}_{\text{estudiante}_i}$$

Donde:

- n es el número de estudiantes presentes en la solución representada por el individuo.

2. Equilibrio de alumnos por clase (maximizar):

Este objetivo refleja el grado de equidad en la distribución de estudiantes entre los diferentes grupos de teoría y subgrupos de prácticas de las asignaturas. Se considera un criterio especialmente relevante, ya que una mala distribución puede afectar directamente a la viabilidad de la docencia debido a la capacidad limitada de las aulas. Además, una distribución desequilibrada puede dificultar la atención personalizada al alumnado y generar problemas logísticos.

Mientras que otros objetivos están orientados a mejorar la experiencia del estudiante, como evitar solapamientos o facilitar horarios cómodos, este objetivo se vincula con la factibilidad real de implementar la solución. Por tanto, su cumplimiento se considera prioritario.

Para calcular esta métrica, se evaluará el equilibrio alcanzado en cada asignatura obligatoria. A continuación, se describen los pasos seguidos para su cálculo.

2.1 Número ideal de estudiantes por grupo de teoría:

Para la asignatura i con N_i estudiantes matriculados y G_i grupos, el número ideal de estudiantes por grupo es $N_{ideal_i} = N_i/G_i$

2.2 Proporción de equilibrio por grupo principal:

Para cada grupo principal g de la asignatura i , el equilibrio se calcula restando la cantidad ideal de estudiantes N_{ideal_i} al número real de estudiantes n_{g_i} en ese grupo. Esta diferencia se normaliza dividiéndola por N_{ideal_i} y el resultado se toma en valor absoluto para evitar resultados negativos. Finalmente, se resta esta proporción de 1 para obtener una medida del equilibrio, donde valores cercanos a 1 significa que el número de estudiantes en ese grupo está cerca del valor ideal y valores menores que 1 indica mayor desequilibrio:

$$E_{g_i} = 1 - \left| \frac{n_{g_i} - N_{ideal_i}}{N_{ideal_i}} \right|$$

2.3 Número ideal de estudiantes por subgrupo de prácticas:

Será el número ideal de alumnos en el grupo principal N_{ideal_i} entre 2

$$N_{ideal_{sg}} = \frac{N_{ideal_i}}{2}$$

2.4 Proporción de equilibrio por subgrupo de prácticas:

Para cada subgrupo de prácticas s_g de la asignatura i , el equilibrio se calcula restando la cantidad ideal de estudiantes $N_{ideal_{sg}}$ al número real de estudiantes n_{sg_i} en ese grupo. Esta diferencia se normaliza dividiéndola por $N_{ideal_{sg}}$ y el resultado se toma en valor absoluto para evitar resultados negativos. Finalmente, se resta esta proporción de 1 para obtener una medida del equilibrio, donde valores cercanos a 1 significa que el número de estudiantes en ese subgrupo está cerca del valor ideal y valores menores que existe mayor desequilibrio:

$$E_{sg_i} = 1 - \left| \frac{n_{sg_i} - N_{ideal_{sg}}}{N_{ideal_{sg}}} \right|$$

2.5 Promedio de equilibrio en la asignatura:

Para calcular el equilibrio total de cada asignatura i se calcula con el promedio de las proporciones de todos los grupos de teoría y subgrupos de prácticas.

$$E_{asignatura_i} = \frac{\sum_{g=1}^G D_{grupo\ principal_{g_i}} + \sum_{s=1}^{S_g} D_{subgrupo_{sg}}}{G + S_g}$$

Donde:

- G es el número de grupos principales de teoría de la asignatura.
- S_g es el número de subgrupos de practicas de la asignatura.

Una vez que tenemos el equilibrio de cada asignatura obligatoria podemos calcular el equilibrio total del individuo de la siguiente manera.

$$E_{total} = \frac{1}{A} \sum_{a=1}^A E_{asignatura_a}$$

Donde:

- A es el número de asignaturas obligatorias.

3. Tasa de cohesión de teoría por curso para un estudiante (maximizar):

Con este objetivo se representa que un alumno este matriculado en el mismo grupo de teoría para asignaturas del mismo curso, con esto se trata de evitar los tiempos muertos, ademas también puede favorecer a reducir los solapes entre horarios de asignaturas del mismo curso, se calcula de la siguiente manera.

Sea S el conjunto total de estudiantes, y para cada estudiante $s \in S$, tenemos C_s como el conjunto de cursos en los que está matriculado. Para cada curso $c \in C_s$, tenemos un conjunto de asignaturas $A_{s,c}$ en las que el estudiante está matriculado dentro de ese curso. Además, para cada asignatura $a \in A_{s,c}$, el estudiante está asignado a un grupo de teoría $g_{s,a}$.

La tasa de cohesión por curso para un estudiante s en un curso c , denotada como $T_{s,c}$ se calcula como la proporción del grupo más frecuente en el que el estudiante está matriculado dentro de ese curso, respecto al total de asignaturas del curso en las que está matriculado. Esto es:

$$T_{s,c} = \frac{\max(\text{freq}(g_{s,a}))}{|A_{s,c}|} \quad \forall a \in A_{s,c}$$

La tasa de cohesión promedio para un estudiante s , que toma en cuenta todos los cursos en los que está matriculado, se calcula como el promedio de las tasas de cohesión por curso de ese estudiante

$$T_s = \frac{1}{|C_s|} \sum_{c \in C_s} T_{s,c}$$

Finalmente, la tasa de cohesión general para todos los estudiantes de un individuo se obtiene como el promedio de las tasas de cohesión de todos los estudiantes:

$$T = \frac{1}{|S|} \sum_{s \in S} T_s$$

4. Tasa de cohesión prácticas por curso para un estudiante (maximizar):

Este objetivo representa lo mismo que el anterior pero esta vez con los grupos de prácticas.

Sea S el conjunto total de estudiantes, y para cada estudiante $s \in S$, tenemos C_s como el conjunto de cursos en los que está matriculado. Para cada curso $c \in C_s$, tenemos un conjunto de asignaturas $A_{s,c}$ en las que el estudiante está matriculado dentro de ese curso. Además, para cada asignatura $a \in A_{s,c}$, el estudiante está asignado a un subgrupo de practicas $sg_{s,a}$.

La tasa de cohesión por curso para un estudiante s en un curso c , denotada como $TP_{s,c}$ se calcula como la proporción del grupo de practicas más frecuente en el que el estudiante está matriculado dentro de ese curso, respecto al total de asignaturas del curso en las que está matriculado. Esto es:

$$TP_{s,c} = \frac{\max(\text{freq}(sg_{s,a}))}{|A_{s,c}|} \quad \forall a \in A_{s,c}$$

La tasa de cohesión promedio para un estudiante s , que toma en cuenta todos los cursos en los que está matriculado, se calcula como el promedio de las tasas de cohesión por curso de ese estudiante

$$TP_s = \frac{1}{|C_s|} \sum_{c \in C_s} TP_{s,c}$$

Finalmente, la tasa de cohesión general para todos los estudiantes de un individuo se obtiene como el promedio de las tasas de cohesión de todos los estudiantes:

$$TP = \frac{1}{|S|} \sum_{s \in S} T_s$$

5. Tasa coincidencias configuración inicial (maximizar):

Uno de los ficheros iniciales proporciona una configuración de matrículas realizada previamente por los estudiantes. Uno de los objetivos planteados consiste en maximizar la similitud entre la solución generada por el algoritmo y dicha configuración inicial, preservando así, en la medida de lo posible, las preferencias manifestadas por los alumnos.

Esta objetivo puede calcularse de la siguiente forma:

Para cada estudiante se calcula la proporción de coincidencias que tienen los grupos de teoría y practica de sus matriculas del individuo respecto a los que hay en la configuración inicial.

$$TC_a = \frac{\text{Numero de coincidencias}_a}{\text{Numero máximo de coincidencias}_a}$$

Después se calcula el promedio de todos los estudiantes

$$TC = \frac{1}{A} \sum_{a=1}^A TC_a$$

3.2. Algoritmo genético

3.2.1. Función de evaluación

Una vez definidos los objetivos a optimizar y la forma de medirlos, es necesario establecer una función de evaluación que permita cuantificar la calidad de una solución. Para ello, es fundamental que todos los objetivos se encuentren en una escala comparable, facilitando su combinación en una única métrica global.

En este caso, tres de los cuatro objetivos definidos se encuentran naturalmente en una escala entre 0 y 1. Sin embargo, el número total de solapes es una variable entera positiva no acotada, lo que requiere su normalización. Una estrategia habitual consiste en dividir el valor observado por un valor máximo teórico, pero dado que es difícil estimar el número máximo posible de solapes en este contexto, se opta por una aproximación alternativa.

En este caso, se toma como referencia la configuración inicial de matrículas proporcionada, calculando el número total de solapes que presenta. Este valor se utilizará como cota superior de referencia para normalizar el número de solapes en cada individuo, de modo que:

$$\text{Solapes Normalizados} = \frac{\text{Solapes}_{\text{individuo}}}{\text{Solapes}_{\text{configuración inicial}}}$$

Este enfoque garantiza una escala relativa adecuada para integrar esta métrica en la función de evaluación, es cierto que, en caso de que algún individuo supere el número de solapes de la configuración inicial, su valor normalizado podrá ser superior a 1. No obstante, este comportamiento es deseable, ya que penaliza aun mas a las soluciones con un alto grado de solapamiento.

Una vez normalizadas todas las métricas, se define la función de evaluación como una combinación lineal ponderada de los objetivos. Dado que se trata de una función que se desea maximizar, los objetivos formulados como métricas a maximizar se suman, mientras que los objetivos a minimizar se restan. De esta manera, se priorizan las soluciones que presentan mejores valores globales, teniendo en cuenta la importancia relativa de cada criterio a través de los pesos asignados en la ponderación.

$$f(i) = -w_1 * \text{Solapes Totales} + w_2 * E_{total} + w_3 * T + w_4 * TP + w_5 * TC$$

La asignación de los pesos intentara transmitir a la función de evaluación como de importante es cada objetivo, antes se ha explicado la importancia del equilibrio del número de alumnos por grupo, por lo que el peso w_3 tendra que tener un valor elevado, en este problema el valor de los pesos sera fijo, y todas las pruebas se harán con los siguiente valores.

- $w_1 = 0.1$
- $w_2 = 0.1$
- $w_3 = 0.55$
- $w_4 = 0.1$
- $w_5 = 0.15$

3.2.2. Población inicial

Una vez definida la representación del problema, el siguiente paso consiste en generar la población inicial. El tamaño de esta población estará determinado por un hiperparámetro previamente establecido. Para construir cada individuo, se parte de un clon de la configuración inicial de matrículas, ya que esta contiene la información sobre las asignaturas en las que cada alumno desea matricularse.

A partir de esta base, se procede a asignar, de forma aleatoria, los grupos de teoría y prácticas. La asignación de grupos de teoría se realiza por curso, de modo que todas las asignaturas de un mismo curso dentro de un individuo pertenecen al mismo grupo de teoría. En cambio, los subgrupos de prácticas se asignan de manera totalmente aleatoria para cada asignatura, independientemente del curso.

Esta estrategia de inicialización se adopta con el objetivo de evitar configuraciones excesivamente desestructuradas que puedan dificultar el proceso de evolución. Si cada asignatura tuviera un grupo completamente aleatorio, se generarían soluciones caóticas que requerirían muchas generaciones para converger, o incluso podrían inducir la convergencia prematura hacia óptimos locales de baja calidad. Por tanto, proporcionar una población inicial con cierta coherencia facilita que, a través de pequeñas modificaciones sucesivas, el algoritmo explore soluciones progresivamente mejores.

3.2.3. Selección

El proceso de selección juega un papel muy importante puesto que se encarga de decidir qué individuos van a pasar directamente a la generación siguiente o serán padres de los futuros descendientes.[Flores et al., 2022] Existen muchos métodos de selección, en este problema se implementarán tres métodos diferentes con los que se harán experimentos para determinar cual funciona mejor con este problema, los métodos implementados son:

1. **Basada en fitness:**

En este método cada individuo tendrá una probabilidad de ser elegido directamente proporcional al fitness, para calcular la probabilidad que tiene cada individuo se puede crear una distribución de probabilidad usando la formula referenciada en[Flores et al., 2022], de forma que la probabilidad de cada individuo es:

$$p_s(i) = \frac{f(i)}{\sum_{i=1}^N f(i)}$$

El problema de este método es que si existe un individuo muy bueno va a tener muchas probabilidades de ser seleccionado, acaparando la población y evitando la diversidad genética lo que supondrá no hacer una exploración de espacio de soluciones efectiva y pudiendo ocasionar un convergencia prematura y teniendo como resultado un optimo local.

2. **Basada en rango:**

Con este método se trata de atenuar la presión selectiva del método anterior[Flores et al., 2022]. La idea es crear un ranking de individuos según su fitness, donde mayor fitness ocupan mejores posiciones en El ranking, y con una formula asignar nuevas probabilidades sin tener en cuenta el fitness y solo teniendo en cuenta la posición en el ranking, teniendo una población de N individuos y que cada individuo ocupa una posición i en el ranking, podemos calcular las nuevas probabilidades selección con la siguiente formula[Flores et al., 2022]

$$p_s(i) = \frac{2(N - i + 1)}{N^2 + N}$$

Con esta formula los elementos con mejor fitness todavía tendrán mayor probabilidad de ser seleccionados pero se ha reducido bastante, de esta forma los elementos con peor fitness también tendrán posibilidades de ser seleccionados con lo que se tiene mayor diversidad genética y ofrece una mejor exploración del espacio de soluciones.

3. **Selección por torneo:** Este método de selección se basa en el mecanismo de torneo, cuya presión selectiva puede ajustarse mediante un parámetro k , que representa el tamaño del torneo. El procedimiento consiste en seleccionar aleatoriamente k individuos de la población y elegir aquel con mayor valor de fitness como candidato para la siguiente generación. Con este método cada individuo participa en k torneos [Flores et al., 2022].

La presión selectiva aumenta con el valor de k , ya que se incrementa la probabilidad de seleccionar individuos con fitness elevado. Por el contrario, valores bajos de k favorecen una mayor diversidad genética al ofrecer más oportunidades de selección a

individuos con menor fitness. Esta característica permite una exploración más amplia del espacio de soluciones, reduciendo la probabilidad de estancamiento en óptimos locales, aunque puede ralentizar la convergencia del algoritmo.

Un valor intermedio de k puede ofrecer un equilibrio adecuado entre exploración y explotación, permitiendo mantener una presión selectiva suficiente para mejorar progresivamente la población, sin comprometer la diversidad genética.

3.2.4. Cruce

Es el primer operador de variación que se aplica, a partir de dos padres seleccionados a través de alguno de los métodos anteriores se cruzaran para generar hijos nuevos, normalmente el cruce se suele realizar bajo una probabilidad bastante grande, en este problema se hará siempre, con este operador se mezcla los estudiantes de dos individuos, nunca se introduce material genético nuevo, por lo que al hacer esto se está explorando el espacio de soluciones para intentar encontrar una solución mejor a partir de otras soluciones buenas. Es importante recalcar que el cruce se hará a nivel de estudiante, cada individuo en la población tiene una lista de estudiantes, y cada estudiante tiene a su vez una lista de matrículas variables y matrículas fijas, el objetivo es optimizar los grupos de teoría y subgrupos de prácticas de cada matrícula variable. Existen dos enfoques de como realizar el cruce para este problema:

1. **Cruzar las matrículas variables individualmente:** Para cada estudiante, se pueden intercambiar las matrículas variables entre los padres, lo que significa que algunas provengan del padre 1 y otras del padre 2.
2. **Cruzar la configuración completa de cada estudiante:** De esta manera se selecciona la configuración completa de matrículas de un padre u otro para cada estudiante de forma independiente. Es decir, para cada estudiante Se decide si hereda toda la configuración del padre 1 o del padre 2.

La segunda opción, donde se hereda la configuración completa del estudiante, se considera más adecuada en este problema. Al cruzar individualmente las matrículas se corre el riesgo de desordenar demasiado el material genético, lo que podría ocasionar dificultades en la convergencia de una buena solución. Al heredar la configuración completa de cada estudiante, se realiza una exploración más controlada del espacio de soluciones, lo que facilita modificaciones graduales a través del operador de mutación.

Una vez explicado el esquema del cruce, hay que explicar su implementación. En los algoritmos genéticos existen distintos tipos de operadores de cruce, cada uno tiene unas características y ventajas. En este trabajo se utilizaran tres variantes que se explican a continuación:

1. **Un punto:** En este caso en cada generación se seleccionara un punto de manera aleatoria en la lista de alumnos para cada cruce, de forma que los segmentos de los padres antes de ese punto se mantienen, mientras que los segmentos posteriores se intercambian. Este es uno de los métodos más sencillos, su principal desventaja es que las combinaciones genéticas están limitadas por la ubicación del único punto de cruce.

-
2. **Varios puntos:** Este método es una evolución del anterior, ahora se eligen dos puntos de corte en cada cruce, las secciones de la lista de alumnos ubicadas entre estos dos puntos se intercambian entre los padres, el resto de segmentos se mantienen igual. Con esto se gana ventaja de aumentar la diversidad genética en comparación con el método anterior, lo que permite explorar un espacio de búsqueda más amplio, aunque puede ser insuficiente si los genes importantes están muy dispersos.
 3. **Uniforme:** Este cruce tiene una forma más aleatoria, ya que se basa en una máscara de cruce formada por bits que indica para cada alumno de la lista si proviene del padre 1 o del padre 2, esta máscara será aleatoria en cada cruce. Este método es el que mayor diversidad genética proporciona ya que cada alumno puede provenir de cualquiera de los padres sin seguir una estructura predeterminada. Este método es muy recomendable cuando se tiene que explorar un espacio de soluciones muy amplio, además combina muy bien con mutaciones de baja probabilidad ya que al combinar configuraciones completas de alumnos se harán pequeños cambios que hagan llegar a una buena solución.

3.2.5. Mutación

Este sera el segundo operador de variación que se use, se aplicara a los hijos generados en el proceso de cruce, consiste en introducir pequeñas modificaciones aleatorias en el individuo, en este caso la mutación cambiara el grupo de teoría o prácticas en las matriculas variables de los alumnos, al igual que el operador de cruce, el de mutación también ayuda a explorar el espacio de soluciones, pero de manera diferente, ya que el operador de cruce combina el material genético existente entre los individuos, el operador de mutación introduce nuevo material genético que no estaba presente en los padres. Esto da la capacidad al algoritmo genético de explorar áreas del espacio de búsqueda que no hubieran sido accesibles a través del cruce. Antes se ha mencionado que el cruce se aplicara siempre, la mutación es todo lo contrario, se suele aplicar con probabilidades muy pequeñas ya que se puede desorganizar la información genética ya adquirida. A continuación se va a explicar que probabilidades de mutación existen en este problema y como funcionan:

- *p_mutacion_alumno*: Este sera el primer filtro, para cada alumno de cada individuo se generara un número real aleatorio del 0 al 1 y si es menor que la probabilidad significa que podría mutar y se pasarían a las siguientes probabilidades.
- *p_mutacion_teoría*: Para los alumnos que han pasado el primer filtro se realizara el mismo proceso en sus matriculas variables, si se cumple la probabilidad se cambiara el grupo de teoría de esa matricula por uno distinto al actual
- *p_mutacion_practica*: Funciona igual que el anterior, se aplicara en las matriculas variables de los alumnos que pasaron el primer filtro, si se cumple la probabilidad se cambiara el grupo de práctica de esa matricula.

3.2.6. Sustitución de la población

Es una fase crucial en los algoritmos genéticos, en está fase de eligen que individuos de una generación se mantendrán o reemplazarán en la siguiente. Esto afecta tanto a la convergencia del algoritmo hacia una buena solución como a la diversidad genética de la población a lo largo de las generaciones. Existen muchos tipos de sustitución, a continuación se explicara cuales se han implementado para este problema y que pros y contras tiene cada uno. En la sustitución está la población con la que se empezó la generación, P_t y existirá otra población que ha sido la resultante de aplicar la selección, cruce y mutación P'

1. **Reemplazo**: Este es el método mas sencillo, P' se convertirá en la nueva generación P_{t+1} , de esta manera se hace una exploración intensa del espacio de soluciones, ya que introduce una población nueva en cada generación, las principales desventajas son que puede llevar a una convergencia mas lenta ya que no se conserva ninguna información de los mejores individuos de la generación anterior, esto también lleva a perder soluciones de alta calidad si los descendientes no logran alcanzar el mismo nivel de fitness.
2. **Elitismo**: Este método de sustitución tiene como objetivo conservar la mejor solución encontrada hasta el momento, corrigiendo una de las principales limitaciones del método anterior. En este enfoque, la nueva población P_{t+1} se construye seleccionando

todos los individuos generados en la población nueva P' , excepto aquel con peor valor de fitness, y reemplazándolo por el mejor individuo de la población actual P_t . Esta estrategia garantiza que el mejor individuo no se pierda durante el proceso evolutivo, lo que favorece una convergencia más rápida hacia soluciones de alta calidad. Sin embargo, la inclusión excesiva de individuos de generaciones anteriores puede reducir la diversidad genética y aumentar el riesgo de estancamiento en óptimos locales. Por este motivo, en esta implementación se conserva únicamente el mejor individuo de P_t , lo que representa un compromiso entre preservación de calidad y mantenimiento de diversidad en la población.

3. **Truncamiento:** Este método de sustitución combina la población actual P_t y la generada P' para formar la nueva población P_{t+1} . El procedimiento consiste en unir ambas poblaciones y ordenarlas de forma descendente según su valor de fitness. A continuación, se seleccionan los individuos con mayor fitness hasta alcanzar el tamaño de población original. En caso de empate entre individuos de P_t y P' , se prioriza la inclusión del individuo perteneciente a P' . Esta estrategia garantiza que la siguiente generación esté compuesta por los individuos más destacados en términos de calidad, lo que puede acelerar la convergencia del algoritmo hacia soluciones óptimas. Además, permite una preservación selectiva de características favorables presentes tanto en los progenitores como en los descendientes, facilitando un equilibrio entre explotación y exploración del espacio de búsqueda. Este método tiene el problema de que al eliminar por completo a los individuos con menor fitness, puede reducir considerablemente la diversidad genética de la población. En consecuencia, existe un mayor riesgo de estancamiento en óptimos locales si no se combina adecuadamente con operadores de mutación y cruce que favorezcan la introducción de variabilidad.

3.2.7. Condición de parada

La condición de parada define el criterio mediante el cual el algoritmo genético finaliza su ejecución. Dado que se trata de un proceso iterativo orientado a mejorar progresivamente la calidad de las soluciones a lo largo de múltiples generaciones, establecer un criterio adecuado de finalización resulta fundamental para alcanzar un equilibrio entre la calidad de la solución obtenida y la eficiencia computacional.

Existen diversos enfoques para definir esta condición. Una opción es detener la ejecución cuando se alcanza una solución óptima. Sin embargo, este enfoque presenta limitaciones importantes, especialmente en problemas complejos en los que no se conoce a priori el valor óptimo, lo que podría implicar un coste computacional elevado e innecesario.

Otra alternativa común es establecer la parada en función de la convergencia de la población, es decir, cuando las soluciones generadas en sucesivas generaciones presentan escasa variabilidad. No obstante, este método puede inducir una convergencia prematura hacia óptimos locales, y su detección requiere un seguimiento continuo de la diversidad poblacional, lo cual incrementa la complejidad del algoritmo.

En esta implementación se ha optado por utilizar un enfoque basado en un número máximo de generaciones como criterio de parada. Este método presenta la ventaja de ser sencillo de implementar y garantiza un control directo sobre el tiempo de ejecución. La principal

desventaja radica en que una cantidad insuficiente de generaciones puede impedir alcanzar soluciones de calidad, mientras que un número excesivo puede suponer un desperdicio de recursos computacionales una vez alcanzada la convergencia. Por este motivo, el número de generaciones se establecerá a partir de los resultados obtenidos en la fase de experimentación, buscando un compromiso adecuado entre eficiencia y rendimiento.

3.2.8. Ficheros de salida

Como resultado de la ejecución del algoritmo genético, se generan dos ficheros de salida principales que documentan los resultados finales de la asignación de alumnos a los grupos de teoría y subgrupos de práctica. Estos ficheros contienen información sobre la configuración de las matriculas para cada alumno y las distribución total de alumnos en cada grupo de teoría y subgrupo de práctica para cada asignatura. El objetivo de estos ficheros es permitir un análisis detallado de la solución obtenida y facilitar la interpretación de los resultados finales.

- *Fichero de configuraciones de matriculas:*

Este fichero almacena la configuración de matriculas de la solución con mayor fitness del algoritmo genético, la estructura sera igual que la del fichero *matriculas.xlsx* que se ha explicado en el capítulo anterior.

- *Fichero de distribución de alumnos por grupo:*

Este fichero constituye un elemento clave dentro del conjunto de resultados generados, ya que permite analizar el grado de cumplimiento del objetivo de equilibrio de alumnos entre los distintos grupos de teoría y subgrupos de prácticas de cada asignatura. Este objetivo es especialmente relevante, ya que una distribución desequilibrada puede derivar en asignaciones inviables desde el punto de vista logístico, debido a las limitaciones físicas de las aulas. Por ello, este archivo resulta esencial para evaluar la viabilidad real de las soluciones propuestas por el algoritmo.

El fichero se presenta en formato Excel y contiene dos hojas diferenciadas:

- La primera hoja recoge la distribución de alumnos correspondiente a la configuración inicial, es decir, aquella proporcionada como punto de partida para la ejecución del algoritmo.
- La segunda hoja muestra la distribución resultante de la mejor solución obtenida por el algoritmo genético tras su ejecución.

En ambas hojas, las columnas del fichero, ordenadas de izquierda a derecha, son las siguientes:

- **NOMBRE:** Nombre completo de la asignatura
- **ALUMNOS:** Número total de alumnos matriculados en la asignatura
- **ESPERADOS POR GRUPO TEORÍA:** Número ideal de alumnos en cada grupo de teoría para esa asignatura

-
- **ESPERADOS POR GRUPO PRÁCTICAS:** Número ideal de alumnos en cada subgrupo de prácticas para esa asignatura
 - **GRUPO 10:** Número de alumnos asignados al grupo 10
 - **GP1:** Número de alumnos en el primer subgrupo de prácticas del grupo 10
 - **GP2:** Número de alumnos en el segundo subgrupo de prácticas del grupo 10
 - **GRUPO 11:** Número de alumnos asignados al grupo 11
 - **GP1:** Número de alumnos en el primer subgrupo de prácticas del grupo 11
 - **GP2:** Número de alumnos en el segundo subgrupo de prácticas del grupo 11
 - **GRUPO 12:** Número de alumnos asignados al grupo 12
 - **GP1:** Número de alumnos en el primer subgrupo de prácticas del grupo 12
 - **GP2:** Número de alumnos en el segundo subgrupo de prácticas del grupo 12

Además, para facilitar la identificación visual de posibles desequilibrios, junto a cada columna que indica el número de alumnos asignados a un grupo o subgrupo, se incluye una columna adicional que refleja el grado de desviación respecto al valor ideal. Cuando esta desviación es igual o superior al 15 %, la celda correspondiente se resalta con un fondo rojo. Esta visualización permite detectar rápidamente situaciones problemáticas y valorar de forma más precisa la viabilidad de la solución desde el punto de vista del equilibrio de carga entre grupos.

3.3. NSGA-III

3.3.1. Población Inicial

Igual que en el algoritmo genético, el primer paso consiste en generar una población inicial de soluciones. El número de individuos de la población inicial lo marca un hiperparámetro, se usará la misma lógica descrita anteriormente 3.2.2, cada uno de ellos se creará clonando la configuración inicial que contiene las asignaturas en las que desea matricularse cada alumno. Posteriormente, se asignarán aleatoriamente los grupos de teoría y prácticas para cada asignatura de forma coherente, un grupo de teoría común por curso y grupos de prácticas aleatorios para cada asignatura, con el objetivo de evitar soluciones excesivamente caóticas y favorecer una convergencia más eficiente.

En NSGA-III esto es la primera parte, una vez generada esta población inicial P_t , se usará y aplicando los operadores de cruce y mutación se creará una segunda población Q_t del mismo tamaño. A continuación, ambas poblaciones se combinan en una única población extendida $R_t = P_t \cup Q_t$, esta unión será la que se usará como población inicial.

Esta combinación es necesaria ya que en los siguientes pasos del algoritmo, hay que seleccionar N individuos para formar la nueva generación. Si solo se crea la única población inicial P_t , no existiría un proceso de competición real, ya que todos los individuos pasarían automáticamente a la siguiente generación.

3.3.2. Puntos de referencia

La principal innovación del algoritmo NSGA-III frente a sus predecesores es el uso de puntos de referencia para guiar el proceso de selección y mantener la diversidad de las soluciones en los problemas de optimización con múltiples objetivos.

Se proyectan las soluciones sobre un conjunto de direcciones predefinidas en el espacio de los objetivos. Estas direcciones están definidas por un conjunto de puntos uniformemente distribuidos sobre el simplex m -dimensional, siendo m el número de objetivos del problema.

La técnica que se utiliza para generar los puntos de referencia es del trabajo de Das y Dennis [Das and Dennis, 1998], quienes propusieron un método de divisiones equiespaciadas para construir una malla de puntos uniformes sobre un simplex. Esta técnica consiste en encontrar todas las combinaciones enteras no negativas (a_1, a_2, \dots, a_m) que suman un valor entero H , llamado número de divisiones por objetivo:

$$\sum_{i=1}^m a_i = H, \quad a_i \in \mathbb{Z}_{\geq 0}$$

Cada combinación generara un punto de referencia al ser normalizada en el simplex unitario:

$$\mathbf{p} = \left(\frac{a_1}{H}, \frac{a_2}{H}, \dots, \frac{a_m}{H} \right), \quad \sum_{i=1}^m p_i = 1$$

El parámetro H determina el nivel de granularidad o resolución de los puntos generados, esto significa que cuanto mayor sea H , mayor será el número de puntos, por lo que la cobertura del espacio objetivo será mejor. Sin embargo, esto también aumenta el coste computacional.

El número total de puntos generados mediante este proceso se obtiene aplicando la fórmula de combinaciones con repetición:

$$N = \binom{H + m - 1}{m - 1}$$

3.3.3. Clasificación no dominada

Una vez evaluados todos los individuos de la población, el siguiente paso consiste en clasificarlos en distintos frentes de no dominancia. Esta clasificación permite organizar las soluciones según su grado de optimalidad relativa, y es fundamental para guiar el proceso evolutivo en algoritmos como NSGA-III.

El primer frente estará formado por aquellos individuos que no son dominados por ningún otro, es decir, soluciones que no presentan otra que sea mejor en todos los objetivos simultáneamente. El segundo frente contendrá individuos que solo son dominados por los del primer frente, el tercero por los dos anteriores, y así sucesivamente.

Para llevar a cabo esta clasificación, se emplea el siguiente procedimiento:

1. Para cada individuo de la población se calcula:

- El número de veces que es dominado por otros individuos (contador de dominancia).
 - Una lista con los individuos a los que domina.
2. Los individuos cuyo contador de dominancia sea cero (es decir, no dominados por ningún otro) se asignan al primer frente.
3. A continuación, para cada individuo en el frente actual, se recorren los elementos a los que domina y se decrementa en una unidad su contador de dominancia. Si tras esta operación el contador de un individuo llega a cero, se incluye en el siguiente frente.
4. Este proceso se repite iterativamente hasta que todos los individuos hayan sido asignados a un frente.

Este método garantiza una separación jerárquica de la población basada en su calidad relativa respecto a los objetivos del problema, lo que permite aplicar una presión selectiva ordenada y progresiva durante el proceso evolutivo.

3.3.4. Asignación a puntos de referencia

Una vez que los individuos han sido organizados en frentes de no dominancia, el siguiente paso en NSGA-III consiste en asignarlos a puntos de referencia. Estos puntos están distribuidos uniformemente sobre un *simplex* m-dimensional, lo que permite preservar la diversidad de soluciones y asegurar una cobertura equilibrada del espacio objetivo.

Para realizar esta asignación correctamente, es necesario normalizar previamente los valores de los objetivos de cada individuo. Aunque estos ya se encuentren en el rango $[0, 1]$, la normalización garantiza que todas las soluciones estén alineadas con la geometría del *simplex*, es decir, que compartan la misma base geométrica que los puntos de referencia. De no hacerse así, las distancias podrían estar sesgadas y conducir a asignaciones inadecuadas.

Dado un individuo \mathbf{x} con objetivos $f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})$, la normalización se realiza como:

$$f_i^{\text{norm}}(\mathbf{x}) = \frac{f_i(\mathbf{x}) - z_i^{\text{mín}}}{z_i^{\text{máx}} - z_i^{\text{mín}} + \varepsilon}$$

donde:

- $z_i^{\text{mín}}$ y $z_i^{\text{máx}}$ son los valores mínimo y máximo observados del objetivo i en la población actual,
- ε es una constante pequeña (por ejemplo, 10^{-10}) que evita divisiones por cero,
- y los objetivos a maximizar deben invertirse previamente, convirtiéndolos en problemas de minimización.

Una vez normalizados, los individuos se proyectan sobre todos los puntos de referencia para identificar el más cercano. En NSGA-III esta cercanía se mide mediante la *distancia perpendicular* a la dirección que define cada punto de referencia, no mediante la distancia euclidiana directa. Esta elección mejora la asignación al identificar mejor la región del espacio objetivo que representa cada punto.

Sea $\mathbf{x} \in \mathbb{R}^m$ el vector de objetivos normalizado y $\mathbf{r} \in \mathbb{R}^m$ un punto de referencia, la distancia perpendicular se define como:

$$d_{\perp}(\mathbf{x}, \mathbf{r}) = \left\| \mathbf{x} - \left(\frac{\mathbf{x} \cdot \mathbf{r}}{\|\mathbf{r}\|^2} \right) \mathbf{r} \right\|$$

donde:

- $\mathbf{x} \cdot \mathbf{r}$ es el producto escalar,
- $\|\mathbf{r}\|^2$ es la norma cuadrada de \mathbf{r} ,
- $\|\cdot\|$ indica la norma euclidiana (L2).

Cada individuo se asocia finalmente al punto de referencia con menor distancia perpendicular. Este mecanismo permite detectar regiones del frente de Pareto que están escasamente representadas, facilitando una selección más equitativa y mejorando la diversidad global. Este aspecto es especialmente importante en problemas con muchos objetivos, donde las soluciones tienden a agruparse en ciertas zonas densas del espacio objetivo.

3.3.5. Selección

Una vez que los individuos han sido clasificados en frentes de no dominancia y asignados a su punto de referencia más cercano, se procede al proceso de selección para determinar cuáles pasarán a la siguiente generación. El número total de individuos a seleccionar está definido por un hiperparámetro que representa el tamaño de la población.

El procedimiento comienza añadiendo directamente todos los individuos de los primeros frentes, mientras no se supere el tamaño máximo. Sin embargo, cuando se alcanza un frente cuyo número de individuos excede la capacidad restante, se requiere aplicar un criterio adicional para decidir cuáles deben ser seleccionados.

En este paso, cobran importancia los puntos de referencia, ya que permiten preservar la diversidad en el espacio de soluciones. Para ello, se consideran todos los puntos de referencia y se ordenan en orden ascendente en función del número de individuos que ya han sido seleccionados y están asociados a cada punto.

A continuación, se revisan los puntos de referencia ordenados. Si un punto tiene actualmente cero individuos asociados (es decir, ningún individuo seleccionado hasta ese momento lo tiene como punto más cercano), se distinguen dos casos:

- Si *ningún individuo del frente actual* tiene ese punto como más cercano, se pasa al siguiente punto.
- Si *uno o más individuos del frente actual* tienen ese punto como más cercano, se selecciona:

-
- directamente si hay un solo individuo;
 - el individuo con **menor distancia angular** al punto, en caso de que haya varios.

En caso de que el punto de referencia ya tenga uno o más individuos asociados, se comprueba si alguno de los individuos del frente actual tiene ese punto como más cercano. Si es así, se selecciona:

- el único individuo, si solo hay uno;
- o el más cercano al punto, si hay varios.

Este proceso se repite iterativamente hasta completar el número exacto de individuos requeridos para la siguiente generación, garantizando de este modo un compromiso adecuado entre calidad de las soluciones (según su nivel de no dominancia) y su dispersión en el espacio objetivo (diversidad).

Una vez seleccionados los individuos que formarán la siguiente generación, el siguiente paso consiste en aplicar los operadores genéticos de cruce y mutación para generar la nueva descendencia.

Antes de aplicar el cruce, se realizan $\frac{N}{2}$ iteraciones (siendo N el tamaño de la población), en cada una de las cuales se seleccionan dos padres sobre los que se aplicará el operador de cruce, generando así dos hijos. A continuación, a los hijos resultantes se les aplicará la mutación.

En el desarrollo del algoritmo genético clásico se implementaron distintos métodos de selección basados en el valor de la función objetivo. Sin embargo, en NSGA-III, al tratarse de un algoritmo evolutivo multiobjetivo, no se dispone de una única función de aptitud, por lo que ha sido necesario adaptar el método de *selección por torneo*.

En este caso, también se introduce un hiperparámetro k que indica el tamaño del torneo. Para seleccionar un padre, se eligen aleatoriamente k individuos de la población, y se escoge aquel que pertenezca al mejor frente de no dominancia. En caso de que varios candidatos estén en el mismo frente, se selecciona el individuo cuya distancia al punto de referencia asignado sea menor, fomentando así la diversidad.

Al igual que ocurre en el algoritmo clásico, un valor más alto de k implica una mayor presión selectiva, ya que al tener más candidatos en el torneo, es más probable que se escojan individuos pertenecientes a frentes de mejor calidad.

3.3.6. Cruce y Mutación

Una vez completado el proceso de selección descrito anteriormente, se procede a aplicar los operadores genéticos de cruce y mutación para generar la nueva población. En esta implementación del algoritmo NSGA-III, se utilizarán los mismos operadores que los definidos previamente en el desarrollo del algoritmo genético clásico.

Se mantienen los tres tipos de cruce implementados (cruce de un punto, de varios puntos y uniforme), así como la estructura del operador de mutación con sus respectivos parámetros de probabilidad: `p_mutacion_alumno`, `p_mutacion_teoría` y `p_mutacion_práctica`. Esta elección permite conservar la lógica de exploración y explotación del espacio de soluciones ya diseñada, aplicándola ahora en un contexto multiobjetivo.

3.3.7. Reemplazo

A diferencia de lo que ocurre en el algoritmo genético clásico, donde se aplican distintos métodos de reemplazo en función del valor de la función objetivo, en NSGA-III se realiza una sustitución completa de la generación. Este enfoque se alinea con la lógica evolutiva del algoritmo, centrada en la clasificación por frentes de no dominancia y el mantenimiento de la diversidad mediante puntos de referencia.

Después de aplicar el proceso de selección, cruce y mutación, se ha obtenido una población de N individuos. Sin embargo, como se ha explicado al inicio, el algoritmo requiere una población extendida de tamaño $2N$ para realizar el proceso evolutivo completo. Por tanto, es necesario generar otros N individuos adicionales a partir de los ya existentes de nuevo.

Para ello, se evalúa a los individuos de la población y se realiza nuevamente la ordenación de la población actual en frentes de no dominancia, seguida de la asignación de los individuos a sus puntos de referencia más cercanos. Con esta estructura, se lleva a cabo una nueva selección de padres, sobre los cuales se aplican los operadores de cruce y mutación. De esta forma, se obtienen nuevos descendientes, que se añaden a la población existente para formar la población extendida de individuos.

Una vez formada esta población extendida, se vuelve a empezar el ciclo. Este ciclo se repite de forma iterativa hasta que se cumpla la condición de parada definida.

3.3.8. Condición de parada

El último paso consiste en definir la condición de parada del algoritmo. Al igual que en el desarrollo del algoritmo genético clásico, se establece un número máximo de generaciones como límite principal para la ejecución. Sin embargo, debido a la mayor complejidad computacional que implica NSGA-III, tanto por la evaluación multiobjetivo como por la clasificación en frentes y el uso de puntos de referencia, se introduce una condición adicional con el objetivo de agilizar las pruebas y optimizar el uso de recursos.

Una de las características clave de NSGA-III es su capacidad para mantener la diversidad entre las soluciones a lo largo de las generaciones. Por tanto, si se detecta que el mejor frente no ha variado entre generaciones consecutivas, se considerará que el algoritmo ha entrado en un estado de estancamiento. Esta falta de evolución sugiere que el algoritmo ha dejado de explorar nuevas regiones del espacio de soluciones, lo cual justifica su detención anticipada. Esta estrategia permite ahorrar tiempo de cómputo sin comprometer la calidad del resultado final, especialmente en casos donde el algoritmo ya ha convergido o ha dejado de mostrar mejoras significativas.

3.3.9. Ficheros de salida

En el caso del algoritmo *NSGA-III*, los ficheros de salida generados siguen la misma estructura descrita en la sección 3.2.8, correspondiente al algoritmo genético clásico. La principal diferencia es el criterio de selección de las soluciones exportadas: mientras que en el caso clásico se guarda únicamente la solución con mayor valor de *fitness*, en *NSGA-III* se consideran todos los individuos que forman parte del frente de Pareto final.

Para cada uno de estos individuos, se crea una carpeta independiente dentro de la ruta de resultados. En cada carpeta se generan los mismos dos archivos explicados anteriormente:

- *Fichero de configuraciones de matrículas*: contiene la asignación concreta de grupos para el individuo correspondiente.
- *Fichero de distribución de alumnos por grupo*: muestra el reparto de estudiantes en grupos de teoría y subgrupos de prácticas, comparado con la distribución inicial, permitiendo evaluar el equilibrio alcanzado.

Esta estrategia permite un análisis individualizado de cada solución no dominada generada por el algoritmo, facilitando la comparación entre ellas y ofreciendo al usuario final varias opciones que representan diferentes compromisos entre los objetivos definidos.

4. Experimentación

4.1. Introducción

En el capítulo anterior se ha descrito el desarrollo de los dos algoritmos genéticos propuestos para resolver el problema: el algoritmo genético clásico y el NSGA-III. Ambos algoritmos cuentan con diferentes mecanismos en algunos pasos como el cruce, así como una serie de hiperparámetros que influyen directamente en su comportamiento y en la calidad de las soluciones obtenidas.

Dado que no existe una configuración única que garantice los mejores resultados para todos los casos, en este capítulo se presentará una serie de experimentos con el objetivo de identificar las combinaciones más eficaces de métodos e hiperparámetros. Para ello se ha diseñado una metodología en tres fases:

- En primer lugar, se evaluarán todas las combinaciones posibles de métodos de selección, cruce y sustitución del algoritmo genético clásico, utilizando una configuración fija de hiperparámetros.
- A partir de los mejores resultados obtenidos en la fase anterior, se llevará a cabo un segundo bloque de experimentación para estudiar distintas combinaciones de hiperparámetros, usando las mejores combinaciones de métodos obtenidas en la primera fase.
- Finalmente, se aplicará el conocimiento obtenido en las fases anteriores para definir y ejecutar una serie de experimentos con el algoritmo NSGA-III, evaluando el impacto de los hiperparámetros en su rendimiento.

Este enfoque progresivo permite analizar de forma estructurada el comportamiento de ambos algoritmos y facilita la identificación de configuraciones óptimas para el problema planteado.

4.2. Entorno de ejecución

Todos los experimentos se han realizado en una maquina con un procesador AMD Ryzen 9 7900X 12-Core Processor a 4.70 GHz y 64 GB de memoria ram, en cuanto al software se ha usado python 3.9. Esta configuración asegura que los resultados obtenidos son reproducibles y permite comparar el rendimiento de distintas configuraciones de manera consistente.

4.3. Evaluación de métodos del algoritmo genético

En esta fase el objetivo es evaluar distintas combinaciones de métodos de selección, cruce y sustitución usando una configuración estática de hiperparámetros. Con esto se podrá identificar la configuración de métodos que maximiza la calidad de la solución y la estabilidad de la convergencia, estableciendo una base sólida para la fase posterior de optimización de hiperparámetros. La idea es probar todas las combinaciones que existen de los métodos que se han implementado, antes se han descrito los ficheros de salida que se crean en la ejecución de este algoritmo genético 3.2.8.

Esos ficheros sirven para ver la configuración resultante de las matriculas de los alumnos y el equilibrio en los grupos de cada asignatura, pero no se pueden usar para ver la evolución que ha tenido el algoritmo, durante la ejecución se va imprimiendo el *fitness* y el valor de cada objetivo de todos los individuos en cada generación para que el usuario pueda ver como evoluciona el algoritmo en una ejecución. Pero en este caso, al tener tantas combinaciones no sería practico ejecutar manualmente por separado cada combinación y ver los resultados, se creara un *script* que ejecute cada combinación y cree ficheros que permitan su análisis posteriormente, dado que la ejecución de cada combinación es independiente y se dispone de una maquina con un procesador potente, el *script* hará uso de concurrencia para aprovechar los recursos de la maquina y que todas las combinaciones se ejecuten en paralelo. Este enfoque permite un aprovechamiento óptimo De los recursos de la máquina, reduciendo el tiempo total de ejecución. Aunque cada combinación individual pueda tardar ligeramente más debido a la carga concurrente, el tiempo total es significativamente menor en comparación con una ejecución secuencial. El código completo del script se encuentra en el repositorio que hay en el Anexo A.1. Los ficheros que se crean con este script que ayudarán a identificar las mejores combinaciones son los explicados a continuación.

■ *resumen.csv*:

En este fichero estará la información clave de cada combinación, las columnas que tiene este fichero de izquierda a derecha son:

- **(Selección,Cruce,Sustitución)**: Indica la combinación de métodos
- **Mejor Generación**: Generación donde se encontró el individuo con mejor fitness
- **Mejor Fitness**: Valor del fitness del mejor individuo, seguido de los valores obtenidos en cada uno de los objetivos definidos (en el siguiente orden: *solapes*, *cohesión de teoría*, *equilibrio*, *cohesión de prácticas* y *tasa de preferencias*).
- **Tiempo Total**: Indica el tiempo que ha tardado en ejecutar la combinación

■ *seleccion-cruce-sustitucion.csv*:

Este fichero se genera por cada combinación, a diferencia del primero este contiene información de cada generación, las columnas que tiene este fichero de izquierda a derecha son:

- **Generacion**: Indica la generación
- **Mejor_Fitness**: Indica el fitness y el valor de los objetivos del mejor individuo de esa generación, en el mismo orden descrito anteriormente
- **Promedio_Fitness**: Indica el fitness promedio de la generación
- **Peor_Fitness**: Indica el fitness y el valor de los objetivos del peor individuo de esa generación, en el mismo orden descrito anteriormente
- **Desviacion_Estandar**: Indica la desviación estándar del fitness en la generación
- **Tiempo_Generacion**: Indica el tiempo que ha tardado en ejecutar la generación

Con estos ficheros se puede analizar de una forma sencilla que combinaciones son las que mejor funcionan, con el primer fichero se detectará rápidamente cuáles son las que mejores resultados están dando, y con el segundo fichero se puede analizar mejor cada combinación y ver si ha convergido o como ha ido evolucionando el fitness.

La configuración fija de hiperparámetros que se ha usado para esta prueba es la siguiente:

- *poblacion* = 500
- *generaciones* = 50
- *mutacion_elemento* = 0.00009
- *mutacion_teorica* = 0.0005
- *mutacion_practica* = 0.0005
- *t_torneo* = 10

Después de ejecutar el script de experimentación de la primera fase se procede al análisis del contenido del fichero de salida *resumen.csv*, a continuación se muestra ordenado de mayor a menor fitness:

Seleccion Cruce Sustitucion	Mejor Generacion	Mejor Fitness	Tiempo Total
(torneo uniforme reemplazo)	50	0.7619 (139 0.9997 0.9601 0.7329 0.5979)	9.58 min
(torneo uniforme elitismo)	50	0.7608 (145 0.9995 0.9603 0.7269 0.6001)	16.99 min
(torneo uniforme truncamiento)	49	0.7427 (202 1.0000 0.9699 0.7207 0.5403)	16.93 min
(torneo varios_puntos reemplazo)	30	0.7216 (250 0.9996 0.9555 0.7186 0.5040)	9.08 min
(rango punto elitismo)	49	0.7189 (260 0.9991 0.9532 0.7143 0.5087)	16.63 min
(rango punto reemplazo)	49	0.7184 (237 0.9997 0.9468 0.7155 0.5055)	9.06 min
(rango varios_puntos reemplazo)	49	0.7182 (209 0.9996 0.9314 0.7152 0.5148)	9.24 min
(rango varios_puntos elitismo)	49	0.7175 (216 0.9997 0.9358 0.7191 0.5051)	16.65 min
(rango punto truncamiento)	49	0.7172 (243 1.0000 0.9456 0.7183 0.5104)	16.52 min
(rango uniforme reemplazo)	48	0.7160 (205 1.0000 0.9268 0.7290 0.5094)	8.11 min
(rango uniforme truncamiento)	44	0.7153 (215 1.0000 0.9285 0.7165 0.5117)	13.07 min
(torneo varios_puntos elitismo)	23	0.7152 (248 0.9997 0.9481 0.7207 0.4850)	16.49 min
(rango uniforme elitismo)	46	0.7151 (216 1.0000 0.9255 0.7172 0.5336)	13.10 min
(torneo punto reemplazo)	22	0.7141 (239 1.0000 0.9423 0.7133 0.5041)	8.91 min
(fitness varios_puntos truncamiento)	50	0.7137 (217 1.0000 0.9398 0.7158 0.4870)	16.74 min
(rango varios_puntos truncamiento)	46	0.7135 (249 0.9992 0.9430 0.7137 0.5052)	16.79 min
(fitness punto truncamiento)	50	0.7129 (246 1.0000 0.9376 0.6968 0.5226)	16.59 min
(torneo varios_puntos truncamiento)	30	0.7111 (264 0.9997 0.9448 0.7203 0.5086)	16.41 min
(fitness uniforme truncamiento)	41	0.7050 (226 1.0000 0.9269 0.7115 0.4795)	17.47 min
(torneo punto elitismo)	43	0.7048 (259 1.0000 0.9363 0.7070 0.4784)	16.41 min
(torneo punto truncamiento)	16	0.7046 (255 0.9996 0.9369 0.7115 0.4704)	16.43 min
(fitness varios_puntos elitismo)	50	0.6974 (257 1.0000 0.9189 0.7175 0.4833)	16.73 min
(fitness uniforme elitismo)	50	0.6953 (265 0.9997 0.9196 0.7106 0.4874)	17.34 min
(fitness varios_puntos reemplazo)	15	0.6941 (255 1.0000 0.9232 0.7041 0.4645)	9.04 min
(fitness punto reemplazo)	45	0.6923 (258 1.0000 0.9187 0.7023 0.4696)	9.09 min
(fitness punto elitismo)	28	0.6919 (290 1.0000 0.9237 0.7066 0.4910)	16.65 min
(fitness uniforme reemplazo)	46	0.6896 (293 1.0000 0.9248 0.7079 0.4844)	9.38 min
			6.26 hr
			23.57 min

Tabla 4.1: Resultados primer script de experimentacion

A primera vista se puede observar que los mejores métodos de selección y cruce son torneo y uniforme respectivamente, ya que esa combinación con los distintos métodos de sustitución forman las tres mejores combinaciones, también se puede ver que en las dos primeras la mejor generación ha sido la última y en el tercer caso ha sido la penúltima, esto es buena señal ya que significa que la evolución no se ha quedado estancada y ha ido mejorando a lo largo de las generaciones haciendo una búsqueda en el espacio de soluciones de manera optima, por ejemplo, tomando la cuarta combinación, la mejor generación ha sido la 30, lo que significa que se ha tenido una convergencia prematura y no ha podido evolucionar más, analizando el contenido del fichero que se crea por cada combinación se puede observar el proceso de evolución que ha tenido durante cada generación.

Generación	Mejor Fitness	Promedio Fitness	Peor Fitness	Desviación Estandar	Tiempo
1	0.6819 (283 1.0000 0.9075 0.7207 0.4740)	0.6586	0.6007 (275 1.0000 0.7625 0.7047 0.4629)	0.0109	12.37 seg
2	0.6892 (279 1.0000 0.9192 0.7181 0.4756)	0.6691	0.6264 (313 1.0000 0.8254 0.6983 0.4623)	0.0077	11.45 seg
3	0.6919 (270 0.9997 0.9213 0.7152 0.4752)	0.6751	0.6498 (277 1.0000 0.8518 0.7037 0.4670)	0.0070	12.13 seg
4	0.6922 (270 0.9997 0.9218 0.7161 0.4748)	0.6792	0.6542 (289 1.0000 0.8616 0.7150 0.4696)	0.0060	12.60 seg
5	0.6967 (257 1.0000 0.9254 0.7169 0.4721)	0.6822	0.6540 (278 1.0000 0.8620 0.7106 0.4544)	0.0064	12.31 seg
6	0.6977 (273 1.0000 0.9246 0.7089 0.5103)	0.6851	0.6632 (270 1.0000 0.8664 0.7199 0.4819)	0.0063	10.77 seg
7	0.6987 (251 1.0000 0.9268 0.7125 0.4751)	0.6873	0.6641 (256 1.0000 0.8622 0.7197 0.4837)	0.0068	10.39 seg
8	0.7017 (251 1.0000 0.9310 0.7114 0.4803)	0.6904	0.6674 (261 1.0000 0.8722 0.7072 0.4843)	0.0061	10.75 seg
9	0.7030 (247 1.0000 0.9305 0.7077 0.4877)	0.6926	0.6706 (263 1.0000 0.8790 0.7127 0.4798)	0.0056	13.33 seg
10	0.7045 (250 1.0000 0.9354 0.7117 0.4812)	0.6945	0.6685 (260 1.0000 0.8765 0.7129 0.4704)	0.0054	10.41 seg
11	0.7071 (248 1.0000 0.9392 0.7115 0.4820)	0.6977	0.6727 (246 1.0000 0.8778 0.7068 0.4776)	0.0045	11.82 seg
12	0.7071 (248 1.0000 0.9392 0.7115 0.4820)	0.7016	0.6894 (249 1.0000 0.9089 0.7077 0.4788)	0.0024	11.01 seg
13	0.7073 (246 1.0000 0.9349 0.7114 0.4958)	0.7042	0.6965 (246 1.0000 0.9201 0.7069 0.4812)	0.0014	10.94 seg
14	0.7087 (244 1.0000 0.9419 0.7119 0.4769)	0.7058	0.7016 (239 1.0000 0.9278 0.7105 0.4749)	0.0011	10.40 seg
15	0.7103 (242 1.0000 0.9434 0.7126 0.4783)	0.7070	0.7029 (244 1.0000 0.9286 0.7147 0.4851)	0.0008	11.22 seg
16	0.7118 (238 1.0000 0.9448 0.7142 0.4764)	0.7077	0.7040 (247 1.0000 0.9343 0.7090 0.4794)	0.0011	10.04 seg
17	0.7118 (238 1.0000 0.9448 0.7142 0.4764)	0.7090	0.7038 (244 1.0000 0.9293 0.7156 0.4875)	0.0013	10.52 seg
18	0.7124 (242 1.0000 0.9473 0.7123 0.4782)	0.7106	0.7051 (242 1.0000 0.9322 0.7121 0.4850)	0.0009	10.38 seg
19	0.7126 (238 1.0000 0.9463 0.7137 0.4766)	0.7116	0.7096 (241 1.0000 0.9415 0.7126 0.4797)	0.0004	14.68 seg
20	0.7126 (238 1.0000 0.9463 0.7137 0.4766)	0.7119	0.7111 (238 1.0000 0.9436 0.7131 0.4771)	0.0003	11.29 seg
21	0.7128 (243 0.9996 0.9489 0.7123 0.4770)	0.7122	0.7091 (241 0.9996 0.9412 0.7126 0.4773)	0.0003	10.41 seg
22	0.7128 (243 0.9996 0.9489 0.7123 0.4770)	0.7124	0.7123 (238 0.9996 0.9459 0.7137 0.4768)	0.0001	10.91 seg
23	0.7133 (240 0.9996 0.9486 0.7129 0.4769)	0.7124	0.7110 (237 0.9996 0.9428 0.7137 0.4781)	0.0002	10.24 seg
24	0.7139 (240 0.9996 0.9497 0.7122 0.4772)	0.7128	0.7113 (241 0.9996 0.9453 0.7131 0.4769)	0.0002	10.22 seg
25	0.7139 (240 0.9996 0.9497 0.7122 0.4772)	0.7131	0.7128 (243 0.9996 0.9489 0.7123 0.4770)	0.0003	10.15 seg
26	0.7139 (240 0.9996 0.9497 0.7122 0.4772)	0.7135	0.7128 (243 0.9996 0.9489 0.7123 0.4770)	0.0003	10.39 seg
27	0.7139 (240 0.9996 0.9497 0.7122 0.4772)	0.7139	0.7133 (240 0.9996 0.9486 0.7129 0.4769)	0.0000	10.64 seg
28	0.7139 (240 0.9996 0.9497 0.7122 0.4772)	0.7139	0.7139 (240 0.9996 0.9497 0.7122 0.4772)	0.0000	10.04 seg
29	0.7143 (240 0.9996 0.9503 0.7129 0.4768)	0.7143	0.7143 (240 0.9996 0.9503 0.7129 0.4768)	0.0000	10.83 seg
30	0.7143 (240 0.9996 0.9503 0.7129 0.4768)	0.7143	0.7143 (240 0.9996 0.9503 0.7129 0.4768)	0.0000	11.21 seg
31	0.7139 (242 0.9993 0.9505 0.7129 0.4768)	0.7139	0.7139 (242 0.9993 0.9505 0.7129 0.4768)	0.0000	13.99 seg
32	0.7139 (242 0.9993 0.9505 0.7129 0.4768)	0.7139	0.7139 (242 0.9993 0.9505 0.7129 0.4768)	0.0000	10.39 seg
33	0.7139 (242 0.9993 0.9505 0.7129 0.4768)	0.7139	0.7139 (242 0.9993 0.9505 0.7129 0.4768)	0.0000	11.73 seg
34	0.7139 (242 0.9993 0.9505 0.7129 0.4768)	0.7139	0.7139 (242 0.9993 0.9505 0.7129 0.4768)	0.0000	11.30 seg
35	0.7139 (242 0.9993 0.9505 0.7129 0.4768)	0.7139	0.7139 (242 0.9993 0.9505 0.7129 0.4768)	0.0000	10.87 seg
36	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	10.24 seg
37	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	10.62 seg
38	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	10.53 seg
39	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	11.02 seg
40	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	11.44 seg
41	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	11.04 seg
42	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	10.55 seg
43	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	10.48 seg
44	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	10.65 seg
45	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	10.44 seg
46	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	10.05 seg
47	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	12.15 seg
48	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	14.85 seg
49	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	10.25 seg
50	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.7140	0.7140 (242 0.9993 0.9507 0.7126 0.4769)	0.0000	10.01 seg

Tabla 4.2: Evolución torneo varios_puntos reemplazo

Al observar la columna correspondiente a la desviación estándar, se aprecia una disminución progresiva a lo largo de las generaciones hasta alcanzar el valor cero. Este comportamiento indica una pérdida gradual de diversidad genética en la población, lo que sugiere que el algoritmo ha convergido prematuramente hacia un óptimo local.

En el fichero *resumen.csv* se indica que la mejor solución fue encontrada en la generación 30. Sin embargo, al examinar los datos detallados, se observa que ya en la generación 29 se había alcanzado el mismo valor de fitness. La razón por la que se registra la generación 30 como la mejor se debe a que, ante soluciones con el mismo fitness, se conserva la más reciente. Tras esta convergencia, el valor del fitness comienza a deteriorarse ligeramente en generaciones posteriores. Este empeoramiento se puede atribuir al efecto de las mutaciones, ya que al haberse perdido la diversidad genética (desviación estándar nula), el operador

de cruce deja de ser efectivo en la generación de nuevas soluciones.

Aunque la mayoría de los objetivos de la solución final presentan valores aceptables, el factor determinante que compromete su viabilidad es el objetivo de equilibrio de alumnos entre los distintos grupos. Un análisis más detallado del fichero de distribución de alumnos revelaría un notable desequilibrio en varias asignaturas, con asignaciones que superan claramente la capacidad real de las aulas. Este aspecto es especialmente crítico, ya que convierte la solución en inviable desde el punto de vista logístico, independientemente del rendimiento alcanzado en los demás objetivos.

Siguiendo con el análisis del fichero *resumen.csv*, se observa que la mejor combinación corresponde al uso de los mismos métodos de selección y sustitución que en la combinación previamente analizada, pero empleando el cruce uniforme. Esta mejora resalta la relevancia del operador de cruce en el rendimiento del algoritmo, especialmente cuando se combina con selección por torneo y el método de reemplazo. En contraste, la tercera combinación posible con esa misma selección y sustitución —pero con el cruce restante— obtiene una de las peores posiciones en el ranking de resultados, presentando además una evolución del fitness similar a la combinación ya analizada, aunque con un rendimiento claramente inferior.

En estos casos, el uso de un torneo con tamaño reducido genera una presión selectiva muy baja, lo que limita la capacidad del algoritmo para favorecer a los mejores individuos. Al combinarse con el método de reemplazo, que no garantiza la preservación de los mejores individuos entre generaciones, el papel del cruce se vuelve crítico. El análisis confirma que los operadores de cruce de un punto y de varios puntos no son adecuados para este problema, ya que no permiten una exploración eficaz del espacio de soluciones.

Otro caso destacado es el de la combinación que utiliza el mismo cruce (uniforme) y sustitución (reemplazo) que la mejor combinación, pero cambia la selección por una basada en fitness. Esta variante ofrece el peor resultado de todas las evaluadas. Aunque el cruce uniforme facilita una buena exploración del espacio de soluciones, la selección basada en fitness tiende a favorecer con demasiada frecuencia a los individuos con mejor rendimiento en cada generación. Esto puede parecer una estrategia efectiva, pero en los algoritmos genéticos no siempre seleccionar a los mejores individuos conduce a las mejores soluciones. En muchos casos, el proceso de cruce entre individuos con fitness medio o incluso bajo puede generar combinaciones más prometedoras. Además, al utilizar el método de reemplazo, no se preservan necesariamente los mejores individuos de generaciones anteriores, lo que incrementa el riesgo de pérdida de calidad en la evolución.

Los resultados muestran que en la mayoría de combinaciones analizadas no se ha producido un estancamiento temprano, por el contrario, el algoritmo ha continuado mejorando su rendimiento hasta las últimas generaciones. Dado que existen tres opciones por cada uno de los métodos de selección, cruce y sustitución, el número total de combinaciones posibles es $3^3 = 27$. Si además se consideran seis hiperparámetros y se evaluarán tres opciones para cada uno, se obtendrían $3^6 = 729$ combinaciones adicionales. La evaluación exhaustiva de todas estas combinaciones implicaría un coste computacional elevado.

Sin embargo, dado que las tres mejores combinaciones presentan una diferencia notable respecto al resto y comparten los mismos métodos de selección y cruce, se ha decidido reducir el espacio de búsqueda y centrarse exclusivamente en dichas combinaciones para la

fase posterior de optimización de hiperparámetros.

4.4. Evaluación de hiperparámetros del algoritmo genético

Tras la primera fase de experimentación, se ha observado que las mejores combinaciones de métodos corresponden a la pareja formada por selección por torneo y cruce uniforme, independientemente del método de sustitución empleado. Esta combinación ha ofrecido resultados consistentemente superiores, por lo que se utilizará como base para esta segunda fase.

En esta etapa se evaluarán de forma sistemática los distintos métodos de sustitución en combinación con la pareja fija de selección-cruce (torneo-uniforme). Además, se explorarán diferentes configuraciones de los hiperparámetros clave del algoritmo (como el tamaño de población, número de generaciones, tamaño del torneo, y probabilidades de mutación), con el objetivo de identificar una configuración óptima que maximice la calidad de las soluciones para el problema planteado.

Al igual que en la fase anterior, se ha desarrollado un script automatizado de experimentación que está disponible en el repositorio A.1, que permite ejecutar múltiples combinaciones de forma concurrente. Esto no solo optimiza el uso de los recursos computacionales disponibles, sino que también permite reducir significativamente el tiempo necesario para completar el conjunto de experimentos.

Los ficheros que se generan con este segundo script de experimentación, cuyo objetivo es evaluar distintas combinaciones de hiperparámetros manteniendo fija la pareja selección-cruce (torneo-uniforme). Estos ficheros permiten analizar de forma tanto global como individual el comportamiento del algoritmo.

■ *resumen.csv*:

En este fichero se recoge información agregada de todas las combinaciones probadas. Las columnas que contiene, de izquierda a derecha, son:

- **(Población, Generaciones, Mutación_Elemento, Mutación_Teoría, Mutación_Práctica, Torneo)**: Representa la configuración de hiperparámetros utilizada.
- **(Selección, Cruce, Sustitución)**: Indica la combinación de métodos utilizada.
- **Mejor Generación**: Generación en la que se encontró el individuo con mejor fitness.
- **Mejor Fitness**: Valor del fitness y los objetivos del mejor individuo, en el mismo orden que se ha explicado antes.
- **Tiempo Total**: Tiempo total que ha tardado en ejecutarse la combinación.

■ *Carpetas por combinación de hiperparámetros*:

Para organizar los resultados, se genera una carpeta por cada combinación de hiperparámetros evaluada. Dentro de cada carpeta se encuentra un fichero con nombre *selección-cruce-sustitución.csv*, con el mismo formato que en la fase anterior. Este fichero contiene información detallada por generación:

-
- **Generación:** Número de generación.
 - **Mejor_Fitness:** Valor del fitness y los objetivos del mejor individuo de la generación, en el mismo orden que se ha explicado antes.
 - **Promedio_Fitness:** Fitness medio de toda la población en esa generación.
 - **Peor_Fitness:** Fitness y objetivos del peor individuo de la generación, en el mismo orden que se ha explicado antes.
 - **Desviación_Estándar:** Variabilidad del fitness en la generación.
 - **Tiempo_Generación:** Tiempo de ejecución de esa generación.

Con estos ficheros, al igual que en la fase anterior, se pueden identificar rápidamente las configuraciones más prometedoras y analizar en detalle la evolución de una combinación concreta a lo largo de las generaciones.

Como se ha explicado previamente, en esta fase los métodos de selección y cruce se mantienen fijos, empleando la combinación identificada como más prometedora en la fase anterior. A partir de esta configuración base, se evaluarán los distintos métodos de sustitución junto con múltiples combinaciones de hiperparámetros, con el objetivo de encontrar la configuración más adecuada para el problema planteado.

Los valores considerados para cada hiperparámetro durante esta fase de experimentación son los siguientes:

- *población* = [100, 300, 500]
- *generaciones* = [50, 80, 100]
- *mutacion_elemento* = [0.00009]
- *mutacion_teoría* = [0.0005, 0.005, 0.05]
- *mutacion_práctica* = [0.0005, 0.005, 0.05]
- *t_torneo* = [10, 20, 50]

En la primera fase se evaluaron un total de 18 combinaciones, resultado de aplicar las tres variantes implementadas para cada uno de los métodos de selección, cruce y sustitución. En esta segunda fase, al mantener fijos los métodos de selección y cruce, el número de combinaciones posibles depende de los seis hiperparámetros restantes junto con los tres métodos de sustitución. Si se probaran tres valores por cada uno de ellos, el número total de combinaciones ascendería a $3^6 \times 3 = 2187$, lo cual supondría una carga computacional muy elevada.

Para reducir significativamente el número de ejecuciones sin perder representatividad en los resultados, se ha optado por fijar un único valor para el hiperparámetro *mutacion_elemento*, dado que la variabilidad ya está suficientemente cubierta mediante los valores asignados a *mutacion_teoría* y *mutacion_práctica*. Con esta decisión, el número total de combinaciones se reduce a $3^5 \times 3 = 729$, lo que permite una exploración amplia y eficiente del espacio de configuraciones con un coste computacional razonable.

Después de ejecutar el script correspondiente a la segunda fase de experimentación, se ha generado el archivo *resumen_fitness.csv*, el cual recoge los resultados obtenidos para

cada combinación de selección, cruce y método de sustitución evaluada. A partir de este fichero, es posible analizar comparativamente el rendimiento de las distintas configuraciones en función de su capacidad de optimización.

En la Tabla 4.3 se muestran los primeros 100 registros, ordenados de forma descendente según su valor de *Mejor Fitness*, lo que permite identificar rápidamente qué combinaciones han ofrecido los mejores resultados en términos de calidad de solución.

Hiperparametros	Selección Cruce Sustitucion	Mejor Generación	Mejor Fitness
(500 100 9e-05 0.0005 0.0005 10)	(torneo uniforme reemplazo)	77	0.7730 (144 0.9999 0.9768 0.7375 0.6227)
(500 80 9e-05 0.0005 0.005 10)	(torneo uniforme reemplazo)	66	0.7718 (149 0.9990 0.9772 0.7309 0.6167)
(500 80 9e-05 0.0005 0.0005 10)	(torneo uniforme reemplazo)	73	0.7704 (158 0.9997 0.9764 0.7375 0.6180)
(500 100 9e-05 0.0005 0.005 10)	(torneo uniforme reemplazo)	76	0.7692 (161 0.9993 0.9781 0.7332 0.6083)
(500 100 9e-05 0.005 0.0005 10)	(torneo uniforme reemplazo)	73	0.7677 (169 0.9958 0.9730 0.7354 0.6299)
(500 80 9e-05 0.005 0.005 10)	(torneo uniforme reemplazo)	68	0.7658 (160 0.9962 0.9745 0.7426 0.5932)
(500 100 9e-05 0.0005 0.005 10)	(torneo uniforme elitismo)	71	0.7650 (170 0.9993 0.9755 0.7429 0.6001)
(500 80 9e-05 0.005 0.0005 10)	(torneo uniforme reemplazo)	71	0.7644 (163 0.9954 0.9745 0.7347 0.5943)
(500 80 9e-05 0.0005 0.05 10)	(torneo uniforme reemplazo)	76	0.7639 (164 0.9997 0.9702 0.7305 0.6100)
(500 80 9e-05 0.005 0.0005 10)	(torneo uniforme elitismo)	73	0.7637 (178 0.9943 0.9794 0.7363 0.5930)
(500 80 9e-05 0.0005 0.0005 10)	(torneo uniforme elitismo)	66	0.7634 (165 0.9997 0.9756 0.7359 0.5898)
(300 80 9e-05 0.0005 0.005 10)	(torneo uniforme reemplazo)	61	0.7634 (164 0.9997 0.9734 0.7255 0.5930)
(500 50 9e-05 0.0005 0.0005 10)	(torneo uniforme reemplazo)	50	0.7625 (142 0.9998 0.9565 0.7362 0.6220)
(500 80 9e-05 0.0005 0.05 10)	(torneo uniforme elitismo)	58	0.7621 (164 0.9998 0.9703 0.7286 0.6026)
(300 100 9e-05 0.0005 0.0005 10)	(torneo uniforme reemplazo)	62	0.7619 (150 1.0000 0.9765 0.7309 0.5557)
(500 100 9e-05 0.0005 0.05 10)	(torneo uniforme reemplazo)	82	0.7618 (174 0.9991 0.9719 0.7281 0.6129)
(500 100 9e-05 0.005 0.005 10)	(torneo uniforme elitismo)	65	0.7612 (169 0.9964 0.9718 0.7287 0.5941)
(500 100 9e-05 0.005 0.005 10)	(torneo uniforme reemplazo)	71	0.7610 (193 0.9938 0.9772 0.7379 0.6014)
(500 50 9e-05 0.0005 0.05 10)	(torneo uniforme elitismo)	50	0.7600 (160 0.9996 0.9597 0.7393 0.6076)
(500 50 9e-05 0.0005 0.005 10)	(torneo uniforme reemplazo)	50	0.7597 (158 0.9997 0.9628 0.7364 0.5945)
(500 100 9e-05 0.005 0.05 10)	(torneo uniforme reemplazo)	69	0.7596 (189 0.9962 0.9715 0.7343 0.6084)
(300 100 9e-05 0.0005 0.005 10)	(torneo uniforme reemplazo)	53	0.7595 (164 1.0000 0.9773 0.7287 0.5537)
(500 50 9e-05 0.0005 0.0005 10)	(torneo uniforme elitismo)	46	0.7588 (155 0.9995 0.9675 0.7292 0.5681)
(500 80 9e-05 0.005 0.05 10)	(torneo uniforme reemplazo)	59	0.7588 (169 0.9978 0.9633 0.7286 0.6147)
(500 80 9e-05 0.0005 0.005 10)	(torneo uniforme elitismo)	59	0.7586 (179 0.9993 0.9775 0.7292 0.5721)
(500 50 9e-05 0.0005 0.005 10)	(torneo uniforme elitismo)	46	0.7583 (157 1.0000 0.9626 0.7378 0.5896)
(500 100 9e-05 0.005 0.0005 10)	(torneo uniforme elitismo)	64	0.7581 (178 0.9977 0.9756 0.7345 0.5771)
(500 80 9e-05 0.005 0.005 10)	(torneo uniforme elitismo)	62	0.7574 (180 0.9967 0.9701 0.7373 0.5961)
(500 50 9e-05 0.005 0.005 10)	(torneo uniforme reemplazo)	50	0.7567 (143 0.9994 0.9540 0.7320 0.5994)
(500 50 9e-05 0.005 0.005 20)	(torneo uniforme reemplazo)	49	0.7564 (172 0.9960 0.9743 0.7224 0.5625)
(500 80 9e-05 0.0005 0.05 20)	(torneo uniforme reemplazo)	45	0.7564 (173 0.9993 0.9681 0.7322 0.5765)
(300 50 9e-05 0.0005 0.0005 10)	(torneo uniforme reemplazo)	50	0.7560 (190 0.9992 0.9762 0.7282 0.5698)
(300 80 9e-05 0.0005 0.0005 10)	(torneo uniforme reemplazo)	63	0.7560 (176 1.0000 0.9713 0.7309 0.5710)
(300 100 9e-05 0.005 0.0005 10)	(torneo uniforme elitismo)	60	0.7560 (173 0.9967 0.9704 0.7285 0.5735)
(500 100 9e-05 0.0005 0.05 10)	(torneo uniforme elitismo)	60	0.7559 (176 0.9995 0.9691 0.7263 0.5964)
(500 50 9e-05 0.005 0.0005 20)	(torneo uniforme elitismo)	44	0.7558 (197 0.9986 0.9766 0.7305 0.5716)
(500 50 9e-05 0.0005 0.0005 20)	(torneo uniforme reemplazo)	45	0.7557 (182 0.9992 0.9769 0.7187 0.5662)
(500 50 9e-05 0.005 0.0005 20)	(torneo uniforme reemplazo)	44	0.7554 (187 0.9980 0.9755 0.7336 0.5639)
(300 50 9e-05 0.0005 0.05 10)	(torneo uniforme reemplazo)	50	0.7553 (178 1.0000 0.9699 0.7333 0.5749)
(300 100 9e-05 0.005 0.0005 10)	(torneo uniforme reemplazo)	63	0.7550 (187 0.9977 0.9740 0.7350 0.5695)
(500 100 9e-05 0.0005 0.0005 10)	(torneo uniforme elitismo)	70	0.7549 (173 0.9997 0.9756 0.7333 0.5450)
(300 80 9e-05 0.0005 0.05 10)	(torneo uniforme reemplazo)	63	0.7548 (186 0.9993 0.9695 0.7222 0.5820)
(300 50 9e-05 0.0005 0.005 10)	(torneo uniforme elitismo)	50	0.7548 (181 1.0000 0.9689 0.7337 0.5680)
(500 50 9e-05 0.005 0.005 10)	(torneo uniforme elitismo)	47	0.7548 (162 0.9978 0.9581 0.7361 0.5959)
(500 50 9e-05 0.0005 0.05 10)	(torneo uniforme reemplazo)	50	0.7542 (166 0.9993 0.9577 0.7333 0.5940)
(300 100 9e-05 0.0005 0.05 10)	(torneo uniforme reemplazo)	58	0.7539 (175 0.9997 0.9687 0.7305 0.5651)
(500 80 9e-05 0.005 0.05 10)	(torneo uniforme elitismo)	62	0.7538 (169 0.9964 0.9652 0.7247 0.5800)
(500 50 9e-05 0.005 0.0005 10)	(torneo uniforme reemplazo)	50	0.7537 (159 0.9961 0.9505 0.7378 0.6102)
(300 50 9e-05 0.005 0.005 10)	(torneo uniforme reemplazo)	50	0.7536 (186 0.9978 0.9722 0.7308 0.5767)
(500 50 9e-05 0.0005 0.0005 10)	(torneo uniforme truncamiento)	50	0.7536 (164 0.9996 0.9582 0.7256 0.5867)

Tabla 4.3: Resultados segundo script de experimentacion

Analizando los resultados obtenidos en esta fase de experimentación, se observa que las mejores combinaciones utilizan el método de sustitución por reemplazo. La primera combinación que emplea elitismo aparece en la séptima posición del ranking, mientras que la primera con truncamiento no aparece hasta la posición número 100. Estos resultados refuerzan la idea de que, para este problema en particular, los métodos que favorecen una alta presión selectiva tienden a explorar menos el espacio de búsqueda, lo cual limita la capacidad del algoritmo para alcanzar soluciones de calidad.

En la fase anterior, el método de reemplazo ya se posicionó como el más eficaz, mientras que truncamiento, a pesar de estar en tercera posición, mostró diferencias de fitness significativas respecto a las mejores combinaciones. En esta segunda fase, el comportamiento se confirma, el reemplazo permite una renovación completa de la población sin priorizar exclusivamente a los individuos con mayor fitness. Esto favorece la conservación de diversidad genética, un factor clave para alcanzar buenas soluciones, ya que estas no siempre surgen de los mejores individuos, sino de otros con potencial genético complementario.

En lo que respecta a las combinaciones de hiperparámetros, se identifica una tendencia clara entre las configuraciones más destacadas. La mayoría utiliza una población de 500 individuos, el valor más alto considerado, lo cual incrementa la diversidad de la población. También se observa un uso mayoritario de un tamaño de torneo igual a 10, el valor más bajo, lo que reduce la presión selectiva y permite que un mayor número de individuos tenga opciones de contribuir a la siguiente generación.

En cuanto a las probabilidades de mutación, las configuraciones más efectivas emplean principalmente los dos valores más bajos tanto en la mutación de teoría como en la de práctica. Esto sugiere que una intensidad de mutación baja favorece una evolución más estable, en contraposición a mutaciones más agresivas que pueden romper estructuras genéticas útiles.

Respecto al número de generaciones, la mayoría de las mejores combinaciones utilizan 100 u 80 generaciones. Sin embargo, al revisar en qué generación se obtuvo el mejor fitness, se observa que en la mayoría de los casos no se alcanzó el máximo. Esto indica que el algoritmo tiende a converger antes, por lo que un número elevado de generaciones no siempre es necesario. Por ejemplo, la combinación en primera posición usó 100 generaciones, mientras que la misma configuración con 80 generaciones aparece en tercera posición, habiendo convergido en la generación 73. Esta diferencia puede atribuirse a la variabilidad introducida en la generación inicial, lo que sugiere que esta tercera combinación podría considerarse la más eficiente para este problema, al ofrecer un resultado competitivo con menor coste computacional.

4.5. Evaluación del algoritmo NSGA-III

Esta será la última fase del proceso de experimentación. En las fases anteriores se evaluaron todas las combinaciones posibles del algoritmo genético clásico, identificando las configuraciones de métodos e hiperparámetros más efectivas. En esta última fase se aplicará todo lo aprendido para estudiar el comportamiento del algoritmo NSGA-III desarrollado en este trabajo.

A diferencia del algoritmo clásico, NSGA-III dispone de un único método para la selección y sustitución de individuos. Basándose en los resultados obtenidos anteriormente, se utilizará el método de cruce *uniforme*, que ha demostrado ofrecer los mejores resultados en las fases previas.

El resto de hiperparámetros utilizados serán los mismos que en la fase anterior, con la inclusión de un nuevo parámetro propio de NSGA-III, el número de particiones, que representa el parámetro H que se usa en la fórmula para determinar el número de puntos de referencia generados que se explica en 3.3.2. A continuación se detallan los valores considerados:

- *población* = [100, 300, 500]
- *generaciones* = [50, 80, 100]
- *mutacion_elemento* = [0.00009]
- *mutacion_teoría* = [0.0005, 0.005, 0.05]
- *mutacion_práctica* = [0.0005, 0.005, 0.05]
- *t_torneo* = [10, 20, 50]
- *particiones* = [6, 7, 8]

Al igual que en las fases anteriores, se ha implementado un script de experimentación que ejecuta de forma concurrente todas las combinaciones posibles de estos hiperparámetros. El script se encuentra disponible en el repositorio referenciado en el Anexo A.1.

El script generará diversos ficheros que facilitarán el análisis posterior de los resultados. Estos archivos permiten evaluar tanto la diversidad del frente de Pareto como la calidad de las soluciones encontradas.

- *resumen.xlsx*: Este archivo recopila los resultados globales de todas las combinaciones de hiperparámetros evaluadas. Contiene dos hojas:
 - **Resumen**: Cada fila representa una combinación distinta de hiperparámetros. Las columnas son:
 - **Población**: Tamaño de la población utilizada.
 - **Generaciones**: Número total de generaciones ejecutadas.
 - **m_elemento**: Tasa de mutación para los elementos.
 - **m_teoría**: Tasa de mutación aplicada a la parte teórica.
 - **m_práctica**: Tasa de mutación aplicada a la parte práctica.
 - **t_torneo**: Tamaño del torneo en la selección.
 - **Particiones**: Número de particiones empleadas para generar los puntos de referencia.
 - **Número de elementos**: Número de individuos distintos en el frente de Pareto final.
 - **Tiempo**: Tiempo que ha tardado la combinación.

El número de elementos permite evaluar la diversidad de las soluciones obtenidas. Por ejemplo, si una combinación ha generado un frente de Pareto formado por soluciones idénticas, se registrará como un único elemento, evidenciando pérdida de diversidad. Por el contrario, combinaciones con múltiples soluciones únicas indican una mayor exploración del espacio de búsqueda.

- **Datos:** Presenta el detalle de los individuos únicos del frente de Pareto final para cada combinación, organizados en bloques con el siguiente formato:
 - **Fila 1:** Cabecera con los nombres de los hiperparámetros usados.
 - **Fila 2:** Valores concretos para esa configuración.
 - **Fila 3:** Indicador del número de soluciones únicas (ej. "Elementos: 5").
 - **Fila 4:** Cabecera con las métricas de evaluación:
 - **Solapes**
 - **Cohesión Teoría**
 - **Equilibrio**
 - **Cohesión Prácticas**
 - **Preferencias**
 - **Filas siguientes:** Métricas de cada solución única.
- **Carpetas por combinación de hiperparámetros:** Por cada combinación específica se genera una carpeta con nombre basado en los valores utilizados. Dentro de cada carpeta se encuentra un archivo:
 - **resumen.xlsx:** Este archivo tiene el mismo formato que el anterior, pero cada bloque corresponde a una generación concreta dentro de la combinación. La estructura es:
 - **Resumen:**
 - **Generación:** Número de generación.
 - **Número de elementos:** Cantidad de soluciones únicas del frente en esa generación.
 - **Tiempo:** Tiempo que ha tardado la combinación.
 - **Datos:** Organizada por bloques, uno por generación. Cada bloque contiene:
 - **Fila 1:** Título de la generación.
 - **Fila 2:** Tiempo que ha tardado cada generación.
 - **Fila 3:** Número de elementos únicos (ej. "Elementos: 7").
 - **Fila 4:** Cabecera con las métricas:
 - **Solapes**
 - **Cohesión Teoría**
 - **Equilibrio**
 - **Cohesión Prácticas**
 - **Preferencias**
 - **Filas siguientes:** Métricas correspondientes a cada solución única del frente de la generación.

Este archivo permite analizar la evolución de la diversidad y la calidad del frente de Pareto a lo largo del tiempo, detectando posibles convergencias prematuras o pérdidas de diversidad en determinadas fases de la ejecución.

Con todos estos ficheros, se puede realizar un análisis exhaustivo sobre el comportamiento del algoritmo NSGA-III en términos de diversidad, calidad de las soluciones, y estabilidad del proceso evolutivo en función de los diferentes valores de los hiperparámetros considerados.

Población	Generaciones	m_elemento	m_teoría	m_práctica	t_torneo	particiones	Nº Elementos	Tiempo
100	50	0.00009	0.0005	0.0005	10	6	1	4.55 min
100	50	0.00009	0.0005	0.0005	10	7	6	6.00 min
100	50	0.00009	0.0005	0.0005	10	8	1	6.17 min
100	50	0.00009	0.0005	0.0005	20	6	1	3.23 min
100	50	0.00009	0.0005	0.0005	20	7	1	2.45 min
100	50	0.00009	0.0005	0.0005	20	8	1	3.13 min
100	50	0.00009	0.0005	0.0005	50	6	1	1.50 min
100	50	0.00009	0.0005	0.0005	50	7	1	1.43 min
100	50	0.00009	0.0005	0.0005	50	8	1	1.47 min
100	50	0.00009	0.0005	0.005	10	6	1	4.13 min

Tabla 4.4: Primeras 10 combinaciones del tercer script de experimentación

5. Desarrollo de la aplicación web

5.1. Introducción

Una vez completado el desarrollo de los dos algoritmos genéticos y tras realizar un análisis exhaustivo de sus hiperparámetros con el objetivo de identificar las configuraciones más eficientes, se ha considerado oportuno implementar una aplicación web que permita utilizar estos algoritmos de manera sencilla y accesible.

Esta aplicación no ha tenido un estudio previo de requisitos ni una necesidad funcional concreta, sino que se plantea como una herramienta práctica que facilite la ejecución y visualización de los resultados obtenidos por ambos algoritmos. De este modo, se busca acercar la solución al usuario final, eliminando la necesidad de interactuar directamente con el código o scripts de experimentación.

5.2. Diseño de la aplicación

El objetivo principal de esta aplicación web es proporcionar a los usuarios una interfaz sencilla e intuitiva que les permita ejecutar los algoritmos genéticos desarrollados. A través de esta interfaz, el usuario podrá seleccionar qué algoritmo desea ejecutar, adjuntar los ficheros de entrada necesarios y definir la configuración de los hiperparámetros de forma personalizada.

Además de la configuración manual, también se ofrece la posibilidad de utilizar la combinación de hiperparámetros óptima que ha sido determinada previamente en el capítulo de experimentación. Esto permite al usuario beneficiarse directamente del análisis realizado sin necesidad de profundizar en la configuración técnica.

Una vez realizada la ejecución del algoritmo, la aplicación mostrará la evolución obtenida a lo largo de las generaciones, permitiendo consultar los valores alcanzados por los objetivos en las mejores soluciones. Asimismo, se indicará la ruta en la que se han generado los ficheros de salida, los cuales incluyen las configuraciones de matrículas correspondientes a las soluciones finales.

5.3. Flask

Dado que la implementación de ambos algoritmos ha sido realizada íntegramente en *Python*, y considerando que la aplicación web se plantea como una capa de interacción directa con esta lógica, se ha optado por utilizar un framework web basado en el mismo lenguaje. Esta elección garantiza una integración fluida entre la interfaz y el núcleo computacional del sistema.

El framework seleccionado ha sido Flask, un microframework ligero y flexible que permite desarrollar aplicaciones web de forma sencilla y rápida. Flask ofrece una arquitectura minimalista que facilita la incorporación de rutas, plantillas HTML y gestión de formularios, sin necesidad de estructuras complejas. Gracias a su simplicidad y compatibilidad con bibliotecas de terceros, resulta especialmente adecuado para proyectos académicos o de carácter experimental como este, donde se prioriza la funcionalidad frente a una infraestructura compleja.

Para más información sobre Flask, se puede consultar la documentación oficial[Flask Project, 2025] o el artículo introductorio en GeeksforGeeks[GeeksforGeeks, 2025].

5.4. Arquitectura del proyecto

En esta sección se describe la estructura del proyecto y el funcionamiento de los distintos archivos que lo componen, los cuales pueden verse en la Figura 5.1. A continuación, se explica el propósito de cada uno de ellos:

- **app.py**: Archivo principal desde el cual se desarrolla la aplicación web. Aquí se reciben las peticiones de los usuarios, se redirigen a la lógica correspondiente y se devuelve una respuesta adecuada a la interfaz.
- **Carpeta model**: Contiene los archivos relacionados con la implementación de los algoritmos genéticos. En particular:
 - `alumno.py`, `matricula.py` y `solucion.py`: Estos archivos definen la estructura de los individuos en la población. Un individuo (o solución) está compuesto por una lista de alumnos, y cada alumno tiene una lista de matrículas. Esta relación jerárquica se representa gráficamente en la Figura 3.1.
 - `genetico.py` y `nsga3.py`: Implementan la lógica completa de los algoritmos genéticos desarrollados: el algoritmo clásico y NSGA-III respectivamente.
- **Carpeta utils**: Contiene archivos de utilidades empleadas en el desarrollo de los algoritmos:
 - `export.py`: Incluye funciones relacionadas con la exportación de resultados y generación de ficheros de salida.
 - `utils.py`: Agrupa diversas funciones de ayuda utilizadas a lo largo del código.

- **Carpeta static:** Contiene los recursos estáticos de la aplicación web. En particular, el archivo `style.css` que define los estilos visuales utilizados en las distintas pantallas.
- **Carpeta templates:** Incluye todas las plantillas HTML que conforman la interfaz web. Aquí se encuentran las vistas que permiten al usuario subir los ficheros de entrada, seleccionar configuraciones y visualizar las soluciones obtenidas tras la ejecución de los algoritmos.

```
C:
├─ app.py
├─ model
│  ├─ alumno.py
│  ├─ genetico.py
│  ├─ matricula.py
│  ├─ nsga3.py
│  └─ solucion.py
├─ utils
│  ├─ export.py
│  └─ utils.py
├─ static
│  └─ style.css
└─ templates
   ├─ index.html
   ├─ loading.html
   ├─ result_genetico.html
   └─ result_nsga3.html
```

Figura 5.1: Estructura del proyecto

5.5. Aplicación web

Una vez explicada la estructura del proyecto, se procede a detallar el funcionamiento de la aplicación web desarrollada. El diseño de la interfaz gráfica es intencionadamente sencillo e intuitivo, con el objetivo de facilitar el uso de los algoritmos sin necesidad de conocimientos técnicos.

La pantalla principal presenta un formulario en el que el usuario puede interactuar con los siguientes elementos:

- **Carga de ficheros de entrada:** se proporcionan dos botones para subir los archivos necesarios para la ejecución del algoritmo: el fichero de horarios y la configuración inicial de matrículas, los cuales servirán como referencia.
- **Selección del algoritmo:** mediante un desplegable, el usuario puede elegir entre ejecutar el algoritmo genético clásico o el algoritmo NSGA-III.
- **Configuración del algoritmo:** el usuario debe seleccionar entre dos modos:

-
- **Configuración óptima:** aplica automáticamente la combinación de hiperparámetros que ha sido determinada como la mejor durante la fase de experimentación. Al seleccionar esta opción, todos los campos relacionados con los hiperparámetros quedan deshabilitados, mostrando sus valores fijados únicamente a modo informativo.
 - **Configuración manual:** permite al usuario definir libremente los valores de los hiperparámetros. Los parámetros categóricos como los métodos de selección, cruce y sustitución se presentan mediante menús desplegables. En el caso del algoritmo NSGA-III, dado que solo dispone de un método de selección y sustitución, estos campos aparecen deshabilitados con el valor correspondiente preestablecido. Los hiperparámetros numéricos deben ser introducidos directamente por el usuario.
- **Campos condicionales:** algunos campos solo se mostrarán cuando sean relevantes:
 - *Tamaño del torneo:* solo aparecerá si el método de selección elegido es por torneo.
 - *Número de particiones:* únicamente visible si se selecciona el algoritmo NSGA-III, ya que este parámetro es necesario para la generación de puntos de referencia.
 - **Ejecución del algoritmo:** una vez completado el formulario, el usuario puede iniciar el proceso pulsando el botón *Ejecutar*. El sistema comenzará la ejecución del algoritmo con los parámetros proporcionados y mostrará el resultado al finalizar.

Antes de iniciar la ejecución, se lleva a cabo una validación completa de los campos del formulario. Se comprueba que los archivos hayan sido cargados correctamente y que todos los hiperparámetros tengan valores válidos y dentro del formato requerido (por ejemplo, que sean números positivos o que no falten valores). En caso de que se detecte algún error de validación, se mostrará un mensaje informativo al usuario indicando claramente qué campo está generando el problema y qué se espera de él.

Además, si durante la ejecución del algoritmo ocurre un error inesperado (por ejemplo, por errores en los datos de entrada o excepciones internas), el usuario será redirigido automáticamente a la pantalla de inicio. En ese caso, se mostrará un mensaje genérico indicando que ha ocurrido un error durante el proceso. Para facilitar el diagnóstico, en la consola desde la que se ejecutó el servidor se mostrará información técnica más detallada sobre el error.

Algoritmo Genético - TFG

Archivo de horarios:
 No se ha seleccionado ningún archivo.

Archivo de matrículas:
 No se ha seleccionado ningún archivo.

Algoritmo:
NSGA-III

Configuración:
Manual

Selección:
Torneo

Tamaño Torneo:

Cruce:
Uniforme

Sustitución:
Reemplazo

Particiones:

Población:

Generaciones:

Mutación Elemento:

Mutación Teoría:

Mutación Práctica:

Figura 5.2: Pagina principal de la aplicación web

Tras pulsar el botón de *Ejecutar*, se inicia la ejecución del algoritmo genético seleccionado con la configuración indicada. La aplicación no muestra una vista en tiempo real de la evolución del algoritmo, en su lugar, se presentará una pantalla de carga mientras se ejecuta el proceso.

Dado que el tiempo de ejecución puede variar significativamente en función de la combinación de hiperparámetros elegida, esta pantalla de carga incluirá un botón que permitirá cancelar la ejecución en cualquier momento. En caso de cancelación, el usuario será redirigido nuevamente a la pantalla de inicio para realizar una nueva configuración o cargar otros datos.

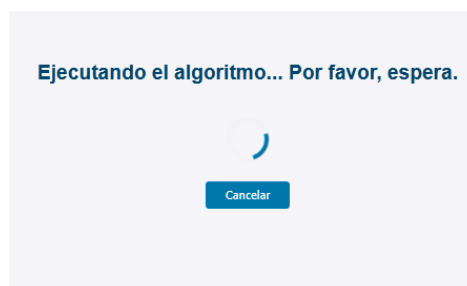


Figura 5.3: Pantalla de carga de la aplicación web

Una vez finaliza el proceso de ejecución, la aplicación carga una nueva pantalla en la que se muestran los resultados obtenidos por el algoritmo genético. Esta pantalla tiene como objetivo proporcionar al usuario una visión general de la evolución del algoritmo, los detalles de la mejor solución encontrada y la localización de los ficheros generados, que contienen la configuración final de matrículas.

La interfaz que se presenta dependerá del tipo de algoritmo ejecutado. En el caso del algoritmo genético clásico, la estructura de la pantalla será la siguiente:

- **Hiperparámetros:** Sección en la que se muestra la configuración completa de hiperparámetros utilizada en la ejecución.
- **Ruta de ficheros de salida:** Se indica la ruta donde se han almacenado los ficheros de salida generados, los cuales contienen tanto la configuración de matrículas como la distribución de alumnos (véase la sección 3.2.8).
- **Mejor solución:** Se presenta información detallada del mejor individuo encontrado durante la ejecución. Se indica en qué generación fue hallado, así como los valores alcanzados en cada uno de los objetivos.
- **Evolución del algoritmo:** Se muestra una tabla resumen con estadísticas por cada generación. Las columnas incluyen:
 - Mejor fitness
 - Fitness promedio
 - Peor fitness
 - Desviación estándar
 - Tiempo de ejecución por generación
- **Navegación:** Al final de la pantalla se incluye un botón que permite regresar a la pantalla principal para iniciar una nueva ejecución si se desea.



Figura 5.4: Parte superior de la pantalla de resultados del algoritmo clásico

Resumen por generación

Generación	Máx	Media	Min	Dev. Estándar	Tiempo (s)
1	0.6827 (294 1.0000 0.9182 0.7098 0.4598)	0.6592	0.5991 (288 1.0000 0.7630 0.7080 0.4644)	0.0108	5.65 seg
2	0.6925 (263 1.0000 0.9224 0.7039 0.4701)	0.6667	0.6324 (287 1.0000 0.8314 0.7060 0.4359)	0.0085	7.30 seg

[Volver al inicio](#)

Figura 5.5: Parte inferior de la pantalla de resultados del algoritmo clásico

En el caso del algoritmo NSGA-III, la pantalla de resultados mantiene una estructura muy similar a la del algoritmo genético clásico, aunque adaptada a las particularidades de un enfoque multiobjetivo. La estructura que se presenta es la siguiente:

- **Hiperparámetros:** Sección en la que se muestra la configuración completa de hiperparámetros utilizada durante la ejecución, tanto los fijados por el usuario como los establecidos automáticamente en el caso de haber seleccionado la configuración óptima.
- **Ruta de ficheros de salida:** Se indica la ruta del directorio donde se han almacenado los resultados. A diferencia del algoritmo clásico, en este caso se genera una carpeta

por cada individuo perteneciente al frente de Pareto final. Dentro de cada una de estas carpetas se encuentran los dos ficheros de salida habituales, que contienen la configuración de matrículas correspondiente a esa solución y su distribución de alumnos (véase la sección 3.2.8).

- **Frente de Pareto:** Se presenta una tabla con los individuos que forman parte del frente de Pareto obtenido en la última generación. Para cada uno de ellos se muestran los valores alcanzados en cada uno de los objetivos. Dado que no existe un único mejor individuo, esta sección permite comparar las distintas soluciones no dominadas que ha encontrado el algoritmo.
- **Evolución del algoritmo:** Se ofrece una tabla resumen que muestra cómo ha evolucionado el frente de Pareto a lo largo de las generaciones. Las columnas de la tabla incluyen:
 - Número de individuos en el primer frente
 - Tiempo de ejecución de la generación
- **Navegación:** Al final de la pantalla se incluye un botón que permite regresar a la pantalla principal para realizar una nueva ejecución o modificar la configuración actual.



Figura 5.6: Parte superior de la pantalla de resultados NSGA-III

Frente de Pareto

Estas son las soluciones no dominadas del frente final:

#	Solapes	Cohesión teoría	Equilibrio grupos	Cohesión prácticas	Preferencias
1	270	1.0000	0.8796	0.7077	0.4847
2	279	1.0000	0.8812	0.6967	0.4810
3	290	1.0000	0.8671	0.7117	0.4442
4	254	1.0000	0.8640	0.7123	0.4684

Resumen por generación

Generación	Tamaño del frente	Tiempo
1	4	0.47 seg
2	4	2.10 seg

[Volver al inicio](#)

Figura 5.7: Parte inferior de la pantalla de resultados NSGA-III

5.6. Conclusiones y futuras mejoras

Una vez detallado el desarrollo y funcionamiento de la aplicación web, se puede concluir que se ha cumplido satisfactoriamente el objetivo principal de esta sección, proporcionar una interfaz gráfica sencilla e intuitiva que permita al usuario ejecutar los algoritmos genéticos implementados para la asignación de grupos en la matrícula de los alumnos.

Gracias a esta herramienta, se facilita el uso de los algoritmos sin necesidad de conocimientos técnicos sobre su implementación, permitiendo cargar los datos de entrada, seleccionar el tipo de algoritmo y configurar los hiperparámetros, todo ello desde un entorno accesible.

Sin embargo, se identifican posibles mejoras que podrían implementarse en el futuro para ampliar la funcionalidad y mejorar la experiencia del usuario. Entre ellas, destacan las siguientes:

- **Implementación como servicio web accesible online:** Actualmente, la aplicación se ejecuta de forma local. Desplegarla en un servidor permitiría a múltiples usuarios acceder desde cualquier dispositivo sin necesidad de instalación previa.
- **Visualización directa de la configuración de matrículas:** Una posible mejora sería mostrar directamente en la interfaz web los resultados de la asignación de matrículas, sin necesidad de acceder al fichero generado en el sistema de archivos. Esto facilitaría el análisis inmediato de la solución obtenida.
- **Mejora del sistema de seguimiento en tiempo real:** Aunque actualmente se muestra una pantalla de carga, una posible extensión sería permitir la visualización del progreso del algoritmo en tiempo real, incluyendo métricas y evolución de los objetivos generación a generación.
- **Validación avanzada de entradas:** Incluir mecanismos para verificar la validez y consistencia de los ficheros de entrada antes de la ejecución del algoritmo ayudaría a evitar errores en tiempo de ejecución.

Estas mejoras permitirían evolucionar la herramienta desde un prototipo funcional hacia una solución más robusta, accesible y completa.

6. Conclusiones

6.1. Conclusiones

A lo largo de este trabajo se ha abordado la problemática asociada al proceso de matriculación de los estudiantes, centrada principalmente en la dificultad de seleccionar grupos compatibles y equilibrados. Tras un análisis detallado del problema, se ha revisado la teoría relacionada con los algoritmos genéticos clásicos y los algoritmos evolutivos multiobjetivo, identificando sus fundamentos y características principales.

Se ha representado formalmente el problema dentro del marco de la computación evolutiva, desarrollando dos implementaciones: una basada en un algoritmo genético clásico y otra en el algoritmo multiobjetivo *NSGA-III*. Para evaluar el comportamiento y rendimiento de ambas propuestas, se ha diseñado una extensa fase de experimentación compuesta por múltiples combinaciones de métodos y valores de hiperparámetros. Esto ha permitido identificar configuraciones óptimas o prometedoras.

En el caso del algoritmo genético clásico, se ha logrado encontrar una configuración que ofrece buenos resultados, generando soluciones que pueden considerarse factibles en un entorno real. Por el contrario, aunque el algoritmo *NSGA-III* ha sido capaz de producir soluciones que cumplen satisfactoriamente la mayoría de los objetivos planteados, ha presentado dificultades en el cumplimiento del objetivo relativo al equilibrio de alumnos entre grupos. Este aspecto es especialmente crítico, dado que las limitaciones físicas de las aulas requieren una distribución de estudiantes lo más equitativa posible. Esta diferencia en el comportamiento se explica porque en el algoritmo clásico se ha utilizado una función de fitness ponderada, lo que ha permitido otorgar mayor importancia a dicho objetivo, mientras que en *NSGA-III* no se realiza dicha ponderación explícita.

Finalmente, se ha desarrollado una aplicación web que permite al usuario ejecutar cualquiera de los algoritmos implementados. La aplicación ofrece la posibilidad de seleccionar configuraciones personalizadas o emplear las que se han identificado como óptimas durante la fase de experimentación. Al finalizar la ejecución, la aplicación presenta un resumen visual del comportamiento del algoritmo y muestra la ruta a los ficheros de salida generados.

6.2. Trabajo futuro

Como posibles líneas de trabajo futuro, una primera mejora consistiría en refinar la definición formal del problema incorporando restricciones adicionales que reflejen con mayor precisión las condiciones reales del proceso de matriculación. Esta ampliación permitiría realizar nuevas pruebas más específicas y profundizar en el comportamiento del algoritmo *NSGA-III*, con el objetivo de obtener soluciones que no solo optimicen múltiples objetivos, sino que también sean factibles y aplicables en un entorno real.

También, sería interesante explorar otros algoritmos genéticos multiobjetivo más allá de *NSGA-III*. Aunque este ha sido elegido por su adecuación a problemas con más de dos objetivos, existen alternativas como *SPEA2*, *MOEA/D* o *NSGA-II* que podrían ofrecer diferentes equilibrios entre diversidad y convergencia, y cuya evaluación podría resultar beneficiosa para mejorar la calidad de las soluciones obtenidas.

En relación con la aplicación web desarrollada, se plantea como mejora futura la incorporación de funcionalidades que permitan visualizar en tiempo real la evolución del algoritmo durante su ejecución. Además, integrar la visualización de los resultados directamente en la interfaz permitiría al usuario analizar de forma más cómoda las soluciones generadas, sin necesidad de acceder manualmente a los archivos exportados.

A. Anexos

A.1. Repositorio del código y scripts

Todo el código fuente desarrollado para este trabajo, incluidos los algoritmos genéticos, los scripts de experimentación, los módulos de apoyo y la aplicación web, se encuentra disponible en el siguiente repositorio de GitHub:

`https://github.com/Mega57/TFG`

El repositorio está organizado de forma estructurada y contiene:

- Implementaciones completas del algoritmo genético clásico y del algoritmo *NSGA-III*.
- Scripts utilizados para la experimentación y análisis de hiperparámetros.
- Código de la aplicación web desarrollada con Flask.
- Ficheros de entrada y salida utilizados durante el proceso de evaluación.
- Documentación adicional en el archivo `README.md`, que explica cómo ejecutar los distintos componentes del proyecto.

Cualquier referencia en esta memoria a scripts concretos o fragmentos de código puede consultarse directamente en el repositorio.

Referencia bibliográfica

- [Das and Dennis, 1998] Das, I. and Dennis, J. E. (1998). Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657.
- [Deb, 2001] Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, New York, NY.
- [Deb and Jain, 2014] Deb, K. and Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601.
- [Escuela Superior de Ingeniería Informática de Albacete, 2024] Escuela Superior de Ingeniería Informática de Albacete (2024). Página web oficial de la escuela.
- [European Commission, 2024] European Commission (2024). Inclusive and connected higher education - bologna process.
- [Flask Project, 2025] Flask Project (2025). Flask documentation.
- [Flores et al., 2022] Flores, M., Gámez, J., and García, I. (2022). *Algoritmos Genéticos*. Departamento de Sistemas Informáticos, UCLM - Albacete. Disponible en formato PDF.
- [GeeksforGeeks, 2025] GeeksforGeeks (2025). Python | introduction to web development using flask.
- [Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Boston, MA.
- [Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.