Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

# Resolución del *Multi Depot Cumulative Capacitated Vehicle Routing Problem* mediante Computación Evolutiva

*Resolution of the Multi Depot Cumulative Capacitated Vehicle Routing Problem with Evolutionary Computation*

Cristo Daniel Navarro Rodríguez

La Laguna, 7 de septiembre de 2021

D. **Eduardo Manuel Segredo González**, con N.I.F. 78.564.242-Z profesor Ayudante Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Casiano Rodríguez León**, con N.I.F. 42.020.072-S profesor Catedrático de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

**C E R T I F I C A (N)**

Que la presente memoria titulada:

”*Resolución del Multi Depot Cumulative Capacitated Vehicle Routing Problem mediante Computación Evolutiva*”

ha sido realizada bajo su dirección por D. **Cristo Daniel Navarro Rodríguez**, con N.I.F. 54.109.974-J.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 7 de septiembre de 2021

# Agradecimientos

A mis tutores Eduardo y Casiano, por ayudarme y aconsejarme a lo largo de todo el desarrollo del proyecto.

A mi familia y amigos por apoyarme siempre y animarme cuando más me ha hecho falta.

# Licencia

**Resumen**

*El Multi Depot Cumulative Capacitated Vehicle Routing Problem es un problema de optimización de rutas de vehículos de reciente formulación. En él se busca minimizar el tiempo de llegada de una flota de vehículos a los clientes establecidos, ajustándose a la capacidad de los vehículos. Este problema puede resultar muy útil aplicado a situaciones de catástrofes naturales, donde el tiempo de llegada para socorrer a las posibles víctimas es crucial.*

*La computación evolutiva nos proporciona mecanismos para resolver este tipo de problemas de optimización, pudiendo así, aportar nuevos resultados a los estudios previos. Es por ello que se ha optado por implementar un algoritmo memético para resolver el problema, realizando el estudio correspondiente sobre los resultados, así como la creación de una aplicación web para mostrar su funcionamiento. Para implementarlo, se ha extendido la librería GeneticsJS para adaptarla a las necesidades del problema.*

**Palabras clave:** Problema de optimización, Problema de Enrutamiento de Vehículos, Computación Evolutiva, Desarrollo de Software

**Abstract**

*The Multi Depot cumulative Capacitated Vehicle routing Problem is an optimization problem for vehicle routes which was recently formulated. Its objective is to minimize the arrival time of a fleet of vehicles to a group of clients, taking into account the capacity of the vehicles. This problem can be very useful when applied to natural disaster situations, where the arrival time to the possible victims is crucial.*

*Evolutionary Computation gives us the tools to solve this type of optimization problem, being able to bring new results to the previous studies of this problem. It is for this reason that we decided to implement a memetic algorithm to solve the problem, performing the corresponding study of the results, and creating a web application to show its functioning. In order to implement it, the library GeneticsJS was extended to adapt it to the needs of the problem.*

**Keywords:** Optimization Problem, Vehicle Routing Problem, Evolutionary Computation, Software Development

# Índice general

# Índice de Figuras

# Índice de Tablas

# Capítulo 1

# Introducción

Este capítulo tiene como intención, introducir los diferentes conceptos necesarios para comprender el trabajo que aquí se presenta, así como los objetivos planteados al comienzo del mismo y las distintas herramientas utilizadas para desarrollarlo.

## 1.1. Problemas de optimización de rutas de vehículos

El problema de optimización de rutas de vehículos o *Vehicle Routing Problem* (*VRP*) [1] y su multitud de variantes son problemas de optimización combinatoria y de programación entera, que buscan obtener las rutas óptimas que deben seguir una flota de vehículos satisfaciendo la demanda de los clientes y cumpliendo una serie de restricciones que difieren de una variante a otra.

El *VRP* se trata de una generalización del *Travelling Salesman Problem* [2], que busca obtener la ruta óptima para recorrer todas las ciudades de un conjunto, pasando por cada una de ellas una única vez. Este es un problema perteneciente al conjunto de problemas conocidos como *NP-Hard* y es muy conocido debido a que es uno de los más estudiados en el ámbito de la optimización, además de las múltiples aplicaciones que tiene en campos como la logística.

El *VRP* fue mencionado por primera vez en 1959, por George Dantzig y John Ramser en el artículo *The Truck Dispatching Problem* [3], donde se trata de encontrar rutas para una flota de camiones de tal forma que la demanda de las estaciones de servicio sea satisfecha y el recorrido de los mismos sea mínimo.

Desde que fue enunciado, han aparecido multitud de variantes del *VRP*, donde se modifican o añaden restricciones del problema base. Entra ellas podemos destacar el *Vehicle Routing Problem with Pickup and Delivery* (*VRPPD*), que hace una diferencia entre nodos en los que recoge un elemento y nodos en los que entrega dicho elemento, de tal forma que se busca la ruta óptima para recoger y entregar todos los productos. También se encuentra el *Capacitated Vehicle Routing Problem* (*CVRP*) del cual hablaremos más adelante, pero cuya peculiaridad es que añade una restricción de capacidad a los vehículos, o el *Multi Depot Vehicle Routing Problem* (*MDVRP*) que dispone de un conjunto de depósitos de los que parten los vehículos.

### 1.1.1. *Multi Depot Cumulative Capacitated Vehicle Routing Problem*

El *Multi Depot Cumulative Capacitated Vehicle Routing Problem* (*MDCCVRP*) es una variante del *VRP*, aunque podríamos asemejarlo en mayor medida al *Cumulative Capacitated Vehicle Routing Problem* (*CCVRP*), cuyo objetivo es el de minimizar la suma de los tiempos de llegada de los vehículos a los nodos clientes, tal y como se define en el artículo *An effective memetic algorithm for the cumulative capacitated vehicle routing problem* [4]. Por lo tanto, el *MDCCVRP* añade la característica de que se disponen de múltiples depósitos de los que parten los vehículos.

Este problema fue definido por primera vez en el artículo *A POPMUSIC approach for the Multi-Depot Cumulative Capacitated Vehicle Routing Problem* [5] en el año 2018. En este artículo se propuso una *"matheuristic"* denomida *POPMUSIC* para resolverlo, además de proponer la formulación matemática para el mismo.

En mayor detalle, en el *MDCCVRP* disponemos de una flota de vehículos con una cierta capacidad que es la misma para cada uno de ellos. Por otro lado, los depósitos no tienen capacidad, por lo que todos los vehículos de la flota podrían partir del mismo depósito, y los clientes tienen una demanda que debe ser satisfecha. El objetivo es el de obtener las rutas óptimas para cada vehículo del tal forma que los vehículos partan de los depósitos y se visiten a todos los clientes satisfaciendo sus demandas. Además, se tienen que tener en cuenta las siguientes restricciones:

- Cada cliente debe ser visitado una única vez
- La demanda de cada cliente debe ser satisfecha cuando se visita
- Cada vehículo parte de un depósito y termina en el mismo
- La demanda total de los clientes visitados por un vehículo no puede superar su capacidad

## 1.2. Computación Evolutiva

Según se describe en el libro *Introduction to Evolutionary Computing* [6], la Computación Evolutiva es un campo de la computación que está fuertemente inspirado por la evolución natural. Esta se puede simplificar de la siguiente manera: partimos de una población de individuos que convive en un entorno y sus objetivos son sobrevivir y reproducirse. Estos individuos tienen una capacidad de supervivencia que depende del entorno en el que viven, y está determinada por la medida en la que alcanzan sus objetivos. Este concepto se podría traducir a la resolución de problemas de tal forma que el entorno se correspondería con el problema a resolver, la población con un conjunto de posibles soluciones cuya calidad depende de cómo resuelvan el problema y aquellas con mayor calidad serán las progenitoras de las futuras generaciones.

Como se puede observar en el párrafo anterior, el concepto detrás de la computación evolutiva está inspirado por la teoría de la evolución de Darwin [7] que explica la diversidad biológica y los mecanismos detrás de la misma, así como la importancia de la

selección natural. Esto último introduce el fenómeno de la supervivencia de los individuos que mejor se adaptan al medio. De esta forma, son aquellos que mejor se adapten los que tienen descendencia, mientras que el resto no sobrevive al proceso de selección natural. Además, se introduce el concepto de mutaciones, que son pequeñas variaciones que se producen en los individuos originados a partir de la reproducción, pudiendo provocar que el individuo mejore sus capacidades para sobrevivir.

Todos estos conceptos y fenómenos comentados previamente se pueden aplicar a la Computación Evolutiva y se verán ejemplificados en mayor medida durante la explicación del algoritmo desarrollado. Es esta idea de la evolución natural la que hace que la Computación Evolutiva sea un campo que genere tanto interés, y es que la complejidad de los problemas que se quieren resolver cada vez es mayor, y existe una necesidad de obtener algoritmos aplicables a una gran diversidad de problemas, que obtengan soluciones óptimas (o cercanas a las óptimas) y que las obtengan en un tiempo razonable. La computación Evolutiva nos permite alcanzar todos estos objetivos mediante los llamados *Algoritmos Evolutivos*.

### 1.2.1. Algoritmos Meméticos

Antes de entrar en detalle con la definición de los Algoritmos Meméticos, hay que hacer una pequeña introducción a los Algoritmos Evolutivos. Como se comentaba previamente, los Algoritmos Evolutivos forman parte del campo de la Computación Evolutiva, y por lo tanto aplican el concepto de la evolución natural. Por lo tanto, la idea fundamental de todo Algoritmo Evolutivo es la siguiente:

**Algoritmo 1:** Algoritmo Evolutivo

```
1 Inicializar la población con soluciones aleatorias
2 Evaluar cada individuo
3 while Condición de parada do
4     Seleccionar padres
5     Combinar padres
6     Mutar la descendencia
7     Evaluar nuevos individuos
8     Seleccionar individuos para la siguiente generación
```

Una vez vista esta idea, podemos describir a los Algoritmos Meméticos como Algoritmos Evolutivos que incluyen un procedimiento de mejora mediante una búsqueda local. Este procedimiento se aplica después de generar la descendencia y por lo tanto, en la mayoría de casos, sustituye al procedimiento de mutación de los individuos. El concepto detrás de la búsqueda local es el de generar un vecindario (soluciones similares que se obtienen aplicando un operador) alrededor de la solución que se quiere mejorar y se sustituye por algún vecino que la mejore. Existen multitud de variantes de las búsquedas locales, por lo que no se va a entrar en mayor detalle, pero en secciones futuras se explicará la búsqueda local implementada para nuestro caso.

## 1.3. Objetivos

Para el desarrollo del proyecto se han marcado una serie de objetivos que se debían cumplir y que permitieron el desarrollo gradual del mismo. A continuación se listan los distintos objetivos marcados, así como una breve explicación de la motivación detrás de los mismos:

- **Análisis e identificación del problema:** A partir de la definición del problema se buscaba detectar si se podía aplicar la computación evolutiva, así como posibles planteamientos del problema y si existían oportunidades que explotar con el proyecto.
- **Estudio del estado del arte:** Se planteó un estudio previo del estado del arte del problema para detectar si la aproximación utilizando un algoritmo memético podría llegar a aportar valor.
- **Implementación de las técnicas algorítmicas de resolución:** El objetivo principal del proyecto es el de implementar un algoritmo capaz de resolver el problema mediante la computación evolutiva y siendo capaz de aportar soluciones óptimas o cercanas a las mismas. Además, se buscaba utilizar y extender la librería GeneticsJS [8].
- **Implementación de las interfaces:** Por último, al desarrollarse el proyecto en *TypeScript*, se buscaba implementar una aplicación web que permitiese a los usuarios resolver instancias del problema utilizando el algoritmo desarrollado.

## 1.4. Tecnologías utilizadas

En este apartado se presentan las distintas tecnologías empleadas para el desarrollo del proyecto durante las distintas etapas del mismo, tanto para el resolutor del *MDCCVRP*, como para la aplicación web.

### 1.4.1. Lenguaje y entorno utilizados

Entre uno de los objetivos planteados para el proyecto se encontraba el de extender la librería *GeneticsJS*, y debido a que esta estaba implementada en *TypeScript*, las modificaciones introducidas se desarrollaron en dicho lenguaje. Además, también fue utilizado para el resolutor del *MDCCVRP* y la aplicación web para la visualización de los resultados.


Figura 1.1: Logo de *TypeScript*

Figura 1.2: Logos de *Prettier*, *ESLint* y *Visual Studio Code*

*TypeScript* [9] es un lenguaje de código abierto que se construye sobre *JavaScript*, y le añade la definición estática de tipos. Los tipos ofrecen una manera para definir la forma de los objetos en *JavaScript*, facilitando la generación de documentación y comprobar que se esté desarrollando el código de forma correcta, así como su funcionamiento. Además, cabe destacar que *TypeScript* es un lenguaje que necesita ser compilado a *JavaScript* para poder ser ejecutado o para poder publicarse un paquete en *npm*.

Para ejecutar código en *TypeScript*, a parte de compilarse como se comentaba previamente, necesita de un entorno de ejecución como es *NodeJS* [10]. Por otro lado, también se hablaba de *npm* [11], que se trata del mayor registro de software en el mundo. Esta plataforma se utiliza tanto en el ámbito del desarrollo de código abierto, como en algunas empresas privadas para publicar y compartir paquetes. En este proyecto se ha utilizado *npm* tanto para integrar la librería *GeneticsJS*, como para publicar un paquete con el algoritmo implementado, de tal forma que pueda ser usado en la aplicación web.

### 1.4.2. Tecnologías utilizadas para el desarrollo

En este apartado se van a describir muy brevemente las distintas tecnologías utilizadas para desarrollar por un lado, el algoritmo, y por otro, la aplicación web, así como la motivación detrás de su uso. En primer lugar se hablará de aquellas que son comunes para ambos, como son el editor de código, las herramientas para el formateo del código o las empleadas para el control de versiones.

#### Editor y formateo del código

El editor de código utilizado es *Visual Studio Code* [12], ya que es de código abierto, muy flexible, permitiendo la instalación de extensiones que mejoran la experiencia del usuario. Entre estas, se encuentran las que permiten utilizar de forma sencilla las herramientas de formateo que se van a comentar más adelante.

Por un lado se utilizó *Prettier* [13], que es una herramienta que permite aplicar el formato que deseamos al código de forma automática. Además, gracias a la extensión de *Visual Studio Code*, se puede establecer que se aplique el formato automáticamente al guardar un fichero, lo que consigue acelerar el proceso del desarrollo del código.

Por otro lado, para detectar posibles errores en la sintaxis o en el formato del código se utilizó *ESLint* [14], que satisface estas necesidades, y tal como sucede con *Prettier*, existe una extensión para integrarlo en el editor.

Figura 1.3: Logo de *Github*

**Control de versiones**

El control de versiones es un elemento fundamental en el desarrollo de software, ya que mejora la organización del mismo y permite volver a casi cualquier punto del desarrollo de forma sencilla mediante el historial de cambios o *commits*.

Para llevar a cabo este control, se han utilizado *Git* [15] como sistema de control de versiones, y *GitHub* [16] para alojar los repositorios con el código fuente.

**Pruebas**

Otro aspecto fundamental en el desarrollo de software son las pruebas o *tests*, con los que se puede garantizar el correcto funcionamiento del código implementado. Esto resulta de gran utilidad en proyectos donde conviven multitud de clases o estructuras de datos, ya que se pueden realizar pruebas a cada uno de ellos de forma individual, permitiendo detectar posibles errores de forma sencilla.

Para el desarrollo de las pruebas se ha optado por *Jest* [17], debido a que no requiere de una configuración extensiva, ofrece mecanismos para la creación de *mocks* y es sencillo de utilizar, entre otras razones. Además, ya es un *framework* con que el que estaba familiarizado y se utilizó en la librería *GeneticsJS*, por lo que durante la extensión de la librería se elaboraron pruebas mediante *Jest*.

**Documentación**

Añadir documentación al código facilita la utilización, lectura y extensión del mismo. Es por ello que se decidió añadirla tanto a los nuevos elementos de la librería *GeneticsJS*, como al algoritmo para resolver el *MDCCVRP*. Para ello se utilizó *TypeDoc* [18], ya que permite comentar las clases, interfaces y métodos, así como los parámetros que reciben y, además, pudiéndose generar un *HTML* con la documentación para que pueda ser publicada.

### 1.4.3. Tecnologías utilizadas en producción

Las tecnologías en producción son aquellas que son utilizadas por los usuarios al ejecutar tanto el algoritmo, como la aplicación web.

En el caso de los algoritmos, sólo tenemos dos dependencias, por un lado, la librería

*GeneticsJS*, y por el otro, *Yargs*. De estas dependencias se habla en más detalle en futuras secciones, por lo que no nos vamos a centrar mucho en ellas.

Sin embargo, para el caso de la aplicación web sí que se requirieron de más dependencias. Entre ellas se encuentra *NextJS* [19], que consiste en un *framework* para el desarrollo de aplicaciones en *React* [20]. Se eligió utilizar la librería *React* ya que es la más utilizada para el desarrollo de interfaces de usuario para aplicaciones web. Por otro lado, existen diversos *frameworks* para trabajar con *React*, pero se optó por *NextJS* ya que es el más popular y simplifica el trabajo, sobre todo para proyectos más grandes. Pese a que no se pudo utilizar todo el potencial de este *framework*, se buscaba aprender a utilizarlo aunque se tratase de una aplicación sencilla.

También se utilizó una librería para la creación de estilos llamada *Stitches* [21]. Esta librería proporciona diversos mecanismos para definir variantes de los componentes, variables, herencia de estilos y propiedades, entre muchos otros.

Por último, caben destacar las librerías *react-dropzone* [22] para implementar un *drag and drop* en la aplicación para la subida de archivos, y la librería *react-chartjs-2* [23] que es un *wrapper* para la librería *chart.js*, con la que se pueden representar multitud de gráficos, y en nuestro caso fue utilizada para la representación visual de las soluciones.

## 1.5. Enlaces de interés

En esta sección se presentan una serie de enlaces a los distintos elementos desarrollados a lo largo del proyecto, con el objetivo de tenerlos todos accesibles desde un mismo punto:

- **Organización GeneticsJS:** https://github.com/GeneticsJS
- **Repositorio de GeneticsJS:** https://github.com/GeneticsJS/GeneticsJS
- **Paquete npm de GeneticsJS:** https://www.npmjs.com/package/genetics-js
- **Repositorio del resolutor:** https://github.com/ULL-ESIT-INF-TFG-2021/TFG-2021-Cristo-Daniel-Navarro-MDCCVRP
- **Repositorio de la aplicación web:** https://github.com/ULL-ESIT-INF-TFG-2021/TFG-2021-Cristo-Daniel-Navarro-Web-App
- **Paquete npm del resolutor:** https://www.npmjs.com/package/genetics-js-mdccvrp
- **Aplicación web:** https://mdccvrp-solver-web-app.vercel.app/
- **Repositorio con las instancias:** https://ull-esit-inf-tfg-2021.github.io/MDCCVRP-Instances/

# Capítulo 2

# Desarrollo

En este capítulo se expone el procedimiento seguido para el desarrollo del proyecto, además de los diferentes elementos implementados y la estructuración de los mismos.

## 2.1. Estudio del Estado del Arte

Para abordar el problema del ***Multi-Depot Cumulative Capacitated Vehicle Rounting Problem (MDCCVRP)*** se ha llevado a cabo, en primer lugar, una revisión sistemática para conocer el estado del arte del problema. En dicha revisión se ha aplicado la metodología ***"Preferred Reporting Items for Systematic Reviews and Meta-Analyses"(PRISMA)*** [24], utilizada para la evaluación y análisis de artículos de índole científica.

A lo largo de esta sección se describe cómo se ha procedido para implementar dicha metodología para nuestro análisis sobre el estado del arte del problema.

### 2.1.1. Objetivo

Con este análisis se busca adquirir un mayor conocimiento sobre el problema y las aproximaciones llevadas a cabo hasta el momento, así como las dificultades que presenta, permitiéndonos responder a las preguntas que nos hemos planteado:

- ¿Cuáles son las aproximaciones desarrolladas para el *MDCCVRP*?
- ¿Dónde se han hallado más dificultades a la hora de resolver el problema?
- ¿Qué se puede aportar a las propuestas actuales?
- ¿Qué aproximaciones se han desarrollado para la variante del problema denominada *Cumulative Capacitated Vehicle Routing Problem* (*CCVRP*)?

Debido a la novedad del problema y escasez de artículos relacionados, se ha decidido estudiar también el *CCVRP*.

### 2.1.2. Estrategia de búsqueda

Para llevar a cabo la búsqueda de artículos, se ha buscado en distintas bases de datos especializadas en artículos de carácter científico. Más específicamente se utilizaron las siguientes palabras claves en los buscadores de las bases de datos:

- Multi-Depot Cumulative Capacitated Vehicle Routing Problem (*MDCCVRP*)
- MultiDepot Cumulative Capacitated Vehicle Routing Problem (*MDCCVRP*)
- Cumulative Capacitated Vehicle Routing Problem (*CCVRP*)

### 2.1.3. Bases de datos

La búsqueda comentada en el apartado anterior se ha efectuado sobre las siguientes bases de datos:

- IEEE (`https://ieeexplore.ieee.org/Xplore/home.jsp`)
- ScienceDirect (`https://www.sciencedirect.com/`)
- SpringerLink (`https://link.springer.com/`)
- DBLP (`https://dblp.org/`)
- Scopus (`https://www.scopus.com/`)
- Web Of Science (`https://www.recursoscientificos.fecyt.es/`)

### 2.1.4. Criterios de rechazo

De entre todos los artículos encontrados, se descartan antes del análisis en profundidad, aquellos que cumplan alguna de las siguientes condiciones:

- El artículo consta solo de un resumen o introducción.
- El artículo está escrito en un idioma distinto al español o el inglés.
- Artículos duplicados.
- Artículos que no hagan referencia al problema en cuestión (o variaciones similares).
- Se propone un algoritmo ya estudiado en otro artículo.

### 2.1.5. Criterios de inclusión

De la misma forma que establecemos unos criterios para rechazar un artículo, tenemos que establecerlos de la misma forma para la inclusión de los artículos:

- Artículos en español o inglés.
- Artículos que abordan el *MDCCVRP* o el *CCVRP*.
- Artículos con un modelo matemático propuesto.
- Artículos con propuestas de algoritmos y con pruebas de rendimiento.
- Artículos con una aproximación propuesta diferente.

### 2.1.6. Búsqueda, cribado y selección

En la Figura 2.1 se muestra el diagrama de flujo que refleja el procedimiento seguido durante la búsqueda y selección de artículos. Como se indica en la metodología PRISMA, el análisis se ha dividido en diversas fases donde se excluyen los artículos que no cumplen con los requisitos establecidos.

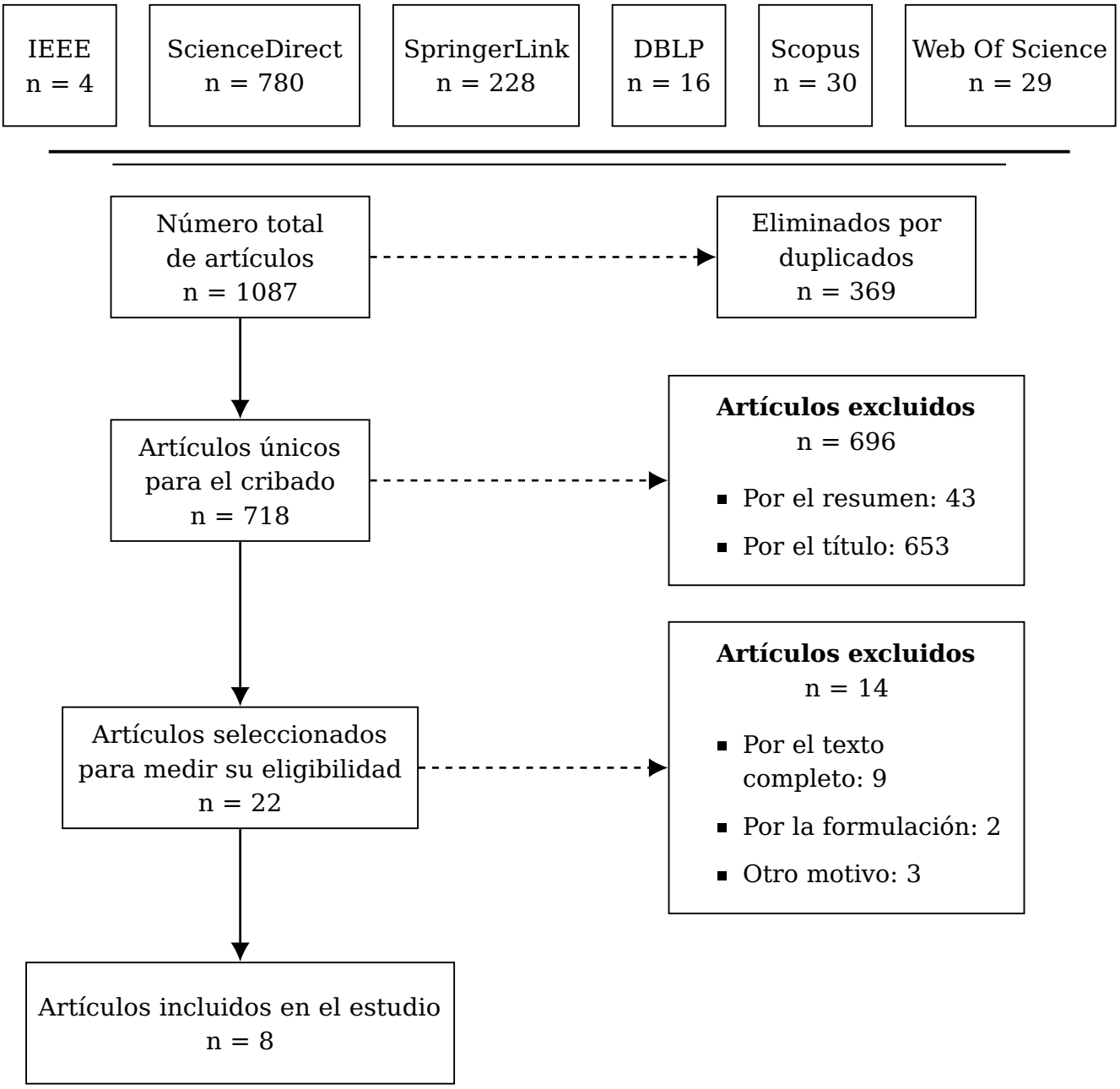


Figura 2.1: Diagrama de PRISMA

### 2.1.7. Artículos seleccionados

A continuación se presenta una pequeña descripción para cada artículo seleccionado en la revisión sistemática. Como se comentaba al comienzo de la sección, no existen muchos artículos relacionados con el *MDCCVRP*, pero sí que hay una mayor variedad tratando con el *CCVRP*.

**A POPMUSIC approach for the Multi-Depot Cumulative Capacitated Vehicle Routing Problem [5]**

Este artículo es el que originalmente planteó el problema del *Multi-Depot Cumulative Capacitated Vehicle Routing Problem*. En él se propone la utilización del framework *POPMUSIC* para subdividir la instancia del problema en partes que se juntan y constituyen subproblemas, que posteriormente se resuelven mediante una aproximación exacta o aproximada.

Para la construcción de los subproblemas se utilizaron dos aproximaciones distintas; una que seleccionaba las partes en función de su índice (lexicográfico) y otra que se seleccionan según la distancia entre los depósitos, escogiendo aquellos más cercanos.

Los resultados computacionales obtenidos, indican que en instancias pequeñas del problema, la aproximación mediante el uso de *POPMUSIC* obtiene los resultados óptimos en menos tiempo que un resolutor de uso general. Por otro lado, en instancias mayores, se suelen obtener mejores resultados sin llegar al tiempo de ejecución máximo, como ocurre con el resolutor previo.

**An Effective Local Search Algorithm for the Multidepot Cumulative Capacitated Vehicle Routing Problem [25]**

En este artículo se propone una búsqueda local para la resolución del *MDCCVRP* de forma eficiente. La solución inicial se genera de forma constructiva, en función del coste de añadir un cliente a la ruta de un vehículo.

Como todo algoritmo de búsqueda local, esta solución inicial se mejora mediante la exploración del vecindario de la misma. Para ello se han utilizado seis movimientos distintos mediante el intercambio entre *sub-rutas* de clientes, depósitos, arcos y segmentos. Por otro lado, para llevar a cabo la diversificación de las soluciones se utilizan dos operadores de perturbación. El primero intercambia arcos y clientes de forma aleatoria varias veces, y el segundo extrae los clientes con un mayor coste asociado y los recoloca en una mejor posición.

En el análisis de los resultados computacionales se hacen comparaciones con otros métodos para resolver el problema, como son los algoritmos *Greedy Randomized Adaptive Search Procedure* (*GRASP*), *Variable Neighborhood Search* (*VNS*) o la aproximación con POPMUSIC [5]. En dicho estudio se obtuvo que el tiempo necesario para resolver distintas instancias es mucho menor que con el resto de métodos, y los resultados también son en su mayoría mejores.

*A partir de este punto, los artículos presentados hacen referencia al CCVRP.*

**An effective memetic algorithm for the cumulative capacitated vehicle routing problem [26]**

Este artículo presenta la primera heurística desarrollada para el *CCVRP*. En dicha heurística se utilizan dos cotas inferiores y un algoritmo memético para la resolución del problema.

La primera cota inferior propuesta se basa en que si existe un número ilimitado de vehículos para el problema, la solución óptima sería enviar un vehículo a cada cliente/nodo.

La segunda cota ordena los costes entre nodos (arcos) de menor a mayor, les asigna a los menores un mayor coeficiente y los suma.

Por otro lado, en el algoritmo memético se parte de una población de soluciones, de las que se seleccionan dos padres por medio de una selección de torneo. Estas dos soluciones dan lugar a la descendencia y de entre dichas soluciones hijas se selecciona la mejor. Una vez se obtenga la mejor, se le aplica una búsqueda local con cierta probabilidad y se sustituye de forma aleatoria una de las peores soluciones de la población.

Debido a que el problema no había sido estudiado anteriormente, para la realización del estudio del rendimiento se utilizaron instancias del *Travelling Repairman Problem* (*TRP*), igual que se compararon los resultados con algoritmos para resolver dicho problema. De esta forma se concluyó que se había obtenido un algoritmo eficiente y que se trataba de la mejor heurística para el *TRP*, aunque no fuese diseñado para resolverlo.

**An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem [27]**

En el artículo se propone un *Adaptive Large Neighborhood Search* (*ALNS*) para afrontar el *CCVRP* de forma eficiente. Este algoritmo se basa en destruir parte de la solución actual y repararla de forma distinta, dando lugar a una nueva solución. En caso de que esta sea mejor que la original, la búsqueda continúa con la nueva solución, y en caso contrario, existe una probabilidad de proseguir con la nueva.

El algoritmo propuesto está compuesto por distintos elementos, como son un vecindario, que se obtiene tras extraer y añadir clientes de la solución mediante distintas heurísticas, un mecanismo de búsqueda adaptativo que utiliza una ruleta y el comportamiento previo de las heurísticas. También posee un ajustador de pesos, que asigna un peso a cada heurística en función de su rendimiento, una función de penalización y los criterios de aceptación y de parada.

En la propuesta se utilizan multitud de heurísticas para la extracción e inserción de clientes. Estas utilizan información como el tiempo de llegada, la distancia entre nodos, los *cluster* que se forman en la instancia, etc.

Para el estudio de rendimiento se han comparado los resultados con los obtenidos por los algoritmos meméticos [26] propuestos en la literatura. De este se pudo concluir que el *ALNS* tiene un mejor rendimiento y obtiene mejores resultados que los algoritmos existentes.

**Skewed general variable neighborhood search for the cumulative capacitated vehicle routing problem [28]**

Este artículo propone una nueva heurística para resolver instancias del *CCVRP*. En ella se pueden identificar tres partes distinguidas; un generador de soluciones iniciales constructivo, un algoritmo *Variable Neighborhood Descent* (*VND*) para la búsqueda de óptimos locales en un conjunto de vecindarios y un algoritmo *Skewed Variable Neighborhood Search* (*SVNS*) para alejarse de los óptimos locales y tratar de encontrar los globales.

Para la construcción de la solución inicial, se crean rutas con un cliente cada una, siendo este uno de los más cercanos al depósito. Posteriormente, se van añadiendo los clientes a las mejores rutas y en la mejor posición posible, tratando de respetar la factibilidad de la solución. El algoritmo *VND* utiliza tres estructuras de vecindarios distintos, en las que se

forma uno nuevo insertando clientes, conjuntos de clientes o intercambiándolos, tanto dentro de la misma ruta, como en otras distintas. Para la diversificación de las soluciones se utiliza un *SVNS*, donde se realiza una fase de perturbación para encontrar un nuevo punto incial para la búsqueda local, y un *"movimiento sesgado"* con el que se establece el criterio de aceptación de una solución.

Para probar el rendimiento del algoritmo se llevaron a cabo diversas pruebas con distintas instancias, comparando los resultados obtenidos con los de otras técnicas algorítmicas. Tras el estudio se concluyó que la heurística implementada era más eficiente que los algoritmos meméticos [26] y *ALNS* [27] publicados hasta la fecha.

**A two-phase metaheuristic for the cumulative capacitated vehicle routing problem [29]**

En este artículo se propone una metaheurística *"en dos fases"* para resolver el *CCVRP*. La solución inicial para el algoritmo se obtiene insertando los clientes en las rutas en función del coste al añadirlo a la misma.

Una vez se tiene la solución inicial, comienza la primera fase del algoritmo. En dicha fase se utiliza un movimiento de perturbación elegido al azar (intercambio o cruce). Posteriormente se pasa a la segunda fase con la solución obtenida tras la perturbación. A cada ruta de la solución se le aplica nuevamente una perturbación aleatoria, y después una última perturbación en concreto para todas las rutas. Por último, se realiza una búsqueda local para mejorar la solución hasta que no se obtenga ningún incremento en la calidad de la misma.

En el estudio de rendimiento se obtuvieron buenos resultados, en los que se alcanzaban mejores resultados y en menor tiempo que con otras técnicas, como son algoritmos meméticos [26] o un *ALNS* [27]. Sin embargo, en otras instancias, los algoritmos meméticos seguían obteniendo mejores resultados.

**The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search [30]**

En el artículo se propone un *Adaptive Variable Neighbourhood Search* (*AVNS*) que utiliza como mecanismo de diversificación mediante un *Large Neighbourhood Search* (*LNS*) para resolver el *CCVRP*. Al final del artículo también se propone una variación para el objetivo del problema, pero sólo nos centramos en la formulación inicial.

El algoritmo propuesto está dividido en dos fases. En la primera, se lleva a cabo un proceso de aprendizaje para definir la importancia y eficiencia de los operadores de las búsquedas locales, y en la segunda fase se utiliza dicha información para seleccionar un grupo pequeño de operadores en cada iteración. Para el aprendizaje durante la ejecución, se utilizan las mejoras y las frecuencias de dichas mejoras para cada operador. Entre dichos operadores, se encuentran el intercambio de nodos, de parejas y de segmentos entre otros.

Durante los experimentos computacionales se han comparado los resultados con los obtenidos por otros algoritmos como son meméticos [26], *ALNS* [27] y un *Two-Phase Metaheuristic* (*TPM*) [29]. Para ello se han utilizado diversas instancias de distintos tamaños, y se pudo concluir que el algoritmo propuesto es muy eficiente y encuentra nuevas mejores soluciones para las instancias utilizadas.

**A brain storm optimization approach for the cumulative capacitated vehicle routing problem [31]**

El artículo plantea un algoritmo del tipo *Brain Storm Optimization* (*BSO*) para resolver el *CCVRP*. Estos algoritmos identifican las soluciones como ”ideas” y tienen tres tipos de ”personas” involucradas en la ejecución. El facilitador controla y supervisa la lluvia de ideas, un grupo de personas genera nuevas ideas y el propietario del problema elige las mejores nuevas ideas.

El algoritmo propuesto parte de una solución inicial, conocida como mejor solución, obtenida de forma constructiva en función del coste de insertar un nodo en una ruta. Una vez inicializada, se utilizan operadores de convergencia y divergencia hasta alcanzar una condición de parada. El operador de convergencia se encarga de perturbar la mejor solución y descomponerla en *sub-soluciones*. Por otro lado, el operador de divergencia elige de forma aleatoria un operador para generar nuevas soluciones parciales. Si estas soluciones parciales son mejores que las obtenidas en la convergencia, se sustituyen y se forma una solución completa con las mismas.

Para el estudio de rendimiento, se han comparado los resultados con otros algoritmos utilizados para resolver el problema: *TPM* [29], *ALNS* [27] y *Hybridisation of Adaptive variable neighbour-hood search and Large neighbourhood search* (*HAL*) [30]. Para instancias pequeñas, el *BSO*, al igual que el *HAL* obtienen los mejores resultados. En instancias medianas el algoritmo propuesto es capaz de obtener nuevas mejores soluciones en algunas instancias y tiene un buen rendimiento en el resto. Por último, en instancias grandes obtiene mejores resultados en la mayoría de instancias, superando al *HAL*.

### 2.1.8. Resultados del estudio

Como se comentaba al comienzo de la sección, el *MDCCVRP* es un problema reciente y no existen muchas propuestas para el mismo, por lo que en el estudio sólo se han podido incluir dos artículos referentes al problema. Sin embargo, se ha añadido al estudio un problema similar al tratado en este proyecto, como es el *CCVRP*, del que sí se ha podido integrar una mayor variedad de artículos.

Entre las implementaciones para el *MDCCVRP*, la propuesta en [25] obtiene mejores resultados mediante la utilización de una búsqueda local con distintos mecanismos para intensificación y diversificación de las soluciones. Por otro lado, para el *CCVRP*, el algoritmo con el que se han conseguido mejores resultados es el *BSO* [31], seguido de otras propuestas como el *AVNS* [30]. Debido a las similitudes existentes entre los problemas estudiados, es interesante intentar adaptar los algoritmos vistos en el análisis para el *MDCCVRP* y estudiar su eficiencia a la hora de resolver el problema.

Los principales problemas detectados durante el estudio son las instancias de mayor tamaño, sobre todo para el caso de *POPMUSIC* [5], donde el tiempo necesario para obtener resultados es muy grande. Estas dificultades, junto a la escasez de propuestas, implican que existe la oportunidad de proponer algoritmos distintos a los existentes para resolver el problema, intentando solucionar dichas dificultades. Es por todo esto que los algoritmos evolutivos pueden resultar muy útiles para acelerar el proceso de obtención de soluciones, y junto a una búsqueda local, lograr un incremento en la calidad de las mismas.

## 2.2. Trabajo previo

En esta sección se va a indicar cómo se ha organizado el espacio de trabajo. Además, se hace una breve introducción al trabajo previo llevado a cabo por antiguos alumnos, y que ha permitido que se disponga de una base para llevar a cabo el proyecto, así de la integración de los cambios realizados y la solución de errores.

### 2.2.1. Librería GeneticsJS

GeneticsJS es una librería desarrollada en TypeScript por Cristian Manuel Abrante Dorta [32] y fue trabajada posteriormente por Francisco Arturo Cruz Zelante [33]. Esta librería consta de distintas herramientas para la implementación de algoritmos evolutivos de forma sencilla. En la librería se ofrecen diversas opciones para cada uno de los elementos de cualquier algoritmo evolutivo, como se detalla a continuación.

#### Individuos

Una de las principales tomas de decisión al enfrentarse a un problema mediante la Computación Evolutiva, es la codificación de los individuos. En el caso de GeneticsJS, se nos presentan tres opciones:

- **Binario.** Estos individuos están compuestos por un *array* con valores 0 o 1.
- **Numérico entero.** Individuos que constan de un *array* de números enteros.
- **Numérico flotante.** Los individuos están formados por un *array* de números en punto flotante.

#### Operadores de selección

Existen dos situaciones durante la ejecución de un algoritmo evolutivo en las que es necesario elegir una serie de individuos: durante la selección de los padres y cuando se reemplazan algunos individuos por la nueva descendencia. Es por esto, que distinguimos entre dos tipos de operadores:

- **Selección de padres:**
  - **Roulette Wheel.** Se representan en una "ruleta" los individuos de la población (cuanto mayor sea su *fitness*, mayor sección ocupan) y se lanza la ruleta tantas veces como individuos se quieran seleccionar.
  - **Stochastic Universal Sampling.** Al igual que el anterior, los individuos se representan en una "ruleta". Sin embargo, la ruleta se lanza una única vez y se seleccionan todos los individuos necesarios a la vez.
- **Reemplazo:**
  - **Basado en edad.** Se eligen aquellos individuos que llevan más tiempo en la población para que sean reemplazados.
  - **Basado en calidad.** Los peores individuos (según su *fitness*) son los seleccionados para sustituirlos.

### Operadores de cruce

Para los distintos tipos de individuos se han implementado diferentes mecanismos de cruce. Con estos operadores se combina el material genético de dos individuos, dando lugar a otros dos individuos. De entre estos mecanismos, algunos se pueden aplicar a cualquier tipo de individuo, mientras que otros están enfocados a los individuos con números en punto flotante.

- **Operadores comunes:**
  - **Cruce de N puntos.** Se eligen $N$ puntos para dividir el material genético de los individuos para luego combinarlos.
  - **Cruce de 1 punto.** Se elige un único punto para dividir el material genético de los individuos.
  - **Cruce uniforme.** Para cada gen se calcula una probabilidad, y en función de dicho valor se decide de qué individuo se coge el gen para generar el material genético de la descendencia.
- **Operadores de punto flotante:**
  - **Recombinación aritmética simple.** Se genera un índice $k$ de forma aleatoria y se mezclan los genes cuyo índice sea superior al valor de $k$. Este operador también depende de un parámetro $\alpha$, siguiendo la siguiente expresión para el primer y segundo individuo de la descendencia respectivamente:

$$< x_1, \ldots, x_k, \alpha \cdot y_{k+1} + (1-\alpha) \cdot x_{k+1}, \ldots, \alpha \cdot y_n + (1-\alpha) \cdot x_n >$$

$$< y_1, \ldots, y_k, \alpha \cdot x_{k+1} + (1-\alpha) \cdot y_{k+1}, \ldots, \alpha \cdot x_n + (1-\alpha) \cdot y_n >$$

  - **Recombinación aritmética única.** Este operador sólo aplica la expresión de mezcla en una posición $k$ elegida aleatoriamente. La expresión de este método para la obtención de descendientes es la siguiente:

$$< x_1, \ldots, x_{k-1}, \alpha \cdot y_k + (1-\alpha) \cdot x_k, x_{k+1}, \ldots, x_n >$$

$$< y_1, \ldots, y_{k-1}, \alpha \cdot x_k + (1-\alpha) \cdot y_k, y_{k+1}, \ldots, y_n >$$

  - **Recombinación aritmética completa.** El operador aplica la siguiente expresión a cada uno de los genes, siendo $z^x$ un gen del primer hijo, y $z^y$, del segundo:

$$z^x = \overline{x} + (1-\alpha) \cdot \overline{y}$$

$$z^y = \overline{y} + (1-\alpha) \cdot \overline{x}$$

### Operadores de mutación

Los operadores de mutación son del tipo unario, ya que trabajan únicamente con un individuo y modifican su genotipo. Al contrario que en el caso anterior, únicamente se han implementado operadores específicos en función del tipo del individuo.

- **Operadores de individuos binarios:**

    - **Bit a bit.** Consiste en cambiar el valor de cada gen en función de una probabilidad de mutación. Por lo que si un gen estaba a *verdadero*, pasa a *falso* y viceversa.

- **Operadores de individuos enteros:**

    - **De arrastre.** Este operador modifica cada uno de los genes del individuo, añadiéndoles una cantidad redondeada y obtenida de una distribución normal.
    - **Reajuste aleatorio.** Equivale al operador bit a bit de los individuos binarios. En este caso, se cambia el valor del gen por un número aleatorio dentro de un rango dado.

- **Operadores de individuos flotantes:**

    - **No uniforme.** Este operador es similar al operador de arrastre de los individuos enteros, ya que se modifican los genes añadiéndoles una cantidad obtenida de una distribución normal.
    - **Uniforme.** El operador equivale al de *bit a bit* de los individuos binarios, donde se modifica el valor por uno aleatorio dentro de un rango establecido.

#### Función de *fitness*

Se encarga de medir cuán de adaptado está un individuo a su entorno. Por defecto, en la librería se establece que la función de *fitness* consiste en maximizar y, por lo tanto, se consideran mejores aquellos individuos que tengan unos valores más altos. Además, se permite que se especifique la función según como sea necesaria para cada problema, pero necesitando que la función reciba como argumento al individuo a evaluar.

#### Condición de finalización

En todo algoritmo evolutivo es necesario establecer una condición de parada, para que el algoritmo termine su ejecución. En la librería se ha implementado como condición a llegar a un número concreto de generaciones.

### 2.2.2. Organización GeneticsJS

Para poder organizar todo lo relacionado con la librería GeneticsJS se ha creado una organización con el mismo nombre (`https://github.com/GeneticsJS`). En dicha organización se han realizado diversos *"fork"* de los trabajos relacionados, entre los que se encuentran:

- **Librería GeneticsJS.** Repositorio con la librería descrita en el apartado anterior.
- **GeneticsJS Knapsack.** Aplicación web que utiliza GeneticsJS para resolver el conocido problema de la mochila.
- **TFG Francisco Cruz.** Repositorio con el trabajo de fin de grado de Francisco Cruz Zelante, donde propone una planificación de un sistema de semáforos haciendo uso de GeneticsJS.

- **TFG Cristian Abrante.** Trabajo de fin de grado de Cristian Abrante Dorta, en el que desarrolla la librería.

Además de realizar los *"fork"* correspondientes, tuvieron que llevarse a cabo una serie de cambios y ajustes que se describen en la sección *"Integración de cambios"*.

### 2.2.3. Integración de cambios

En este apartado se van a abarcar las distintas modificaciones y ajustes que han sido necesarios para preparar la librería para poder comenzar con la extensión de la misma. Como se comentaba previamente, GeneticsJS fue desarrollada por Cristian Abrante y su versión es desde la que se ha partido. Sin embargo, Francisco Cruz llevó a cabo ciertas modificaciones en la librería (reestructuración, documentación, bugs, etc.), de entre los que sólo nos interesaban los relacionados con la reestructuración y la corrección de bugs.

Para ello se ha utilizado el comando *"git cherry-pick"*, con el que se selecciona el *commit* deseado y se incorpora a la rama actual, para posteriormente poder comprobar si los tests siguen funcionando. Este proceso se ha llevado a cabo secuencialmente para cada uno de los cambios que queríamos aplicar.

Además de las modificaciones realizadas por Francisco Cruz, se han modificado, tanto la licencia, como el fichero *package.json* y *README.md* para reflejar los nuevos contribuidores y ubicación del repositorio, así como algunos detalles que se comentarán más adelante.

Una vez integrados los cambios, se publicó la versión 1.0.0 de la librería en npm (`https://www.npmjs.com/package/genetics-js`), sin embargo, debido a la reestructuración del repositorio, aparecieron duplicidades del código generado y que fueron corregidas en parches posteriores.

#### Node

Una de las adiciones realizadas para la librería, fueron los individuos de tipo lista, y para ello, era necesario definir las estructuras de datos necesarias, como son los nodos que forman la lista. Estos tienen dos propiedades principales, una es el dato que contienen y la otra es un puntero hacia el siguiente nodo de la lista, como se muestra en la figura 2.2.

#### List

El siguiente elemento añadido fueron las propias listas, implementadas como un nodo raíz al cual se le van conectando nodos. Las listas implementadas son listas simplemente enlazadas, y por lo tanto, los nodos sólo contienen una referencia al siguiente y no a su predecesor. Además, se implementaron los métodos comunes de los objetos iterables [34] como se muestra en la figura 2.3.

#### isEqual

Debido a que en la librería sólo se habían contemplado los individuo con información simple, no se habían tenido en cuenta la posibilidad de que el genotipo del individuo

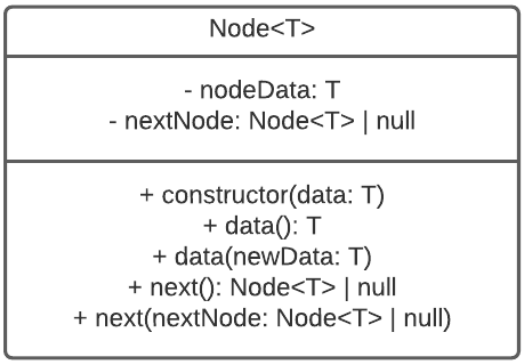


Figura 2.2: Diagrama de clases de un nodo de una lista

estuviese compuesto por objetos complejos. Es por ello que hubo que modificar diversas comprobaciones de igualdad que se realizaban en la implementación, ya que no se hacía una comparación en profundidad en el caso de los objetos. Para ello, se implementó una función recursiva que compara las claves de los objetos, llamándose recursivamente en caso de que el valor de alguna de las claves sea un objeto.

### ListIndividual

El principal motivo de estas modificaciones previas era el de añadir un nuevo tipo de individuo: los individuos de tipo lista. Estos se caracterizan porque sus genes corresponden con listas de nodos. Esta clase 2.4 implementa la clase abstracta *MutableIndividual* previamente definida en la librería, e incorpora los métodos necesarios para la misma.

### UniformListMutation

Además de añadir el nuevo tipo de individuo, se implementaron nuevos operadores para el mismo, tanto de mutación, como de cruce. Para el caso de la mutación, se implementó una clase abstracta común para los distintos operadores, que extiende a *MutationBase*. Esta clase implementa los métodos comunes para cada uno de los operadores, como se muestra en la figura 2.5. A continuación se presentan los distintos operadores desarrollados que implementan el método abstracto *mutateGeneUniformly*:

- **GeneRelocationMutation:** Este operador se aplica sobre cada gen y desplaza uno de los nodos del gen a otro.
- **GeneExchangeMutation:** A diferencia del operador anterior, este intercambia un nodo del gen actual con otro nodo de un gen distinto.
- **InnerExchangeMutation:** Para cada gen del individuo, intercambia dos nodos dentro del mismo gen.

### Crossover

El último operador implementado fue un operador de cruce denominado *NodeExchangeCrossover*, cuya idea fundamental es la de intercambiar un nodo del primer individuo

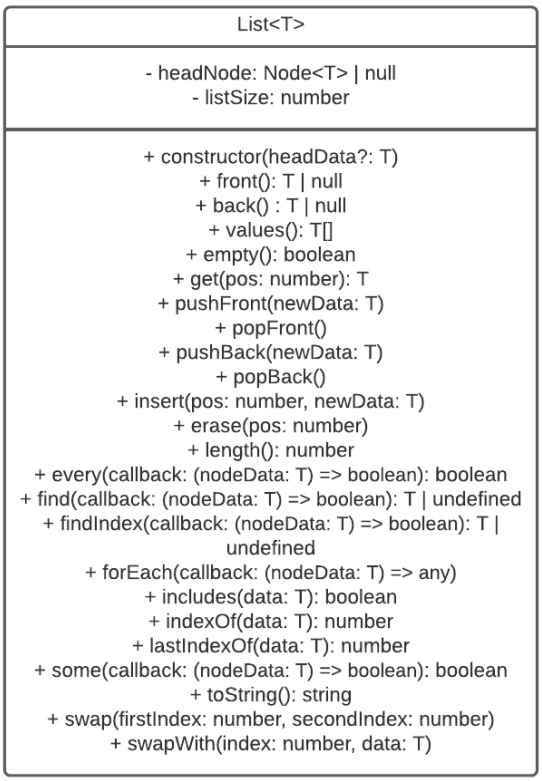


Figura 2.3: Diagrama de clases de una lista

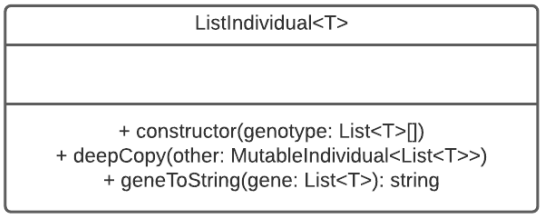


Figura 2.4: Diagrama de clases de un individuo de tipo lista

con otro nodo del segundo individuo. Para ello, en primer lugar se implementó una clase abstracta *BaseListCrossover* 2.6 de la que pueden heredar los nuevos operadores de cruce. La clase *NodeExchangeCrossover* implementa esta clase abstracta, y por lo tanto, sus métodos abstractos.

**Population**

La población ya se había implementado en la librería, sin embargo, sólo se había planteado el caso de la maximización para elegir al mejor individuo. Como el *MDCCVRP* se trata de un problema de minimización, se añadió la opción de indicarle a la población qué método debe usar para comparar a los individuos.

Además de todas estas adiciones, se incorporaron las pruebas correspondientes para comprobar su funcionamiento, así como la documentación asociada.

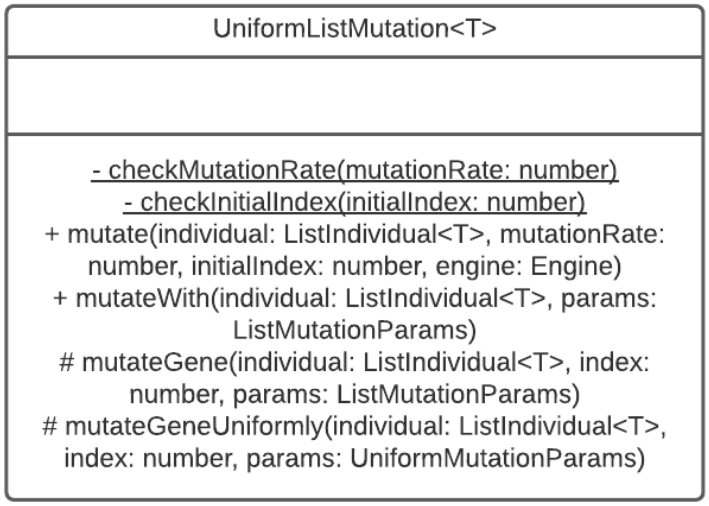


Figura 2.5: Diagrama de la clase abstracta para las mutaciones

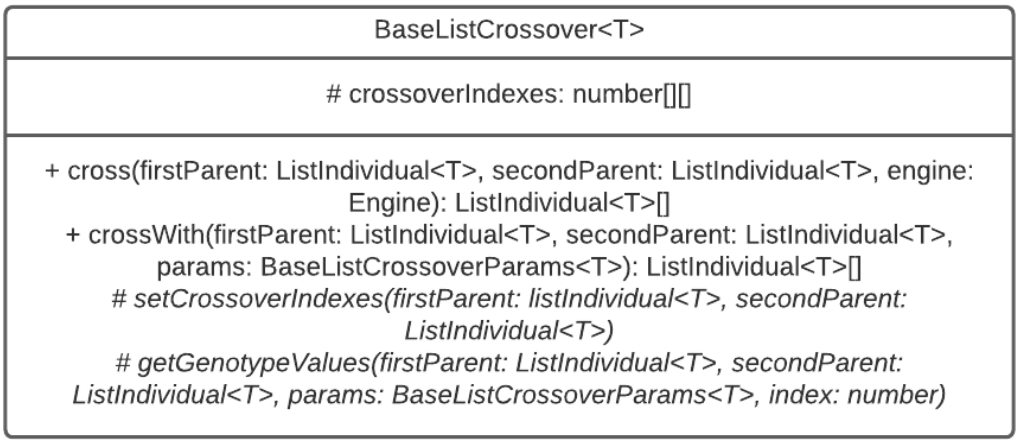


Figura 2.6: Diagrama de la clase abstracta para los operadores de cruce

### Problemas encontrados

Aparte de los problemas de duplicidad de código debidos a la reestructuración del repositorio, se han tenido que ir solucionando distintas dificultades que han ido apareciendo y que se describen a continuación.

- **Versiones de las dependencias.** A la hora de compilar el código de la librería, aparecían ciertos errores que parecían estar relacionados con la antigüedad de la versión de TypeScript que se estaba utilizando, y por lo tanto, se actualizó a una de las más recientes.
- **Formateo del código.** Prettier es una herramienta para formatear código en distintos lenguajes y es la que se utilizó en GeneticsJS. Sin embargo, al incorporar una de las modificaciones y correr Prettier, se generaba un error debido a que modificaba una línea por su longitud. Es por ello que se estableció que Prettier no debía alterar dicha línea de código.

- **Documentación.** Originalmente, la documentación estaba alojada en una página de WordPress. Sin embargo, se prefería desplegar la documentación generada mediante TypeDoc utilizando GitHub-Pages. Para llevar esto a cabo, tuvieron que realizarse diversas modificaciones:

  - **Actualización de versión.** Debido a que se utiliza una versión más reciente de TypeScript, se tuvo que actualizar la herramienta TypeDoc, para generar la documentación.
  - **Recuperación de los assets.** En alguno de los cambios introducidos, se perdieron algunas imágenes a las que se les hacía referencia en el fichero *README.md* y tuvieron que ser recuperadas y añadidas nuevamente al repositorio.
  - **Actualización de iconos.** Algunos de los iconos utilizados estaban en formato *Markdown* y, por lo tanto, cuando se generaba el fichero *html*, no aparecían correctamente. Es por ello que fueron sustituidos para que apareciesen tanto en el fichero en *Markdown*, como en *html*.
  - **Enlaces.** GiHub-Pages utiliza la herramienta *Jekyll* para construir las páginas web, y esta, por defecto, oculta todos los archivos que empiezan por "_". El problema reside en que TypeDoc genera los ficheros de los módulos empezando su nombre con "_" y por lo tanto, los enlaces de la documentación no podían redirigir al usuario a los módulos. Para solucionarlo, se modificó el script que genera la documentación para que crease, además, un fichero *.nojekyll* que le indica a GitHub que no utilice *Jekyll* a la hora de construir la página.

#### Adaptación de GeneticsJS-Knapsack

Debido a que se publicó una nueva versión de la librería GeneticsJS, se quiso probar su funcionamiento con el problema de la mochila [35] que funcionaba previamente con la versión anterior. Para que este funcionase correctamente, tuvieron que llevarse a cabo ciertas modificaciones en las referencias a archivos de la librería, ya que se modificó la estructuración de la misma.

## 2.3. Estructuración del software

En esta sección se explica cómo se ha estructurado el software desarrollado, así como las funciones que tienen cada uno de los componentes implementados y ejemplos de uso de los mismos.

### 2.3.1. Nodos

Como se comentaba previamente, los problemas de optimización de rutas de vehículos se representan mediante grafos y estos están compuestos por nodos, pudiendo ser de distintos tipos. En el caso del *MDCCVRP* tenemos dos tipos de nodos: clientes y depósitos.

Debido a que ambos tienen un comportamiento base común, se decidió implementar la jerarquía de clases representada en la figura 2.7.

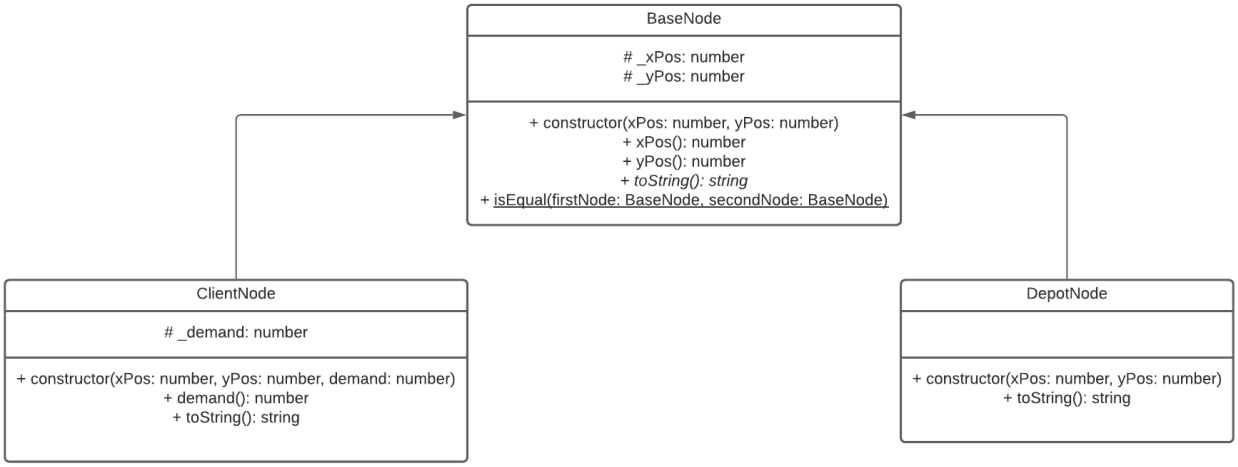


Figura 2.7: Diagrama de clases de los nodos

### *Base Node*

Esta clase representa a un nodo cualquiera del grafo, y sus únicas propiedades son las coordenadas que ocupan en el mismo. Del mismo modo define un método estático que determina si dos nodos son iguales, y un método abstracto para mostrar el nodo como un *string*.

### *Client Node*

El elemento que caracteriza a los nodos clientes es la demanda, la cual tiene que ser suplida cuando un vehículo lo visita.

### *Depot Node*

Los nodos que actúan como depósitos no tienen ninguna característica especial a nivel de código, por lo que tendrían las mismas propiedades que un nodo base.

Estos dos tipos de nodos implementan el método abstracto para representar el nodo como un *string*. A continuación se muestra un ejemplo de cómo crear un node de cada tipo y su correspondiente representación:

```
1 // Creacion de un nodo cliente (coordenada x, coordenada y, demanda)
2 const client = new ClientNode(100, 200, 50)
3
4 // Representacion: ((100, 200), 50)
5
6 // Creacion de un nodo deposito (coordenada x, coordenada y)
7 const depot = new DepotNode(50, 100)
8
9 // Representacion: (50, 100)
```

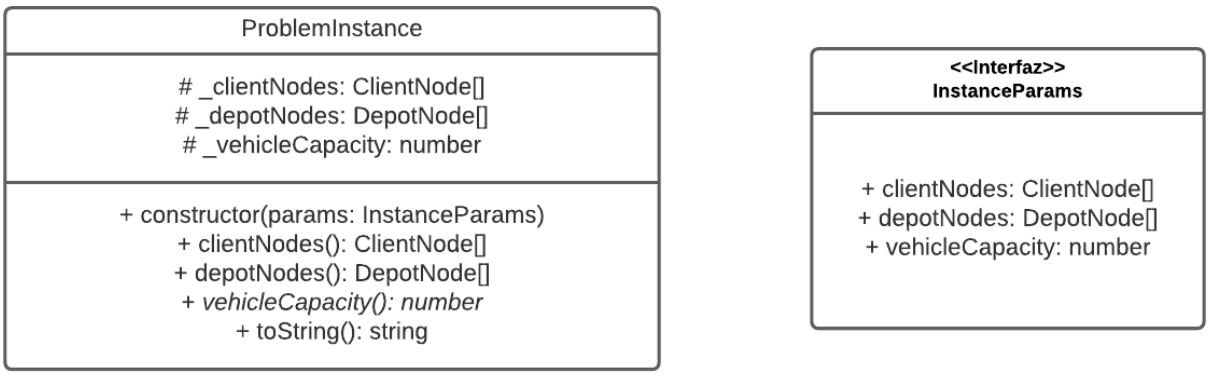


Figura 2.8: Diagrama de clases de la instancia

### 2.3.2. Instancia del problema

Para poder representar una instancia del *MDCCVRP* se ha implementado una clase propia que contiene los detalles de la misma. Por un lado, tiene la representación del grafo mediante un conjunto de nodos clientes y otro conjunto con los nodos que corresponden con los depósitos. Por otro lado, también contempla la capacidad de los vehículos ya que esta también es dependiente de la instancia del problema. En la figura 2.8 se muestra un diagrama con los distintos métodos que tiene la clase, así como la interfaz que define sus parámetros.

A continuación se presenta un ejemplo con la creación de una instancia del problema, asumiendo que los *arrays* tienen diversos nodos:

```
1  // Creacion de los arrays de nodos y la capacidad de los vehiculos
2  const clientNodes: ClientNode[] = []
3  const depotNodes: DepotNode[] = []
4  const vehicleCapacity = 200
5
6  // Se rellenan los arrays con nodos
7
8  // Creacion de los parametros
9  const instanceParams = { clientNodes, depotNodes, vehicleCapacity }
10
11 // Creacion de la instancia
12 const instance = new ProblemInstance(instanceParams)
```

### 2.3.3. Lector de instancias

Las instancias utilizadas en este estudio y los realizados en los artículos incluidos en el análisis bibliográfico tienen una estructura fija. En estas instancias se reflejan, en primer lugar, el número de nodos, tanto clientes como depósitos, junto a la capacidad de los vehículos de la flota. Como ya se comentaba previamente, el número de vehículos es independiente de la instancia y por lo tanto, no se refleja en los ficheros. Posteriormente, se presentan las coordenadas de los clientes, seguidas de las de los depósitos. Por último, se encuentran las demandas de los nodos siguiendo el mismo orden que las coordenadas.

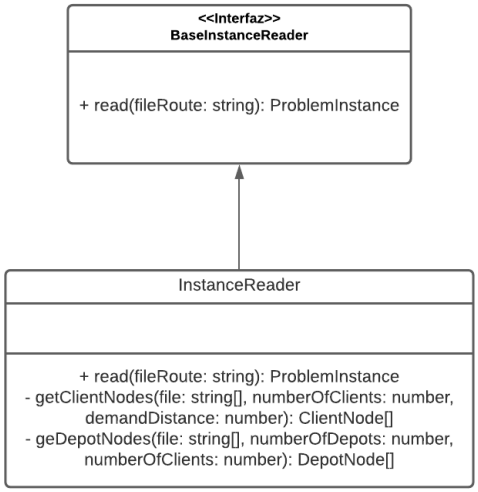


Figura 2.9: Diagrama de clases del lector de instancias

Nótese que las demandas de los depósitos siempre es cero.

A continuación se muestra un ejemplo del formato, omitiendo los datos de los nodos intermedios para ilustrar lo comentado previamente:

```
10  // Numero de clientes
4   // Numero de depositos
60  // Capacidad de los vehiculos

140.000000 30.000000    // Coordenadas del primer nodo cliente
// ...
0.000000 60.000000      // Coordenadas del ultimo nodo cliente
40.000000 110.000000    // Coordenadas del primer nodo deposito
// ...
-10.000000 -10.000000   // Coordenadas del ultimo nodo deposito

1.000000    // Demanda del primer nodo cliente
// ...
1.000000    // Demanda del ultimo nodo cliente
0.000000    // Demanda del primer nodo deposito
// ...
0.000000    // Demanda del ultimo nodo deposito
```

Para implementar el lector de instancias se ha utilizado la jerarquía mostrada en la figura 2.9. Se parte de una interfaz común para los posibles lectores, que variarían si se cambiase el formato de los ficheros. Esta interfaz sólo presenta un método público que leería un fichero y devuelve la instancia correspondiente al mismo.

### 2.3.4. Resolutor de instancias

Para resolver las instancias se ha tratado de seguir el patrón estrategia, en el que se tendría un contexto, siendo en nuestro caso el *InstanceSolver*, que recibiría una estrategia o algoritmo para resolver el problema. Como se verá en el siguiente apartado,

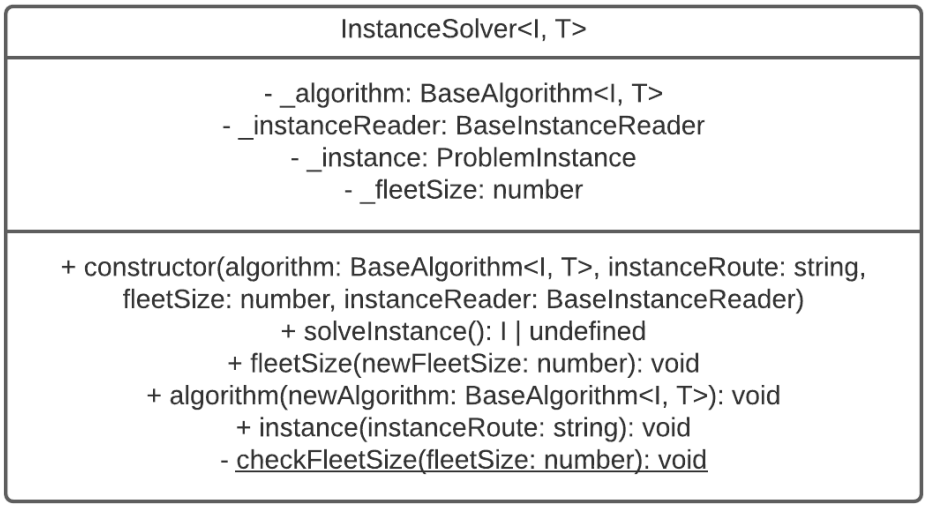


Figura 2.10: Diagrama de clases del resolutor de instancias

los algoritmos planteados se han abstraído para alcanzar este objetivo. En la figura 2.10 se muestra un diagrama de la clase que representa al resolutor.

Esta clase se va a encargar de resolver la instancia indicada utilizando el algoritmo proporcionado. Para ello, en primer lugar crea la instancia mediante el lector de instancias, para luego aplicar el algoritmo sobre la misma.

A continuación se muestra un pequeño ejemplo en el que se inicializan los parámetros del resolutor y se se resuelve una instancia concreta del problema:

```
const algorithm = new SimpleMemetic(algorithmParams);
const instanceReader = new InstanceReader();
const fleetSize = 5;
const instance = 'instance.txt';
const solver = new InstanceSolver(
    algorithm,
    instance,
    fleetSize,
    instanceReader
);
```

### 2.3.5. Algoritmos

Para el desarrollo del proyecto se han implementado dos algoritmos distintos. Sin embargo, sólo el algoritmo memético ofrece unos resultados adecuados, ya que la funcionalidad del algoritmo aleatorio era la de comprobar si las implementaciones realizadas para el modelado del problema y los operadores básicos eran correctos.

### Algoritmo aleatorio

Como se comentaba previamente, con el algoritmo aleatorio no se buscaba obtener soluciones óptimas para el problema, sino comprobar que el modelado del problema era adecuado y que los operadores implementados en la librería para individuos de tipo lista estaban implementados correctamente.

Más concretamente, el operador utilizado es el de mutación, ya que el algoritmo consiste en mutar un individuo inicializado aleatoriamente hasta que se cumpla la condición de parada y reemplazándolo en caso de que el nuevo individuo sea mejor. Además, se aplica una operación de reparación en caso de que la solución obtenida tras la mutación no sea factible. Esta operación busca la primera ruta que exceda la capacidad de los vehículos y se recoloca un nodo aleatorio en otra ruta en la que no se haya superado dicha capacidad.

Debido a la naturaleza del algoritmo aleatorio y del operador de reparación parcial, no siempre se alcanzan soluciones factibles. Esto viene derivado de que hay rutas de las soluciones en las que se supera la capacidad máxima establecida para los vehículos. Esto podría evitarse modificando el operador de reparación, sin embargo, debido a que el objetivo del mismo no era el de obtener soluciones óptimas y factibles (como se comentaba previamente), se decidió no dedicar un mayor esfuerzo en el algoritmo.

A continuación se muestra el pseudocódigo correspondiente al algoritmo aleatorio:

**Algoritmo 2:** Aleatorio

```
1 Inicializar el primer individuo
2 Actual = Inicial
3 while Condición de parada do
4 |   Nuevo = Mutar Actual
5 |   if Nuevo no es factible then
6 |   |   Reparar Nuevo
7 |   if Nuevo mejor que Actual then
8 |   |   Actual = Nuevo
9 return Actual
```

### Algoritmo memético

A diferencia del algoritmo aleatorio, mediante el algoritmo memético se buscaban obtener las mejores soluciones posibles; pudiendo, si fuese posible, compararlas con los resultados obtenidos en los artículos previos sobre el *MDCCVRP*.

La idea fundamental del algoritmo memético, como se comentaba en apartados anteriores, es la de partir de un conjunto de soluciones o individuos, que se cruzan entre ellos y se les aplica una mejora mediante una búsqueda local. El algoritmo está completamente parametrizado, por lo que existen multitud de elementos del mismo que se pueden personalizar y en el apartado siguiente, se entrará en mayor detalle sobre los mismos. Sin embargo, se va a hacer una breve descripción de dichos componentes para ilustrar el funcionamiento básico del algoritmo.

Al tratarse de un algoritmo evolutivo, se trabaja con una población de individuos inicializada con un mecanismo aleatorio o constructivo en este caso. Un aspecto importante de la población es que los individuos no sólo tienen un valor objetivo, sino que también tienen un grado de insatisfacibilidad, cuyo valor aumenta con las restricciones que incumple; siendo en este caso, que se supere la capacidad máxima de los vehículos o no se visite algún nodo. Además, cuantas más rutas que superen la capacidad o nodos que no son visitados, mayor será el grado de insatisfacibilidad. Por lo tanto, para determinar si un individuo es mejor que otro, se compararía en primer lugar el grado de insatisfacibilidad, eligiendo el de menor valor y en caso de tener el mismo grado, se comprobaría el valor de la función objetivo, escogiéndose el más pequeño debido a que nos encontramos en un problema de minimización.

Sobre la primera población se aplica un procedimiento de mejora haciendo uso de una búsqueda aleatoria, que intenta mejorar la solución dada. Posteriormente, se repite el siguiente procedimiento hasta que se cumpla la condición de parada. Se genera la descendencia a partir de todo el conjunto de la población; para ello se seleccionan dos individuos mediante un torneo binario (se escoge el mejor entre dos opciones aleatorias) y se les aplica un operador de cruce dependiendo de la probabilidad de cruce establecida, dando lugar a dos descendientes. Esto se repite hasta que la descendencia tenga el mismo tamaño que la población actual, y pudiéndose dar el caso de que se seleccione más de una vez al mismo individuo. Posteriormente, se mejoran las soluciones obtenidas del cruce o los padres, en caso de que no se aplique el operador de cruce.

Una vez obtenida la descendencia, se reemplaza la población siguiendo un mecanismo de remplazo generacional con elitismo. Esto consiste en seleccionar el mejor individuo entre la población actual y la descendencia e introduciéndolo en la nueva población. A continuación, se van insertando los mejores individuos de la descendencia en la nueva población hasta que alcance el tamaño adecuado.

Finalmente, una vez se cumpla la condición de parada, se devuelve el mejor individuo de la población, que va a coincidir con el mejor individuo encontrado durante toda la ejecución al haber utilizado el reemplazo con elitismo.

A continuación se muestra el pseudocódigo del algoritmo memético:

**Algoritmo 3:** Memético

```
1 Inicializar la población con el mecanismo seleccionado
2 Aplicar mejora sobre la población
3 while Condición de parada do
4 |   Generar descendencia
5 |   Aplicar mejora sobre la descendencia
6 |   Reemplazar la población
7 return Mejor individuo de la población
```

### 2.3.6. Componentes de los algoritmos

En este apartado se presentan los distintos elementos que componen los algoritmos comentados previamente, siendo muchos de ellos, parametrizables a la hora de ejecutar el algoritmo.

#### Constructor de individuos

Es el encargado de generar los individuos que forman parte de la primera generación. Para este proyecto se han implementado dos mecanismos, uno aleatorio y uno constructivo, siguiendo el método utilizado en el artículo *"An effective local search algorithm for the multidepot cumulative capacitated vehicle routing problem"* [25].

- **Aleatorio:** Este mecanismo introduce en primer lugar un depósito aleatorio para cada posible ruta y, posteriormente, va introduciendo de forma aleatoria los nodos clientes en las distintas rutas.
- **Constructivo:** Al igual que el aleatorio, introduce los depósitos de forma aleatoria en cada una de las rutas. A continuación, como se comentaba en el artículo [25], se van introduciendo los nodos en aquellas rutas cuyo *"valor de arrepentimiento"* es menor. Este valor se calcula obteniendo, en primer lugar, los costes de inserción de un nodo para cada ruta y ordenándolos de menor a mayor. Una vez ordenados, el *"valor de arrepentimiento"* va a ser igual a la diferencia entre el tercer y el primer coste de inserción.

Para generar la población inicial, habría que generar tantos individuos como tamaño tenga la población mediante uno de estos mecanismos e ir introduciéndolos en la misma, pasando como parámetros la instancia a resolver y el tamaño de la flota de vehículos, como se muestra a continuación.

```
1
2 const fleetSize = 5;
3 const newIndividual = randomInitialIndividualGenerator(instance, fleetSize);
```

#### Operador de cruce

En la librería *GeneticsJS* existían distintos operadores de cruce que se pueden aplicar a los individuos de tipo lista, como se comentaba en secciones anteriores. Sin embargo, se decidió implementar un operador más específico para el problema.

Para ello, se partió del operador conocido como *Order Crossover* utilizado en el artículo *A simple and effective evolutionary algorithm for the vehicle routing problem* [36] donde se resuelve un problema de optimización de rutas de vehículos aplicando un algoritmo evolutivo. Este operador mantenía parte de la solución fija, de tal forma que si tenemos dos padres *P1* y *P2*, el primer hijo *C1* tendría el fragmento correspondiente a *P1* y el segundo hijo *C2*, el del padre restante. Posteriormente, se van rellenando los hijos con los nodos que les faltan utilizando el orden del otro padre.

De esta forma, el operador planteado selecciona una de las rutas para mantenerlas fijas en ambos individuos. A continuación, se van introduciendo los nodos que aparecen en

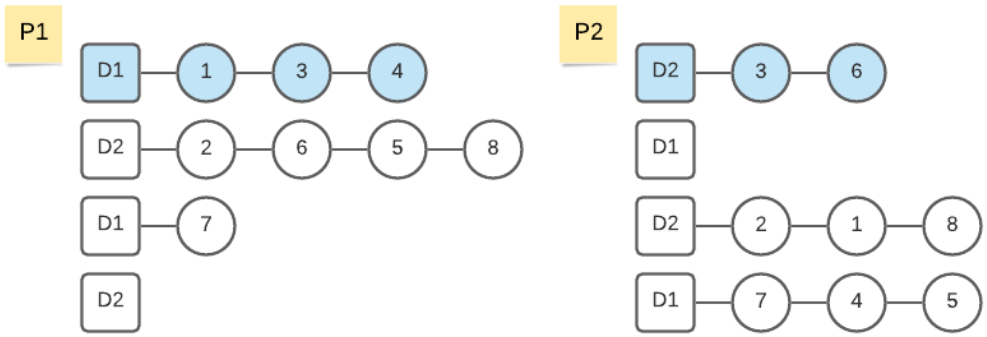


Figura 2.11: Individuos para aplicar el operador de cruce

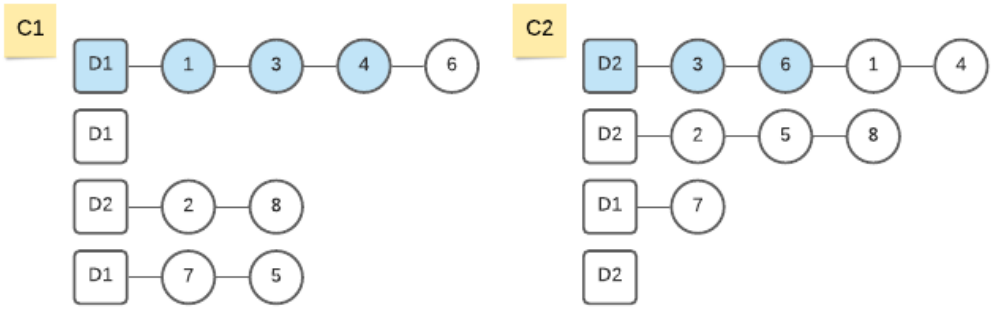


Figura 2.12: Individuos resultantes del operador de cruce

cada ruta del *P1* en el *C2*, siguiendo el orden establecido por las rutas del primer padre, pero introduciendo sólo aquellos que no se encuentran ya en *C2*. Lo mismo ocurriría con el *P2* y el *C1*. En la figura 2.11 se muestra un ejemplo de dos individuos sobre los que se aplica el operador de cruce, que selecciona la primera ruta como fija, teniendo en cuenta que hay una flota de cuatro vehículos, dos depósitos y ocho clientes. Los individuos obtenidos tras aplicar el operador se pueden ver en la figura 2.12.

### Función objetivo

Durante la ejecución del algoritmo se permite que aparezcan soluciones no factibles donde la capacidad del vehículo se sobrepasa, por lo tanto, a la hora de calcular el valor objetivo de un individuo, hay que penalizar aquellos que no cumplen con las restricciones establecidas. Para penalizarlas se le establece a cada individuo un valor que representa su grado de infactibilidad y que a la hora de comparar individuos, va a tener mayor prioridad que el valor objetivo, de tal forma que un individuo con mayor grado de infactibilidad siempre va a ser peor que uno con menor grado, independientemente de su valor objetivo.

Para calcular el valor objetivo se calculan los tiempos de llegada acumulados para cada nodo cliente, siendo el tiempo que tarda en llegar un vehículo de un nodo a otro, la distancia euclídea entre ambos. Por otro lado, el grado de infactibilidad se calcula multiplicando el número de vehículos cuya capacidad es sobrepasada por $10^3$ junto al número de nodos que no se visitan (este caso no debería darse por la naturaleza de los operadores empleados) por $10^6$.

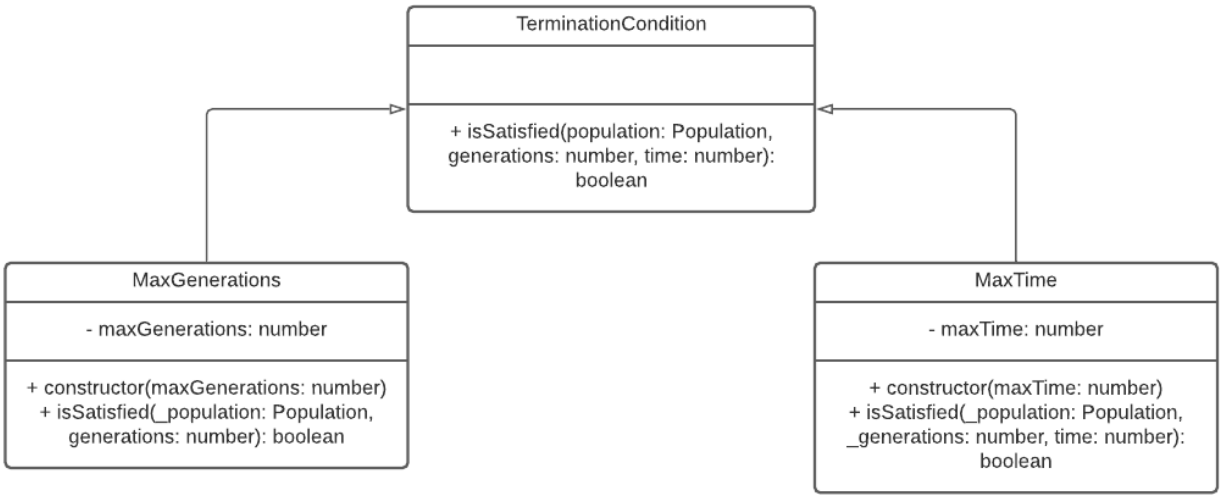


Figura 2.13: Diagrama de clases de la condición de parada

### Población

La librería *GeneticsJS* ofrece una implementación para una población de individuos. Sin embargo, esta implementación sólo tenía en cuenta que los individuos tenían únicamente un valor objetivo, pero como se comentaba con anterioridad, en nuestro caso, los individuos también poseen un grado de infactibilidad. Por lo tanto, fue necesario implementar la población contemplando este factor, pero siguiendo la metodología previa, en la que cada vez que se introduce un nuevo individuo en la población, se actualiza el mejor detectado en caso de que el nuevo individuo lo supere. Además, se incorporó el mecanismo para el reemplazo con elitismo, el cual mantiene en la población al mejor individuo entre la población y la descendencia y rellena la población con los mejores individuos de la descendencia. En el caso de que el mejor individuo se encuentre en la descendencia, la población será reemplazada por completo.

### Condición de parada

Como condiciones de parada se han implementado dos mecanismos, por un lado, por un número máximo de generaciones, y por el otro, por tiempo. La implementación de ambas condiciones es trivial debido a que la comprobación necesaria solamente requiere comparar dos de los parámetros recibidos. La jerarquía de clases asociada se puede apreciar en la figura 2.13.

### Búsqueda local

La búsqueda local implementada para mejorar un individuo está basada en la propuesta en el artículo *An Effective Local Search Algorithm for the Multidepot Cumulative Capacitated Vehicle Routing Problem* [25]. Esta búsqueda local consta de seis movimientos para generar los vecindarios, en los que entraremos en detalle más adelante. La búsqueda se va a ejecutar mientras se consiga mejorar el individuo, de tal forma que si aplicando un operador, se obtiene una mejor solución, se volvería a comenzar a aplicar el primer operador sobre el nuevo individuo. En caso de que no se mejore, se pasa al siguiente operador hasta que ninguno consiga una mejora y entonces se finaliza el algoritmo. A

continuación se muestra el pseudocódigo de la búsqueda:

**Algoritmo 4:** Búsqueda Local

```
1  S = Individuo a mejorar
2  I = 0
3  while I <Número de movimientos do
4  |   Generar vecindario
5  |   Nuevo = Mejor individuo del vecindario
6  |   if Nuevo es mejor que S then
7  |   |   S = Nuevo
8  |   |   I = 0
9  |   else
10 |   |   I++
11 return S
```

Los movimientos implementados para la búsqueda local son los siguientes:

- **ArcExchange:** Intercambia dos arcos entre dos rutas o la misma ruta. En caso de que sean distintas, también se intercambian los nodos posteriores a los arcos. Por otro lado, si ambos arcos están en le misma ruta, se invierte el orden de los nodos que se encuentren entre ellos.
- **ArcNodeExchange:** Intercambia un nodo por un arco pertenecientes a rutas distintas.
- **CrossExchange:** Se intercambian dos segmentos de tamaños arbitrarios entre dos rutas distintas.
- **Exchange:** Intercambia dos nodos clientes o dos depósitos entre rutas distintas.
- **OrOpt:** Mueve un segmento de tres nodos clientes a una nueva posición en una ruta distinta o en la misma. En caso de que se produzca en la misma ruta, el orden del segmento es invertido.
- **Relocation:** Desplaza un nodo cliente de una ruta a otra distinta.

#### Búsqueda local iterada

A parte de la búsqueda local, se implementó una búsqueda local iterada cuya característica es que ejecuta la búsqueda local un número de iteraciones determinado. Sin embargo, debido a que el tiempo que consume es muy elevado, se decidió no utilizarla para los experimentos.

## 2.4. Modo de uso

Para permitir la ejecución de los algoritmos se han implementado diversos comandos que se pueden ejecutar en la línea de comandos (*CLI*) utilizando la librería *Yargs*. De esta forma, se dispone del comando *"memetic"* para ejecutar el algoritmo memético y del comando *random"* para ejecutar el algoritmo aleatorio. Debido a que los algoritmos son

```
npm run solve -- memetic -c --cr 0.8 -p 50 --ns 1 -t 20 -f instances/10x4-1.txt --fs 5
```

Figura 2.14: Ejemplo de ejecución en la línea de comandos

parametrizables, se han establecido diversos parámetros para que se pueda personalizar la ejecución del algoritmo.

A continuación se muestran los parámetros para el algoritmo memético:

- ***f:*** Parámetro obligatorio que indica la ubicación de la instancia a ejecutar.
- ***fs:*** Parámetro obligatorio que especifica el tamaño de la flota de vehículos.
- ***cr:*** Probabilidad de aplicar el operador de cruce. Tiene como valor por defecto 0,8.
- ***p:*** Tamaño de la población. Tiene como valor por defecto 20.
- ***c:*** *Booleano* que indica si el método de inicialización es constructivo.
- ***r:*** *Booleano* que indica si el método de inicialización de individuos es aleatorio. Si no se especifica el tipo, se utiliza el método aleatorio por defecto.
- ***ls:*** Indica el tipo de búsqueda local, pudiendo tomar los valores *LS* o *ILS*, siendo por defecto *LS*.
- ***ns:*** Tamaño del vecindario a generar en la búsqueda local. Tiene como valor por defecto 5.
- ***i:*** Número de iteraciones para la búsqueda local iterada. Realiza 10 iteraciones por defecto.
- ***g:*** Número máximo de generaciones para el algoritmo. Por defecto finaliza la ejecución tras 100 generaciones.
- ***t:*** Tiempo máximo para ejecutar el algoritmo. Si se especifica, no se tiene en cuenta el número de generaciones aunque se haya indicado previamente.

Para el caso del algoritmo aleatorio, sólo se podrían indicar tres parámetros: la ruta de la instancia, el tamaño de la flota de vehículos y el número máximo de generaciones para ejecutar el algoritmo, manteniendo las mismas propiedades de obligatoriedad y nomenclatura que para el caso del algoritmo memético.

En la figura 2.14 se muestra un ejemplo de cómo se podría lanzar una ejecución desde la raíz del proyecto y haciendo uso del script *"solve"* declarado en el *package.json*.

# Capítulo 3

## Experimentos y resultados

En este apartado se presentan los resultados obtenidos del estudio realizado con el algoritmo memético. Para ello se han definido un total de 36 configuraciones para los parámetros del algoritmo, tal como se muestra en la tabla 3.1. El experimento consistió en ejecutar, para las 18 instancias propuestas en el artículo [5], 30 repeticiones para cada configuración.

Para cada ejecución se utilizó como condición de parada el tiempo de ejecución, siendo, para cada instancia el siguiente:

- **Instancias con 10 nodos:** 5 minutos
- **Instancias con 25 nodos:** 5 minutos
- **Instancias con 50 nodos:** 10 minutos
- **Instancias con 100 nodos:** 20 minutos

Como se puede comprobar en las siguientes tablas, para las instancias de menor tamaño (10 y 25 nodos), se alcanza para alguna de las configuraciones la mejor solución obtenida en estudios previos. Sin embargo, para el resto de instancias, no se alcanza la mejor solución previa, pero se acerca al resultado. Cabe destacar que para la instancia *100x6x25-1* se obtuvo una nueva solución que reduce el valor objetivo previo, como se puede ver en la tabla 3.18. Además, se ha preparado una gráfica que se puede ver en la figura 3.1 donde se muestra la evolución de la mejor solución para dicha instancia.

Si observamos los resultados obtenidos por el algoritmo propuesto en el artículo *A POPMUSIC approach for the Multi-Depot Cumulative Capacitated Vehicle Routing Problem* [5], podemos concluir que el algoritmo memético tiene un mejor rendimiento. Por otro lado, el propuesto en *An effective local search algorithm for the Multidepot Cumulative Capacitated Vehicle Routing Problem* [25] obtiene mejores resultados para las instancias grandes, salvo para el caso comentado previamente.

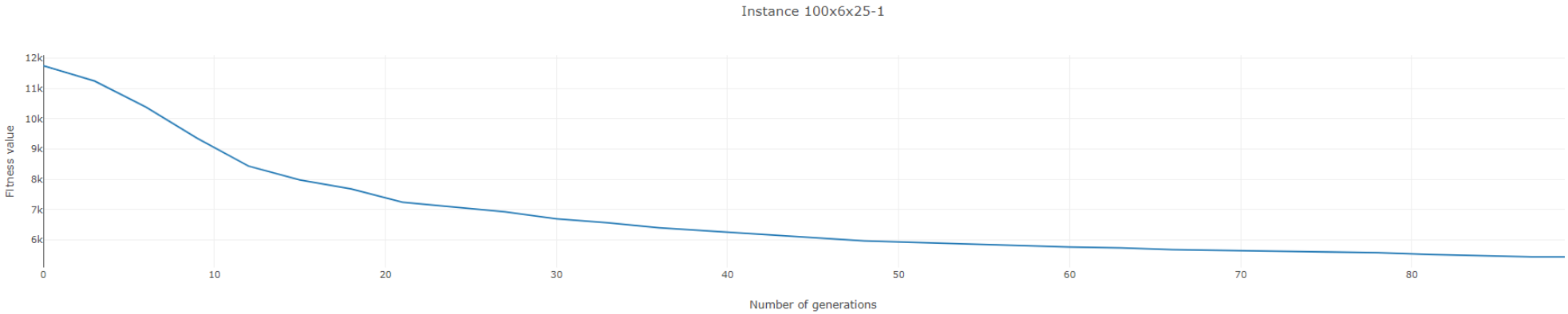


Figura 3.1: Gráfica para la instancia *100x6x25-1*

| Configuración | Inicializador | Probabilidad de cruce | Tamaño de la población | Tamaño del vecindario |
|---|---|---|---|---|
| Conf1 | Constructivo | 0,8 | 50 | 1 |
| Conf2 | Constructivo | 0,8 | 50 | 10 |
| Conf3 | Constructivo | 0,8 | 100 | 1 |
| Conf4 | Constructivo | 0,8 | 100 | 10 |
| Conf5 | Constructivo | 0,8 | 150 | 1 |
| Conf6 | Constructivo | 0,8 | 150 | 10 |
| Conf7 | Constructivo | 0,9 | 50 | 1 |
| Conf8 | Constructivo | 0,9 | 50 | 10 |
| Conf9 | Constructivo | 0,9 | 100 | 1 |
| Conf10 | Constructivo | 0,9 | 100 | 10 |
| Conf11 | Constructivo | 0,9 | 150 | 1 |
| Conf12 | Constructivo | 0,9 | 150 | 10 |
| Conf13 | Constructivo | 1 | 50 | 1 |
| Conf14 | Constructivo | 1 | 50 | 10 |
| Conf15 | Constructivo | 1 | 100 | 1 |
| Conf16 | Constructivo | 1 | 100 | 10 |
| Conf17 | Constructivo | 1 | 150 | 1 |
| Conf18 | Constructivo | 1 | 150 | 10 |
| Conf19 | Aleatorio | 0,8 | 50 | 1 |
| Conf20 | Aleatorio | 0,8 | 50 | 10 |
| Conf21 | Aleatorio | 0,8 | 100 | 1 |
| Conf22 | Aleatorio | 0,8 | 100 | 10 |
| Conf23 | Aleatorio | 0,8 | 150 | 1 |
| Conf24 | Aleatorio | 0,8 | 150 | 10 |
| Conf25 | Aleatorio | 0,9 | 50 | 1 |
| Conf26 | Aleatorio | 0,9 | 50 | 10 |
| Conf27 | Aleatorio | 0,9 | 100 | 1 |
| Conf28 | Aleatorio | 0,9 | 100 | 10 |
| Conf29 | Aleatorio | 0,9 | 150 | 1 |
| Conf30 | Aleatorio | 0,9 | 150 | 10 |
| Conf31 | Aleatorio | 1 | 50 | 1 |
| Conf32 | Aleatorio | 1 | 50 | 10 |
| Conf33 | Aleatorio | 1 | 100 | 1 |
| Conf34 | Aleatorio | 1 | 100 | 10 |
| Conf35 | Aleatorio | 1 | 150 | 1 |
| Conf36 | Aleatorio | 1 | 150 | 10 |

Tabla 3.1: Configuraciones para el algoritmo

| 10x4x5-1 | | | **Mejor encontrado: 545.69** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | **545,69** | 552,24 | 546,35 | **545,69** |
| Conf2 | 545,69 | **545,69** | **545,69** | 545,69 |
| Conf3 | 545,69 | 552,24 | 546,35 | 545,69 |
| Conf4 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf5 | 545,69 | 574,87 | 550,04 | 545,69 |
| Conf6 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf7 | 545,69 | 574,87 | 547,97 | 545,69 |
| Conf8 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf9 | 545,69 | 552,24 | 546,56 | 545,69 |
| Conf10 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf11 | 545,69 | 574,87 | 549,82 | 545,69 |
| Conf12 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf13 | 545,69 | 574,87 | 549,06 | 545,69 |
| Conf14 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf15 | 545,69 | 552,24 | 547,00 | 545,69 |
| Conf16 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf17 | 545,69 | 574,87 | 548,63 | 545,69 |
| Conf18 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf19 | 545,69 | 574,87 | 552,00 | 552,24 |
| Conf20 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf21 | 545,69 | 552,24 | 548,53 | 545,69 |
| Conf22 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf23 | 545,69 | 589,97 | 565,44 | 563,56 |
| Conf24 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf25 | 545,69 | 574,87 | 551,78 | 552,24 |
| Conf26 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf27 | 545,69 | 574,87 | 552,76 | 552,24 |
| Conf28 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf29 | 545,69 | 589,97 | 559,39 | 552,24 |
| Conf30 | 545,69 | 552,24 | 546,56 | 545,69 |
| Conf31 | 545,69 | 574,87 | 556,96 | 552,24 |
| Conf32 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf33 | 545,69 | 574,87 | 553,07 | 552,24 |
| Conf34 | 545,69 | 545,69 | 545,69 | 545,69 |
| Conf35 | 545,69 | 589,97 | 559,04 | 552,24 |
| Conf36 | 545,69 | 552,24 | 545,91 | 545,69 |

Tabla 3.2: Resultados de la 1ª instancia

| 10x4x5-2 | | | **Mejor encontrado: 832.69** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | **832,69** | **832,69** | **832,69** | **832,69** |
| Conf2 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf3 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf4 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf5 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf6 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf7 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf8 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf9 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf10 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf11 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf12 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf13 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf14 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf15 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf16 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf17 | 832,69 | 840,62 | 832,95 | 832,69 |
| Conf18 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf19 | 832,69 | 840,62 | 833,22 | 832,69 |
| Conf20 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf21 | 832,69 | 840,62 | 833,22 | 832,69 |
| Conf22 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf23 | 832,69 | 885,12 | 843,96 | 832,69 |
| Conf24 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf25 | 832,69 | 840,62 | 833,48 | 832,69 |
| Conf26 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf27 | 832,69 | 840,62 | 833,75 | 832,69 |
| Conf28 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf29 | 832,69 | 873,66 | 837,85 | 832,69 |
| Conf30 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf31 | 832,69 | 885,12 | 838,54 | 832,69 |
| Conf32 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf33 | 832,69 | 862,55 | 837,00 | 832,69 |
| Conf34 | 832,69 | 832,69 | 832,69 | 832,69 |
| Conf35 | 832,69 | 875,49 | 841,11 | 832,69 |
| Conf36 | 832,69 | 832,69 | 832,69 | 832,69 |

Tabla 3.3: Resultados de la 2ª instancia

| 10x4x5-3 | | | **Mejor encontrado: 832.78** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | **832,78** | 836,91 | 833,33 | **832,78** |
| Conf2 | 832,78 | **832,78** | **832,78** | 832,78 |
| Conf3 | 832,78 | 836,91 | 832,92 | 832,78 |
| Conf4 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf5 | 832,78 | 836,91 | 833,24 | 832,78 |
| Conf6 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf7 | 832,78 | 836,91 | 832,94 | 832,78 |
| Conf8 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf9 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf10 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf11 | 832,78 | 836,91 | 833,80 | 832,78 |
| Conf12 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf13 | 832,78 | 836,91 | 832,97 | 832,78 |
| Conf14 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf15 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf16 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf17 | 832,78 | 843,01 | 834,43 | 832,78 |
| Conf18 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf19 | 832,78 | 836,91 | 833,05 | 832,78 |
| Conf20 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf21 | 832,78 | 833,55 | 832,83 | 832,78 |
| Conf22 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf23 | 832,78 | 876,50 | 835,69 | 833,17 |
| Conf24 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf25 | 832,78 | 836,91 | 832,99 | 832,78 |
| Conf26 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf27 | 832,78 | 833,55 | 832,80 | 832,78 |
| Conf28 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf29 | 832,78 | 843,01 | 833,99 | 832,78 |
| Conf30 | 832,78 | 836,91 | 832,92 | 832,78 |
| Conf31 | 832,78 | 833,55 | 832,80 | 832,78 |
| Conf32 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf33 | 832,78 | 833,55 | 832,80 | 832,78 |
| Conf34 | 832,78 | 832,78 | 832,78 | 832,78 |
| Conf35 | 832,78 | 836,91 | 833,30 | 832,78 |
| Conf36 | 832,78 | 832,78 | 832,78 | 832,78 |

Tabla 3.4: Resultados de la 3ª instancia

| 25x4x10-1 | | | **Mejor encontrado: 2082.28** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | **2082,28** | 2179,13 | 2114,82 | 2108,80 |
| Conf2 | 2082,28 | 2112,18 | 2089,62 | 2085,38 |
| Conf3 | 2082,28 | 2143,46 | 2109,77 | 2106,88 |
| Conf4 | 2082,28 | 2106,79 | 2085,48 | **2082,28** |
| Conf5 | 2082,28 | 2183,41 | 2120,51 | 2113,48 |
| Conf6 | 2082,28 | 2125,06 | 2098,37 | 2099,96 |
| Conf7 | 2082,28 | 2202,15 | 2121,43 | 2122,71 |
| Conf8 | 2082,28 | 2104,50 | 2087,21 | 2082,28 |
| Conf9 | 2082,28 | 2163,54 | 2108,13 | 2105,89 |
| Conf10 | 2082,28 | 2106,22 | 2084,54 | 2082,28 |
| Conf11 | 2082,28 | 2199,59 | 2126,44 | 2117,00 |
| Conf12 | 2082,28 | 2116,02 | 2096,98 | 2098,04 |
| Conf13 | 2088,49 | 2191,97 | 2115,43 | 2106,22 |
| Conf14 | 2082,28 | 2106,12 | 2085,84 | 2082,28 |
| Conf15 | 2082,28 | 2148,97 | 2105,07 | 2102,99 |
| Conf16 | 2082,28 | **2088,49** | **2082,79** | 2082,28 |
| Conf17 | 2088,49 | 2215,01 | 2136,09 | 2123,70 |
| Conf18 | 2082,28 | 2124,47 | 2092,82 | 2088,49 |
| Conf19 | 2105,21 | 2203,84 | 2138,79 | 2127,63 |
| Conf20 | 2082,28 | 2115,52 | 2091,70 | 2088,49 |
| Conf21 | 2082,28 | 2196,05 | 2123,55 | 2115,23 |
| Conf22 | 2082,28 | 2101,48 | 2085,37 | 2082,28 |
| Conf23 | 2100,01 | 2212,46 | 2148,22 | 2146,91 |
| Conf24 | 2082,28 | 2144,54 | 2106,27 | 2105,52 |
| Conf25 | 2082,28 | 2205,78 | 2135,61 | 2124,90 |
| Conf26 | 2082,28 | 2097,79 | 2084,03 | 2082,28 |
| Conf27 | 2082,28 | 2213,86 | 2118,56 | 2110,93 |
| Conf28 | 2082,28 | 2094,45 | 2083,72 | 2082,28 |
| Conf29 | 2082,28 | 2255,03 | 2150,23 | 2143,56 |
| Conf30 | 2082,28 | 2125,33 | 2098,56 | 2099,98 |
| Conf31 | 2082,28 | 2247,82 | 2138,57 | 2127,12 |
| Conf32 | 2082,28 | 2109,15 | 2087,41 | 2082,28 |
| Conf33 | 2082,28 | 2200,79 | 2119,20 | 2110,45 |
| Conf34 | 2082,28 | 2088,49 | 2083,71 | 2082,28 |
| Conf35 | 2082,28 | 2287,79 | 2144,58 | 2135,67 |
| Conf36 | 2082,28 | 2138,25 | 2095,94 | 2097,20 |

Tabla 3.5: Resultados de la 4ª instancia

| 25x4x10-2 | | | **Mejor encontrado: 1827.41** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | **1827,41** | 2019,30 | 1870,70 | 1862,09 |
| Conf2 | 1827,41 | 1856,04 | **1832,67** | **1829,48** |
| Conf3 | 1827,41 | 1892,37 | 1859,52 | 1856,04 |
| Conf4 | 1827,41 | 1856,04 | 1834,47 | 1829,48 |
| Conf5 | 1829,48 | 2001,48 | 1891,65 | 1892,72 |
| Conf6 | 1827,41 | 1926,77 | 1848,39 | 1844,86 |
| Conf7 | 1827,41 | 1948,72 | 1862,22 | 1858,10 |
| Conf8 | 1827,41 | 1858,10 | 1835,63 | 1829,48 |
| Conf9 | 1827,41 | 1925,89 | 1855,79 | 1851,48 |
| Conf10 | 1827,41 | **1846,92** | 1834,55 | 1829,48 |
| Conf11 | 1827,41 | 1977,12 | 1876,33 | 1862,09 |
| Conf12 | 1827,41 | 1880,90 | 1841,71 | 1840,93 |
| Conf13 | 1827,41 | 1952,88 | 1870,20 | 1862,09 |
| Conf14 | 1827,41 | 1867,66 | 1840,66 | 1844,86 |
| Conf15 | 1827,41 | 1919,17 | 1852,38 | 1851,48 |
| Conf16 | 1827,41 | 1852,28 | 1837,69 | 1844,86 |
| Conf17 | 1827,41 | 1999,40 | 1890,64 | 1887,93 |
| Conf18 | 1827,41 | 1869,72 | 1845,36 | 1846,92 |
| Conf19 | 1827,41 | 2005,67 | 1874,83 | 1857,07 |
| Conf20 | 1827,41 | 1852,28 | 1835,94 | 1829,48 |
| Conf21 | 1827,41 | 1950,63 | 1866,96 | 1855,19 |
| Conf22 | 1827,41 | 1852,28 | 1834,48 | 1829,48 |
| Conf23 | 1827,41 | 1993,43 | 1900,27 | 1907,80 |
| Conf24 | 1827,41 | 1890,31 | 1847,85 | 1846,92 |
| Conf25 | 1827,41 | 1986,12 | 1885,62 | 1868,97 |
| Conf26 | 1827,41 | 1867,66 | 1838,86 | 1844,86 |
| Conf27 | 1827,41 | 1950,08 | 1888,24 | 1892,11 |
| Conf28 | 1827,41 | 1867,66 | 1836,70 | 1829,48 |
| Conf29 | 1827,41 | 2066,16 | 1905,82 | 1900,84 |
| Conf30 | 1827,41 | 1885,54 | 1845,95 | 1846,92 |
| Conf31 | 1827,41 | 2029,71 | 1891,71 | 1885,61 |
| Conf32 | 1827,41 | 1846,92 | 1836,10 | 1829,48 |
| Conf33 | 1827,41 | 1982,25 | 1892,10 | 1895,50 |
| Conf34 | 1827,41 | 1892,11 | 1833,87 | 1829,48 |
| Conf35 | 1827,41 | 2048,69 | 1923,28 | 1913,78 |
| Conf36 | 1827,41 | 1869,72 | 1840,82 | 1844,86 |

Tabla 3.6: Resultados de la 5ª instancia

| 25x4x10-3 | | | **Mejor encontrado: 1786.95** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | **1786,95** | 1891,42 | 1840,85 | 1835,87 |
| Conf2 | 1786,95 | 1813,36 | 1791,58 | 1786,95 |
| Conf3 | 1786,95 | 1873,28 | 1828,71 | 1828,35 |
| Conf4 | 1786,95 | 1803,52 | 1788,71 | **1786,95** |
| Conf5 | 1786,95 | 1946,85 | 1843,93 | 1838,93 |
| Conf6 | 1786,95 | 1841,23 | 1800,83 | 1796,79 |
| Conf7 | 1786,95 | 1918,18 | 1825,81 | 1818,54 |
| Conf8 | 1786,95 | 1856,72 | 1791,92 | 1786,95 |
| Conf9 | 1786,95 | 1869,31 | 1823,74 | 1822,49 |
| Conf10 | 1786,95 | 1803,52 | 1788,71 | 1786,95 |
| Conf11 | 1786,95 | 1893,58 | 1839,17 | 1836,39 |
| Conf12 | 1786,95 | 1854,76 | 1800,33 | 1792,51 |
| Conf13 | 1786,95 | 1918,18 | 1841,06 | 1838,93 |
| Conf14 | 1786,95 | 1823,50 | 1794,86 | 1786,95 |
| Conf15 | 1786,95 | 2001,92 | 1825,96 | 1818,54 |
| Conf16 | 1786,95 | 1813,66 | 1788,72 | 1786,95 |
| Conf17 | 1796,79 | 1911,89 | 1848,85 | 1842,25 |
| Conf18 | 1786,95 | 1859,70 | 1802,48 | 1803,52 |
| Conf19 | 1786,95 | 1936,86 | 1840,23 | 1837,69 |
| Conf20 | 1786,95 | 1842,75 | 1791,70 | 1786,95 |
| Conf21 | 1786,95 | 1937,93 | 1840,06 | 1842,02 |
| Conf22 | 1786,95 | 1803,52 | 1788,16 | 1786,95 |
| Conf23 | 1786,95 | 2026,90 | 1856,12 | 1848,19 |
| Conf24 | 1786,95 | 1841,23 | 1804,50 | 1803,52 |
| Conf25 | 1786,95 | 1934,56 | 1841,83 | 1841,54 |
| Conf26 | 1786,95 | 1842,69 | 1790,72 | 1786,95 |
| Conf27 | 1786,95 | 1916,37 | 1836,03 | 1839,85 |
| Conf28 | 1786,95 | 1803,52 | 1788,39 | 1786,95 |
| Conf29 | 1803,52 | 2125,47 | 1871,63 | 1841,23 |
| Conf30 | 1786,95 | 1854,76 | 1803,80 | 1796,79 |
| Conf31 | 1786,95 | 1941,75 | 1850,00 | 1849,73 |
| Conf32 | 1786,95 | 1818,54 | 1790,03 | 1786,95 |
| Conf33 | 1786,95 | 1888,77 | 1830,51 | 1832,32 |
| Conf34 | 1786,95 | **1786,95** | **1786,95** | 1786,95 |
| Conf35 | 1786,95 | 2130,90 | 1858,36 | 1846,39 |
| Conf36 | 1786,95 | 1849,51 | 1797,84 | 1786,95 |

Tabla 3.7: Resultados de la 6ª instancia

| 50x4x20-1 | | | **Mejor encontrado: 5424.27** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | 5458,22 | 5795,16 | 5547,19 | 5535,80 |
| Conf2 | 5438,58 | 5502,07 | 5466,37 | 5462,89 |
| Conf3 | 5444,79 | 5767,97 | 5546,93 | 5517,69 |
| Conf4 | 5424,87 | 5538,39 | 5463,40 | 5460,45 |
| Conf5 | 5459,26 | 5786,61 | 5612,04 | 5613,49 |
| Conf6 | 5445,33 | 5564,18 | 5492,57 | 5489,66 |
| Conf7 | 5473,45 | 5838,91 | 5607,99 | 5598,75 |
| Conf8 | 5447,18 | 5526,49 | 5470,10 | 5465,71 |
| Conf9 | 5444,92 | 5710,72 | 5548,62 | 5528,09 |
| Conf10 | 5444,81 | 5524,64 | 5471,17 | 5468,00 |
| Conf11 | 5469,06 | 5862,75 | 5607,54 | 5603,91 |
| Conf12 | 5438,91 | 5604,30 | 5500,05 | 5493,74 |
| Conf13 | 5462,00 | 5800,99 | 5561,42 | 5557,47 |
| Conf14 | 5426,55 | 5499,94 | 5466,83 | 5464,07 |
| Conf15 | 5445,25 | 5737,44 | 5561,91 | 5549,76 |
| Conf16 | 5438,37 | 5617,82 | 5492,28 | 5484,40 |
| Conf17 | 5479,99 | 5893,23 | 5584,56 | 5554,93 |
| Conf18 | 5434,33 | 5639,88 | 5478,98 | 5472,24 |
| Conf19 | 5464,71 | 5888,02 | 5591,66 | 5558,40 |
| Conf20 | **5424,57** | 5547,75 | 5472,39 | 5470,26 |
| Conf21 | 5464,87 | 5808,17 | 5561,27 | 5526,12 |
| Conf22 | 5433,85 | 5516,20 | 5465,66 | 5462,90 |
| Conf23 | 5444,92 | 5807,03 | 5598,63 | 5590,86 |
| Conf24 | 5449,89 | 5592,69 | 5505,88 | 5496,26 |
| Conf25 | 5464,55 | 5784,44 | 5561,91 | 5549,70 |
| Conf26 | 5424,57 | **5496,61** | **5457,27** | **5454,10** |
| Conf27 | 5442,57 | 5721,20 | 5535,18 | 5516,86 |
| Conf28 | 5439,46 | 5552,42 | 5479,43 | 5476,28 |
| Conf29 | 5465,75 | 5995,06 | 5592,99 | 5573,88 |
| Conf30 | 5441,11 | 5628,93 | 5495,02 | 5494,87 |
| Conf31 | 5446,32 | 5763,42 | 5565,65 | 5546,15 |
| Conf32 | 5429,29 | 5502,28 | 5460,64 | 5456,31 |
| Conf33 | 5444,33 | 5726,04 | 5543,81 | 5526,98 |
| Conf34 | 5443,92 | 5580,11 | 5495,73 | 5488,18 |
| Conf35 | 5459,96 | 5831,91 | 5591,54 | 5582,19 |
| Conf36 | 5433,85 | 5574,14 | 5482,19 | 5477,46 |

Tabla 3.8: Resultados de la 7ª instancia

| 50x4x20-2 | | | **Mejor encontrado: 3737.38** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | 3750,31 | 3925,52 | 3820,97 | 3813,73 |
| Conf2 | 3749,98 | 3829,91 | **3780,38** | 3779,63 |
| Conf3 | **3741,02** | 3958,81 | 3818,10 | 3810,90 |
| Conf4 | 3746,58 | 3850,71 | 3786,66 | 3782,27 |
| Conf5 | 3755,97 | 3999,65 | 3835,32 | 3828,76 |
| Conf6 | 3750,98 | 3885,97 | 3803,16 | 3794,47 |
| Conf7 | 3761,83 | 3974,74 | 3826,22 | 3812,66 |
| Conf8 | 3757,78 | 3849,14 | 3796,13 | 3797,62 |
| Conf9 | 3767,23 | 3914,46 | 3820,62 | 3813,58 |
| Conf10 | 3748,58 | **3827,26** | 3781,20 | 3774,84 |
| Conf11 | 3768,23 | 4052,14 | 3844,92 | 3820,24 |
| Conf12 | 3765,17 | 3895,00 | 3811,72 | 3805,63 |
| Conf13 | 3749,98 | 3944,47 | 3820,42 | 3812,14 |
| Conf14 | 3748,50 | 3854,89 | 3790,06 | 3791,19 |
| Conf15 | 3750,31 | 3895,88 | 3810,97 | 3803,04 |
| Conf16 | 3742,51 | 3860,74 | 3784,32 | 3778,73 |
| Conf17 | 3758,26 | 4090,87 | 3829,42 | 3809,37 |
| Conf18 | 3742,51 | 3895,97 | 3804,88 | 3794,57 |
| Conf19 | 3754,63 | 4004,14 | 3865,54 | 3851,51 |
| Conf20 | 3749,98 | 3833,64 | 3789,43 | 3783,63 |
| Conf21 | 3749,98 | 4093,44 | 3834,92 | 3814,10 |
| Conf22 | 3742,51 | 3845,50 | 3783,34 | 3781,05 |
| Conf23 | 3786,22 | 4068,59 | 3880,83 | 3872,86 |
| Conf24 | 3748,50 | 3902,21 | 3808,45 | 3804,44 |
| Conf25 | 3761,82 | 4200,31 | 3872,90 | 3847,59 |
| Conf26 | 3759,74 | 3856,17 | 3797,00 | 3787,60 |
| Conf27 | 3757,88 | 4106,48 | 3851,66 | 3844,14 |
| Conf28 | 3752,75 | 3853,61 | 3780,83 | **3770,79** |
| Conf29 | 3755,97 | 4096,51 | 3869,62 | 3843,72 |
| Conf30 | 3749,98 | 3848,07 | 3803,31 | 3805,28 |
| Conf31 | 3760,74 | 3987,44 | 3848,50 | 3844,80 |
| Conf32 | 3752,28 | 3841,74 | 3791,46 | 3784,18 |
| Conf33 | 3767,23 | 4006,88 | 3840,51 | 3825,74 |
| Conf34 | 3742,51 | 3845,79 | 3792,29 | 3789,68 |
| Conf35 | 3768,15 | 4095,51 | 3859,06 | 3830,13 |
| Conf36 | 3763,58 | 3886,82 | 3807,10 | 3800,55 |

Tabla 3.9: Resultados de la 8ª instancia

| 50x4x20-3 | | | **Mejor encontrado: 3802.88** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | 3829,92 | 4162,17 | 3931,83 | 3921,51 |
| Conf2 | **3802,88** | 3946,47 | 3847,28 | 3841,35 |
| Conf3 | 3803,00 | 4075,30 | 3878,74 | 3869,42 |
| Conf4 | 3802,88 | 3921,61 | 3843,71 | 3837,29 |
| Conf5 | 3803,00 | 4054,63 | 3921,01 | 3925,50 |
| Conf6 | 3802,88 | 3985,24 | 3864,54 | 3856,27 |
| Conf7 | 3822,56 | 4046,53 | 3899,91 | 3895,03 |
| Conf8 | 3802,88 | 3901,22 | 3840,11 | 3835,34 |
| Conf9 | 3810,01 | 4101,92 | 3900,09 | 3879,38 |
| Conf10 | 3805,28 | 3948,15 | 3848,77 | 3838,86 |
| Conf11 | 3803,11 | 4097,62 | 3903,26 | 3900,07 |
| Conf12 | 3802,88 | 4040,30 | 3876,07 | 3851,10 |
| Conf13 | 3803,00 | 4328,05 | 3924,49 | 3901,53 |
| Conf14 | 3802,88 | 3909,61 | 3836,54 | 3832,92 |
| Conf15 | 3803,00 | 3997,08 | 3882,22 | 3878,69 |
| Conf16 | 3802,88 | 3952,92 | 3835,31 | 3828,41 |
| Conf17 | 3830,07 | 4240,63 | 3946,17 | 3943,54 |
| Conf18 | 3803,00 | 3984,62 | 3865,40 | 3852,47 |
| Conf19 | 3814,17 | 4369,81 | 3935,41 | 3903,45 |
| Conf20 | 3803,00 | 3948,24 | 3837,99 | 3831,14 |
| Conf21 | 3805,28 | 4086,30 | 3909,45 | 3890,79 |
| Conf22 | 3803,00 | 3911,36 | **3834,71** | **3826,44** |
| Conf23 | 3828,00 | 4115,46 | 3951,73 | 3930,46 |
| Conf24 | 3812,56 | 3950,01 | 3865,72 | 3850,30 |
| Conf25 | 3802,88 | 4133,73 | 3914,06 | 3886,60 |
| Conf26 | 3803,00 | **3900,41** | 3840,54 | 3839,58 |
| Conf27 | 3806,23 | 4038,00 | 3897,25 | 3876,75 |
| Conf28 | 3802,88 | 3994,68 | 3838,39 | 3827,85 |
| Conf29 | 3802,88 | 4168,21 | 3952,32 | 3940,13 |
| Conf30 | 3805,28 | 3989,95 | 3868,06 | 3851,10 |
| Conf31 | 3806,23 | 4164,12 | 3912,58 | 3904,09 |
| Conf32 | 3803,00 | 3914,45 | 3842,19 | 3834,75 |
| Conf33 | 3815,51 | 4034,18 | 3890,58 | 3886,04 |
| Conf34 | 3815,40 | 3923,49 | 3851,38 | 3840,51 |
| Conf35 | 3828,58 | 4177,20 | 3959,53 | 3948,08 |
| Conf36 | 3803,00 | 3936,47 | 3862,65 | 3860,32 |

Tabla 3.10: Resultados de la 9ª instancia

| 50x6x20-1 | | | **Mejor encontrado: 2868.39** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | 2972,04 | 3183,97 | 3071,34 | 3065,93 |
| Conf2 | 2980,78 | 3129,32 | 3049,55 | 3045,10 |
| Conf3 | 2977,76 | 3124,60 | 3058,20 | 3061,50 |
| Conf4 | 2985,49 | 3126,69 | 3046,36 | 3041,18 |
| Conf5 | 3017,74 | 3243,61 | 3089,36 | 3071,67 |
| Conf6 | 3002,75 | 3206,96 | 3087,95 | 3088,78 |
| Conf7 | 3001,83 | 3150,62 | 3053,48 | 3046,39 |
| Conf8 | 2980,17 | 3195,94 | 3047,32 | 3041,20 |
| Conf9 | 2995,39 | 3142,26 | 3059,39 | 3049,14 |
| Conf10 | 2985,49 | 3170,18 | 3040,73 | 3031,63 |
| Conf11 | 3015,39 | 3273,12 | 3102,77 | 3092,14 |
| Conf12 | 2988,36 | 3259,79 | 3093,44 | 3089,73 |
| Conf13 | 3010,79 | 3199,32 | 3087,56 | 3081,61 |
| Conf14 | 2987,69 | **3107,23** | 3037,24 | 3028,33 |
| Conf15 | 3003,80 | 3234,24 | 3065,74 | 3057,77 |
| Conf16 | 2983,43 | 3164,67 | 3051,46 | 3040,89 |
| Conf17 | 2989,70 | 3277,24 | 3083,68 | 3062,38 |
| Conf18 | 2975,56 | 3170,70 | 3061,96 | 3061,44 |
| Conf19 | 3002,75 | 3332,71 | 3128,24 | 3125,98 |
| Conf20 | 2995,42 | 3176,70 | 3050,67 | 3041,97 |
| Conf21 | 3047,48 | 3483,59 | 3152,65 | 3136,64 |
| Conf22 | 2983,29 | 3108,50 | 3031,81 | **3024,76** |
| Conf23 | 3000,88 | 3865,70 | 3255,75 | 3209,72 |
| Conf24 | 3013,29 | 3203,39 | 3072,07 | 3059,54 |
| Conf25 | 3016,16 | 3521,33 | 3174,61 | 3152,59 |
| Conf26 | 2981,97 | 3113,28 | 3037,89 | 3036,93 |
| Conf27 | 3005,54 | 3336,65 | 3131,52 | 3135,53 |
| Conf28 | 2983,29 | 3168,90 | 3037,52 | 3033,26 |
| Conf29 | 3012,34 | 3635,23 | 3258,41 | 3246,03 |
| Conf30 | 2988,36 | 3202,16 | 3069,92 | 3070,49 |
| Conf31 | 3032,92 | 3314,80 | 3154,64 | 3138,78 |
| Conf32 | 2979,77 | 3144,68 | 3043,91 | 3036,88 |
| Conf33 | 2987,69 | 3366,49 | 3120,20 | 3097,94 |
| Conf34 | **2969,84** | 3124,26 | **3029,95** | 3028,93 |
| Conf35 | 3008,53 | 3508,85 | 3198,40 | 3188,25 |
| Conf36 | 2972,04 | 3233,24 | 3066,24 | 3064,46 |

Tabla 3.11: Resultados de la 10ª instancia

| 50x6x20-2 | | | **Mejor encontrado: 2983.66** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | 3114,84 | 3293,84 | 3203,51 | 3193,09 |
| Conf2 | 3122,52 | 3239,92 | 3167,37 | 3166,40 |
| Conf3 | 3122,56 | 3336,39 | 3189,56 | 3179,50 |
| Conf4 | 3115,93 | **3190,10** | **3141,68** | **3136,95** |
| Conf5 | 3117,79 | 3483,63 | 3240,14 | 3219,35 |
| Conf6 | 3107,76 | 3308,37 | 3181,56 | 3171,24 |
| Conf7 | 3103,22 | 3405,70 | 3210,61 | 3203,13 |
| Conf8 | 3116,11 | 3231,31 | 3159,50 | 3160,70 |
| Conf9 | 3116,46 | 3369,31 | 3196,56 | 3181,07 |
| Conf10 | 3103,22 | 3209,93 | 3147,47 | 3139,91 |
| Conf11 | 3104,30 | 3376,35 | 3220,78 | 3213,29 |
| Conf12 | 3132,67 | 3308,81 | 3176,01 | 3160,42 |
| Conf13 | 3127,70 | 3335,98 | 3189,22 | 3178,54 |
| Conf14 | 3119,18 | 3233,57 | 3152,81 | 3151,14 |
| Conf15 | 3106,00 | 3321,22 | 3195,20 | 3179,56 |
| Conf16 | 3102,81 | 3216,19 | 3151,70 | 3150,17 |
| Conf17 | 3109,98 | 3457,82 | 3221,10 | 3198,11 |
| Conf18 | 3125,65 | 3241,88 | 3167,72 | 3163,29 |
| Conf19 | 3123,92 | 3504,28 | 3258,20 | 3252,91 |
| Conf20 | 3113,24 | 3244,19 | 3165,50 | 3165,82 |
| Conf21 | 3144,67 | 3471,28 | 3268,77 | 3255,98 |
| Conf22 | 3113,84 | 3191,44 | 3150,39 | 3153,58 |
| Conf23 | 3133,29 | 3527,50 | 3296,00 | 3296,66 |
| Conf24 | 3126,59 | 3374,72 | 3191,41 | 3171,86 |
| Conf25 | 3118,11 | 3392,35 | 3254,72 | 3249,09 |
| Conf26 | **3100,70** | 3270,60 | 3158,40 | 3157,00 |
| Conf27 | 3107,01 | 3362,01 | 3229,98 | 3222,35 |
| Conf28 | 3118,73 | 3249,32 | 3159,25 | 3152,58 |
| Conf29 | 3144,56 | 3854,07 | 3327,17 | 3289,17 |
| Conf30 | 3129,65 | 3263,98 | 3175,11 | 3167,09 |
| Conf31 | 3122,22 | 3545,04 | 3268,53 | 3260,41 |
| Conf32 | 3111,13 | 3240,84 | 3162,16 | 3158,56 |
| Conf33 | 3122,59 | 3404,22 | 3236,58 | 3223,65 |
| Conf34 | 3106,61 | 3217,89 | 3155,76 | 3150,53 |
| Conf35 | 3148,56 | 3798,77 | 3305,64 | 3296,30 |
| Conf36 | 3124,68 | 3331,43 | 3187,24 | 3167,79 |

Tabla 3.12: Resultados de la 11ª instancia

| 50x6x20-3 | | | **Mejor encontrado: 3090.38** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | 3201,46 | 3595,58 | 3330,45 | 3308,07 |
| Conf2 | 3186,59 | 3300,43 | 3242,04 | 3240,39 |
| Conf3 | 3205,16 | 3511,75 | 3295,17 | 3275,26 |
| Conf4 | 3189,97 | 3298,95 | 3232,03 | 3224,56 |
| Conf5 | 3195,02 | 3593,13 | 3343,97 | 3312,86 |
| Conf6 | 3207,90 | 3383,00 | 3275,52 | 3260,08 |
| Conf7 | 3200,11 | 3817,76 | 3337,34 | 3307,06 |
| Conf8 | 3183,29 | 3304,85 | 3233,57 | 3229,00 |
| Conf9 | 3201,33 | 3498,62 | 3297,82 | 3280,76 |
| Conf10 | 3190,19 | 3298,11 | 3233,25 | **3221,28** |
| Conf11 | 3206,56 | 3465,62 | 3334,50 | 3327,71 |
| Conf12 | 3190,14 | 3292,48 | 3238,78 | 3237,49 |
| Conf13 | 3203,16 | 3519,76 | 3316,00 | 3301,07 |
| Conf14 | **3175,05** | 3323,11 | 3244,20 | 3240,40 |
| Conf15 | 3202,19 | 3486,98 | 3291,04 | 3285,13 |
| Conf16 | 3186,59 | **3289,82** | **3226,78** | 3223,80 |
| Conf17 | 3206,23 | 3785,80 | 3357,18 | 3326,66 |
| Conf18 | 3191,97 | 3358,08 | 3273,36 | 3283,55 |
| Conf19 | 3203,51 | 3545,60 | 3325,50 | 3303,85 |
| Conf20 | 3175,05 | 3329,56 | 3237,24 | 3224,33 |
| Conf21 | 3208,00 | 3454,81 | 3299,67 | 3296,08 |
| Conf22 | 3186,59 | 3314,21 | 3240,84 | 3234,34 |
| Conf23 | 3244,13 | 3791,95 | 3376,35 | 3367,04 |
| Conf24 | 3180,43 | 3499,50 | 3277,81 | 3289,38 |
| Conf25 | 3239,00 | 3700,48 | 3379,01 | 3348,85 |
| Conf26 | 3189,88 | 3312,93 | 3242,54 | 3232,64 |
| Conf27 | 3212,49 | 3575,44 | 3349,92 | 3324,56 |
| Conf28 | 3187,00 | 3339,73 | 3238,49 | 3231,53 |
| Conf29 | 3232,62 | 3792,48 | 3398,85 | 3376,87 |
| Conf30 | 3184,00 | 3397,48 | 3263,58 | 3254,72 |
| Conf31 | 3203,87 | 3922,16 | 3343,04 | 3321,96 |
| Conf32 | 3183,71 | 3329,30 | 3235,97 | 3234,06 |
| Conf33 | 3201,39 | 3509,19 | 3308,88 | 3309,13 |
| Conf34 | 3191,92 | 3339,89 | 3244,36 | 3237,38 |
| Conf35 | 3203,87 | 3623,27 | 3392,16 | 3371,99 |
| Conf36 | 3198,52 | 3505,59 | 3284,14 | 3263,95 |

Tabla 3.13: Resultados de la 12ª instancia

| 100x4x25-1 | | | **Mejor encontrado: 8293.42** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | 8378,99 | 8991,09 | 8558,37 | 8527,29 |
| Conf2 | 8539,43 | 8884,76 | 8638,59 | 8628,08 |
| Conf3 | 8388,27 | 8986,03 | 8555,78 | 8527,63 |
| Conf4 | 8828,49 | 9786,70 | 9263,33 | 9254,44 |
| Conf5 | 8391,02 | 8954,07 | 8642,86 | 8616,56 |
| Conf6 | 8373,23 | 8650,12 | 8473,16 | **8456,14** |
| Conf7 | **8306,96** | 8931,70 | 8524,99 | 8482,28 |
| Conf8 | 8501,63 | 9001,49 | 8706,78 | 8695,87 |
| Conf9 | 8362,30 | 8808,31 | 8521,57 | 8510,35 |
| Conf10 | 9035,63 | 9962,36 | 9450,98 | 9391,68 |
| Conf11 | 8335,68 | 8804,58 | 8548,66 | 8519,70 |
| Conf12 | 8385,09 | 8713,43 | 8524,29 | 8508,76 |
| Conf13 | 8386,78 | 8769,93 | 8524,15 | 8474,22 |
| Conf14 | 8538,76 | 9100,68 | 8801,60 | 8774,79 |
| Conf15 | 8357,30 | 8982,57 | 8560,31 | 8534,63 |
| Conf16 | 9017,24 | 10486,56 | 9773,89 | 9766,00 |
| Conf17 | 8373,45 | 8904,47 | 8564,16 | 8499,16 |
| Conf18 | 8336,59 | 8658,42 | 8480,88 | 8473,90 |
| Conf19 | 8386,13 | 8993,54 | 8558,02 | 8514,02 |
| Conf20 | 8431,47 | 9038,79 | 8712,52 | 8693,23 |
| Conf21 | 8383,55 | 8730,27 | 8512,43 | 8508,87 |
| Conf22 | 9002,52 | 9882,92 | 9360,69 | 9327,56 |
| Conf23 | 8388,55 | 8998,17 | 8597,34 | 8547,20 |
| Conf24 | 8379,91 | 8700,95 | 8501,66 | 8494,30 |
| Conf25 | 8384,50 | 8975,49 | 8577,66 | 8530,31 |
| Conf26 | 8547,06 | 9607,73 | 8817,39 | 8754,73 |
| Conf27 | 8370,92 | 8748,94 | 8538,53 | 8497,11 |
| Conf28 | 9112,00 | 10520,28 | 9553,99 | 9548,11 |
| Conf29 | 8374,20 | 9281,19 | 8598,82 | 8537,14 |
| Conf30 | 8354,52 | **8633,70** | **8471,44** | 8467,25 |
| Conf31 | 8342,92 | 9407,92 | 8600,77 | 8548,81 |
| Conf32 | 8686,24 | 9581,45 | 8965,61 | 8907,23 |
| Conf33 | 8385,23 | 8751,45 | 8541,43 | 8516,91 |
| Conf34 | 9505,06 | 10749,99 | 9972,77 | 9900,69 |
| Conf35 | 8375,29 | 9336,67 | 8595,38 | 8550,97 |
| Conf36 | 8335,87 | 8817,00 | 8478,52 | 8463,41 |

Tabla 3.14: Resultados de la 13ª instancia

| 100x4x25-2 | | | **Mejor encontrado: 7273.03** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | 7449,53 | 8192,61 | 7667,15 | 7637,53 |
| Conf2 | 7424,93 | 8139,18 | 7647,19 | 7594,23 |
| Conf3 | **7305,25** | 7942,58 | 7614,44 | 7596,23 |
| Conf4 | 7706,07 | 8770,26 | 8191,91 | 8191,10 |
| Conf5 | 7382,02 | 8151,69 | 7666,47 | 7619,11 |
| Conf6 | 7307,01 | 7928,71 | 7571,34 | 7571,86 |
| Conf7 | 7357,57 | **7848,92** | 7628,87 | 7648,90 |
| Conf8 | 7465,19 | 8103,00 | 7760,37 | 7770,64 |
| Conf9 | 7342,85 | 7851,30 | **7563,01** | 7533,51 |
| Conf10 | 8130,86 | 9121,69 | 8527,64 | 8494,22 |
| Conf11 | 7385,25 | 8013,78 | 7639,94 | 7616,38 |
| Conf12 | 7356,10 | 7862,28 | 7568,07 | 7557,23 |
| Conf13 | 7320,96 | 7917,80 | 7592,71 | 7608,25 |
| Conf14 | 7568,08 | 8355,36 | 7938,16 | 7886,47 |
| Conf15 | 7333,80 | 7896,99 | 7569,50 | 7550,58 |
| Conf16 | 8180,18 | 9225,19 | 8676,88 | 8660,42 |
| Conf17 | 7352,77 | 7979,88 | 7640,07 | 7643,35 |
| Conf18 | 7324,10 | 7939,07 | 7578,35 | 7526,75 |
| Conf19 | 7357,96 | 8135,75 | 7617,70 | 7585,18 |
| Conf20 | 7423,16 | 8055,02 | 7733,24 | 7699,78 |
| Conf21 | 7342,65 | 8183,10 | 7636,31 | 7626,59 |
| Conf22 | 7915,06 | 9020,30 | 8416,24 | 8386,44 |
| Conf23 | 7343,14 | 8952,38 | 7708,08 | 7660,61 |
| Conf24 | 7357,71 | 8341,19 | 7580,55 | **7526,24** |
| Conf25 | 7322,89 | 8035,77 | 7677,63 | 7673,02 |
| Conf26 | 7467,53 | 8489,49 | 7810,52 | 7771,44 |
| Conf27 | 7362,89 | 8272,30 | 7624,20 | 7607,71 |
| Conf28 | 8191,19 | 9689,19 | 8740,21 | 8669,56 |
| Conf29 | 7374,94 | 8515,76 | 7776,40 | 7750,70 |
| Conf30 | 7410,39 | 7959,34 | 7593,41 | 7542,22 |
| Conf31 | 7434,67 | 8486,17 | 7706,19 | 7693,41 |
| Conf32 | 7551,92 | 8430,62 | 7903,87 | 7878,08 |
| Conf33 | 7386,87 | 8019,88 | 7627,78 | 7612,31 |
| Conf34 | 8186,97 | 9821,93 | 9044,43 | 9019,12 |
| Conf35 | 7312,43 | 7971,07 | 7589,58 | 7603,73 |
| Conf36 | 7365,27 | 8002,70 | 7589,99 | 7539,80 |

Tabla 3.15: Resultados de la 14ª instancia

| 100x4x25-3 | | | **Mejor encontrado: 8626.13** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | 8628,12 | 9538,73 | 8928,35 | 8869,33 |
| Conf2 | 8782,75 | 9391,78 | 8982,93 | 8962,87 |
| Conf3 | 8712,14 | 9461,47 | 8912,11 | 8845,93 |
| Conf4 | 9125,15 | 9953,62 | 9508,52 | 9454,60 |
| Conf5 | 8678,58 | 9420,44 | 8849,50 | 8807,09 |
| Conf6 | 8660,87 | 8999,29 | 8802,86 | 8784,18 |
| Conf7 | **8626,13** | 9229,27 | 8854,08 | 8815,63 |
| Conf8 | 8741,59 | 9489,37 | 9089,60 | 9078,10 |
| Conf9 | 8693,87 | 9174,64 | 8818,25 | 8794,73 |
| Conf10 | 9343,56 | 10158,07 | 9729,61 | 9711,83 |
| Conf11 | 8718,47 | 9443,74 | 8916,52 | 8892,48 |
| Conf12 | 8665,38 | 9020,03 | **8751,20** | **8742,79** |
| Conf13 | 8663,84 | 9869,69 | 8902,10 | 8820,25 |
| Conf14 | 8945,37 | 9553,15 | 9213,20 | 9205,27 |
| Conf15 | 8700,36 | 9283,96 | 8838,93 | 8788,85 |
| Conf16 | 9677,88 | 10408,83 | 10044,83 | 10067,47 |
| Conf17 | 8722,52 | 9998,68 | 8898,03 | 8813,24 |
| Conf18 | 8689,31 | 9011,61 | 8776,30 | 8766,38 |
| Conf19 | 8695,07 | 9095,66 | 8835,97 | 8827,80 |
| Conf20 | 8824,93 | 9268,23 | 8995,89 | 8964,92 |
| Conf21 | 8667,11 | 9018,28 | 8817,99 | 8829,48 |
| Conf22 | 9005,05 | 10102,93 | 9549,28 | 9575,82 |
| Conf23 | 8663,66 | 9309,30 | 8846,37 | 8804,68 |
| Conf24 | 8659,70 | 8997,28 | 8786,57 | 8770,28 |
| Conf25 | 8680,42 | 9249,53 | 8889,59 | 8871,90 |
| Conf26 | 8868,93 | 9668,77 | 9123,36 | 9110,16 |
| Conf27 | 8674,75 | 9338,36 | 8865,31 | 8825,38 |
| Conf28 | 9330,74 | 10343,81 | 9871,82 | 9880,36 |
| Conf29 | 8674,39 | 9367,15 | 8895,47 | 8825,69 |
| Conf30 | 8660,94 | 8971,09 | 8773,67 | 8777,30 |
| Conf31 | 8673,44 | 9160,72 | 8836,76 | 8806,40 |
| Conf32 | 8952,09 | 10166,93 | 9362,29 | 9330,36 |
| Conf33 | 8645,58 | 9871,20 | 8871,15 | 8834,26 |
| Conf34 | 9665,02 | 10573,40 | 10111,03 | 10105,71 |
| Conf35 | 8716,28 | 9161,96 | 8867,81 | 8845,59 |
| Conf36 | 8662,49 | **8945,26** | 8784,27 | 8782,04 |

Tabla 3.16: Resultados de la 15ª instancia

| 100x6x25-1 | | | **Mejor encontrado: 5306.50** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | 5440,86 | 5903,35 | 5634,74 | 5595,81 |
| Conf2 | 5360,68 | 6055,52 | 5610,39 | 5602,37 |
| Conf3 | **5292,05** | 6098,00 | 5577,42 | 5549,58 |
| Conf4 | 5670,60 | 6525,64 | 6043,26 | 6034,24 |
| Conf5 | 5433,80 | 6148,36 | 5659,11 | 5647,00 |
| Conf6 | 5388,72 | 5930,77 | 5626,01 | 5624,79 |
| Conf7 | 5453,16 | 6435,33 | 5677,45 | 5638,41 |
| Conf8 | 5411,31 | 5885,33 | 5622,38 | 5613,33 |
| Conf9 | 5334,78 | 5947,72 | **5565,71** | 5569,83 |
| Conf10 | 5701,14 | 6779,23 | 6093,24 | 6074,90 |
| Conf11 | 5482,52 | 6890,18 | 5758,99 | 5693,92 |
| Conf12 | 5381,19 | 6026,70 | 5599,82 | 5587,27 |
| Conf13 | 5386,33 | 6262,95 | 5616,20 | 5584,46 |
| Conf14 | 5453,37 | 5996,04 | 5621,40 | 5577,75 |
| Conf15 | 5422,26 | **5845,52** | 5598,69 | 5589,01 |
| Conf16 | 5814,43 | 7391,53 | 6279,05 | 6194,12 |
| Conf17 | 5325,26 | 6258,17 | 5694,56 | 5637,45 |
| Conf18 | 5395,90 | 6050,78 | 5598,09 | 5560,85 |
| Conf19 | 5388,52 | 6307,09 | 5724,42 | 5727,65 |
| Conf20 | 5437,49 | 6001,21 | 5728,36 | 5730,23 |
| Conf21 | 5506,61 | 6960,34 | 5770,55 | 5702,48 |
| Conf22 | 5911,59 | 7061,45 | 6420,45 | 6435,43 |
| Conf23 | 5426,74 | 6594,33 | 5842,64 | 5784,64 |
| Conf24 | 5473,70 | 6089,61 | 5720,63 | 5693,57 |
| Conf25 | 5364,37 | 6006,11 | 5663,67 | 5649,22 |
| Conf26 | 5488,67 | 6069,64 | 5765,31 | 5750,20 |
| Conf27 | 5398,48 | 6117,89 | 5695,14 | 5648,53 |
| Conf28 | 5985,91 | 7496,92 | 6659,85 | 6573,16 |
| Conf29 | 5346,66 | 6624,25 | 5844,87 | 5769,52 |
| Conf30 | 5368,72 | 5912,03 | 5601,74 | **5536,96** |
| Conf31 | 5428,14 | 6438,58 | 5815,35 | 5760,31 |
| Conf32 | 5559,32 | 6501,32 | 5900,71 | 5893,14 |
| Conf33 | 5460,66 | 6324,87 | 5745,09 | 5639,55 |
| Conf34 | 6264,61 | 7688,81 | 6954,86 | 6902,11 |
| Conf35 | 5388,58 | 6279,67 | 5732,96 | 5633,66 |
| Conf36 | 5297,04 | 6233,51 | 5649,00 | 5642,59 |

Tabla 3.17: Resultados de la 16ª instancia

| 100x6x25-2 | | | **Mejor encontrado: 6141.07** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | 6309,53 | 7531,66 | 6691,23 | 6651,09 |
| Conf2 | 6397,22 | 7166,62 | 6605,85 | 6559,77 |
| Conf3 | **6202,41** | 7191,88 | 6647,87 | 6531,64 |
| Conf4 | 6678,75 | 7339,87 | 6999,62 | 6997,91 |
| Conf5 | 6288,16 | 8057,47 | 6814,36 | 6749,97 |
| Conf6 | 6241,01 | 6984,02 | 6529,31 | 6495,73 |
| Conf7 | 6231,55 | 7239,90 | 6694,16 | 6641,92 |
| Conf8 | 6335,63 | 7379,70 | 6680,93 | 6676,47 |
| Conf9 | 6285,90 | 7114,66 | 6646,66 | 6585,61 |
| Conf10 | 6749,28 | 7961,22 | 7193,69 | 7187,46 |
| Conf11 | 6237,54 | 8108,23 | 6932,31 | 6911,34 |
| Conf12 | 6228,47 | 7011,46 | 6549,64 | 6496,34 |
| Conf13 | 6269,35 | 7857,28 | 6707,23 | 6652,55 |
| Conf14 | 6419,51 | 7003,75 | 6715,63 | 6711,98 |
| Conf15 | 6327,23 | 7214,36 | 6667,82 | 6643,93 |
| Conf16 | 7012,24 | 8138,65 | 7492,62 | 7385,63 |
| Conf17 | 6389,60 | 7778,40 | 6848,73 | 6762,52 |
| Conf18 | 6276,51 | 7008,08 | **6515,68** | **6489,11** |
| Conf19 | 6268,57 | 8753,75 | 6919,87 | 6889,77 |
| Conf20 | 6355,24 | **6887,41** | 6630,82 | 6633,41 |
| Conf21 | 6231,29 | 7516,33 | 6674,80 | 6598,65 |
| Conf22 | 6842,86 | 7795,41 | 7356,52 | 7362,89 |
| Conf23 | 6277,26 | 7447,59 | 6910,38 | 6906,84 |
| Conf24 | 6319,22 | 7095,58 | 6547,57 | 6514,16 |
| Conf25 | 6252,85 | 7750,15 | 6797,19 | 6724,14 |
| Conf26 | 6325,46 | 7322,52 | 6822,65 | 6831,38 |
| Conf27 | 6325,32 | 7247,46 | 6773,07 | 6788,93 |
| Conf28 | 7056,37 | 8349,62 | 7650,36 | 7589,79 |
| Conf29 | 6246,73 | 7618,99 | 6923,73 | 6910,60 |
| Conf30 | 6304,03 | 6993,98 | 6535,53 | 6508,66 |
| Conf31 | 6206,52 | 7482,81 | 6729,61 | 6649,27 |
| Conf32 | 6529,87 | 7273,28 | 6862,23 | 6835,12 |
| Conf33 | 6278,78 | 7903,78 | 6805,33 | 6705,84 |
| Conf34 | 7303,68 | 8551,87 | 7813,58 | 7742,06 |
| Conf35 | 6357,59 | 8510,77 | 7081,28 | 6966,44 |
| Conf36 | 6270,13 | 6916,01 | 6545,34 | 6522,29 |

Tabla 3.18: Resultados de la 17ª instancia

| 100x6x25-3 | | | **Mejor encontrado: 5804.19** | |
|---|---|---|---|---|
| Configuración | Mínimo | Máximo | Media | Mediana |
| Conf1 | 5853,06 | 6583,92 | 6119,93 | 6088,13 |
| Conf2 | 6033,77 | 6547,98 | 6242,00 | 6212,93 |
| Conf3 | 5881,67 | 6647,31 | 6129,24 | 6108,85 |
| Conf4 | 6333,56 | 7517,95 | 6696,68 | 6658,18 |
| Conf5 | 5959,12 | 6984,79 | 6226,49 | 6164,39 |
| Conf6 | 5965,40 | 6539,66 | 6220,09 | 6244,18 |
| Conf7 | 5906,91 | 6601,26 | 6180,91 | 6164,82 |
| Conf8 | 5986,28 | 6671,40 | 6276,48 | 6255,26 |
| Conf9 | 5858,86 | 6378,92 | 6084,22 | 6069,29 |
| Conf10 | 6367,39 | 7202,81 | 6777,29 | 6784,08 |
| Conf11 | 5885,72 | 6562,32 | 6174,29 | 6138,36 |
| Conf12 | 5924,26 | 6444,89 | 6160,11 | 6123,37 |
| Conf13 | 5917,42 | 6523,53 | 6147,31 | 6127,32 |
| Conf14 | 5922,60 | 6630,89 | 6279,16 | 6275,90 |
| Conf15 | 5922,36 | 6571,95 | 6147,10 | 6135,53 |
| Conf16 | 6595,44 | 7974,92 | 7145,16 | 7044,75 |
| Conf17 | 5910,53 | 6866,52 | 6272,97 | 6234,47 |
| Conf18 | 5898,58 | 6687,83 | 6164,22 | 6116,60 |
| Conf19 | 5879,84 | 7524,22 | 6313,36 | 6200,07 |
| Conf20 | 5978,65 | 6693,08 | 6227,10 | 6203,89 |
| Conf21 | 5886,72 | 6733,66 | 6233,42 | 6233,52 |
| Conf22 | 6512,76 | 7727,47 | 6937,16 | 6888,81 |
| Conf23 | 5878,06 | 6999,24 | 6354,15 | 6331,66 |
| Conf24 | 5908,43 | 6396,27 | 6074,55 | **6043,18** |
| Conf25 | 5964,89 | 6912,49 | 6268,84 | 6242,29 |
| Conf26 | 6020,52 | 6929,13 | 6305,90 | 6294,82 |
| Conf27 | 5953,41 | 6461,85 | 6211,96 | 6193,80 |
| Conf28 | 6336,49 | 7777,64 | 7075,00 | 7095,76 |
| Conf29 | 5912,14 | 7403,96 | 6375,09 | 6240,82 |
| Conf30 | **5815,92** | **6353,79** | **6064,69** | 6060,10 |
| Conf31 | 5862,82 | 6966,80 | 6225,49 | 6172,22 |
| Conf32 | 6088,79 | 6825,92 | 6357,54 | 6319,80 |
| Conf33 | 5964,82 | 6966,99 | 6205,99 | 6176,69 |
| Conf34 | 6470,69 | 8200,69 | 7373,07 | 7345,93 |
| Conf35 | 5929,36 | 6932,75 | 6322,90 | 6275,16 |
| Conf36 | 5920,59 | 6446,72 | 6100,10 | 6064,11 |

Tabla 3.19: Resultados de la 18ª instancia

# Capítulo 4

# Aplicación Web

Uno de los objetivos marcados para el desarrollo del proyecto, era el de desarrollar una aplicación web para mostrar la funcionalidad del algoritmo implementado, tal y como se comentaba en apartados anteriores. En este capítulo se presenta el desarrollo seguido para la implementación de la aplicación, así como muestras del funcionamiento de la misma. En el siguiente enlace (`https://mdccvrp-solver-web-app.vercel.app/`) se puede acceder a la aplicación en cuestión.

## 4.1. Desarrollo de la aplicación

La primera fase del desarrollo consistió en la definición de las funcionalidades que se deseaban para la aplicación, así como las tecnologías que se querían utilizar para su implementación. A continuación se enumeran los distintos objetivos marcados:

- Lograr que el algoritmo se ejecute de forma fluida
- Establecer los parámetros para la ejecución del algoritmo
- Visualizar el valor objetivo de la solución
- Visualizar de forma gráfica la solución obtenida
- Permitir que el usuario pueda subir sus propias instancias
- Visualizar la evolución de la mejor solución de cada generación
- Añadir controles a la visualización de soluciones

Posteriormente se diseñaron las pantallas de las que dispondría la aplicación. Estos diseños fueron implementados utilizando la plataforma *Figma* y su objetivo era el de presentar la interfaz básica que debería presentar como se muestra en las figuras 4.1 y 4.2.

Estos diseños sufrieron modificaciones mientras se desarrollaba el código para adaptarse a las necesidades que iban surgiendo, como fue el caso del tamaño de algunos elementos o los mensajes de los campos de texto, entre otros. Las interfaces finales de la pantalla principal y el modal para la subida de instancias se muestran en las figuras 4.3 y 4.4 respectivamente.

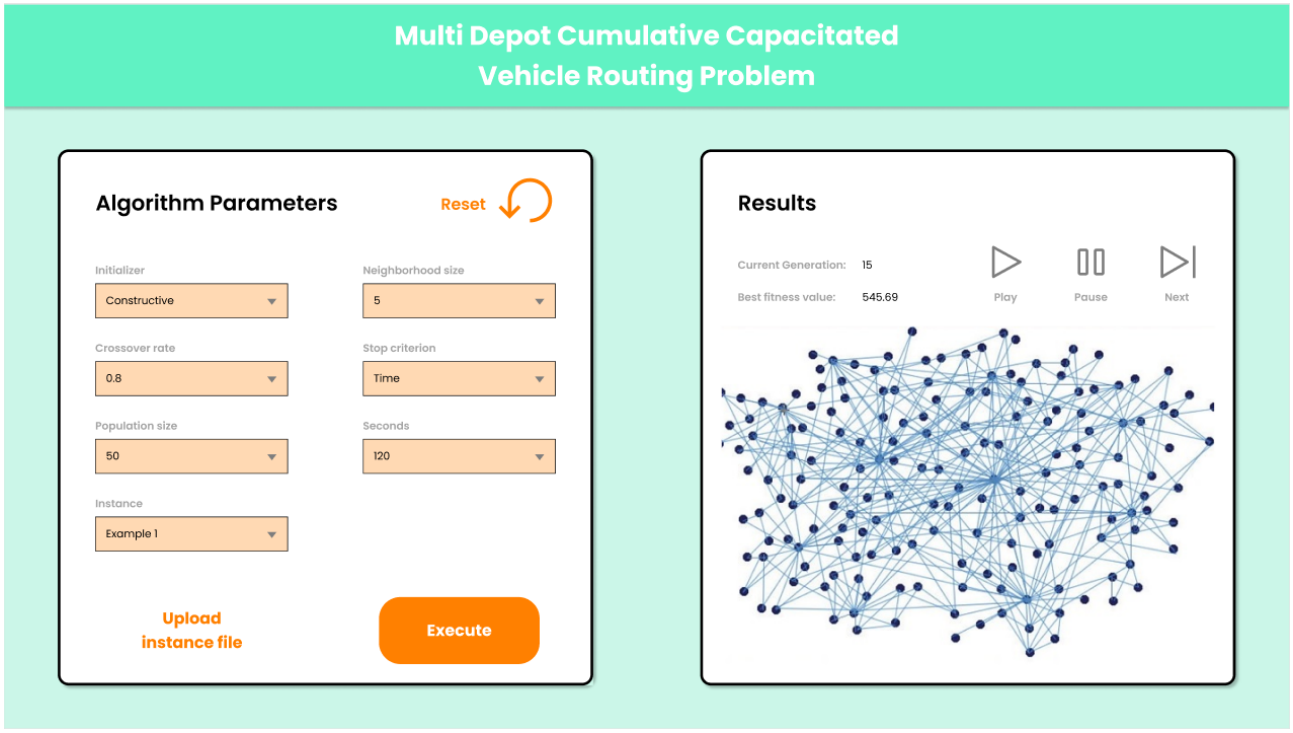


Figura 4.1: Diseño inicial de la pantalla principal

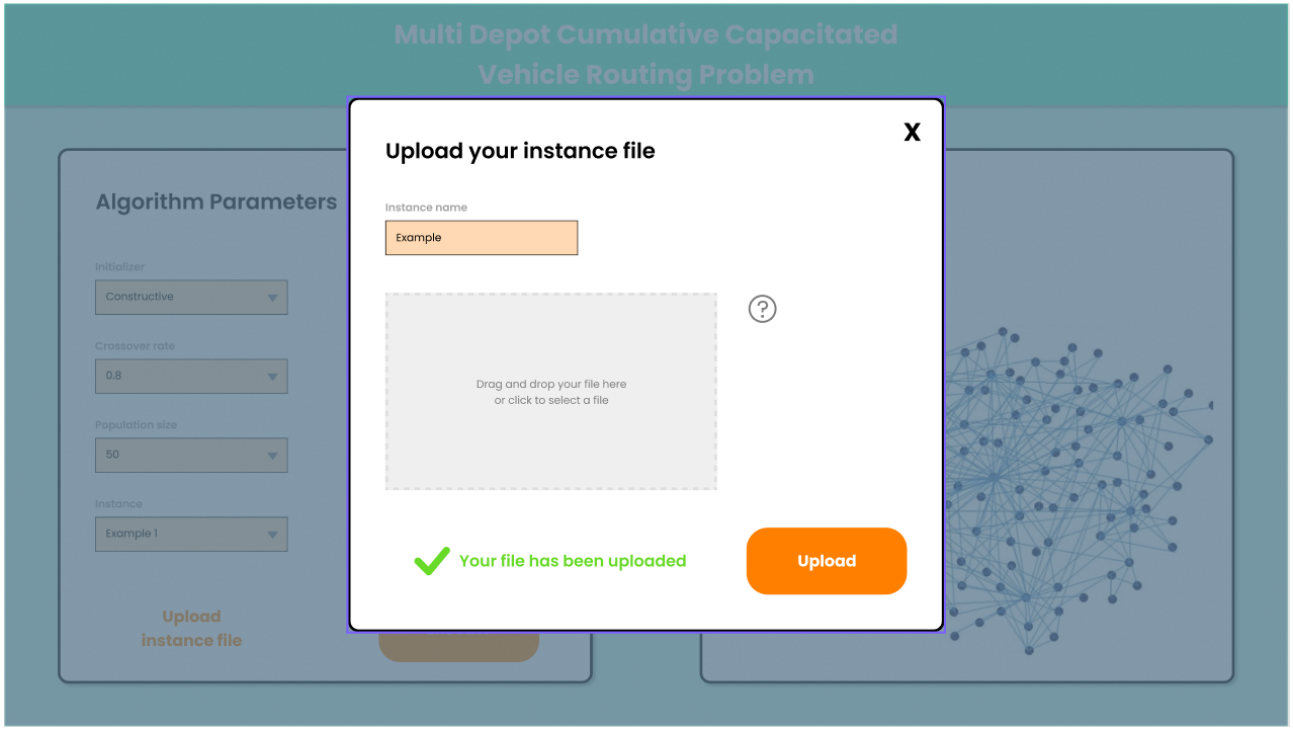


Figura 4.2: Diseño inicial del modal de subida de instancias

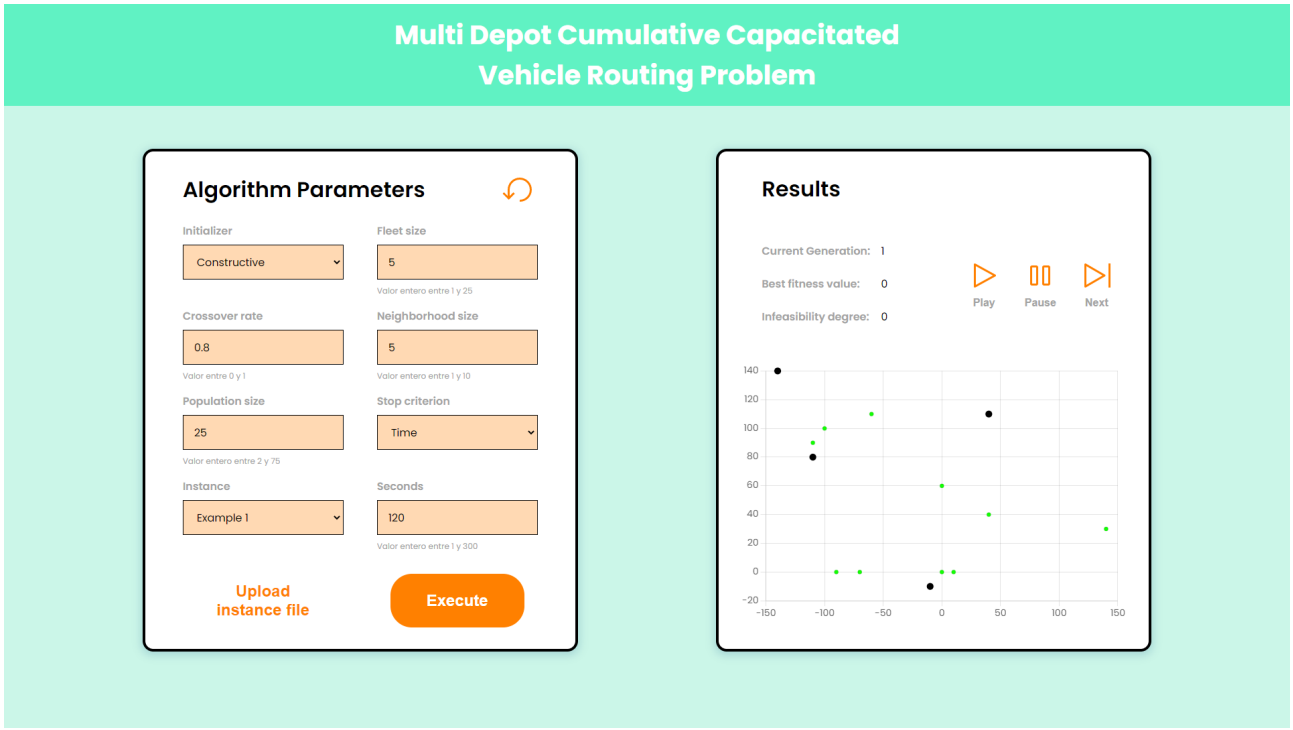


Figura 4.3: Pantalla principal

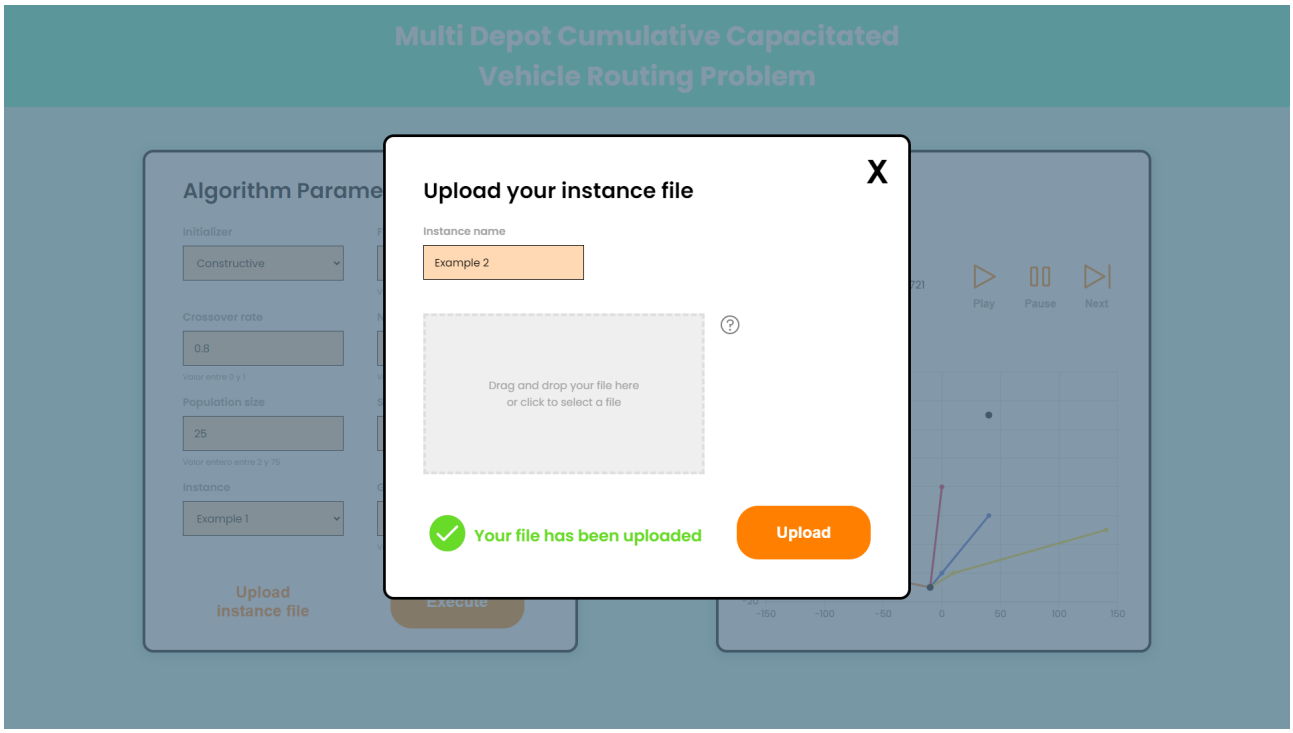


Figura 4.4: Modal de subida de instancias

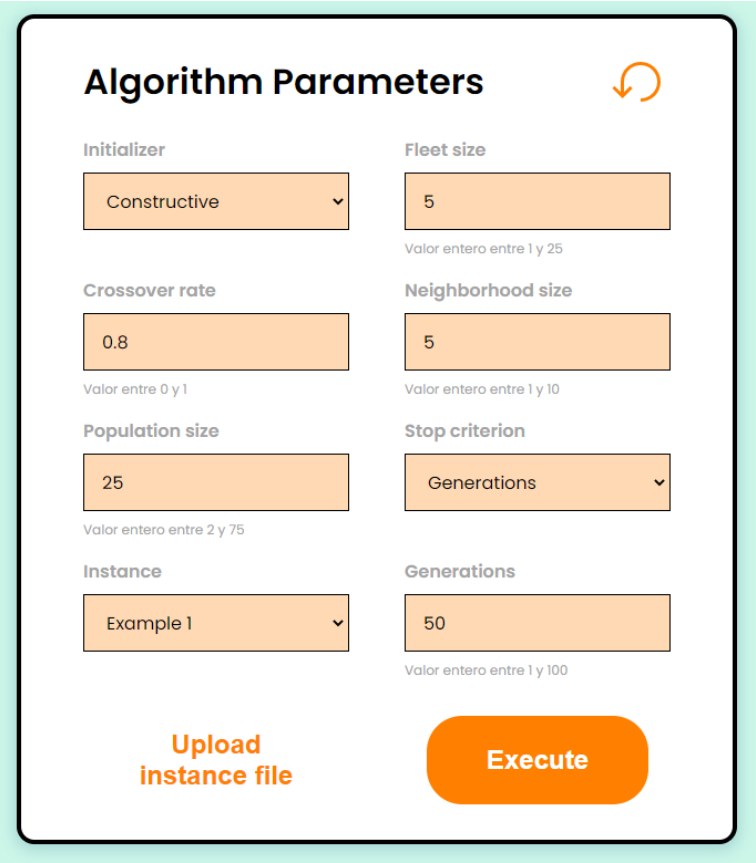


Figura 4.5: Formulario de los parámetros del algoritmo

Como se puede apreciar en la figura 4.5, se implementó un sencillo formulario para establecer los parámetros de ejecución del algoritmo. Para ello se utilizan tanto campos de texto, como desplegables para elegir una opción entre las ofertadas. Además, los campos están limitados debido a que se trata de una aplicación de ejemplo y las ejecuciones podrían tener una duración muy elevada si no se limitasen algunos parámetros como las condiciones de parada o el tamaño de la población. Además, se incluye una instancia de ejemplo para que si el usuario no dispone de alguna para cargar en la aplicación, pueda probarla.

Por otro lado, para implementar la subida de instancias se optó por un modal con una *dropzone* que permite al usuario arrastrar un fichero o elegirlo de su sistema de archivos. Además, comprueba el formato de la misma como se puede ver en la figura 4.6, impidiendo así que el usuario pueda cargar una instancia errónea y el algoritmo pueda fallar.

Por último, para la ejecución y visualización del algoritmo, se ha utilizado un *web worker* para evitar colapsar el hilo de ejecución de la pestaña, que devuelve la mejor solución de cada solución. Para la visualización de las soluciones, al tratarse de grafos, se podían representar de forma sencilla en una gráfica, y para ello se utilizó la librería *react-chartjs-2*, con la que se muestra cada una de las rutas de un color y los depósitos en negro. Además, se incorporaron botones para controlar la visualización, permitiendo pausarla o mostrar las soluciones de una en una.

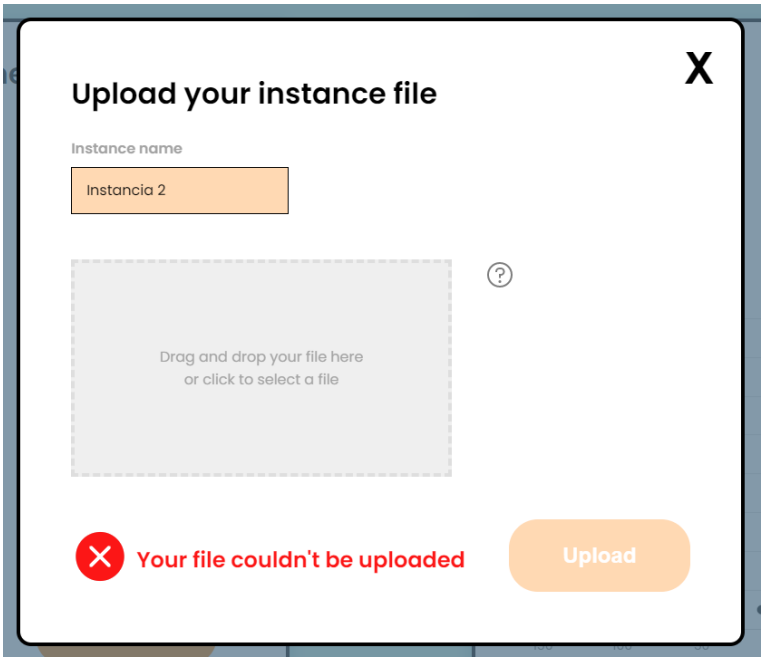


Figura 4.6: Instancia con formato erróneo


Figura 4.7: Visualización de soluciones

## 4.2. Método de uso

Como se comentaba en secciones anteriores, el objetivo de la aplicación web era el de servir de ejemplo de la ejecución del algoritmo, y, por lo tanto, en la pantalla principal se presentan tanto parámetros por defecto, como una instancia de ejemplo para que el usuario pueda ejecutar directamente el algoritmo.

El formulario con los parámetros permite personalizar diversos aspectos de la ejecución del algoritmo, tanto mediante campos de texto, como selectores de opciones. Estos están limitados y se realiza una comprobación del dato introducido cuando el usuario se focaliza en otro campo o intenta ejecutar el algoritmo, mostrando un error en caso de que estuviese fuera de rango. Entre los elementos que se pueden personalizar se encuentra la instancia a ejecutar, pudiéndose elegir mediante un selector y actualizando en el apartado de visualización [4.7] la vista previa de la instancia con sus depósitos y sus nodos clientes. Por último cabe destacar que se presenta un botón para reiniciar todos los parámetros a los establecidos por defecto, pero manteniendo las instancias que ha cargado el usuario en la aplicación.

En el modal para la subida de instancias [4.4], se permite tanto arrastrar un archivo, como seleccionarlo entre el sistema de archivos. Además, hay que indicar un nombre único para la instancia (si ya se había registrado una instancia con ese nombre, se muestra un error en el campo). Por otro lado, se dispone de un botón de ayuda que redirige al usuario a la siguiente página (`https://ull-esit-inf-tfg-2021.github.io/MDCCVRP-Instances/`), donde se presenta el formato de las instancias y permite la descarga de las utilizadas en los artículos que han trabajado sobre el *MDCCVRP* hasta la fecha. Debido a que no se consta de un *backend*, las instancias se mantendrán almacenadas mientras perdure la sesión del usuario en la aplicación.

Al pulsar el botón para ejecutar el algoritmo, se mostrará una animación de carga mientras se esté ejecutando en donde se mostraba la vista previa de la instancia. Una vez finalizada, se muestra la mejor solución de la primera generación tanto en el gráfico, como los datos de la generación actual, el valor objetivo y el valor de infactibilidad. Para iniciar la visualización del resto de soluciones, hay que presionar el botón de *play* y se empezarán a mostrar las mejores soluciones de cada generación sucesivamente hasta que se alcance la última o se presione el botón de pausa. Además, se dispone de un botón para mostrar únicamente la siguiente solución, que se podrá utilizar tanto cuando la visualización se está llevando a cabo, como cuando está pausada.

# Capítulo 5

# Conclusiones y líneas futuras

Una vez finalizado el proyecto, es necesario hacer un balance del mismo para analizar si se han alcanzado los objetivos marcados para el mismo, cómo se ha realizado el trabajo, la calidad del resultado y las posibles mejoras o adiciones que se pueden hacer de cara al futuro.

Siguiendo la enumeración de los objetivos realizada en la sección de objetivos, el primero de ellos consistía en analizar e identificar el problema. Esto implicó realizar una investigación tanto sobre el problema, como sobre la computación evolutiva, para decidir si se podría aportar valor a la investigación sobre el *MDCCVRP* con nuestro enfoque y cuál sería el mejor acercamiento. A continuación, se llevó a cabo el estudio del estado del arte, pero al tratarse de un problema de reciente formulación, no existían muchos artículos al respecto, por lo que se expandió la búsqueda a variantes cercanas, lo que permitió aprender técnicas que fueron útiles para el desarrollo de algunos de los componentes del algoritmo como es el operador de cruce o la búsqueda local.

El objetivo principal del proyecto era el de implementar un algoritmo memético capaz de resolver el *MDCCVRP* de forma eficiente y que se pudiesen obtener soluciones óptimas o cercanas a las mismas. Como se puede ver en el capítulo de los resultados obtenidos, este objetivo se pudo alcanzar de forma satisfactoria, llegando incluso a obtener una solución mejor que las que se habían encontrado en los artículos estudiados. Sin embargo, como se comentará más adelante, el código implementado puede ser optimizado para evitar cálculos repetitivos y mejorar su rendimiento, así como mejorar el algoritmo en sí mismo.

Por último, se pretendía implementar una aplicación web sencilla en la que los usuarios pudiesen resolver sus instancias del *MDCCVRP* y visualizar los resultados de gráfica. Esto también se logró cumplir, pero también se puede mejorar tanto a nivel de código, que debido al tiempo limitado, no se pudo refinar demasiado. Otro aspecto que se puede mejorar es a nivel de *User Experience* (*UX*), ya que no se disponían de los conocimientos ni del tiempo necesarios para hacer un estudio adecuado sobre este ámbito.

Por lo tanto, se puede decir que a nivel general se han alcanzado todos los objetivos marcados, pero se es consciente de las mejoras que se pueden aplicar al proyecto, como se listan a continuación:

- **Mejora de la documentación:** Debido a la escasez de tiempo, la documentación

realizada para los últimos elementos implementados es nula o escasa, y debiera tenerse la documentación completa de cara a que se facilite la utilización del paquete.

- **Optimizar el algoritmo:** Existen cálculos repetitivos en el código que se pueden llegar a optimizar, mejorando así el rendimiento del mismo.
- **Mejora de la implementación de la aplicación:** Como se comentaba previamente, debido al tiempo limitado, hay algunos elementos en el código que se pueden mejorar.

Por último, existen distintas líneas de trabajo futuras que se pueden realizar para mejorar tanto el algoritmo, como la aplicación web:

- **Añadir gestión de diversidad:** En los resultados obtenidos por el algoritmo, se puede observar que las soluciones convergen rápido, por lo tanto, puede darse el caso de que se quede atrapado en un óptimo local. Es por ello que se podría implementar un mecanismo para gestionar la diversidad de la población y evitar que se converja de forma tan acelerada.
- **Realizar estudio estadístico:** En la parte final del proyecto se decidió centrarse en la implementación de la aplicación, pero se es consciente de que al trabajar con algoritmos estocásticos, es necesario la realización de estudios estadísticos sobre los resultados obtenidos, para determinar, entre otras cosas, la calidad de las configuraciones para cada una de las instancias.
- **Almacenar instancias:** Actualmente la aplicación web sólo cuenta con un *frontend*, y por lo tanto, las instancias subidas por un usuario sólo se mantienen en la aplicación mientras dure su sesión. Por lo tanto, se podría implementar un *backend* para gestionar usuarios y que estos puedan guardar sus instancias, exportarlas, e incluso modificarlas.

# Capítulo 6

## Summary and Conclusions

Once the project was ended, it is needed to make a balance of it in order to analize if the objectives were achieved, how the work was performed, the quality of the results and the possible improvements or additions that can be made in the future.

Following the enumeration given in the objectives section, the first of them consisted in analizing and indentifying the problem. This involved doing a research both for the problem, and Evolutionary Computation, in order to decide if we could add value to the investigation of the *MDCCVRP* with our point of view and what could be the best apporach. Next, an study of the state of the art was performed, but because it is a recent problem, there are not many articles about it, so the search was expanded to similar variants, which allowed us to learn techniques that were useful to the development of some of the features of the algortihm, like the crossover operator or the local search.

The main goal of the project was implementing a memetic algorithm to solve the *MDCCVRP* in an efficient way and that could give optimal or near optimal solutions. As we can see in the chapter with the obtained results, this objective could be achieved satisfactorily, achieving even a better solution than the others found in previuos studies for one of the instances. Nevertheless, as we will mention later, the implementation can be optimized to avoid repetitive calculations and improve its performance, as well as improve the algorithm to achieve better results.

Finally, we wanted to implement a simple web application in which users could solve their instances of the *MDCCVRP* and visualize graphically the results. This also was achieved, but because of the limited time, the source code can also be improved. Another aspect that could be improved is the *User Experience* (*UX*), because the knowledge about the subject and the time to perform a suitable research was lacking.

Therefore, we can say that all the objectives were achieved, but we are also conscious of the improvements that can be applied, as we list below:

- **Improvement of the documentation:** Because of the shortage of time, the documentation for the last elements is low or null, and it is desirable to have the full documentation in order to facilitate the usage for the user.
- **Optimize the algorithm:** There are repetitive calculations in the implementation that can be optimized, improving the algorithm performance.

- **Improvement of the implementation for the application:** As we mentioned previously, because of the lack of time, the are some aspects of the code that can be improved.

Finally, there are lines of future work that can be done to improve the algorithm and the web application:

- **Add diversity control:** In the results obtained by the algortihm, we can see that the solutions converge fast, and because of that, the algorithm can get trapped in a local optimal. For this, a mechanism to control de diversity can be added, and avoid that it converges too fast.
- **Perform an statistical study:** At the end of the project, the focus was on the implementation of the application, but we are aware that working with stochastic algorithms, it is needed to perform a statistical study to determine, amoing other things, the quality of the configurations for each instance.
- **Save the instances:** The web application only has a *frontend*, and therefore, the instances uploaded by the users are lost once the user closes the tab of the application. Because of this, we could implement a *backend* in order to manage users and allow them to save their instances, export, or even modify them.

# Capítulo 7

# Presupuesto

Para elaborar el presupuesto se han diferenciado dos partes, por un lado, los costes humanos, y por otro, los tecnológicos, sobre todo para el alojamiento tanto de la documentación, como de la aplicación web. Para los costes humanos, se ha estimado un total de 480 horas, y un coste de 15€ por hora de trabajo, y los costes de *hosting* se estiman para un año, como se muestra a continuación:

| **Tipos** | **Descripción** | **Coste** |
|---|---|---|
| Trabajo de desarrollo | Total de 480 horas | 7200€ |
| Dominio | Dominio para el*Hosting* | 20€ |
| *Hosting* de la documentación | *Hosting* para la página de documentación | 180€ |
| *Hosting* de la aplicación web | *Hosting* para la aplicación web | 180€ |
| | | 7580€ |

Tabla 7.1: Resumen de tipos

# Bibliografía


[1] "Problema de enrutamiento de vehículos." https://es.wikipedia.org/wiki/Problema_de_enrutamiento_de_veh%C3%ADculos. [Online; Accedido 29-Agosto-2021].

[2] "Problema del viajante de comercio." https://es.wikipedia.org/wiki/Problema_del_viajante. [Online; Accedido 31-Agosto-2021].

[3] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, 1959.

[4] S. U. Ngueveu, C. Prins, and R. W. Calvo, "An effective memetic algorithm for the cumulative capacitated vehicle routing problem," *Computers and Operations Research*, vol. 37, 2010.

[5] E. Lalla-Ruiz and S. Voß, "A popmusic approach for the multi-depot cumulative capacitated vehicle routing problem," *Optimization Letters*, vol. 14, pp. 671–691, 4 2020.

[6] A. Eiben and J. Smith, "Introduction to evolutionary computing," 2015.

[7] C. Darwin, *On the Origin of the Species*, vol. 5. 1859.

[8] "GeneticsJS." https://github.com/GeneticsJS/GeneticsJS/. [Online; Accedido 01-09-2021].

[9] "TypeScript." https://www.typescriptlang.org/. [Online; Accedido 01-09-2021].

[10] "NodeJS." https://nodejs.org/es/about/. [Online; Accedido 01-09-2021].

[11] "NPM." https://docs.npmjs.com/about-npm. [Online; Accedido 01-09-2021].

[12] "Visual Studio Code." https://code.visualstudio.com/. [Online; Accedido 01-09-2021].

[13] "Prettier." https://prettier.io/docs/en/index.html. [Online; Accedido 01-09-2021].

[14] "ESLint." https://eslint.org/. [Online; Accedido 01-09-2021].

[15] "Git." https://git-scm.com/. [Online; Accedido 01-09-2021].

[16] "GitHub." https://github.com/. [Online; Accedido 01-09-2021].

[17] "Jest." https://jestjs.io/es-ES/. [Online; Accedido 01-09-2021].

[18] “TypeDoc.” https://typedoc.org/. [Online; Accedido 01-09-2021].

[19] “NextJS.” https://nextjs.org/. [Online; Accedido 01-09-2021].

[20] “React.” https://es.reactjs.org/. [Online; Accedido 01-09-2021].

[21] “Stitches.” https://stitches.dev/docs/introduction. [Online; Accedido 01-09-2021].

[22] “react-dropzone.” https://react-dropzone.js.org/. [Online; Accedido 01-09-2021].

[23] “react-chartjs-2.” https://www.npmjs.com/package/react-chartjs-2. [Online; Accedido 01-09-2021].

[24] D. Moher, A. Liberati, J. Tetzlaff, and D. G. Altman, “Preferred reporting items for systematic reviews and meta-analyses: The prisma statement,” *PLoS Medicine*, vol. 6, p. e1000097, 7 2009.

[25] X. Wang, T. M. Choi, Z. Li, and S. Shao, “An effective local search algorithm for the multidepot cumulative capacitated vehicle routing problem,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 12, pp. 4948–4958, 2020.

[26] S. U. Ngueveu, C. Prins, and R. W. Calvo, “An effective memetic algorithm for the cumulative capacitated vehicle routing problem,” *Computers and Operations Research*, vol. 37, pp. 1877–1885, 11 2010.

[27] G. M. Ribeiro and G. Laporte, “An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem,” *Computers and Operations Research*, vol. 39, pp. 728–735, 3 2012.

[28] N. Smiti, M. M. Dhiaf, B. Jarboui, and S. Hanafi, “Skewed general variable neighborhood search for the cumulative capacitated vehicle routing problem,” *International Transactions in Operational Research*, vol. 27, pp. 651–664, 1 2020.

[29] L. Ke and Z. Feng, “A two-phase metaheuristic for the cumulative capacitated vehicle routing problem,” *Computers and Operations Research*, vol. 40, pp. 633–638, 2 2013.

[30] J. F. Sze, S. Salhi, and N. Wassan, “The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search,” *Transportation Research Part B: Methodological*, vol. 101, pp. 162–184, 7 2017.

[31] L. Ke, “A brain storm optimization approach for the cumulative capacitated vehicle routing problem,” *Memetic Computing*, vol. 10, pp. 411–421, 12 2018.

[32] C. M. Abrante Dorta, “Framework web de computación evolutiva.” https://riull.ull.es/xmlui/handle/915/14535, 2019. [Online; Accedido 06-09-2021].

[33] F. A. Cruz Zelante, “Planificación optimizada de un sistema de semáforos mediante algoritmos evolutivos: una aplicación a la rotonda del Padre Anchieta, en Santa Cruz de Tenerife.” http://riull.ull.es/xmlui/handle/915/21321, 2020. [Online; Accedido 06-09-2021].

[34] “Iterators and Generators.” https://www.typescriptlang.org/docs/handbook/iterators-and-generators.html. [Online; Accedido 06-09-2021].

[35] “Ejemplo del problema de la mochila con GeneticsJS.” https://github.com/GeneticsJS/GeneticsJsKnapsack. [Online; Accedido 01-09-2021].

[36] C. Prins, “A simple and effective evolutionary algorithm for the vehicle routing problem,” *Computers Operations Research*, vol. 31, no. 12, pp. 1985–2002, 2004.