Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

TFG del Grado en Ingeniería Informática

**Sibling Rewiring 2.0**

Optimización multiobjetivo para reducir el riesgo de contagio en entornos escolares

Imagen que contiene Logotipo

Descripción generada automáticamente

Presentado por Rubén Arasti Blanco

en la Universidad de Burgos — 5 de julio de 2024

Tutores: Dr. José Manuel Galán Ordax

y Dra. Virginia Ahedo García

# 

# Índice General

[Índice General 1](#_Toc169950851)

[Índice de Ilustraciones 2](#_Toc169950852)

[1. Introducción 3](#_Toc169950853)

[1.1 Estructura de la memoria 4](#_Toc169950854)

[1.2 Materiales adjuntos 4](#_Toc169950855)

[2. Objetivos del proyecto 5](#_Toc169950856)

[2.1 Objetivos generales 5](#_Toc169950857)

[2.2 Objetivos técnicos 5](#_Toc169950858)

[2.3 Objetivos personales 5](#_Toc169950859)

[3. Conceptos teóricos 6](#_Toc169950860)

[3.1 Definiciones básicas de las redes 6](#_Toc169950861)

[3.2 Procesamiento de los datos 7](#_Toc169950862)

[3.2 Resolución del problema 7](#_Toc169950863)

[3.3 Aproximaciones para la Optimización Multiobjetivo 8](#_Toc169950864)

[3.4 Algoritmo genético 9](#_Toc169950865)

[3.4 Estrategia NSGA-II para optimización multiobjetivo 10](#_Toc169950866)

[4. Técnicas y herramientas 10](#_Toc169950867)

[Entorno de desarrollo 10](#_Toc169950868)

[Gestor de proyectos 10](#_Toc169950869)

[Documentación 10](#_Toc169950870)

[5. Aspectos relevantes en el desarrollo del proyecto 10](#_Toc169950871)

[6. Trabajos relacionados 10](#_Toc169950872)

[7. Conclusiones y Líneas de trabajo futuras 11](#_Toc169950873)

[Bibliografía 11](#_Toc169950874)

# Índice de Ilustraciones

[Ilustración 1: Ejemplo gráfico del frente de Pareto 13](#_Toc170050835)

[Ilustración 2: Frente de Pareto con 3 objetivos 14](#_Toc170050836)

# Índice de Tablas

Tabla 1: Matriz de hermanos 9

# 1. Introducción

La pandemia del COVID-19 ha resaltado la importancia de implementar medidas eficaces para minimizar la transmisión de enfermedades en diversos entornos, incluyendo las instituciones educativas. En este contexto, las escuelas enfrentan el desafío de organizar a los estudiantes de manera que se reduzcan los riesgos de contagio.

Una de las estrategias que se implantaron durante la pandemia fue la implementación de los llamados grupos burbuja [1]. Estos grupos consisten en conjuntos de alumnos y profesores que interactúan exclusivamente entre ellos. Esta medida permite que los miembros de un mismo grupo compartan asistencia, espacio y actividad, mientras se limita estrictamente el contacto con personas de otros grupos. El objetivo principal de esta estrategia es contener y reducir la propagación del virus al minimizar las interacciones entre distintos grupos de estudiantes, haciendo más fácil el rastreo y la contención de los posibles brotes.

En esta implementación, se juntaba a los hermanos que estuvieran en la misma etapa y curso en el mismo grupo. Esta idea parte de la premisa de que los hermanos, al convivir en el mismo hogar, tenían una alta probabilidad de contagiarse mutuamente, y, por tanto, deberían estar en la misma clase para evitar la transmisión entre diferentes grupos.

Sin embargo, esta medida no contempló adecuadamente la situación de los hermanos que no estuvieran en la misma etapa o curso. Al pertenecer a clases distintas, estos hermanos seguían representando un potencial riesgo entre grupos diferentes, lo que minaba la efectividad de los grupos burbuja.

Al incluir a los hermanos en la ecuación, el problema se convierte en un problema de **optimización multiobjetivo**, ya que se deben equilibrar múltiples factores como la segregación de conjuntos de grupos, el tamaño de los mismos y la interacción entre hermanos.

Sibling Rewiring (SiRe) es una aplicación diseñada para optimizar la organización de alumnos en grupos burbuja con el objetivo de reducir el riesgo de contagio. Fue desarrollada como Trabajo de Fin de Grado por María Ojeda Ruiz en la Universidad de Burgos en 2021 [2]. En esta continuación del trabajo original, se introducirán nuevas funcionalidades y mejoras significativas que culminarán en Sibling Rewiring 2.0.

El proyecto se basa en el marco conceptual de redes complejas [3]. Se han modelado dos redes principales. La primera, incluye a todos los alumnos como nodos con atributos de grupo, etapa y curso, y los enlaces representan sus interacciones: la pertenencia a la misma clase o al mismo hogar. La segunda, tiene como nodos las distintas clases del colegio y como enlaces las relaciones de hermanos entre las clases, siendo los pesos de los enlaces el número de relaciones y pudiendo existir autoenlaces.

La principal limitación del trabajo original radicaba en la metodología de optimización multiobjetivo. La solución se obtenía sumando todos los objetivos en una única función de costo, lo que no permitía explorar el conjunto completo de soluciones posibles que podría ofrecer un enfoque basado en el frente de Pareto. Esto limitaba la capacidad para encontrar soluciones diversas y equilibradas que pudieran satisfacer los diferentes objetivos simultáneamente.

Para abordar esta limitación, Sibling Rewiring 2.0 incorpora una optimización multiobjetivo que utiliza el frente de Pareto, permitiendo identificar un conjunto de soluciones eficientes que ofrecen un compromiso equilibrado entre los diferentes objetivos.

Naturalmente, la metodología desarrollada en este proyecto no sólo es aplicable a la contención del COVID-19, sino a cualquier enfermedad contagiosa, como la gripe o la varicela. Esta versatilidad hace que el sistema sea una herramienta valiosa para la prevención y control de diversas infecciones en cualquier situación epidemiológica.

## 1.1 Estructura de la memoria

## 1.2 Materiales adjuntos

# 2. Objetivos del proyecto

A continuación, se detallan los objetivos del proyecto en tres categorías principales: generales, técnicos y personales.

## 2.1 Objetivos generales

* Desarrollar una aplicación web para optimizar la organización de los alumnos en un colegio y reducir el riesgo de contagio.
* Implementar un algoritmo genético como método de optimización multiobjetivo para aportar diversidad de soluciones al problema.
* Mostrar gráficamente la evaluación de distintas soluciones exploradas y resaltar el frente de Pareto.
* Aportar una solución intuitiva para el usuario y con información relevante para la toma de decisiones.

## 2.2 Objetivos técnicos

* Desarrollar un algoritmo genético mediante el framework DEAP.
* Trabajar con la librería NetworkX para el manejo y análisis de redes.
* Desplegar la aplicación web en Netlify.
* Utilizar Git como herramienta de control de versiones.
* Utilizar Trello como herramienta de gestor de proyectos.
* Aplicar la metodología ágil SCRUM para la gestión del proyecto.

## 2.3 Objetivos personales

* Consolidar conocimientos sobre algoritmos genéticos y la optimización multiobjetivo.
* Adquirir experiencia en metodologías ágiles.
* Aprender el proceso de despliegue de una web.
* Poner en práctica los conocimientos de la ingeniería de software adquiridos durante la carrera.

# 3. Conceptos teóricos

En este apartado se abordarán los conceptos fundamentales que subyacen el desarrollo del proyecto. Se presentarán las definiciones básicas relacionadas con las redes, se explicará el procesamiento de los datos, se explicarán los principios de la optimización multiobjetivo y se detallará el funcionamiento del algoritmo genético.

## 3.1 Definiciones básicas de las redes

Una **red** (también llamada grafo en el ámbito matemático) es un conjunto de objetos llamados nodos (o vértices) con conexiones entre ellos llamadas enlaces (o aristas). Algunos ejemplos de redes pueden ser las redes sociales o Internet. [3]

Cada **nodo** puede tener atributos que describen características específicas del nodo, como nombre, peso, categoría, entre otros. Además, los nodos de una red pueden pertenecer a distintos tipos. Si todos los nodos de una red son del mismo tipo, la red se denomina unimodal. Por otro lado, si los nodos son de tipos diferentes, la red se clasifica como multimodal.

En cuanto a los **enlaces**, pueden ser dirigidos o no dirigidos. En un enlace dirigido, la relación va de un nodo a otro, lo que es útil para representar relaciones como jerarquías o flujos. Por otro lado, los enlaces no dirigidos representan relaciones bidireccionales o simétricas, como la amistad en redes sociales, donde la relación es recíproca. Los enlaces también pueden tener **pesos**, que asignan un valor numérico a la conexión. Estos pesos son cruciales para representar la importancia relativa de cada conexión dentro de la red.

Entre dos nodos pueden existir varios enlaces, conocidos como **enlaces múltiples**, que permiten representar diversas relaciones o interacciones simultáneas entre los mismos nodos. Un nodo también puede estar conectado consigo mismo, lo cual se denomina **autoenlace**, utilizado para modelar relaciones internas o autorreferencias.

Un **camino** se define como una secuencia de enlaces que conecta nodos en la red y que va desde un nodo inicial hasta un nodo final.

Un **componente** en es un conjunto de nodos que están conectados entre sí mediante caminos dentro de la red. Es decir, en un componente cada nodo puede alcanzar a cualquier otro nodo dentro del mismo siguiendo una serie de enlaces. Un nodo sin enlaces también se considera un componente.

Por otro lado, un **subgrafo** se define como un conjunto de nodos y los enlaces que existen entre ellos en una red determinada.

## 3.2 Procesamiento de los datos

En esta sección se explicará qué datos se utilizarán y cómo se manejarán para la resolución del problema. Primero se explicarán los datos de entrada, después el proceso de tratamiento de estos datos y por último los datos de salida obtenidos.

### Datos de entrada

Para definir el problema se utilizan dos archivos de entrada: un archivo .gexf, que almacena la red de estudiantes, y un archivo .csv, que contiene una tabla con la información de los estudiantes que tienen hermanos.

La red de estudiantes está compuesta por nodos que representan a cada estudiante y enlaces que indican las posibles interacciones entre ellos. Es decir, existe un enlace entre dos estudiantes si comparten la misma clase o conviven en el mismo hogar. Cada estudiante tiene los siguientes atributos: un identificador numérico que sirve como nombre, la etapa educativa a la que pertenece (infantil o primaria), el curso (1º, 2º, 3º, etc.), y el grupo asignado (A, B, C, etc.).

**(AQUÍ INSERTARÉ UNA IMAGEN DE LA RED de estudiantes EN GEPHI)**

Para identificar quién es hermano de quién y diferenciar las relaciones de hermandad en la red de estudiantes, utilizaremos la siguiente tabla para definir una matriz que incluirá los atributos de cada estudiante junto con el nombre de su hermano en cada fila. Esto nos permitirá mapear fácilmente las conexiones de hermandad entre los estudiantes.

Tabla 1: Matriz de hermanos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | nombre | etapa | curso | clase | hermano de |
| 0 | 594 | primaria | 1 | A | 784 |
| 1 | 784 | primaria | 2 | B | 594 |
| 2 | 685 | primaria | 1 | B | 827 |
| 3 | 827 | primaria | 2 | C | 685 |
| 4 | 817 | primaria | 2 | B | 2120 |
| 5 | 2120 | secundaria | 2 | C | 817 |

Aunque en el ejemplo de la Tabla 1 las filas de los hermanos se presenten juntas, no es necesario que sigan ningún orden específico. Además, no es obligatorio que cada hermano aparezca solo una vez; es posible que un estudiante aparezca varias veces en la tabla si tiene varios hermanos.

### Procesamiento de los datos

Para trabajar con los datos de entrada es necesario transformarlos a una red de clases como grafo de NetworkX. Para hacer esto, definiremos un grafo cuyos nodos serán las clases y cuyos enlaces serán las relaciones de hermanos entre clases. Es decir, si un hermano de la clase 1º de primaria A tiene un hermano en la clase 2º de primaria B, los nodos “primaria1A” y “primaria2B” estarán conectados entre sí. Cada nodo tiene como atributos un nombre (cadena con la etapa, el curso y el grupo), una etapa, un curso, un grupo y unos estudiantes (lista con los nombres de los estudiantes de esa clase).

**(AQUÍ INSERTARÉ UNA IMAGEN DE LA RED de clases EN NETWORKX)**

Antes de explicar la construcción del grafo, es necesario destacar que se requiere una solución previa, que se define como la asignación de los hermanos a los distintos grupos. Esto se debe a que necesitamos conocer la clase asignada a cada hermano para poder establecer las conexiones en el grafo.

El grafo se construye a partir de la red de estudiantes mencionada en la sección anterior. Se consideran todos los atributos de los nodos, exceptuando el nombre, y se generan todas las posibles combinaciones de clases.

A partir de la matriz de hermanos se crea un diccionario donde las claves son los nombres de los estudiantes, y los valores son una lista que incluye sus atributos y una lista de sus hermanos. Los hermanos quedan ordenados según su nombre.

Para crear los enlaces, se recorre un diccionario de hermanos y se establece un enlace entre las clases correspondientes a los hermanos. Si ya existe un enlace entre dos clases, en lugar de crear enlaces múltiples, se incrementa el peso del enlace en uno. En caso de que dos hermanos estén en la misma clase, se crea un autoenlace. Las listas de estudiantes de cada nodo se actualizan con los nombres de los hermanos correspondientes.

El grafo no está completamente definido ya que faltan los estudiantes que no tienen hermanos, pero es útil para evaluar las conexiones entre hermanos y seleccionar la mejor solución. Una vez elegida la solución final, se verifica que ninguna clase exceda su capacidad máxima, calculada como la división del número máximo de estudiantes de un curso entre el número total de grupos, redondeado hacia arriba. Este cálculo se basa en la idea que todos los estudiantes de un curso tienen el derecho de avanzar al siguiente, sin considerar a los repetidores. Si alguna clase excede la capacidad máxima, se reubican los hermanos sobrantes en otra clase y se realiza una nueva verificación para asegurar la viabilidad de la solución.

Una vez confirmada la viabilidad de la solución, se itera sobre los nodos del grafo de estudiantes y se completan las listas de estudiantes de los nodos del grafo de clases hasta alcanzar la capacidad máxima. Con este proceso, el grafo de clases queda completamente definido.

### Datos de salida

Se entregan cuatro archivos: un .gexf con la red de clases, un .png que muestra la red de clases dibujada por NetworkX, un .csv con los atributos de los estudiantes y otro .csv que detalla los pares de clases conectadas y los estudiantes que establecen dichas conexiones. Todos estos archivos vienen dentro de una carpeta cuyo nombre indica la evaluación de la solución, el número de solución y si la solución ha sido modificada para que sea viable.

El archivo en grafo.gexf contiene la representación estructural de la red de clases. Este formato es ampliamente utilizado para la visualización y análisis de redes, permitiendo que los datos sean importados y manipulados en diversas aplicaciones de análisis de redes, como Gephi.

El archivo grafo.png contiene el grafo con los nodos dibujados en círculo y con los componentes diferenciados por colores. También se muestra una leyenda que detalla el tamaño y el número de enlaces de cada componente para ayudar a la toma de decisiones.

**(IMAGEN DEL GRAFO CON LA LEYENDA)**

Por ejemplo, la leyenda permite identificar los componentes que presentan un mayor riesgo de contagio, lo cual es crucial para decisiones estratégicas, como asignar a un profesor en situación de riesgo a una clase ubicada en un componente de menor riesgo.

El archivo estudiantes.csv contiene la tabla de estudiantes muestra los atributos de cada estudiante para poder conocer de forma eficiente a qué grupo va asignado cada estudiante.

Tabla 2: Asignación de estudiantes

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | Etapa | Curso | Clase |
| 0 | infantil | 1 | A |
| 1 | infantil | 1 | B |
| 2 | infantil | 1 | C |

El archivo par\_clases.csv contiene la tabla de las clases que están conectadas y los estudiantes que las conectan. Esta información es especialmente útil para comprender las interacciones entre las clases y evaluar cómo la movilidad de los estudiantes entre diferentes grupos puede influir en la propagación de enfermedades. Por ejemplo, permite identificar qué clases están vinculadas a través de hermanos y, por lo tanto, podrían necesitar medidas adicionales de contención.

Tabla 3: Pares de clases conectadas

|  |  |  |
| --- | --- | --- |
| Clase 1 | Clase 2 | Estudiantes |
| infantil1A | secundaria4A | 146, 2510 |
| infantil1B | infantil2B | 17, 134, 136, 181, 235, 249, 269, 354 |
| infantil1B | primaria1C | 74, 99, 633, 659 |

## 3.3 Optimización Multiobjetivo

En esta sección se define el problema de optimización multiobjetivo y se explica concepto del frente de Pareto.

### Definición del problema

Un problema de optimización multiobjetivo consiste en un conjunto de funciones objetivo, sujetas a un conjunto de restricciones. Cada solución se representa mediante un vector de n variables de decisión, cada una de las cuales tiene límites superior e inferior específicos.[4]

En nuestro problema se busca que los grupos burbuja estén bien compartimentados y que el riesgo de contagio entre ellos sea homogéneo, evitando así que un grupo se vea desproporcionadamente afectado en comparación con otros. Para lograr esto, se deben considerar varios objetivos en la red de clases:

1. Maximizar el número de componentes. Se busca dividir la red en el mayor número posible de grupos burbuja que no tengan conexiones entre sí. Este objetivo se enfoca en minimizar las interacciones entre grupos, lo que reduce la probabilidad de propagación de la enfermedad de un grupo a otro.
2. Minimizar la variación en el tamaño de los componentes. Se busca que los componentes tengan tamaños similares. Si un componente es significativamente más grande que otro, el riesgo de contagio se incrementa debido a la mayor cantidad de interacciones totales.
3. Minimizar la variación en el número de enlaces dentro de cada componente. Se busca que la densidad de conexiones dentro de cada componente sea uniforme. En componentes de igual tamaño, la probabilidad de contagio será mayor en el que tenga más relaciones entre hermanos, ya que un mayor número de interacciones aumenta el riesgo de contagio.

Estos objetivos se corresponden con las funciones objetivo del problema.

La solución del problema será la asignación de los alumnos que tengan hermanos a un grupo de su curso y etapa. Para cada hermano necesitaremos una variable de decisión para decidir a qué grupo va. Por lo tanto, cada solución será un vector con tantas variables de decisión como el número de hermanos existentes. Dichas variables de decisión podrán tomar tantos valores como grupos por curso haya.

Un aspecto crucial para asegurar la viabilidad de una solución es garantizar que ninguna clase exceda su capacidad máxima. Esto puede ocurrir si, en un curso, hay más hermanos que el número de plazas disponibles en una clase y se asignan todos a un mismo grupo. Aunque es una situación poco probable, se aborda durante la fase final de la asignación de grupos. Al descargar la solución, se verifica si alguna clase supera su capacidad máxima. Si se detecta esta situación, se reubican los estudiantes excedentes a otras clases. Este proceso se repite hasta que todas las clases cumplan con la capacidad permitida.

### Frente de Pareto

Para comparar las distintas soluciones en este tipo de optimización, se suele optar por tres aproximaciones: la combinación de objetivos, la priorización de objetivos y el óptimo de Pareto.

La combinación de objetivos consiste en crear una única función objetivo resultado de la suma de las demás funciones objetivo. Se pueden ponderar los objetivos para que tengan la misma relevancia y multiplicar por -1 dependiendo si un objetivo se quiere minimizar o maximizar.

La priorización de objetivos consiste en establecer un orden para los objetivos de forma que dos soluciones tienen su primer objetivo igualado, se comparen mediante el segundo objetivo.

Estos dos últimos métodos sólo permiten ver una solución final, a pesar de que pueda haber varias soluciones con igual beneficio pero distintos valores para cada objetivo.

El óptimo de Pareto soluciona este problema. Se define de la siguiente manera:

Una solución domina otra solución si las dos siguientes condiciones se cumplen:

1. La solución es igual o mejor que para todas las funciones objetivo
2. Gráfico, Gráfico de cajas y bigotes

   Descripción generada automáticamenteLa solución es estrictamente mejor que en al menos un objetivo

Ilustración 1: Ejemplo gráfico del frente de Pareto

En la ilustración 1 se muestra una gráfica donde los ejes representan los valores de dos funciones objetivo ζ1(x) y ζ2(x). En este ejemplo las dos funciones se quieren minimizar. Se puede ver como la solución 1 domina a la 3 porque tiene valores menores al menos en el primer objetivo y es menor o igual en el segundo objetivo. La solución 1 no domina a la 2 ni es dominada por esta, debido a que es mejor en el primer objetivo pero peor en el segundo. La solución 3 no domina a la 2 ni es dominada por esta por la misma razón que la comparación anterior. Las soluciones 1 y 2 son no dominadas porque ninguna supera completamente a la otra en ambos objetivos.

El frente de Pareto está compuesto por las soluciones no dominadas y representa las mejores opciones disponibles, ofreciendo diversidad de soluciones para la toma de decisiones. Esta es la aproximación que se utilizará para resolver el problema.

En nuestro caso, el frente de Pareto tiene tres dimensiones. Presentamos una gráfica que muestra las combinaciones posibles para cada par de objetivos:

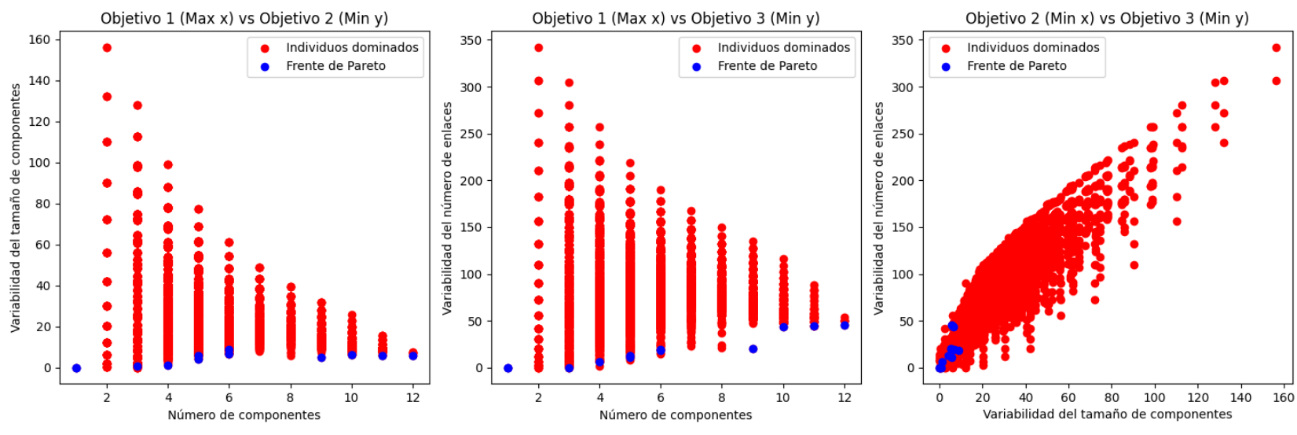
Se muestran las soluciones dominadas en rojo y el frente de Pareto en azul.

Ilustración 2: Frente de Pareto con 3 objetivos

## 3.4 Algoritmo genético

Los Algoritmos Genéticos (AG) son herramientas poderosas en la optimización y resolución de problemas complejos. Inspirados en procesos evolutivos naturales, los AG operan mediante la simulación de una población de posibles soluciones que evoluciona a través de generaciones. Utilizando operadores como cruce y mutación, exploran y explotan diversas combinaciones genéticas para buscar soluciones óptimas. Este enfoque no solo permite mejorar gradualmente las soluciones existentes (convergencia), sino que también asegura que se explore activamente el espacio de búsqueda en busca de nuevas y potencialmente mejores soluciones (diversidad y exploración).[5], [6], [7]

El ciclo de vida de un algoritmo genético sigue un proceso iterativo que incluye los siguientes pasos clave:

1. Inicialización de la población. Se genera una población inicial de posibles soluciones (individuos). En nuestro caso, la población inicial es generada aleatoriamente.
2. Evaluación de la población. Cada individuo es evaluado mediante la función de fitness para determinar su calidad o adaptación respecto al problema planteado.
3. Selección de padres. Se seleccionan los individuos que se reproducirán para generar la siguiente generación. En nuestro AG, se seleccionan a dos individuos aleatorios.[8]
4. Aplicación de operadores genéticos. Se aplican los operadores de cruce y mutación para crear nuevos individuos (descendientes).
5. Evaluación de descendientes. Los nuevos individuos generados son evaluados de nuevo utilizando la función de fitness.
6. Selección de la nueva población. Se seleccionan los individuos que formarán parte de la siguiente generación, combinando individuos antiguos y nuevos para mantener la diversidad genética y permitir la evolución hacia soluciones óptimas.
7. Iteración. Este proceso se repite durante un número definido de generaciones o hasta que se cumple un criterio de parada, como la convergencia de la población.

A continuación, es explicarán en detalle las partes básicas de los AG y cómo se implementan.

### Representación del genotipo

El genotipo se refiere a la representación genética de un individuo. Es una estructura que contiene la información genética codificada, que determina las características o variables de decisión de la solución al problema optimización. Cuando hablamos de un gen, nos referimos a una variable de decisión dentro del genotipo.

En nuestro problema, el genotipo se representa como una lista cuyo tamaño equivale al número de hermanos. Cada elemento de la lista es un valor que varía entre 0 y el número de grupos menos 1, correspondiendo cada valor a las letras del grupo: A, B, C, etc. Estos elementos representan el grupo asignado a cada hermano. La posición de cada elemento en la lista determina a qué hermano se asigna dicho grupo.

Por ejemplo, el individuo [0,1,0,3] quiere decir que el primer hermano está asignado al grupo A, el segundo al grupo B, el tercero al grupo A y el cuarto al grupo D.

Entonces, el número total de individuos (espacio de búsqueda) está definido por donde G es el número de grupos posibles y N es el número total de hermanos.

### Representación del fenotipo

El fenotipo se refiere a la representación o expresión de las soluciones en términos de variables observables o manipulables dentro del problema. Es la forma en que se traduce el genotipo en características o valores concretos que pueden ser evaluados y comparados según el objetivo del problema.

Nuestro fenotipo será la red de clases, que se construirá recorriendo el diccionario de hermanos. Para cada hermano, se examinará su asignación de grupo dentro del individuo y se establecerán enlaces entre las clases correspondientes.

### Función de adaptación o de fitness

La función de fitness es una métrica que evalúa la calidad de una solución candidata para resolver un problema dado. Esta función asigna un valor numérico que indica el rendimiento o la calidad de una solución particular, basándose en cómo se acerca esa solución a cumplir con los objetivos o restricciones del problema.

La función de fitness evaluada en un individuo específico calcula tres métricas principales para determinar su adaptación dentro del contexto del problema de optimización. Primero, construye una red de clases utilizando el fenotipo representado por el individuo. Luego, identifica los componentes en esta red y cuenta cuántos hay, lo que se corresponde con el primer objetivo. El segundo objetivo, se calcula como la varianza en los tamaños de estos componentes. Finalmente el tercer objetivo, mide la varianza en el número de enlaces dentro de cada componente. El número de enlaces se calcula como la suma de los pesos de todos los enlaces del componente.

### Operador de cruce

El operador de cruce facilita la combinación de información genética entre dos individuos para generar nuevos descendientes. Se han implementado tres tipos de cruce: cruce de punto único, cruce de dos puntos y cruce uniforme.[9]

Imagen que contiene Diagrama

Descripción generada automáticamenteEl **cruce de punto único** selecciona un punto de corte en la secuencia de genes y combina los segmentos de los dos progenitores para crear dos nuevos descendientes. Es una técnica sencilla que permite mezclar segmentos grandes de información genética de cada progenitor, manteniendo la estructura de los genes adyacentes.

Ilustración 3: Cruce de punto único

Diagrama

Descripción generada automáticamente con confianza mediaEn el **cruce de dos puntos**, se seleccionan dos puntos de corte y se intercambian los segmentos situados entre estos puntos entre los dos progenitores. Este método permite una mayor diversidad en la combinación de genes.

Ilustración 4: Cruce de dos puntos

**En el cruce uniforme** cada gen del descendiente se elige aleatoriamente de uno de los dos progenitores, sin considerar la posición. Este método garantiza una mezcla completa de la información genética, ya que cada gen se selecciona de forma independiente, favoreciendo una mayor diversidad genética en la población resultante.

Se han elegido estos métodos de cruce porque son los que conocía de asignaturas previas, como “Computación Neuronal y Evolutiva” y “Organización y Gestión de Empresas”. Los métodos de cruce específicos para permutaciones no se consideraron relevantes para este problema, ya que están diseñados principalmente para problemas de ordenación**.**

### Operador de mutación

Los operadores de mutación en un algoritmo genético introducen variaciones aleatorias en la descendencia para mantener la diversidad genética y explorar nuevas áreas del espacio de búsqueda.

Se ha implementado la **mutación uniforme**, que modifica los valores de los genes de un individuo de manera aleatoria. En este proceso, cada gen del individuo tiene una probabilidad independiente de ser cambiado a otro valor dentro de su rango permitido. Este tipo de mutación es particularmente eficaz para mantener la diversidad en la población. La probabilidad para la mutación de cada gen es de 0.2, lo que significa que aproximadamente 1 de cada 5 genes experimentará una mutación.

### Operador de selección: NSGA-II para optimización multiobjetivo

Diagrama

Descripción generada automáticamenteEl operador de selección en un algoritmo genético determina qué individuos de la población actual se reproducirán para crear la siguiente generación. Para dicha selección, se utilizará el NSGA-II (Nondominated Sorting Genetic Algorithm II) [10], que es un método eficaz diseñado específicamente para la optimización multiobjetivo en algoritmos genéticos.

Ilustración 5: Procedimiento del NSGA-II

El funcionamiento del NSGA-II se resume en los siguientes pasos:

1. A partir de la población actual () se produce la descendencia ().
2. Se combina la población actual con la descendencia y se ordenan los individuos en una lista de frentes (listas de individuos) según su nivel de dominancia. Por ejemplo, el primer frente () contiene soluciones no dominadas por ninguna otra en la población, mientras que los frentes subsiguientes (, , …) contienen soluciones dominadas por una, dos o más soluciones de los frentes anteriores.
3. Gráfico, Gráfico de dispersión

   Descripción generada automáticamenteEn el proceso de selección para la siguiente generación, se seleccionan tantos individuos como permita el tamaño deseado de la población, comenzando desde el primer frente de Pareto y continuando con los frentes siguientes en orden. Si el tamaño del frente excede la capacidad de la población, se utiliza la distancia de agrupación (crowding distance) para priorizar qué individuos conservar. Esta métrica es calculada para mantener la diversidad de soluciones con distinto fitness. Se calcula como el perímetro del cuboide formado por los vértices de sus soluciones vecinas:

Ilustración 6: Crowding Distance. Los puntos rellenos son soluciones del mismo frente

La población obtenida del corte pasa a ser la población actual () y se repite el paso 1 hasta llegar al criterio de parada del algoritmo genético.

Al combinar la población actual con la descendencia generada, se previene la pérdida de las mejores soluciones óptimas durante el proceso de evolución. Cuando el tamaño del frente de Pareto excede el tamaño deseado de la población, se priorizan los individuos con mayor distancia de agrupación, lo cual asegura la preservación de la diversidad de soluciones respecto a los objetivos del problema.

### Configuración de parámetros

Normalmente en los AG se seleccionan cuatro parámetros clave: tamaño de la población, número de generaciones, probabilidad de cruce y probabilidad de mutación. En nuestra implementación, estos parámetros pueden ser ajustados manualmente por el usuario o se pueden utilizar valores por defecto. A continuación, se explicarán estos parámetros y sus efectos en el desempeño del algoritmo.

El **tamaño de la población** se refiere al número de individuos presentes en cada generación. Este parámetro es crucial porque una población más grande puede explorar una mayor parte del espacio de búsqueda, aumentando la probabilidad de encontrar soluciones óptimas. Sin embargo, también incrementa el tiempo de cómputo. Por el contrario, una población demasiado pequeña puede llevar a una exploración insuficiente y a la convergencia prematura a soluciones subóptimas. El **número de generaciones** determina cuántas iteraciones realizará el algoritmo antes de finalizar. Un mayor número de generaciones permite al algoritmo explorar y refinar las soluciones durante más tiempo, lo que aumenta las posibilidades de encontrar una solución óptima. No obstante, esto también implica un mayor costo computacional. Un número insuficiente de generaciones puede hacer que el algoritmo no tenga tiempo suficiente para converger a una solución adecuada.[11]

Dado que el tamaño del espacio de búsqueda no se conoce de antemano, se ha optado por establecer como parámetros por defecto un tamaño de población de 200 individuos y ejecutar 200 generaciones, ya que este enfoque generalmente requiere menos de un minuto para la mayoría de los casos.

En cada generación, la **probabilidad de cruce** controla la proporción de soluciones que se someten a la recombinación genética. Si es muy alto puede causar una pérdida de diversidad genética y a una convergencia prematura y si es muy bajo puede ralentizar la capacidad de exploración del algoritmo. La mutación es sólo un operador secundario para restaurar el material genético. Sin embargo, la **probabilidad de mutación** es un parámetro crítico para el desempeño del algoritmo y altos valores de este parámetro transforman al algoritmo genético en una búsqueda aleatoria. La mutación previene al algoritmo de la convergencia prematura. Según la literatura científica, los valores de suelen estar entre 0.5 y 1.0 y los valores de suelen estar entre 0.005 y 0.05. [5]

Para los valores por defecto se escogerán 0.6 para , que no es un valor muy elevado y nos asegura una correcta exploración, y 0.05 para , que nos garantiza que hay suficiente diversidad genética.

# 4. Técnicas y herramientas

## Entorno de desarrollo

## Gestor de proyectos

He elegido Trello como gestor de proyectos. Es una aplicación web gratuita y fácil de utilizar.[12]

## Documentación

La fuente tipográfica escogida es EB Garamond que tiene una licencia SIL Open Font License.[13]

# 5. Aspectos relevantes en el desarrollo del proyecto

# 6. Trabajos relacionados

# 7. Conclusiones y Líneas de trabajo futuras

# Bibliografía

[1] «¿Qué son los grupos burbuja? | Glosario Covid». Accedido: 20 de junio de 2024. [En línea]. Disponible en: https://www.unilabs.es/glosario/grupos-burbuja

[2] «Mariaojruiz/Sibling-Rewiring: Proyecto de la universidad de Burgos tutelado por José Manuel Galán y Virginia Ahedo. Aplicación que realizará las modificaciones necesarias en las aulas para disminuir el número de contagios entre los alumnos al máximo.» Accedido: 20 de junio de 2024. [En línea]. Disponible en: https://github.com/Mariaojruiz/Sibling-Rewiring

[3] M. E. J. Newman, «The Structure and Function of Complex Networks», *SIAM Rev.*, vol. 45, n.o 2, pp. 167-256, ene. 2003, doi: 10.1137/S003614450342480.

[4] S. B. Selçuklu, «Multi-objective Genetic Algorithms», en *Handbook of Formal Optimization*, A. J. Kulkarni y A. H. Gandomi, Eds., Singapore: Springer Nature, 2023, pp. 1-37. doi: 10.1007/978-981-19-8851-6\_31-1.

[5] M. Srinivas y L. M. Patnaik, «Adaptive probabilities of crossover and mutation in genetic algorithms», *IEEE Trans. Syst. Man Cybern.*, vol. 24, n.o 4, pp. 656-667, abr. 1994, doi: 10.1109/21.286385.

[6] «CNE\_ejemplos/genetic\_prog at master · bbaruque/CNE\_ejemplos». Accedido: 1 de junio de 2024. [En línea]. Disponible en: https://github.com/bbaruque/CNE\_ejemplos/tree/master/genetic\_prog

[7] «Genetic algorithm», *Wikipedia*. 28 de marzo de 2024. Accedido: 23 de junio de 2024. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Genetic\_algorithm&oldid=1215928299

[8] «deap.algorithms — DEAP 1.4.1 documentation». Accedido: 23 de junio de 2024. [En línea]. Disponible en: https://deap.readthedocs.io/en/master/\_modules/deap/algorithms.html#varOr

[9] «Crossover (genetic algorithm)», *Wikipedia*. 15 de enero de 2024. Accedido: 23 de junio de 2024. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Crossover\_(genetic\_algorithm)&oldid=1195846537

[10] K. Deb, A. Pratap, S. Agarwal, y T. Meyarivan, «A fast and elitist multiobjective genetic algorithm: NSGA-II», *IEEE Trans. Evol. Comput.*, vol. 6, n.o 2, pp. 182-197, abr. 2002, doi: 10.1109/4235.996017.

[11] D. Vrajitoru, «Large Population or Many Generations for Genetic Algorithms? Implications in Information Retrieval», en *Soft Computing in Information Retrieval*, vol. 50, F. Crestani y G. Pasi, Eds., en Studies in Fuzziness and Soft Computing, vol. 50. , Heidelberg: Physica-Verlag HD, 2000, pp. 199-222. doi: 10.1007/978-3-7908-1849-9\_9.

[12] «Gestiona los proyectos de tu equipo desde cualquier lugar | Trello». Accedido: 29 de abril de 2024. [En línea]. Disponible en: https://trello.com/es

[13] E. Dunham, «License for font family 'EB Garamond’ » Font Squirrel». Accedido: 28 de abril de 2024. [En línea]. Disponible en: https://www.fontsquirrel.com/license/eb-garamond