uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Informática

2019-2023

Trabajo Fin de Grado

"Optimización de la programación de citas en estudios clínicos"

Daniel Roa Santín

Tutor

Agapito Ismael Ledezma Espino

Leganés / Septiembre 2023

RESUMEN

La presente Memoria corresponde al estudio realizado sobre la optimización de la programación de citas en estudios clínicos. El estudio revisa la situación general de la programación de citas en el entorno sanitario, y se centrará en los distintos métodos de optimización de planificaciones basándose en algoritmos genéticos multiobjetivo.

Dentro del estudio se hará hincapié en cómo se ha gestionado la programación de citas en estudios clínicos y la diferencia que hay con centros de salud u hospitales. Para estos últimos se han encontrado muchas investigaciones en torno a cómo optimizar la programación de citas o de cirugías, y servirán de base para adaptar esos algoritmos de optimización a un entorno como es el de los estudios clínicos. En lo referente a la parte técnica, se realizará un amplio análisis de los Algoritmos Genéticos, en especial en los Algoritmos Genéticos Multiobjetivo, que son los más adecuados a la optimización de planificaciones, ya que permiten optimizarlas con respecto a más de un objetivo, mejorando así la calidad de las soluciones.

Finalmente, se tomará como referencia los datos de la empresa de estudios clínicos CEVAXIN, y a partir de estos se procederá al desarrollo de dos algoritmos que permitan enfocar la planificación tanto diaria como semanal de una clínica, buscando optimizar las horas útiles de los recursos y minimizar el tiempo de espera de los pacientes.

Palabras clave: Programación de Citas; Optimización Multiobjetivo; Algoritmos Genéticos; Gestión Sanitaria; Optimización de Scheduling

ABSTRACT

This report pertains to a study conducted on the optimization of appointment scheduling in clinical trials. The study examines the general state of appointment scheduling in the healthcare setting and highlights various methods of scheduling optimization based on multi-objective genetic algorithms.

It places particular emphasis on appointment scheduling management in clinical trials and how it varies from that in health centres or hospitals. For the latter, extensive research has been conducted on the optimal scheduling of appointments or surgeries, which can serve as a foundation for modifying these optimisation algorithms to suit the clinical trials environment. On the technical side, this paper presents a comprehensive analysis of Genetic Algorithms, with a particular focus on Multiobjective Genetic Algorithms. These algorithms are particularly suitable for scheduling optimization since they enable the optimization of multiple objectives simultaneously, thereby enhancing the quality of solutions. During this investigation, technical abbreviation, GA, will refer to Genetic Algorithms.

Finally, we will utilise the clinical study data from CEVAXIN as a reference. From this, we will develop two algorithms to optimise resource utilisation and minimise patient waiting times for daily and weekly clinic planning.

Keywords: Appointment Scheduling; Multiobjective Optimisation; Genetic Algorithms; Healthcare Management; Scheduling Optimisation

DEDICATORIA

Quiero dedicar unas palabras de agradecimiento a todas aquellas personas que me han apoyado durante esta bonita etapa que llega a su fin.

En primer lugar, a mis padres y a mi hermano, los cuales siempre me han apoyado a lo largo de estos cuatro años, en los momentos más duros de pandemia sobre todo cuando tan difícil se hacía continuar estudiando, siempre me daban las ganas de seguir adelante, y sin ellos hubiera sido imposible. Os quiero.

También quería agradecer a mi pareja Marta, su apoyo incondicional y largas noches acompañando durante el desarrollo de este trabajo ha sido mi principal pulmón para culminarlo.

No podía olvidarme de mis compañeros de batallas durante esta carrera, los chicos de La Squad. Vuestra actitud y ganas de trabajar eran la gasolina que necesitaba sobre todo en estos últimos dos años, siempre acompañados de unas buenas risas.

Sin todas estas personas y mis amigos de tantos años no habría conseguido llegar hasta la meta de ser por fin Ingeniero. Gracias.

Por último, agradecer a mi tutor por su apoyo y su capacidad de compartir su conocimiento con los demás, se nota cuando un profesor ama lo que hace y así lo demuestra a sus alumnos.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

1. INTRODUCCIÓN

1.1. Motivación del trabajo

Este trabajo surge como un proyecto que busca solucionar un problema con el que se encontraba una empresa de estudios clínicos a la hora de planificar sus distintas actividades. Este es un trabajo que les llevaba mucho tiempo semanalmente, debiendo tener personas expertas en el sector encargadas diariamente de planificar el horario de las citas de cada uno de los estudios.

El desarrollo del trabajo se centra en utilizar algoritmos genéticos multiobjetivo para poder automatizar el proceso de planificación, investigando sobre si se había llegado a utilizar este sistema en centros de salud y aplicarlo al caso de una empresa de estudios clínicos.

Poder automatizar este proceso supondría un gran avance dentro de la gestión de citas en empresas de estudios clínicos, mejorando así tanto la organización y gestión de recursos por parte de la empresa como también la satisfacción general de los pacientes que acuden a colaborar con estos estudios.

1.2. Objetivos

El trabajo consistirá en desarrollar un algoritmo que permita dando como entrada una serie de recursos físicos y temporales y un conjunto de actividades y estudios a planificar, poder sacar una planificación ajustada a unos objetivos determinados. Se cogerá como referencia la estructura base de cómo funciona un estudio clínico en una empresa como CEVAXIN, cuáles son sus fases y distintos recursos, y se buscará la manera de adaptarlo a un algoritmo genético multiobjetivo.

El objetivo es sacar una solución que sobre la teoría sea útil, aunque luego para poder llevarlo a la práctica habría que realizar muchas pruebas en real y adaptaciones que quedan un poco fuera del marco de este trabajo. Pero aún así se busca crear una base sobre la que trabajar en futuras investigaciones para conseguir aplicarlo de manera real en una clínica.

1.3. Marco Regulador

A continuación, se expondrán los aspectos éticos y legales que se ven afectados a lo largo de este proyecto. Este estudio hace uso de una gran cantidad de herramientas de software, para ello conoceremos las condiciones de su uso.

El proyecto a su vez también usará como inspiración datos provenientes de la empresa CEVAXIN, asegurando la seguridad y privacidad de los mismos.

1.3.1. Licencias de Software

Para el desarrollo del proyecto se ha usado como lenguaje de programación Python, y multitud de librerías de ayuda que son todas de licencia de libre uso al tener licencias de código *open source*.

En cuanto al entorno de desarrollo, se ha utilizado Pycharm, empleando la licencia profesional asociada a estudiantes de la UC3M.

Para el almacenamiento del resto de datos se ha utilizado Google Drive, el proporcionado por la UC3M, así como un repositorio de Github con la cuenta también de la propia universidad.

1.3.2. Aspectos Legales

En cuanto al uso de los datos de la empresa CEVAXIN se firmó un acuerdo mutuo para el uso de los archivos Excel con los datos de las clínicas. Este acuerdo de confidencialidad destacaba el consentimiento previo para recopilar los datos, derechos individuales de acceso, rectificación, supresión, etc. Asegurando por tanto su seguridad y privacidad.

2. ESTADO DEL ARTE

Este apartado busca proporcionar una visión comprensiva y detallada de la literatura y las tecnologías existentes que son relevantes para la optimización de la programación de citas en estudios clínicos.

Se comenzará explorando los desafíos y enfoques comunes en la programación de citas en contextos médicos. A continuación, se abordarán los modelos de optimización específicos que se han aplicado a las citas médicas para identificar las mejores prácticas y los enfoques más eficientes. Dado que los Algoritmos Genéticos representan una estrategia de optimización ampliamente utilizada, se examinarán en profundidad, incluyendo detalles sobre operadores genéticos y sus diversas técnicas. Posteriormente, se discutirán los enfoques de optimización multiobjetivo, con especial énfasis en los algoritmos NSGA y NSGA-II, que son fundamentales en este campo. Finalmente, se considerará la utilidad del lenguaje de programación Python y la librería pymoo, abarcando desde su arquitectura hasta sus aplicaciones específicas en optimización y análisis.

Este apartado servirá como una base sólida para entender los métodos y herramientas que informan y apoyan el desarrollo de este trabajo.

2.1. ¿Qué son los estudios clínicos?

Los estudios clínicos son investigaciones cuidadosamente diseñadas que buscan responder preguntas específicas sobre la eficacia y seguridad de nuevos tratamientos, medicamentos o dispositivos médicos [1]. Estos estudios son esenciales para el desarrollo de nuevas intervenciones médicas y para avanzar en el conocimiento médico.

Los participantes en los estudios clínicos, a menudo llamados también “sujetos”, son voluntarios que cumplen con criterios específicos y que están dispuestos a someterse a pruebas médicas para evaluar el impacto de una nueva intervención.

Si bien los estudios clínicos y los centros de salud tienen el objetivo común de mejorar la atención médica, existen diferencias clave entre estos dos:

1. Objetivo: Los centros de salud están orientados principalmente hacia el diagnóstico y tratamiento de enfermedades ya conocidas y validadas, mientras que los estudios clínicos están diseñados para investigar nuevas formas de diagnóstico, tratamiento o prevención.
2. Estructura: Los estudios clínicos están estructurados en torno a una investigación rigurosa, con controles estrictos y variables claramente definidas. Los centros de salud, por otro lado, ofrecen una gama más amplia de servicios médicos sin un protocolo específico.
3. Participación del paciente: En un centro de salud, los pacientes buscan atención médica basada en sus necesidades personales, mientras que, en un estudio clínico,

los sujetos son seleccionados en función de criterios específicos y deben seguir un régimen predefinido.

4. Riesgo y Beneficio: Los centros de salud utilizan tratamientos y enfoques que ya han sido validados y se consideran seguros. En los estudios clínicos, hay un elemento de incertidumbre, ya que se están probando nuevas intervenciones que aún no han sido completamente validadas.
5. Duración y Seguimiento: Los estudios clínicos suelen ser proyectos a corto o medio plazo con un seguimiento estricto, mientras que la atención en los centros de salud es más continua y adaptada a las necesidades a largo plazo del paciente.

Dadas todas estas diferencias, la programación de citas en estudios clínicos presenta desafíos únicos. Estos van desde la coordinación de múltiples actividades de investigación hasta el manejo de la disponibilidad limitada de recursos especializados, lo que hace que la optimización en este contexto sea una tarea compleja pero crucial.

2.2. Programación de citas en estudios clínicos

La programación de citas es un componente crítico en la gestión de estudios clínicos. Un calendario eficiente asegura el uso adecuado de los recursos, reduce los tiempos de espera, mejora la satisfacción del paciente y, en última instancia, aumenta la eficiencia de todo el proceso clínico. Sin embargo, el diseño de un horario eficiente es un desafío logístico considerable. La variabilidad en las necesidades de los pacientes, la disponibilidad de los profesionales y el uso de las instalaciones y recursos introducen complejidad, haciendo de la programación de citas una tarea no trivial.

Los principales desafíos asociados con esta programación incluyen:

- Variabilidad e imprevisibilidad: los servicios de atención médica a menudo experimentan variabilidad e incertidumbre en la duración del servicio y la llegada de los pacientes. Esto hace que sea desafiante crear horarios eficientes que puedan acomodar estas fluctuaciones [1].
- Equilibrio entre tiempos de espera e inactividad: la programación en atención médica implica encontrar el equilibrio adecuado entre los tiempos de espera e inactividad. Los tiempos de espera excesivos pueden llevar a la insatisfacción del paciente, mientras que los tiempos de inactividad excesivos pueden resultar en la subutilización de los recursos. Lograr un equilibrio apropiado es crucial para la calidad percibida, la eficiencia total de costes y la capacidad de los servicios de atención médica [2].

Estos desafíos destacan la necesidad de enfoques de programación flexibles y óptimos que puedan manejar la variabilidad, incorporar información distribucional limitada y lograr un equilibrio adecuado entre los tiempos de espera y los de inactividad.

No existe una gran investigación enfocada en la programación de citas en ensayos clínicos, pero sí es cierto que en la configuración general de atención médica o en el caso

de los horarios para quirófanos, se han utilizado varios métodos y sistemas para gestionar la planificación:

- Programación basada en la experiencia: muchas clínicas confían en la experiencia y las prácticas que han evolucionado a lo largo de los años para programar citas. Este enfoque se basa en la intuición y en el conocimiento tanto de médicos como de personal [3].
- Por orden de llegada: este enfoque implica programar las citas por orden de llegada FIFO (*First In First Out)*, es decir, el primer paciente que llega es asignado antes que el siguiente que llega, sin considerar factores como la urgencia del paciente o la disponibilidad de recursos. Esto es inviable con ensayos clínicos, ya que algunos requieren recursos específicos e intervalos de tiempo concretos entre sus fases [4].
- Simulación de eventos discretos: se han utilizado modelos de simulación de eventos discretos para evaluar diferentes secuencias de citas. Estos modelos simulan el flujo de pacientes a través del sistema y pueden proporcionar información sobre los tiempos de espera y la utilización de recursos. Estas simulaciones permiten la consideración de varios factores relevantes para las citas, como las ausencias o las citas el mismo día, que pueden impactar el proceso de la programación [5].
- Modelos de optimización: se han propuesto enfoques de optimización matemática para diseñar horarios de citas en entornos de atención médica. Estos modelos buscan lograr algoritmos apropiados para absorber la variabilidad e incertidumbre que conllevan estos entornos de atención médica. Gracias a los avances tecnológicos, estos modelos pueden lograr reducir el trabajo manual de semanas a pocos segundos [6].

Es importante destacar que los métodos y sistemas específicos utilizados en los ensayos clínicos pueden variar dependiendo de la naturaleza del ensayo y los requisitos específicos del protocolo del estudio. Siempre la programación de citas va a depender del tipo de ensayo, algunos requieren de espacio entre cada una de sus fases, con lo que hay muchos factores a tener en cuenta.

Sin embargo, las nuevas tecnologías y la digitalización han influido significativamente en la programación de citas. Con la llegada de los sistemas de captura de datos electrónicos (*EDC; Electronic Data Capture*) y los registros de salud electrónicos (EHR; *Electronic Health Records*), la programación de citas para los participantes en los ensayos clínicos se ha vuelto más eficiente y simplificada [7]. Estos sistemas pueden integrarse con los propios softwares de programación, permitiendo la disponibilidad en tiempo real de las citas y la programación automática en función de la elegibilidad del participante y los protocolos de cada estudio. Esta automatización reduce la carga administrativa del personal de investigación y también asegura que las citas se programen de manera oportuna.

La digitalización también ha permitido el uso de monitoreo remoto y telemedicina en ensayos clínicos. Los participantes ahora pueden tener visitas virtuales con proveedores

de atención médica, eliminando la necesidad de citas presenciales para ciertas actividades de estudio, permitiendo así también un mejor seguimiento durante las semanas siguientes a la realización del estudio, detectando posibles problemas o efectos secundarios cuanto antes. Esto no mejora solo en comodidad para los participantes, sino que también reduce los costos y posibles desafíos logísticos que conlleva la realización de visitas en persona. Además, las tecnologías digitales han facilitado la implementación de sistemas recordatorios para los participantes, a través de correos electrónicos, SMS o aplicaciones móviles, asegurándose de que estén conscientes de sus próximas citas y reduciendo la posibilidad de visitas perdidas.

En general, la programación de citas se ha vuelto más eficiente y sobre todo más accesible para los participantes. Estas mejoras tienen el potencial de mejorar las tasas de reclutamiento y retención de los estudios, mejorar la calidad de los datos y acelerar con ello el desarrollo de nuevos tratamientos y terapias [8].

2.3. Modelos de optimización aplicados a citas médicas

En las investigaciones realizadas hasta ahora, siempre se han encontrado complicaciones al pasar del marco teórico a la aplicación real. Se ha investigado mucho acerca de la planificación de citas médicas, y principalmente se han enfocado en dos visiones distintas:

- Planificación de recursos a pacientes
- Planificación de pacientes en un horizonte de tiempo según los recursos disponibles que se tengan.

Lo más efectivo hasta el momento y lo más estudiado es el segundo caso, y es el enfoque que va a tratar este trabajo.

El primer trabajo previo que se va a analizar es un enfoque que se realizó para mejorar la planificación de citas en Atención Primaria utilizando minería de datos [9]. Los autores del estudio parten del supuesto de que el tiempo que se dedica a un paciente en tareas administrativas, como por ejemplo la prescripción de medicación, es diferente del tiempo que se dedica a la atención clínica.

Con este planteamiento, desarrollaron un modelo predictivo basado en la teoría de *Support Vector Machine* (SVM). Las SVM son un tipo de algoritmo de aprendizaje automático que se utiliza para clasificar datos. Funcionan encontrando la línea o el hiperplano que mejor separa los datos en diferentes clases, y las predicciones se hacen viendo de qué lado del hiperplano caen los nuevos datos. El objetivo de este modelo es proporcionar datos sobre cuántos pacientes visitarán un centro de salud para solicitar un servicio específico, ya sea clínico o administrativo.

El estudio parte de la premisa de que determinadas condiciones meteorológicas pueden afectar a enfermedades crónicas como las alergias o el asma, lo que a su vez puede aumentar la demanda de pacientes que acuden a sus centros de salud o servicios de

urgencia simplemente para solicitar nuevas recetas de medicamentos, lo que conllevaría únicamente una tarea administrativa, sin necesitar una gran atención clínica.

Los autores argumentan que el uso de un modelo de optimización, en este caso el ya explicado previamente SVM, utilizando variables meteorológicas, es muy útil para predecir cuántos pacientes acudirán a los centros de salud por diversos motivos.

Los resultados presentados por los autores muestran que el error medio de predicción en los centros de salud de la provincia estudiada se sitúa en torno al 4%, lo que permite planificar mejor las citas médicas de los pacientes al tener en cuenta tiempos más realistas para la ejecución de las tareas en los servicios médicos. Los datos comparativos del año 2011 han demostrado que se pueden alcanzar una mejora en la eficiencia media del 21% en comparación con las prácticas tradicionales.

Otro de los principales problemas que pueden encontrarse a la hora de planificar citas clínicas es el tiempo de espera de los pacientes. Esto, como ya se ha comentado es un hecho que depende de muchos factores y con un alto grado de incertidumbre, ya que pueden aparecer nuevas citas urgentes en un mismo día, por adelantamiento de estudios o necesidades de la compañía, o también pueden ocurrir los llamados *no shows*, es decir que el paciente no se presente a la cita asignada y pierdas ese hueco.

Sobre esto hay un estudio que propone una idea acerca de cómo gestionar la programación de citas teniendo en cuenta los pacientes urgentes y las posibilidades de *no shows* [10].

Los investigadores consideran la posibilidad de que un paciente no acuda a su cita como un componente crucial de su modelo. Su objetivo es reducir la suma del tiempo medio de espera de los pacientes, el tiempo perdido por los médicos cuando los pacientes no acuden y el tiempo adicional que pueda ser necesario.

En su primer modelo, suponen que los tiempos de servicio clínico se conocen de forma determinista. También consideran que el tiempo de servicio sigue una distribución exponencial, lo que significa que la función objetivo no es multimodal, lo que no garantiza la posibilidad de encontrar una solución globalmente óptima.

Para superar estos retos planteados, comienzan a desarrollar un algoritmo heurístico. Las primeras simulaciones que realizan ilustran cómo los elementos clave considerados en el modelo influyen en la eficiencia de los servicios del hospital que han sido objeto de estudio, demostrando así que es posible mejorar los procesos computacionales para poder identificar soluciones que se consideren adecuadas en el entorno de la investigación.

El problema de la asignación de citas médicas también ha sido planteado en otro estudio [11], formulándolo como un proceso de decisión de Markov (MDP). Un MDP es un modelo matemático utilizado en la toma de decisiones óptimas en situaciones en las que los resultados son parcialmente aleatorios y parcialmente controlados por un agente de decisión. El objetivo en un MDP es encontrar una política, que es una estrategia que especifica qué acción tomar en cada estado, de manera que se maximice la suma total de recompensa a lo largo del tiempo. Los MDP son utilizados en muchas áreas de la inteligencia artificial, pero en gran parte son clave para el aprendizaje por refuerzo.

Los investigadores plantearon el MDP teniendo en cuenta restricciones como la capacidad máxima diaria de pacientes programados y el número limitado de enfermeras y camas disponibles.

El estudio pretendía determinar la distribución óptima de franjas horarias para pacientes de urgencias. Los investigadores examinaron el impacto de varios factores en la eficacia de la programación, como la carga del sistema y la variabilidad de los tiempos de servicio.

Aunque en un principio los investigadores modelizaron el reto como un MDP, demostraron que era intratable en términos de tiempo computacional. Entonces lo adaptaron a un modelo lineal para obtener soluciones en un tiempo más manejable. Sus resultados ponen de relieve las ventajas de utilizar métodos científicos para programar las citas de los pacientes en comparación con los métodos tradicionales.

Por otra parte, también son de importancia aquellas investigaciones realizadas que tengan en cuenta las características de los pacientes como elemento fundamental para realizar la planificación médica. Este apartado es complejo ya que por temas de privacidad es muy complicado poder disponer de la información necesaria sobre los pacientes.

En un estudio de 2016 [12] , Salzarulo y su equipo presentaron un modelo que vinculaba la productividad de los servicios con la programación de citas de los pacientes de un hospital clínico de Estados Unidos. Su objetivo era optimizar la planificación de citas médicas, incorporando información específica del paciente, por ejemplo, si el paciente es nuevo o de seguimiento previo, si tiene enfermedades crónicas como asma o diabetes, y si su visita médica puede ser más probable que esté relacionada con problemas respiratorios, alergias, infecciones, etc.

El equipo realizó una serie de estimaciones de modelos de regresión para determinar el mejor ajuste que explicara el comportamiento de la variable dependiente para distintos números de variables independientes. En el modelo propuesto, la variable dependiente se definió como el tiempo que cada paciente pasaba en el hospital, desde que es diagnosticado hasta que es tratado y dado de alta.

Este estudio considera tres distintos modelos que explotan esto parcial o totalmente, utilizando parte o toda la información de entrada proporcionada por la variable independiente. El objetivo del proceso es ir más allá de solo obtener información y predecir en base a la consulta médica media. Los autores para ello proponen tres modelos de regresión.

En el primer modelo propuesto, definen solo una variable independiente, si el paciente es nuevo o está en seguimiento, y por lo tanto representa un modelo de regresión simple para predecir la duración de las citas. Los resultados de este primer modelo es que esta variable es significativa, y que el coeficiente de determinación es alto (0,9247).

El segundo y el tercer modelo tienen en cuenta información adicional sobre el paciente. En cada modelo se añadieron 8 y 10 nuevas variables significativas, con unos resultados con coeficientes de determinación similares a los del primer modelo. En este contexto, se considera que los dos últimos modelos presentan un problema de inclusión de posibles

variables irrelevantes, que no afecta a la coherencia de la estimación de los efectos de las variables, pero puede provocar una pérdida de eficiencia en la estimación.

Como conclusión del estudio, los miembros concluyen que, en comparación con otras formas de planificar citas médicas, los costes basados en el tiempo perdido de los médicos, las posibles horas extra empleadas y el tiempo de espera de los pacientes, se reducen en un 24%. Sin embargo, estas estimaciones no han sido probadas en otros modelos que permitan garantizar una completa automatización de todo el sistema de planificación de citas.

Por último, el último estudio que se va a analizar sobre el proceso de planificación de pacientes es uno en el que los autores estudiaron los problemas existentes en la planificación de citas en un hospital materno de Estados Unidos [13]. Algunos de los problemas que encontraron son los largos tiempos de espera de los pacientes, la sobrecarga de trabajo de los profesionales o los altos niveles de *no-shows* (en algunas de las especialidades, la mitad de los pacientes citados no acuden). Otro problema que también destacan es que a veces los pacientes tienen citas a la misma hora, lo que provoca por tanto largos tiempos de espera.

Los autores describen el hospital como un área de servicios médicos con elementos muy específicos que hacen que su gestión sea bastante compleja. Para dicha gestión, es necesario tener en cuenta varios tipos de servicios, como: atención prenatal, atención neonatal rutinaria, atención prenatal de alto riesgo o atención ginecológica.

Estas actividades requieren distintos tipos de equipos, que a menudo deben intercambiarse debido a las limitaciones de espacio de las subespecialidades dentro del hospital. Este problema provoca muchas pérdidas de tiempo durante la jornada laboral.

Este estudio propone dos modelos de optimización que pretenden mejorar el proceso de toma de decisiones para la planificación semanal de citas médicas. El primer modelo pretende conseguir un equilibrio en términos de carga de trabajo entre las diferentes sesiones que operan en paralelo, para ello, determina una variable de decisión que describe el número total de pacientes que pueden ser reservados en las distintas sesiones clínicas del hospital. Para ello, uno de los problemas a resolver es asignar a las sesiones clínicas un grupo de servicios con características similares.

Los autores de este modelo inicial describen tres tipos fundamentales de restricciones. El primer tipo pretende garantizar que todas las solicitudes de servicios médicos de los pacientes puedan atenderse dentro de las sesiones disponibles para el servicio correspondiente. El segundo y el tercer tipo garantizan que las citas médicas dentro de cada sesión clínica se programen para el mismo tipo de servicio. El número de pacientes que solicitarán cada tipo de servicio, incluido el porcentaje de pacientes que no acuden a sus citas, se consideran parámetros estimables e inciertos. Para resolver este problema se utiliza el sistema informático LINGO. LINGO (*LINear Generalize Optimizer*) es una herramienta para formular problemas lineales y no lineales, resolverlos y analizar su solución. El resultado que LINGO proporciona es la optimización que ayuda a encontrar el mejor resultado, ya sea la ganancia más alta o el coste más bajo [14].

Los autores, en función de los recursos disponibles y del número medio de pacientes, desarrollan un segundo modelo de programación estocástica, que asigna franjas horarias a los pacientes, con el objetivo de minimizar los tiempos de espera, el tiempo perdido y la sobrecarga de trabajo de los profesionales. Para lograr este objetivo se definen varias variables, entre ellas una binaria que toma el valor uno si un paciente es asignado a una franja horaria disponible y cero en caso contrario. Las dos variables restantes se refieren a los otros dos objetivos.

Para que el modelo sea manejable desde el punto de vista computacional, hay que partir de varios supuestos. En primer lugar, se supone que los pacientes acuden puntualmente a las citas médicas en cada categoría de servicio. En segundo lugar, el tiempo que un médico pasa con cada paciente se distribuye de forma aleatoria pero idéntica. En tercer lugar, se conocen las probabilidades de *no-show* a una cita médica.

Este segundo modelo, que utiliza la solución obtenida en el primer modelo, tiene en cuenta las siguientes restricciones:

1. Varios médicos están disponibles en una sesión clínica y, por tanto, pueden atender a los pacientes según la disciplina de colas.
2. Los pacientes que llegan más de cinco minutos tarde son tratados como no presentados y deben ser reprogramados.
3. Los tiempos de atención son independientes y se distribuyen por igual.
4. Los pacientes que no se presentan son independientes y se conoce la probabilidad asociada a este suceso.

El primero de los puntos es un supuesto específico del hospital objeto del estudio, que ayuda a simplificar el tratamiento del modelo. Sin embargo, los autores reconocen que en otros entornos en los que un paciente debe ser atendido por un equipo específico de médicos, restringiría y complicaría el modelo.

En la aplicación práctica del modelo, los autores consideraron cuatro sesiones clínicas, repartidas a lo largo de dos días, para planificar pacientes de las distintas subespecialidades disponibles en el hospital. Los resultados indican que, de media, se necesitan diez horas para planificar los pacientes de estas sesiones haciendo uso de este modelo. Estos resultados demuestran que es muy importante continuar investigando y optimizando los gastos computacionales vinculados a este tipo de actividades.

Aunque los modelos matemáticos planteados por este estudio funcionan desde el punto de vista teórico, es poco realista la generalización de la aplicación a otros hospitales maternos, y la alta carga computacional tampoco invita a la implantación en otros centros con características similares.

Después del análisis de todos estos estudios, se sacan las siguientes conclusiones que serán adoptadas en el desarrollo de este trabajo:

1. Es importante incorporar características de los pacientes o de los distintos estudios a los modelos, para lograr una mejor gestión personalizada en cada caso.

2. El paciente es el centro de todos los procesos, con lo cual se considerará como objetivo el tiempo de espera que estos tienen que sufrir cuando acuden a su centro clínico.
3. Se tendrá en cuenta también la sobrecarga de trabajo de los trabajadores del centro de estudios clínicos, evitando al máximo sus horas extra, así como optimizando el horario para que tengan el menor número de horas vacías.

El estudio acerca de optimización de planificaciones en centros de estudios clínicos tiene muy poco recorrido bibliográfico, pero se han decidido estudiar los casos hospitalarios o en centros de salud ya que son parte del mismo entorno y se enfrentan a los mismos problemas.

Los estudios clínicos como por ejemplo los de vacunación, se enfrentan a otras características distintas a los centros médicos, como pueden ser la importancia de las franjas temporales a la hora de realizar las siguientes pruebas, así como la implementación de un sistema de seguimiento de los pacientes fuera del centro clínico.

2.4. Algoritmos Genéticos

Los algoritmos metaheurísticos se utilizan cada vez más en los últimos años para ayudar a resolver problemas complejos de la vida real en diversos campos, como la economía, la política, la ingeniería y la medicina. Los elementos clave de un algoritmo metaheurístico son la intensificación y la diversificación. Una solución eficaz a los problemas de la vida real requiere un equilibrio adecuado entre estos elementos. La mayoría de estos algoritmos están inspirados en el proceso evolutivo biológico, el comportamiento de los enjambres y de las leyes físicas [15]. Estos algoritmos son clasificados en dos categorías, los algoritmos basados en una solución única y los basados en poblaciones Figura 1.

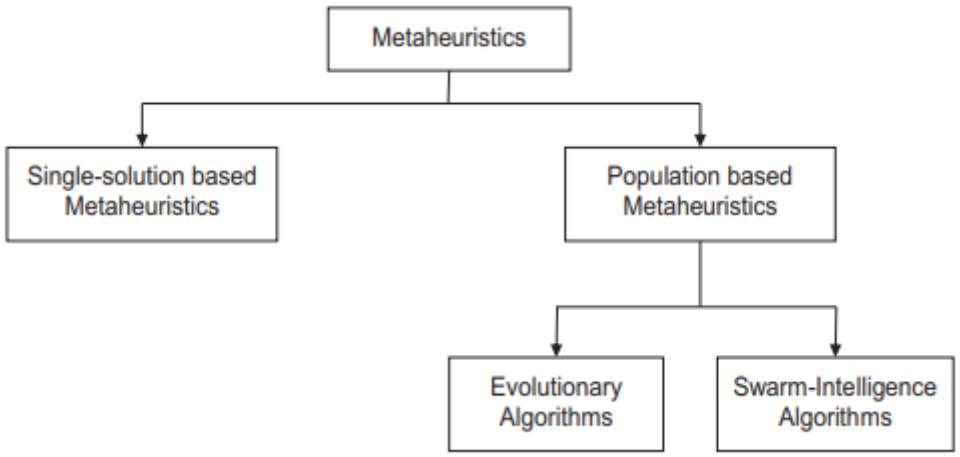

Figura 1: Clasificación de Algoritmos Metaheurísticos [16]

Los algoritmos metaheurísticos basados en una única solución utilizan una única solución candidata y la mejoran empleando la búsqueda local. Sin embargo, la solución obtenida con metaheurísticas basadas en una única solución puede quedar atrapada en óptimos

locales, lo que hace que no sean óptimos para algunos problemas. Algunos de los algoritmos metaheurísticos basados en una única solución más conocidos son la búsqueda tabú (*Tabu Search*) y la búsqueda local guiada (*Guided Local Search*).

Por otro lado, los algoritmos metaheurísticos basados en poblaciones utilizan múltiples soluciones candidatas durante el proceso de búsqueda. Mantienen la diversidad en la población y evitan así que las soluciones se estanquen en óptimos locales. Entre los algoritmos metaheurísticos basados en poblaciones más conocidos se encuentran el algoritmo genético (GA) [17], la optimización por enjambre de partículas (PSO) [18] y la optimización basada en colonias de hormigas (ACO) [19].

El algoritmo genético (GA) es un conocido algoritmo metaheurístico inspirado en el proceso de evolución biológica. El GA imita la teoría darwiniana de la selección natural, es decir, la supervivencia del más apto. J.H. Holland propuso el GA en 1975 [20]. Los componentes básicos del GA son la representación cromosómica, la selección de la aptitud y los operadores de inspiración biológica. Holland también introdujo un elemento novedoso llamado Inversión, que se utiliza en las implementaciones del GA.

Los cromosomas suelen tener un formato de cadena binaria. Cada una de las posiciones específicas del cromosoma (locus), tiene dos posibles alelos: 0 y 1. Se consideran puntos en el espacio de soluciones. La población se sustituye iterativamente aplicando operadores genéticos. La función de *fitness* (adecuación) asigna un valor a todos los cromosomas de la población.

Los operadores inspirados en la biología son la selección, la mutación y el sobrecruzamiento. Durante la selección, los cromosomas se eligen en función de su valor de *fitness* para su posterior procesamiento. El sobrecruzamiento selecciona aleatoriamente un locus y modifica las subsecuencias entre cromosomas para crear descendencia. Durante la mutación, ciertos bits de los cromosomas se cambian aleatoriamente en función de probabilidades.

El algoritmo genético sigue el siguiente procedimiento. Para empezar, se inicializa aleatoriamente una población *Y* formada por *n* cromosomas. Se calcula el valor de *fitness* de cada cromosoma de *Y*. Se seleccionan dos cromosomas, *C1* y *C2*, de la población *Y* en función de su valor de *fitness*. Se aplica un operador de sobrecruzamiento con una probabilidad C_p en *C1* y *C2*. El resultado es un descendiente, *O*. A continuación, se aplica un operador de mutación M_p para generar *O´*. El nuevo descendiente *O´* se añade a la nueva población. Las operaciones de selección, cruce y mutación se repetirán en la población actual hasta completar la nueva población (Figura 3).

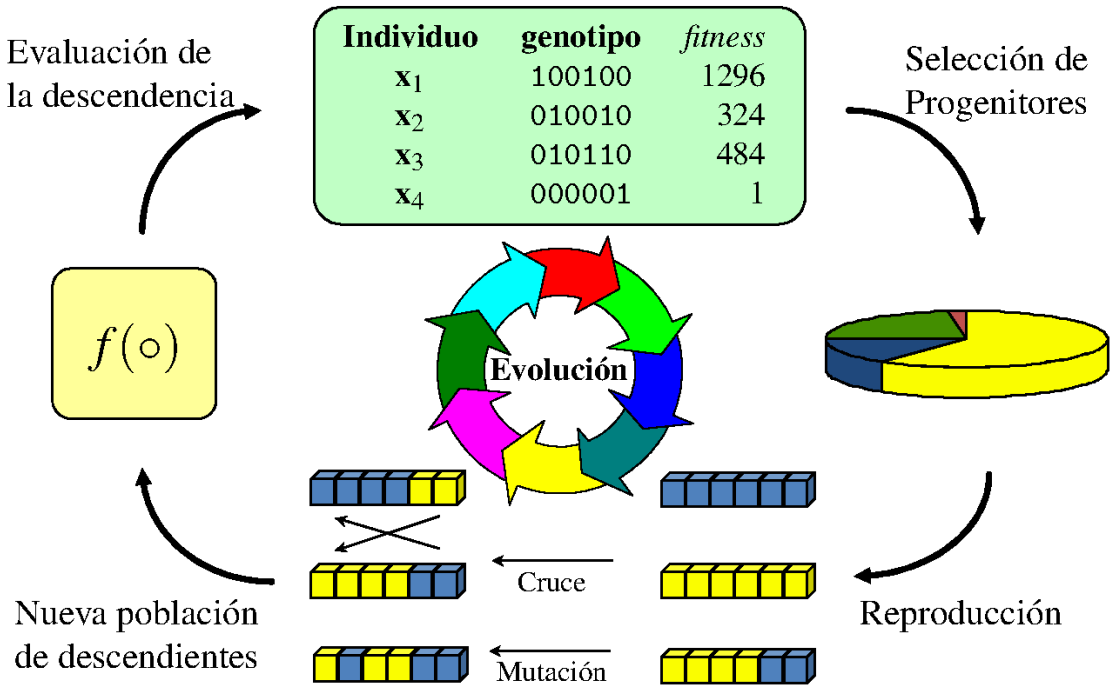

Figura 2: Ciclo algoritmo genético [21]

2.4.1. Operadores genéticos

Los algoritmos genéticos usan cuatro distintas categorías de operadores: los tres mencionados previamente, inspirados en la biología, selección mutación y sobrecruzamiento; y también hay que tener en cuenta los distintos esquemas de codificación, que deciden cómo transformamos una solución del problema en un individuo o cromosoma que el algoritmo genético puede entender y trabajar. Se pueden ver desplegados en la siguiente ilustración (Figura 3).

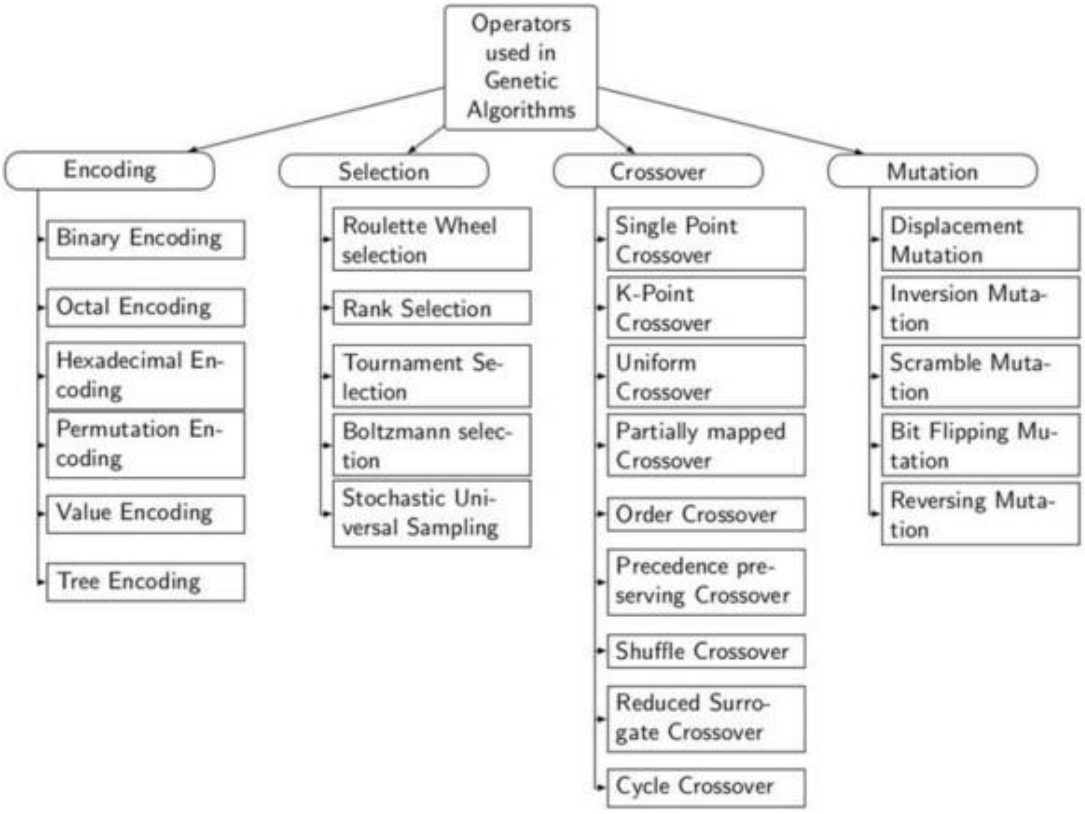

Figura 3: Operadores genéticos usados en los GA [16]

2.4.1.1. Esquemas de codificación

El esquema de codificación, que se refiere al proceso de convertir la información en una forma específica que procese el algoritmo, es crucial para muchos problemas computacionales. La información proporcionada debe codificarse en una cadena binaria específica.

Los distintos esquemas de codificación se distinguen en función de su relevancia para el dominio de un problema concreto. Los esquemas de codificación más conocidos son el binario, el octal, el hexadecimal, el de permutación, el basado en valores y el de árbol.

El que más se suele utilizar es el esquema de codificación binaria [22]. Una cadena de 1 y 0 representa cada gen o cromosoma. Cada bit de la codificación binaria representa una característica de la solución. Esto proporciona una implementación más rápida de los operadores de cruce y mutación. Sin embargo, se requiere un esfuerzo adicional para convertirlo en forma binaria, y la precisión del algoritmo depende de esta conversión. El flujo de bits se modifica en función del problema.

El esquema de codificación octal representa el gen o cromosoma utilizando números octales (0-7), mientras que el esquema de codificación hexadecimal representa el gen o cromosoma mediante números hexadecimales (0-9, A-F).

El esquema de codificación de permutación se utiliza habitualmente en problemas que requieren ordenación. En este esquema de codificación, el gen o cromosoma se denota mediante una cadena de números que indica la posición en una secuencia.

En un esquema de codificación de valores, el gen o cromosoma se representa mediante una cadena de valores determinados. Los valores pueden ser números reales, enteros o caracteres. Este esquema puede ser útil para resolver problemas que implican valores más complejos, porque la codificación binaria es propensa a fallar en este tipo de problemas. Este método se aplica principalmente en redes neuronales para determinar los pesos óptimos.

El esquema de codificación en árbol representa el gen o cromosoma mediante un árbol de funciones o comandos. Las funciones y comandos pueden estar relacionados con cualquier lenguaje de programación.

En la Tabla 1 se puede ver la comparación de los distintos esquemas de codificación:

Esquema de Codificación	Beneficios	Contras	Aplicación
Binario	Fácil de implementar y rápida ejecución	No admite operador de inversión	Problemas que permiten codificación binaria
Octal	Fácil de implementar	No admite operador de inversión	Uso limitado
Hexadecimal	Fácil de implementar	No admite operador de inversión	Uso limitado
Permutación	Admite operador de inversión	No admite operadores binarios	Problemas de ordenación
Valores	No necesita conversión de valores	Necesita mutación y sobrecruzamiento específico	Problemas de redes de neuronas
Árbol	Operadores son fácilmente aplicables	Difícil diseñar árboles para algunos problemas	Problemas evolutivos

Tabla 1: Comparación esquemas de codificación

2.4.1.2. Técnicas de selección

En los algoritmos genéticos, la selección es un paso crucial que determina si una cadena específica participará o no en el proceso de reproducción. A veces, el paso de selección también se conoce como operador de reproducción [23]. La presión de selección influye significativamente en la tasa de convergencia de los GA. Las técnicas de selección más conocidas son: la ruleta, el rango, el torneo, Boltzmann, el muestreo universal estocástico y el elitismo.

La técnica de selección de la ruleta asigna posibles cadenas de soluciones a una rueda, con asignaciones basadas en sus correspondientes valores de *fitness*. La rotación aleatoria de la rueda se utiliza para seleccionar las soluciones particulares que contribuirán a la generación del siguiente conjunto. Sin embargo, este método está plagado de problemas, como los errores que surgen de su propia naturaleza estocástica. Para rectificar esto, De Jong y Brindle [24] propusieron modificaciones al método de selección de la ruleta que minimizarían los errores empleando un enfoque de selección determinista.

La técnica de selección conocida como selección por rangos es una modificación del método de selección de la ruleta. En este método, se utilizan los rangos de los individuos en lugar de sus valores de *fitness*. Los rangos se asignan en función de los valores de *fitness* correspondientes de los individuos, de modo que cada uno de ellos tiene la oportunidad de ser elegido en función de su rango. Al utilizar la selección por rangos, se reduce el riesgo de que la solución converja prematuramente a un mínimo local.

La técnica de selección por torneo fue propuesta por primera vez por Brindle en 1983. Se seleccionan parejas de individuos en función de sus valores de *fitness* en una ruleta estocástica. Los individuos con valores de *fitness* más altos se añaden a la reserva de la siguiente generación tras el proceso de selección. En esta técnica, cada individuo se compara con todos los otros n-1 individuos, siempre que alcance la población final de soluciones [25].

El muestreo universal estocástico SUS (Stochastic Univesal Sampling) amplía el método de selección de la ruleta. El método SUS selecciona un nuevo individuo a intervalos uniformemente espaciados de la lista de individuos de una generación, partiendo de un punto aleatorio. El método SUS proporciona la misma oportunidad a todos los individuos de participar en el cruce para la siguiente generación.

El método de selección de Boltzmann se basa en técnicas de entropía y muestreo empleadas habitualmente en la simulación Monte Carlo [26]. Esta técnica ayuda a resolver el problema de la convergencia prematura. Ofrece una alta probabilidad de seleccionar la mejor opción en un tiempo relativamente corto. No obstante, existe la posibilidad de pérdida de información. Sin embargo, esto puede solucionarse aplicando el elitismo.

La propuesta de utilizar el método de selección elitista pretende mejorar el rendimiento también de la selección de la ruleta. Se garantiza que el individuo elitista de una generación se propague a la siguiente. Si el individuo con el valor de *fitness* más alto está ausente en la siguiente generación tras el procedimiento de selección normal, el elitista también se incluye automáticamente en la siguiente generación.

En la siguiente Tabla 2 se puede ver la comparación de los distintos esquemas de codificación:

Técnicas de selección	Beneficios	Contras
Ruleta	Fácil de implementar y simple	Riesgo de convergencia prematura y depende de variaciones presentes en la función de *fitness*
Rangos	Preserva la diversidad	Se requiere ordenación, convergencia lenta y computacionalmente cara
Torneo	Preserva la diversidad, no requiere ordenación e implementación en paralelo	Pérdida de diversidad cuando el tamaño del torneo es grande
Boltzmann	Óptimo global alcanzable	Computacionalmente cara
SUS	Método rápido	Convergencia prematura
Elitismo	Mantiene al mejor individuo en la población	El mejor individuo puede perderse en el sobrecruzamiento o en la mutación

Tabla 2: Comparación técnicas de selección

2.4.1.3. Técnicas de sobrecruzamiento

Los operadores de sobrecruzamiento o cruce generan descendientes combinando la información la información genética de dos o más progenitores. Los operadores de cruce más conocidos son: un punto, dos puntos, punto k, uniforme, parcialmente emparejado, orden, cruce que preserva la precedencia, mezcla, sustituto reducido y cíclico.

En un cruce de un único punto se elige un punto de cruce aleatorio [27]. Más allá de este punto elegido, la información genética de los dos progenitores se intercambia entre sí. La nueva descendencia se obtiene sustituyendo los bits de la matriz de cola de ambos progenitores. En la siguiente Figura 4 se puede observar el proceso.

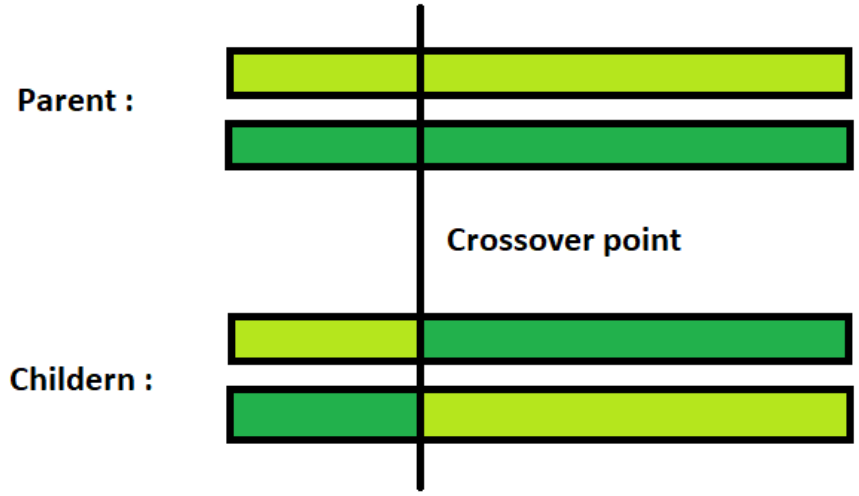

Figura 4: Cruce de un único punto [28]

Durante un cruce de dos puntos o de k puntos, se seleccionan dos o más (k) puntos de cruce aleatorios. A continuación, la información genética de los progenitores se intercambia en función de los segmentos recién formados. Para crear nuevos descendientes, se sustituye el segmento central del material genético parental. En la Figura 5 se puede observar el funcionamiento del cruce de dos puntos.

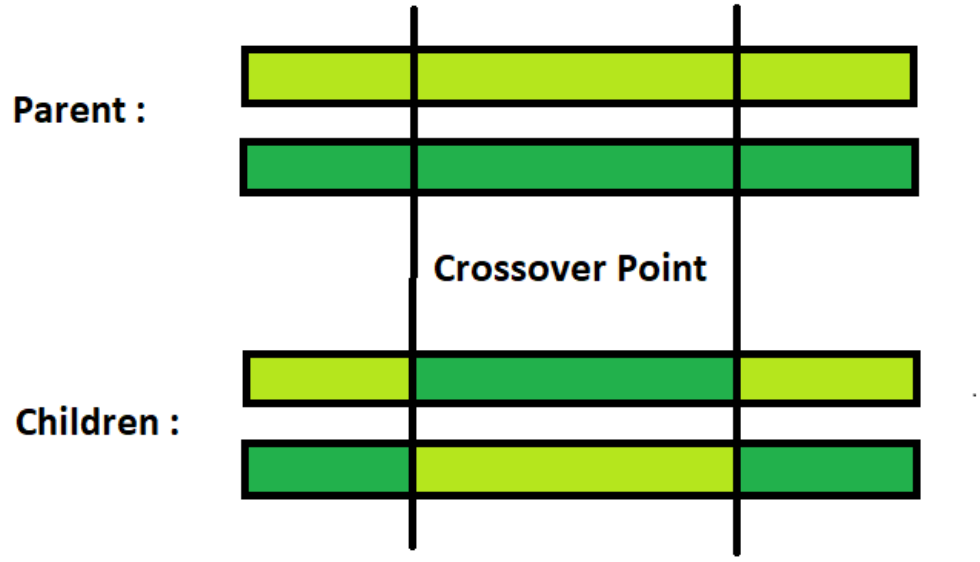

Figura 5: Cruce de dos puntos [28]

Durante un cruce uniforme, los cromosomas padres no pueden descomponerse en segmentos, sino que cada gen del progenitor puede tratarse por separado. La decisión de

intercambiar un gen en el mismo lugar de otro cromosoma se toma al azar. En la Figura 6 se puede observar como funciona el cruce uniforme.

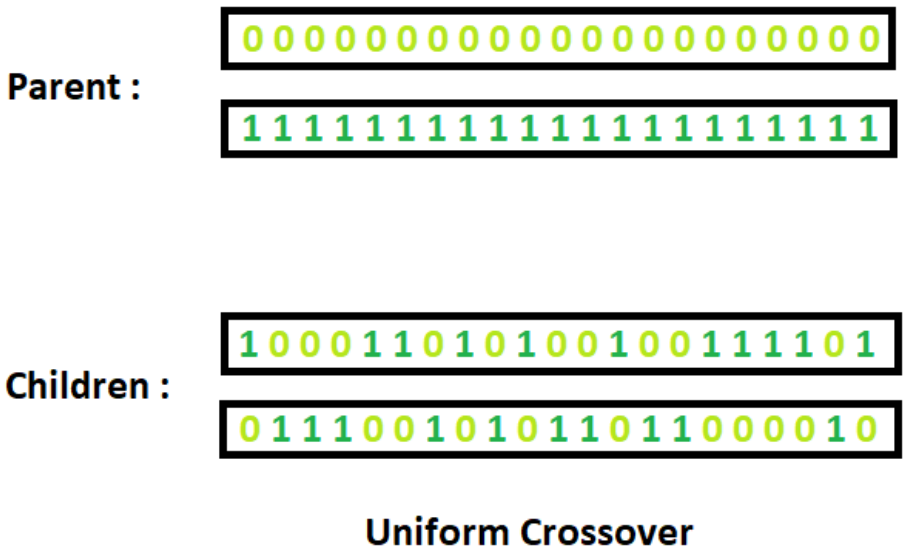

Figura 6: Cruce uniforme [28]

El cruce por emparejamiento parcial (*Partially Matched Crossover*) es uno de los operadores de cruce más utilizados, ya que funciona mejor que la mayoría de los demás operadores de cruce. El PMX o cruce mapeado fue introducido en 1985 [29]. En el proceso, se eligen dos padres para el apareamiento. Durante este, uno de los padres dona una porción de material genético y la parte correspondiente es aportada por el otro padre para formar el hijo. Una vez completado el proceso de apareamiento, los alelos restantes se copian del segundo progenitor. En la Figura 7 se puede observar el funcionamiento de este método.

El cruce ordenado también se conoce como OX. En los puntos de corte seleccionados, OX copia una o más selecciones del progenitor al descendiente y rellena la sección restante con valores no incluidos en la selección copiada. Diferentes investigadores han propuesto variantes de OX para abordar distintos tipos de problemas, descubriendo que es muy eficaz para resolver problemas de ordenación, mientras que en otros como por ejemplo el Problema del Vendedor Viajero tiene peor rendimiento.

El cruce que preserva la precedencia (PPX) garantiza que el orden de las soluciones individuales en el progenitor de la descendencia se preserva antes de aplicar la operación de cruce. La descendencia se inicializa con una cadena de 0s y 1s asignados aleatoriamente, lo que determina qué individuos de ambos padres se seleccionan o no.

La técnica de cruce mezcla fue propuesta como método para reducir los sesgos introducidos por otras técnicas de cruce. Esta técnica baraja los valores de una solución individual antes de la operación de cruce y los vuelve a barajar después para garantizar

que el punto de cruce no está sesgado. Sin embargo, el uso de esta técnica en los últimos años ha sido limitado en comparación a otras.

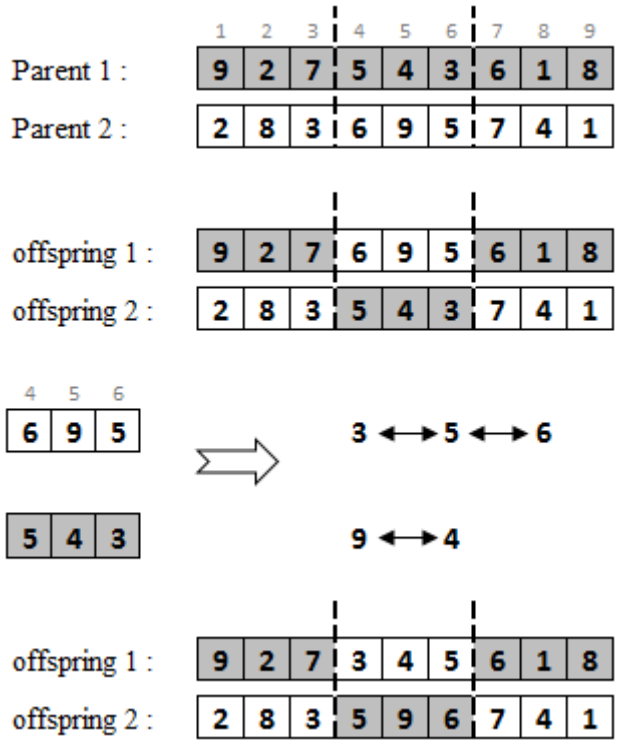

Figura 7: PMX [30]

El cruce del sustituto reducido (RCX) reduce los cruces innecesarios si los padres tienen la misma secuencia genética para las representaciones de la solución. RCX se basa en la suposición de que un GA produce mejores individuos si los padres son suficientemente diversos en su composición genética. Sin embargo, RCX no puede producir mejores individuos para aquellos progenitores que tienen la misma composición.

La técnica de cruce cíclico genera una descendencia seleccionando los padres en función de la posición de cada elemento. La técnica toma algunos elementos del primer padre durante el primer ciclo. Durante el segundo ciclo, los elementos restantes se toman del segundo progenitor, tal y como se puede ver en la Figura 8.

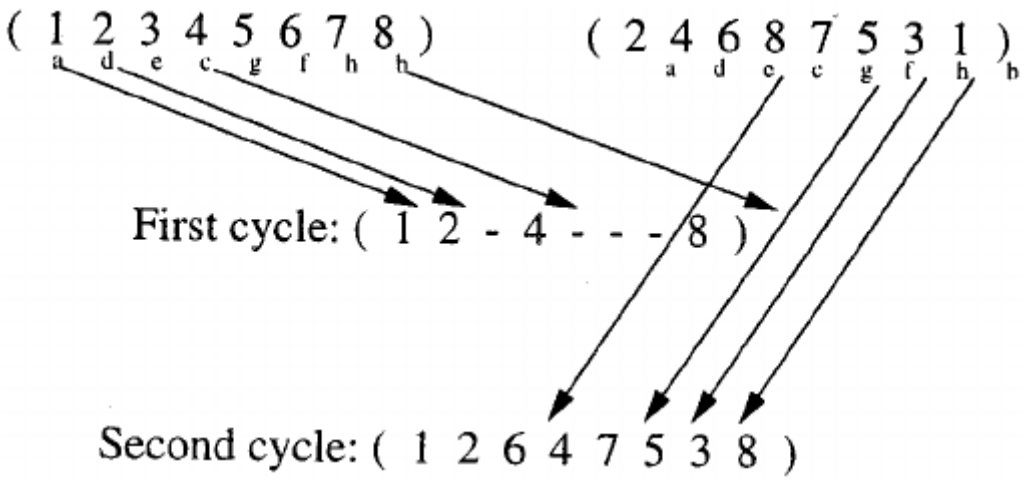

Figura 8: Cruce cíclico [31]

En la Tabla 3 se puede ver la comparación de los distintos esquemas de codificación:

Técnicas de cruce	Beneficios	Contras
Un punto	Fácil de implementar y simple	Menor diversidad de soluciones
Dos puntos / K puntos	Fácil de implementar	Menor diversidad de soluciones y aplicable a pequeñas poblaciones
Uniforme	Exploración imparcial, aplicable a grandes poblaciones y mejor potencial de recombinación	Menor diversidad de soluciones
Parcialmente emparejado	Mejor tasa de convergencia y superior a las otras técnicas	-
Cruce que preserva la precedencia	Mejor generación de descendencia	Problema de redundancia
Mezcla	Reduce los sesgos de otras técnicas	No es muy utilizada
Sustituto reducido	Mejor rendimiento en problemas de optimización pequeños	Convergencia prematura
Cíclico	Exploración imparcial	Convergencia prematura

Tabla 3: Comparación técnicas de cruce

2.4.1.4. Técnicas de mutación

La mutación es un operador que preserva la diversidad genética entre poblaciones. Las técnicas de mutación más utilizadas son el desplazamiento, la inversión simple y la mutación de *scramble*.

El operador de mutación por desplazamiento desplaza una sección de una solución individual dada dentro de sí misma. La posición se selecciona aleatoriamente de la subcadena específica para el desplazamiento, lo que garantiza que la solución resultante sea válida y estocástica. Las variaciones de este operador de mutación incluyen la mutación por intercambio y la mutación por inserción. Los operadores de mutación por intercambio y mutación por inserción implican intercambiar una parte de una solución individual con otra parte o insertarla en un lugar diferente.

El operador de inversión simple o SIM invierte la subcadena entre dos ubicaciones específicas en una solución individual. SIM es un operador de inversión que invierte una cadena seleccionada al azar y la coloca en una ubicación aleatoria.

El operador de mutación de *scramble* organiza los elementos dentro de un rango especificado de la solución individual en un orden aleatorio y comprueba si el valor de *fitness* mejora en la solución resultante.

En la Tabla 4 se puede ver la comparación de los distintos esquemas de codificación:

Técnicas de mutación	Beneficios	Contras
Mutación por desplazamiento	Fácil de implementar y aplicable en pequeños problemas	Riesgo de convergencia prematura
Mutación por inversión simple	Fácil de implementar	Riesgo de convergencia prematura
Mutación por *scramble*	Afecta a gran cantidad de genes y es aplicable a grandes problemas	Deterioro de la calidad de la solución en algunos problemas

Tabla 4: Comparación técnicas de mutación

2.5. Optimización Multiobjetivo

En problemas de optimización que buscan optimizar en base a múltiples objetivos, estos suelen entrar en conflicto, lo que hace imposible optimizarlos todos simultáneamente. La mayoría de los problemas del mundo real tienen múltiples objetivos, como la minimización de costes, la maximización del rendimiento y la maximización de la fiabilidad, y el objetivo del problema es optimizar en base a todos estos objetivos. Resolver este tipo de problemas es difícil, pero se han encontrado maneras de hacerlo a lo largo de los años.

Los algoritmos genéticos, como ya se ha visto, son metaheurísticas ampliamente utilizadas, especialmente adecuadas para esta categoría de problemas de optimización. Los GA convencionales se adaptan para abordar problemas multiobjetivo empleando funciones de *fitness* específicas e incorporando enfoques para mejorar la diversidad de soluciones.

Existen dos enfoques generales para optimizar múltiples objetivos. El primero consiste en dos opciones: combinar funciones objetivo individuales en una función compuesto o en trasladar todos los objetivos menos uno al conjunto de restricciones. En la primera de ellas, puede determinarse un único objetivo basándose en herramientas estadísticas, como son la teoría de la utilidad o el método de la suma ponderada. Sin embargo, la selección precisa de ponderaciones o funciones de utilidad para representar las preferencias que se tengan hacia cada restricción es una tarea compleja. Además, pequeñas perturbaciones en las ponderaciones pueden dar lugar a soluciones muy diferentes. En la segunda de las opciones, trasladar los objetivos al conjunto de restricciones implica establecer un valor de restricción para cada uno de los objetivos anteriores, que puede ser bastante arbitrario. En ambos casos, los métodos de optimización solo producen una única solución, en lugar de un conjunto de soluciones que puedan evaluarse en función de las compensaciones, que podría beneficiar a los responsables de la toma de decisiones, adoptando según su experiencia la solución más adecuada.

El segundo enfoque general para optimizar múltiples objetivos suele consistir en determinar con conjunto completo de soluciones Pareto-óptimas. Por conjunto óptimo de Pareto se entiende un conjunto de soluciones no dominadas mutuamente. Cuando se pasa de una solución Pareto a otra, hay que sacrificar un objetivo para conseguir ganancia en los otros objetivos. A menudo se prefieren los conjuntos de soluciones óptimas de Pareto a una solución única, ya que pueden resultar más prácticos en los problemas de la vida real. Los conjuntos óptimos de Pareto pueden variar en tamaño, pero tienden a aumentar con el número de funciones objetivo.

La capacidad de los GA para explorar simultáneamente diferentes áreas de un espacio de soluciones permite encontrar un amplio conjunto de ellas para casos difíciles, con espacios de soluciones no convexos, discontinuos y multimodales.

El operador de sobrecruzamiento del GA puede utilizar las estructuras de soluciones óptimas relativas a objetivos distintos para producir nuevas soluciones no dominadas en las regiones inexploradas del frente de Pareto. Además, la mayoría de los GA multiobjetivo no obligan al usuario a priorizar, cuantificar o ponderar los objetivos, aunque en algunos casos sea beneficioso para obtener un resultado más adecuado a los intereses del usuario. En consecuencia de todo esto, los GA se han convertido en el enfoque heurístico más popular para resolver problemas de diseño y optimización multiobjetivo.

Según un estudio del 2002 [32], el 90% de las técnicas de optimización multiobjetivo pretendían aproximarse al verdadero frente de Pareto para el problema en cuestión. La mayoría de ella utilizaban técnicas metaheurísticas, y el 70% de todos los enfoques metaheurísticos se basaban en técnicas evolutivas. Estas cifras no han hecho más que subir a lo largo de estos últimos 20 años, demostrando así que es la mejor aproximación posible para problemas multiobjetivo.

El primer GA multiobjetivo, llamado GA Evaluado Vectorialmente (o VEGA), fue propuesto por Schaffer [33]. Posteriormente, se desarrollaron muchos más algoritmos multiobjetivo, siendo los más relevantes: el Algoritmo Genético Multiobjetivo (MOGA), el Algoritmo Genético de Ordenación No Dominante (NSGA), el Algoritmo de Ordenación No Dominante Rápido (NSGA-II), el Algoritmo Evolutivo de Pareto Fortalecido (SPEA), el SPEA mejorado (SPEA2), y el Algoritmo de Selección Basado en la Envolvente de Pareto (PESA).

2.5.1. NSGA y NSGA-II

Después de determinar que la mejor manera para evaluar un problema multiobjetivo es el uso de GA multiobjetivo, se va a centrar el estudio sobre uno de ellos, el NSGA-II. Pero antes se ha de estudiar el origen de este algoritmo, el cuál proviene de su antecesor el NSGA.

2.5.1.1 NSGA

El NSGA fue propuesto en 1994 por Srinivas y Deb [34]. Como uno de los primeros algoritmos genéticos multiobjetivo, es una técnica que combina el GA con el método de ordenación no dominada. El coste de cada solución se asigna en función de su dominancia.

El primer paso consiste en generar aleatoriamente una población de n soluciones y calcular los valores de la función objetivo para todas las soluciones. El siguiente paso consiste en almacenar las soluciones en una población temporal T en la que se encuentran todas las soluciones no dominadas.

Este conjunto de soluciones no dominadas se denomina conjunto *B*, y se le asigna un valor de coste de 1. Cabe señalar que una solución *y* domina a una solución *x* si *y* es al menos tan buena como *x* para todos los objetivos, y es superior a *x* para al menos un objetivo. Se dice que una solución no está dominada si no hay ninguna *x* en la población que la domine.

A continuación, las soluciones de *B* se eliminan de *T*, y las soluciones no dominadas se obtienen del conjunto resultante de *T*. A estas soluciones se les asigna un valor de coste de 2. Este proceso iterativo continúa hasta que a todas las soluciones se les ha asignado un valor de coste basado en sus niveles no dominados. Posteriormente, los valores de coste se emplean para realizar las técnicas de recombinación y mutación que se deseen, repitiéndose este proceso en cada generación. Se puede observar en la Figura 9 el marco general de funcionamiento del algoritmo NSGA.

Randomly initialize a population of candidate solutions $P=\{x_j\}$ for $j\in[1, N]$
While termination condition is not satisfied **do**
    Temporary population $T \leftarrow P$
    Non-dominated level $c \leftarrow 1$
    While the size of temporary population $|T|>0$ **do**
        $B \leftarrow$ non-dominated solutions in T
        Cost $\phi(x) \leftarrow c$ for all $x \in B$
        Remove *B* from *T*
        $c \leftarrow c+1$
    End while
    $C \leftarrow N$ children created from recombining the solutions in *P*
    Probabilistically mutate the children in *C*
    $P \leftarrow C$
End while

Figura 9: Algoritmo NSGA [35]

A veces se critica la ineficacia del NSGA por el bucle interno de cálculo de los costes de las funciones, que supone la principal carga computacional.

2.5.1.2. NSGA-II

El NSGA-II es una versión modificada del NSGA surgida en 2002 [36]. NSGA-II evalúa el coste de una solución *x* evaluando su dominancia sobre otras soluciones y el hecho de estar dominado por ellas. Para cada solución, se calcula una distancia de apilamiento determinando la distancia a las soluciones más cercanas a lo largo de cada función objetivo. Esta distancia de apilamiento se emplea para ajustar el *fitness* de cada solución.

En el método de la distancia de apilamiento, para empezar, los valores mayor y menor más cercanos para cada función objetivo se calculan de la siguiente manera:

$$f_i^-(x) = \max_y \{f_i(y) \text{ such that } f_i(y) < f_i(x)\}$$
$$f_i^+(x) = \min_y \{f_i(y) \text{ such that } f_i(y) > f_i(x)\}$$

Ecuación 1: Cálculo de los valores mayor y menor más cercanos para cada función objetivo

Donde $f_i(x)$ es el i-ésimo valor de la función objetivo de x. Después, la distancia de apilamiento $d(x)$ se calcula:

$$d(x) = \sum_{i=1}^{k} \left(f_i^+(x) - f_i^-(x)\right)$$

Ecuación 2: Cálculo de la distancia de apilamiento d(x)

Las soluciones situadas en regiones densamente pobladas del espacio de la función objetivo tienden a tener una distancia de apilamiento menor. Las soluciones situadas en los valores extremos del espacio de la función objetivo tienen una distancia de apilamiento infinita.

Cada solución de la población incluye ahora una distancia de apilamiento que se utiliza como parámetro de ordenación secundario para determinar su rango. Al igual que en el NSGA, las soluciones se ordenan en función de su nivel de no dominación. Sin embargo, también se incluye una clasificación más detallada basada en la distancia de apilamiento. En otras palabras, se considera que *x* está mejor clasificado que *y* si el nivel de no dominación de *x* es menor que el de *y*, o si son iguales pero $d(x)$ es mayor que $d(y)$. A diferencia del NSGA, que utiliza el nivel de no dominación para elegir a los padres para la recombinación, el NSGA-II selecciona a los padres basándose en los rangos antes mencionados. Se puede observar en la Figura 10 [37] cómo se realiza la selección de individuos del NSGA-II. También se puede observar el flujo del algoritmo en la Figura 11.

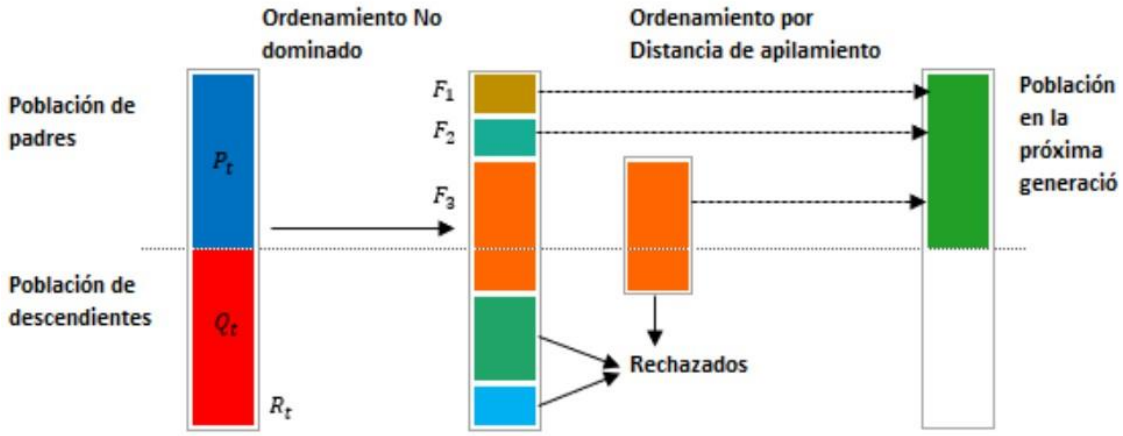

Figura 10: Selección de individuos NSGA-II [38]

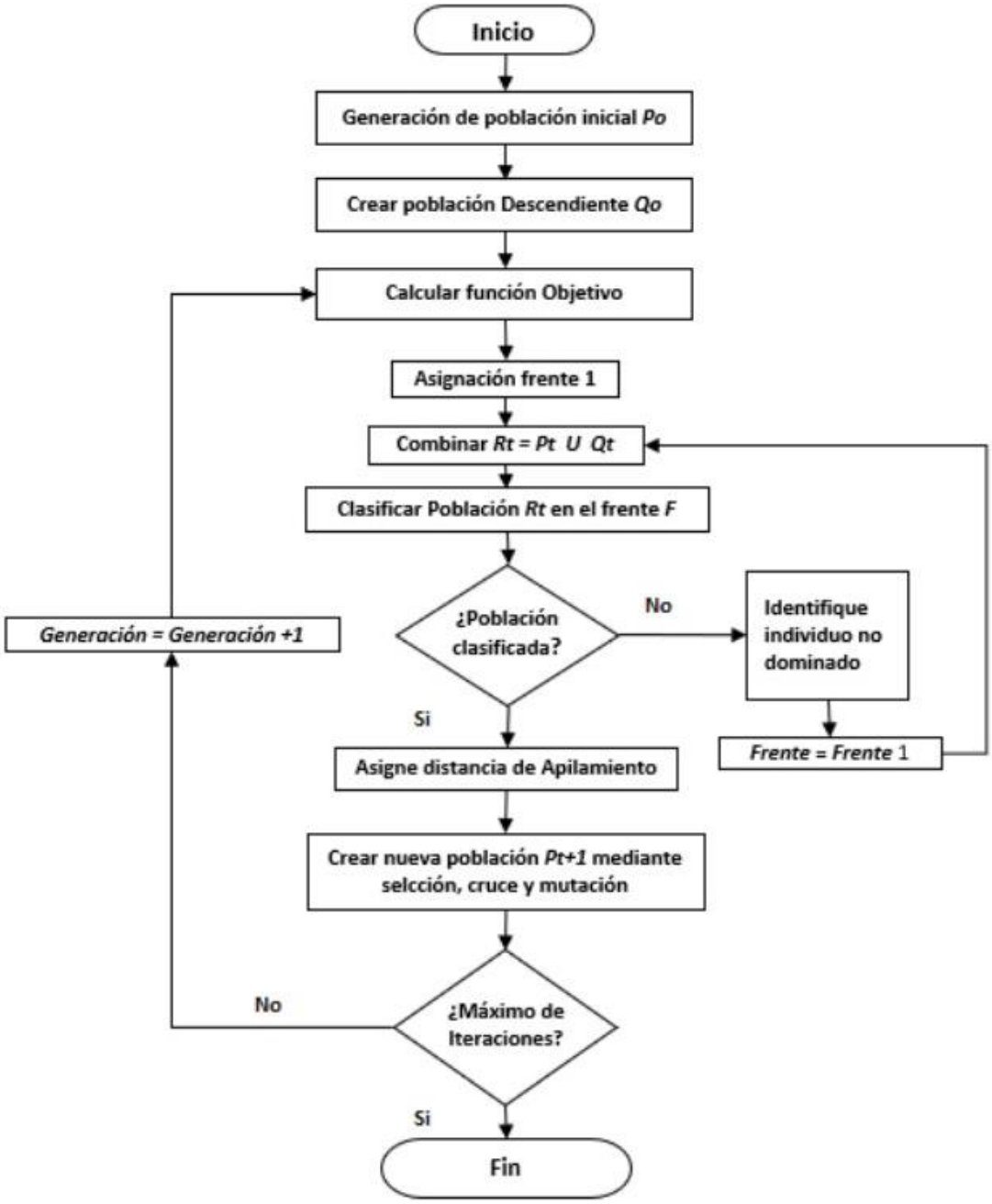

Figura 11: Flujo del algoritmo NSGA-II [39]

2.6. El uso de Python y la librería pymoo

Una implementación eficiente en un lenguaje de programación adecuado es esencial siempre que un algoritmo deba manejar una gran cantidad de datos. En los últimos años, Python se ha convertido en el lenguaje de programación preferido para estos casos, no solo por su facilidad de uso, sino también por el buen apoyo de su comunidad.

Python es un lenguaje de programación interpretado, multiplataforma y de alto nivel, que hace hincapié en la alta legibilidad del código. Dispone de numerosas librerías de alta calidad, que garantizan el soporte de todos los tipos de cálculo científico. Las propiedades de Python lo convierten en una herramienta adecuada para numerosos proyectos complejos de investigación e industria.

Sin embargo, la ejecución de algoritmos de optimización puede ser una tarea ardua, especialmente en lo que se refiere a la evaluación comparativa, que puede conllevar mucho tiempo. El acceso a una buena colección de diferentes códigos fuente o a una librería completa puede ahorrar tiempo y evitar errores que pueden surgir al ejecutar algoritmos desde cero.

Para poder afrontar la optimización multiobjetivo en Python, se va a utilizar la librería "pymoo" [40]. Esta librería proporciona algoritmos de optimización de última generación, y cubre varios aspectos relacionados con el propio proceso de optimización. Tiene implementados problemas de prueba de uno y múltiples objetivos, para que sirvan de banco de pruebas para los algoritmos. Además, la diferenciación automatizada pretende recuperar información sobre el gradiente, aparte de los valores objetivo y de restricción de los problemas. La evaluación paralela de las soluciones se realiza mediante cálculos vectoriales, ejecución multihilo y computación distribuida.

Además, pymoo ofrece indicadores de rendimiento para medir la calidad de los resultados obtenidos por un algoritmo de optimización multiobjetivo. Hay herramientas disponibles para el análisis exploratorio de datos de dimensiones inferiores y superiores mediante técnicas de visualización, y los métodos de toma de decisiones multicriterio ayudan a seleccionar una única solución entre un conjunto de soluciones en función de las preferencias del usuario.

2.6.1. Arquitectura de pymoo

La arquitectura del software tiene una importancia fundamental para mantener organizado el código. Facilita a los desarrolladores y a los usuarios de la librería la comprensión de las clases existentes, al tiempo que proporciona flexibilidad y extensibilidad mediante la incorporación de nuevos módulos.

La arquitectura de pymoo [40] es la que se puede ver en la Figura 12:

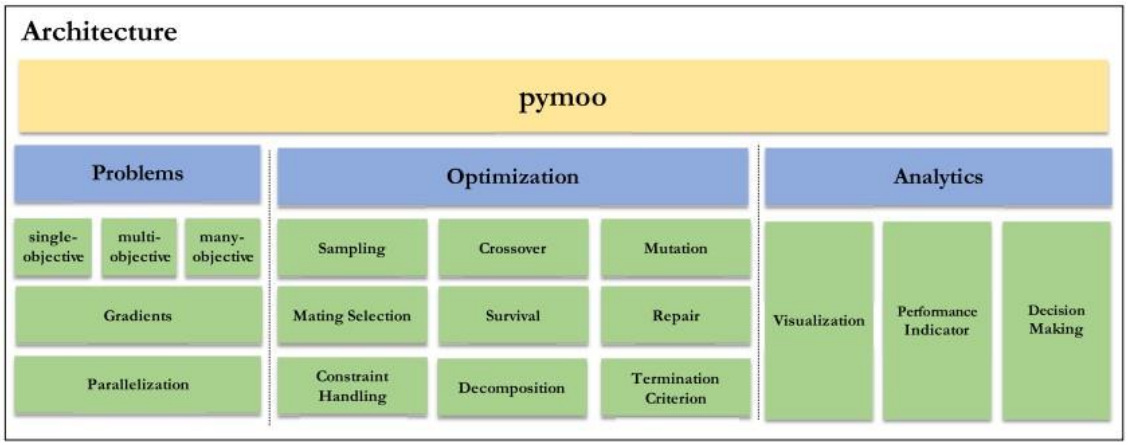

Figura 12: Arquitectura de pymoo [41]

1) Problemas: los problemas de optimización se clasifican como problemas de un único objetivo, de multiobjetivo o de muchos objetivos. La diferenciación automática proporciona también acceso a los gradientes, y pueden emplearse diversas técnicas para aplicar la paralelización.
2) Optimización: como la mayoría de los algoritmos se basan en cálculos evolutivos, como se ha comentado en puntos anteriores, requieren la selección o implementación de operadores como el muestreo, la selección, el cruce y la mutación. Además, las dificultades prácticas suelen plantear una o varias restricciones, por lo que debe incluirse una metodología para gestionarlas eficazmente. Algunos algoritmos se basan en la descomposición, en la que el problema multiobjetivo se divide en muchos problemas de un solo objetivo. Además, al emplear el algoritmo para resolver el problema, debe definirse un criterio de terminación, ya sea explícita o implícitamente a través de la implementación del algoritmo.
3) Análisis: El análisis ayuda a comprender los datos durante y después de un proceso de optimización. En primer lugar, el espacio de diseño, el espacio objetivo u otras métricas pueden explorarse fácilmente a través de la visualización. Además, se pueden utilizar indicadores de rendimiento para medir la convergencia y/o la diversidad de un conjunto Pareto-óptimo.

2.6.2. Optimización en pymoo

El módulo de optimización en pymoo ofrece varios tipos de submódulos que pueden emplearse en algoritmos. Algunos son genéricos, como la descomposición y los criterios de terminación, mientras que otros están más relacionados con la computación evolutiva. Para construir los distintos algoritmos, hay que ensamblar estos módulos.

2.6.2.1. Algoritmos

Los algoritmos implementados más importantes disponibles en pymoo son los que aparecen en la siguiente Tabla 5 [40]:

Algoritmo	Tipo de objetivo
GA	Un único objetivo
BRKGA	Un único objetivo
DE	Un único objetivo
Nelder-Mead	Un único objetivo
CMA-ES	Un único objetivo
NSGA-II	Multiobjetivo
RNSGA-II	Multiobjetivo
NSGA-III	Muchos objetivos
UNSGA-III	Muchos objetivos
RNSGA-III	Muchos objetivos
MOEAD	Muchos objetivos

Tabla 5: Tipos de algoritmos en pymoo

Cada algoritmo de optimización multiobjetivo se puede personalizar con variantes que se pueden inicializar utilizando diferentes parámetros. Cabe señalar que los algoritmos multiobjetivo requieren que se proporcionen direcciones de referencia, y por lo general, se desea que estas sean uniformes o tengan un sesgo hacia una región de interés.

2.6.2.2. Operadores

Los siguientes operadores evolutivos están disponibles para su uso y personalización:

1) Selección: En la mayoría de los casos, la población inicial se basa en la selección. A veces, puede crearse a través del conocimiento del dominio o evaluando algunas soluciones utilizadas directamente como población inicial. En otros casos, se pueden muestrear aleatoriamente variables reales, enteras o binarias.
2) Sobrecruzamiento: Se implementan una serie de operadores de cruce para distintos tipos de variables. Algunos de ellos se pueden ver en la Figura 13. De las figuras a) a la d) ofrecen una representación visual del intercambio de información durante un cruce entre dos progenitores. En cada fila se representa una descendencia, y cada columna representa una variable. Las casillas respectivas indican si los valores de la descendencia se heredan del primer o del segundo progenitor. En los cruces de uno y dos puntos, puede observarse que existe un único corte o dos cortes en la secuencia de variables. En cambio, el Cruce Uniforme (UX) no presenta ningún patrón claro, ya

que cada variable se selecciona aleatoriamente del primer o del segundo progenitor. La mitad de las variables que difieren se intercambian para el Cruce Medio Uniforme (HUX). El cruce binario simulado (SBX) es conocido por ser un método de cruce eficiente para variables reales. Este método imita el cruce de variables codificadas en binario.

3) Mutación: La mutación polinómica se emplea para variables reales y enteras, mientras que la mutación Bitflip se utiliza para variables binarias.

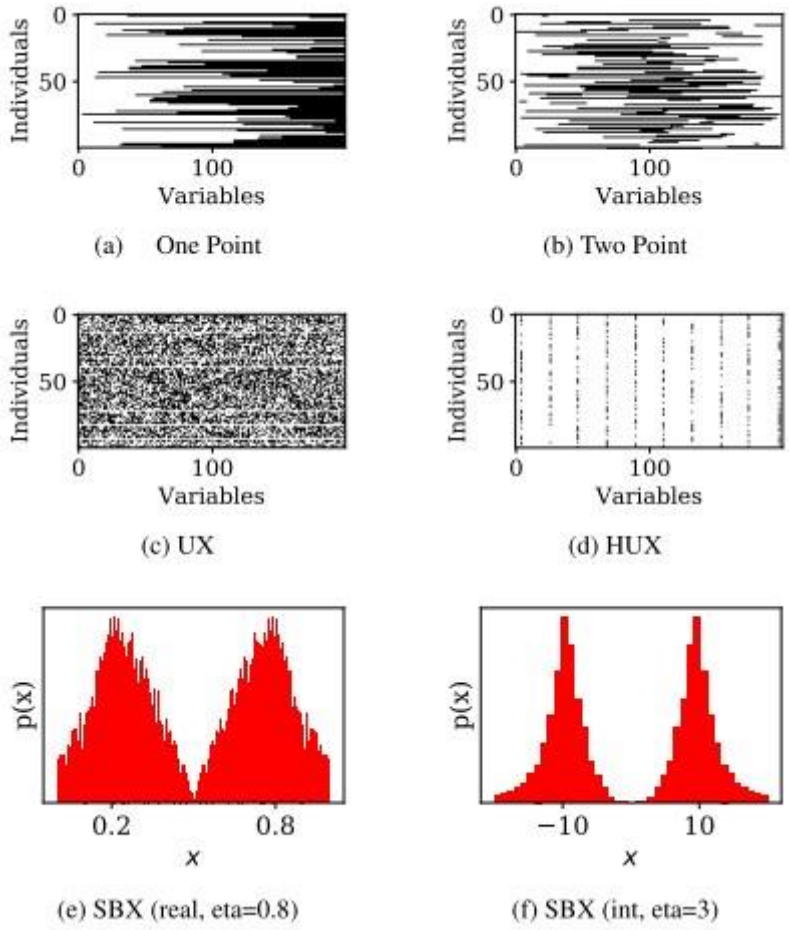

Figura 13: Operadores de cruce pymoo [41]

Pero hay que tener en cuenta que diversos problemas requieren tipos específicos de operadores. En la práctica, personalizar los operadores evolutivos podría mejorar la convergencia de los algoritmos al resolver problemas de forma rutinaria y repetida. Además, para tipos de variables personalizadas, como árboles o variables mixtas, se pueden implementar operadores personalizados, que luego pueden ser llamados por la clase del algoritmo.

2.6.2.3. Criterios de Terminación

Es necesario determinar cuándo terminar una ejecución para cada algoritmo. Esto puede basarse en un número predefinido de evaluaciones de funciones o iteraciones. También puede utilizarse un criterio más avanzado, como el cambio de una métrica de rendimiento a lo largo del tiempo.

Como ejemplo, viene implementado un criterio de terminación [40] que se basa entre los espacios variable y objetivo a lo largo de las generaciones. Para garantizar la solidez del criterio de terminación, se consideran las últimas *k* generaciones. Se hace un seguimiento del mayor movimiento desde una solución a su vecina más cercana a lo largo de las generaciones. Siempre que el movimiento esté por debajo de un cierto umbral, consideramos que el algoritmo ha convergido. Del mismo modo, podemos utilizar el movimiento en el espacio objetivo; sin embargo, la normalización es más difícil y requiere una cuidadosa consideración.

En pymoo, el criterio de terminación por defecto para los problemas multiobjetivo controla los puntos límites en el espacio objetivo. Una vez estabilizados, los puntos límite se utilizan para la normalización. Aunque en la mayoría de los casos, el criterio de terminación va a ir marcado por un número de iteraciones del algoritmo definido por el usuario

2.6.3. Análisis en pymoo

2.6.3.1. Indicadores de rendimiento

Comparar el rendimiento de los algoritmos de optimización de un solo objetivo es bastante sencillo, ya que cada ejecución de optimización da como resultado una única mejor solución. Sin embargo, en la optimización multiobjetivo, cada ejecución devuelve un conjunto no dominado de soluciones. A continuación, se describen los indicadores de rendimiento para comparar conjuntos de soluciones más utilizados en pymoo:

1) GD/IGD: La desviación entre el conjunto no dominado *S* encontrado por el algoritmo y el óptimo puede medirse utilizando el frente de Pareto. Basándose en este principio, el indicador de distancia generacional (GD) calcula la distancia euclídea media en el espacio objetivo entre cada solución en *S* y la solución más cercana en el frente de Pareto. Esto mide la convergencia de *S*, pero no si se ha logrado una buena diversidad en el frente de Pareto. Del mismo modo, el indicador de distancia generacional invertida (IGD) mide la distancia euclídea media en el espacio objetivo entre cada solución en el frente de Pareto y la solución más cercana en S. Para minimizar la métrica de rendimiento, el frente de Pareto debe estar cubierto en su totalidad por

soluciones de S. Por lo tanto, cuanto más bajos sean los valores de GD e IGD, mejor es el conjunto.

2) GD+/IGD+: Una versión alternativa de GD e IGD es el GD+ y el IGD+. En lugar de la distancia euclídea, se utiliza una medida de distancia que considera la relación de dominancia.
3) Hipervolumen: La sección dominada del espacio objetivo puede utilizarse para evaluar la calidad de las soluciones no dominadas. Un hipervolumen mayor indica un conjunto superior. Es necesario proporcionar un punto de referencia en lugar del frente de Pareto.

2.6.3.2. Herramientas de visualización

En la optimización multiobjetivo y de muchos objetivos, la visualización del espacio objetivo es interesante para experimentar fácilmente la información de compensación entre soluciones a partir de gráficos. En función de la dimensión del espacio objetivo, existen varios tipos de gráficos adecuados para representar una solución o un conjunto de soluciones.

En pymoo, las visualizaciones se basan en la popular biblioteca de Python matplotlib [42]. Se pueden utilizar las propias funciones de personalización de matplotlib para modificar por ejemplo el color o el grosor de líneas, puntos y formas de los gráficos. Como resultado, todas las técnicas de visualización son adaptables y pueden ampliarse para satisfacer necesidades específicas del usuario.

En el caso de dos o tres objetivos, los gráficos de dispersión o *scatter plots* pueden ofrecer una visión útil de la solución. Estos gráficos pueden ser en 2D o en 3D dependiendo del número de objetivos, como podemos ver en Figura 14. Las compensaciones se hacen evidentes observando la distancia entre dos puntos. Puede ser necesario normalizar cada objetivo para garantizar que la comparación entre valores se basa en una escala relativa y no absoluta.

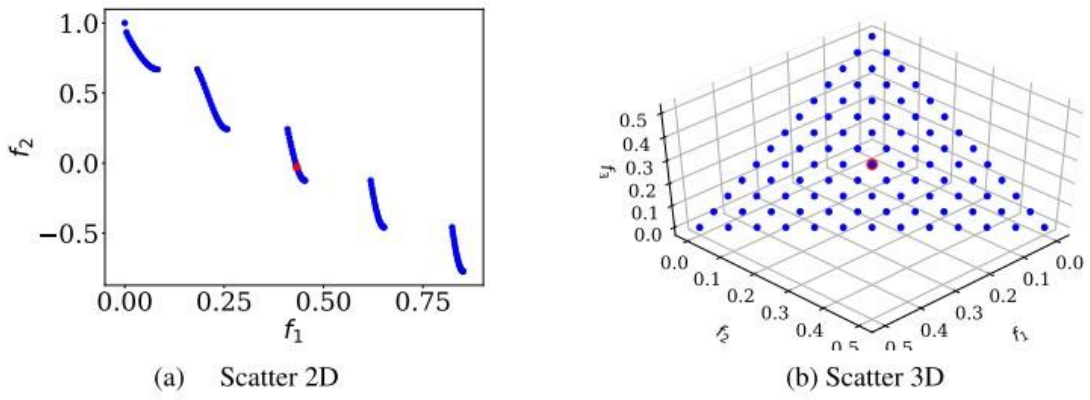

Figura 14: Gráficos de dispersión 2D y 3D [41]

Los gráficos de dispersión por pares muestran más de tres objetivos al mostrar cada par de ejes por separado. Los objetivos correspondientes están marcados en la diagonal.

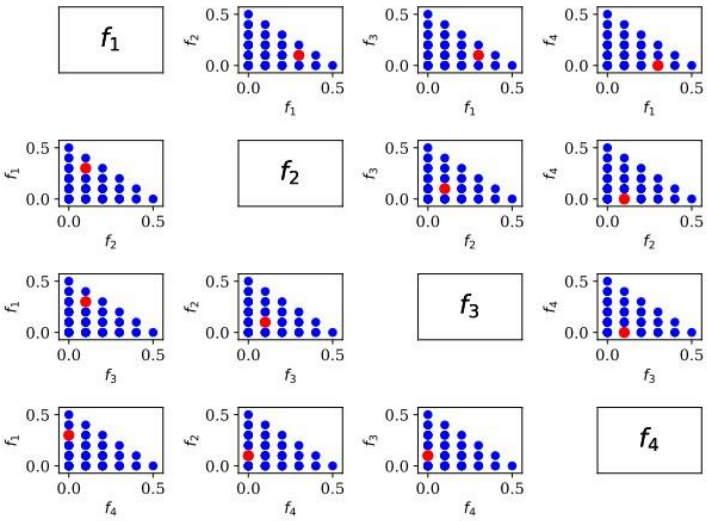

Figura 15: Gráfico de dispersión por pares [41]

Otra práctica habitual consiste en proyectar los valores objetivo de dimensiones superiores en un plano 2D mediante una figura de transformación. El método Radviz [43] ilustra todos los puntos dentro de un círculo, con los ejes de los objetivos colocados uniformemente alrededor del perímetro. Cuando se trata de un problema de minimización y un grupo de soluciones no dominadas, un punto extremo muy próximo a un eje significa la peor opción para el objetivo correspondiente, pero puede seguir siendo relativamente favorable en uno o varios objetivos más. Se puede ver un ejemplo en la Figura 16.

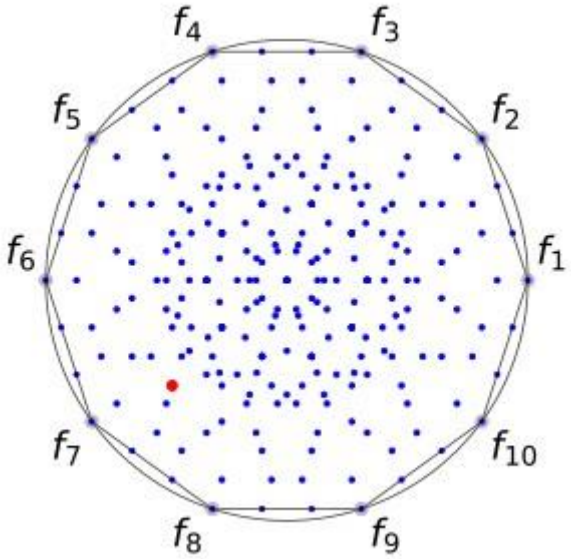

Figura 16: Radviz [41]

2.6.3.2. Toma de decisiones

En la práctica, al obtener un conjunto de soluciones no dominadas, hay que seleccionar una única solución para su aplicación. Pymoo ofrece un conjunto de métodos "a posteriori" para la toma de decisiones:

1) Programación por función de compromiso: un método para tomar una decisión consiste en calcular el valor de una función escalar y agregada, y elegir una solución basada en el valor mínimo o máximo de esta función.
2) Uso de pseudopesos: el enfoque del vector de pseudo-pesos proporciona un medio más intuitivo para seleccionar una solución de un frente de Pareto [44]. El pseudopeso w_i de la función objetivo i-ésima se calcula mediante la Ecuación 1. Se anota la distancia normalizada calculada para cada objetivo *i*. Una solución con un pseudopeso lo más cercano a un vector de preferencia de objetivos (por ejemplo un vector de preferencia (0,667 ; 0,333), donde f1 es dos veces más importante que d2), puede seleccionarse entonces como la solución preferida del conjunto eficiente.
3) Soluciones con grandes compensaciones: las soluciones con altas compensaciones suelen ser significativas pero difíciles de identificar en espacios objetivo de dimensiones más altas. El proceso que utiliza pymoo en estos casos es el propuesto por Rachmawati y Srinivasan [45]. La métrica asignada a cada par de soluciones x_i y x_j en un conjunto no dominado viene dada por la Ecuación 2, donde el numerador representa el sacrificio agregado y el denominador representa la ganancia agregada. Se calcula también la medida de compensación μ para cada solución x_i con respecto a un conjunto de soluciones vecinas *S*, mediante la Ecuación 3. A continuación se determina la $T(x_i,x_j)$ más baja desde x_i a todas las demás soluciones x_j contenidas dentro del espacio *S*. La complejidad computacional puede reducirse evaluando la métrica solo con los *k* vecinos más próximos en el espacio objetivo. Los operadores *min* pueden ser sustituidos por *max* u otro operador apropiado, dependiendo de la situación. A continuación, la solución con la medida de compensación μ más alta puede seleccionarse como la mejor solución. Esto significa que esta solución produce el mayor sacrificio en un valor objetivo por cada unidad de ganancia en otro valor objetivo, lo que la convierte en la solución más valiosa.

$$w_i = \frac{(f_i^{\max} - f_i(x)) \,/\, (f_i^{\max} - f_i^{\min})}{\sum_{m=1}^{M} (f_m^{\max} - f_m(x)) \,/\, (f_m^{\max} - f_m^{\min})}$$

Ecuación 1: Cálculo de pseudopeso

$$T(x_i, x_j) = \frac{\sum_{i=1}^{M} \max[0, f_m(x_j) - f_m(x_i)]}{\sum_{i=1}^{M} \max[0, f_m(x_i) - f_m(x_j)]}$$

Ecuación 2: Cálculo de T para cada par de soluciones

$$\mu(x_i, S) = \min_{x_j \in S} T(x_i, x_j)$$

Ecuación 3: Cálculo de la medida de compensación μ

3. ANÁLISIS DEL PROBLEMA

A continuación, se presentará el problema a abordar, y se examinarán las condiciones que el sistema propuesto debe satisfacer para lograr el objetivo planteado. Para esto, se definirá el alcance del problema, se reconocerá el contexto tecnológico y se listarán los requisitos necesarios.

3.1. Presentación del problema

El trabajo afronta el problema de optimizar la planificación de citas para una empresa de estudios clínicos. La empresa de estudios clínicos que ha proporcionado los datos es CEVAXIN, que es un centro de investigación médica que realiza ensayos clínicos y epidemiológicos. Desde su fundación en el 2013 en la República de Panamá, ha implementado y realizado numerosos estudios, ofreciendo soluciones innovadoras y procesos estandarizados para alcanzar resultados de alta calidad científica [46].

Los estudios clínicos, epidemiológicos y de salud pública abarcan todo tipo de enfermedades prevenibles por vacunación, y se realizan en todas las fases de investigación.

El objetivo del trabajo es ayudar a estos modelos de trabajo en empresas de estudios clínicos a optimizar la manera en la que se programan las distintas citas médicas, un trabajo que es manual y realizado por gente con mucha experiencia, pero que puede ser automatizado buscando mejores resultados y un ahorro de tiempo y esfuerzo.

Para ello se creará un programa que mediante el uso de algoritmos genéticos multi-objetivo con la librería pymoo de Python, consiga generar las programaciones de la manera óptima posible.

3.2. Alcance del problema

El modelo en el que estará basado el programa será sobre los datos dados por CEVAXIN, garantizando la seguridad y privacidad de los datos a lo largo de todo el alcance del proyecto.

3.3. Definición de los Requisitos

El formato con el que se expondrán los requisitos del problema es el siguiente:

- Identificador: se asigna un código a cada requisito. Estos códigos se referirán a los requisitos funcionales como RF y a los no funcionales como RNF.

- Nombre: se asigna un nombre identificativo que representa el contexto del requisito
- Prioridad: indica el grado de importancia con respecto al resto de requisitos.
- Necesidad: indica el grado de importancia con respecto al conjunto del problema.
- Descripción: se describe en qué consiste el requisito.

Se van a definir los requisitos con el siguiente diseño de tabla:

Identificador	
Nombre	
Prioridad	
Necesidad	
Descripción	

Tabla 6: Modelo de requisitos

3.3.1. Requisitos Funcionales

Identificador	RF1
Nombre	Ingreso de citas
Prioridad	Alta
Necesidad	Esencial
Descripción	El programa debe permitir la entrada de citas médicas que la clínica necesita asignar durante la semana.

Tabla 7: RF1

Identificador	RF2
Nombre	Implementación de algoritmos genéticos
Prioridad	Alta
Necesidad	Esencial
Descripción	El programa debe implementar algoritmos genéticos multiobjetivo para encontrar soluciones subóptimas.

Tabla 8: RF2

Identificador	RF3
Nombre	Visualización de soluciones
Prioridad	Alta
Necesidad	Esencial
Descripción	El programa debe mostrar al usuario un conjunto de soluciones subóptimas generadas por el algoritmo.

Tabla 9: RF3

Identificador	RF4
Nombre	Selección y visualización de solución
Prioridad	Media
Necesidad	Deseable
Descripción	El programa debe ofrecer una solución subóptima y visualizarla para que sea más fácil para el usuario ver el resultado.

Tabla 10: RF4

Identificador	RF5
Nombre	Configuración de parámetros del algoritmo
Prioridad	Media
Necesidad	Deseable
Descripción	El programa debe permitir al usuario configurar ciertos parámetros del algoritmo genético multiobjetivo, como la tasa de mutación, tamaño de la población y número de generaciones, para adaptar la búsqueda según sus necesidades.

Tabla 11: RF5

3.3.2. Requisitos No Funcionales

Identificador	RNF1
Nombre	Tiempo de respuesta
Prioridad	Alta
Necesidad	Esencial
Descripción	El programa debe ser capaz de generar soluciones subóptimas en un tiempo razonable, no superando los 5 minutos.

Tabla 12: RNF1

Identificador	RNF2
Nombre	Usabilidad
Prioridad	Media
Necesidad	Deseable
Descripción	La interfaz del programa debe ser intuitiva y fácil de usar.

Tabla 13: RNF2

Identificador	RNF3
3Nombre	Seguridad
Prioridad	Alta
Necesidad	Esencial
Descripción	Los datos ingresados en el programa deben ser tratados de manera segura, evitando cualquier tipo de vulnerabilidad.

Tabla 14: RNF3

Identificador	RNF4
Nombre	Mantenibilidad
Prioridad	Baja
Necesidad	Opcional
Descripción	El código del programa debe estar bien estructurado y documentado para facilitar futuras modificaciones o correcciones.

Tabla 15: RNF4

3.3.3. Casos de prueba

Se van a definir un conjunto de casos de prueba para comprobar que se cumplen todos los requisitos definidos, tanto funcionales como no funcionales. Los casos de prueba son los siguientes:

Identificador	**Descripción**
Caso de Prueba 1 – CP1	Validar que el sistema permite ingresar citas médicas en un tiempo razonable y de forma segura.
Caso de Prueba 2 – CP2	Validar que el sistema implementa algoritmos genéticos y permite su configuración, todo en un tiempo razonable.
Caso de Prueba 3 – CP3	Validar que el sistema muestra y permite la selección de soluciones subóptimas de forma intuitiva y fácil de usar.

Caso de Prueba 4 – CP4	Medir el tiempo de respuesta del sistema, evaluar la usabilidad y verificar las medidas de seguridad.
Caso de Prueba 5 – CP5	Validar el ingreso de citas y la configuración de algoritmos, y evaluar la estructura y documentación del código.

Tabla 16: Casos de Prueba

3.3.4. Matriz de trazabilidad

Se va a mostrar la matriz de trazabilidad que relaciona cada caso de prueba con los requisitos que se ven afectados:

ID	RF1	RF2	RF3	RF4	RF5	RNF1	RNF2	RNF3	RNF4
CP1									
CP2									
CP3									
CP4									
CP5									

Tabla 17: Matriz de trazabilidad

3.4. Contexto tecnológico

Python

Python es un lenguaje de programación de alto nivel, interpretado, con una sintaxis clara y legible que favorece la productividad y la facilidad de aprendizaje [47]. Es ampliamente utilizado para desarrollo web, análisis de datos, inteligencia artificial y muchas otras aplicaciones. Es el lenguaje de programación elegido para el desarrollo del programa.

PyCharm

PyCharm es un entorno de desarrollo integrado (IDE) para Python, desarrollado por JetBrains, que ofrece herramientas avanzadas para la programación, depuración y pruebas de código Python [48]. Es el IDE elegido para el desarrollo del programa.

Pymoo

Pymoo es una biblioteca de Python para optimización multiobjetivo que ofrece algoritmos y herramientas para resolver problemas complejos de optimización con múltiples criterios de decisión, como ya se ha desarrollado con amplitud en puntos anteriores del trabajo [41].

NumPy

NumPy es una biblioteca de Python para operaciones matemáticas que proporciona soporte para arrays multidimensionales y ofrece funciones matriciales de alto rendimiento [49].

Pandas

Pandas es una biblioteca de Python para manipulación y análisis de datos que ofrece estructuras de datos flexibles, como DataFrame, para trabajar con datos tabulares de manera eficiente [50].

Matplotlib

Matplotlib es una biblioteca de Python para la visualización de datos que permite la creación de gráficos estáticos, animados e interactivos en diversos formatos [51].

Excel

Se utilizarán como base para el estudio del problema dos archivos Excel proporcionados por CEVAXIN en los que hay información sobre la estructura real de sus centros, así como de las fases de sus distintos estudios clínicos.

Equipo usado para el desarrollo del programa y ejecución de modelos

Para realizar el estudio, se ha utilizado un ordenador personal con Windows 11 Pro. El ordenador está compuesto por un procesador AMD Ryzen 7 5700G 3800 Mhz, memoria RAM 16 GB, y una tarjeta gráfica NVIDIA GeForce RTX 3060 Ti.

4. DESARROLLO DEL PROBLEMA

A continuación, se plasmarán todos los pasos del diseño y desarrollo de los algoritmos implementados. Se seguirán los siguientes pasos:

- Configuración de la entrada para el problema
- Proceso de evaluación
- Elección y configuración del algoritmo
- Resultados

4.1. Configuración de la entrada para el problema

El primer paso a la hora de desarrollar el algoritmo era determinar cuál iba a ser el tipo de entrada que se deseaba para el algoritmo. Esto es un paso esencial para el buen desarrollo del proyecto, ya que hay que determinar el cómo se quiere desarrollar la planificación, si en base a los huecos disponible, el personal o a las citas que se tienen.

Basándose en la información proporcionada por CEVAXIN, se determinan que hay distintos tipos de actividades o fases por las que tiene que pasar un paciente cuando se enfrenta a un estudio clínico. En el caso de que fuera a una consulta médica normal simplemente habría que planificar la cita sin tener en cuenta más eventos, pero en el caso de un estudio clínico para el mismo estudio hay muchas fases distintas, como se puede observar en la Figura 17.

Recepción	R
Consentimiento	Cs
Muestra embarazo	ME
Consulta médica	CM
Laboratorio	LB
Vacunación	Vc
Observación	Obs
Procesos de cierre	PC

Figura 17: Fases de un estudio clínico

Por otro lado, se deben tener en cuenta los recursos disponibles que se tienen en la clínica. Estos recursos son tanto físicos como de personal. Dentro de los recursos físicos se van a tener en cuenta los diferentes consultorios que hay dentro de la clínica, en este caso 6 consultorios, como viene en los datos dados por CEVAXIN de la clínica de Avenida de México.

También hay que tener en cuenta como recurso "físico" los recursos temporales, es decir cuál es el horario de la clínica. En el caso de las clínicas de CEVAXIN tienen un horario de 7 de la mañana a 22 de la noche, con lo cual ese es el rango de horas en las cuáles las actividades pueden ser asignadas.

En el caso de los recursos humanos, todos los que participan dentro de esa misma clínica se pueden ver en la Figura 18. Pero para el desarrollo de este programa únicamente se va a hacer uso de Médico general, Médico seguridad, Enfermería, Laboratorista, asignando cada una de las fases anteriores a cada tipo de personal, como se puede ver en la Figura 19.

Médico general	MD
Médico vigilancia/seguridad	MV
Médico de observación	MO
Enfermera vacunadora	EV
Laboratorista	LB
Personal RAU	RAU
Asistente clínica	AC
Analista de datos	AD

Figura 18: Recursos Humanos Clínica

Fase	**Personal encargado**
R	Enfermería
Cs	Enfermería
ME	Médico general
CM	Médico general
LB	Laboratorista
Vc	Médico seguridad
Obs	Enfermería
PC	Médico general

Figura 19: Relación Fases Estudio con su Personal Asignado

Cada una de las fases deberá tener asignada una duración predefinida, pero para este programa en particular se decidió que todas tuvieran una duración base de 1 hora.

La entrada del problema contará con un número *x* de estudios clínicos, con un nombre definido como puede ser “COVID” o “Gripe”, que servirá como identificador para todas las actividades relacionadas con dicho estudio.

Para cada uno de los estudios, se asignarán todas las fases que se han establecido previamente, y esas serán las variables a las que se tendrán que asignar los recursos, como se va a explicar a continuación.

Con toda esta información definida, debe inicializarse el problema. Para ello hay que definir las variables del problema y el número de funciones objetivo que va a tener. Para definir las variables del problema, hay que establecer relaciones entre los recursos que hay disponibles y las distintas fases de cada estudio.

Para ello, se van a establecer en un primer lugar todas las posibles combinaciones de consultorio, horas y personal. Es decir, va a haber una combinación para cada consultorio, con una hora y un personal distintos. Esto permite que luego cada una de las fases de los

estudios tenga la opción de ser asignada a cualquier hora en cualquier consultorio y con cualquier personal.

El formato que tiene la combinación de recursos es el siguiente:

```
'Room: {self.operation_room} ST: {self.start_time} ET: {self.end_time} Personal: {self.personal}'
```

Mientras que el formato que tendrá cada actividad es:

```
'Estudio: {self.estudio} Tipo: {self.tipo_actividad} Personal: {self.personal} Duration: {self.duration}'
```

Cada una de las actividades tendrá asignada una combinación de recursos entre todas las que hay, e iterando a partir de eso y mediante la función de evaluación del algoritmo, se decidirá cuál es la mejor combinación de actividades y recursos para las funciones objetivo que el usuario defina.

```
for i in self.consultorios:
    for j in self.personal:
        for k in self.horas:
            for l in self.horas:
                if int(k) < int(l) and (int(l) - int(k)) <= int(max(self.duracion_actividad.values())):
                    actividad = Actividad(i,k,l,j)
                    self.combinations.append(actividad)
```

Por lo tanto, para iniciar el problema, se establecerán todo el conjunto de posibles combinaciones de actividades con recursos, como el espacio de variables, y también el número de funciones objetivo que luego serán definidas en el proceso de evaluación.

4.2. Proceso de evaluación

Una vez se ha definido la inicialización del problema se pasa a definir el proceso de evaluación que sufrirá el conjunto de soluciones en cada ciclo del algoritmo. Este consiste en un conjunto de restricciones y funciones objetivo que vienen descritas a continuación.

4.2.1. Restricciones

En pymoo existe la posibilidad de que al definir el problema puedas añadir al igual que el número de funciones objetivo también el número de restricciones, y gestionar estas mediante su propia configuración. Pero después de realizar pruebas, se decidió utilizar otro método que es mediante la aplicación de penalizaciones.

El método de aplicar penalizaciones es sencillo de entender, simplemente se definen restricciones sobre tu espacio de soluciones, y cuando se incumple una de ellas, se aplica una penalización (puede variar el tamaño de la penalización según interprete el desarrollador el nivel de gravedad de la restricción). Las penalizaciones son un valor acumulativo a la hora de evaluar una solución, y que luego se añade al calcular el valor de las funciones objetivo.

Por ejemplo, en un problema se tienen 3 restricciones y se incumplen 2 de ellas, cada una de ellas aplicando 100 puntos de penalización, pues al calcular la función objetivo tendrá un valor exagerado, lo que hace que esa solución sea mucho peor que en el caso de una solución que no incumpla ninguna restricción.

En el caso de este estudio, se aplicará esa penalización extra por igual a cada una de las funciones objetivo, para no aplicar un sesgo a las mismas. De esta manera los resultados que se obtengan al final de la ejecución serán aquellos que cumplan todas las restricciones, ya que el valor de las funciones objetivo no se verá alterado por estas restricciones.

Las restricciones que se han decidido implementar son las siguientes:

- Restricción 1: No puede haber dos actividades en la misma sala al mismo tiempo. Esta restricción garantiza que las actividades no se solapen en la misma sala, comprobando para cada sala si hay cirugías que tienen la misma hora de inicio. En caso afirmativo, se añade una penalización de 100 a la penalización global por cada cirugía solapada.
- Restricción 2: Todas las actividades deben tener la duración correcta. Para hacer cumplir esta restricción, el código comprueba que, para cada actividad, si restas la hora de fin y la de inicio, coincide con la duración prevista para ese tipo de actividad. Si no es así, se añade una penalización de 100 a la penalización global por cada actividad que dure más o menos de lo debido.
- Restricción 3: Todas las actividades deben estar asignadas al personal adecuado. Esta restricción garantiza que el personal asignado a cada actividad es el correcto y definido previamente en las condiciones iniciales. Si no es así, se añade una penalización de 100 a la penalización global.
- Restricción 4: Ningún miembro del personal puede tener dos actividades al mismo tiempo. Para hacer cumplir esta restricción, el código extrae todas las actividades para cada miembro del personal y comprueba que ninguna se solapa. Si no es así, se añade una penalización de 100 a la penalización global por cada cirugía solapada.
- Restricción 5: Las actividades deben ser completadas en el orden correcto. Esta restricción garantiza que hasta que no se acabe la anterior fase en el ciclo de un estudio no empiece la siguiente. Esto es imprescindible a la hora de planificar estudios clínicos, ya que las fases no son independientes entre ellas. Si no se cumple esta restricción, se añade una penalización de 100 a la penalización global por cada cirugía mal ordenada.
- Restricción 6: Reducir el número de horas vacías. Esta restricción trata de castigar aquellas soluciones que presenten una programación con horas vacías en todos sus consultorios, evitando así que se pierda tiempo útil de uso de esos consultorios. Si el programa encuentra una hora vacía, aplicará una penalización de 100 a la penalización global.

Estas restricciones finalmente elegidas son generales, ya que son restricciones vitales para cualquier estudio clínico o posible alteración de las condiciones iniciales. Se probaron

otras ideas, como restringir el número de actividades que podía hacer cada miembro del personal al día, pero al final sesgaban demasiado las posibles soluciones y hacían que el algoritmo multiobjetivo perdiera calidad tanto en tiempo de ejecución como en número de ciclos para llegar a buenas soluciones.

Que sean tan generales y básicas implica que las soluciones subóptimas que acaba generando el algoritmo nunca incluyen ninguna restricción incumplida, ya que aprende de manera adecuada y evita cometerlas, quedándose solo con los conjuntos de soluciones que vengan limpios en cuanto a penalizaciones.

4.2.2. Funciones objetivo

Las funciones objetivo son funciones matemáticas empleadas para evaluar la calidad de una solución a un problema de optimización. En el contexto de este programa, las funciones objetivo valoran la calidad de las programaciones generadas por el algoritmo de optimización.

Las funciones objetivo que se han decidido implementar son las siguientes:

- Función objetivo 1: Minimizar la duración total de todas las actividades. Es decir, calcular la hora a la que se van a finalizar todas las actividades asignadas para ese día de la programación. Para ello se coge la mayor de las horas de finalización de cada uno de los consultorios. Minimizar esta hora ayuda a reducir costes y reducir las horas que cada uno de los empleados tiene que trabajar, pudiendo quizá dedicar el resto de horas laborables de la clínica para urgencias de cualquier tipo u otras actividades.
- Función objetivo 2: Minimizar el tiempo total de espera de los pacientes. Una mejor conexión entre las actividades garantizará un flujo lógico para los pacientes, reduciendo su tiempo de espera. El tiempo de espera se calcula como el tiempo transcurrido entre el inicio de una actividad y el final de la actividad anterior en la misma sala.

El algoritmo de optimización intentará encontrar una planificación que equilibre estos dos objetivos. Se han tenido en cuenta también añadir más funciones objetivo, como por ejemplo intentar minimizar el número de consultorios que estarían en funcionamiento, o minimizar el personal que se debe utilizar, pero al final se decidió no implementarlas, ya que las dos funciones objetivo elegidas tienen como prioridad mejorar la experiencia para el paciente o miembro de los estudios clínicos, lo que conllevaría la mayor mejora a largo plazo.

4.2.3. Métricas de rendimiento

Como métrica de rendimiento en el análisis de las posibles soluciones se van a obtener las soluciones subóptimas en base a la combinación del valor total de penalización con el valor de las funciones objetivo.

Teniendo todo esto en cuenta, una vez se concluye el proceso de optimización, se obtiene un conjunto de soluciones subóptimas al problema inicial, pero, ¿cómo se puede seleccionar una solución única que implementar?

El enfoque de la función de satisfacción aproximada ("Achievement Scalarazing Function" ASF) es una técnica utilizada para seleccionar una solución única a partir de un conjunto de soluciones en el Frente de Pareto.

El método ASF convierte el problema multiobjetivo en un problema de optimización escalar mediante una función escalarizante. La forma más común de la función ASF es:

$$\mathrm{ASF}(\mathbf{x}, \mathbf{w}, \mathbf{z}^*) = \max_{i=1}^{n} \left[w_i \left(f_i(\mathbf{x}) - z_i^* \right) \right]$$

Ecuación 4: Función ASF

Donde:

- *x* es la solución bajo consideración
- *w* es un vector de pesos que refleja la importancia relativa de cada objetivo, este vector es elegido por el usuario, dándole la importancia que considere a cada función objetivo
- *z** es el Punto Ideal, que contiene los mejores valores posibles para cada objetivo
- $f_i(x)$ es el valor del i-ésimo objetivo para la solución *x*

La idea es minimizar la función ASF para encontrar una solución que ofrezca un buen equilibrio entre los diferentes objetivos, teniendo en cuenta los pesos y el Punto Ideal.

A continuación, se analizan los puntos tanto Ideal como Nadir del frente de Pareto. El punto Ideal es un vector hipotético que contiene los mejores valores posibles para cada objetivo, mientras que el punto Nadir es el opuesto conceptual, que contiene los peores valores posibles. Estos puntos Nadir e Ideal se utilizan para normalizar los valores de los objetivos, para asegurar así que todos tengan la misma escala y, por lo tanto, el mismo impacto en la función ASF.

También a estos valores se le puede aplicar un vector de pesos, que determina la importancia que el usuario quiere darle a cada una de las funciones objetivo, dándole un valor de peso más alto a aquella función que sea más relevante para la solución final.

Después de normalizar los valores y aplicar los pesos, se calcula el valor de la función ASF para cada una de las soluciones sub-óptimas obtenidas. Y con estos valores, en el caso de que el algoritmo busque la minimización, el usuario elegirá como solución final el valor más pequeño de la función ASF, en el caso de maximización, el valor más alto.

4.3. Enfoque 1: Programación diaria

En primer lugar se va a hacer un primer análisis del problema para una planificación diaria. Es decir, se tienen dos estudios distintos que tienen que completarse en una jornada de trabajo, y se va a aplicar el algoritmo para optimizar la planificación de las actividades a lo largo de la jornada de 7 a 22 y con los consultorios y personal definidos para ese día.

4.3.1. Configuración del Algoritmo

Para la implementación del algoritmo NSGA-II el entorno pymoo es necesario adaptar varias configuraciones específicas relacionadas con las variables y parámetros del problema que se aborda. La configuración del algoritmo se ilustra en la Ilustración 21.

```
algorithm = NSGA2(pop_size=100,
                  sampling=MixedVariableSampling,
                  mating=MixedVariableMating(eliminate_duplicates=MixedVariableDuplicateElimination()),
                  mutation=PolynomialMutation(prob=0.1),
                  eliminate_duplicates=MixedVariableDuplicateElimination(),
                  )
```

Figura 20: Declaración Algoritmo NSGA-II

- Tamaño de la Población: Se ha configurado el tamaño de la población en 100 individuos. Este número determina la cantidad de soluciones candidatas que el algoritmo generará y evaluará en cada ciclo o iteración. Un tamaño de población mayor puede permitir una exploración más exhaustiva del espacio de soluciones, pero también aumentará la carga computacional.
- *Sampling* (método de muestreo): Para el muestreo inicial de la población, se utiliza la clase *MixedVariableSampling* proporcionada por pymoo. Esta clase es particularmente útil para problemas que involucran una combinación de variables continuas y discretas. El método de muestreo es fundamental para establecer un punto de partida diverso para el algoritmo, lo cual puede influir en la calidad de las soluciones encontradas.
- *Mating* (método de cruce): El método de cruce se especifica mediante la clase *MixedVariableMating*. Al igual que con el muestreo, se utiliza una clase adaptada para variables mixtas debido a la naturaleza del problema que estamos resolviendo.
- Mutación: Se ha utilizado la clase *PolynomialMutation* con una probabilidad del 0.1 para la mutación. En el contexto de algoritmos genéticos, la mutación introduce pequeños cambios aleatorios en las soluciones candidatas. Esto se hace para evitar el estancamiento en mínimos locales y permitir al algoritmo explorar más ampliamente el espacio de búsqueda.
- Eliminación de Duplicados: Para garantizar que todas las soluciones en la población sean únicas, se utiliza la clase *MixedVariableDuplicateElimination*. Esta clase identifica y elimina soluciones duplicadas, lo que ayuda a mantener la diversidad dentro de la población y mejora la eficiencia del algoritmo.

Con todas estas configuraciones en su lugar, se ejecuta el algoritmo NSGA-II con el objetivo de minimizar las funciones objetivo definidas. Pymoo ofrece diversas estrategias para detener la ejecución del algoritmo, tales como criterios basados en la convergencia o en el tiempo de ejecución. Sin embargo, en este caso, ninguno de estos criterios era adecuado para las condiciones específicas del problema. Por lo tanto, la terminación del algoritmo se basa únicamente en el número de iteraciones que se hayan preestablecido.

4.3.2. Detalles Implementación Diaria

Para la implementación diaria, se busca planificar 2 estudios completos, COVID y Gripe, cada uno con todas las fases comentadas anteriormente. También se obtendrá como entrada los trabajadores disponibles para ese día, que son 7 médicos generales, 3 médicos de seguridad, 7 enfermeros y 3 laboratoristas, cada uno con su propio ID identificativo y su rol, así como los 5 distintos consultorios que pueden ser utilizados para cada una de esas fases. Por último, el horario de la clínica será de 7 a 22, con lo que podrán ser establecidas actividades en cada una de esas horas para cada uno de los consultorios.

En cuanto al algoritmo, se harán uso de todas las restricciones planteadas con anterioridad y también las dos funciones objetivo: minimizar la duración total de todas las actividades y minimizar el tiempo de espera de los pacientes.

4.3.3. Análisis de resultados

Se llevaron a cabo 20,000 iteraciones del algoritmo para calcular los resultados, proceso que tomó aproximadamente 5 minutos en nuestro equipo de computación. Es importante señalar que el tiempo de ejecución puede variar según la capacidad computacional del hardware utilizado.

Tras concluir las 20,000 iteraciones, se obtuvo un conjunto de soluciones subóptimas, específicamente: [[19, 62], [18, 84], [21, 45]]. En cada par de valores, el primer número representa el valor de la primera función objetivo *f1*, mientras que el segundo número representa el valor de la segunda función objetivo *f2*.

Para una comprensión más profunda de estas soluciones, se recurrió a una representación gráfica, como se ilustra en la Figura 21.

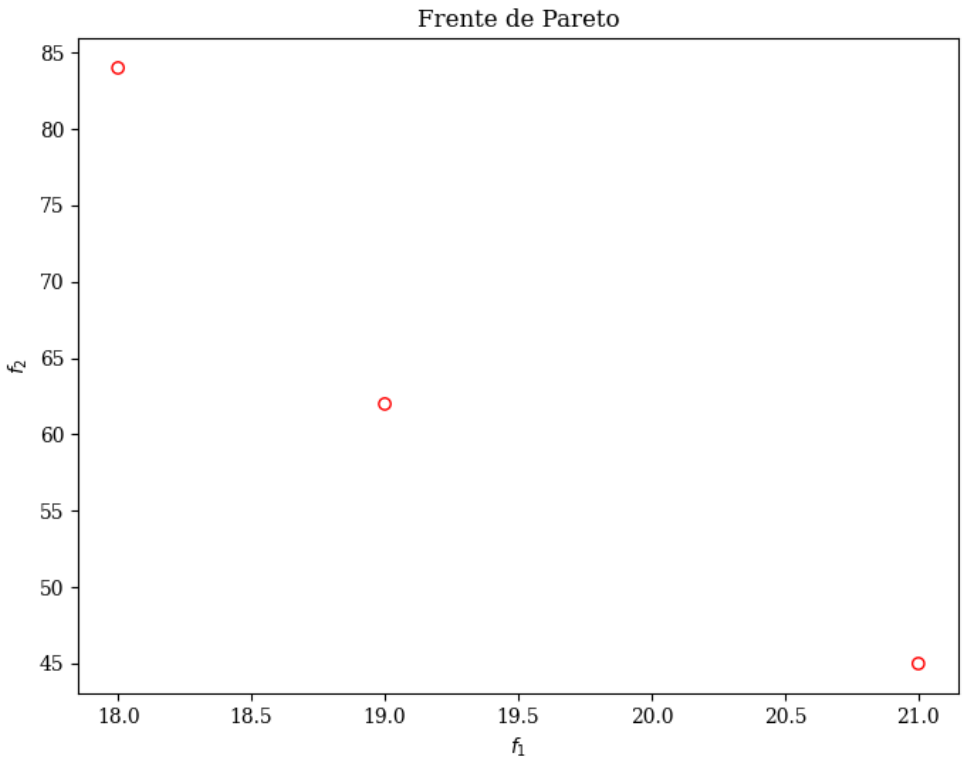

Figura 21: Frente de Pareto Resultados

En la gráfica, es posible observar que las soluciones se distribuyen formando un Frente de Pareto. En este frente, la esquina inferior derecha representa la solución que tiene el peor desempeño respecto a f1, mientras que la esquina superior izquierda muestra la solución con el peor rendimiento en *f2*. Es crucial destacar que todas las soluciones son parte del Frente de Pareto, lo que indica que cada una de ellas supera a las demás en al menos uno de los objetivos formulados.

Para analizar las distintas soluciones subóptimas se va a recurrir a calcular los valores ASF para cada solución y a los pesos elegidos. Los pesos elegidos serán (0.7,0.3). Esto implica que la *f1* tendrá una importancia del 70% y la *f2* de un 30%, esto es porque se ha decidido dar una mayor importancia a que todas las actividades sean acabadas cuando antes.

Los Puntos Nadir e Ideal aproximados para el conjunto de soluciones son los siguientes:

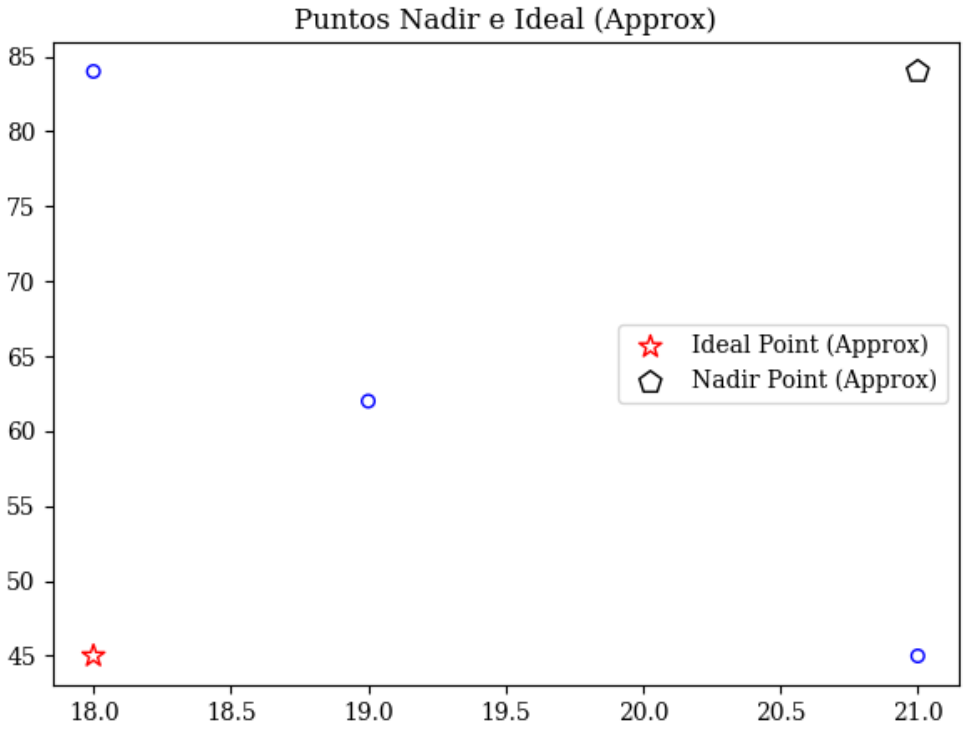

Figura 22: Puntos Nadir e Ideal

Siendo el Punto Nadir el (21,84) y el Punto ideal (18,45), se normalizan los valores de la solución, y se calculan los valores de la función ASF:

- Para la solución (19,62), el valor de ASF es 0.233
- Para la solución (18,84), el valor de ASF es 0.3
- Para la solución (21,45), el valor de ASF es 0.7

Por lo tanto, la solución seleccionada por ASF después de la normalización es la (19,62), que es la que la minimiza, como se puede ver en Figura 23:

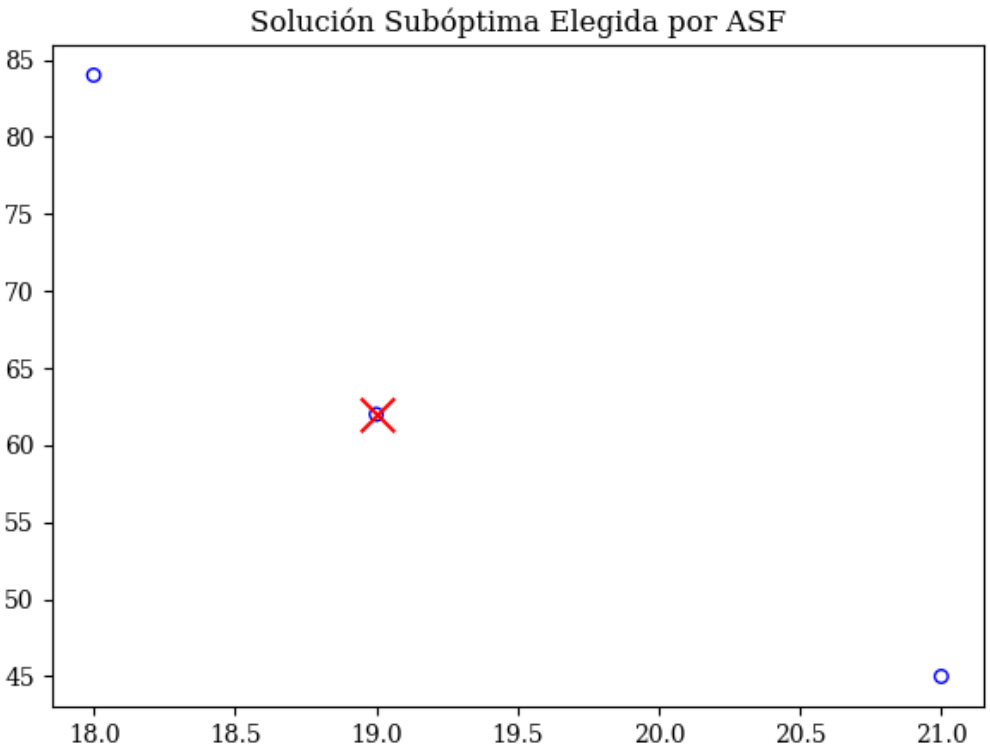

Figura 23: Solución Subóptima Elegida por ASF

Esta solución elegida implica que la planificación del día para ambos estudios acabará a las 19 de la tarde, y que el número total de horas vacías de los consultorios, que implican que hay una espera de pacientes, es de 62 horas. Estos resultados cumplen también con todas las restricciones definidas para el problema, ya que ambas acarrean 0 penalización.

Por lo tanto, la programación que se corresponde con la solución subóptima elegida es la siguiente Figura 24:

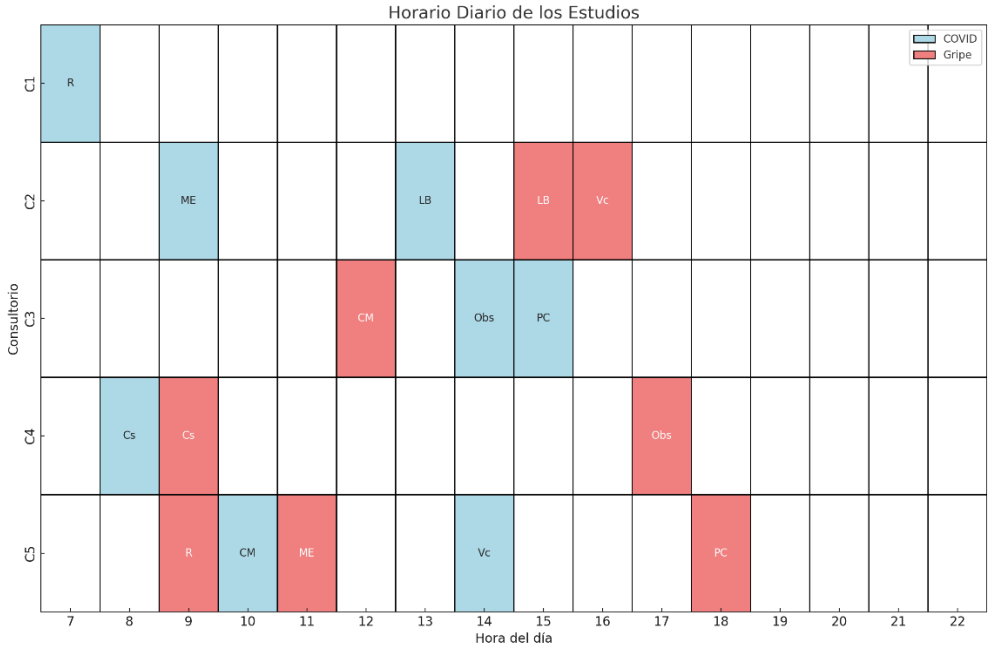

Figura 24: Programación Elegida Diaria

Como se puede observar en la Figura, esta es la configuración elegida para cada consultorio y cada estudio, empezando a las 7 de la mañana con la Recepción del COVID y acabando a las 19 con el Proceso de Cierre de Gripe.

4.3.4. Ventajas y desventajas del enfoque diario

Se van a analizar las ventajas y desventajas de realizar la planificación con el enfoque diario. Las ventajas principales son las siguientes:

- Respuesta Rápida: Al centrarse solo en un día, es posible adaptar rápidamente la planificación a cambios inesperados, como un aumento en el número de pacientes o la disponibilidad del personal.
- Simplicidad: Dado que se planea solo para un día, la complejidad de la planificación es mucho menor que en un enfoque semanal. Esto puede facilitar su implementación y seguimiento.
- Precisión: La planificación diaria permite una optimización más centrada y, posiblemente, más precisa para ese día específico.
- Recursos Definidos: La planificación diaria tiene en cuenta los recursos disponibles (personal, consultorios, etc.) para ese día en específico, lo cual puede resultar en una asignación de recursos más eficiente.

- Menor Tiempo Computacional: El número de variables y restricciones sería presumiblemente menor en una planificación diaria, resultando en un tiempo de cómputo más rápido para la optimización.

Por otro lado, las desventajas principales son las siguientes:

- Falta de Continuidad: Planificar día a día podría no tener en cuenta una visión más amplia o de largo plazo, lo que podría llevar a decisiones subóptimas en el contexto de una semana o más.
- Reactividad en lugar de Proactividad: Al enfocarse en un solo día, este enfoque podría ser más reactivo que proactivo, lo cual no es ideal para la gestión a largo plazo.
- Mayor Carga Administrativa: Cada día requeriría su propia planificación, lo cual podría ser una tarea administrativa considerable.
- Menor Flexibilidad: Al no tener una vista de múltiples días, podría haber menos flexibilidad para mover recursos o reprogramar actividades que no se realizaron según lo planeado en un día particular.
- Riesgo de Suboptimización: Podría haber una tendencia a optimizar para el corto plazo (es decir, el día individual) sin tener en cuenta las implicaciones a más largo plazo, como la satisfacción del paciente o el bienestar del personal.

4.4. Enfoque 2: Programación semanal

En este segundo enfoque, se ha decidido expandir un poco la idea inicial, buscando una planificación a nivel semanal, de lunes a domingo. Como se aumenta la cantidad de días también se amplia el número de estudios a planificar, siendo ahora 4 estudios distintos y que pueden ser sus actividades repartidas a lo largo de la semana como el algoritmo considere, siempre y cuando se sigan cumpliendo las restricciones determinadas por el usuario.

4.4.1. Elección y Configuración del Algoritmo

En cuanto a la elección y configuración del algoritmo se vuelve a utilizar el mismo algoritmo que para el primer enfoque, el NSGA-II.

Únicamente se realiza un cambio en uno de los parámetros:

- Tamaño de la población: se aumenta el tamaño de la población de 100 a 250 para la optimización. En problemas más complejos, un tamaño de población más grande es útil para poder lograr una buena aproximación al frente de Pareto.

4.4.2. Detalles Implementación Semanal

Para el enfoque de implementación semanal, se han realizado algunos cambios con respecto al diario. Al tener que planificar para 7 días en vez de uno y 4 estudios en vez de 2, hay que reducir el problema para que sea factible el proceso de optimización.

Para ello, se han eliminado las restricciones 3 y 4, las que tienen que ver con el personal. Esto facilita la optimización, ya que reduce el espacio de posibles variables bastante dejando fuera el personal, y permitiendo que la optimización en tiempos computacionales sea mucho más rápida. Con otros medios más potentes se podrían volver a añadir, pero para el estudio que se pretende hacer en este trabajo no es necesario. Por lo demás, el resto de restricciones son las mismas.

En cuanto a las funciones objetivo, varía la primera de ellas, ya que, al no tener una única hora de finalización de un día, sino que se tienen siete de ellas, se usa como función objetivo el minimizar lo máximo posible la media de las horas de finalización de los 7 días de la semana. Por otro lado la otra función objetivo sigue siendo minimizar el tiempo de espera de los pacientes.

4.4.3. Análisis de resultados

Se llevaron a cabo 200,000 iteraciones del algoritmo para calcular los resultados, proceso que tomó aproximadamente 6 horas. Es normal el aumento tanto en iteraciones como en duración, porque el tamaño del espacio de variables y de soluciones es mucho mayor, por lo que cada iteración dura más que las de la programación diaria y a su vez se tardan más iteraciones en llegar a soluciones aceptables.

Tras concluir las 200,000 iteraciones, se obtuvo un conjunto de soluciones subóptimas, con una única solución subóptima: [116.14, 419]. En cada par de valores, el primer número representa el valor de la primera función objetivo *f1*, mientras que el segundo número representa el valor de la segunda función objetivo *f2*.

Es importante destacar que, aunque NSGA-II generalmente produce un frente de Pareto como resultado, la calidad excepcional de esta solución en particular se justifica por su rendimiento superior en múltiples funciones objetivo simultáneamente. También es justificable debido a que no se ha logrado llegar a pesar de la duración de la ejecución del algoritmo a un punto donde no se cumplieran todas las restricciones, quedando en estos resultados un incumplimiento en el orden de actividades de uno de los estudios, y por lo tanto añadiendo una penalización de 100 a ambas funciones objetivo.

Sin esa penalización, la solución subóptima sería [16.14,319].

Esta solución indica que la primera función objetivo tiene un valor de 16.14, que se traduce en que la hora media de finalización de cada uno de los días son las 16. Esto significa que de media se podrían utilizar 6 horas cada día para poder atender a otras

urgencias o adelantar otras tareas distintas, como a su vez poder mandar personal a casa o en su defecto reducirlo para próximas semanas.

En cuanto a la segunda función objetivo, tiene un valor de 319, esto implica que hay 319 horas vacías en los consultorios a lo largo de toda la semana. Esto es un número bastante alto para lo que se podía esperar, pero es lógico teniendo en cuenta la cantidad de consultorios que hay disponibles, y los pocos estudios, siempre se va a tener algún consultorio vacío.

Pero cabe destacar que el principal problema en esta función objetivo es que hay un día al que el algoritmo ha decidido no ponerle ninguna actividad, esto es debido a que el programa ha decidido equilibrar el valor de la media de horas, dejando uno de los días vacíos y por lo tanto mejorando el valor de la primera función objetivo, pero sin embargo ha aumentado mucho el valor de la segunda función objetivo.

A continuación, se van a mostrar las planificaciones semanales que el algoritmo ha decidido para cada Estudio, y que al final son las que el usuario directamente podría implementar en su centro:

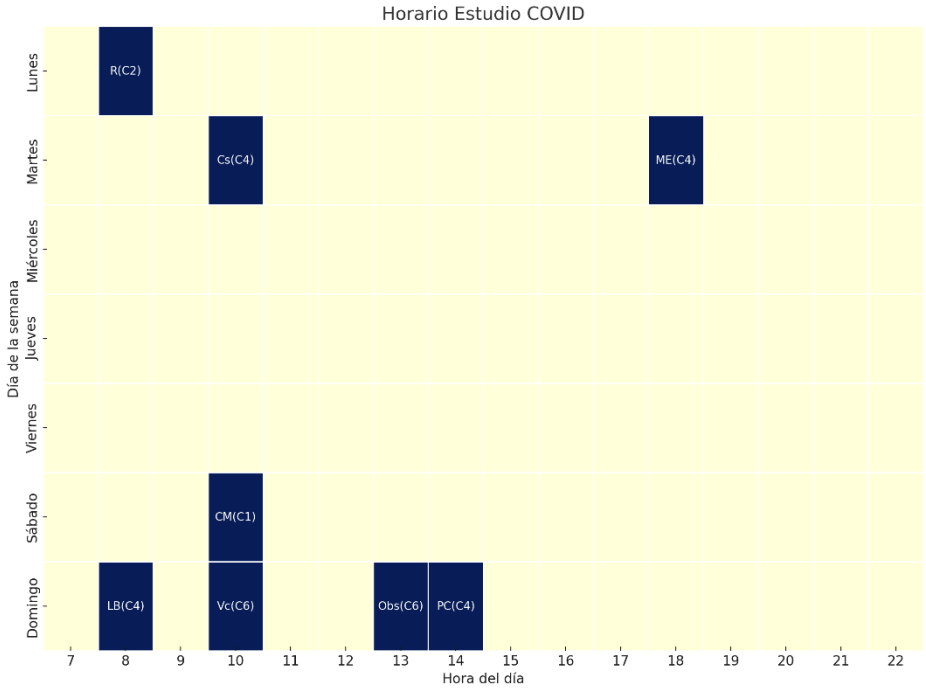

Figura 25: Planificación Semanal Estudio COVID

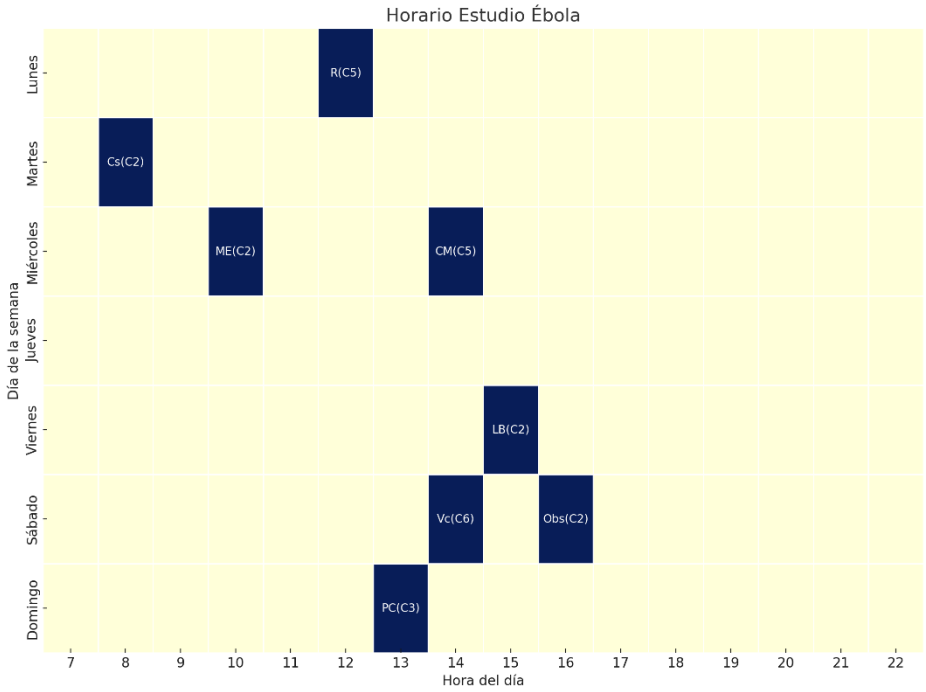

Figura 26: Planificación Semanal Estudio Ébola

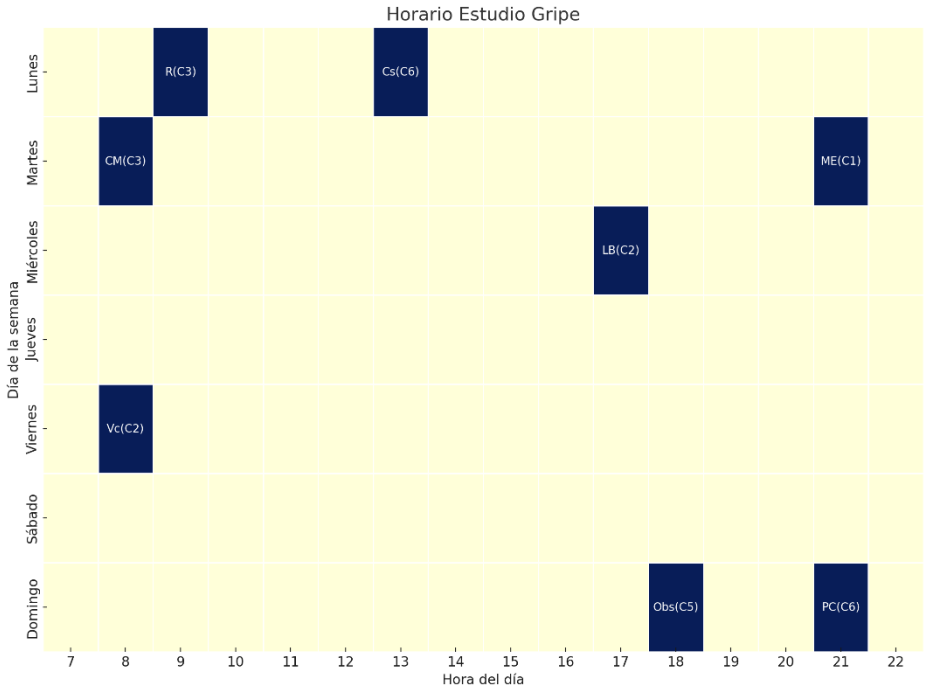

Figura 27: Planificación Semanal Estudio Gripe

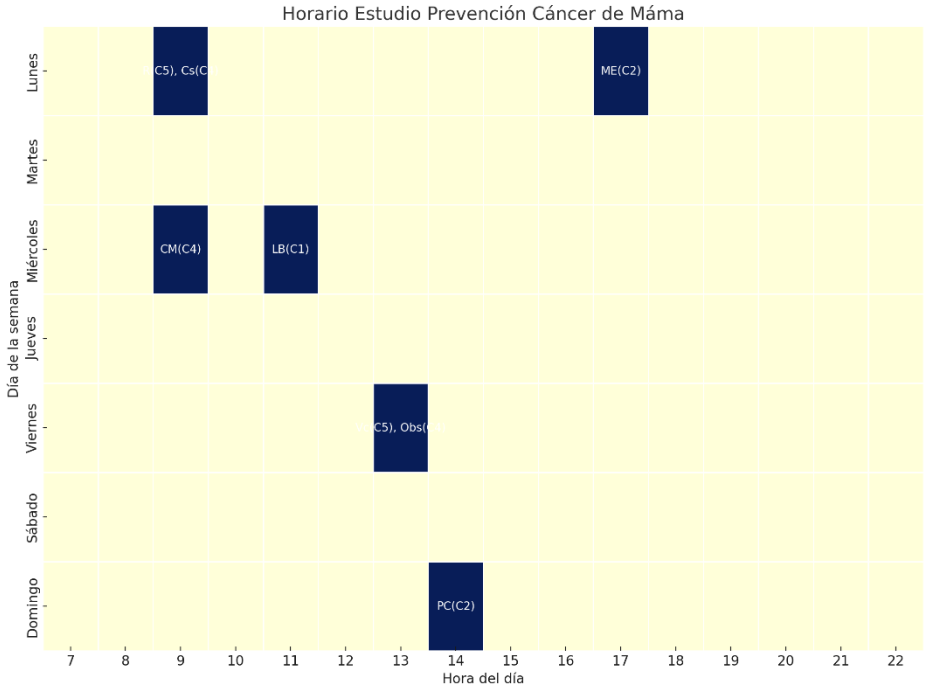

Figura 28: Planificación Semanal Estudio Prevención Cáncer de Mama

4.4.4. Ventajas y desventajas del enfoque semanal

Se van a analizar las ventajas y desventajas de realizar la planificación con el enfoque semanal. Las ventajas principales son las siguientes:

- Visión a Largo Plazo: La planificación semanal permite tener una visión más amplia y coherente, facilitando la toma de decisiones estratégicas.
- Eficiencia de Recursos: Podría permitir una asignación más eficiente de recursos al tener en cuenta una gama más amplia de variables y escenarios.
- Menor Carga Administrativa: Al planificar solo una vez a la semana, se podría reducir la carga de trabajo administrativo en comparación con la planificación diaria.
- Flexibilidad: La planificación semanal podría permitir más flexibilidad para adaptarse a cambios imprevistos, ya que se tiene una "visión de conjunto" y se pueden hacer ajustes durante la semana.
- Mejora del Bienestar del Personal: Al tener un horario más estable y predecible, el personal podría experimentar menos estrés y una mayor satisfacción en el trabajo.

Las desventajas más relevantes son:

- Menor Capacidad de Adaptación: Al tener un plan establecido para toda la semana, podría ser más difícil hacer ajustes rápidos en respuesta a cambios imprevistos.
- Complejidad: La planificación semanal podría ser más compleja debido al mayor número de variables y restricciones que deben tenerse en cuenta.
- Mayor Tiempo de Ejecución: Puede requerir más tiempo al inicio de cada semana para planificar debido a la mayor complejidad del problema.
- Riesgo de Inexactitud: Dado que se está proyectando más hacia el futuro, hay un mayor riesgo de que las predicciones (como la disponibilidad del personal o el número de pacientes) sean inexactas.
- Requiere Mayor Coordinación: Mantener un plan semanal coherente podría requerir una mayor coordinación entre diferentes departamentos o equipos, lo que podría ser un desafío en grandes organizaciones.

4.5. Conclusión del Desarrollo del Problema

Tras una evaluación exhaustiva de los enfoques de planificación diaria y semanal, es claro que cada uno presenta un conjunto único de ventajas y desventajas, lo que hace que su idoneidad varíe dependiendo de diversos factores como el tipo de proyecto, los recursos disponibles y el entorno operativo.

La planificación diaria destaca por su capacidad para manejar la incertidumbre y responder a situaciones cambiantes. Esto puede ser particularmente útil en entornos de alta volatilidad, donde los requerimientos y las circunstancias pueden cambiar rápidamente. Sin embargo, este enfoque puede llevar a una visión miope, centrada en solucionar problemas del día a día, en detrimento de los objetivos estratégicos y de largo plazo. Además, la constante necesidad de planificar puede convertirse en una carga administrativa que desvía tiempo y energía de la ejecución efectiva de las tareas.

Por otro lado, la planificación semanal proporciona un panorama más amplio que facilita la alineación de tareas individuales con objetivos más grandes. Este enfoque permite una mejor coordinación entre equipos y una asignación más eficiente de recursos. Sin embargo, la planificación semanal puede resultar en una estructura más rígida que puede ser difícil de adaptar cuando surgen imprevistos. Además, la falta de revisión diaria podría llevar a una menor responsabilidad y seguimiento, lo que puede ser perjudicial para la productividad y el cumplimiento de metas.

En el contexto de la optimización y planificación de tareas, el Algoritmo NSGA-II claramente emerge como una herramienta potentemente relevante. Su capacidad para encontrar un conjunto de soluciones óptimas en problemas con múltiples objetivos lo convierte en un candidato ideal para la adaptación tanto en la planificación diaria como en la semanal, demostrándose así en los resultados obtenidos. Con un equipo con mayor

capacidad computacional y optimizando recursos del algoritmo, se puede lograr llevar estos desarrollos a herramientas realmente útiles para el día a día.

Una opción que merece consideración es la adopción de un enfoque híbrido, que combine la adaptabilidad de la planificación diaria con la visión estratégica de la planificación semanal. Este sistema mixto podría iniciar con una planificación semanal para establecer un marco general de lo que se espera alcanzar. Luego, las sesiones de planificación diarias podrían usarse para ajustes tácticos que respondan a las circunstancias cambiantes. De esta manera, un enfoque híbrido puede ofrecer un equilibrio entre flexibilidad y estructura, maximizando las ventajas de ambos métodos mientras minimiza sus respectivas desventajas.

En conclusión, la elección entre la planificación diaria y la planificación semanal no es excluyente la una de la otra. En lugar de eso, cada organización o proyecto debe evaluar cuidadosamente sus propias necesidades, capacidades y contextos para desarrollar un sistema de planificación que aproveche las fortalezas de ambos enfoques. Se trata de un ejercicio de equilibrio, donde la clave está en adaptar el método de planificación al problema específico que se enfrenta, manteniendo siempre la flexibilidad para ajustar la estrategia conforme evolucionan las circunstancias.

5. CONCLUSIONES

El presente trabajo se inscribe en un ámbito de investigación de creciente importancia: la intersección entre tecnología y atención sanitaria. En un mundo donde la eficiencia operativa es vital, este trabajo no sólo se enfrenta a un desafío técnico, sino que también responde a necesidades prácticas en la administración de recursos y la atención al paciente en el entorno clínico.

La metodología empleada en este proyecto es una combinación de técnicas de optimización y algoritmos especializados, diseñados para abordar la complejidad del problema de programación de citas. No se trata sólo de un enfoque teórico, sino que se ha llevado a cabo una implementación práctica, cuyos resultados indican que es posible minimizar la media de las horas de finalización de las citas durante toda una semana, o también reducir el tiempo de espera de los pacientes del estudio. Este es un logro notable, que pone de manifiesto el potencial de aplicar técnicas avanzadas de ingeniería informática en desafíos del mundo real.

El trabajo, sin embargo, no está exento de desafíos y limitaciones. Aunque se logró una optimización significativa en términos de horarios, el algoritmo de planificación semanal no cumplió con todas las restricciones, lo que evidencia la complejidad inherente al problema. Por otro lado, el enfoque de planificación diaria sí que cumplió con estas restricciones, demostrando que cuanto menor es la complejidad del problema más efectivo puede resultar el algoritmo. Este aspecto crítico señala la necesidad de futuras investigaciones que puedan incorporar métodos más sofisticados, o usar máquinas con mayor capacidad computacional, para superar estas limitaciones.

Este trabajo abre múltiples líneas para futuras investigaciones y aplicaciones prácticas. En el corto plazo, la exploración de algoritmos más avanzados, como algoritmos genéticos y técnicas de aprendizaje profundo, podría ofrecer mejoras significativas en la eficacia del sistema de programación de citas. Además, la implementación del algoritmo en entornos clínicos reales a través de pruebas piloto sería un paso crucial para validar y ajustar el modelo según las condiciones del mundo real. Desde una perspectiva más amplia, la interdisciplinariedad podría enriquecer este proyecto, integrando conocimientos de medicina, gestión de la atención médica y psicología para crear un sistema más holístico. Asimismo, aspectos como la ética y la privacidad en el manejo de datos de pacientes se vuelven cada vez más relevantes a medida que se contempla una implementación más amplia. Finalmente, la escalabilidad y la adaptabilidad del algoritmo en diferentes entornos clínicos, así como su integración con sistemas de gestión hospitalaria existentes, representan direcciones futuras esenciales que podrían llevar este trabajo desde el ámbito académico hasta un impacto real en la atención médica.

Finalmente, este Trabajo de Fin de Grado representa una contribución significativa al campo de la Ingeniería Informática, especialmente en la aplicación de algoritmos y técnicas de optimización a problemas prácticos en el ámbito de la salud. No solo ofrece un nuevo enfoque metodológico, sino que también plantea preguntas y desafíos que pueden ser el punto de partida para futuras investigaciones.

BIBLIOGRAFÍA

[1] A. Kuiper, B. Kemper y M. Mandjes, «A computational approach to optimized appointment scheduling,» *Queueing Systems,* vol. 79, pp. 5-36.

[2] A. Kuiper, M. Mandjes y J. De Mast, «Optimal stationary appointment schedules,» *Operations Research Letters,* vol. 45, nº 6, pp. 549-555, 2017.

[3] U. College, «Guía paso a paso para la programación de citas médicas,» [En línea]. Available: https://www.unitekcollege.edu/es/blog/a-step-by-step-guide-to-medical-appointment-scheduling/. [Último acceso: 29 08 2023].

[4] C. M. Rudloff, Agendamiento de citas médicas mediante asignación óptima de capacidad para distintos tipos de pacientes, Santiago de Chile: PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE, 2015.

[5] J. M. R. Barrios, D. Serrano, T. Monleón y J. Car, «Los modelos de simulación de eventos discretos en la evaluación económica de tecnologías y productos sanitarios,» *Gac Sanit.*, vol. 2, nº 22, pp. 151-161, 2008.

[6] M. A. B. Odio, Gestión óptima de citas médicas mediante la aplicación de un modelo de optimización multicriterio, Zaragoza: Universidad de Zaragoza, 2019.

[7] G. Mitsi, T. Grinnell, S. Giordano, T. Goodin, S. Sanjar, E. Marble y A. Pikalov, «Implementing Digital Technologies in Clinical Trials: Lessons Learned.,» *Innovations in clinical neuroscience,* 2022.

[8] M. Reza Mazaheri Habibi, F. Mohammadabadi, H. Tabesh, H. Vakili-Arki, A. Abu-Hanna y S. Eslami, «Effect of an Online Appointment Scheduling System on Evaluation Metrics of Outpatient Scheduling System: a before-after MulticenterStudy,» *Journal of Medical Systems,* 2019.

[9] J. Cubillas, M. Ramos, F. Feito y T. Ureña, «An improvement in the appointment scheduling in Primary Health Care Centers using data mining.,» *J Med Syst,* vol. 38, pp. 89-99, 2014.

[10] J. Tang, C. Yan y P. Cao, «Appointment scheduling algorithm considering routine and urgent patients,» *Elsevier Ltd,* 2014.

[11] Y. Gocgun y M. L. Puterman, «Dynamic scheduling with due dates and time windows: an application to chemotherapy patient appointment booking,» *Health Care Manag Sci,* nº 17, pp. 60-76, 2013.

[12] P. A. Salzarulo, S. Mahar y S. Modi, «Beyond Patient Classification: Using Individual Patient Characteristics in Appointment Scheduling,» *Production and Operations Manaement Society,* vol. 6, nº 25, pp. 1056-1072, 2016.

[13] X. Qu, Y. Peng, N. Kong y J. Shi, «A two-phased approach to schedulin multi-category outpatient appointments - A case study of a women´s clinic,» *Health Care ManagSci,* nº 16, pp. 197-216, 2013.

[14] E. Canizo y P. Lucero, «Software Para Programación Lineal -LINGO/LINDO-,» 2002.

[15] B. E., D. M. y T. G., Swarm intelligence: from natural to artificial systems., Oxford University Press, Inc, 1999.

[16] S. Katoch, S. S. Chauhan y V. Kumar, «A review on genetic algorithm: past, present, and future,» *Multimedia Tools and Applications,* nº 80, pp. 8091-8126, 2021.

[17] M. Z, Genetic algorithms + data structures = evolution programs, New York: Springer-Verlag, 1992.

[18] K. J y E. RC, «Particle swarm optimization,» de *Proceedings of IEEE International conference on neural networks*, 1995, pp. 1942-1948.

[19] D. M, B. M y S. T, «Ant colony optimization - artificial ants as a computational intelligence technique,» *IEEE Comput Intell Mag,* nº 1, pp. 28-39, 2006.

[20] H. JH, Adaptation in natural and artificial systems, The University of Michigan Press, 1975.

[21] N. Visual, «Nucleo Visual - ¿Qué es un algoritmo de optimización evolutiva?,» [En línea]. Available: https://nucleovisual.com/que-es-un-algoritmo-de-optimizacion-evolutiva/?expand_article=1. [Último acceso: 2023 08 29].

[22] S. SN y D. SN, Introduction to genetic algorithm, Berlin Heidelberg: Springer-Verlag, 2008.

[23] F. B y M. M, «Genetic operators for sequencing problems,» *Foundations of Genetic Algorithms,* pp. 283-300, 1991.

[24] D. De Jong y A. Brindle, «Learning with genetic algorithms: An overview,» *Machine Learning,* nº 3, pp. 121-138, 1988.

[25] J. K, «Selection methods for genetic algorithms,» *International Journal of Emerging Sciences,* vol. 4, nº 3, pp. 333-344, 2013.

[26] C.-Y. Lee, «Entropy-Boltzmann selection in the genetic algorithms,» *IEEE Transactions on Systems,* vol. 33, nº 1, pp. 138-149, 2003.

[27] S. GK, G. TT, O. CK, A. R y A. P, «A comparison on the performance of crossover techniques in video game,» *IEEE international conference on control system,* pp. 493-498, 2013.

[28] Avik_Dutta, «geeksforgeeks - Crossover in Genetic Algorithm,» [En línea]. Available: https://www.geeksforgeeks.org/crossover-in-genetic-algorithm/. [Último acceso: 29 08 2023].

[29] G. D y L. R, «Alleles, loci and the traveling salesman problem,» *Proceedings of the 1st international conference on genetic algorithms and their applications,* vol. 1985, pp. 154-159, 1985.

[30] A. L. E. Idrissi, C. Tajani y M. Sabbane, «New crossover operator for genetic algorithm to resolve the fixed charge transportation problem,» 2017.

[31] P. Larranaga, C. Kuijpers, R. Murga y I. Inza, «Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators,» *Artificial Intelligence Review,* vol. 2, nº 13, pp. 129-170, 1999.

[32] J. DF, M. SK y T. M, «Multiobjective meta-heuristics: an overview of the current state-of-the-art,» *Eur J Oper Res,* vol. 1, nº 137, pp. 1-9, 2002.

[33] S. JD, «Multiple objective optimization with vector evaluated genetic algorithms,» de *Proceedings of the international conference on genetic algorithm and their applications,* 1985.

[34] D. K. Srinivas N, «Multi-objective optimization using nondominated sorting in genetic algorithms,» *Evol Comput* , vol. 2, nº 3, pp. 221-248, 1994.

[35] H. Ma, Y. Zhang, S. Sun, T. Liu y Y. Shan, «A comprehensive survey on NSGA-II for multi-objective optimization and applications,» *Artificial Intelligence Review,* 2023.

[36] D. K, P. A, A. S y M. T, «A fast and elitist multi-objective genetic algorithm: NSGA-II,» *IEEE Trans Evol Comput,* vol. 2, nº 6, pp. 182-197, 2002.

[37] M. Shahriari, «Multi-objective optimization of discrete time-cost tradeoff problem in project networks using non-dominated sorting genetic algorithm.,» *Journal of Industrial Engineering International,* vol. 2, nº 12, pp. 159-169, 2016.

[38] N. Y. Á. Herrera, Algoritmos de inteligencia artificial y MiniMax para la optimización en operación de embalses Multipropósito, Caso de estudio embalse EL Cercado - la Guajira, Universidad Escuela Colombiana de Ingeniería Julio Garavito , 2023.

[39] J. D. V. Sánchez y J. M. V. Rodríguez, APLICACIÓN DEL ALGORITMO GENÉTICO NO DOMINADO NSGA-II PARA LA ACELERACIÓN DE PROYECTOS DE CONSTRUCCIÓN A TRAVÉS DE LA SOLUCIÓN DEL PROBLEMA DE COMPENSACIÓN DISCRETA TIEMPOCOSTO (DTCTP) EN UNA OBRA CIVIL DEL VALLE DEL CAUCA, Universidad del Valle, 2019.

[40] B. Julian y D. Kalyanmoy, «Pymoo: Multi-Objective Optimization in Python,» *IEEE Access*, 2020.

[41] J. Blank, «Pymoo,» [En línea]. Available: https://pymoo.org/. [Último acceso: 22 08 2023].

[42] J. Hunter, «Matplotlib: A 2D graphics environment.,» *Computer Science Engineer,* vol. 9, nº 3, pp. 90-95, 2007.

[43] P. Hoffman, G. Grinstein y D. Pinkney, «Dimensional anchors: A graphic primitive for multidimensional multivariate information visualizations,» de *Proc. Workshop New Paradigms Inf. Vis. Manipulation Conjunct*, New York, 1999.

[44] K. Deb, «Multi-Objective Optimization Using Evolutionary Algorithms,» de *Wiley*, New York, 2001.

[45] L. Rachmawati y D. Srinivasan, «Multiobjective evolutionary algorithm with controllable focus on the knees of the Pareto front,» *IEEE Trans. Evol. Comput.,* vol. 13, nº 4, pp. 810-824, 2009.

[46] «CEVAXIN,» [En línea]. Available: https://cevaxin.com/aboutus.

[47] P. Foundation, «Python,» [En línea]. Available: https://www.python.org/. [Último acceso: 22 08 2023].

[48] JetBrains, «PyCharm - Python IDE for Professional Developers,» [En línea]. Available: https://www.jetbrains.com/pycharm/. [Último acceso: 22 08 2023].

[49] N. Developers, «NumPy,» [En línea]. Available: https://numpy.org/. [Último acceso: 22 08 2023].

[50] p. D. Team, «pandas - Python Data Analysis Library,» [En línea]. Available: https://pandas.pydata.org/. [Último acceso: 22 08 2023].

[51] M. Team, «Matplotlib: Visualization with Python,» [En línea]. Available: https://matplotlib.org/. [Último acceso: 22 08 2023].

ANEXO A. REPOSITORIO DE CÓDIGO EN GITHUB

https://github.com/danielroasantin/TFG_Optimizacion_ProgramacionDeCitas