



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería Informática

Sibling Rewiring 2.0

Optimización multiobjetivo para
reducir el riesgo de contagio en
entornos escolares



Presentado por Rubén Arasti Blanco
en la Universidad de Burgos — 9 de julio de 2024

Tutores: Dr. José Manuel Galán Ordax
y Dra. Virginia Ahedo García

Resumen

La pandemia global del COVID-19 ha resaltado la necesidad de reestructurar la organización escolar para mitigar los riesgos de contagio entre los estudiantes.

La estrategia principal en los colegios consistía en aislar las clases para evitar el contacto con otros grupos y en agrupar a los hermanos que compartían el mismo curso, partiendo de la premisa de que sus interacciones en el hogar aumentaban el riesgo de contagio entre grupos diferentes. No obstante, este enfoque no tenía en cuenta adecuadamente a los hermanos que no estaban en la misma edad ni curso, lo cual continuaba siendo un potencial riesgo de transmisión entre diferentes grupos.

Para abordar esta complejidad, este proyecto introduce Sibling-Rewiring 2.0 (SiRe), una aplicación que emplea un algoritmo genético y principios de redes complejas para reconfigurar eficazmente la distribución de estudiantes en las escuelas. SiRe establece conexiones entre clases basadas en relaciones de hermandad, buscando optimizar tanto el riesgo de contagio individual como grupal. Este enfoque innovador no solo se dirige al COVID-19, sino que ofrece una solución adaptable para gestionar otros brotes de enfermedades transmitidas por el aire o contacto cercano en entornos educativos.

La herramienta ha sido desplegada como una aplicación web

Descriptores

Redes complejas, optimización combinatoria, minimización del contagio, análisis de comunidades, pandemia, COVID-19, Python, Servidor Flask, MySQL.

Abstract

The global COVID-19 pandemic has underscored the need to restructure school organization to mitigate contagion risks among students.

The primary strategy in schools involved isolating classes to prevent contact with other groups and grouping siblings who shared the same grade, based on the premise that their interactions at home increased the risk of transmission between different groups. However, this approach did not adequately consider siblings who were not in the same age or grade, which remained a potential risk for transmission between different groups.

To address this complexity, this project introduces Sibling-Rewiring 2.0 (SiRe), an application that utilizes a genetic algorithm and complex network principles to effectively reconfigure student distribution in schools. SiRe establishes connections between classes based on sibling relationships, aiming to optimize both individual and group contagion risks. This innovative approach not only targets COVID-19 but also provides an adaptable solution for managing other outbreaks of airborne or close-contact diseases in educational settings.

The tool has been deployed as a web application.

Keywords

Complex networks, combinatorial optimization, contagion minimization, community analysis, pandemic, COVID-19, Python, Flask server, MySQL.

Índice General

Índice General	3
Índice de Ilustraciones	5
Índice de Tablas	6
1. Introducción	7
1.1 Estructura de la memoria.....	8
1.2 Materiales adjuntos.....	8
2. Objetivos del proyecto	9
2.1 Objetivos generales	9
2.2 Objetivos técnicos.....	9
2.3 Objetivos personales	9
3. Conceptos teóricos.....	10
3.1 Definiciones básicas de las redes.....	10
3.2 Procesamiento de los datos.....	11
3.3 Optimización Multiobjetivo	15
3.4 Algoritmo genético	18
4. Técnicas y herramientas.....	24
4.1 Metodologías.....	24
4.2 Gestión de proyectos	24
4.3 Control de versiones	24
4.4 Gestión del repositorio	24
4.5 Entorno de desarrollo	24
4.6 Lenguaje de programación	25
4.7 Base de datos	26
4.8 Documentación	26
4.9 Otras herramientas utilizadas	27
5. Aspectos relevantes del desarrollo del proyecto	28
5.1 Formación	28

5.2 Inicio del proyecto	29
5.3 Problemas en el desarrollo	30
5.4 Despliegue de la aplicación.....	31
5.5 Análisis de la calidad de código	33
6. Trabajos relacionados.....	36
7. Conclusiones y Líneas de trabajo futuras.....	37
7.1 Conclusiones.....	37
7.2 Líneas de trabajo futuras	40
Bibliografía	42

Índice de Ilustraciones

Ilustración 1: Red de estudiantes visualizada en Gephi.....	11
Ilustración 2: Red de clases dibujada por NetworkX.....	12
Ilustración 3: Imagen del grafo de clases resultado	14
Ilustración 4 [4]: Ejemplo gráfico del frente de Pareto.....	17
Ilustración 5: Frente de Pareto con tres objetivos.....	17
Ilustración 6 [8]: Cruce de punto único	20
Ilustración 7 [8]: Cruce de dos puntos.....	20
Ilustración 8 [9]: Procedimiento del NSGA-II.....	21
Ilustración 9 [9]: Crowding Distance.	22
Ilustración 10: Soluciones del primer ejemplo de prueba	30
Ilustración 11: Bug en el frente de Pareto	31
Ilustración 12: Error en el despliegue por la base de datos	32
Ilustración 13: Análisis inicial SonarCloud	34
Ilustración 14: Análisis final SonarCloud.....	35

Índice de Tablas

Tabla 1: Matriz de hermanos	12
Tabla 2: Asignación de estudiantes	15
Tabla 3: Pares de clases conectadas	15

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 juntaban 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, tienen 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 la aplicación sea una herramienta valiosa para la prevención y control de diversas infecciones en cualquier situación epidemiológica.

1.1 Estructura de la memoria

Introducción. En este capítulo se presenta una descripción general del proyecto, explicando su contexto, importancia y los motivos que llevaron a su realización.

Objetivos del proyecto. Se detallan los objetivos generales, técnicos y personales que guiaron el desarrollo del proyecto.

Conceptos teóricos. En este apartado se exploran y desarrollan los conceptos teóricos fundamentales necesarios para la comprensión del proyecto.

Técnicas y herramientas. Se presenta un listado detallado de las principales herramientas, frameworks y tecnologías utilizadas durante el desarrollo del proyecto.

Aspectos relevantes del desarrollo del proyecto. En este apartado se destacan los eventos más significativos durante el proceso de desarrollo.

Trabajos relacionados. Se analiza el contexto actual de trabajos previos relacionados con el tema del proyecto.

Conclusiones y líneas de trabajo futuras. Se exponen las conclusiones derivadas del proyecto. Además, se identifican posibles líneas de trabajo futuras para mejorar el proyecto.

1.2 Materiales adjuntos

- Link de la web: <https://sire-ddeac2c1782a.herokuapp.com/>
- Link del repositorio: <https://github.com/rubenarasti/Sibling-Rewiring>

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.
- Organizar y planificar el proyecto utilizando Trello y GitHub como herramientas de gestión 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** 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.).

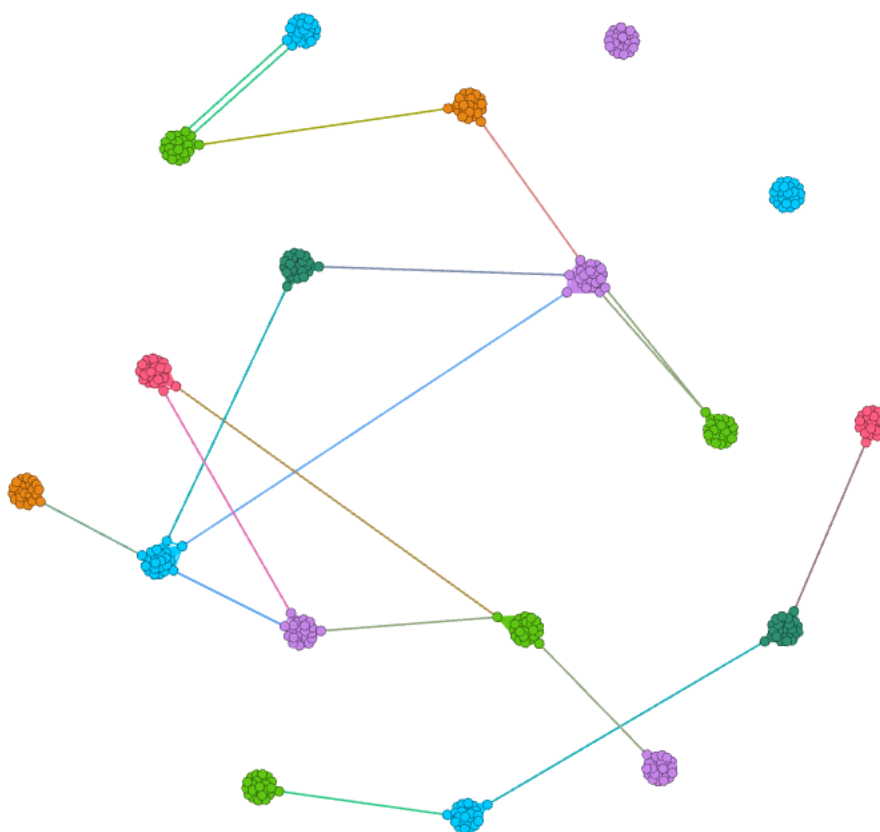


Ilustración 1: Red de estudiantes visualizada en Gephi

En la red de estudiantes de la ilustración 1 aparecen los estudiantes agrupados por clases y coloreados según su curso. La red incluye dos grupos distintos por curso en las etapas infantil y primaria y se pueden ver fácilmente las relaciones entre estudiantes de diferentes clases.

Conceptos teóricos

Para identificar quién es hermano de quién y diferenciar las relaciones de hermandad en la red de estudiantes, se utilizará una tabla para definir una matriz que incluirá los atributos de cada estudiante junto con el nombre de su hermano en cada fila. Esto 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, se define 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.

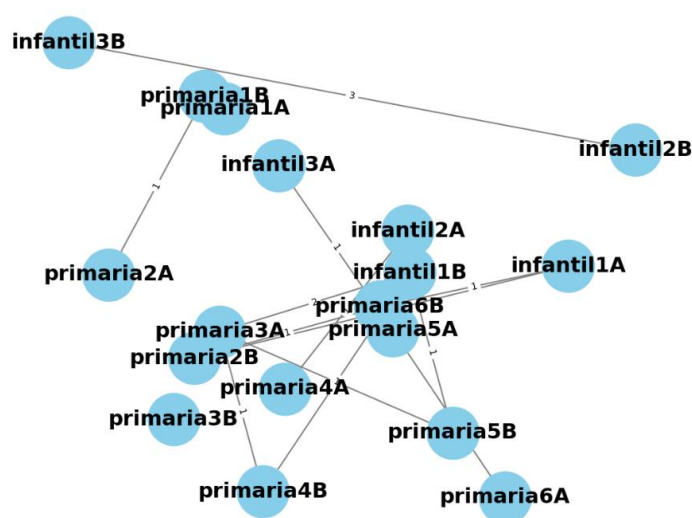


Ilustración 2: Red de clases dibujada por NetworkX

Conceptos teóricos

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).

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, ya que es necesario 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 explicada 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 con peso 1 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.

Conceptos teóricos

El archivo en grafo_clases.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_clases.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.

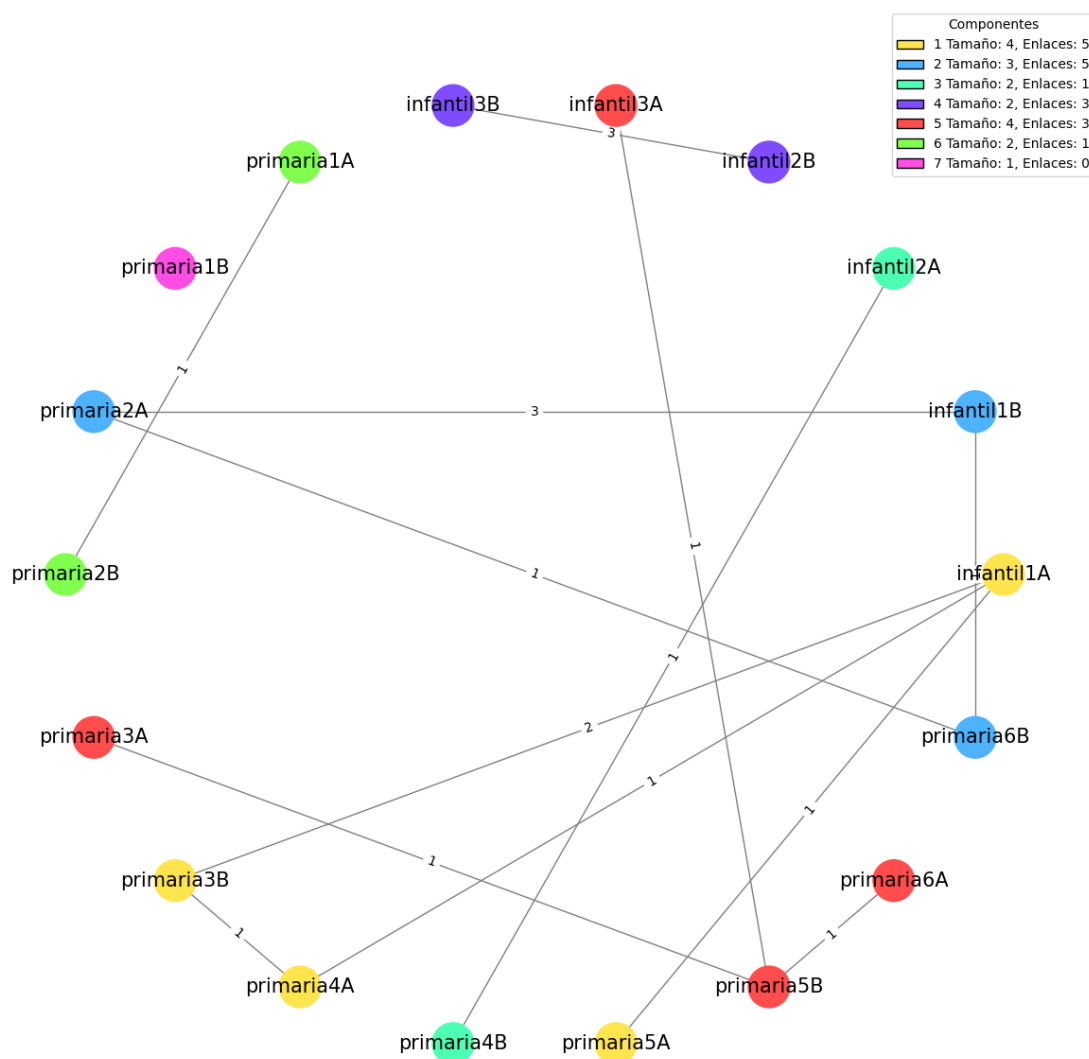


Ilustración 3: Imagen del grafo de clases resultado

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.

Conceptos teóricos

Tabla 2: Asignación de estudiantes

Nombre	Etapas	Curso	Clase
0	infantil	1	A
1	infantil	1	B

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 identificar qué clases son las que se ven afectadas directamente. Por ejemplo, si se ha contagiado el estudiante en una de las clases listadas en el archivo, las clases conectadas mediante dicho estudiante podrían necesitar ser notificadas o evaluadas para tomar medidas preventivas adecuadas.

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 el 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 se necesita 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.

Conceptos teóricos

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

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

1. La solución x_i es igual o mejor que x_j para todas las funciones objetivo
2. La solución x_i es estrictamente mejor que x_j en al menos un objetivo

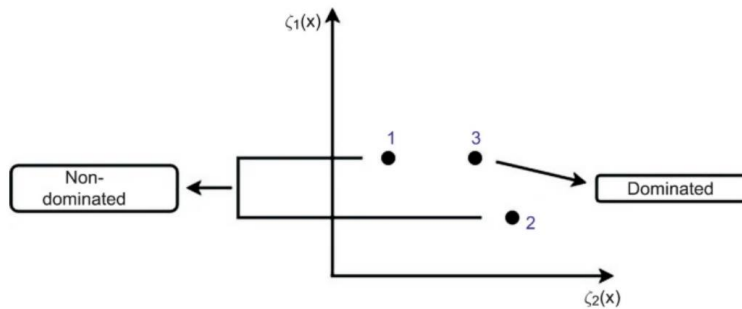


Ilustración 4 [4]: Ejemplo gráfico del frente de Pareto

En la ilustración 1 se muestra una gráfica donde los ejes representan los valores de dos funciones objetivo $\zeta_1(x)$ y $\zeta_2(x)$. 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:

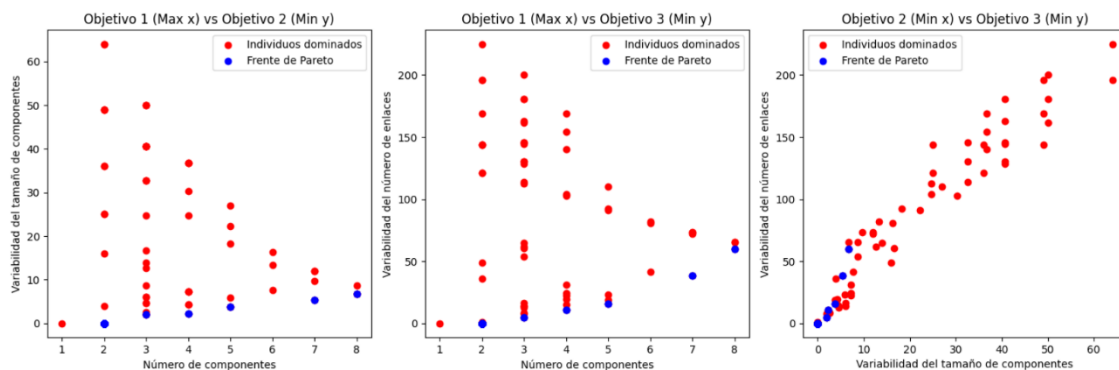


Ilustración 5: Frente de Pareto con tres objetivos

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

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 el cruce y la 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.
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, se 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 se habla de un gen, se hace referencia 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.

Conceptos teóricos

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 G^N 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.[8]

El **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.

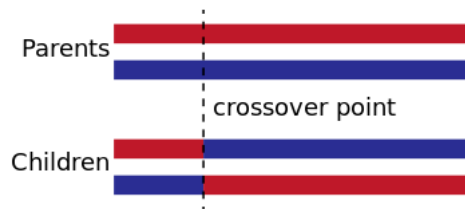


Ilustración 6 [8]: Cruce de punto único

En 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.

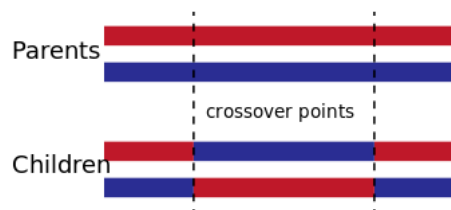


Ilustración 7 [8]: Cruce de dos puntos

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 tres métodos de cruce porque son los que se conocían 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 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

El operador de selección determina qué individuos de la población actual se reproducirán para crear la siguiente generación. Generalmente hay dos tipos de enfoques para la selección de los AG multiobjetivo: los basados en Pareto y los basados en indicadores. Los basados en Pareto, como NSGA-II, SPEA2 y MOEA/D, utilizan la dominancia de Pareto para seleccionar soluciones no dominadas que forman el frente de Pareto. Por otro lado, los algoritmos basados en indicadores, como MOEA/D-IBEA y NSGA-III, emplean funciones indicadoras como el hipervolumen para evaluar la calidad de las soluciones. Estos algoritmos son útiles para problemas con múltiples objetivos donde los métodos basados en Pareto pueden enfrentar dificultades debido a la dimensionalidad alta. [4]

Para nuestra selección, se utilizará el NSGA-II (Nondominated Sorting Genetic Algorithm II) [9], ya que es sencillo de implementar y nuestro problema no tiene una alta dimensionalidad.

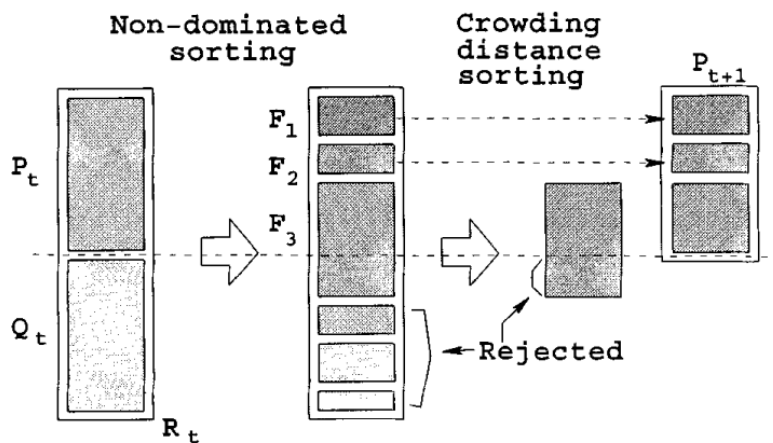


Ilustración 8 [9]: Procedimiento del NSGA-II

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

1. A partir de la población actual (P_t) se produce la descendencia (Q_t).
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 (F_1) contiene soluciones no dominadas por ninguna otra en la población, mientras que los frentes subsiguientes (F_2, F_3, \dots) contienen soluciones dominadas por una, dos o más soluciones de los frentes anteriores.

3. En 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. En la ilustración 9 se puede ver cómo se calcula. Es el perímetro del cuboide formado por los vértices de sus soluciones vecinas. Los puntos rellenos son soluciones del mismo frente.

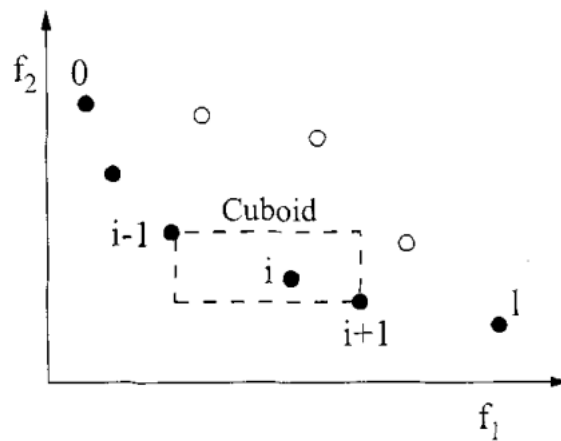


Ilustración 9 [9]: Crowding Distance.

La población obtenida del corte pasa a ser la población actual (P_{t+1}) 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. [10]

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** p_c 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** p_m 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 p_c suelen estar entre 0.5 y 1.0 y los valores de p_m suelen estar entre 0.005 y 0.05. [5]

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

4. Técnicas y herramientas

4.1 Metodologías

Se ha elegido la metodología Ágil, específicamente **SCRUM** [11], para gestionar el desarrollo del proyecto. SCRUM permite una gestión flexible y adaptable del proyecto, facilitando la respuesta rápida a los cambios y la incorporación de mejoras continuas. Esta metodología se basa en la realización de entregas iterativas e incrementales llamados sprints, lo que permite revisar y ajustar el desarrollo de forma constante, asegurando que el proyecto se mantenga alineado con los objetivos y expectativas del usuario.

4.2 Gestión de proyectos

Se ha optado por utilizar **Trello** [12] como herramienta principal de gestión de proyectos debido a su interfaz intuitiva y capacidad para organizar tareas de manera visual. Además, se emplean las issues de **GitHub** [13] para asignar y gestionar las tareas específicas del proyecto. Desde Trello se enlazan las issues de GitHub a las tareas correspondientes.

4.3 Control de versiones

Para el control de versiones, se utiliza **Git** [14]. Git es una herramienta de control de versiones distribuido que permite a los desarrolladores gestionar y registrar los cambios en el código fuente de manera eficiente. Ofrece funcionalidades como ramas (branches), fusiones (merges) y la posibilidad de trabajar en equipo sin necesidad de una conexión constante a un servidor central. La elección de Git se basa en su amplia adopción en la industria y a la familiaridad obtenida a lo largo de la carrera.

4.4 Gestión del repositorio

Para la gestión del repositorio se ha utilizado **GitHub** [13]. Esta plataforma no solo facilita el alojamiento y la organización de repositorios, sino que también proporciona herramientas para una colaboración eficaz entre los desarrolladores. GitHub permite la creación de pull requests y revisiones de código, además de ofrecer integración con otras herramientas de desarrollo, como aquellas dedicadas al despliegue y la calidad del código. Además, GitHub incluye funcionalidades de seguimiento, como las issues ya comentadas, que facilitan una gestión más eficiente de las tareas del equipo.

4.5 Entorno de desarrollo

Se ha optado por utilizar **Visual Studio Code** [15] como el entorno de desarrollo integrado (IDE) principal. Visual Studio Code es un editor de código fuente gratuito desarrollado por Microsoft que se ha convertido en una opción popular debido a su versatilidad y extensibilidad. Ofrece soporte para la depuración, control de versiones y una gran

variedad de lenguajes de programación a través de extensiones. Visual Studio Code es especialmente popular en sistemas operativos Windows, donde se utiliza como el principal IDE debido a su rica funcionalidad y soporte activo de la comunidad.

Para la edición de código en entornos Linux, especialmente durante la configuración de contenedores Docker, se ha empleado **Gedit** [16]. Gedit es un editor de texto ligero y versátil que proporciona soporte para la sintaxis de múltiples lenguajes de programación mediante la coloración de código, lo que facilita la visualización y edición del código fuente.

4.6 Lenguaje de programación

El proyecto está principalmente en **Python** [17], un lenguaje de programación conocido por su simplicidad y legibilidad. Python es ampliamente utilizado en el desarrollo web, análisis de datos, inteligencia artificial y más, debido a su sintaxis clara y la amplia gama de bibliotecas disponibles. Entre los frameworks y bibliotecas utilizados en este proyecto se incluyen:

Flask. Un framework de desarrollo web en Python que facilita la creación de aplicaciones web robustas con una estructura mínima. Flask es ideal para proyectos que requieren flexibilidad y control total sobre la arquitectura de la aplicación. Está basado en la especificación WSGI de Werkzeug [18] y el motor de templates Jinja2 [19] y tiene una licencia BSD.[20]

Gunicorn [21]. Un servidor WSGI HTTP para aplicaciones web Python que permite manejar múltiples solicitudes simultáneamente, mejorando la escalabilidad y el rendimiento de la aplicación.

Mysql-connector [22]. Una biblioteca que permite conectar y ejecutar consultas en bases de datos MySQL desde Python, facilitando la interacción con la base de datos.

Flask-Bootstrap [23]. Es una extensión de Flask que integra de manera eficiente el framework de CSS **Bootstrap** [24] en aplicaciones Flask. Bootstrap es un conjunto de herramientas de código abierto para desarrollar con HTML, CSS y JS. Bootstrap facilita la creación de interfaces de usuario modernas y responsive, mejorando significativamente la apariencia y la funcionalidad de las aplicaciones web.

NetworkX [25]. Una biblioteca para la creación, manipulación y estudio de la estructura y dinámica de grafos complejos, utilizada para la representación y análisis de redes.

Pymoo [26]. Una biblioteca que proporciona algoritmos de optimización multiobjetivo, útil para encontrar soluciones óptimas en problemas complejos con múltiples criterios.

Numpy [27] y **Pandas** [28]. Bibliotecas fundamentales para el análisis y manipulación de datos, que proporcionan estructuras de datos y funciones de alto rendimiento para realizar cálculos numéricos y análisis de datos.

Matplotlib [29]. Una biblioteca para la generación de gráficos y visualizaciones de datos, que facilita la creación de figuras complejas de manera sencilla.

Para la creación de las interfaces web se ha utilizado el lenguaje **HTML** [30]. HTML (o HyperText Markup Language) es el lenguaje estándar utilizado para estructurar y presentar contenido en la web. Define la estructura básica de una página web mediante etiquetas que describen diferentes elementos como encabezados, párrafos, listas, enlaces, imágenes, formularios, entre otros. Esta tecnología es fundamental para desarrollar interfaces de usuario accesibles y compatibles con diversos navegadores y dispositivos.

4.7 Base de datos

Para la gestión de la base de datos, se ha utilizado **MySQL** [31], un sistema de gestión de bases de datos relacional conocido por su fiabilidad y capacidad para manejar grandes volúmenes de datos. MySQL permite la creación de bases de datos y la ejecución de consultas complejas de manera eficiente. Esta elección se basa en la continuidad del uso de MySQL desde el proyecto anterior, del cual deriva el actual.

Para el despliegue en producción, se ha utilizado **JawsDB MySQL** [32] proporcionado por Heroku. JawsDB es un servicio gratuito de base de datos en la nube que ofrece Heroku, facilitando la gestión de bases de datos MySQL en un entorno de nube.

4.8 Documentación

La herramienta elegida para la documentación del proyecto es **Microsoft Word** [33]. Este editor de texto se ha seleccionado debido a su facilidad de uso y la capacidad para gestionar revisiones de manera eficaz. También se consideró LaTeX [34], pero debido a la falta de familiaridad con la herramienta y a la escasa necesidad de usar fórmulas matemáticas se descartó.

La fuente tipográfica escogida es **EB Garamond** [35], con licencia SIL Open Font License [15]. Esta fuente ofrece claridad y legibilidad, gracias a sus serifas, lo que mejora la lectura y la presentación de la documentación del proyecto.

Para generar diagramas técnicos, como diagramas de casos de uso o de diseño procedimental se utilizó **Draw.io** [36]. Esta herramienta en línea permite generar todo tipo de diagramas técnicos de manera intuitiva, facilitando la visualización y comprensión de la estructura y el funcionamiento del sistema.

En cuanto a la visualización de grafos, se ha empleado **Gephi** [37]. Esta plataforma de software libre es ideal para la exploración y análisis de redes complejas, proporcionando gráficos interactivos que ayudan a entender las relaciones y estructuras de datos, como las conexiones entre estudiantes y clases en este proyecto.

4.9 Otras herramientas utilizadas

Zotero [38]. Utilizado para la gestión de la bibliografía. Zotero es una herramienta de software libre que permite recopilar, organizar y citar referencias bibliográficas, facilitando la creación de bibliografías y la organización de fuentes de información.

Docker [39] y **Docker Compose** [40]. Utilizados para crear entornos de desarrollo aislados y replicables. Docker permite empaquetar aplicaciones y sus dependencias en contenedores, asegurando que se ejecuten de manera consistente en cualquier entorno. Docker Compose se utiliza para definir y administrar aplicaciones multicontenedor, permitiendo así lanzar el servidor y la base de datos a la vez.

Heroku [41]. Empleado para el despliegue de la aplicación en la nube. Heroku es una plataforma como servicio (PaaS) que permite a los desarrolladores desplegar, gestionar y escalar aplicaciones. Ofrece integración sencilla con GitHub para la implementación continua y facilita la gestión de la infraestructura. El costo aproximado es de 0.01 \$ por hora, con un límite de 7 \$ al mes en su plan básico. Antes de optar por Heroku, se evaluaron otras alternativas como Render [42], Koyeb [43] y Code Capsules [44], aunque no se obtuvieron los resultados deseados. Estos intentos y sus razones para no ser seleccionados se explican en detalle en la sección de aspectos relevantes del proyecto.

SonarCloud [45]. Utilizado para el análisis de la calidad del código. SonarCloud es un servicio en la nube que proporciona análisis estático de código, ofreciendo métricas y recomendaciones para mejorar la mantenibilidad, seguridad y calidad del código fuente. Permite identificar vulnerabilidades, duplicaciones y otros problemas potenciales en el código, contribuyendo a mantener altos estándares de calidad en el desarrollo del software.

5. Aspectos relevantes del desarrollo del proyecto

5.1 Formación

Durante el desarrollo de este proyecto, se ha logrado una significativa ampliación de conocimientos en varias áreas clave, tanto técnicas como metodológicas, que han sido fundamentales para la culminación exitosa del mismo. A continuación, se detallan las principales áreas de aprendizaje y cómo han contribuido al proyecto:

Ampliación de conocimientos de Flask

Se profundizó en el uso del framework Flask, ya introducido en la asignatura de “Diseño y Mantenimiento del Software”. Durante el desarrollo del proyecto, se extendieron estos conocimientos para comprender y manejar aspectos avanzados de Flask, como la implementación de descargas en una web, la integración con bases de datos o el lanzamiento de un servidor WSGI con gunicorn.

Optimización multiobjetivo y Algoritmo Genético

En la asignatura “Computación Neuronal y Evolutiva” se introdujeron conceptos básicos de optimización multiobjetivo y algoritmos evolutivos. Sin embargo, el conocimiento del algoritmo NSGA-II era bastante superficial. Inicialmente, se usó la biblioteca DEAP [46], pero debido a limitaciones específicas, se decidió cambiar a pymoo, un framework más robusto y especializado para optimización multiobjetivo, lo que implicó una curva de aprendizaje adicional y la adquisición de nuevas habilidades para manejar esta herramienta.

Bootstrap y HTML

Para mejorar la interfaz de usuario y la presentación de datos, fue necesario adquirir conocimientos básicos en Bootstrap y HTML. Bootstrap, un popular framework de CSS, se utilizó para crear una interfaz responsiva y moderna, mientras que HTML se usó para estructurar y presentar contenido web de manera efectiva.

Docker y Docker Compose

La necesidad de crear entornos de prueba estables y reproducibles llevó a la familiarización con Docker y Docker Compose. Estas herramientas fueron esenciales para la creación de contenedores que replicaran el entorno de producción de la aplicación.

Despliegue en Heroku

Finalmente, se adquirieron conocimientos sobre cómo desplegar una aplicación web en Heroku. Este aprendizaje incluyó la configuración del CLI de Heroku [47], la vinculación del repositorio de GitHub con el de Heroku, la gestión de bases de datos en la nube, y la resolución de problemas comunes asociados con el despliegue de aplicaciones en producción que se comentarán más adelante en este apartado.

5.2 Inicio del proyecto

El proyecto fue propuesto por la Universidad de Burgos (UBU) a través de una idea de uno de los tutores, José Manuel Galán Ordax. La alumna María Ojeda Ruiz fue la encargada de llevar a cabo el desarrollo del proyecto. Sus responsabilidades incluyeron la implementación de los datos de entrada, la creación y gestión de la base de datos, el diseño de la web, el desarrollo de la aplicación en Flask, la implementación de la funcionalidad de subida de ficheros, la creación de redes aleatorias y la aplicación del recocido simulado para la organización de los hermanos en las clases.

Continuación del Trabajo Anterior

Inicialmente, se consideró la opción de comenzar el proyecto desde cero. Sin embargo, debido a los retrasos en el inicio, se decidió continuar con el trabajo realizado previamente. La primera tarea consistió en intentar ejecutar la aplicación existente en el sistema de desarrollo actual.

Problemas Iniciales y Solución

Se encontraron dificultades para iniciar la aplicación anterior, principalmente debido a la falta de una base de datos y a un conocimiento insuficiente sobre el funcionamiento de la aplicación. La situación se resolvió al obtener una imagen de la máquina virtual, proporcionada durante una reunión con los tutores. Esta imagen contenía un script que especificaba los datos necesarios para configurar correctamente la base de datos en el archivo `database.py`.

Estructura del Proyecto

La estructura del proyecto se heredó del trabajo anterior, utilizando el patrón Modelo-Vista-Controlador (MVC). Este patrón separa los archivos de vista (HTML), los archivos de controlador (implementación del servidor en Flask), y los archivos de modelo (lógica de negocio y datos).

Estructura del Algoritmo Genético

La estructura del código del algoritmo genético se basó en un código proporcionado en la asignatura "Computación Neuronal y Evolutiva" por el profesor de la UBU Bruno Baruque. [6] Esta estructura promueve un alto nivel de desacoplamiento mediante la separación de diferentes responsabilidades en archivos específicos:

Aspectos relevantes del desarrollo del proyecto

global_def.py: Definición de variables globales.

data_management.py: Gestión de datos.

individual_evaluation.py: Operaciones relacionadas con la evaluación del genotipo.

ga_function.py: Configuración y definición del algoritmo genético (en el proyecto original, denominado `algorithm_config.py`).

Evolución de los objetivos a optimizar

En la primera iteración del algoritmo genético, se implementaron los criterios de evaluación de la aplicación anterior. Durante una reunión con los tutores, se determinó que estos criterios eran innecesariamente complejos. Por ello, se simplificaron a dos objetivos nuevos: maximizar el número de componentes y minimizar la variabilidad del tamaño de los componentes.

Durante el desarrollo, el tutor José Manuel observó la necesidad de incorporar un tercer objetivo: la variabilidad en el número de enlaces dentro de los componentes. Esto se debe a que componentes de igual tamaño pero con diferentes números de hermanos presentan riesgos distintos.

5.3 Problemas en el desarrollo

Falta de ejemplos de prueba

Uno de los principales desafíos encontrados fue la escasez de ejemplos de prueba para los datos de entrada del algoritmo. El único ejemplo disponible generaba únicamente dos posibles soluciones, lo que no permitía visualizar de manera efectiva las ventajas de la implementación multiobjetivo.

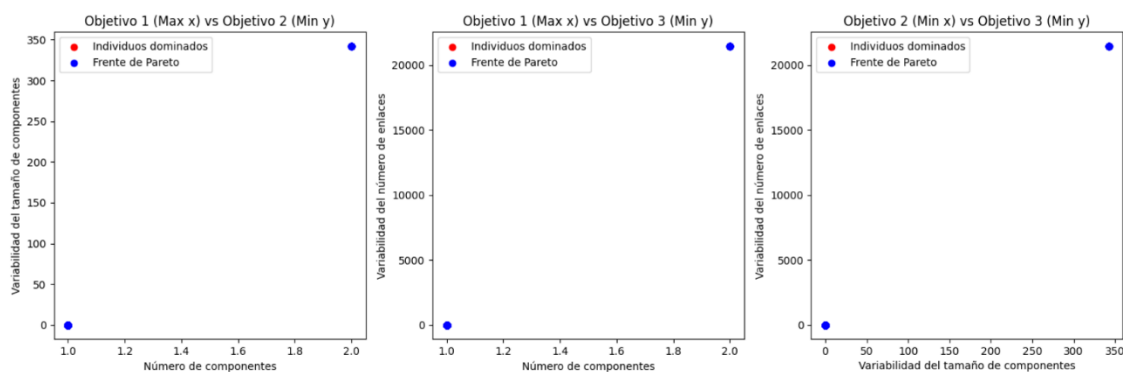


Ilustración 10: Soluciones del primer ejemplo de prueba

Esto se debía a que el espacio de búsqueda era extremadamente amplio, con tres grupos y 526 hermanos, resultando en aproximadamente 10^{249} posibles soluciones. A pesar de que el algoritmo se ejecutaba durante dos horas, no lograba encontrar más de dos soluciones.

Aspectos relevantes del desarrollo del proyecto

La función del proyecto anterior para crear datos aleatorios no funcionaba correctamente, ya que la función que generaba la tabla de hermanos no añadía la columna “hermano de”, crucial para identificar los enlaces en el problema. Se corrigió la función para permitir la creación de casos de prueba con espacios de búsqueda más manejables y así comprobar el correcto funcionamiento del algoritmo.

Además, se realizaron varias mejoras en la función `create_initial_network` para generar la red aleatoria ya que, originalmente, esta función no permitía hermanos múltiples y no creaba todos los pares de enlaces solicitados si los alumnos ya estaban en la misma clase.

Error en el frente de Pareto

Se encontraron problemas significativos con la implementación anterior del frente de Pareto utilizando la biblioteca DEAP. La implementación de su NSGA-II no mantenía la diversidad de las soluciones y a veces calculaba incorrectamente el frente de Pareto, lo que generaba errores en la evaluación de las soluciones óptimas.

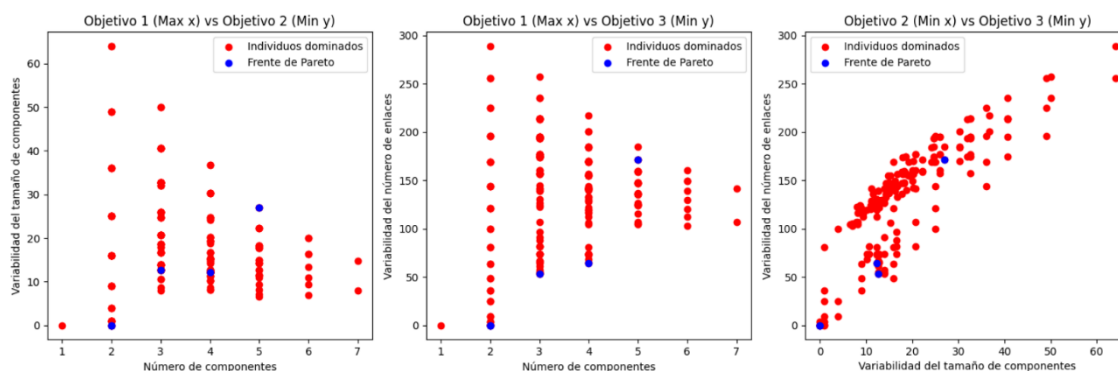


Ilustración 11: Bug en el frente de Pareto

Para resolver los problemas encontrados con DEAP, se decidió migrar a Pymoo con el objetivo de evaluar si esta herramienta manejaba el frente de Pareto de manera más precisa. Pymoo demostró ser efectiva, ya que en todas las generaciones de soluciones calculó correctamente el frente de Pareto, lo que nos lleva a concluir que definitivamente había un error en la implementación del NSGA-II en DEAP. La migración fue relativamente sencilla debido a la estructura desacoplada del proyecto, que separaba las funciones de manejo de datos y evaluación en diferentes archivos. Esto permitió que el cambio a Pymoo se enfocara principalmente en ajustar la configuración del algoritmo genético.

5.4 Despliegue de la aplicación

Creación de imagen de Docker Compose

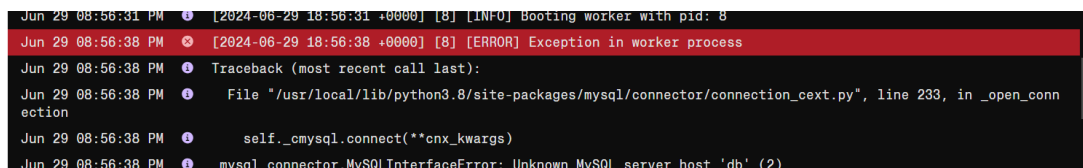
Se creó una imagen de Docker Compose con la idea de desplegarlo en un servidor y para la creación de entornos de prueba consistentes. Al intentar ejecutar la imagen en otra máquina, surgió un problema donde la aplicación no esperaba a la base de datos, a pesar de incluir la

Aspectos relevantes del desarrollo del proyecto

directiva "depends-on". Este problema se solucionó configurando el contenedor de la aplicación para que reinicie en caso de fallo.

Plataformas donde fallaron los intentos de despliegue

Se intentó subir la imagen de Docker a **Render** [42] y **Koyeb** [43], pero ambos servicios arrojaron un error de "unknown MySQL server".



```
Jun 29 08:56:31 PM [2024-06-29 18:56:31 +0000] [8] [INFO] Booting worker with pid: 8
Jun 29 08:56:38 PM [2024-06-29 18:56:38 +0000] [8] [ERROR] Exception in worker process
Jun 29 08:56:38 PM Traceback (most recent call last):
Jun 29 08:56:38 PM   File "/usr/local/lib/python3.8/site-packages/mysql/connector/connection_cext.py", line 233, in _open_connection
Jun 29 08:56:38 PM     self._cmysql.connect(**cnx_kwargs)
Jun 29 08:56:38 PM   _mysql_connector.MySQLInterfaceError: Unknown MySQL server host 'db' (2)
```

Ilustración 12: Error en el despliegue por la base de datos

Se concluyó que estos servicios solo permitían usar un contenedor por máquina, lo que planteó la opción de desplegar la base de datos y la aplicación en contenedores separados en dos servidores distintos, aunque se descartó por su complejidad.

También se intentó desplegar en **Code Capsules** [44], pero el intento fracasó debido a problemas para configurar correctamente las dependencias al subir los archivos.

Despliegue en Heroku

Se optó por Heroku debido a intentos previos en el proyecto anterior. Inicialmente, surgió un problema al configurar el repositorio en Heroku, ya que no se accedía correctamente a la carpeta de ejecución. La solución fue agregar un Build pack llamado "subdir-heroku-buildpack" [48] para cambiar al subdirectorio adecuado, requerido con la variable de entorno "PROJECT_PATH".

Inicialmente, se intentó subir la imagen de Docker Compose a Heroku, obteniendo resultados similares a los encontrados en la plataforma Render. Al no tener éxito con esta estrategia, se decidió utilizar la herramienta de línea de comandos de Heroku (Heroku CLI) para realizar el despliegue. Con este nuevo enfoque, se presentaron varios problemas, que se detallarán a continuación.

El primer problema que apareció en este despliegue decía que no se encontraba el comando gunicorn. El problema era que no se habían instalado los requirements porque no se había añadido el Build pack de Python a Heroku.

Otro problema crítico fue la incompatibilidad y errores de dependencias en los requirements. Estos errores impedían que las dependencias se instalaran correctamente, lo que causaba fallos en la ejecución de la aplicación. La solución fue actualizar la versión de Python a 3.10, lo que no solo resolvió los problemas de dependencia sino que también corrigió un error en la biblioteca NetworkX, que impedía la correcta visualización de los pesos de los autoenlaces en las gráficas generadas por la aplicación.

Aspectos relevantes del desarrollo del proyecto

Una vez corregidos estos dos primeros errores la aplicación se desplegó correctamente y se paró el despliegue para que no consumiese tiempo de uso.

Se encontró un error al reiniciar el despliegue de la aplicación con el procedimiento `sp_createUser`. Los registros de Heroku decían que la tabla ya existía y que se estaba intentando crear. Se solucionó moviendo la línea “`DROP PROCEDURE IF EXISTS sp_createUser;`” a el archivo `procedure.sql`.

Al probar otras funcionalidades del proyecto en un entorno local, se detectaron varios problemas. La función destinada a descargar todas las soluciones generaba archivos vacíos en lugar de contenido válido. Además, en ocasiones, la ejecución del algoritmo se interrumpía, mostrando un mensaje de error que indicaba que el grafo inicial no estaba definido. Se identificó que Gunicorn, el servidor WSGI utilizado, emplea varios procesos de ejecución conocidos como workers. Por defecto, estos workers tienen un tiempo de espera (timeout) de 30 segundos, después del cual son reemplazados si no han completado su tarea. Para resolver este inconveniente, se modificó el tiempo de espera de los workers a 86400 segundos (un día) utilizando la opción `--timeout 86400`.

Sin embargo, al desplegar la aplicación en Heroku, surgió un nuevo problema relacionado con los timeouts. A pesar de la configuración de los nuevos tiempos de espera en el entorno local, en Heroku todas las solicitudes seguían siendo interrumpidas después de 30 segundos. Heroku devolverá una página de error y figurará en los logs el error H12, de timeout de la petición. Al contrario que con Gunicorn, este tiempo de espera no puede modificarse.[49]

Dado que no se encontraba una solución viable sin incurrir en costos adicionales, se decidió mantener el despliegue en ese estado. Esto implica que, al intentar descargar una solución, en algunas ocasiones se obtiene un archivo vacío. Reintentando la descarga varias veces, eventualmente se consigue el archivo completo, lo que sugiere que el problema podría estar relacionado con la forma en que los workers manejan las solicitudes. En la mayoría de los casos, la opción de descargar todas las soluciones tarda tanto tiempo que termina generando archivos vacíos.

5.5 Análisis de la calidad de código

SonarCloud se integró en el flujo de trabajo del proyecto para evaluar la calidad del código de manera continua. Al inicio del análisis, se identificaron varias áreas problemáticas que requerían atención:

Aspectos relevantes del desarrollo del proyecto

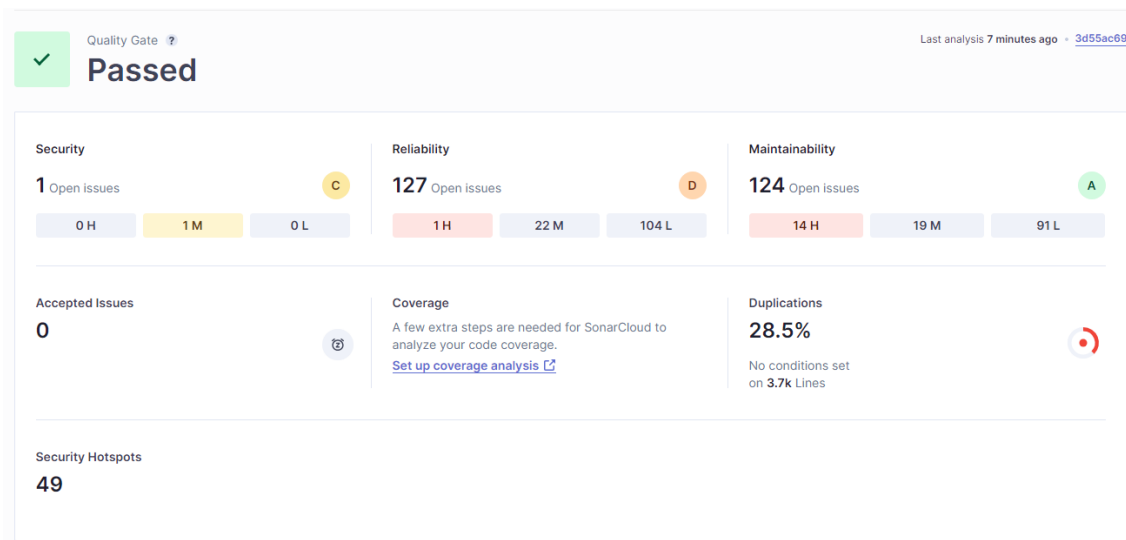


Ilustración 13: Análisis inicial SonarCloud

El análisis inicial indicaba problemas graves en la fiabilidad (Reliability) y en la mantenibilidad del código. Además, había un 28,5 % de código duplicado y 49 piezas de código con vulnerabilidades (Security Hotspots).

La barra de navegación se encontraba replicada en múltiples archivos HTML, lo que generaba una alta tasa de duplicidad. Para abordar esto, se modularizó la barra de navegación, permitiendo su reutilización sin replicar el código en cada archivo. Esto redujo la duplicidad de código significativamente, bajando al 5%.

Se detectaron varios problemas relacionados con el manejo de variables y bucles sin límite claro, lo que podría causar bucles infinitos. Para resolverlo, se mejoró la gestión de las variables y se establecieron límites precisos para los bucles, asegurando que todos los ciclos tuvieran condiciones de parada bien definidas. Estas mejoras contribuyeron a solucionar la mayoría de los problemas de calidad reportados por SonarCloud.

Se identificó que varios de los errores más graves relacionados con la mantenibilidad del código estaban concentrados en funciones que contenían un número excesivo de líneas de código. Aunque se valoró la posibilidad de dividir estas funciones en bloques más pequeños y manejables para mejorar la legibilidad y mantenibilidad, se decidió no hacerlo debido al tiempo significativo que requeriría realizar esos cambios. Esta decisión se basó en la evaluación de costos y beneficios a corto plazo, priorizando otros aspectos críticos del proyecto.

Aspectos relevantes del desarrollo del proyecto

El análisis final quedó así:

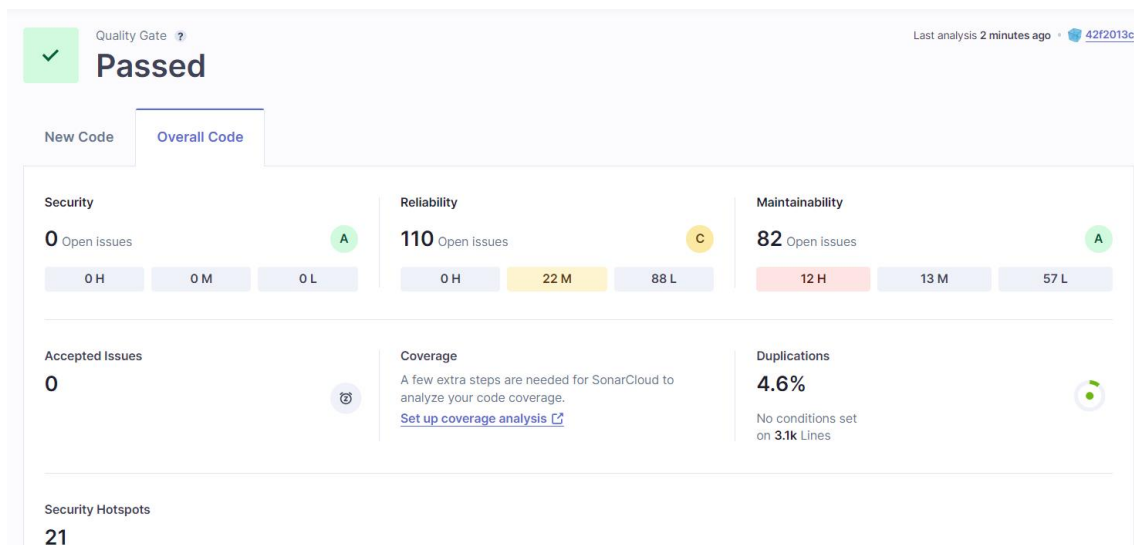


Ilustración 14: Análisis final SonarCloud

Se han reducido los problemas de vulnerabilidades en el código de 49 a 21, los problemas de fiabilidad de 127 a 120, los problemas de mantenibilidad de 124 a 82 y la duplicidad del código de un 28% a un 5%.

6. Trabajos relacionados

Este proyecto se sustenta en el trabajo anterior de la alumna María Ojeda Ruiz [2], quien exploró la implementación de algoritmos y modelos para la gestión de contagios en redes. En su revisión de la literatura, no se encontraron estudios que utilizaran una metodología y enfoque similares a los empleados en su proyecto.

De manera similar, en este trabajo tampoco se identificaron investigaciones previas con un enfoque y metodología coincidentes. Se encontró un estudio relacionado titulado "Benders decomposition algorithms for minimizing the spread of harmful contagions in networks" [50]; sin embargo, su enfoque y metodología difieren significativamente de los utilizados aquí, destacando la originalidad y novedad de este trabajo.

7. Conclusiones y Líneas de trabajo futuras

7.1 Conclusiones

En este apartado se expondrán las conclusiones relacionadas con los objetivos del proyecto y el nivel de satisfacción alcanzado. Las conclusiones se dividen en tres categorías: generales, técnicas y personales, cada una reflejando diferentes aspectos del proceso de desarrollo y resultados del proyecto.

Conclusiones generales

Se ha logrado desarrollar una aplicación web para minimizar el riesgo de contagio en un entorno escolar. La aplicación considera múltiples objetivos y presenta diversas soluciones que equilibran diferentes compromisos entre estos objetivos, permitiendo al usuario seleccionar la opción más adecuada según sus necesidades específicas.

La implementación del algoritmo genético no considera la factibilidad de las soluciones durante su ejecución, evaluándola únicamente al momento de la descarga. Sin embargo, dado que la probabilidad de exceder la capacidad de las clases es generalmente baja, se puede concluir que la implementación del algoritmo cumple satisfactoriamente con el objetivo inicial.

La aplicación no solo permite la organización óptima de los estudiantes, sino que también facilita la visualización gráfica de las soluciones generadas en color rojo y destacando el frente de Pareto en azul. Esta funcionalidad proporciona a los usuarios una visión clara de la exploración del algoritmo, ayudando al usuario a comprender qué parámetros escoger para conseguir mejores resultados. Una implementación más intuitiva podría haber permitido a los usuarios hacer clic en los puntos azules para que se muestren las soluciones asociadas a esas evaluaciones específicas.

Los archivos de solución que se proporcionan contienen información clave para la toma de decisiones y, en su mayoría, son fáciles de interpretar:

- Archivo `par_clases.csv`: Permite identificar las clases afectadas por conexiones entre hermanos y especifica cuáles son los hermanos que conectan estas clases. Esto es crucial para entender cómo se distribuyen el contagio entre los estudiantes y facilita la toma de decisiones para minimizar el riesgo de contagio.
- Archivo `estudiantes.csv`: Muestra la asignación de cada estudiante a su respectiva clase, proporcionando una visión clara de la distribución estudiantil.
- Archivo `grafo_clases.gexf`: Este archivo permite a los usuarios avanzados generar métricas adicionales, si lo consideran necesario. Ofrece flexibilidad para el análisis detallado de las interacciones y la estructura de la red de clases, facilitando un enfoque personalizado según las necesidades específicas del usuario.

Conclusiones y Líneas de trabajo futuras

- Archivo grafo_clases.png: Proporciona una representación visual donde se puede observar de forma rápida el número de conexiones y el tamaño de cada grupo de clases o componentes en la leyenda. Aunque este archivo es útil, podría mejorarse para que el riesgo se entienda sin necesidad de un conocimiento profundo del problema. Esta mejora podría incluir indicadores más claros de riesgo, como porcentajes relativos al total. Esta característica ayuda a la decisión en la asignación de profesores con condiciones de salud delicadas. Además, la utilización de múltiples colores para representar diferentes componentes puede ser problemática para personas con daltonismo. Se podría haber considerado una forma alternativa de diferenciación, como el uso de etiquetas adicionales o patrones, para mejorar la accesibilidad y la claridad de la información presentada.

Con todo lo mencionado, se considera que los objetivos generales de desarrollar una aplicación web que optimiza la organización estudiantil y facilita la comprensión de las soluciones a través de una presentación visual clara y detallada han sido logrados satisfactoriamente. Sin embargo, hay margen para mejorar la claridad y accesibilidad de la información visual para todos los usuarios.

Conclusiones técnicas

El objetivo de implementar un algoritmo genético utilizando DEAP no se logró como se esperaba debido a un problema técnico con el algoritmo NSGA-II, que presentaba un bug impidiendo su funcionamiento adecuado para la optimización multiobjetivo. La migración a la biblioteca Pymoo resultó ser una solución efectiva. Pymoo permitió implementar el algoritmo NSGA-II de manera correcta, superando las limitaciones encontradas con DEAP.

El manejo de redes con NetworkX fue generalmente sencillo y eficiente. Se presentó un problema con la visualización de autoenlaces debido a una incompatibilidad de versiones. La solución fue actualizar NetworkX y Python, lo que corrigió el problema y mejoró la visualización de los grafos.

El despliegue en Netlify no se intentó debido a su enfoque principalmente en frontend. En cambio, se optó por Heroku, que presentó la limitación de que las peticiones no pueden exceder los 30 segundos. Para superar esta restricción, se creó una imagen de Docker Compose, permitiendo la ejecución de la aplicación en cualquier servidor sin restricciones de tiempo en las solicitudes.

Se utilizaron Git y GitHub de manera eficaz tanto para el control de versiones como para la gestión de tareas del proyecto. GitHub, en particular, facilitó la organización y colaboración en el código.

Trello también se utilizó con éxito para la planificación y seguimiento de las tareas del proyecto. La herramienta ayudó a mantener una visión clara del progreso y a gestionar el flujo de trabajo de manera organizada.

Conclusiones y Líneas de trabajo futuras

En lo relativo a SCRUM, aunque se planificaron sprints de una semana, en algunas ocasiones no se avanzó según lo previsto, extendiendo los sprints a dos semanas para acomodar el trabajo pendiente. Esta flexibilidad en la duración de los sprints revela áreas donde se podría mejorar la consistencia y la eficiencia en la planificación del proyecto.

En cuanto a los objetivos, se alcanzaron en su mayoría, destacando la optimización multiobjetivo y la visualización clara de soluciones mediante el frente de Pareto. Sin embargo, la aplicación de SCRUM mostró áreas donde la consistencia y la planificación de sprints podrían mejorar para futuros proyectos.

Conclusiones personales

Trabajar sobre el código de un proyecto anterior permitió comprender la importancia de la mantenibilidad y la documentación interna del código. La experiencia adquirida en la mejora y optimización del código heredado subrayó la necesidad de mantener un código limpio, bien documentado y fácil de entender para facilitar futuras modificaciones y mejoras.

El proyecto proporcionó una oportunidad para consolidar mis conocimientos en algoritmos genéticos y optimización multiobjetivo. La implementación y refinamiento del algoritmo genético, así como la integración de diversas herramientas y tecnologías, enriquecieron considerablemente mi conjunto de habilidades técnicas y prácticas.

En cuanto a las metodologías ágiles, especialmente SCRUM, he adquirido una comprensión más profunda de la planificación de sprints, la estimación de historias de usuario y la gestión de la carga de trabajo. Esta experiencia me permitió apreciar la importancia de la iteración y la flexibilidad en la gestión de proyectos de software.

En el documento de anexos, pude aplicar los conocimientos adquiridos en asignaturas de ingeniería de software, aunque reconozco que el manejo de patrones de diseño podría haber sido más sólido. Esto subraya la necesidad continua de mejorar y expandir mi comprensión de los principios de diseño y arquitectura de software en proyectos futuros.

En resumen, este proyecto no solo ha demostrado la utilidad y relevancia de los conocimientos adquiridos en mi formación académica, sino que también ha consolidado mi comprensión de su aplicación práctica en situaciones reales. A lo largo del desarrollo, he llegado a apreciar profundamente cómo los fundamentos teóricos enseñados en la universidad se traducen directamente en soluciones tangibles y efectivas. Este proceso no solo ha fortalecido mi confianza en las habilidades técnicas y metodológicas que he adquirido, sino que también ha avivado mi entusiasmo por enfrentar nuevos desafíos en el vasto campo de la ingeniería de software.

7.2 Líneas de trabajo futuras

Restricción de la capacidad

El principal problema en la implementación de este proyecto. Actualmente, la solución implementada ajusta las soluciones no válidas al finalizar la ejecución, lo cual puede resultar en la pérdida de soluciones potencialmente útiles durante la búsqueda. Se considera explorar alternativas como una representación diferente del genotipo o la penalización de soluciones no factibles en la evaluación para mejorar este proceso.

Mejorar la seguridad de la aplicación

Implementar medidas adicionales de seguridad, como la encriptación de claves de usuarios en la base de datos y la introducción de tokens CSRF para proteger las solicitudes HTTP, fortaleciendo así la seguridad y la integridad de los datos.

Implementación de tests

Integrar pruebas automatizadas para verificar la funcionalidad y robustez del sistema, asegurando que futuras modificaciones no afecten negativamente la estabilidad ni el rendimiento de la aplicación.

Agregar funcionalidades para gestionar datos de entrada

Considerando que los datos de los alumnos suelen estar en tablas o bases de datos, sería interesante integrar funcionalidades que permitan importar y manejar estos datos de manera más eficiente y adaptable a diferentes formatos de entrada.

Añadir un método automatizado para comparar soluciones con igual fitness

Actualmente la forma de comparar las soluciones con igual fitness es manual. Se podría desarrollar un método sistemático para identificar y comparar las diferencias entre soluciones con igual valor de fitness, facilitando la selección de la solución óptima en base a criterios específicos y eliminando la necesidad de evaluaciones manuales.

Optimizar las funciones del algoritmo

Mejorar la complejidad algorítmica y explorar el uso de múltiples hilos para optimizar la ejecución del algoritmo, incrementando así la eficiencia computacional y reducir los tiempos de procesamiento.

Añadir el tiempo restante de los algoritmos

Agregar funcionalidades que proporcionen información sobre el tiempo restante de ejecución de los algoritmos, permitiendo que los usuarios conozcan la situación en todo momento.

Refactorización de código

Reestructurar el código existente para reducir la complejidad de algunas funciones, limitar el uso de variables globales y mejorar la modularidad y la legibilidad del código, facilitando así el mantenimiento y la evolución del sistema.

Mejorar el sistema de sesiones de usuario

Se identifican errores de consistencia en la gestión de sesiones de usuario, como la posibilidad de acceder a rutas sin autorización mediante manipulación de URL. Se sugiere implementar una solución robusta utilizando bibliotecas como flask_login para mejorar la gestión de sesiones y garantizar la protección adecuada de las rutas y recursos sensibles.

Internacionalizar la aplicación

Para aumentar la accesibilidad y el alcance del proyecto, se contempla la internacionalización de la aplicación mediante la integración de soporte multilingüe, permitiendo a los usuarios interactuar con la aplicación en su idioma.

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] «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](https://en.wikipedia.org/w/index.php?title=Crossover_(genetic_algorithm)&oldid=1195846537)
- [9] 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.
- [10] D. Vrajjitoru, «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.
- [11] «Qué es scrum y cómo empezar». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://www.atlassian.com/es/agile/scrum>
- [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] «Build software better, together», GitHub. Accedido: 24 de junio de 2024. [En línea]. Disponible en: <https://github.com>

Bibliografía

- [14] «Git». Accedido: 24 de junio de 2024. [En línea]. Disponible en: <https://git-scm.com/>
- [15] «Visual Studio Code - Code Editing. Redefined». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://code.visualstudio.com/>
- [16] «gedit», *Wikipedia, la enciclopedia libre*. 27 de marzo de 2024. Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://es.wikipedia.org/w/index.php?title=Gedit&oldid=159055717>
- [17] «Python», *Wikipedia, la enciclopedia libre*. 2 de julio de 2024. Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://es.wikipedia.org/w/index.php?title=Python&oldid=161085729>
- [18] «Werkzeug — Werkzeug Documentation (3.0.x)». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://werkzeug.palletsprojects.com/en/3.0.x/>
- [19] «Jinja — Jinja Documentation (3.1.x)». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://jinja.palletsprojects.com/en/3.1.x/>
- [20] «Flask», *Wikipedia, la enciclopedia libre*. 28 de febrero de 2024. Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://es.wikipedia.org/w/index.php?title=Flask&oldid=158492722>
- [21] «Gunicorn - Python WSGI HTTP Server for UNIX». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://gunicorn.org/>
- [22] «mysql-connector: MySQL driver written in Python». Accedido: 5 de julio de 2024. [OS Independent]. Disponible en: <http://dev.mysql.com/doc/connector-python/en/index.html>
- [23] «Flask-Bootstrap — Flask-Bootstrap 3.3.7.1 documentation». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://pythonhosted.org/Flask-Bootstrap/>
- [24] D. A, «¿Qué es Bootstrap? - Una guía para principiantes», Tutoriales Hostinger. Accedido: 31 de mayo de 2024. [En línea]. Disponible en: <https://www.hostinger.es/tutoriales/que-es-bootstrap>
- [25] «NetworkX — NetworkX documentation». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://networkx.org/>
- [26] «pymoo: Multi-objective Optimization in Python». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://pymoo.org/>
- [27] «NumPy -». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://numpy.org/>
- [28] «pandas - Python Data Analysis Library». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://pandas.pydata.org/>
- [29] «Matplotlib — Visualization with Python». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://matplotlib.org/>
- [30] Christopher, «How to display Base64 images in HTML», Stack Overflow. Accedido: 31 de mayo de 2024. [En línea]. Disponible en: <https://stackoverflow.com/q/8499633>
- [31] «MySQL». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://www.mysql.com/>

Bibliografía

- [32] «JawsDB: Fast, reliable, no-bullshark Database as a service». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://jawsdb.com>
- [33] «Edición gratuita de documentos en línea con Microsoft Word | Microsoft 365». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://www.microsoft.com/es-es/microsoft-365/word>
- [34] «LaTeX - Wikipedia, la enciclopedia libre». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/LaTeX>
- [35] «EB Garamond Font Free by Georg Duffner » Font Squirrel». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://www.fontsquirrel.com/fonts/eb-garamond>
- [36] «draw.io». Accedido: 3 de julio de 2024. [En línea]. Disponible en: <https://app.diagrams.net/>
- [37] «Gephi - The Open Graph Viz Platform». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://gephi.org/>
- [38] Ángel M. Delgado-Vázquez [@amdelvaz], «Ayer se estaba hablando de #Zotero y @Malnutridos comentó que no terminaba de convencerle del todo. Voy a intentar explicar en un pequeño hilo algunas de las bondades de este gestor de referencias bibliográficas <https://t.co/Qhoh6YYYv0>», Twitter. Accedido: 30 de abril de 2024. [En línea]. Disponible en: <https://twitter.com/amdelvaz/status/1607775331251027968>
- [39] «Docker: Accelerated Container Application Development». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://www.docker.com/>
- [40] «Docker Compose overview», Docker Documentation. Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://docs.docker.com/compose/>
- [41] «Heroku», *Wikipedia, la enciclopedia libre*. 15 de mayo de 2023. Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://es.wikipedia.org/w/index.php?title=Heroku&oldid=151170100>
- [42] «Cloud Application Hosting for Developers | Render». Accedido: 6 de julio de 2024. [En línea]. Disponible en: <https://render.com/>
- [43] «Koyeb: Push code to production, everywhere, in minutes», Koyeb. Accedido: 6 de julio de 2024. [En línea]. Disponible en: <https://www.koyeb.com>
- [44] «Code Capsules - One Platform for MEAN/MERN/MEVN Devs», <https://codecapsules.io/>. Accedido: 6 de julio de 2024. [En línea]. Disponible en: <https://codecapsules.io/>
- [45] «SonarCloud Documentation». Accedido: 5 de julio de 2024. [En línea]. Disponible en: <https://docs.sonarsource.com/sonarcloud/>
- [46] «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
- [47] «The Heroku CLI | Heroku Dev Center». Accedido: 6 de julio de 2024. [En línea]. Disponible en: <https://devcenter.heroku.com/articles/heroku-cli#download-and-install>

Bibliografía

- [48] A. Timanovsky, «timanovsky/subdir-heroku-buildpack». 5 de junio de 2024. Accedido: 6 de julio de 2024. [En línea]. Disponible en: <https://github.com/timanovsky/subdir-heroku-buildpack>
- [49] «Request Timeout | Heroku Dev Center». Accedido: 3 de julio de 2024. [En línea]. Disponible en: <https://devcenter.heroku.com/articles/request-timeout>
- [50] K. Tanınmış, N. Aras, E. Güney, y M. Sinnl, «Benders decomposition algorithms for minimizing the spread of harmful contagions in networks», *Comput. Oper. Res.*, vol. 167, 2024, doi: 10.1016/j.cor.2024.106675.