

UNIVERSIDAD POLITÉCNICA DE MADRID

E.T.S. DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

PROYECTO FIN DE GRADO

GRADO EN INGENIERÍA DEL SOFTWARE

Planificación nutricional mediante algoritmos evolutivos

Autor: Javier Quesada Pajares

Director: Cristian Ramírez Atencia

Madrid, 9 de noviembre de 2024



Planificación nutricional mediante algoritmos evolutivos

Proyecto Fin de Grado, 9 de noviembre de 2024

Autor: Javier Quesada Pajares

Director: Cristian Ramírez Atencia

E.T.S. de Ingeniería de Sistemas Informáticos

Campus Sur UPM, Carretera de Valencia (A-3), km. 7

28031, Madrid, España

Esta obra está bajo una licencia [Creative Commons «Atribución-NoComercial-CompartirIgual 4.0 Internacional»](https://creativecommons.org/licenses/by-nc-sa/4.0/).



Agradecimientos

A mi familia y amigos por apoyarme todo este tiempo.

A mis compañeros y tutores por esta etapa.

Resumen

Este proyecto explora el uso e implementación de algoritmos genéticos en la planificación nutricional. La utilización de técnicas de inteligencia artificial, combinadas con datos nutricionales, permite diseñar dietas adaptadas específicamente a las necesidades y preferencias de cada usuario.

Por medio de técnicas de computación evolutiva, se diseña un menú alimenticio que cumple con distintos objetivos y restricciones nutricionales, como la ingesta calórica diaria o la distribución de macronutrientes. El algoritmo permite la personalización de la lista de alimentos para que se ajuste a las preferencias y a las alergias de los usuarios.

Se ha desarrollado un menú semanal que consta de 77 platos diferentes, distribuidos en 5 comidas diarias. Los alimentos son elegidos de una base de datos compuesta por más de 2500 registros, creada a partir de la información proporcionada por el gobierno del Reino Unido. Para ajustarse a las necesidades del problema, se han modificado diversos parámetros del algoritmo, como la inicialización o la mutación.

Para optimizar el rendimiento y encontrar las mejores soluciones posibles, se evaluaron diferentes estrategias de cruce y mutación. También se realizaron pruebas comparativas entre distintas técnicas de manejo de restricciones y entre distintos algoritmos de optimización multiobjetivo, como NSGA-II, SPEA2 o MOEA/D.

Los resultados demuestran que el enfoque propuesto cumple con las restricciones y los objetivos en la gran mayoría de los sujetos de prueba analizados, mejorando la eficacia y la personalización de la planificación nutricional.

Palabras clave: Algoritmos evolutivos; Planificación nutricional; Inteligencia artificial; Optimización multiobjetivo

Abstract

This project explores the use and implementation of genetic algorithms in nutritional planning. The utilization of artificial intelligence techniques, combined with nutritional data, allows for designing diets specifically adapted to each user's needs and preferences.

Through evolutionary computing techniques, an alimentary menu is designed that meets various nutritional objectives and restrictions, such as daily caloric intake or macronutrient distribution. The algorithm enables the personalization of the food list to adjust to users' preferences and allergies.

A weekly menu has been developed consisting of 77 different dishes, distributed across 5 daily meals. The foods are selected from a database comprising over 2,500 entries, created from information provided by the UK government. To suit the problem's needs, various algorithm parameters, such as initialization and mutation, have been modified.

To optimize performance and find the best possible solutions, different crossover and mutation strategies were evaluated. Comparative tests were also conducted between various constraint-handling techniques and different multi-objective optimization algorithms, such as NSGA-II, SPEA2, and MOEA/D.

The results demonstrate that the proposed approach meets the restrictions and objectives in the vast majority of the analyzed test subjects, improving the efficiency and personalization of nutritional planning.

Keywords: Evolutionary algorithms; Nutritional planning; Artificial intelligence; Multi-objective optimization

Índice general

1	Introducción	1
1.1	Contexto	1
1.2	Motivación	2
1.3	Objetivos	4
1.4	Estructura de la memoria	4
2	Estado del arte	6
2.1	Inteligencia artificial	6
2.2	Algoritmos evolutivos	8
2.3	Planificación nutricional mediante algoritmos evolutivos	10
3	Marco teórico	12
3.1	Población	13
3.2	Evaluación	14
3.3	Operadores	17
3.4	Tipos de algoritmo	20
4	Desarrollo	26
4.1	Base de datos	26
4.2	Conceptos nutricionales	30
4.3	Cromosoma planteado	32

4.4	Objetivos y restricciones	33
4.5	Construcción del algoritmo	36
4.6	Interfaz gráfica	43
5	Experimentación	48
5.1	Manejo de restricciones	50
5.2	Algoritmos multi-objetivo	53
6	Conclusiones y líneas de investigación	55
6.1	Conclusiones	55
6.2	Líneas de investigación	56
7	Impacto social y medioambiental	58
A	Apéndice: Grupos de comida	59
B	Apéndice: Entorno y sujetos	62
C	Apéndice: Tablas de resultados	64
C.1	Manejo de restricciones	64
C.2	Algoritmos multi-objetivo	70
	Bibliografía	75

Índice de figuras

1.1	Prevalencia (%) de obesidad y exceso de peso por sexo y edad. Fuente [9]	3
3.1	Algoritmo genético simple.	12
3.2	Estructura de un cromosoma.	13
3.3	Operadores.	17
3.4	Cruce de un punto.	19
3.5	Mutación uniforme.	20
3.6	Estructura general del algoritmo NSGA-II. Fuente [41]	22
3.7	Cálculo del fitness en SPEA2. Fuente [43]	23
3.8	Vecindarios en MOEA/D. Fuente [45]	25
4.1	Importación de los datos usando MySQL Workbench.	28
4.2	Resultado final de la base de datos.	29
4.3	Estructura de la solución.	32
4.4	Matriz de inicialización.	39
4.5	Mutación personalizada.	40
4.6	Ventana principal.	44
4.7	Ventana de la base de datos.	44
4.8	Ventana de las preguntas al usuario.	46
4.9	Ventana del menú generado.	47

Índice de tablas

4.1	Resultados del algoritmo genético.	40
5.1	Resultados en Penalización estática: probabilidad de mutación.	51
5.2	Comparación de métodos de manejo de restricciones.	52
5.3	Comparación de algoritmos.	54
B.1	Seeds utilizados.	62
B.2	Características y preferencias alimenticias de los sujetos	63
C.1	Resultados en Penalización estática: probabilidad de mutación.	64
C.2	Resultados en Penalización estática: tipo y probabilidad de cruce.	65
C.3	Resultados en Método separatista: probabilidad de mutación.	66
C.4	Resultados en Método separatista: tipo y probabilidad de cruce.	67
C.5	Resultados en Restricción como objetivo: probabilidad de mutación.	68
C.6	Resultados en Restricción como objetivo: tipo y probabilidad de cruce.	69
C.7	Resultados en SPEA2: manejo de restricciones.	70
C.8	Resultados en MOEA/D: número de vecinos.	71
C.9	Resultados en MOEA/D: probabilidad de selección de vecinos.	72
C.10	Resultados en MOEA/D: direcciones de referencia en Das-Dennis.	73
C.11	Resultados en MOEA/D: direcciones de referencia en Incremental.	74

Índice de listados

2.1	Algoritmo evolutivo.	9
3.1	Direcciones de referencia.	24
4.1	Creación de la tabla y sus campos.	28
4.2	Clase para la evaluación.	37
4.3	Factores de penalización.	42

1.

Introducción

1.1. Contexto

En los últimos años, la inteligencia artificial (IA) se ha posicionado como una de las herramientas más útiles e interesantes de las que se disponen. El término «*inteligencia artificial*» fue acuñado por primera vez por John McCarthy en la Conferencia de Darmouth de 1956 [1], años después de que Alan Turing formulase la pregunta sobre si las máquinas podían pensar y plantease el famoso “*Test de Turing*” [2]. Varios científicos se reunieron con el objetivo de discutir acerca de la posibilidad de que un artefacto se comportase de manera inteligente. Se llegó a la conclusión de que es posible describir cada aspecto del aprendizaje con tal precisión que se puede crear una máquina que lo imite.

La IA se enfoca en crear sistemas que puedan realizar tareas que normalmente requerirían de inteligencia humana. Aunque ha tenido un reciente auge debido a los chatbots o los asistentes personales, presenta una gran cantidad de finalidades. Dentro de los usos que se le dan a la IA destacan la creación y el análisis de productos o la automatización de servicios, además de la optimización de procesos. Esta optimización se enfoca en encontrar la mejor solución posible a un problema dado dentro de un conjunto de opciones factibles. Los algoritmos de optimización y de personalización permiten, por ejemplo, crear aplicaciones que permiten organizar tareas de manera inteligente [3] o termostatos que ajustan la calefacción automáticamente [4].

En el ámbito de la nutrición, la IA tiene un gran potencial para la creación de dietas. La capacidad de analizar grandes volúmenes de datos y entender las necesidades nutricionales de cada paciente permite diseñar planes alimenticios personalizados. No solo cumple con los requisitos calóricos o de nutrientes, sino que también se adapta a los gustos del usuario para tomar en cuenta las preferencias personales o las restricciones dietéticas.

Este Proyecto Fin de Grado (PFG) se centra en la resolución de un problema de optimización, la creación de un menú semanal de comidas personalizado que cumpla distintos objetivos, como el número de calorías diarias o la cantidad de macronutrientes ingeridos. Se hace uso de los algoritmos evolutivos, que diseñan una dieta equilibrada a partir de la selección, el cruce y la mutación de los distintos alimentos.

Los algoritmos evolutivos son muy útiles para estos problemas de optimización porque permiten explorar una gran cantidad de combinaciones posibles de alimentos para encontrar una solución óptima. El proceso incluye una población inicial de menús, la evaluación de estos según los requisitos nutricionales y la selección de los mejores para la creación de nuevas generaciones mediante cruce y mutación. Este ciclo se repetirá hasta que se obtenga un plan alimenticio optimizado que cumpla con todos los objetivos nutricionales y de preferencia.

1.2. Motivación

La motivación para realizar este proyecto viene dada por el interés creciente en el área de la inteligencia artificial y cómo ha cambiado la manera de plantear los problemas respecto al pasado. En España, el sector TIC (Tecnologías de la Información y la Comunicación) cuenta con más del 40% de empresas que usan estas herramientas para la automatización de flujos de trabajo, el análisis de datos o la gestión de la cadena de suministro [5]. Buscan mejorar la precisión y la eficiencia a la hora de diseñar soluciones.

No obstante, no solo en sectores tecnológicos se hace uso de la IA. El médico o el alimentario también están incorporándola de manera gradual. Los diagnósticos de imágenes médicas [6] o la agricultura de precisión [7] son cada vez más comunes. También en la nutrición, relacionada con estos ámbitos, estas tecnologías permiten nuevas posibilidades.

Recientes estudios han demostrado que la IA puede aumentar la adherencia a planes de alimentación saludables un 20%, al ofrecer recomendaciones personalizadas y un seguimiento continuo. Además, estas herramientas permiten ajustar dinámicamente las recomendaciones en función de la evolución de la salud del usuario, mejorando así los planes nutricionales [8].

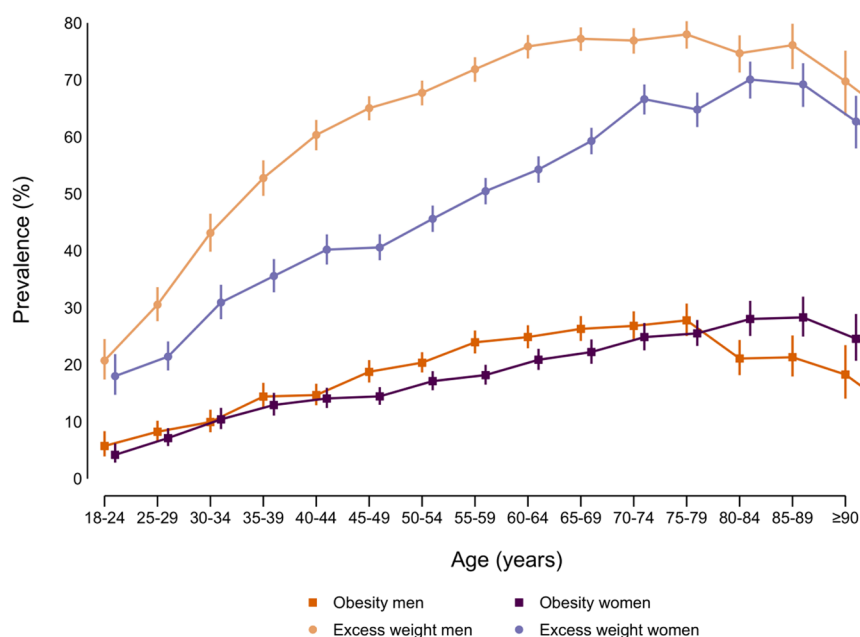


Figura 1.1. Prevalencia (%) de obesidad y exceso de peso por sexo y edad. Fuente [9]

Según un estudio llevado a cabo por la *Agencia Española de Seguridad Alimentaria y Nutrición (AESAN)* y por el *Instituto de Salud Carlos III (ISCIII)*, un 55,8% de la población adulta española tiene exceso de peso y un 18,7% padece obesidad, tal y como se muestra en la figura 1.1. Esto trae consigo múltiples problemas de salud, como la aparición de enfermedades crónicas o cardiovasculares, complicaciones respiratorias o dificultades al moverse.

La inteligencia artificial puede transformar la forma de entender la nutrición al utilizar datos para crear planes alimentarios personalizados. Esto permite comprender mejor los hábitos alimenticios y sus efectos en la salud. Con estas tecnologías, no solo se puede gestionar y prevenir mejor el sobrepeso y la obesidad, sino también fomentar prácticas alimenticias más saludables.

La planificación nutricional mediante algoritmos evolutivos ofrece una forma innovadora de mejorar los hábitos alimenticios y la nutrición. Por lo tanto, estas técnicas, además de promover una vida sana y bienestar para todos, se alinean con el tercero de los Objetivos de Desarrollo Sostenible (ODS) [10], por lo que es un gran aliciente a la hora de realizar este PFG.

1.3. Objetivos

El objetivo general de este proyecto es la creación de una planificación semanal de comidas mediante algoritmos evolutivos. Experimentando con diferentes algoritmos se busca crear un menú que cumpla con distintos objetivos nutricionales, como la ingesta calórica diaria, a la vez que limitarse según las restricciones dadas. Los objetivos específicos de este PFG son:

- Desarrollar un algoritmo evolutivo capaz de generar menús que cumplan con las restricciones y los objetivos nutricionales establecidos.
- Personalizar el algoritmo para la variación de las comidas según las necesidades específicas de los individuos.
- Experimentar con distintas configuraciones y variantes del algoritmo evolutivo en busca de encontrar la mejor solución posible.
- Evaluar la sensibilidad y eficacia del algoritmo.
- Ejecutar pruebas que validen el algoritmo.
- Documentar los resultados obtenidos.

1.4. Estructura de la memoria

En este subapartado se explicará la estructura del documento.

En el capítulo 2, [Estado del arte](#), se da un enfoque general al proyecto. Se da contexto sobre las áreas en las que se basa el proyecto: la inteligencia artificial y los algoritmos evolutivos, además de sobre el tema central del PFG, la planificación nutricional mediante algoritmos evolutivos.

En el capítulo 3, [Marco teórico](#), se aporta la base teórica en la que se desarrolla el proyecto. Se indaga en el algoritmo evolutivo y en sus diversas partes, y se explica su funcionamiento.

En el capítulo 4, [Desarrollo](#), se describen los pasos a seguir para la construcción del algoritmo evolutivo y del planificador de menús, además de la explicación de la base de datos y los conceptos nutricionales necesarios para ello.

En el capítulo 5, [Experimentación](#), se realizan análisis de sensibilidad, cálculos de hipervolumenes y comparaciones entre los diferentes algoritmos genéticos utilizados, además de documentar los resultados. Estas pruebas permiten evaluar y comparar el rendimiento y la eficiencia de los algoritmos.

En el capítulo 6, [Conclusiones y líneas de investigación](#), se resumen las tareas realizadas en el proyecto y se evalúa si se han cumplido los objetivos propuestos. También se comentan posibles pasos a seguir tras la elaboración de este trabajo, como las posibles mejoras al algoritmo y las diversas pruebas que se pueden llevar a cabo.

En el capítulo 7, [Impacto social y medioambiental](#), se explica la importancia y la valía que el trabajo puede aportar para promover una nutrición saludable y el bienestar global.

2.

Estado del arte

2.1. Inteligencia artificial

La inteligencia artificial se puede definir como el estudio y diseño de agentes inteligentes, es decir, de sistemas que perciben su entorno y toman decisiones o acciones que maximizan sus posibilidades de éxito. Este campo se basa en disciplinas como la informática, la lógica o la neurociencia, que contribuyen a la simulación de capacidades cognitivas humanas en máquinas.

Los agentes inteligentes, que son la base de este campo, se clasifican según su capacidad de reconocer y actuar en el entorno. Se pueden encontrar desde agentes simples, que responden directamente a estímulos, hasta agentes basados en utilidad, que evalúan si los resultados de sus acciones son satisfactorios. En estos agentes avanzados, la racionalidad es esencial. Es la habilidad de realizar elecciones óptimas que maximicen la posibilidad de alcanzar objetivos. Utilizando distintas herramientas de lógica, los agentes inteligentes pueden formular y modificar conocimientos, deduciendo nueva información.

El razonamiento lógico también juega un papel fundamental en muchos métodos de inteligencia artificial. Un ejemplo de esto son los algoritmos evolutivos, que se inspiran en los procesos biológicos de evolución para resolver problemas complejos. Estos algoritmos utilizan principios como la selección natural, el cruce y la mutación para mejorar progresivamente una población de posibles soluciones hacia un objetivo deseado. Al aplicar estos procesos evolutivos, pueden desarrollar soluciones eficientes a problemas que serían difíciles de resolver mediante métodos tradicionales.

El campo de la IA en la actualidad se encuentra en un estado de rápida evolución. Su capacidad para analizar y procesar grandes cantidades de datos o para mejorar la eficiencia en distintas industrias han hecho que se trate de una tecnología en auge. El estudio de *McKinsey & Company* (2022) [11] estima que el 50 % de las empresas ya usan IA en sus tareas diarias, donde destacan la optimización de servicios, la creación de productos y el análisis del servicio al cliente.

El informe "*AI Index Report 2024*" (2024) [12] muestra que la IA ya supera a las habilidades humanas en ciertas tareas, como en la clasificación de imágenes, con una precisión del 97 % respecto al 95 % de los humanos, o en juegos como el ajedrez, donde supera consistentemente a los jugadores humanos.

Si bien existen diversas formas de clasificar la IA, una de las más interesantes respecto a este PFG es la basada en la representación y el procesamiento del conocimiento. Según esta clasificación, existen dos tipos: la simbólica y la subsimbólica. La IA simbólica utiliza representaciones explícitas y legibles para los humanos, como símbolos y reglas lógicas, para resolver problemas. Consiste en incorporar conocimientos humanos y reglas de comportamiento a programas informáticos. Por otro lado, la subsimbólica se centra en métodos que no dependen de representaciones explícitas y se inspiran en procesos naturales.

En las últimas décadas, la IA subsimbólica ha sido la más desarrollada, destacándose en tres campos principalmente.

Primero, el aprendizaje automático, particularmente a través de redes neuronales, que permite aprender y procesar grandes volúmenes de datos. Un ejemplo de esto es el proyecto *AlphaFold* de *DeepMind*, que utiliza IA para predecir las estructuras de proteínas, acelerando la investigación científica para la creación de nuevos medicamentos [14].

El segundo es la lógica borrosa, que simula la forma en la que los humanos toman decisiones en situaciones de incertidumbre. Un ejemplo interesante de esto es su uso en la industria automotriz para mejorar la estabilidad del vehículo mediante la regulación adaptativa de los amortiguadores en respuesta a las condiciones del camino y las dinámicas del vehículo [15].

Por último, la computación evolutiva, que emplea procesos evolutivos biológicos para la resolución de problemas. Dentro de esta rama destacan principalmente dos tipos de algoritmos.

Uno de ellos son los algoritmos de enjambre. Inspirados en el comportamiento colectivo de sistemas naturales como colonias de hormigas o bandadas de aves, son métodos de optimización que solucionan problemas complejos mediante la colaboración de múltiples agentes simples. Un uso actual destacado de estos algoritmos es en la optimización de granjas eólicas, donde se emplean para mejorar la ubicación de turbinas y maximizar la eficiencia energética [16].

La otra categoría son los algoritmos evolutivos, foco principal de este proyecto. Basados en la teoría de la evolución, aplican procesos como selección y mutación para mejorar iterativamente soluciones hasta alcanzar un objetivo.

2.2. Algoritmos evolutivos

Un algoritmo se puede definir como un procedimiento computacional bien definido que toma un valor, o un conjunto de valores, como entrada y produce un valor, o un conjunto de valores, como salida. Se trata de un conjunto de instrucciones que permiten realizar una actividad mediante sucesivos pasos.

Un algoritmo evolutivo es un tipo de algoritmo que proviene de la computación evolutiva. Esta rama de la IA emplea principios inspirados en la evolución biológica para la resolución de problemas. Teoría propuesta por Charles Darwin en 1859 [17], expone que en la naturaleza, en un entorno dado que puede albergar un número limitado de individuos, la selección natural favorecen a aquellos que poseyeran características que les permitiesen adaptarse mejor al medio, teniendo una mayor posibilidad de sobrevivir y reproducirse. A lo largo del tiempo, las características ventajosas se propagan a través de las generaciones, mientras que aquellas menos favorables tienden a desaparecer. Por lo tanto, estas poblaciones irán evolucionando y adaptándose gradualmente al entorno.

Los algoritmos evolutivos han tenido varios proyectos que han marcado el desarrollo de esta tecnología. Los primeros acercamientos tuvieron lugar en las décadas de 1960 y 1970. Por un lado, Lawrence J. Fogel (1966) [18] comenzó a explorar la programación evolutiva, centrándose en la evolución de autómatas finitos. Fue uno de los primeros intentos de aplicar principios evolutivos a la informática. Por otro lado, Rechenberg (1973) [19] desarrolló estrategias de evolución para problemas de optimización. Centradas en la selección y en la mutación, resultaron ser muy útiles para aplicarlas en la realidad.

Tras estos primeros pasos, John Holland fue fundamental para el desarrollo de los algoritmos genéticos (AGs), subrama que se inspira en la genética para la optimización de problemas. Su libro *"Adaptation in Natural and Artificial Systems"* (1975) [20] se centró en la idea de que la evolución biológica podía ser simulada y utilizada para resolver problemas complejos en computación. Como se observa en el listado 2.1, estos algoritmos hacen uso la selección, el cruce o la mutación para evolucionar una población candidata de individuos hacia una mejor solución. Cada solución es evaluada según una función de fitness, que mide qué tan buena es la solución al problema en cuestión.

Listado 2.1. Algoritmo evolutivo.

```
INICIAR
  INICIALIZAR población
  EVALUAR fitness
  REPETIR HASTA CUMPLIR condición de parada:
    SELECCIONAR individuos
    CRUZAR padres
    MUTAR hijos
    EVALUAR individuos
    FORMAR nueva generación
  RETORNAR solución
FINALIZAR
```

Para que un AG se vaya reconduciendo hacia soluciones más favorables, se hace uso de las restricciones. Se usan diversas técnicas, como penalizar las soluciones que violan las restricciones, emplear métodos de reparación para modificar las soluciones no válidas o descartar directamente las soluciones que no sean factibles. Estas estrategias permiten que los algoritmos genéticos mantengan la diversidad de la población a la vez que mejora la convergencia hacia una solución válida.

Después de que Holland sentara las bases de los algoritmos genéticos, en los años siguientes fueron surgiendo estudios que hicieron evolucionar la rama. Varios de los más importantes fueron de John Koza [21] [22], quien introdujo la programación genética, técnica usada para desarrollar automáticamente programas que realicen una tarea definida por el usuario. Se optimiza una población de individuos (programas) respecto a una función de aptitud. Koza probó su viabilidad para resolver problemas de robótica o de optimización.

Tras él han seguido surgiendo nuevas técnicas dentro de los algoritmos evolutivos. Entre ellas se puede destacar el desarrollo de la *Programación genética cartesiana (CGP)*, que sustituye los árboles de búsqueda usados en la programación genética tradicional por grafos dirigidos acíclicos, lo que es muy útil, por ejemplo, en el diseño de circuitos electrónicos [23].

Otro avance que se ha popularizado es el de los *Algoritmos genéticos híbridos (AGHs)*, que combinan los algoritmos genéticos con otras técnicas de optimización para mejorar las soluciones a los problemas complejos.

En esta subrama destacan los algoritmos meméticos. Estos combinan algoritmos evolutivos con búsqueda local para mejorar la calidad y velocidad de convergencia de las soluciones. Después del cruce y de la mutación, cada nueva solución se refina explorando soluciones cercanas para encontrar mejoras. [24]

A lo largo de los años se han desarrollado distintos proyectos pioneros que han hecho uso de los algoritmos genéticos. Se puede destacar la misión de la NASA *Space Technology 5 (ST5)*, donde ayudaron a la creación de una antena ultracompacta, con resultados extraordinarios [25] [26]. También han sido empleados en áreas tan diversas como la economía o la bioingeniería, donde pueden ayudar a tareas tan diversas como predecir movimientos del mercado [27] o a modelar secuencias genéticas [28], respectivamente.

2.3. Planificación nutricional mediante algoritmos evolutivos

Los primeros acercamientos para intentar resolver problemas de optimización nutricional se dan en la década de 1940. George Stigler (1945) [29] planteó el problema de encontrar la dieta de menor coste que cumpliese con unos objetivos nutricionales. Al no poseer ordenadores, utilizó técnicas manuales.

El problema fue resuelto dos años después por Jack Laderman (1947) [30] usando programación lineal. Fue capaz de calcular la combinación de alimentos que cumpliese con los requisitos económicos y nutricionales. Con esto se demostró la utilidad de técnicas computacionales en tareas de optimización.

Tras los estudios de Holland, en las siguientes décadas aparecieron distintos trabajos que hacían uso de los algoritmos genéticos. Si bien estos no estaban relacionados directamente con la planificación nutricional, sí que sentaron las bases para la resolución de problemas de optimización. Se puede destacar los trabajos de Scott Kirkpatrick et al. (1983) [31] y de David E. Goldberg (1989) [32].

Fue a partir de la década de los 2000 cuando aparecieron muchos de los trabajos y estudios relacionados con la planificación nutricional.

El trabajo de Kahraman y Seven (2005) [33] desarrolla un algoritmo genético bi-objetivo que propone comidas diarias saludables basados en parámetros especificados por el usuario a través de la interfaz gráfica, como la edad o el género. Optimiza la selección de platos para cumplir con restricciones nutricionales, minimizar costos y maximizar las preferencias del usuario.

Kaldirim y Köse (2006) [34] hacen una continuación de este trabajo. Hace uso del algoritmo multiobjetivo NSGA-II para gestionar de manera separada los objetivos de coste mínimo y máxima satisfacción, mejorando el manejo de restricciones nutricionales con la inclusión de límites. También presenta una mejora a la interfaz gráfica del anterior proyecto.

El estudio de Kashima et al. (2009) [35] presenta una interesante diferencia respecto a trabajos anteriores. Busca crear una aplicación web para ofrecer un servicio para compartir menús, creados para cada individuo haciendo uso de algoritmos genéticos, entre una comunidad de usuarios con el objetivo de fomentar los hábitos saludables.

Heinonen y Juuso (2016) [36] presentan el uso que le dan a los algoritmos genéticos en su aplicación Nutri-Flow, un software que proporciona guías dietéticas personalizadas. Emplea un sistema experto difuso (*Fuzzy Expert System, FES*) junto con algoritmos genéticos para optimizar las recomendaciones. El FES evalúa los alimentos según sus características nutricionales y su contribución a las necesidades dietéticas individuales, mientras que los AGs buscan la combinación óptima de estos alimentos para ajustar la ingesta diaria.

En el trabajo realizado por Kilicarslan et al. (2021) [37] se propone un modelo híbrido que combina algoritmos genéticos con aprendizaje profundo para la predicción y clasificación de anemias nutricionales. Optimiza los parámetros de los algoritmos de aprendizaje mediante computación evolutiva, lo que mejora la precisión de la planificación nutricional del paciente.

Joanne B. Cole y Rosita Gabbiannelli (2022) [38] muestran cómo se puede integrar la IA con el análisis genético para la nutrición personalizada del paciente. Basándose en el perfil genético y otros datos biométricos, es posible, junto a los algoritmos evolutivos, generar planes de comida y predicciones sobre la salud del usuario.

3.

Marco teórico

En 1975 John Holland propone imitar los procesos biológicos naturales que rigen la selección natural usando ordenadores, lo que sería el principio de los algoritmos genéticos (AGs) [20]. En un AG, las soluciones son modeladas como individuos o cromosomas, que generalmente se representan como cadenas de bits, aunque también se puede usar otro tipo de cadenas. Estas soluciones son evaluadas por una función de aptitud o fitness, y las más adecuadas son seleccionadas para reproducirse mediante cruzamiento y mutación, procesos que mezclan y alteran aleatoriamente los cromosomas para generar diversidad. Los descendientes resultantes forman nuevas generaciones que vuelven a ser evaluadas, creando un ciclo que se repite hasta cumplir alguna condición de parada, como pudiera ser un número limitado de generaciones o que se alcance la solución deseada.

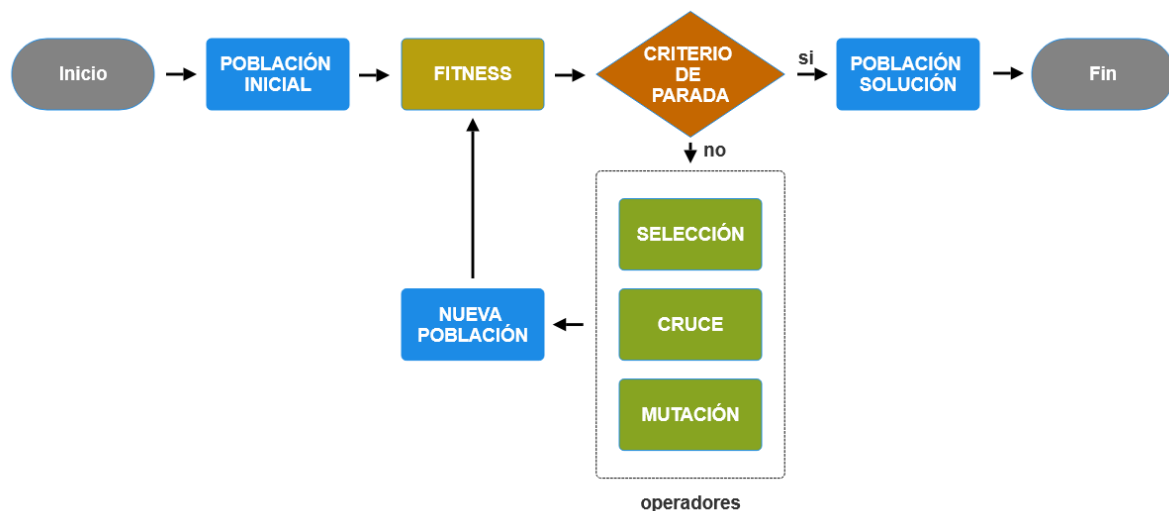


Figura 3.1. Algoritmo genético simple.

Explicado el concepto general, se va a desglosar cada una de las fases de la figura 3.1, que muestra el esquema básico de un algoritmo genético.

3.1. Población

Antes de explicar la generación de una población inicial, es necesario conocer algunos conceptos que son usados para representar y entender las soluciones. Son términos que son usados en la biología y en el campo de la genética.

- **Gen.** Es la unidad básica de información en un cromosoma. En una cadena de bits que representa una solución, cada bit puede considerarse un gen. En el caso que se trata en este PFG, el menú semanal, cada tipo de comida se podría considerar un gen. Por ejemplo, un gen sería bebida, plato principal o postre.
- **Alelo.** Es la forma específica o valor que puede tomar un gen. Siguiendo el ejemplo de bebida, posibles alelos serían agua, té o cerveza.
- **Cromosoma.** Es una colección de genes y representa una solución completa al problema de optimización. El cromosoma es la cadena de bits completa. En nuestro caso, la lista completa de alimentos seleccionados para una semana es el cromosoma.
- **Fenotipo.** Es la manifestación real de la solución codificada. En el PFG sería cómo se preparan y sirven estos alimentos en la realidad.

En la figura 3.2 se expone la estructura de un cromosoma con el ejemplo del menú semanal de comida.

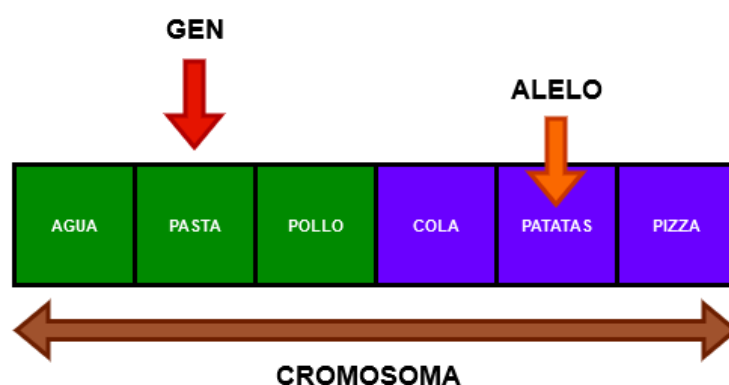


Figura 3.2. Estructura de un cromosoma.

La generación de una población inicial implica crear un conjunto de soluciones candidatas. Generalmente, los individuos son seleccionados aleatoriamente dentro de los límites definidos en cada problema, asegurando que todas las áreas del espacio de búsqueda puedan ser exploradas. También existen métodos alternativos de generación que aplican ciertas restricciones para formar soluciones iniciales más prometedoras. Tomando de ejemplo el menú semanal, el espacio de búsqueda comprendería la base de datos en la que aparecen todos los alimentos con sus respectivas calorías y macronutrientes.

3.2. Evaluación

Se mide lo bueno que es un individuo para nuestros propósitos (la calidad del individuo). Lo más importante es definir una función de fitness correcta y representativa del problema a evaluar. Si se trata de un problema multiobjetivo, se tendrá que diseñar una función de fitness diferente por cada objetivo.

Según vayan pasando generaciones, la población inicial irá evolucionando hacia poblaciones candidatas que presentarán una mejor aptitud. Si la población ha alcanzado el objetivo, la condición de parada se activará, convirtiendo el conjunto de individuos candidatos en la población solución. En caso de no alcanzarlo, seguirá evolucionando hacia otra población candidata distinta.

Para diseñar una función de fitness acorde al problema y al objetivo (u objetivos) se pueden seguir los siguientes pasos:

1. Determinar si el objetivo es maximizar o minimizar un valor específico. En el caso que se trata en este proyecto, se busca que la diferencia entre las calorías necesarias y la suma de las calorías de los alimentos sea la menor posible, por lo que se trataría de un problema de minimización.
2. Identificar las restricciones. En el ejemplo de los alimentos, una podría ser los límites superior e inferior de calorías permitidas.
3. Definir la función matemática que represente el objetivo del problema. Si existen múltiples objetivos, se pueden ponderar.
4. Incorporar las penalizaciones para soluciones que no cumplan con las restricciones.

Por lo tanto, la mejor solución debería recibir la mejor evaluación. Esto ayudará al algoritmo a entender el camino a seguir, ya que penalizará las soluciones no factibles.

3.2.1. Restricciones

Uno de los apartados a destacar cuando se diseña la función de calidad es entender cómo se manejan las restricciones para asegurar que las soluciones generadas sean viables y de alta calidad.

Existen tres tipos de restricciones que limitan el espacio de soluciones válidas. Sea x las variables del problema:

- Restricciones de igualdad. $h_j(x) = 0$
- Restricciones de desigualdad. $g_k(x) \geq 0$ ó $g_k(x) \leq 0$
- Restricciones de contorno o de caja (*box-constraints*). $x^L \leq x \leq x^U$

Estos tipos de restricciones se pueden tratar mediante distintos métodos, que también son probados de manera práctica en el apartado 4.5.1. En el trabajo de Carlos A. Coello (2022) [39] se hace un recopilatorio de algunos de los métodos más usados y sus consiguientes ventajas y desventajas. Los más importantes son:

- **Funciones de penalización.** Se penalizan las soluciones no factibles. Estas penalizaciones representan cómo de lejos está la solución de la región factible. Se convierte un problema con restricciones en uno sin restricciones usando una función objetivo de penalización:

$$F(x) = f(x) + \sum_{j=1}^m R_j \cdot p_j(x)^\beta$$

- $f(x)$ es la función objetivo original que se desea optimizar.
- R_j es un parámetro de penalización definido por el usuario para la restricción $c_j(x)$, cuya función de penalización es $p_j(x)$
- β (comúnmente 1 o 2) es una constante definida por el usuario que determina cómo se pondera la violación de la restricción en la función de penalización.
- En problemas de maximización la suma se convierte en negativa.

Existen varios métodos de penalización:

- **Penalización estática.** Se usa un valor fijo R_j . Se puede ver su implementación en el problema del PFG en el punto 4.5.1.
 - **Penalización dinámica.** Se usa un factor $R = (C \cdot t)^\alpha$, donde C y α son constantes y t es el número de iteración.
 - **Penalización adaptativa.** La penalización se adapta en función de la factibilidad de las soluciones.
-
- **Separatista.** Dividen el problema en subproblemas más manejables que se pueden resolver por separado y luego combinar. Las restricciones y los objetivos se tratan de forma separada. Un método separatista se trata en el punto 4.5.1.
 - **Reparación.** Se modifican las soluciones candidatas inviables de forma que no violen las restricciones.

3.2.2. Condición de parada

Tras la evaluación de calidad la condición de parada determina cuándo el algoritmo debe detenerse y presentar la solución. Si bien existen una gran variedad de condiciones de parada, las más comunes son:

- **Aptitud del individuo.** El algoritmo se detiene si se alcanza una solución, ya sea alcanzando el valor objetivo o estando dentro de unos umbrales predefinidos.
- **Número máximo de generaciones.** Se para si la población ya ha evolucionado un número de veces predefinido.
- **Tiempo máximo de ejecución.** Se detiene si el tiempo de ejecución es demasiado elevado.
- **Convergencia.** Se puede detener la ejecución si la población muestra poca o ninguna mejora en la aptitud durante un número determinado de generaciones consecutivas.

3.3. Operadores

Son los procesos que se aplican a poblaciones de individuos para desarrollar generaciones futuras. Sirven para la exploración del espacio de soluciones y para la mejora de las poblaciones a través de las generaciones. Se busca que el conjunto de individuos presente una alta diversidad, ya que si es baja se corre el riesgo de caer en mínimos locales, lo que no permitiría explorar el espacio de soluciones ampliamente. En la figura 3.3 aparecen los operadores básicos en los algoritmos evolutivos, que son los que se van a explicar en más detalle.



Figura 3.3. Operadores.

3.3.1. Selección

El primero de los operadores básicos. Elige un individuo para reproducirlo. Comúnmente, se busca un equilibrio entre la diversidad y la convergencia. El algoritmo debe explorar el espacio de búsqueda en amplitud, ya que podría caer en mínimos locales si la población es muy parecida entre sí, lo que no permitiría encontrar buenas soluciones. Pero también es de interés que las poblaciones candidatas vayan evolucionando hacia una mayor aptitud, reconduciendo el algoritmo por zonas del espacio de búsqueda más probables de encontrar una solución.

Si bien se puede seleccionar de manera equiprobable, existen métodos basados en la aptitud del individuo que buscan respetar este equilibrio. Estos son algunos:

- **Método estándar (rueda de la fortuna).** Asigna a cada individuo una probabilidad proporcional a su fitness, por lo que los más aptos tienen mayor probabilidad de reproducirse. Existe un derivado de este método en el que se normalizan las probabilidades, lo que favorece aún más a los de mayor calidad.

- **Método del rango.** Se ordenan los individuos según su aptitud, y la probabilidad de selección se asigna según este ranking.
- **Selección por torneo.** Se escoge aleatoriamente un subconjunto de individuos de la población, y el mejor de este es elegido. Este es el método de selección usado en el desarrollo del algoritmo nutricional (sección 4.5).

Selección ambiental (Reemplazo)

A parte de los métodos basados en la aptitud, se puede usar la técnica de la selección ambiental para seleccionar aquellos individuos que puedan influir significativamente en la eficacia del algoritmo genético. Existen varios métodos de selección ambiental:

- **Reemplazo generacional completo.** Método tradicional en la selección y explicado en el apartado 3.3.1. Toda la población actual es reemplazada por una nueva. Promueve altas tasas de diversidad, pero conlleva una aptitud de los individuos irregular.
- **Reemplazo generacional parcial (*Steady-State*).** Solo una parte de la población es reemplazada en cada generación. Se promueve una mejora constante en la calidad ya que las soluciones con un fitness elevado pueden mantenerse.
- **Reemplazo elitista.** Escoge los mejores individuos de una generación para que se mantengan en la siguiente. Esto ayuda a que la calidad del mejor individuo de una generación sea siempre igual o superior a su equivalente de la generación anterior, consiguiendo una progresión constante hacia la solución óptima, es decir, una mayor convergencia.
- **Selección (μ, λ) .** De μ padres se generan λ hijos. Los μ mejores de los λ hijos forman la siguiente generación de padres. Es decir, si se tiene 50 padres (μ) y se generan 100 descendientes (λ), se selecciona los 50 mejores de los 100 descendientes para la siguiente generación.
- **Selección $(\mu + \lambda)$.** Igual que en la anterior, de μ padres se generan λ hijos, pero en este caso los μ mejores de la combinación entre los μ padres y los λ hijos forman la siguiente generación. En este caso, con el mismo ejemplo de los 50 padres y los 100 descendientes, se selecciona los 50 de los 150 individuos combinados para la siguiente generación.

3.3.2. Cruce

Se combina el material genético de dos individuos, padres, para producir descendencia, hijos. Es decir, se utiliza para intercambiar características de dos soluciones parentales con el objetivo de generar nuevas soluciones. Se aplica con probabilidad P_c . Al promover la mezcla de genes, ayuda a aumentar la diversidad y la convergencia, debido a la generación de nuevos descendientes con características deseables. Se usan distintos métodos:

- **Cruce de un punto.** Como se muestra en la figura 3.4, se selecciona aleatoriamente una posición en el cromosoma. Todos los genes que están antes de este punto se intercambian con los que están después en la otra cadena, y viceversa, para producir dos nuevos descendientes.

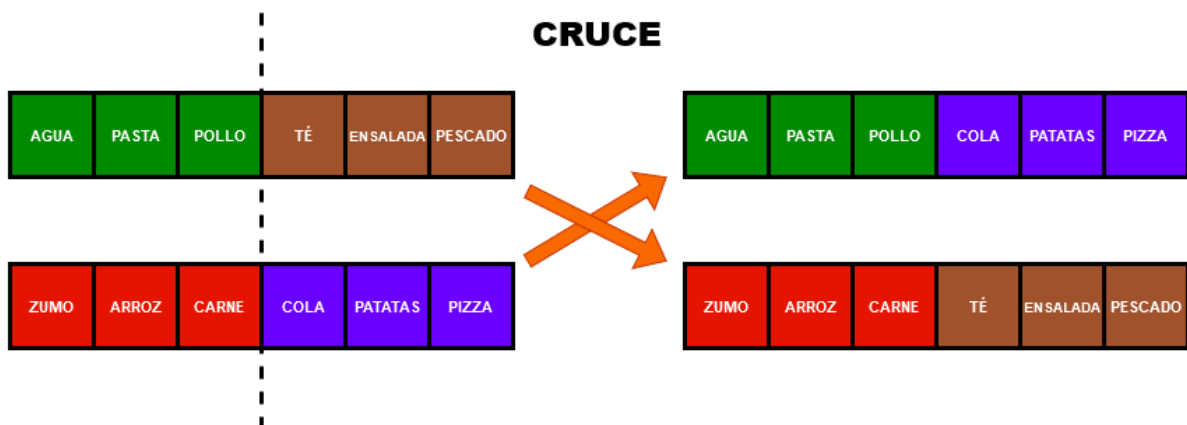


Figura 3.4. Cruce de un punto.

- **Cruce de dos puntos.** Similar al cruce de un punto, pero se seleccionan dos puntos de corte. Las cadenas de genes entre estos dos puntos se intercambian entre los dos padres.
- **Cruce uniforme.** Cada gen tiene una probabilidad igual de ser elegido de uno de los dos padres.

3.3.3. Mutación

Último de los operadores básicos. Se recorre toda la cadena, mutando cada gen con probabilidad P_m , es decir, eligiendo un nuevo valor mediante una elección equiprobable sobre el alfabeto. La mutación aumenta la diversidad y ayuda a no caer en mínimos locales. Algunos de los métodos usados son:

- **Mutación uniforme.** Cada gen puede cambiar a otro valor con probabilidad P_m , como se expone en la figura 3.5.



Figura 3.5. Mutación uniforme.

- **Mutación por intercambio.** Dos genes aleatorios en el cromosoma son seleccionados y sus posiciones se intercambian.

3.4. Tipos de algoritmo

Si bien existen casos en los que solo es necesario la optimización de un objetivo, existe una gran diversidad de problemas en la que van a aparecer varios que se deben maximizar o minimizar. Debido a esto, existe una clasificación de los problemas según el número de objetivos a optimizar. Destacan dos grandes grupos: los problemas de optimización con un solo objetivo o *Single-objective Optimization Problem (SOP)*, y los problemas de optimización multiobjetivo o *Multi-objective Optimization Problem (MOP)*.

3.4.1. Single-objective Optimization Problem (SOP)

Los algoritmos que resuelven problemas *SOP* se enfocan en optimizar una única función objetivo. La meta es encontrar la mejor solución posible según se busque maximizar o minimizar la función. Se usan algoritmos mono-objetivo cuando existe un único criterio de éxito.

Los algoritmos mono-objetivo presentan una función objetivo que asigna un valor numérico a cada solución, indicando su calidad. Cada solución está representada por un vector de variables de decisión, que son los parámetros que se pueden ajustar dentro del espacio de decisión. Estas soluciones deben pertenecer al conjunto factible, que incluye aquellas que cumplen con todas las restricciones del problema.

El algoritmo más usado para los problemas mono-objetivo es el "*Genetic Algorithm*" (GA), cuyo funcionamiento sigue los pasos explicados a lo largo del capítulo 3. Destaca por su simplicidad y su capacidad para explorar grandes espacios de búsqueda.

3.4.2. Multi-objective Optimization Problem (MOP)

Los algoritmos multi-objetivo buscan optimizar múltiples funciones simultáneamente. La meta es encontrar un conjunto de soluciones que represente un equilibrio entre los distintos criterios de éxito, conocido como conjunto de Pareto.

Una solución se considera no dominada si no existe otra que mejore en, al menos, uno de los objetivos sin empeorar en los demás. El conjunto de todas las soluciones no dominadas se conoce, como se ha dicho antes, como el conjunto de Pareto.

Al igual que en los mono-objetivo, cada solución está representada por un vector de variables de decisión. Estas soluciones deben pertenecer también al conjunto factible. La diferencia se encuentra en la existencia de múltiples funciones objetivo que asignan valores numérico a cada solución, indicando su calidad en diferentes dimensiones según el número de objetivos.

Algunos de los algoritmos más usados para los *MOP* son el "*Non-dominated Sorting Genetic Algorithm II*" (NSGA-II), el "*Strength Pareto Evolutionary Algorithm 2*" (SPEA2) o el "*Multi-Objective Evolutionary Algorithm based on Decomposition*" (MOEA/D). Usan criterios basados en Pareto para evaluar, clasificar y seleccionar las soluciones. Buscan dispersar las soluciones para aumentar la diversidad y explorar el espacio de soluciones. Aun así, presentan grandes diferencias a la hora de desarrollar el algoritmo evolutivo.

Non-dominated Sorting Genetic Algorithm II (NSGA-II)

Este algoritmo sigue el funcionamiento general de un algoritmo genético, pero presenta ciertas modificaciones para los algoritmos multi-objetivo, como se muestra en la figura 3.6. Tras realizar la evaluación, las soluciones se ordenan en diferentes frentes de Pareto. En el primer frente se encuentran las soluciones no dominadas, en el segundo las soluciones dominadas solo por soluciones del primer frente, y así sucesivamente. Tras esto, se calcula el "crowd distance", una medida usada para preservar la diversidad de las soluciones. Se evalúa cómo de cerca están las soluciones entre sí en el mismo frente. Cuanto mayor sea la distancia, más aislada y diversa es la solución, lo que ayuda a explorar en profundidad el espacio de soluciones.

NSGA-II usa elitismo, ya que combina la población de los padres y los descendientes y escoge a los mejores individuos para la siguiente generación basados en la clasificación de Pareto y en el "crowd distance". Además, se hace uso de la selección por torneo para elegir a los padres que se usarán en la reproducción, también usando de referencia las medidas antes citadas.

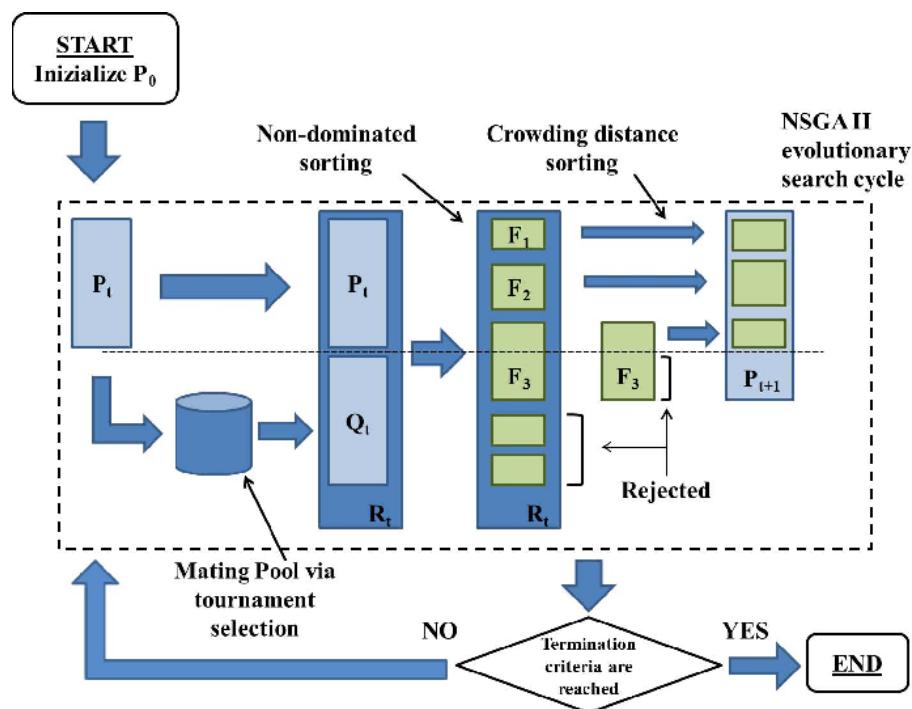


Figura 3.6. Estructura general del algoritmo NSGA-II. Fuente [41]

Strength Pareto Evolutionary Algorithm 2 (SPEA2)

El *SPEA2* también está optimizado para problemas multi-objetivo y, al igual que el *NSGA-II*, tiene modificaciones respecto al funcionamiento básico del algoritmo genético.

En la evaluación, a cada solución se le asigna una fuerza que refleja el número de soluciones que domina, como se expone en la figura 3.7. Tras esto, se usa una medida de densidad para evaluar cuán poblada está el área alrededor de cada solución. El fitness se calcula sumando la fuerza y esta medida de densidad, lo que ayuda a la diversidad de la población. En la selección se escogen aquellas soluciones que mayor fitness presentan.

SPEA2 hace uso de un archivo externo para almacenar soluciones no dominadas. Este archivo se actualiza en cada generación y mantiene un conjunto de soluciones que representan la mejor aproximación al frente de Pareto en ese momento.

Similar al *NSGA-II*, este algoritmo escoge los mejores individuos de la combinación de padres y descendientes para la siguiente generación. [42]

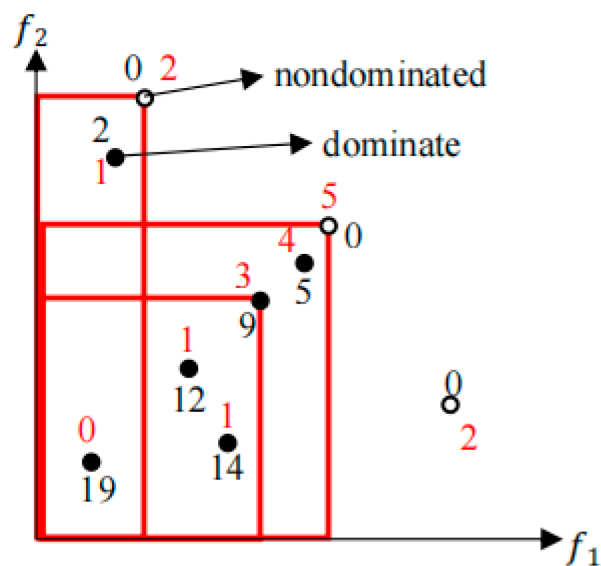


Figura 3.7. Cálculo del fitness en SPEA2. Fuente [43]

Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)

MOEA/D hace uso de direcciones de referencia para dividir el problema multi-objetivo en problemas más pequeños de un solo objetivo. Cada dirección de referencia representa una combinación específica de los distintos objetivos. De manera ponderada, cada referencia se centra en la optimización de un objetivo particular. En el listado 3.1 se muestra un ejemplo con 12 direcciones de referencia que puede ser utilizado en este proyecto, donde se busca una buena cobertura del espacio de búsqueda, que presenta 3 dimensiones (3 objetivos).

Listado 3.1. Direcciones de referencia.

(1, 0, 0),	(0.67, 0.33, 0),	(0.33, 0.67, 0),
(0, 1, 0),	(0.67, 0, 0.33),	(0.33, 0, 0.67)
(0, 0.67, 0.33),	(0, 0.33, 0.67),	(0, 0, 1),
(0.33, 0.33, 0.33),	(0.67, 0.16, 0.16),	(0.16, 0.67, 0.16)

Para generar estas direcciones existen distintos métodos, incluyendo "*das-dennis*" e "*incremental*". El método "*das-dennis*" es usado para distribuir uniformemente las direcciones, dividiendo el espacio en particiones equitativas. Mientras tanto, el método "*incremental*" ajusta las direcciones de referencia gradualmente, permitiendo una adaptación más precisa a las áreas del espacio de búsqueda que son más prometedoras.

Para cada subproblema, se selecciona un número determinado de subproblemas más cercanos. Estos subproblemas forman el vecindario del primero, como se ilustra en la figura 3.8. Existe una probabilidad P_{cv} de que una solución se cruce con otra de su mismo vecindario. Compartir información entre vecinos ayuda a generar soluciones que cumplan cada uno de los subproblemas, por lo que son útiles para mejorar la calidad y la diversidad. [44]

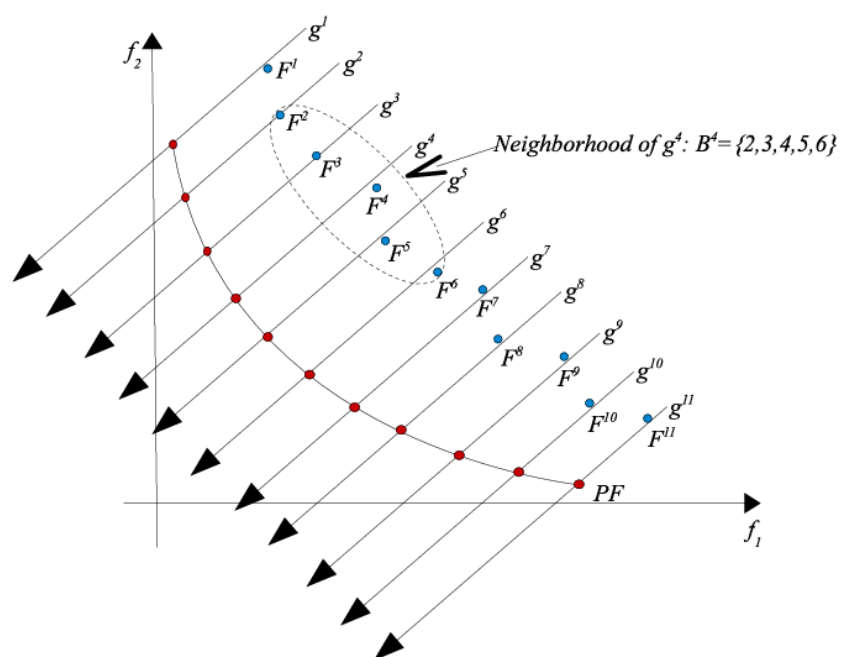


Figura 3.8. Vecindarios en MOEA/D. Fuente [45]

4.1. Base de datos

4.1.1. Selección de la base de datos

La base de datos de alimentos y platos utilizada para el desarrollo del algoritmo es la perteneciente al gobierno de Reino Unido. El proyecto proporciona información detallada sobre la composición nutricional de los alimentos consumidos comúnmente en Reino Unido. [46]

La base de datos "*Composition of foods integrated dataset (CoFID)*" se originó a partir del trabajo de Widdowson y McCance en la década de 1940 y ha sido periódicamente actualizada desde entonces para reflejar los últimos cambios en la dieta y las nuevas investigaciones científicas. La última actualización se publicó en marzo de 2021.

La elección de esta base de datos se basa en la gran cantidad de datos nutricionales que se aportan sobre los alimentos, junto con una amplia documentación que explica en profundidad cada uno de los detalles necesarios para entender el trabajo. A todo esto hay que sumar la facilidad para la exportación, que permite modificarla y analizarla con suma sencillez.

CoFID incluye más de 2500 alimentos, cada uno de ellos categorizado según su grupo alimenticio. Cada alimento incorpora información sobre una amplia gama de nutrientes, como vitaminas, minerales, azúcares y otros componentes, además de los macronutrientes (carbohidratos, proteínas y grasas), que van a ser los principalmente usados en este PFG.

En la página web se pueden encontrar principalmente dos archivos. El primero es una guía de usuario en formato PDF que explica en detalle en qué consiste la base de datos y la información que se va a encontrar en ella.

El segundo archivo trata de una hoja de cálculo de Microsoft Excel donde se encuentra toda la información nutricional relativa a 2.887 alimentos. Existen múltiples casos en los que se presenta un mismo alimento con diversos tipos de cocción o de aliño, lo que provoca un cambio en los valores nutricionales, razón por la que se considera como dos alimentos o platos distintos.

La información requerida se encuentra en la tabla “1.3. Proximates”, que es donde se localizan los campos que se van a usar: “Food Code”, identificador del alimento; “Food Name”, nombre del alimento; “Group”, categoría alimenticia a la que pertenece; “Protein(g)”, que muestra los gramos de proteína por cada 100g del alimento; “Fat(g)”, lo mismo que la proteína, pero con la grasa; “Carbohydrate(g)”, al igual que los dos últimos campos, muestra datos de los hidratos de carbono o carbohidratos; y “Energy(kcal)”, que representa las kilocalorías del alimento por cada 100g.

Como se puede comprobar por las descripciones previas, todos los datos están representados en base a 100 gramos del alimento correspondiente. Esto también ocurre con las bebidas, cuyos valores nutricionales se han calculado en base 100 mililitros. La planificación nutricional se realizará usando estas proporciones.

Por último, destacar que, debido a que la información es extraída de una web oficial del Reino Unido, todos los datos, incluyendo entre ellos el nombre de los alimentos o platos, están en inglés. En el menú de comidas resultante de la ejecución del algoritmo genético se mantendrá este idioma.

4.1.2. Creación de la base de datos

Tras haber seleccionado la fuente, se necesita un gestor de bases de datos para poder manipular los datos eficientemente. Para este trabajo se ha seleccionado MySQL Workbench, que dispone de una interfaz gráfica intuitiva para diseñar, modelar y administrar bases de datos MySQL. [47]

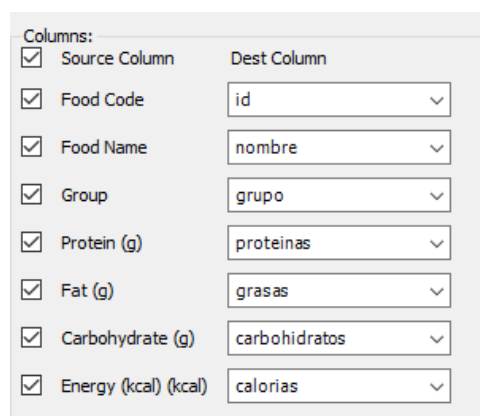
Dentro del gestor, lo primero que se debe crear es la propia base de datos. Se va a hacer uso de sentencias SQL para ello. A la base de datos se le ha llamado “food_database”.

Tras la configuración de la base de datos, hay que crear la tabla con los campos en los que se almacenarán los valores. Mostrando en el listado 4.1 el código ejecutado, esta será la única tabla necesaria a lo largo del proyecto.

Listado 4.1. Creación de la tabla y sus campos.

```
CREATE DATABASE food_database;  
CREATE TABLE comida (  
    id VARCHAR(10) NOT NULL,  
    nombre VARCHAR(255) NOT NULL,  
    grupo CHAR(3),  
    proteinas FLOAT,  
    grasas FLOAT,  
    carbohidratos FLOAT,  
    calorías INT,  
    PRIMARY KEY(id)  
);
```

Ahora que la estructura ya está creada, falta introducir los datos. MySQL Workbench dispone de una herramienta para la importación desde archivos Comma Separated Values (valores separados por comas) o CSV, que permite con facilidad identificar los numerosos datos de la hoja de cálculo. Consigue emparejar fácilmente los campos de la tabla recién creada con los del archivo CSV, como muestra la figura 4.1, lo que da como resultado una correcta importación masiva de datos.



Columns:	
<input checked="" type="checkbox"/> Source Column	Dest Column
<input checked="" type="checkbox"/> Food Code	id
<input checked="" type="checkbox"/> Food Name	nombre
<input checked="" type="checkbox"/> Group	grupo
<input checked="" type="checkbox"/> Protein (g)	proteinas
<input checked="" type="checkbox"/> Fat (g)	grasas
<input checked="" type="checkbox"/> Carbohydrate (g)	carbohidratos
<input checked="" type="checkbox"/> Energy (kcal) (kcal)	calorias

Figura 4.1. Importación de los datos usando MySQL Workbench.

4.1.3. Cambios en la base de datos original

Para optimizar la base de datos y mejorar el funcionamiento del algoritmo con el fin de alcanzar mejores soluciones, existen ciertos cambios a realizar a los datos aportados por el gobierno de Reino Unido.

El primer cambio es la eliminación de los datos incompletos. En la base de datos original existen diversos alimentos que no presentan toda la información que se va requerir en el trabajo. El principal promotor de este cambio es la existencia de registros que no tienen calorías asignadas (representadas en el archivo Excel con una "N"), que han pasado con el valor 0.

Sin dejar de lado la eliminación de registros, se filtra las categorías de comidas que pueden ser ingeridas por el consumidor. Existen categorías como, por ejemplo, "*Harinas, granos y almidones*" (AA) o "*Grasas y aceites*" (O), que no son consumidas por el usuario como plato único, sino que participan en la cocción o en el acompañamiento del alimento principal. Por lo que se eliminan. También se elimina la categoría "*Bebidas Alcohólicas*" (Q), ya que sería necesario un análisis más exhaustivo y conocer las recomendaciones de los nutricionistas respecto a su consumo. De esta categoría se mantienen la cerveza, el vino, la sidra y el cava, presentes en la dieta mediterránea y de las que la Sociedad Española de Nutrición Comunitaria (SENC) recomienda el consumo opcional, moderado y responsable [48].

También se ha modificado el grupo "*Zumos*" (PE), donde se ha cambiado PE por FE. Se ha movido al grupo "*Frutas*" (F) porque los zumos son mucho más parecidos a los alimentos de este grupo que al de las bebidas (P).

En el apéndice A se encuentran todos los grupos con los que se va a trabajar.

Tras los cambios realizados queda un total de 2616 registros. En la figura 4.2 se puede ver un ejemplo de algunos de los alimentos.

id	nombre	grupo	proteinas	grasas	carbohidratos	calorias
11-1000	Stuffing, sage and onion, homemade	AT	5.6	13.3	29.4	253
11-1001	Bread, white, toasted	AF	9.7	2	56.2	267
11-1003	Bread, white, 'with added fibre'	AF	8.7	2.2	45.1	224
11-1004	Bread, white, 'with added fibre', to...	AF	10.4	2.6	53.7	266
11-1005	Cake, fruit, rich, homemade	AN	4	11.1	61.5	349
11-1006	Bread rolls, white, soft	AF	9.3	2.6	51.5	254
11-1007	Pie, fruit, one crust, homemade	AS	2.1	9.1	26.7	190
11-1008	Pie, fruit, pastry top and bottom, ...	AS	3.1	15.1	30.4	262
11-1009	Currant buns, retail	AF	8	5.6	52.6	280
11-1010	Pastry, wholemeal, uncooked, ho...	AO	7.2	25.9	42.3	421
11-1011	Pizza, fish topped, takeaway	AE	13.3	7.5	28	226
11-1012	Pizza, chicken topped, retail	AE	13.4	8.3	31.3	246
11-1013	Pizza, ham and pineapple, retail	AE	13.5	8.6	34.4	260

Figura 4.2. Resultado final de la base de datos.

4.2. Conceptos nutricionales

Se ha hecho uso de distintas fuentes para los valores nutricionales utilizados para el cálculo de los objetivos, restricciones y límites que se representan en el problema.

En el algoritmo se busca crear un menú equilibrado y personalizado para el usuario, con el propósito de mantener el peso y la masa corporal del individuo.

4.2.1. Cálculo de calorías

El primer objetivo es el calórico, explicado en el apartado 4.4.1. Para ello es necesario saber cuántas kilocalorías diarias necesita el usuario. La Tasa Metabólica Basal (TMB) es la cantidad de energía que el cuerpo necesita para mantener las funciones vitales en reposo. Existen diversas fórmulas de calcularla, pero en este proyecto se hace uso de una las más extendidas actualmente, la fórmula de Harris-Benedict revisada por Mifflin et al. [49]. Esta fórmula toma en cuenta el peso, la altura, la edad y el sexo para estimar las kilocalorías.

- Para hombres:

$$TMB = (10 \times \text{peso en kg}) + (6,25 \times \text{altura en cm}) - (5 \times \text{edad en años}) + 5$$

- Para mujeres:

$$TMB = (10 \times \text{peso en kg}) + (6,25 \times \text{altura en cm}) - (5 \times \text{edad en años}) - 161$$

Como se explica en la definición de la TMB, esta energía es calculada en reposo. Por lo tanto, para alcanzar una aproximación real será necesario multiplicar el resultado por un factor que depende del nivel de actividad física. [50]

- Sedentario (poco o ningún ejercicio): $TMB \times 1.2$
- Actividad ligera (ejercicio ligero o deportes 1-3 días a la semana): $TMB \times 1.375$
- Actividad moderada (ejercicio moderado o deportes 3-5 días a la semana): $TMB \times 1.55$
- Actividad alta (ejercicio intenso o deportes 6-7 días a la semana): $TMB \times 1.725$
- Actividad muy alta (ejercicio muy intenso, trabajos físicos o entrenamiento dos veces al día): $TMB \times 1.9$

Por ejemplo, para un hombre de 23 años que pesara 75 kg y midiera 175 cm su TMB sería de 1734 kilocalorías por día. Pero como también realiza una actividad moderada, habría que multiplicar por 1.55, dando un total de 2687.31 kcal/día.

4.2.2. Distribución de macronutrientes

Otro objetivo, tratado en el punto 4.4.2, es la correcta distribución diaria de macronutrientes, es decir, de los hidratos de carbono, de las proteínas y de las grasas. El *Institute of Medicine (IOM)*, en su trabajo de 2005 [51], define los AMDR (Acceptable Macronutrient Distribution Ranges) como los rangos de ingesta para macronutrientes que se asocian con un riesgo reducido de enfermedades crónicas y aseguran una ingesta adecuada de nutrientes esenciales. Estos límites para adultos son:

- $45\% \leq \% \text{ de carbohidratos} \leq 65\%$
- $10\% \leq \% \text{ de proteínas} \leq 35\%$
- $20\% \leq \% \text{ de grasas} \leq 35\%$

Para saber si la solución respeta estos límites, primero es necesario saber cuántas kilocalorías aporta cada macronutriente. Siguiendo el estudio del IOM, las kilocalorías por gramo de cada macronutriente son:

- Carbohidratos: 4 kcal por gramo
- Proteínas: 4 kcal por gramo
- Grasas: 9 kcal por gramo

La suma de las kilocalorías individuales de cada macronutriente equivale al total de calorías del alimento, dato del que ya se dispone.

Con las kilocalorías de cada macronutriente ya calculadas, solo falta dividir estas entre las calorías totales y multiplicarlo por 100 para saber los porcentajes de cada macronutriente. Por ejemplo, para calcular el porcentaje de carbohidratos sería:

$$\left(\frac{\text{kcal de carbohidratos}}{\text{kcal totales}} \right) \times 100$$

Siguiendo el caso de los hidratos de carbono, 11 g de carbohidratos de un alimento de 98 kcal representa un porcentaje del 44.9 % respecto al total de calorías.

4.3. Cromosoma planteado

Se ha buscado plantear un menú siguiendo los hábitos alimenticios y nutricionales de los consumidores españoles.

La primera parte es la elección de la estructura del cromosoma solución. En este problema se ha planteado que cada gen represente un alimento diferente consumido. Tal y como se muestra en la figura 4.3, cada día del menú semanal consta de un total de 5 comidas repartidas a lo largo del mismo: desayuno, tentempié, almuerzo, merienda y cena. De estas comidas, las tres principales, el desayuno, el almuerzo y la cena, consta cada una de 2 alimentos y 1 bebida, haciendo un total de 3 genes. Las otras dos, el tentempié y la merienda, consideradas como comidas entre horas, tienen 1 alimento cada una, es decir, añadiendo 2 genes más al cromosoma.

Cada día presenta al final 11 genes (o alimentos). Como el proyecto trata de una planificación semanal, se repite esta cadena por cada día de la semana, dando un cromosoma que presenta un total de 77 genes.

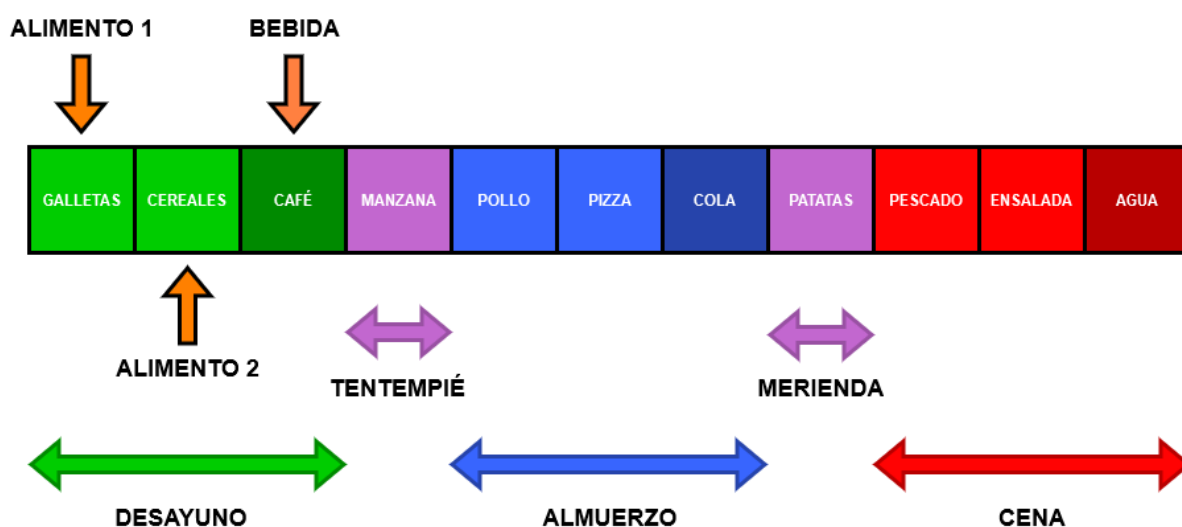


Figura 4.3. Estructura de la solución.

Como se ha explicado, el tercer gen de cada comida principal está reservado para una bebida, por lo que es necesario filtrar los grupos de comida para que seleccione solo un alimento de esta categoría. Estos cambios que se realizan en la inicialización y mutación son explicados en el apartado 4.5, pero primero se define qué categorías de alimentos (detalladas en el apéndice A) están reservadas para ciertos genes o comidas.

- El tercer gen de cada comida principal está destinado a una bebida. De las categorías disponibles, solo se puede seleccionar de los grupo "*Bebidas*" (P), "*Jugos de frutas*" (FC), "*Zumos*" (FE) o "*Bebidas Alcohólicas*" (Q) (solo si el sujeto es mayor de edad).
- Debido a que la bebida del desayuno suele ser una bebida con leche (no presente en el grupo P), café o un zumo, se ha decidido que el valor que tenga este gen sea de las categorías "*Leche de vaca*" (BA), "*Bebidas a base de leche*" (BH), "*Bebidas en polvo, esencias e infusiones*" (PA), "*Jugos de frutas*" (FC) o "*Zumos*" (FE).
- También los alimentos del desayuno suelen diferenciarse con respecto a los del almuerzo y la cena, que son comúnmente intercambiables entre sí. Por ello los alimentos de esta comida son del grupo "*Huevos*" (C), "*Frutas*" (FA), "*Bacon*" (MAA) o "*Cereales*" (A), excepto "*Arroz*" (AC), "*Pasta*" (AD) y "*Pizza*" (AE), que se suelen consumir en el almuerzo o la cena.
- Para las comidas entre horas, tentempié y merienda, se han seleccionado alimentos de las categorías "*Frutas*" (F) y "*Azúcares*" (S).
- Para las otras dos comidas principales, el almuerzo y la cena, los alimentos pueden ser de cualquier grupo, excepto los cereales, salvo "*Panes*" (AF) y las excepciones antes mencionadas: el arroz, la pasta y las pizzas.

4.4. Objetivos y restricciones

4.4.1. Objetivo y restricción de calorías

El primer objetivo que se introduce en el problema es de las kilocalorías. Se busca que la diferencia entre las kilocalorías que el usuario necesita diariamente y la suma de las kilocalorías de todos los alimentos consumidos en un día sea 0 ó lo más cercano posible. La función de evaluación mide cuánto se ha desviado en total en toda la semana, es decir, se va sumando cada desviación de kilocalorías diaria hasta obtener un valor semanal.

Se le pregunta al usuario por distintos atributos físicos y por su nivel de actividad, con lo que se calcula la tasa metabólica basal y, en un última instancia, las kilocalorías, como se explica en el apartado [4.2.1](#).

El objetivo está atado a una restricción *box-constraint*. Se penaliza aquellas desviaciones diarias que superen unos límites superior e inferior del 10% respecto a las kilocalorías que el usuario necesita. Busca que las soluciones que se alejen bastante del objetivo sean menos seleccionadas.

$$\begin{aligned} \text{Minimizar } f_{\text{calorías}}(c_{i,d}) &= \sum_{d=1}^7 \left| k - \sum_{i=1}^{n_d} c_{i,d} \right| \\ \text{Sujeto a } 0,9k &\leq \sum_{i=1}^{n_d} c_{i,d} \leq 1,1k, \quad \forall d \in \{1, 2, \dots, 7\} \end{aligned}$$

Donde:

- $c_{i,d}$ es la cantidad de calorías del alimento i consumido en el día d .
- k es el objetivo calórico diario.
- n_d es el número total de alimentos consumidos en el día d .

4.4.2. Objetivo y restricción de macronutrientes

Este objetivo trata de distribuir correctamente los macronutrientes a lo largo del día. Se busca que cada tipo de macronutriente se encuentre dentro de los límites recomendado en el apartado 4.2.2, poniendo como objetivo la media de estos límites. Parecido al anterior objetivo, se calcula la diferencia diaria entre estas medias y los valores reales de proteínas, carbohidratos y grasas obtenidos. Posteriormente, la función evalúa la desviación total en la semana.

Existe también una restricción de caja asociada a este objetivo, donde se requiere que la ingesta diaria de cada macronutriente se encuentre dentro de los límites antes citados.

$$\begin{aligned} \text{Minimizar } f_{\text{macronutrientes}} &= \sum_{d=1}^7 \left(\left| \frac{\sum_{i=1}^{n_d} C_{i,d}}{\sum_{i=1}^{n_d} E_{i,d}} - 0,55 \right| \right. \\ &\quad \left. + \left| \frac{\sum_{i=1}^{n_d} P_{i,d}}{\sum_{i=1}^{n_d} E_{i,d}} - 0,225 \right| + \left| \frac{\sum_{i=1}^{n_d} G_{i,d}}{\sum_{i=1}^{n_d} E_{i,d}} - 0,275 \right| \right) \end{aligned}$$

Sujeto a

$$\begin{aligned} 0,45 &\leq \frac{\sum_{i=1}^{n_d} C_{i,d}}{\sum_{i=1}^{n_d} E_{i,d}} \leq 0,65, \\ 0,10 &\leq \frac{\sum_{i=1}^{n_d} P_{i,d}}{\sum_{i=1}^{n_d} E_{i,d}} \leq 0,35, \\ 0,20 &\leq \frac{\sum_{i=1}^{n_d} G_{i,d}}{\sum_{i=1}^{n_d} E_{i,d}} \leq 0,35, \\ \forall d &\in \{1, 2, \dots, 7\} \end{aligned}$$

Donde:

- $E_{i,d}$ es la cantidad de energía (calorías) del alimento i consumido en el día d .
- $C_{i,d}$ es la cantidad de calorías provenientes de carbohidratos del alimento i consumido en el día d .
- $P_{i,d}$ es la cantidad de calorías provenientes de proteínas del alimento i consumido en el día d .
- $G_{i,d}$ es la cantidad de calorías provenientes de grasas del alimento i consumido en el día d .
- n_d es el número total de alimentos consumidos en el día d .

4.4.3. Objetivo de preferencia

Este objetivo busca la personalización del menú incluyendo los grupos de alimentos favoritos del usuario, o excluyendo aquellos que no le gustan. Se favorecerá los grupos preferidos y se penalizará los que no. Se puede considerar como restricción de igualdad débil ya que, aunque penalice las soluciones que la violan, no las invalida. A diferencia de los otros objetivos, esta función de minimización puede alcanzar valores negativos si existe una gran variedad de alimentos con grupo predilecto en la planificación.

$$\text{Minimizar } f_{\text{preferencia}} = \sum_{i=1}^N \begin{cases} -P & \text{si } g(a_i) = G_+ \\ P & \text{si } g(a_i) = G_- \\ 0 & \text{en otro caso} \end{cases}$$

Donde:

- $f_{\text{preferencia}}$ es la penalización total de preferencia de grupo durante la semana.
- N es el número total de alimentos consumidos en la semana.
- a_i es el alimento i consumido durante la semana.
- G_+ es el grupo que gusta.
- G_- es el grupo que no gusta.
- P es la penalización de preferencia.
- $g(a_i)$ es el grupo del alimento a_i .

4.4.4. Restricción de alergia

Similar al objetivo de preferencia, es una restricción de igualdad donde se penaliza de gran manera si el usuario es alérgico al grupo del alimento del menú. Al finalizar la semana, se suman todas las penalizaciones de alergias, haciendo que sea inviable para el algoritmo volver a seleccionar un alimento que produzca alergia.

$$\text{Minimizar } f_{\text{alergia}} = \sum_{i=1}^N \begin{cases} P_{\text{alergia}}^2 & \text{si } g(a_i) = G_{\text{alergia}} \\ 0 & \text{en otro caso} \end{cases}$$

Donde:

- f_{alergia} es la penalización total por alergia durante la semana.
- N es el número total de alimentos consumidos en la semana.
- a_i es el alimento i consumido durante la semana.
- G_{alergia} es el grupo al que el usuario es alérgico.
- P_{alergia} es la penalización fuerte por alergia.
- $g(a_i)$ es el grupo del alimento a_i .

4.5. Construcción del algoritmo

El código completo se encuentra en el repositorio de Github del autor [52].

La biblioteca usada para la creación del algoritmo evolutivo ha sido *Pymoo*. *Pymoo* es un framework de *Python* diseñado para la optimización multiobjetivo. Entre sus ventajas se encuentran la amplia gama de algoritmos para abordar los problemas, como *NSGA-II*, *SPEA2* o *MOEA/D*. Permite modificar fácilmente los operadores de selección, cruce y matación, lo que ayuda a la personalización del algoritmo según el problema. Además, *Pymoo* presenta un conjunto de herramientas de visualización para la interpretación de resultados, permitiendo entender el comportamiento de los algoritmos. [53]

Se ha creado una clase personalizada que hereda de "*ElementwiseProblem*", ya que esta última proporciona una estructura eficiente para evaluar cada solución individualmente, como se muestra en el listado 4.2.

Listado 4.2. Clase para la evaluación.

CLASE `PlanningComida` extiende `ElementwiseProblem`

```
MÉTODO __init__(self, num_variables, limites_variables,
                  num_objetivos, num_restricciones)
    INICIALIZAR num_variables
    INICIALIZAR limites_variables
    INICIALIZAR num_objetivos
    INICIALIZAR num_restricciones
    LLAMAR al constructor de ElementwiseProblem

MÉTODO evaluate(self, solución, out)
    CALCULAR valores_objetivo
    CALCULAR valores_restricciones
    out["F"] = valores_objetivo
    out["G"] = valores_restricciones
```

El primer método creado es "`__init__`", que inicializa los parámetros del problema, incluyendo el tamaño de la solución (77 genes), el número de objetivos y restricciones (3 cada uno) y los límites del problema (el tamaño de la base de datos).

El segundo método es "`evaluate`", que es donde se define cómo se evalúan las soluciones. Este método recibe una solución individual y calcula su aptitud a partir de los objetivos y restricciones definidos en el apartado [4.4](#).

Se recorre la cadena y, por cada gen, se extraen las calorías, el grupo y los macronutrientes. Se comprueba el objetivo de la preferencia comparando el grupo del alimento con los grupos de preferencia del usuario y, en caso de ser necesario, se suma una penalización a una variable que contará todas las sanciones totales de la solución respecto a este objetivo. El mismo procedimiento se hace con la restricción de la alergia.

Por cada 11 genes de la cadena (que representa un día) se comprueba el objetivo calórico, donde la desviación de kilocalorías diarias respecto al objetivo marcado para el usuario se suma a un contador que refleja la desviación semanal total. Este es el valor a minimizar. Pero, además, existe otro contador que suma todas las penalizaciones de la restricción calórica. Cada vez que las kilocalorías diarias se sale de los límites del 10% se agrega una penalización a este contador. Este es otro valor que se busca minimizar.

El cálculo de los valores a minimizar del objetivo y restricción de macronutrientes sigue el mismo proceso que el de las calorías.

Los resultados de las evaluaciones se almacenan en los diccionarios *"out[F]"* y *"out[G]"*. *"out[F]"* contiene los valores de las funciones objetivos, que intentamos optimizar. *"out[G]"*, muy usado en métodos separatistas para el manejo de restricciones, contiene los valores de las restricciones que deben ser satisfechas para que las soluciones se consideren factibles. Si se usase, por ejemplo, métodos de penalización para las restricciones, no sería necesario incluir *"out[G]"*, ya que las restricciones se tratan como funciones objetivo.

Tras la creación de la clase, hay que crear una función que recoja los resultados de esta clase para incluirlos en el algoritmo evolutivo. El funcionamiento del algoritmo es, en esencia, el mismo que se ha explicado en la sección 3, aunque se puede modificar o añadir algún operador según el tipo de algoritmo usado, razón de la experimentación con distintos algoritmos del apartado 5.2. Por lo tanto, lo primero a seleccionar es el algoritmo con el que se va a trabajar, ya sea NSGA-II, MOEA/D, etc.

Se selecciona el número de individuos por generación y el número de generaciones que el algoritmo evaluará. Como no se ha puesto ninguna condición de parada específica el algoritmo terminará tras haber evaluado todas las generaciones predefinidas.

Lo primero que el algoritmo hace es crear la matriz en la que se aloja toda la población, que tiene un tamaño del número de genes (o variables) por solución multiplicado por el número de individuos de una generación. Por ejemplo, si el menú se compone de 77 genes y se selecciona un tamaño de población de 100 individuos, se crea una matriz de $100 \text{ individuos} \times 77 \text{ genes}$, en la que cada solución candidata se evalúa individualmente.

Pymoo dispone de varios métodos de inicialización, entre los que destaca *"IntegerRandomSampling"*, usado para la generación de matrices de enteros. Este se puede usar en el planificador de comidas al seleccionar el índice del alimento que se añade al menú, el cual es un número entero. Pero se ha decidido crear una versión modificada del mismo para que genere soluciones que sean interesantes en el contexto del proyecto. En vez de seleccionar un índice aleatorio de un alimento de la base de datos, selecciona basándose en las características propias de la solución, explicadas en el apartado 4.3. Es decir, se ha configurado para que se seleccione una bebida cuando el gen sea el tercero de una comida principal o para que los alimentos del desayuno sean de la categoría *"Cereales"* (A), por ejemplo. Así se consigue que la primera población de la generación cumpla las particularidades del problema, como se observa en la figura 4.4.

$$A_{100 \times 77} = \begin{pmatrix} \begin{matrix} \text{Cereal} & \text{Cereal} & \text{Leche} & \text{Snack} \\ \downarrow & \downarrow & \downarrow & \downarrow \end{matrix} & \begin{matrix} a_{00} & a_{01} & a_{02} & a_{03} & \dots & a_{0n} \\ a_{10} & a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{20} & a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{30} & a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ & & \cdot & & \cdot & \\ & & \cdot & & \cdot & \\ & & \cdot & & \cdot & \\ a_{990} & a_{991} & a_{992} & a_{993} & \dots & a_{99n} \end{matrix} \end{pmatrix}$$

Figura 4.4. Matriz de inicialización.

Cada solución de esta población es evaluada según lo explicado al comienzo de este apartado. Si no ha llegado al número total de generaciones, el algoritmo se prepara para generar una nueva población a partir de los operadores.

En la selección se eligen individuos de la población basado en la aptitud. En este caso concreto los valores más bajos presentarán una mayor aptitud, al ser funciones de minimización. Se va a usar la selección por torneo, predeterminada por *Pymoo* para este tipo de problemas.

Tras elegir los individuos, se cruzan para la generación de una nueva población. Se puede variar entre el cruce de un punto y el de dos puntos, asignándoles a cada uno de ellos una probabilidad de cruce P_c . Aparte de los mecanismos elitistas que usen los algoritmos multi-objetivo, no se ha realizado ningún método extra de selección ambiental, por lo que se realiza un reemplazo generacional completo.

Para la mutación ocurre como con la inicialización. En vez de elegir un método de los que ofrece *Pymoo*, se ha decidido crear un método personalizado. El nuevo método recorre cada individuo creado en el cruce y decide si el gen es reservado para una bebida, para un snack, etc. Con una probabilidad P_m a determinar, muta una bebida por otra bebida o un alimento de una categoría determinada por otro de la misma categoría, como se muestra en la figura 4.5. Esto logra, junto con la inicialización, que todas las soluciones a lo largo de las generaciones cumplan con las peculiaridades de este problema.

MUTACIÓN PERSONALIZADA



Figura 4.5. Mutación personalizada.

Por último, esta nueva población resultante vuelve a ser evaluada en busca de ser óptima para el problema.

El algoritmo evolutivo se ejecuta utilizando la función *"minimize"* de *Pymoo*, que toma el problema y la configuración del algoritmo. Como se muestra en la tabla 4.1, el algoritmo va avanzando en busca de mejores soluciones.

En la tabla se aprecia, en orden: *"n_gen"* muestra el número de generación, *"n_eval"* es el contador de evaluaciones de individuos, *"n_nds"* es el número de soluciones no dominadas encontradas, *"cv_min"* es el valor mínimo de violación de una población, *"n_avg"* es el valor promedio de violación de una población, *"eps"* es una métrica que mide la convergencia del algoritmo hacia soluciones óptimas e *"indicator"* se usa para clasificar la calidad del frente de Pareto en cada generación.

n_gen	n_eval	n_nds	cv_min	cv_avg	eps	indicator
1	100	1	2.415619E+03	5.717019E+03	-	-
2	200	1	1.789955E+03	4.574975E+03	-	-
3	300	1	1.290943E+03	3.787494E+03	-	-
4	400	1	1.195388E+03	3.107680E+03	-	-
5	500	1	1.070258E+03	2.478680E+03	-	-
6	600	1	0.000000E+00	1.867363E+03	1.0000000000	ideal
7	700	2	0.000000E+00	1.400831E+03	0.5000000000	ideal
8	800	3	0.000000E+00	9.870077E+02	0.2500000000	f
9	900	4	0.000000E+00	6.756844E+02	0.1250000000	ideal
10	1000	5	0.000000E+00	4.510310E+02	0.0625000000	ideal

Tabla 4.1. Resultados del algoritmo genético.

El algoritmo, con este manejo de restricciones, está configurado para que, hasta que no alcance una solución viable (*"cv_min"*=0), no empieza a evaluar la convergencia y la calidad del frente de Pareto.

En el caso de que, como se ha explicado antes, no fuera necesario *"out[G]"* por el manejo de las restricciones, simplemente desaparecen *"cv_min"* y *"cv_avg"*, y *"n_nds"*, *"eps"* e *"indicator"* se calculan desde la primera generación.

Todas las métricas que obtiene el algoritmo son guardadas en una variable *"resultado"*, a la que se puede acceder.

- En *"resultado.X"* se encuentra una matriz que contiene las soluciones no dominadas encontradas. En este proyecto son cada uno de los posibles menús factibles que el algoritmo ha encontrado. Siguiendo el ejemplo 4.1, en 10 generaciones el algoritmo ha encontrado 5 soluciones no dominadas, por lo que la matriz tiene un tamaño de $5 n_nds \times 77 genes$.
- En *"resultado.F"* se encuentra una matriz que contiene los valores de las funciones objetivo para cada solución no nominada encontrada. Como el algoritmo presenta 3 objetivos, la matriz puede tener un tamaño de $5 n_nds \times 3 objetivos$.
- En *"resultado.G"* se encuentra una matriz muy parecida a *"resultado.F"*, pero con las restricciones en vez de los objetivos. Esta matriz se encuentra vacía si los objetivos y las restricciones no se tratan de manera separada.

4.5.1. Manejo de restricciones

Penalización estática

En este método se penalizan las soluciones que violan las restricciones. En las funciones en las que se definen las restricciones de calorías y macronutrientes, se calcula una penalización proporcional a la diferencia entre el objetivo y los nutrientes consumidos, multiplicada por una constante de penalización, mostradas en el listado 4.3. Poniendo de ejemplo la restricción de calorías, si el objetivo calórico es 2500 kilocalorías y la diferencia con las kilocalorías ingeridas está fuera del 10 %, se multiplica esa diferencia por un factor de penalización. Por lo tanto, cuanto mayor es la diferencia, mayor es la penalización, lo que ayuda a guiar al algoritmo en busca de soluciones que se penalicen menos y, en última instancia, que se acerquen lo máximo posible al objetivo.

En el caso de la restricción de alergia, se penaliza con un factor de penalización si el grupo alérgico aparece en el menú. Si no aparece, se devuelve 0. Tras realizar diversos tests, se ha llegado a la conclusión de que el algoritmo encuentra mejores resultados si el factor de penalización se eleva al cuadrado, lo que hará inviable buscar soluciones que incumplan esta restricción.

En el objetivo de preferencias, que se puede considerar una restricción débil, se penaliza con un factor de penalización (de menor valor que el de la alergia) en el caso de que aparezca en el menú un grupo de alimentos que disguste al usuario. En cambio, si el alimento es del gusto del individuo, se devuelve ese mismo factor con un signo negativo delante, lo que ayuda a que baje el valor a minimizar.

Los valores retornados de las funciones de restricción se suman entre sí. El valor resultante se suma a cada uno de los valores objetivo del problema. El algoritmo pasa de calcular 3 objetivos y 3 restricciones a calcular solo los 3 objetivos a los que se le ha sumado la penalización de las restricciones.

Listado 4.3. Factores de penalización.

PENALIZACION_CALORIAS	=	50
PENALIZACION_MACRONUTRIENTES	=	30
PENALIZACION_PREFERENCIA	=	10
PENALIZACION_ALERGIA	=	100

Método separatista

En el método separatista los objetivos y las restricciones se tratan por separado. A diferencia de los métodos de penalización, en las funciones de restricción no se hace uso de los factores. Se devuelve únicamente, en el caso de las restricciones de las calorías y los macronutrientes, la diferencia entre el valor objetivo y el valor obtenido de la solución candidata.

La función de la restricción de la alergia (y la de las preferencias) se trata igual que en los métodos con penalización, por lo que aquí sí que se usa el factor de penalización correspondiente.

En "out[F]" se introducen los valores de las funciones de los objetivos (3 en total), y en "out[G]" se introducen los valores de las funciones de las restricciones (3). El algoritmo busca primero soluciones que no violen las restricciones y, cuando las encuentra, busca soluciones que optimicen el problema. En la tabla 4.1 se puede ver este funcionamiento.

Restricciones como objetivo

Si bien en los anteriores casos se desarrollaron versiones personalizadas de los métodos, en este tercer caso se va a hacer uso de la herramienta de *Pymoo* "*ConstraintsAsObjective*" [54].

Este método se caracteriza por incorporar las restricciones como objetivos. La finalidad no es solo encontrar soluciones que cumplan con todas las restricciones, sino también evaluar cuánto se puede mejorar la optimización si se relajan las restricciones. La construcción del método de evaluación se realiza igual que en el método separatista. El cambio se lleva a cabo en la función de optimización, donde se indica que en el algoritmo a minimizar se empleará "*ConstraintsAsObjective*".

4.6. Interfaz gráfica

Si bien el objetivo principal de este PFG es la construcción del algoritmo evolutivo y la experimentación del mismo (apartado 1.3), se ha creado una interfaz simple que sirve de guía para la ejecución del algoritmo. Se ha hecho uso de la biblioteca "*tkinter*" para ello [55].

La ventana principal, que es la que se muestra en la figura 4.6, es el punto de entrada para la ejecución del algoritmo. Se ha nombrado "*Planificación nutricional mediante algoritmos evolutivos*". Dispone de un título, que comparte con el nombre de la ventana, y dos botones. El botón de la izquierda, "*Planificar el menú*", lleva a la ventana que se encuentra en la figura 4.8, que es donde se le pregunta al usuario para poder calcular sus kilocalorías necesarias y sus preferencias alimenticias. El segundo botón, "*Visualizar la base de datos*", dirige a la ventana "*Base de datos*", mostrada en la figura 4.7. Aquí se indica cada alimento con sus grupo y sus valores nutricionales.

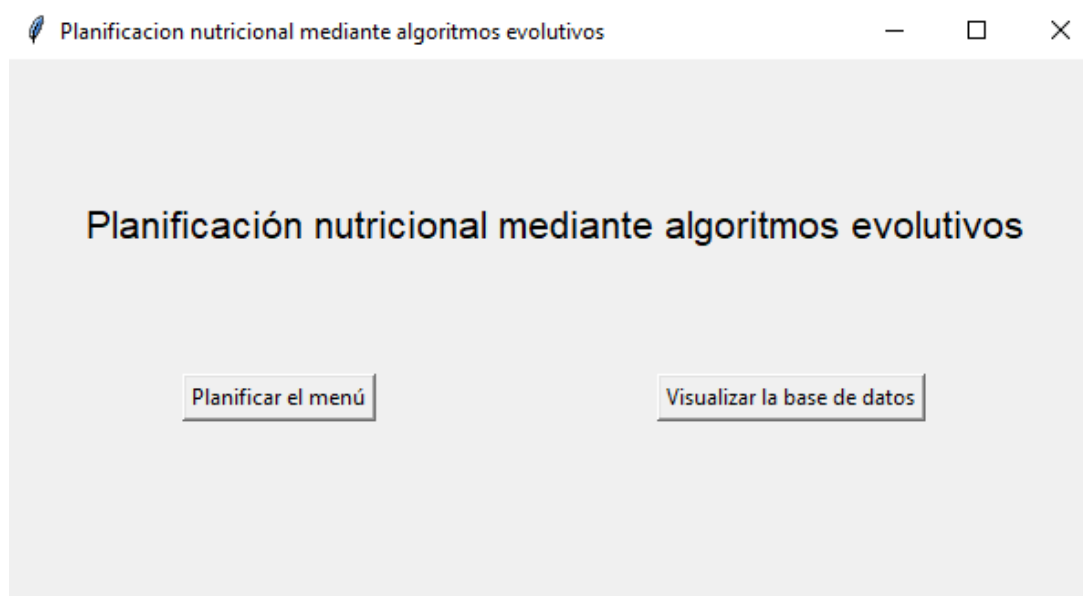


Figura 4.6. Ventana principal.

En la ventana "Base de datos", figura 4.7, se ha creado un "TreeView" para la representación de los datos. Es una herramienta que permite visualizar y gestionar datos en una estructura de árbol. En este caso tiene una forma de tabla y muestra una lista completa de alimentos, cada uno con su categoría, sus calorías y sus macronutrientes. Además, asociado a este "TreeView", se ha creado un *scrollbar* vertical para poder navegar entre los numerosos alimentos.

Se ha incluido un botón "Volver" que redirige a la ventana principal "Planificación nutricional mediante algoritmos evolutivos".

The screenshot shows a window titled "Base de Datos". It contains a table with the following data:

Nombre	Grupo	Calorias (kcal)	Grasas (g)	Proteinas (g)	Carbohidratos (g)
Porridge oats, unfortified	A	381	8.1	10.9	70.7
Porridge oats, unfortified, cooked	A	84	2.3	4.6	12.1
Pizza, cheese and tomato, retail	A	272	9.8	12.2	36.1
Sandwich, white bread, bacon, 1	AB	230	11.9	8.2	24.1
Sandwich, white bread, cheddar	AB	282	14.4	12.0	27.7
Sandwich, white bread, chicken	AB	172	4.9	10.7	22.5
Sandwich, white bread, ham sal	AB	163	4.1	8.2	25.0
Sandwich, white bread, egg may	AB	243	11.2	8.8	28.5
Sandwich, white bread, tuna may	AB	237	10.2	12.5	25.3
Rice, egg fried, takeaway	AC	186	4.9	4.3	33.3

At the bottom of the window, there is a button labeled "Volver".

Figura 4.7. Ventana de la base de datos.

La ventana *"Preguntas al usuario"*, figura 4.8, es usada para calcular las kilocalorías que el usuario necesita y sus grupos alimenticios de preferencia.

Se pregunta al usuario por medio de un *"label"* y este responde en un *"entry"*, que almacena la respuesta. Así ocurre con las preguntas sobre el peso, la edad y la altura. Para el sexo y el nivel de actividad se ha hecho uso de un *"combobox"*, que permite introducir varias opciones posibles dentro de una lista. Para la alergia y las preferencias alimenticias se ha utilizado *"listbox"*, que permite mostrar, bien tabulados, todos los grupos de alimentos. Se puede hacer selección múltiple. Estos grupos quedan guardados para luego usarlos durante la ejecución del algoritmo.

El botón *"Calcular calorías"* activa el cálculo de las kilocalorías objetivo del usuario y las muestra. El botón *"Mostrar menú"* activa la ejecución del algoritmo para que encuentre una solución óptima. El último botón, *"Volver"*, regresa a la ventana principal.

Preguntas al usuario

Introduce tu peso en kg: 75

Introduce tu altura en cm: 175

Introduce tu edad: 23

Introduce tu sexo: hombre

Elige tu nivel de actividad: Moderadamente activo (ejercicio moderado/deportes 3-5 días a la semana)

Grupos de comida a los que eres alérgico:

- MCO Turkey
- ME Game
- MEA Hare
- MEC Rabbit
- MEE Venison
- MG Offal
- MBG Burgers and grillsteaks
- MI Meat products
- MIG Other meat products
- MR Meat dishes

Grupos de comida que te gustan:

- A Cereals and cereal products
- AB Sandwiches
- AC Rice
- AD Pasta
- AE Pizzas
- AF Breads
- AG Rolls
- AI Breakfast cereals
- AK Infant cereal foods
- AM Biscuits

Grupos de comida que no te gustan:

- FA Fruit, general
- FC Fruit juices
- G Nuts and seeds
- GA Nuts and seeds, general
- J Fish and fish products
- JA White fish
- JC Fatty fish
- JK Crustacea
- JM Molluscs
- JR Fish products and dishes

Calcular Calorías

Mostrar Menú

Calorías diarias: 2687,31

Volver

Figura 4.8. Ventana de las preguntas al usuario.

Tras pulsar el botón "Mostrar menú", el algoritmo se ejecuta con los datos introducidos. Cuando termina, se selecciona la primera de las soluciones no dominadas y se obtiene un menú compuesto por 77 alimentos.

Se ha creado la ventana "Menú Generado", mostrada en la figura 4.9, donde se ha hecho uso de un "grid", muy útil para la creación de tablas estáticas. Se han introducido 5 filas (comidas diarias) por 7 columnas (días de la semana). Se muestran todos los alimentos que componen la solución elegida, además del grupo al que pertenecen. Se han añadido una fila y una columna como cabeceras, correspondientes a las comidas del día y a los días de la semana, respectivamente. Además, se ha incorporado una fila al final para ayudar a entender si la solución escogida cumple con los objetivos de las kilocalorías y los macronutrientes.

Se han añadido dos botones, 'Atrás' y 'Cerrar'. El primero vuelve a la ventana "Preguntas al usuario" para la posibilidad de generar otro menú distinto. El segundo botón termina con la ejecución del programa.

Menú Generado

	Lunes	Martes	Miercoles	Jueves	Viernes	Sabado	Domingo
Desayuno	- Bread, brown, toasted (AF) - Eccles cakes, retail (AN) - Milk, Channel islands, whole, summer (BAB)	- Scotch pancakes, retail (AP) - Biscuits, iced (AM) - Milk, whole, pasteurised, winter and spring (BAK)	- Cake, sponge, with dairy cream and jam, frozen (AN) - Sandwich, white bread, egg mayonnaise (AB) - Tomato juice (PE)	- Pie, fruit, wholemeal, one crust, homemade (AS) - Cake, coconut, homemade (AN) - Milk shake, powder, made up with whole milk (BH)	- Pie, fruit, with pie filling, homemade (AS) - Fig Rolls (AM) - Milk shake, powder, made up with semi-skimmed milk (BH)	- Cake bars, chocolate (AN) - Biscuits, semi-sweet (AM) - Milk, skimmed, dried, fortified (BAR)	- Bread, brown, average (AF) - Bread, brown, average (AF) - Milk, skimmed, dried, fortified (BAR)
Tentempie	- Nougat, homemade (SEC)	- Papaya, flesh only, raw (FA)	- Twiglets (SN)	- Durian, flesh only (FA)	- Figs, dried, stewed with sugar (FA)	- Durian, flesh only (FA)	- Toffees, mixed (SEC)
Almuerzo	- Beans, aduki, whole, dried, raw (DB) - Plums, Victoria, stewed without sugar (FA) - Apple juice concentrate, unsweetened, commercial (PE)	- Lamb, stewing, raw, lean and fat, weighed with bone (MAE) - Chicken, breast, casserole, meat and skin, weighed with bone (MCA) - Complian powder, original and sweet (PAA)	- Macaroon, homemade (CDH) - Pork, belly joint/slices, raw, lean and fat (MAG) - Tea, infusion, average, with semi-skimmed milk (PAC)	- Gooseberries, cooking, stewed with sugar (FA) - Turkey, dark meat, raw (MCO) - Complian powder, savoury (PAA)	- Halibut, flesh only, poached (JA) - Chicken, drumsticks, roasted, meat and skin (MCA) - Instant drinks powder, malted, unfortified (PAA)	- Pork, loin steaks, stewed, lean and fat (MAG) - Cheese, cottage, plain, reduced fat (BL) - Complian powder, savoury (PAA)	- Sprats, fried, weighed with bones (JC) - Roe, cod, hard, raw (JR) - Apple juice concentrate, unsweetened, commercial (PE)
Merienda	- Sultanas (FA)	- Apricots, dried (FA)	- Sweets, chew sweets (SEC)	- Chocolate, dark, with cream or mint fondant centres (SEA)	- Truffles, mocha, homemade (SEA)	- Honey (SC)	- Kiwi fruit, flesh only, raw, weighed with skin (FA)
Cena	- Turkey, self-basting, light meat, roasted (MCO) - Peanuts, dry roasted (GA) - Tomato juice (PE)	- Rice, white, basmati, raw (AC) - Salmon, farmed, flesh only, raw (J) - Build-up powder, soup, assorted flavours (PAA)	- Burger, hamburger, takeaway (MBG) - Lobster, boiled (JK) - Build up, powder, shake, assorted flavours (PAA)	- Gelatine (WY) - Taro, baked (DG) - Milk shake, powder (PAA)	- Pollock, Alaskan, flesh only, baked (JA) - Tripe, dressed, stewed in milk (MG) - Milk shake, powder (PAA)	- Burger, Quarter Pounder with cheese, takeaway (MBG) - Patra leaves, raw (DG) - Coffee, cappuccino, latte (P)	- Cho cho fritters, fried in rapeseed oil, homemade (DR) - Lamb, shoulder, whole, roasted, lean and fat (MAE) - Complian powder, savoury (PAA)
Objetivo calórico: 2687.31	Calorías: 2688 Proteínas: 15.14% Carbohidratos: 55.39% Grasas: 29.47%	Calorías: 2686 Proteínas: 16.83% Carbohidratos: 55.21% Grasas: 27.96%	Calorías: 2688 Proteínas: 16.22% Carbohidratos: 55.07% Grasas: 28.72%	Calorías: 2711 Proteínas: 20.43% Carbohidratos: 56.74% Grasas: 22.83%	Calorías: 2680 Proteínas: 17.18% Carbohidratos: 56.03% Grasas: 26.79%	Calorías: 2694 Proteínas: 18.76% Carbohidratos: 55.07% Grasas: 26.18%	Calorías: 2686 Proteínas: 19.79% Carbohidratos: 49.45% Grasas: 30.76%

Atrás Cerrar

Figura 4.9. Ventana del menú generado.

5.

Experimentación

Se han creado 5 sujetos de prueba con diferentes características para la experimentación. Dado que los algoritmos son estocásticos, se han seleccionado 31 semillas distintas para asegurar la reproducibilidad de las ejecuciones. Por lo tanto, se han realizado 155 tests por cada configuración. En el anexo B se encuentran detallados todos los sujetos de prueba, los *seeds* y el entorno usado para la ejecución de las pruebas.

Lo primero que se ha realizado ha sido un análisis de sensibilidad. Se ha variado entre distintos valores para cada hiperparámetro (tipos de operadores, probabilidad, tamaño de la población, ...) y se han comparado para poder encontrar la mejor configuración posible.

Para comparar conjuntos de resultados se han usado dos métodos. El primero, *success rate*, mide la viabilidad de cada configuración, evaluando el número de soluciones factibles que propone. La mejor configuración es la que mayor tasa de éxito tiene.

El otro método para poder hacer comparaciones entre las configuraciones de un problema multiobjetivo es calculando el *hipervolumen*, que es una medida que evalúa la calidad de un conjunto de soluciones. Mide el volumen de la región del espacio que está dominada por las soluciones del conjunto y una referencia externa. Cuanto mayor sea, mejor será la calidad de las soluciones y la diversidad. [56]. Solo se han calculado los hipervolumenes de las soluciones factibles, eliminando las que no cumplen las restricciones.

Para comparar hipervolumenes se hace uso de un test de significatividad estadística. Se trata de una prueba estadística propone una hipótesis nula (H_0), es decir, que ambos conjuntos provienen de la misma distribución, ya que no existe una diferencia significativa entre las muestras. El test devuelve un "*p_value*" que señala qué tan probable es H_0 . Si el valor p es inferior a un umbral (α), que suele ser 0.05, se puede rechazar H_0 . Esto quiere decir que hay una probabilidad menor del 5% de que dos muestras vengan de la misma distribución.

Rechazar H_0 propone evidencia en contra de la hipótesis y que, efectivamente, existe una diferencia significativa entre los conjuntos. Sin embargo, no rechazar H_0 deja en un estado de incertidumbre y solo indica que no se tiene suficiente evidencia para descartarla.

Según el número de conjuntos a comparar se pueden usar diversos tests no paramétricos. En este caso, este trabajo se ha centrado en dos.

Para la comparación de dos conjuntos de resultados se usa el *test de rangos con signos de Wilcoxon*. El proceso calcula las diferencias entre pares (en total 31), ordena los resultados absolutos y les asigna un rango. Después suma los rangos de las diferencias positivas y negativas por separado. La prueba se queda con el valor menor. Si el valor es menor que un valor crítico, obtenido de una tabla de Wilcoxon, se rechaza la hipótesis nula. [57]

Para la comparación de n conjuntos se usa el *test de rangos alineados de Friedman*. Dentro de cada conjunto se ordenan los valores de menor a mayor, dándoles a cada uno un rango de manera ascendente. Se calcula la media de todos los valores con el mismo rango y este valor se resta a cada uno de los números que presenten ese rango. Se suman los rangos que sean iguales entre todos los conjuntos. Es decir, el rango 1 del primer conjunto se suma con el rango 1 del segundo conjunto, y así sucesivamente con todos los rangos. Estas sumas son introducidas en la fórmula de Friedman. El resultado de esta fórmula se compara con un valor crítico de la distribución chi-cuadrado. Al igual que en la de Wilcoxon, si el resultado es menor que el valor crítico, se rechaza H_0 . [58]

El problema de este método es que no determina qué conjunto de resultados es mejor. Si se comparase cada par de conjuntos por separado la probabilidad de error por cada comparativa se iría acumulando, haciendo las siguientes comparativas inviables. Por ello se propone la variación del test de *rangos alineados de Friedman* con la corrección *post-hoc* de *Shaffer*. Permite realizar comparaciones más precisas entre cada par de conjuntos. [59]

Si se rechaza H_0 se hace una comparativa de las medias y las desviaciones típicas de los hipervolumenes para saber que conjunto de resultados es mejor.

Tras haber analizado individualmente cada configuración de hiperparámetros del algoritmo, ya se puede hacer la comparación entre las variaciones del algoritmo propuestas.

5.1. Manejo de restricciones

Todas las tablas de resultados y comparación de hiperparámetros, expuestas en el apéndice C, van a seguir la misma estructura, la cual es la mostrada en la tabla 5.1. Cada fila muestra los resultados para un sujeto en una configuración de hiperparámetros. En este proyecto se ha comparado la probabilidad de mutación y el tipo y probabilidad de cruce. Por cada sujeto se muestra el tiempo en ejecutar un test, siendo este resultado una media de los 31 tests antes mencionados. Las dos siguientes columnas muestran el número de soluciones que se han obtenido en total en todas las ejecuciones y el número de esas soluciones que son factibles, respectivamente. La columna final muestra el *success rate*, dividiendo el número de soluciones factibles entre el número total de soluciones.

Las tablas del método separatista presentan modificaciones, ya que todas las soluciones que muestra este procedimiento son factibles. Por lo tanto, en vez de mostrar el número de soluciones factibles, se muestra el número de ejecuciones que han obtenido soluciones factibles. El *success rate* se calcula dividiendo este nuevo número entre el total de ejecuciones realizadas para un sujeto en una configuración concreta, que es 31. En el anexo, la tabla C.3 muestra la estructura a seguir en estos casos.

Tipo de Mutación	Sujeto	Tiempo Medio de Ejecución (s)	Nº Total de Soluciones	Nº de Soluciones que Cumplen Restricciones	% de Soluciones que Cumplen Restricciones
Baja (1/77)	1	6.11	811	804	99.14 %
	2	6.21	738	721	97.70 %
	3	6.08	872	868	99.54 %
	4	6.11	1092	1092	100.00 %
	5	6.25	1979	1979	100.00 %
Media (0.05)	1	7.24	44	1	2.27 %
	2	7.51	41	8	19.51 %
	3	7.40	73	43	58.90 %
	4	7.40	118	100	84.75 %
	5	7.25	900	900	100.00 %
Alta (0.1)	1	8.84	32	0	0.00 %
	2	8.97	32	0	0.00 %
	3	8.90	35	0	0.00 %
	4	8.91	31	0	0.00 %
	5	8.95	170	169	99.41 %

Tabla 5.1. Resultados en Penalización estática: probabilidad de mutación.

El *success rate* de las tablas de resultados de la probabilidad de mutación indica que en todos los casos el algoritmo propone mejores soluciones cuanto menor sea la probabilidad. La probabilidad baja favorece una convergencia más rápida y propone soluciones muchas veces similares entre sí, pero que cumplen con todas las restricciones.

Al contrario que la mutación, el cruce no altera en gran medida la viabilidad de las soluciones propuestas. A partir de los resultados no se puede determinar cuál es la mejor configuración posible, por lo que es necesario la comparación de los hipervolumenes. Tras realizar el test de Friedman con la corrección post-hoc de Schaffer, el algoritmo con cruce de dos puntos y probabilidad alta es el más efectivo en todos los tipos de manejo de restricciones, lo que significa que la mejor opción es la que mayor recombinación de genes entre padres presenta, ayudando a mejorar la exploración del espacio de soluciones.

La comparación de algoritmos se realiza entre el método separatista y la penalización estática, ya que los resultados que se obtienen con el método "ConstraintsAsObjective" muestra resultados muy malos y soluciones no factibles en más del 95 % de los casos (tablas C.5 y C.6). La prueba de Wilcoxon obtiene un *p_value* de 0.1037, por lo que no hay suficiente evidencia para demostrar que hay una diferencia significativa entre ambos conjuntos de resultados.

En la tabla 5.2 se muestra de manera gráfica las conclusiones que se obtienen tras la comparación de los métodos. Como se ha explicado, la calidad de las soluciones de la penalización estática y del método separatista son prácticamente iguales. En cuanto a la estabilidad, la penalización estática presenta un *success rate* más cercano al 100 % en los sujetos 1, 2 y 3 respecto al método separatista. En cambio, en los tiempos de ejecución destaca el método separatista, que es, de media, 1 segundo más rápido que el otro procedimiento.

Por lo tanto, no es posible determinar cuál es el mejor método de manejo de restricciones con las pruebas realizadas con el algoritmo multiobjetivo NSGA-II. Ambos métodos van a ser usados en las pruebas de comparación de algoritmos para comprender la configuración de parámetros óptima para el proyecto.

/// Esta tabla es un borrador ///

Método de manejo de restricciones	Estabilidad	Calidad	Velocidad
Penalización Estática	▲	≈	≈
Método Separatista	≈	≈	▲
Restricción como Objetivo	▼	▼	▼

Tabla 5.2. Comparación de métodos de manejo de restricciones.

5.2. Algoritmos multi-objetivo

Por los resultados mostrados en la tabla C.7 no es posible saber el mejor método de manejo de restricciones con el algoritmo SPEA2, por lo que se realiza el test de Wilcoxon entre pares. El resultado arroja un p_value de 0.87, por lo que no hay suficiente evidencia para demostrar que son diferentes.

Para MOEA/D se va a comparar diferente cantidad de vecinos, la probabilidad de selección de vecinos y el número de direcciones de referencia con dos métodos muy usados para generarlas, *das-dennis* e *incremental*, explicadas en el punto 3.4.2. Este algoritmo solo se va a probar con el método de penalización estática, ya que no permite restricciones explícitas.

Es interesante ver que todas las soluciones generadas por las distintas configuraciones de la tabla C.8 son factibles, lo que puede demostrar que MOEA/D es un buen algoritmo para nuestro problema. Tras realizar el test estadístico, la mejor configuración es la que presenta un mayor número de vecinos, 30.

La probabilidad de selección entre vecinos tampoco supone un gran cambio en la factibilidad de las soluciones que propone el algoritmo. Tras realizar el test para casos $N \times N$, no hay suficiente evidencia para rechazar H_0 . La media entre calidad, estabilidad y rapidez es más alta en la probabilidad más alta (0.9), por lo que es la que se usa.

Para establecer un número de direcciones con el método de generación *das-dennis*, se compara el *success rate* de la tabla C.10 y se realizan los tests no paramétricos. Aunque la que presenta un número más elevado de direcciones es la configuración con más calidad, se ha decidido seleccionar la configuración con 12 direcciones debido a los altos tiempos que necesita "Alto (18)" en cada ejecución, haciendo inviable su uso en un entorno real. Con los resultados obtenidos tras la generación de direcciones de referencia con *incremental* ocurre lo mismo. el número más alto es el que arroja mejores resultados, pero los altos tiempos de ejecución descartan esa configuración. Tras hacer el test de Wilcoxon entre ambas configuraciones, se concluye que *incremental* presenta mejores soluciones que *das-dennis*.

Con todas las configuraciones elegidas ya se puede realizar la comparación entre algoritmos. Los resultados muestran que, en cuanto a calidad de soluciones propuestas, el algoritmo de NSGA-II con el método separatista es el mejor, seguido por NSGA-II con penalización estática. En tercer y cuarto lugar se encuentra SPEA2 con penalización estática y con manejo separatista, respectivamente. Finalmente, el algoritmo MOEA/D tiene el peor rendimiento según los rankings obtenidos.

El MOEA/D presenta un success rate del 100 %. Los métodos de penalización estática, tanto de NSGA-II como de SPEA2, lo siguen de cerca, con más del 99 % de soluciones factibles propuestas. Los métodos separatistas empeoran un poco respecto al resto, aunque ambos siguen presentando más del 97 % de success rate.

En cuanto a los tiempos, todos los métodos de los algoritmos NSGA-II y SPEA2 no se llevan más de 1 segundo de diferencia, siendo el mejor el método separatista de NSGA-II con 5.64 segundos y el peor la penalización estática del mismo algoritmo con 6.61 segundos. El algoritmo con los peores tiempos es MOEA/D, donde la media sube a 8.76 segundos por ejecución.

/// Esta tabla es un borrador ///

Algoritmo	Estabilidad	Calidad	Velocidad
NSGA-II (Penalización estática)	≈	▲	≈
NSGA-II (Método separatista)	▼	▲	▲
SPEA2 (Penalización estática)	≈	≈	≈
SPEA2 (Método separatista)	▼	≈	≈
MOEA/D	▲	▼	▼

Tabla 5.3. Comparación de algoritmos.

6. Conclusiones y líneas de investigación

6.1. Conclusiones

Este proyecto se ha centrado en la realización de un algoritmo evolutivo capaz de diseñar una planificación semanal de comidas, que permite ajustar las calorías y los nutrientes que el sujeto necesita. Además, puede identificar las alergias y las preferencias alimenticias del usuario y modificar el menú según sea conveniente. Para ello se ha utilizado *Pymoo*, que proporciona una estructura muy útil para la optimización multiobjetivo. Si bien se ha usado un tipo de cruce y de selección diseñados por este framework, también se han creado métodos personalizados para la inicialización y para la mutación, que permiten generar soluciones que sean coherentes con los hábitos alimenticios.

Tras el desarrollo del algoritmo, se han utilizado 5 sujetos para comprobar la eficacia del algoritmo. Se han probado distintas configuraciones, en las que se ha variado la probabilidad de cruce y mutación, el tipo de manejo de restricciones usado o los algoritmos proporcionados por *Pymoo*. Se han documentado los resultados para entender cuáles son los parámetros que más benefician a la hora de encontrar soluciones factibles para el problema.

Tras la construcción del algoritmo y las distintas pruebas realizadas, se consigue diseñar, en una gran mayoría de los casos, un menú alimenticio que cumple con las restricciones de la alergia y de los límites superiores e inferiores de calorías y macronutrientes. Las soluciones presentan una gran aptitud y se acercan mucho a los objetivos marcados. Los resultados demuestran la eficacia del algoritmo en una gran diversidad de usuarios de prueba, además de unos tiempos de ejecución dentro de unos márgenes esperados. La base de datos escogida también ha ayudado, al proporcionar una gran cantidad de alimentos entre los que formar soluciones y de datos nutricionales que facilitan la realización del proyecto.

El trabajo ha servido para conocer más en profundidad la computación evolutiva y los algoritmos genéticos. Ha permitido entender la estructura de un algoritmo genético básico y de otros más avanzados y comprender cómo estas herramientas pueden ser de gran ayuda para mejorar los hábitos alimenticios de los consumidores, proponiendo planes nutricionales acorde a las necesidades de cada usuario.

/// TODO: Comentar la mejor configuración encontrada ///

6.2. Líneas de investigación

Tras realizar este TFG, existen ciertos conceptos en los que no se ha profundizado y que sería interesante desarrollar como ampliaciones futuras.

Para evolucionar el algoritmo y aumentar la calidad nutricional del menú, se pueden realizar varias mejoras. Si bien se han elegido 3 objetivos para simplificar el desarrollo del problema, se pueden incluir más objetivos. Uno de los que se tuvo en cuenta y se descartó es el de la correcta distribución de calorías a lo largo del día, siguiendo las pautas del Ministerio de Sanidad [60]. Su implementación en el algoritmo sería muy similar al objetivo y restricción de macronutrientes.

Otro aspecto a desarrollar es una mayor personalización del algoritmo en base a las preferencias del individuo. Se pueden considerar razones de religión o de ideología, o incluso la edad del usuario a la hora de aumentar o disminuir la ingesta de un alimento. También la posibilidad de elegir un alimento concreto del listado, más allá de elegir todos los que pertenecen a un mismo grupo.

También a destacar la elección de la base de datos. Se ha seleccionado la base de datos del Reino Unido debido a que es una de las más fáciles a la hora de exportar los datos y trabajar con ellos, pero presenta tres problemas. El primero es la mezcla de registros entre alimentos ya cocinados y alimentos crudos. Algunos de estos alimentos crudos se pueden ingerir directamente y otros deben pasar previamente por una cocción, por lo que es muy difícil distinguirlos y eliminar los que no se puedan consumir. El segundo problema son los alimentos y platos que aparecen. Se asemejan mucho más a los hábitos de consumo de un ciudadano inglés que al de un español, cuyos hábitos se basan en la dieta mediterránea. El tercero es el idioma, ya que todas las comidas se presentan en inglés. Tomando de referencia la base de datos "BEDCA" (Base de Datos Española de Composición de Alimentos), la creación de una base de datos propia para el problema y en español ayudaría a mejorar el resultado que propone el algoritmo.

Si bien no está señalado dentro de los objetivos del trabajo, se puede desarrollar el software o aplicación que albergue el algoritmo. Aunque se ha hecho un primer acercamiento con la interfaz gráfica propuesta en el apartado 4.6, se puede mejorar su estética y funcionalidad, haciéndola más intuitiva para el usuario.

Por último, destacar las variaciones del algoritmo propuestas para la comparación. Siguiendo la explicación de la sección 3, se pueden añadir más algoritmos multiobjetivo o diferentes manejos de restricciones. Cuantas más opciones se comparen, más optimizado será el resultado que proponga el algoritmo final. Un ejemplo, que no se ha incluido por la falta de eficiencia y el alto tiempo de ejecución, es una función de reparación que busca que todas las restricciones se cumplan antes de evaluar el algoritmo. El código se puede encontrar en el repositorio Github del trabajo [6].

7.

Impacto social y medioambiental

La realización de una planificación nutricional mediante algoritmos evolutivos tiene un impacto social al influir sobre la salud de las personas. Al optimizar la selección de alimentos, el algoritmo obtiene menús equilibrados según las necesidades de los usuarios. Esto puede ayudar a prevenir enfermedades relacionadas con la mala alimentación, como la obesidad o las enfermedades cardiovasculares. Al ayudar a promover una mejor alimentación, este trabajo se alinea con el *Objetivo de Desarrollo Sostenible* (ODS) 3, llamado "*Salud y bienestar*", que busca garantizar una vida sana y promover el bienestar para todos en todas las edades.

En cuanto al impacto medioambiental, el conocer la planificación semanal y los alimentos a ingerir ayuda a lograr el ODS 12, "*Producción y consumo responsables*", que busca asegurar patrones de consumo y producción sostenibles. Es decir, es posible reducir el desperdicio de alimentos y minimizar la huella ecológica que los residuos producen. Pero también, en un futuro se puede personalizar más el algoritmo y la base de datos, como se comenta en la sección [6.2](#), para que, por ejemplo, se puedan elegir los alimentos locales y de temporada, que generan menos contaminación.

Por último, relacionarlo con el ODS 15, "*Vida de ecosistemas terrestres*", que tiene como objetivo proteger, restaurar y promover el uso sostenible de los ecosistemas terrestres. Al igual que en el ejemplo anterior, se pueden buscar soluciones que tengan un mínimo impacto ambiental. Fomentar a través del algoritmo prácticas amigables y respetuosas con el medio ambiente, como por ejemplo la ganadería extensiva o la pesca sostenible.

A. Apéndice: Grupos de comida

Cereales y productos de cereales (Cereals and cereal products)

	A
Sándwiches (Sandwiches)	AB
Arroz (Rice)	AC
Pasta (Pasta)	AD
Pizzas (Pizzas)	AE
Panes (Breads)	AF
Panecillos (Rolls)	AG
Cereales de desayuno (Breakfast cereals)	AI
Alimentos infantiles de cereales (Infant cereal foods)	AK
Galletas (Biscuits)	AM
Pasteles (Cakes)	AN
Pastelería (Pastry)	AO
Bollos y pasteles (Buns and pastries)	AP
Pudines (Puddings)	AS
Aperitivos (Savouries)	AT

Leche y productos lácteos (Milk and milk products)

	B
Leche de vaca (Cow's milk)	BA
Leche para desayuno (Breakfast milk)	BAB
Leche desnatada (Skimmed milk)	BAE
Leche semi-desnatada (Semi-skimmed milk)	BAH
Leche entera (Whole milk)	BAK
Leches procesadas (Processed milks)	BAR
Bebidas a base de leche (Milk-based drinks)	BH
Cremas (Creams)	BJ
Cremas frescas (pasteurizadas) (Fresh creams (pasteurised))	BJC
Cremas UHT (UHT creams)	BJP
Cremas imitadas (Imitation creams)	BJS
Quesos (Cheeses)	BL
Yogures (Yogurts)	BN
Yogures de leche entera (Whole milk yogurts)	BNE
Yogures bajos en grasa (Low fat yogurts)	BNH
Otros yogures (Other yogurts)	BNS
Helados (Ice creams)	BP
Pudines y postres refrigerados (Puddings and chilled desserts)	BR
Platos salados y salsas (Savoury dishes and sauces)	BV

Huevos (Eggs)**C**

Huevos (Eggs)	CA
Platos de huevo (Egg dishes)	CD
Platos de huevo salados (Savoury egg dishes)	CDE
Platos de huevo dulces (Sweet egg dishes)	CDH

Verduras (Vegetables)**D**

Patatas (Potatoes)	DA
Patatas tempranas (Early potatoes)	DAE
Patatas de cosecha principal (Main crop potatoes)	DAM
patatas viejas picadas (Chipped old potatoes)	DAP
Productos de patata (Potato products)	DAR
Judías y lentejas (Beans and lentils)	DB
Guisantes (Peas)	DF
Verduras, en general (Vegetables, general)	DG
Verduras, secas (Vegetables, dried)	DI
Platos de verduras (Vegetable dishes)	DR

Fruta (Fruit)**F**

Fruta, en general (Fruit, general)	FA
Zumos de fruta (Fruit juices)	FC
Zumos (Juices)	FE

Frutos secos y semillas (Nuts and seeds)**G**

Frutos secos y semillas, en general (Nuts and seeds, general)	GA
---	----

Pescado y productos de pescado (Fish and fish products)**J**

Pescado blanco (White fish)	JA
Pescado graso (Fatty fish)	JC
Crustáceos (Crustacea)	JK
Moluscos (Molluscs)	JM
Productos y platos de pescado (Fish products and dishes)	JR

Carne y productos cárnicos (Meat and meat products)**M**

Carne (Meat)	MA
Bacon (Bacon)	MAA
Vaca (Beef)	MAC
Cordero (Lamb)	MAE
Cerdo (Pork)	MAG
Ternera (Veal)	MAI
Aves (Poultry)	MC
Pollo (Chicken)	MCA
Pato (Duck)	MCC
Ganso (Goose)	MCE
Urogallo (Grouse)	MCG
Perdiz (Partridge)	MCI

Faisán (Pheasant)	MCK
Paloma (Pigeon)	MCM
Pavo (Turkey)	MCO
Caza (Game)	ME
Liebre (Hare)	MEA
Conejo (Rabbit)	MEC
Venado (Venison)	MEE
Despojos (Offal)	MG
Hamburguesas y filetes para parrilla (Burgers and grillsteaks)	MB
Productos cárnicos (Meat products)	MI
Otros productos cárnicos (Other meat products)	MIG
Platos de carne (Meat dishes)	MR
Bebidas (Beverages)	P
Bebidas en polvo, esencias e infusiones (Powdered drinks, essences and infusions)	PA
Bebidas en polvo y esencias (Powdered drinks and essences)	PAA
Infusiones (Infusions)	PAC
Bebidas sin alcohol (Soft drinks)	PC
Bebidas carbonatadas (Carbonated drinks)	PCA
Concentrados y cordiales (Squash and cordials)	PCC
Bebidas alcohólicas (Alcoholic beverages)	Q
Cervezas (Beers)	QA
Sidras (Ciders)	QC
Vinos (Wines)	QE
Azúcares, conservas y snacks (Sugars, preserves and snacks)	S
Confitería (Confectionery)	SE
Confitería de chocolate (Chocolate confectionery)	SEA
Confitería sin chocolate (Non-chocolate confectionery)	SEC
Snacks salados (Savoury snacks)	SN
Snacks con base de patata (Potato-based snacks)	SNA
Snacks sin patata (Non-potato snacks)	SNC
Sopas, salsas y alimentos varios (Soups, sauces and miscellaneous foods)	W
Sopas (Soups)	WA
Sopas caseras (Homemade soups)	WAA
Sopas enlatadas (Canned soups)	WAC
Sopas de paquete (Packet soups)	WAE
Encurtidos y chutneys (Pickles and chutneys)	WE

B. Apéndice: Entorno y sujetos

Entorno de pruebas

- Editor de código fuente: Microsoft Visual Studio Code
- Sistema Operativo: Microsoft Windows 10
- Procesador: AMD Ryzen 5 5600
- Memoria Ram: Vengeance LPX 16GB (2x8GB) DRAM DDR4 3000MHz
- SSD: GIGABYTE M.2 PCIe SSD 256GB
- Tarjeta gráfica: ZOTAC GeForce RTX 2060

Seeds utilizados

6	26	59	60	
79	93	489	608	
634	684	784	930	
3364	3608	4845	5375	
8542	9397	11320	14648	
39843	73886	83186	97870	
152822	211193	226880	315529	
428397	489166	526568		

Tabla B.1. Seeds utilizados.

Sujetos de prueba

Campo	Descripción
Sujeto 1	78 kg, 185 cm, 17 años, Masculino, Moderadamente activo, 2877.19 kcal
Gustos	Arroz (AC), Pasta (AD), Pollo (MCA)
Disgustos	Pescado (J), Postres (BR)
Alergias	Azúcares (S)
Sujeto 2	60 kg, 170 cm, 30 años, Femenino, Muy activo, 2567.85 kcal
Gustos	Leche descremada (BAE), Jugos de frutas (FC), Zumos (FE)
Disgustos	Huevo (C)
Alergias	Cereales (A)
Sujeto 3	90 kg, 175 cm, 40 años, Masculino, Activo, 3102.84 kcal
Gustos	Carne de res (MAC), Patatas fritas (DAP), Confitura de chocolate (SEA)
Disgustos	Bebidas lácteas (BH), Imitación de cremas (BJS), Otros productos cárnicos (MIG)
Alergias	Infusiones (PAC), Bebidas carbonatadas (PCA), Aperitivos no de patata (SNC)
Sujeto 4	68 kg, 160 cm, 55 años, Femenino, Poco Activo, 1710.50 kcal
Gustos	Panes (AF), Yogures desnatados (BNH)
Disgustos	Hamburguesas (MB), Cervezas (QA), Sidras (QC)
Alergias	Fruta (F)
Sujeto 5	72 kg, 155 cm, 72 años, Femenino, Sedentario, 1401.30 kcal
Gustos	Galletas (AM), Pescado graso (JC)
Disgustos	Despojos (MG), Platos de carne (MR), Infusiones (PAC)
Alergias	Leche de vaca (BA), Bebidas con base de leche (BH)

Tabla B.2. Características y preferencias alimenticias de los sujetos

C.

Apéndice: Tablas de resultados

C.1. Manejo de restricciones

C.1.1. Penalización estática

Tipo de Mutación	Sujeto	Tiempo Medio de Ejecución (s)	Nº Total de Soluciones	Nº de Soluciones que Cumplen Restricciones	% de Soluciones que Cumplen Restricciones
Baja (1/77)	1	6.11	811	804	99.14 %
	2	6.21	738	721	97.70 %
	3	6.08	872	868	99.54 %
	4	6.11	1092	1092	100.00 %
	5	6.25	1979	1979	100.00 %
Media (0.05)	1	7.24	44	1	2.27 %
	2	7.51	41	8	19.51 %
	3	7.40	73	43	58.90 %
	4	7.40	118	100	84.75 %
	5	7.25	900	900	100.00 %
Alta (0.1)	1	8.84	32	0	0.00 %
	2	8.97	32	0	0.00 %
	3	8.90	35	0	0.00 %
	4	8.91	31	0	0.00 %
	5	8.95	170	169	99.41 %

Tabla C.1. Resultados en Penalización estática: probabilidad de mutación.

Tipo de Cruce	Sujeto	Tiempo Medio de Ejecución (s)	Nº Total de Soluciones	Nº de Soluciones que Cumplen Restricciones	% de Soluciones que Cumplen Restricciones
Un Punto Bajo (0.6)	1	6.57	473	450	95.14 %
	2	6.44	476	456	95.80 %
	3	6.43	703	690	98.15 %
	4	6.47	976	976	100.00 %
	5	6.52	1847	1847	100.00 %
Un Punto Alto (0.9)	1	6.11	811	804	99.14 %
	2	6.21	738	721	97.70 %
	3	6.08	872	868	99.54 %
	4	6.11	1092	1092	100.00 %
	5	6.25	1979	1979	100.00 %
Dos Puntos Bajo (0.6)	1	6.47	852	847	99.41 %
	2	6.55	720	715	99.31 %
	3	6.76	856	856	100.00 %
	4	7.09	1096	1096	100.00 %
	5	6.96	1795	1795	100.00 %
Dos Puntos Alto (0.9)	1	6.71	881	880	99.89 %
	2	6.79	955	928	97.17 %
	3	6.69	1068	1068	100.00 %
	4	6.67	1149	1149	100.00 %
	5	6.18	1751	1751	100.00 %

Tabla C.2. Resultados en Penalización estática: tipo y probabilidad de cruce.

C.1.2. Método separatista

Tipo de Mutación	Sujeto	Tiempo Medio de Ejecución (s)	Nº Total de Soluciones	Nº de Ejecuciones con Soluciones	% de Ejecuciones con Soluciones
Baja (1/77)	1	5.54	640	28	90.32 %
	2	5.43	598	25	80.65 %
	3	5.50	937	29	93.55 %
	4	5.57	1102	31	100.00 %
	5	5.89	1856	31	100.00 %
Media (0.05)	1	6.27	7	4	12.90 %
	2	6.39	1	1	3.23 %
	3	6.35	41	6	19.35 %
	4	6.40	69	11	35.48 %
	5	6.77	960	31	100.00 %
Alta (0.1)	1	8.20	0	0	0.00 %
	2	8.59	0	0	0.00 %
	3	8.14	0	0	0.00 %
	4	8.13	0	0	0.00 %
	5	8.15	194	30	96.77 %

Tabla C.3. Resultados en Método separatista: probabilidad de mutación.

Tipo de Cruce	Sujeto	Tiempo Medio de Ejecución (s)	Nº Total de Soluciones	Nº de Ejecuciones con Soluciones	% de Ejecuciones con Soluciones
Un Punto Bajo (0.6)	1	5.70	539	26	83.87 %
	2	5.83	448	22	70.97 %
	3	5.80	716	30	96.77 %
	4	5.79	903	31	100.00 %
	5	6.07	1884	31	100.00 %
Un Punto Alto (0.9)	1	5.54	640	28	90.32 %
	2	5.43	598	25	80.65 %
	3	5.50	937	29	93.55 %
	4	5.57	1102	31	100.00 %
	5	5.89	1856	31	100.00 %
Dos Puntos Bajo (0.6)	1	5.61	794	28	90.32 %
	2	5.72	642	26	83.87 %
	3	5.73	848	30	96.77 %
	4	5.82	1113	31	100.00 %
	5	6.08	1802	31	100.00 %
Dos Puntos Alto (0.9)	1	5.48	1018	29	93.55 %
	2	5.58	827	29	93.55 %
	3	5.61	962	31	100.00 %
	4	5.64	1226	31	100.00 %
	5	5.91	1693	31	100.00 %

Tabla C.4. Resultados en Método separatista: tipo y probabilidad de cruce.

C.1.3. Restricción como objetivo

Tipo de Mutación	Sujeto	Tiempo Medio de Ejecución (s)	Nº Total de Soluciones	Nº de Soluciones que Cumplen Restricciones	% de Soluciones que Cumplen Restricciones
Baja (1/77)	1	7.12	3100	0	0.00 %
	2	7.01	3100	0	0.00 %
	3	7.01	3090	11	0.36 %
	4	7.03	3100	0	0.00 %
	5	7.08	3092	98	3.17 %
Media (0.05)	1	7.88	3100	0	0.00 %
	2	8.02	3100	0	0.00 %
	3	8.03	3094	0	0.00 %
	4	8.01	3100	0	0.00 %
	5	8.06	3100	59	1.90 %
Alta (0.1)	1	9.26	3100	0	0.00 %
	2	9.40	3100	0	0.00 %
	3	9.44	3080	0	0.00 %
	4	9.43	3100	0	0.00 %
	5	9.24	3079	12	0.39 %

Tabla C.5. Resultados en Restricción como objetivo: probabilidad de mutación.

Tipo de Cruce	Sujeto	Tiempo Medio de Ejecución (s)	Nº Total de Soluciones	Nº de Soluciones que Cumplen Restricciones	% de Soluciones que Cumplen Restricciones
Un Punto Bajo (0.6)	1	7.12	3100	0	0.00 %
	2	7.01	3100	0	0.00 %
	3	7.01	3090	11	0.36 %
	4	7.03	3100	0	0.00 %
	5	7.08	3092	98	3.17 %
Un Punto Alto (0.9)	1	6.69	3100	0	0.00 %
	2	6.87	3100	0	0.00 %
	3	6.87	3051	3	0.10 %
	4	6.92	3100	0	0.00 %
	5	6.98	3090	123	3.98 %
Dos Puntos Bajo (0.6)	1	7.05	3100	0	0.00 %
	2	7.19	3100	0	0.00 %
	3	7.28	3058	8	0.26 %
	4	7.40	3100	0	0.00 %
	5	7.42	3100	71	2.29 %
Dos Puntos Alto (0.9)	1	6.98	3100	0	0.00 %
	2	7.15	3100	0	0.00 %
	3	7.09	2997	12	0.40 %
	4	7.00	3100	0	0.00 %
	5	7.27	3094	118	3.81 %

Tabla C.6. Resultados en Restricción como objetivo: tipo y probabilidad de cruce.

C.2. Algoritmos multi-objetivo

C.2.1. SPEA2

Manejo de Restricciones	Sujeto	Tiempo Medio de Ejecución (s)	Nº Total de Soluciones	Nº de Ejecuciones con Soluciones	% de Ejecuciones con Soluciones
Método separatista	1	5.81	1099	29	93.55 %
	2	5.51	813	29	93.55 %
	3	5.57	1063	31	100.00 %
	4	5.77	1282	31	100.00 %
	5	6.18	1744	31	100.00 %
Manejo de Restricciones	Sujeto	Tiempo Medio de Ejecución (s)	Nº Total de Soluciones	Nº de Soluciones que cumplen Restricciones	% Soluciones que cumplen Restricciones
Penalización estática	1	5.91	1138	1129	99.21 %
	2	5.82	752	735	97.74 %
	3	5.79	1076	1076	100.00 %
	4	5.80	1267	1267	100.00 %
	5	6.02	2094	2094	100.00 %

Tabla C.7. Resultados en SPEA2: manejo de restricciones.

C.2.2. MOEAD

Nº de Vecinos	Sujeto	Tiempo Medio de Ejecución (s)	Nº Total de Soluciones	Nº de Soluciones que Cumplen Restricciones	% de Soluciones que Cumplen Restricciones
Bajo (10)	1	8.22	1343	1343	100.00 %
	2	8.23	1376	1376	100.00 %
	3	8.22	1258	1258	100.00 %
	4	8.21	1181	1181	100.00 %
	5	8.20	1157	1157	100.00 %
Medio (20)	1	7.97	1311	1311	100.00 %
	2	8.06	1592	1592	100.00 %
	3	8.10	1494	1494	100.00 %
	4	8.20	1434	1434	100.00 %
	5	8.11	1441	1441	100.00 %
Alto (30)	1	8.18	1643	1643	100.00 %
	2	8.26	1529	1529	100.00 %
	3	8.21	1536	1536	100.00 %
	4	8.18	1333	1333	100.00 %
	5	8.20	1426	1426	100.00 %

Tabla C.8. Resultados en MOEA/D: número de vecinos.

Probabilidad de Selección de Vecinos	Sujeto	Tiempo Medio de Ejecución (s)	Nº Total de Soluciones	Nº de Soluciones que Cumplen Restricciones	% de Soluciones que Cumplen Restricciones
Baja (0.5)	1	7.97	1556	1556	100.00 %
	2	7.97	1707	1707	100.00 %
	3	7.91	1427	1427	100.00 %
	4	7.94	1372	1372	100.00 %
	5	7.98	1520	1485	97.70 %
Media (0.7)	1	8.18	1643	1643	100.00 %
	2	8.26	1529	1529	100.00 %
	3	8.21	1536	1536	100.00 %
	4	8.18	1333	1333	100.00 %
	5	8.20	1426	1426	100.00 %
Alta (0.9)	1	8.15	1630	1630	100.00 %
	2	8.06	1337	1337	100.00 %
	3	7.98	1565	1565	100.00 %
	4	8.01	1524	1524	100.00 %
	5	8.99	1543	1543	100.00 %

Tabla C.9. Resultados en MOEA/D: probabilidad de selección de vecinos.

N° de Direcciones de Referencia	Sujeto	Tiempo Medio de Ejecución (s)	N° Total de Soluciones	N° de Soluciones que Cumplen Restricciones	% de Soluciones que Cumplen Restricciones
Bajo (5)	1	3.37	520	426	81.92 %
	2	3.40	585	345	58.97 %
	3	3.42	543	403	74.22 %
	4	3.45	581	386	66.44 %
	5	3.49	499	452	90.58 %
Medio (12)	1	8.15	1630	1630	100.00 %
	2	8.06	1337	1337	100.00 %
	3	7.98	1565	1565	100.00 %
	4	8.01	1524	1524	100.00 %
	5	8.99	1543	1543	100.00 %
Alto (18)	1	15.23	3009	3009	100.00 %
	2	14.72	2629	2629	100.00 %
	3	14.73	2782	2782	100.00 %
	4	14.86	2612	2612	100.00 %
	5	14.86	2926	2926	100.00 %

Tabla C.10. Resultados en MOEA/D: direcciones de referencia en Das-Dennis.

N° de Direcciones de Referencia	Sujeto	Tiempo Medio de Ejecución (s)	N° Total de Soluciones	N° de Soluciones que Cumplen Restricciones	% de Soluciones que Cumplen Restricciones
Bajo (5)	1	4.89	925	904	97.73 %
	2	5.02	910	810	89.01 %
	3	4.89	840	840	100.00 %
	4	4.93	825	806	97.70 %
	5	4.96	935	912	97.54 %
Medio (12)	1	8.86	1634	1634	100.00 %
	2	8.93	1708	1708	100.00 %
	3	7.89	1752	1752	100.00 %
	4	8.98	1571	1571	100.00 %
	5	9.15	1837	1837	100.00 %
Alto (18)	1	34.70	5538	5538	100.00 %
	2	34.61	4981	4981	100.00 %
	3	34.35	5596	5596	100.00 %
	4	34.38	4803	4803	100.00 %
	5	34.24	5548	5548	100.00 %

Tabla C.11. Resultados en MOEA/D: direcciones de referencia en Incremental.

Bibliografía

- [1] J. McCarthy, M. Minsky, N. Rochester y C. Shannon, «Dartmouth Summer Research Project on Artificial Intelligence,» Dartmouth College, Hanover, New Hampshire, USA, 1956. dirección: <https://home.dartmouth.edu/about/artificial-intelligence-ai-coined-dartmouth>.
- [2] A. M. Turing, «Computing Machinery and Intelligence,» *Mind*, vol. 59, n.º 236, págs. 433-460, 1950. eprint: <https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf>. dirección: <https://doi.org/10.1093/mind/LIX.236.433>.
- [3] Todoist, *Todoist AI Assistant Integrations*, Último acceso: 23 de mayo de 2024, 2024. dirección: <https://todoist.com/es/integrations/apps/ai-assistant>.
- [4] Google, *Behind the scenes with the new Nest Thermostat*, Último acceso: 23 de mayo de 2024, 2020. dirección: <https://blog.google/products/google-nest/behind-scenes-new-nest-thermostat/>.
- [5] O. N. de Tecnología y Sociedad (ONTSI), *Uso de inteligencia artificial y big data en las empresas españolas*, 2023. dirección: <https://www.ontsi.es/es/publicaciones/uso-de-inteligencia-artificial-y-big-data-en-las-empresas-espanolas>.
- [6] Philips, *Bringing Confident Diagnosis to More Patients at Low Cost: Philips Launches New AI-enabled CT 5300 at ECR 2024*, Último acceso: 23 de mayo de 2024, 2024. dirección: <https://www.philips.com/a-w/about/news/archive/standard/news/press/2024/bringing-confident-diagnosis-to-more-patients-at-low-cost-philips-launches-new-ai-enabled-ct-5300-at-ecr2024.html>.
- [7] Y. Majeed, L. Fu y L. He, «Editorial: Artificial Intelligence-of-Things (AIoT) in Precision Agriculture,» *Frontiers in Plant Science*, vol. 15, 2024, ISSN: 1664-462X. dirección: <https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2024.1369791>.
- [8] Y. J. Oh, J. Zhang, M. L. Fang et al., «A systematic review of artificial intelligence chatbots for promoting physical activity, healthy diet, and weight loss,» *International Journal of Behavioral Nutrition and Physical Activity*, vol. 18, n.º 1, pág. 160, 2021. dirección: <https://doi.org/10.1186/s12966-021-01224-6>.

- [9] E. Gutiérrez-González, M. García-Solano, R. Pastor-Barriuso et al., «Socio-geographical disparities of obesity and excess weight in adults in Spain: insights from the ENE-COVID study,» *Frontiers in Public Health*, vol. 11, 2023, ISSN: 2296-2565. dirección: <https://www.frontiersin.org/articles/10.3389/fpubh.2023.1195249>.
- [10] Naciones Unidas, *Objetivos de Desarrollo Sostenible*, 2024. dirección: <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/>.
- [11] McKinsey & Company, «The state of AI in 2022—and a half decade in review,» *McKinsey Global Institute*, 2022. dirección: <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-in-2022-and-a-half-decade-in-review>.
- [12] S. U. Human-Centered AI Institute, «AI Index Report 2024,» Stanford University, 2024. dirección: <https://aiindex.stanford.edu/>.
- [13] McKinsey & Company, «The state of AI in 2023: Generative AI's breakout year,» *McKinsey Global Institute*, 2023. dirección: <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-in-2023-generative-ais-breakout-year>.
- [14] DeepMind, *AlphaFold: Using AI for scientific discovery*, Accedido: 23 de mayo de 2024, 2024. dirección: <https://deepmind.com/research/case-studies/alphafold>.
- [15] V. Ivanov, «A review of fuzzy methods in automotive engineering applications,» *European Transport Research Review*, vol. 7, pág. 29, 2015. dirección: <https://doi.org/10.1007/s12544-015-0179-z>.
- [16] J. Dong, «Recent Advances in Swarm Intelligence Algorithms and Their Applications,» *Mathematics*, vol. 11, n.º 12, pág. 2624, 2023. dirección: https://www.mdpi.com/journal/mathematics/special_issues/Y758SX8ZQC.
- [17] C. Darwin, *On the Origin of Species*. London: John Murray, 1859.
- [18] L. J. Fogel, A. J. Owens y M. J. Walsh, *Artificial Intelligence through Simulated Evolution*. New York: John Wiley & Sons, Inc., 1966.
- [19] I. Rechenberg, *Evolution Strategy: Optimization of Technical Systems According to the Principles of Biological Evolution*. Stuttgart, Germany: Frommann-Holzboog, 1973.
- [20] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press, 1975.
- [21] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

- [22] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA, USA: MIT Press, 1994.
- [23] J. F. Miller y P. Thomson, «Cartesian Genetic Programming,» en *Genetic Programming*, R. Poli, W. Banzhaf, W. B. Langdon, J. Miller, P. Nordin y T. C. Fogarty, eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, págs. 121-132, ISBN: 978-3-540-46239-2. dirección: https://doi.org/10.1007/978-3-540-46239-2_9.
- [24] P. Moscato y C. Cotta, «A Gentle Introduction to Memetic Algorithms,» en *Handbook of Metaheuristics*, F. Glover y G. A. Kochenberger, eds., Springer, 2003, págs. 105-144. dirección: https://link.springer.com/chapter/10.1007/978-1-4419-1665-5_5.
- [25] NASA Jet Propulsion Laboratory, *Space Technology 5*, Accedido: 23 de mayo de 2024, 2006. dirección: <https://www.jpl.nasa.gov/nmp/st5>.
- [26] J. D. Lohn, G. S. Hornby y D. S. Linden, *An Evolved Antenna for Deployment on NASA's Space Technology 5 Mission*, Accedido: 23 de mayo de 2024, 2004. dirección: <https://ntrs.nasa.gov/citations/20040152147>.
- [27] R. Abraham, M. E. Samad, A. M. Bakhach et al., «Forecasting a Stock Trend Using Genetic Algorithm and Random Forest,» *Journal of Risk and Financial Management*, vol. 15, n.º 5, 2022, ISSN: 1911-8074. dirección: <https://www.mdpi.com/1911-8074/15/5/188>.
- [28] C. Notredame y D. G. Higgins, «SAGA: Sequence Alignment by Genetic Algorithm,» *Nucleic Acids Research*, vol. 24, n.º 8, págs. 1515-1524, 1996, ISSN: 0305-1048. dirección: <https://doi.org/10.1093/nar/24.8.1515>.
- [29] G. J. Stigler, «The Cost of Subsistence,» *Journal of Farm Economics*, vol. 27, n.º 2, págs. 303-314, 1945.
- [30] U. de Cádiz, *¿Qué son los Problemas de Dieta?* Accedido: 2024-06-25, 2024. dirección: <https://knuth.uca.es/moodle/mod/page/view.php?id=4498>.
- [31] S. Kirkpatrick, C. D. Gelatt y M. P. Vecchi, «Optimization by Simulated Annealing,» *Science*, vol. 220, n.º 4598, págs. 671-680, 1983. dirección: <https://science.sciencemag.org/content/220/4598/671>.
- [32] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.
- [33] A. Kahraman y H. Seven, «Healthy Daily Meal Planner,» jun. de 2005, págs. 390-393. dirección: <https://doi.org/10.1145/1102256.1102345>.
- [34] E. Kaldirim y Z. Köse, «Application of a Multi-objective Genetic Algorithm to the Modified Diet Problem,» en *Genetic and Evolutionary Computation Congress (GECCO), Undergraduate Student Workshop*, Seattle, USA, 2006.

- [35] T. Kashima, S. Matsumoto y H. Ishii, «Evaluation of Menu Planning Capability Based on Multi-dimensional 0/1 Knapsack Problem of Nutritional Management System,» *IAENG International Journal of Applied Mathematics*, vol. 39, n.º 3, págs. 163-170, 2009. dirección: https://www.iaeng.org/IJAM/issues_v39/issue_3/IJAM_39_3_04.pdf.
- [36] P. Heinonen y E. Juuso, «Development of a Genetic Algorithms Optimization Algorithm for a Nutritional Guidance Application,» en *Proceedings of The 9th EUROSIM Congress on Modelling and Simulation, EUROSIM 2016, The 57th SIMS Conference on Simulation and Modelling SIMS 2016*, E. Juuso, E. Dahlquist y K. Leiviskä, eds., ép. Linköping Electronic Conference Proceedings, Linköping University Electronic Press, 2016, págs. 755-761. dirección: <https://doi.org/10.3384/ecp1714255>.
- [37] S. Kilicarslan, M. Celik y S. Sahin, «Hybrid models based on genetic algorithm and deep learning algorithms for nutritional Anemia disease classification,» *Biomedical Signal Processing and Control*, vol. 63, págs. 102231, 2021, ISSN: 1746-8094. dirección: <https://doi.org/10.1016/j.bspc.2020.102231>.
- [38] J. B. Cole y R. Gabbianelli, «Recent Advances in Nutrigenomics: Making Strides Towards Precision Nutrition,» *Frontiers in Genetics*, vol. 13, 2022. dirección: <https://doi.org/10.3389/fgene.2022.997266>.
- [39] C. A. C. Coello, «Constraint-handling techniques used with evolutionary algorithms,» en *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ép. GECCO '22, Boston, Massachusetts: Association for Computing Machinery, 2022, págs. 1310-1333. dirección: <https://doi.org/10.1145/3520304.3533640>.
- [40] J. Blank y K. Deb, *NSGA-II*, Accedido: 2024-06-29, 2024. dirección: <https://pymoo.org/algorithms/moo/nsga2.html>.
- [41] G. Acampora, U. Kaymak, V. Loia y A. Vitiello, «Applying NSGA-II for solving the Ontology Alignment Problem,» oct. de 2013, págs. 1098-1103, ISBN: 978-1-4799-0652-9. DOI: [10.1109/SMC.2013.191](https://doi.org/10.1109/SMC.2013.191).
- [42] J. Blank y K. Deb, *pymoo: Multi-objective Optimization in Python - SPEA2 Implementation*, Accedido: 2024-06-29, 2024. dirección: <https://github.com/anyoptimization/pymoo/blob/main/pymoo/algorithms/moo/spea2.py>.
- [43] Y. Song y X. Fang, «An Improved Strength Pareto Evolutionary Algorithm 2 with Adaptive Crossover Operator for Bi-Objective Distributed Unmanned Aerial Vehicle Delivery,» *Mathematics*, vol. 11, n.º 15, 2023, ISSN: 2227-7390. DOI: [10.3390/math11153327](https://doi.org/10.3390/math11153327). dirección: <https://www.mdpi.com/2227-7390/11/15/3327>.

- [44] J. Blank y K. Deb, *MOEA/D: Multi-Objective Evolutionary Algorithm based on Decomposition*, Accedido: 2024-06-29, 2024. dirección: <https://pymoo.org/algorithms/moo/moead.html>.
- [45] A. Zhou, Q. Zhang y G. Zhang, «A multiobjective evolutionary algorithm based on decomposition and probability model,» jun. de 2012, págs. 1-8, ISBN: 978-1-4673-1510-4. DOI: [10.1109/CEC.2012.6252954](https://doi.org/10.1109/CEC.2012.6252954).
- [46] P. H. England, *McCance and Widdowson's Composition of Foods Integrated Dataset (CoFID)*, Accedido: 2024-06-04, 2021. dirección: <https://www.gov.uk/government/publications/composition-of-foods-integrated-dataset-cofid>.
- [47] MySQL, *MySQL Workbench*, Accedido: 2024-06-06, 2024. dirección: <https://www.mysql.com/products/workbench/>.
- [48] S. E. de Nutrición Comunitaria (SENC), *Guía de la Alimentación Saludable para Atención Primaria y Colectivos Ciudadanos*, Accedido: 2024-06-07. dirección: <https://www.nutricioncomunitaria.org/es/noticia/guia-alimentacion-saludable-ap>.
- [49] M. D. Mifflin, S. T. St Jeor, L. A. Hill, B. J. Scott, S. A. Daugherty e Y. O. Koh, «A new predictive equation for resting energy expenditure in healthy individuals,» *American Journal of Clinical Nutrition*, vol. 51, n.º 2, págs. 241-247, 1990.
- [50] L. K. Mahan y J. L. Raymond, *Krause's Food & the Nutrition Care Process*, 14th. Elsevier Health Sciences, 2016, págs. 370-372.
- [51] I. of Medicine, *Dietary Reference Intakes for Energy, Carbohydrate, Fiber, Fat, Fatty Acids, Cholesterol, Protein, and Amino Acids*. Washington, DC: The National Academies Press, 2005, Disponible en: <https://nap.nationalacademies.org/read/10490/chapter/7> y <https://nap.nationalacademies.org/read/10490/chapter/13>.
- [52] J. Quesada Pajares, *NutritionPlanning*, Accedido: 2024-07-03, 2024. dirección: <https://github.com/JaviQuesada/NutritionPlanning/tree/main/PYTHON>.
- [53] J. Blank, *pymoo: Multi-objective Optimization in Python*, Accedido: 2024-06-15, 2024. dirección: <https://pymoo.org/%20y%20https://github.com/anyoptimization/pymoo>.
- [54] J. Blank, *Violación de Restricciones (CV) Como Objetivo*, Accedido: 2024-07-07, 2020. dirección: [https://pymoo.org/constraints/as_obj.html#Constraint-Violation-\(CV\)-As-Objective](https://pymoo.org/constraints/as_obj.html#Constraint-Violation-(CV)-As-Objective).
- [55] Python Software Foundation, *tkinter — Interfaces gráficas de usuario — documentación de Python 3.11.4*, Accedido: 2024-07-04. dirección: <https://docs.python.org/es/3/library/tkinter.html>.

- [56] J. Blank y K. Deb, *pymoo: Multi-objective Optimization in Python - Hypervolume Calculation*, Accedido: 2024-06-27, 2024. dirección: [https : //pymoo.org/misc/indicators.html#hypervolume](https://pymoo.org/misc/indicators.html#hypervolume).
- [57] SciPy Developers, *scipy.stats.wilcoxon*, Accedido: 2024-06-27, 2024. dirección: <https://docs.scipy.org/doc/scipy-1.12.0/reference/generated/scipy.stats.wilcoxon.html>.
- [58] STAC Developers, *stac.nonparametric_tests.friedman_aligned_ranks_test*, Accedido: 2024-06-27, 2024. dirección: https://tec.citius.usc.es/stac/doc/stac.nonparametric_tests.friedman_aligned_ranks_test.html#stac.nonparametric_tests.friedman_aligned_ranks_test.
- [59] STAC Developers, *stac.nonparametric_tests.shaffer_multitest*, Accedido: 2024-06-27, 2024. dirección: https://tec.citius.usc.es/stac/doc/stac.nonparametric_tests.shaffer_multitest.html#stac.nonparametric_tests.shaffer_multitest.
- [60] Ministerio de Sanidad, Consumo y Bienestar Social, *Distribución diaria de la alimentación saludable*, Accedido: 2024-06-14. dirección: [https : // estilosdevidasaludable . sanidad . gob . es / alimentacionSaludable / queSabemos/enLaPractica/distribuir/diario/home.htm](https://estilosdevidasaludable.sanidad.gob.es/alimentacionSaludable/queSabemos/enLaPractica/distribuir/diario/home.htm).
- [61] J. Quesada Pajares, *NutritionPlanning - Reparación*, Accedido: 2024-07-03, 2024. dirección: https://github.com/JaviQuesada/NutritionPlanning/blob/main/PYTHON/manejo_restricciones/ag_reparacion.py.
- [62] Plan de Recuperación, Transformación y Resiliencia, *¿Qué es la inteligencia artificial (IA)? - PRTR*, Accedido: 2024-06-27, 2024. dirección: [https : // planderecuperacion . gob . es / noticias / que - es - inteligencia - artificial-ia-prtr](https://planderecuperacion.gob.es/noticias/que-es-inteligencia-artificial-ia-prtr).
- [63] A. E. Eiben y J. E. Smith, *Introduction to Evolutionary Computing*, 1st. Berlin, Heidelberg: Springer, 2003, ISBN: 978-3-540-40184-1. dirección: <https://doi.org/10.1007/978-3-662-05094-1>.

