Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Desarrollo de algoritmos dirigido por retos. Planificación optimizada de entrega de asteroides

Challenge-driven algorithms development. Optimised asteroid delivery scheduling

Alberto Rios de la Rosa

La Laguna, 22 de mayo de 2024

D. **Eduardo Manuel Segredo González**, Profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

D. **Alejandro Marrero Díaz**, investigador postdoctoral 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:

”*Desarrollo de algoritmos dirigido por retos. Planificación optimizada de entrega de asteroides*”

ha sido realizada bajo su dirección por D. **Alberto Rios de la Rosa**.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 22 de mayo de 2024.

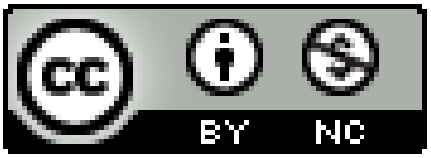
Agradecimientos

Quiero expresar mi sincero agradecimiento a todas las personas que contribuyeron de alguna manera al desarrollo y finalización de este Trabajo de Fin de Grado.

A mis tutores Eduardo Manuel Segredo y Alejandro Marrero, por su orientación y apoyo constante que fueron fundamentales para llevar a cabo este trabajo de manera exitosa.

Y agradecer a mis padres por su amor incondicional, su constante apoyo y su sacrificio para brindarme las oportunidades que me han permitido llegar hasta aquí.

Licencia



Resumen

El problema de entrega de asteroides es un desafío crucial en la exploración y aprovechamiento de recursos espaciales, donde la eficiencia en la planificación de rutas y la asignación de recursos son de suma importancia. La computación evolutiva ofrece una herramienta potente para abordar este problema, ya que permite encontrar soluciones de alta calidad mediante la simulación de procesos evolutivos como la selección natural y la reproducción. Al aplicar algoritmos evolutivos a este problema, se pueden explorar y optimizar diversas estrategias de recolección y entrega de recursos de los asteroides, teniendo en cuenta múltiples variables como la disponibilidad de recursos, la distancia entre asteroides y puntos de entrega, y las restricciones logísticas y técnicas. Este enfoque no solo puede mejorar la eficiencia en la explotación de recursos espaciales, sino también abrir nuevas posibilidades para la exploración y colonización del espacio exterior.

Palabras clave: Computación evolutiva, algoritmo memético, optimización, planificación, programación competitiva, agencia espacial europea.

Abstract

The asteroid delivery scheduling problem is a crucial challenge in the exploration and utilization of space resources, where efficiency in route planning and resource allocation is of paramount importance. Evolutionary computation offers a powerful tool for addressing this problem, as it allows for finding high-quality solutions by simulating evolutionary processes such as natural selection and reproduction. By applying evolutionary algorithms to this problem, various strategies for asteroid resource collection and delivery can be explored and optimized, taking into account multiple variables such as resource availability, distance between asteroids and delivery points, and logistical and technical constraints. This approach can not only improve efficiency in space resource exploitation but also open new possibilities for space exploration and colonization.

Keywords: Evolutionary computation, memetic algorithm, optimization, scheduling, competitive programming, european space agency.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

En este capítulo se tiene como objetivo presentar el problema de optimización que se llevará a cabo en este trabajo, además de introducir los conceptos necesarios para poder comprenderlo. A su vez se llevará una explicación sobre los objetivos establecidos en el proyecto y las contribuciones realizadas.

1.1. Modelo de caja negra

Un **modelo de caja negra o black-box model** [10] se trata de un sistema en el que se espera introducir una o múltiples entradas de información proveniente del exterior. Ya recibida, esta información se procesa a través de un modelo computacional, para el que los usuarios desconocen sus detalles (es por esto el nombre del modelo). Una vez procesada la información de entrada, se obtienen una o varias salidas diferentes.

En esencia, el sistema de una caja negra esta dividido en tres partes fundamentales, la entrada, el modelo y la salida. Dependiendo de la información de la que se disponga para esos tres componentes, se distinguirán tres tipos de problemas: optimización, modelización y simulación.

Se trata de un **problema de modelización** cuando se tiene noción de las entradas y sus respectivas salidas, pero se desconoce el modelo de computación que se necesita para poder entregar la salida correcta para cada entrada conocida.

Se habla de un **problema de simulación** en aquellas situaciones en las que se conoce la entrada y el modelo, pero se desconoce su salida.

Y por último nos encontramos ante un **problema de optimización** cuando lo que se conoce es el modelo y la salida o una descripción de esta, y se tiene como objetivo calcular la entrada o posibles entradas que devuelvan la salida deseada.

1.2. Modelo de optimización

Los modelos de optimización [9] son herramientas matemáticas fundamentales que permiten abordar problemas complejos en diversas áreas, desde la gestión empresarial hasta la ingeniería y la logística. Estos modelos proporcionan un marco estructurado para tomar decisiones informadas al buscar la mejor solución posible entre múltiples opciones, considerando limitaciones y objetivos específicos.

1.2.1. Utilidades y ventajas

En diversos contextos, los modelos de optimización han demostrado ser herramientas esenciales para mejorar la eficiencia y la toma de decisiones en situaciones complejas. Estos modelos, basados en algoritmos matemáticos y técnicas específicas, son aplicados para resolver problemas de asignación de recursos, planificación temporal y optimización global, entre otros muchos.

Las ventajas clave incluyen:

- **Mejora de la eficiencia**: Al optimizar la asignación de recursos, se mejora la eficiencia global del sistema o proceso, maximizando los resultados deseados.
- **Toma de decisiones informada**: Proporcionan una base cuantitativa para la toma de decisiones, permitiendo evaluar diferentes estrategias y escenarios.
- **Adaptabilidad a restricciones**: Son adaptables a restricciones específicas del problema, permitiendo abordar una variedad de situaciones y condiciones.
- **Optimización de recursos financieros**: Contribuyen a una asignación más eficiente de recursos financieros, evitando ineficiencias y maximizando los beneficios.
- **Mejora continua**: Al identificar patrones y oportunidades, estos modelos permiten implementar mejoras continuas en diversos procesos y operaciones.

Algunos problemas de optimización muy conocidos que llevan a cabo una combinatoria de un objetivo son el problema de viajante de comercio, donde el objetivo es encontrar la ruta más corta que visita un conjunto dado de ciudades exactamente una vez y regresa al punto de partida, o el problema de planificación de proyectos, donde el objetivo es asignar recursos a tareas en un proyecto para minimizar la duración total del proyecto.

En resumen, los modelos de optimización ofrecen una metodología efectiva y versátil para abordar problemas complejos en distintos dominios, mejorando la eficiencia y facilitando la toma de decisiones fundamentada.

1.2.2. Reto

En este Trabajo de Fin de Grado (TFG) se aborda un problema de optimización dentro del marco de un reto propuesto en la *Genetic and Evolutionary Computation Conference (GECCO)* [7] de 2022, uno de los congresos más importantes a nivel mundial [8] que reúne a investigadores, académicos y profesionales de todo el mundo que trabajan en áreas relacionadas con la computación evolutiva.

El reto titulado *Space Optimisation Competition (SpOC)* [13] trata de múltiples problemas de optimización aplicados en distintos escenarios espaciales, con un enfoque en meta-heurísticas y optimización de caja negra donde cada problema difiere en naturaleza y objetivo, por lo que un mismo enfoque de solución no puede ser aplicable a todos ellos. De entre los distintos problemas que se proponen, en este TFG nos centraremos en el que tiene como título planificación de entrega o *(Delivery Scheduling)*, el cual se trata de un problema de optimización combinatoria de un solo objetivo, donde se busca encontrar la mejor solución entre un conjunto finito de posibles soluciones, y se evalúan esas soluciones en función de un único objetivo que consistente en maximizar la masa mínima recolectada de tres materiales diferentes entre todas las estaciones de procesamiento.

1.3. Computación evolutiva

En este TFG nos centraremos en aplicar heurísticas y algoritmos evolutivos para resolver el problema propuesto, todo gracias a la computación evolutiva que ofrece una sólida base teórica y práctica para abordar problemas complejos de optimización, como los planteados en el reto *Space Optimisation Competition (SpOC)*. Las heurísticas proporcionan soluciones aproximadas rápidamente, mientras que los algoritmos evolutivos, inspirados en los procesos biológicos de evolución, permiten una exploración exhaustiva del espacio de soluciones, aumentando la probabilidad de encontrar soluciones cercanas a la óptima, e incluso, la óptima.

La Computación Evolutiva [5] es un área de investigación dentro de las Ciencias de la Computación inspirada en los procesos evolutivos observados en la naturaleza. Este enfoque se basa en la idea fundamental de evolucionar soluciones a problemas complejos utilizando principios similares a los procesos biológicos de selección natural y reproducción. A través de algoritmos evolutivos, la computación evolutiva busca generar soluciones o individuos cada vez mejores a lo largo de generaciones o iteraciones de los algoritmos, mejorando así la adaptación de los individuos a su entorno, o lo que es lo mismo, proporcionar mejores soluciones al problema que está siendo optimizado.

La computación evolutiva se inspira en la evolución natural, la cual se percibe como un intrincado proceso en el cual una población de individuos busca adaptarse al entorno para garantizar su supervivencia y la transmisión de sus características a la siguiente generación. La supervivencia del más apto, un principio fundamental introducido por Charles Darwin [4], dicta que aquellos individuos mejor adaptados son los que tienen más posibilidades de crear descendencia, asegurando así la continuidad de los rasgos que han facilitado su supervivencia.

La analogía de la evolución natural con la computación evolutiva radica en la aplicación de estos principios en la resolución de problemas computacionales. Al igual que en la naturaleza, los algoritmos evolutivos buscan soluciones más eficaces adaptándose dinámicamente a su entorno. La selección natural, la reproducción mediante un operador de cruce, y la introducción de mutaciones proporcionan un marco flexible para la búsqueda y mejora de soluciones.

Es crucial destacar que, al igual que los rasgos genéticos evolucionan con mutaciones, las soluciones en la computación evolutiva no son estáticas, experimentan cambios a lo largo del tiempo para enfrentar nuevos desafíos. Este enfoque innovador ha llevado a la computación evolutiva a ser una opción atractiva para resolver problemas complejos en constante evolución. La capacidad de adaptación y mejora continua son las piedras angulares que vinculan de manera intrigante la evolución natural y la eficacia de la computación evolutiva.

En la naturaleza, la aptitud de un organismo se relaciona con su capacidad para adaptarse al entorno, obtener recursos y sobrevivir. Así, un individuo más apto tiene mayores posibilidades de sobrevivir y reproducirse junto a otro individuo seleccionado de manera similar. En contraste, los individuos menos aptos tienen menos probabilidad de sobrevivir y reproducirse.

El cruce entre individuos genera descendencia con algunas de las características de sus progenitores. En consecuencia, con cada generación, los individuos tienden a tener mejores características que las generaciones anteriores. Este proceso favorece la exploración de soluciones más prometedoras en el espacio de búsqueda al abordar problemas con esta técnica.

De esta manera, si se diseña adecuadamente y el caso de uso lo permite, un algoritmo evolutivo podría converger hacia la solución óptima, ya que la población tiende a mejorar a lo largo del tiempo.

1.4. Objetivos

Con el objetivo de establecer un marco aproximado para el proyecto, se establecieron metas que facilitaran la división de las diversas etapas de desarrollo y proporcionaran una secuencia lógica para su ejecución. A continuación, se enumeran estos objetivos junto con breves descripciones que las ponen en contexto:

- **Investigación y estado del arte del problema**: En este apartado se busca ampliar conocimientos acerca de la computación evolutiva y todo lo que hay tras ella, así como buscar información sobre el problema escogido y encontrar diferentes enfoques que pueden ser viables para resolverlo.
- **Implementación del problema a optimizar, además de un algoritmo básico como, por ejemplo, búsqueda aleatoria, que lo resuelva**: El trabajo gira en torno al desarrollo de algoritmos dirigidos por retos, por lo que para tener una primera aproximación al problema y tener nociones acerca de las soluciones al mismo, se propuso implementar un algoritmo de búsqueda aleatoria.
- **Implementación de diferentes técnicas heurísticas**: Teniendo ya una primera aproximación del problema, se propondrá diferentes heurísticas buscando obtener mejores soluciones.
- **Implementación de un algoritmo memético que permita llevar a cabo la resolución del problema**: Por otro lado, se ha propuesto también el diseño de un algoritmo memético que resuelva el problema y que nos permita obtener soluciones de más alta calidad.
- **Diseño, ejecución y análisis del rendimiento de los algoritmos a través de la evaluación experimental**: Por último, se propuso realizar una evaluación experimental de los distintos algoritmos implementados. Esta evaluación servirá para proporcionar un análisis completo acerca del rendimiento de los mismos a la hora de resolver el problema.

Capítulo 2

Estudio del estado del arte

Antes de llevar a cabo la resolución del problema se ha hecho un estudio previo del estado del arte del mismo. Para ello, se realizó una búsqueda sobre artículos o publicaciones relacionadas, pero al tratarse de un problema reciente propuesto por el equipo de conceptos avanzados de la Agencia Espacial Europea en GECCO, no había mucha información al respecto.

Lo que se pudo encontrar fue un sitio web sobre un workshop [12] que tuvo lugar en el mes de septiembre de 2022 sobre los problemas de la SpOC, en la que se encontraron algunas presentaciones utilizadas en la ponencia de presentación de resultados de los equipos participantes en la competición.

Dado que solo se conoce esta web sobre el problema que se va a tratar en este trabajo, a continuación, se procederá al análisis de las diferentes conclusiones a las que se llegaron.

En la presentación *"Metaheuristics based on local search and populations for the Delivery Scheduling problem"* [1], se describe un enfoque para resolver el problema de planificación de entregas de asteroides a estaciones de procesamiento en escenarios de minería espacial utilizando la meta-heurística Ant Colony Optimization (ACO).

La presentación comienza con la motivación detrás del problema, que surge en el contexto de la exploración y explotación de recursos en asteroides y la necesidad de optimizar la entrega de materiales a las estaciones de procesamiento. Luego, describe detalladamente el problema y las restricciones que deben ser consideradas.

La solución propuesta se basa en el algoritmo ACO, que es una meta-heurística inspirada en el comportamiento de las colonias de hormigas en la naturaleza. El algoritmo ACO construye soluciones iterativamente, utilizando información sobre la calidad de las soluciones previas para guiar la búsqueda hacia soluciones de mejor calidad.

También se describe cómo se adaptó el algoritmo ACO al problema de programación de entregas de asteroides y presenta los detalles del diseño y configuración del algoritmo, incluyendo la codificación de soluciones, la definición de heurísticas, y la implementación de operadores de búsqueda.

Los autores realizaron una evaluación de el rendimiento del algoritmo propuesto en un conjunto de instancias del problema y compararon los resultados con los obtenidos por otros algoritmos de la literatura. Los resultados mostraron que el algoritmo ACO propuesto fue capaz de encontrar soluciones de alta calidad en tiempos razonables de cómputo.

En resumen, este trabajo presenta una solución innovadora y eficiente para el problema de programación de entregas de asteroides utilizando la meta-heurística ACO. Los resultados obtenidos demostraron la efectividad del algoritmo propuesto y su potencial para su aplicación en futuras misiones de minería espacial.

Capítulo 3

Definición del problema

En este apartado se abordará en mayor profundidad la definición del problema propuesto por la SPOC en la GECCO 2022 [13] que se va a llevar a cabo en el TFG. Se trata del tercero de los tres problemas que propone la competición de optimización en el espacio considerando un hipotético escenario futurista en un sistema solar llamado Trappist-1 [2].

3.1. Descripción de Space Optimisation Competition - SpOC

El problema [13] cuenta que desde los albores de los viajes espaciales, los humanos han estado soñando con dejar su rincón provincial de la Vía Láctea y explorar el resto de la galaxia. En el año 3275, más de 100 años después de la invención de la innovadora tecnología de propulsión que permite el viaje interestelar, este sueño finalmente se está convirtiendo en realidad. Con el lanzamiento de varias sondas robóticas sofisticadas, los humanos han dado los primeros pasos para establecer un asentamiento humano en uno o más planetas en otro sistema solar: Trappist-1

Trappist-1 es un sistema estelar pequeño pero curioso. Se sospecha que alberga hasta siete planetas terrestres, cuatro de los cuales probablemente se encuentren en la zona habitable donde podría existir agua líquida. No hace falta decir que esto es de gran interés para nuestros exploradores del siglo 33. Sin embargo, el agua por sí sola no será suficiente para sostener un asentamiento humano. En reconocimiento de eso, los científicos han preparado una serie de misiones utilizando sondas automáticas para evaluar la perspectiva de un asentamiento exitoso y establecer las bases para la llegada de los humanos. Como todas las misiones a gran escala a largo plazo, este esfuerzo se divide en fases individuales separadas: una fase de exploración del sistema, una fase de adquisición de recursos y una fase de entrega de recursos.

Estas fases representan a cada uno de los problemas que se proponen en la competición. A continuación, se describirán brevemente cada una de las fases, centrándonos luego en la fase de entrega de recursos que será el problema que se tratará a lo largo del TFG.

3.1.1. Exploración del sistema (Trappist Tour)

La misión pionera es un recorrido exploratorio del sistema que tiene como objetivo evaluar el planeta (o planetas) más adecuado para los colonos. Se envía una sonda a Trappist-1 con la tarea explícita de averiguar el punto de liquidación óptimo. Como este sería efectivamente uno de los primeros esfuerzos de exploración del espacio profundo

de la humanidad, todavía hay un fuerte enfoque en realizar la tarea lo más rápido posible usando la menor cantidad de combustible posible.

3.1.2. Minería del cinturón de asteroides (Mine the Belt)

Durante el recorrido planetario, se descubre un cinturón de asteroides en las afueras del sistema Trappist-1, más allá de la órbita del planeta más exterior. En un afortunado giro del destino, los asteroides contienen una rica fuente de materiales considerados esenciales para asentarse, incluida la materia prima utilizada para el exótico sistema de propulsión de la propia sonda. Por lo tanto, la segunda fase de la misión está orientada a extraer la mayor cantidad posible de tres materiales básicos en un corto período de tiempo.

3.1.3. Planificación de entrega de asteroides (Delivery Scheduling)

Mientras el primer pequeño grupo de colonos viaja hacia su nuevo hogar, en la Tierra, los científicos e ingenieros expresan cada vez más su preocupación de que, si bien los materiales extraídos en la fase anterior serán suficientes para un asentamiento a corto plazo, el establecimiento a largo plazo en Trappist-1 requerirá muchos más materiales y combustible que los extraídos. Por lo tanto, en la tercera fase de la misión se identifican una serie de asteroides clave como objetivos para transferirlos a una órbita mucho más cercana al planeta que se está asentando. Se lanzan sondas especiales no tripuladas que pueden resistir los efectos de la aceleración titánica con la tarea de adelantar a los colonos y enviar tantos asteroides como sea posible a una nueva órbita en el corto tiempo antes de que los humanos lleguen a su destino.

3.2. Problema de planificación de entrega de asteroides

En este apartado se definirá de forma detallada toda la información que nos proporciona la SpOC [6] para poder llevar a cabo la fase de entrega de recursos. Se trata de un problema de optimización combinatoria de un solo objetivo, cuya meta final es la de programar la entrega de 340 asteroides desde un cinturón de asteroides hipotético a doce estaciones de procesamiento robótico ubicadas en órbita en el sistema TRAPPIST-1 de la manera más eficiente posible.

3.2.1. Descripción

Los asteroides del cinturón trapense son cuerpos celestes rocosos que orbitan alrededor de la estrella enana roja Trappist-1. Algunos de estos asteroides contienen grandes cantidades de materiales particulares de interés, que se han identificado como A, B y C. Estos materiales pueden ser valiosos para la colonización humana del sistema Trappist-1, ya que pueden ser utilizados en la construcción de estructuras, naves espaciales, y para otros fines.

Para recolectar y procesar estos materiales, se han desplegado doce estaciones de procesamiento robótico en el sistema Trappist-1. Cada una de estas estaciones solo puede estar activa durante un intervalo de tiempo determinado, que se indica como su "ventana de actividad". Durante esta ventana, la estación solo puede recibir materiales y

procesarlos para su uso. Cada estación de procesamiento solo puede estar activa durante su ventana de actividad y solo puede recibir materiales durante ese tiempo.

Para permitir la salida de órbita de los asteroides y la transferencia a las estaciones de procesamiento, se han desplegado pequeñas unidades de propulsión en todos los asteroides candidatos. Sin embargo, estas unidades requieren la producción de propulsor mediante el procesamiento de los materiales A, B y C. Para cada asteroide candidato, se ha identificado un número finito de posibles oportunidades de transferencia a las diferentes estaciones de procesamiento. La cantidad de material entregado varía según la oportunidad de transferencia elegida, ya que parte del material se procesa para crear propulsor para la transferencia.

El objetivo del programa es maximizar la masa mínima recolectada de los tres materiales en todas las estaciones de procesamiento en 80 días. Esto significa que se debe tratar de recolectar la mayor cantidad posible de los tres materiales en todas las estaciones de procesamiento durante el período de tiempo asignado. Para lograr este objetivo, se debe programar cuidadosamente la entrega de los asteroides candidatos a las estaciones de procesamiento, eligiendo la oportunidad de transferencia adecuada para cada asteroide y estación de procesamiento.

3.2.2. Datos

El problema proporciona una base de datos en forma de diccionario de Python, que contiene una lista de asteroides candidatos y las oportunidades de entrega a un subconjunto de las estaciones de procesamiento para cada asteroide. Esta base de datos debe tener el nombre “candidates.txt” y estar ubicada en la ruta relativa ”./data/spoc/scheduling/candidates.txt” en el directorio de trabajo para poder utilizar el código de evaluación.

Cada asteroide se identifica por un número entero único en el rango de 1 a 340, denominado “asteroid_id”. De manera similar, las estaciones de procesamiento se identifican mediante un número entero único en el rango de 1 a 12, denominado "station_id", donde la primera estación se indexa con el número 1.

Para un par dado asteroide-estación ($asteroid\_id$, $station\_id$), las oportunidades de entrega se representan como una lista de 4 elementos $[T\_arr, metro_1, metro_2, metro_3]$, donde T_arr es la época de llegada a la estación y $metro_1$, $metro_2$ y $metro_3$ representan las masas entregadas de los materiales A, B y C, respectivamente. Cada oportunidad de entrega se identifica con un número entero único en el rango de 1 a 8, denominado $opportunity\_id$. Por lo tanto, una oportunidad de entrega se puede identificar por una tupla de tres números enteros $[asteroid\_id, station\_id, opportunity\_id]$. Se debe tener en cuenta que la primera oportunidad se indexa con el número 1.

Conceptualmente, la base de datos tiene la siguiente estructura:

```
{
    "ID de asteroide" :
    {
        "ID de estacion" : [[ "Hora de llegada" , "Masa1" , "Masa2" , "Masa3" ],
                            [ "Hora de llegada" , "Masa1" , "Masa2" , "Masa3" ],
                            ... ],
        "ID de estacion" : [[ "Hora de llegada" , "Masa1" , "Masa2" , "Masa3" ],
                            [ "Hora de llegada" , "Masa1" ,"Masa2" , "Masa3" ],
                            ... ],
```

```
    },
    "ID de asteroide" :
    {
        "ID de estacion" : [[ "Hora de llegada" , "Masa1" , "Masa2" , "Masa3" ],
                             [ "Hora de llegada" , "Masa1" , "Masa2" , "Masa3" ],
                             ... ],
        "ID de estacion" : [[ "Hora de llegada" , "Masa1" , "Masa2" , "Masa3" ],
                             [ "Hora de llegada" , "Masa1" ,"Masa2" , "Masa3" ],
                             ... ],
    },
}
```

A continuación, se muestra un ejemplo de cómo se representa un asteroide en esta estructura:

```
{
    "42" :
    {
        "5" :  [[ 17.34, 0.012923, 0.045365, 0.093846 ],
                [ 38.77, 0.010074, 0.039874, 0.083761 ]]
    }
}
```

En este ejemplo, se muestra la información para el asteroide 42 y la estación de procesamiento 5. Se puede observar que hay dos oportunidades de entrega disponibles para este asteroide en esta estación. La primera oportunidad tiene una época de llegada de 17.34 días y las masas entregadas de los materiales A, B y C son 0.012923, 0.045365 y 0.093846, respectivamente. La segunda oportunidad tiene una época de llegada de 38.77 días y las masas entregadas de los materiales A, B y C son 0.010074, 0.039874 y 0.083761, respectivamente. Cada oportunidad se identifica mediante la tupla $[asteroid\_id, station\_id, opportunity\_id]$. En este caso, la primera oportunidad se identifica con la tupla [42, 5, 1], mientras que la segunda oportunidad se identifica con la 3-tupla [42, 5, 2].

3.2.3. Codificación de soluciones

El vector de decisión x que se utilizará en el problema de optimización consta de dos partes:

- La primera parte del vector contiene las épocas inicial y final que definen las ventanas de actividad de cada una de las 12 estaciones disponibles para la transferencia de asteroides. Cada ventana de actividad está definida por dos épocas, una de inicio y otra de finalización, y hay un total de 24 épocas en este vector de decisión. Todas las épocas son números decimales en un rango de 0 a 80. Es importante destacar que las épocas se ordenan según la identificación de cada estación ($station\_id$), por lo que no necesariamente están en orden cronológico.
- La segunda parte del vector de decisión contiene las asignaciones de asteroides a estaciones. Cada asignación está compuesta por una tripleta de valores enteros $[asteroid\_id, station\_id, opportunity\_id]$,tal y como se ha definido anteriormente, las cuales definen lo siguiente:

- El número de identificación $asteroid\_id$ del asteroide que se está asignando. Este número debe estar en el rango de 1 a 340.
- El número de identificación $station\_id$ de la estación a la que se está asignando el asteroide. Este número debe estar en el rango de 0 a 12. Los asteroides no asignados que no se entregan a ninguna estación se asociarán a un valor de $station\_id$ igual a 0.
- El número de identificación $opportunity\_id$ indica qué oportunidad de transferencia se utilizará para la asignación de la $asteroid\_id$ a la $station\_id$ correspondiente. No todos los pares $(asteroid\_id, station\_id)$ tienen la misma cantidad de oportunidades de transferencia disponibles, y hay como máximo 8 oportunidades de transferencia por par. Es importante destacar que la $opportunity\_id$ utilizada en la asignación debe corresponder a una asignación existente; de lo contrario, la asignación se marcará como inviable. Los asteroides no asignados, asociados a un valor de $station\_id$ igual a 0, pueden asignarse a cualquier $opportunity\_id$ en el rango de 1 a 8, y no entregarán masa.

El vector de decisión completo es la concatenación de todas las entradas de la primera y segunda parte. Esto significa que el vector de decisión es de longitud fija y consta de 1044 variables de decisión en total. Todos los asteroides en la base de datos deben estar representados en este vector de decisión, incluso aquellos que no se asignan a ninguna estación y, por lo tanto, tienen asociado un valor de $station\_id$ igual a 0.

3.2.4. Restricciones

Se listarán aquellas restricciones que nos impone en el problema de entrega de recursos:

- Cada estación $k = 1, 2, \ldots, 12$ solo puede estar activa durante un intervalo de tiempo dado, que está definido por su ventana de actividad $[T_{ki}, T_{kf}]$. El tiempo de inicio T_{ki} y el tiempo de finalización T_{kf} deben cumplir la condición de $0 \leq T_{ki} \leq T_{kf} \leq 80$. Esto se aplica a través de los límites del problema y garantiza que todas las entregas de asteroides a una estación deben ocurrir dentro del tiempo especificado y que las estaciones no pueden aceptar asteroides después del límite de tiempo de 80 días. Matemáticamente, esta restricción se puede expresar como:

$$0 \leq T_{ki} \leq T_{kf} \leq 80, para\ k = 1, ..., 12.$$

- Solo puede haber una estación activa en cualquier momento, lo que significa que las ventanas activas de las estaciones no pueden superponerse. Además, solo se pueden asignar asteroides a estaciones que estén en su ventana activa en ese momento. La secuencia en la que las estaciones reciben asteroides no está restringida y puede ser cualquier permutación de los números del 1 al 12.

  Matemáticamente, esta restricción se expresa como:

$$[T_{ji}, T_{jf}] \cap [T_{ki}, T_{kf}] = \emptyset \quad \text{para } j, k = 1, \ldots, 12 \text{ y } j \neq k$$

Donde T_{ji} y T_{jf} son los tiempos de inicio y finalización de la ventana activa de la estación j. Esta restricción asegura que no hay superposición entre las ventanas activas de las estaciones, lo que garantiza que solo una estación esté activa en cualquier momento.

- Cada asteroide solo se puede asignar una vez y las asignaciones deben ser coherentes con las oportunidades proporcionadas en la base de datos de candidatos. Implementado como dos restricciones de igualdad, satisfechas cuando su valor es cero:

  - eq_constraint_1 = udp.fitness$(x)[1]$ representa el número de asteroides restantes o incorrectamente indexados en el vector de decisión x.
  - eq_constraint_2 = udp.fitness$(x)[2]$ representa el número de violaciones de asignación de asteroides en el vector de decisión x, es decir, el número de tuplas [asteroid_id, station_id, opportunity_id] con valores para el campo opportunity_id no válidos y que, por lo tanto, no son consistentes con la base de datos.

- La restricción del intervalo de tiempo indica que debe haber, al menos, un día de separación entre la ventana activa de una estación de procesamiento y la siguiente. Es decir, el tiempo entre el final de la ventana activa de una estación y el comienzo de la ventana activa de la siguiente estación debe ser mayor o igual que 1.

  Para todas las estaciones consecutivas j y k, la ventana activa de la estación k debe comenzar al menos un día después del final de la ventana activa de la estación j. Matemáticamente, esto se expresa como $T_{kyo} - T_{jf} > 1$, donde T_{kyo} es el tiempo de inicio de la ventana activa de la estación k y T_{jf} es el tiempo de finalización de la ventana activa de la estación j. Esta restricción asegura que los asteroides se entreguen en un horario ordenado y sin retrasos innecesarios.

  Lo anterior se trata de una restricción de desigualdad, calculada como uno menos el intervalo de tiempo mínimo entre estaciones y que se satisface cuando es menor o igual que cero (ineq_constraint_1 = udp.fitness$(x)[3]$).

- Los asteroides enviados a una estación determinada deben tener horas de llegada incluidas en la ventana de actividad de la estación. Una vez que una estación deja de recibir nuevos asteroides y otra estación abre sus puertas, no se le pueden enviar más asteroides durante el resto del período de 80 días.

  Denotemos, para una estación dada identificada por el station_id k con $k \in [1, 12]$, el conjunto de tiempos de llegada de los asteroides entregados a esa estación. Entonces, esta restricción es matemáticamente equivalente a:

  $$T_{kyo} \leq T_{arr} \leq T_{kf} \; \forall \; T_{arr} \in T_k \; con \; k \in [1, 12]$$

  Lo anterior se trata de otra restricción de desigualdad que representa el número de violaciones de esta restricción entre los asteroides asignados y que se satisface cuando es cero (ineq_constraint_2 = udp.fitness$(x)[4]$).

3.2.5. Función objetivo

El objetivo de este desafío es maximizar la masa mínima recolectada entre todas las estaciones de procesamiento y entre los tres materiales M_A, M_B y M_C. Denotemos por M_{A_i}, M_{B_i} y M_{C_i} la masa total de los materiales A, B y C acumulados en la i-ésima estación después de que todos sus asteroides asignados hayan sido entregados. La masa mínima se define como:

$$M_{min} = min(min(M_{j_i}))$$

$$i = 1, ..., 12 \land j \in A, B, C.$$

Para cualquier vector de decisión x, el valor de esta función objetivo se puede obtener a través de $M_{min} = \text{udp.fitness}(x)[0]$.

Capítulo 4

Técnicas algorítmicas

En este capítulo, se presentarán y describirán las diferentes técnicas algorítmicas utilizadas en este trabajo para abordar el problema de optimización de la entrega de asteroides. Estas técnicas han sido seleccionadas y adaptadas cuidadosamente con el objetivo de encontrar soluciones eficientes y efectivas que optimicen la asignación de recursos en el espacio. A lo largo de este capítulo, se analizarán en detalle cada una de estas técnicas, explorando sus fundamentos teóricos, su implementación práctica y su desempeño en la resolución del problema propuesto.

4.1. Algoritmo de búsqueda aleatoria

Primero se ha implementado una búsqueda aleatoria para ver si mediante esta técnica se pueden obtener soluciones factibles al problema. Para llevar a cabo esta técnica algorítmica se ha creado la clase ´DecisionVectorGenerator´, en el que se ha desarrollado las funciones necesarias para asignar aleatoriamente la estación y la oportunidad para cada asteroide, así como, el tiempo de llegada de dicha estación al asteroide.

En el constructor **(__init__)**, se especifica el número de estaciones y asteroides. Luego se llama a **load_candidates** para cargar los datos de candidatos desde un archivo JSON llamado “candidates.txt”.

Luego la función **load_candidates** lee los datos de los candidatos desde el archivo JSON, ´candidates.txt´ y los almacena en un diccionario llamado **candidates_data.**

Por último, antes de la búsqueda aleatoria tenemos la función de asignación aleatoria llamada **generate_decision_vector** la cual se usa para obtener un vector de decisión aleatorio, en el que los 24 primeros elementos de ese vector nos muestra de forma ordenada el tiempo de entrada y el de salida de cada una de las estaciones y luego contendrá de forma ordenada 340 elementos en el que cada uno será una 3-tupla que contendrá el asteroide, la estación escogida y su respectiva oportunidad. El algoritmo de asignación aleatoria de asteroides a estaciones de procesamiento en ventanas de tiempo sigue una serie de pasos estructurados para generar una distribución válida y aleatoria de las asignaciones. El algoritmo comienza creando una solución inicial con una lista de ventanas de tiempo no solapadas. Para cada asteroide, asigna una estación aleatoriamente, con la posibilidad de no asignarlo a ninguna estación (asignación cero). Si el asteroide se asigna a una estación, selecciona aleatoriamente una oportunidad dentro de las ventanas de tiempo disponibles y verifica si la oportunidad es válida. Si es válida, agrega la asignación a la lista de soluciones; de lo contrario, intenta con otra oportunidad. Este proceso se repite para cada asteroide, asegurando que todas las asignaciones sean

válidas y cumplan con los plazos de las estaciones de procesamiento. El pseudocódigo desarrollado es el que se muestra en Algoritmo 1:

Algoritmo 1: Función asignación aleatoria

```
asteroid_assignments = create_initial_solution();
ventana = generate_initial_value();
ventana = generate_remaining_values(ventana);
ventana = sort(ventana);
ventana = split_and_shuffle(ventana);
for cada asteroide do
    estacion = assign_random_station();
    if estacion ≠ 0 then
        oportunidades = determine_opportunities();
        if oportunidades disponibles then
            oportunidad = select_random_opportunity(oportunidades);
            if opportunity_valid(oportunidad) then
                asteroid_assignments.append(estacion, oportunidad);
                break;
            end
        end
    end
    else
        oportunidad = select_random_opportunity();
    end
    asteroid_assignments.append(0, oportunidad);
end
ventana.append(asteroid_assignments);
return ventana;
```

Esta función genera un vector de decisión que cumple con ciertas restricciones y representa una asignación de asteroides a estaciones y oportunidades que satisface las condiciones del problema. La generación es aleatoria, lo que nos sirve para aplicar una función de búsqueda aleatoria, como la implementada en la función **randomsearch**.

Los resultados obtenidos en las pruebas con este algoritmo no son realmente significativos, ya que simplemente se generan posibles soluciones al problema y se van evaluando, pero no se sigue una técnica concreta que permita converger hacia una solución óptima del problema. Sin embargo, si es de destacar que al elevar el número de iteraciones del algoritmo se brinda una mayor posibilidad de obtener mejores soluciones.

La función **randomsearch** realiza un número especificado de iteraciones (iterations) para generar distintos vectores de decisión aleatorios y evaluar su aptitud utilizando la función de aptitud que se nos proporciona llamada udp.fitness(x). Registra y devuelve el mejor vector de decisión encontrado y su aptitud.

El pseudocódigo del algoritmo de búsqueda aleatoria es el que se muestra en el Algoritmo 2.

Algoritmo 2: Búsqueda Aleatoria

```
1  Best = Inicializar solución candidata aleatoriamente;
2  for numero de iteraciones do
3  |   Current = Llamar a la función de asignación aleatoria;
4  |   CurrentFitness = Evaluar Current;
5  |   if CurrentFitness menor que BestFitness then
6  |   |   Best = Current;
7  |   |   BestFitness = CurrentFitness;
8  |   end
9  end
10 return Best;
```

4.2. Algoritmo heurístico *first-window*

A diferencia del algoritmo de búsqueda aleatorio, donde asignábamos de forma aleatoria una oportunidad y una estación a cada asteroide, esta heurística llamada first window se caracteriza por primero ordenar las estaciones según su tiempo de apertura, para luego asignar al asteroide la primera oportunidad que tenga disponible en la estación según su apertura. La clase desarrollada es similar a la de búsqueda aleatoria, lo que cambia es la función principal 'First_window'.

Como vemos en el Algoritmo 3 la función principal **first_window** se usa para obtener el vector de decisión resultante, en el que los 24 primeros elementos de ese vector nos muestra de forma ordenada el tiempo de entrada y el de salida de cada una de las estaciones y luego contendrá de forma ordenada 340 elementos en el que cada uno será una 3-tupla que contendrá el asteroide, la estación escogida y su respectiva oportunidad. Paso a paso la función realiza lo siguiente:

- Se inicializa la lista **asteroid_assignments** para almacenar las asignaciones de asteroides. Esta lista se utilizará para construir el vector de decisión final.
- Se cargan los datos de los candidatos desde un archivo JSON usando la función **load_candidates**.
- Se inicializan diccionarios **rango_inicial** y **rango_final** para realizar un seguimiento de la hora de llegada más temprana y más tardía para cada estación. Inicialmente, se configuran con valores extremos. También se establece a 24 el numero de ventanas, correspondiente a la fecha de inicio y final de cada estación.
- Se elige aleatoriamente un valor para representar el tiempo de inicio de la ventana. Luego, se generan los demás valores asegurándose de que haya al menos una unidad de diferencia entre ellos. Finalmente, los valores se ordenan para obtener una secuencia con 24 valores.
- La secuencia de tiempo se agrupa en pares, luego estos pares se mezclan aleatoriamente, para así tener establecido el tiempo de inicio y fin de cada una de las estaciones. Cada estación se numera y se organiza según el tiempo de apertura de sus ventanas. El resultado es una lista de números de estación, ordenados desde la estación con la ventana más temprana hasta la más tardía.

- Teniendo las estaciones ordenadas, se abre un bucle **for** para asignar a cada asteroide, donde cada asteroide (identificado por asteroid_id), se itera sobre las estaciones ordenadas (station_open). Dentro de ambos bucles obtenemos el número de oportunidades que tiene ese asteroide en la estación. Si hay oportunidades, se itera sobre ellas. Se verifica si el tiempo de la oportunidad (opportunity_time) cae dentro de la ventana de tiempo de la estación, y en el caso de que encuentre una , se asigna el asteroide a esa estación y oportunidad, y se establece assigned como **True** se rompe el bucle. En el caso de que no se haya asignado el asteroide a ninguna estación en las iteraciones anteriores (si assigned sigue siendo **False**), se asigna aleatoriamente a la estación 0 y a una oportunidad aleatoria.

- Las asignaciones de asteroides (identificación de asteroide, estación y oportunidad) se agregan a la lista de ventanas, siendo esta lista de ventanas la que se devuelve como resultado final.

La función de evaluación realiza un número especificado de iteraciones (iterations) para generar distintos vectores de decisión con la heurística comentada y evaluar su aptitud utilizando la función de aptitud que se nos proporciona llamada udp.fitness(x). Registra y devuelve el mejor vector de decisión encontrado y su aptitud.

Como veremos posteriormente en la parte de experimentos y resultados, se podrá comprobar que las soluciones que nos aporta está heurística son siempre óptimas y obtiene mejores resultados en la función aptitud.

Algoritmo 3: Función asignación first_window

```
asteroid_assignments = create_initial_solution()
start_range, end_range, num_windows = define_ranges() window = []
first_value = random(start_range, end_range) window.append(first_value)
for i in 1 to num_windows − 1 do
    new_value = random(start_range, end_range) while
    mín(|new_value − w| ≥ 1 for all w ∈ window) do
        new_value = random(start_range, end_range)
    end
    window.append(new_value)
end
window.sort() window = shuffle_pairs(window) stations = get_ordered_stations()
for asteroid_id in 1 to 340 do
    assigned = False for station in stations do
        if assigned then
            break
        end
        opportunities = get_opportunities(asteroid_id, station) if opportunities then
            for opportunity in opportunities do
                arrival_time = get_arrival_time(opportunity) if
                arrival_time ∈ station.window then
                    asteroid_assignments.append((asteroid_id, station, opportunity))
                    assigned = True break
                end
            end
        end
    end
    if not assigned then
        asteroid_assignments.append((asteroid_id, 0, random_opportunity()))
    end
end
return asteroid_assignments
```

4.3. Algoritmo heurístico *best-fit*

Como ya se sabe el objetivo del problema es maximizar la masa mínima recolectada entre todas las estaciones de procesamiento y entre los tres materiales, pues esta heurística es la que proporciona con mayor diferencia los mejores resultados en la función aptitud, y su principal característica respecto a las anteriores es que se asigna los asteroides en función de la masa ya recolectada. El objetivo es intentar asignar la primera oportunidad disponible avanzando desde la estación que menos menos masa tenga en uno de los materiales hasta la que más, a diferencia del anterior que se asignaba según su ventana de apertura.

La función **best_fit** tiene como objetivo obtener el vector de decisión resultante, en el que los 24 primeros elementos de ese vector nos muestra de forma ordenada el tiempo de entrada y el de salida de cada una de las estaciones y luego contendrá de forma ordenada 340 elementos en el que cada uno será una 3-tupla que contendrá el asteroide, la estación escogida y su respectiva oportunidad. El Algoritmo 4 paso a paso la realiza lo siguiente:

- Se inicializa la lista **asteroid_assignments** para almacenar las asignaciones de asteroides. Esta lista se utilizará para construir el vector de decisión final.
- Se cargan los datos de los candidatos desde un archivo JSON usando la función **load_candidates**.
- Se inicializan diccionarios **rango_inicial** y **rango_final** para realizar un seguimiento de la hora de llegada más temprana y más tardía para cada estación. Inicialmente, se configuran con valores extremos. También se establece a 24 el numero de ventanas, correspondiente a la fecha de inicio y final de cada estación.
- Se elige aleatoriamente un valor para representar el tiempo de inicio de la ventana. Luego, se generan los demás valores asegurándose de que haya al menos una unidad de diferencia entre ellos. Finalmente, los valores se ordenan para obtener una secuencia con 24 valores.
- Hasta este punto del código es igual que la heurística anterior. A continuación se crea un diccionario (masa_recolectada_por_estacion) para almacenar la masa recolectada por cada estación, inicialmente establecida en 0 para cada elemento A, B y C.
- Se itera a través de los asteroides que se les iniciará la variable assigned como False, luego ordenamos las estaciones en función de la masa recolectada de menor a mayor y comenzamos a iterar sobre estas, dentro del bucle de estaciones, buscamos la cantidad de oportunidades que tiene cada asteroide e iteramos a través de estas oportunidades.
- Dentro de todos estas iteraciones se comprueba si el tiempo de actividad está dentro de las ventanas de tiempo de la estación. Si es así, se asigna el asteroide a la estación y se actualiza la masa recolectada por esa estación y se establece assigned en True y se rompe el bucle. La información de asignación se agrega al vector de decisión y se devuelve el vector decisión resultante.

La función de evaluación realiza un número especificado de iteraciones (iterations) para generar distintos vectores de decisión con la heurística comentada y evaluar su

aptitud utilizando la función de aptitud que se nos proporciona llamada udp.fitness(x). Registra y devuelve el mejor vector de decisión encontrado y su aptitud.

Como veremos posteriormente en la parte de experimentos y resultados comprobaremos como esta heurística es aquella obtiene individuos con mejor fitness entre las distintas heurísticas creadas.

Algoritmo 4: Función asignación best_fit

```
asteroid_assignments = create_initial_solution()
start_range, end_range, num_windows = define_ranges() window = []
first_value = random(start_range, end_range) window.append(first_value)
for i in 1 to num_windows − 1 do
    new_value = random(start_range, end_range) while
    mín(|new_value − w| ≥ 1 for all w ∈ window) do
        new_value = random(start_range, end_range)
    end
    window.append(new_value)
end
window.sort() window = shuffle_pairs(window) flattened_window = flatten_list(window)
masa_recolectada_por_estacion = {}
for asteroid_id in 1 to 340 do
    assigned = False stations = sort_by_min_mass(masa_recolectada_por_estacion)
    for station in stations do
        if assigned then
            break
        end
        opportunities = get_opportunities(asteroid_id, station) if opportunities then
            for opportunity in opportunities do
                arrival_time = get_arrival_time(opportunity)
                masses_collected = get_masses_collected(opportunity) if
                arrival_time ∈ station.window then
                    asteroid_assignments.append((asteroid_id, station, opportunity))
                    update_mass(masa_recolectada_por_estacion, station, masses_collected)
                    assigned = True break
                end
            end
        end
    end
    if not assigned then
        asteroid_assignments.append((asteroid_id, 0, random_opportunity()))
    end
end
return asteroid_assignments
```

4.4. Algoritmo memético

Los algoritmos meméticos, pertenecientes a la familia de los algoritmos evolutivos, se tratan de combinaciones de algoritmos evolutivos y procedimientos de mejora local de soluciones. En esencia, los algoritmos meméticos aprovechan la capacidad de exploración global de los algoritmos evolutivos y la capacidad de explotación local de los procedimientos de mejora de soluciones.

El algoritmo memético implementado en este trabajo aprovecha la capacidad de exploración y explotación de los algoritmos evolutivos y los procedimientos de mejora local para encontrar soluciones de alta calidad para el problema de optimización de la entrega de asteroides. Esta combinación de enfoques permite encontrar soluciones más robustas y eficientes que las obtenidas utilizando solo técnicas de búsqueda global o local.

El proceso comienza con la generación aleatoria de una población inicial de individuos. Cada individuo representa una posible configuración de parámetros. Luego, se evalúa el rendimiento de cada individuo utilizando una función de aptitud que calcula la media de múltiples ejecuciones del simulador con los parámetros del individuo. Un individuo se considera mejor que otro si su rendimiento promedio es inferior al del otro. Después de la evaluación, se seleccionan dos individuos aleatorios de la población actual para generar descendencia. Estos individuos se cruzan utilizando un operador de cruce con una probabilidad predefinida. El proceso de cruce se repite hasta que se obtiene una cantidad suficiente de descendientes. Una vez que se ha generado la población descendiente, el siguiente paso es la búsqueda local, donde cada individuo pasa por un proceso de mejora local para explorar y explotar aún más el espacio de búsqueda en busca de soluciones de alta calidad.

La búsqueda local se lleva a cabo iterativamente para cada individuo de la población descendiente utilizando un enfoque de hill-climbing. En cada iteración de la búsqueda local, se intenta mejorar el individuo actual modificando sus parámetros de manera incremental dentro de un vecindario definido. Este vecindario se compone de soluciones que se pueden obtener haciendo pequeñas modificaciones en la asignación de asteroides a estaciones y en la selección de oportunidades. Específicamente, se exploran configuraciones donde un asteroide puede ser reasignado a una estación diferente o se puede cambiar la oportunidad seleccionada para ese asteroide.

El objetivo de la búsqueda local es encontrar configuraciones de parámetros que optimicen aún más la función de aptitud, mejorando así la calidad de las soluciones encontradas por el algoritmo memético. Esto se hace explorando vecinos cercanos en el espacio de búsqueda y evaluando cómo afectan estos cambios al rendimiento del individuo. Si una nueva configuración mejora la función de aptitud, se actualiza el individuo actual con esta nueva configuración. Este proceso se repite hasta que no se encuentran mejoras significativas en un número determinado de intentos.

Una vez completada la búsqueda local para todos los individuos, se lleva a cabo un proceso de reemplazo donde se seleccionan los mejores individuos para formar la siguiente generación, eliminando los peores individuos de ambas poblaciones. Este enfoque de reemplazo se conoce como reemplazar el peor"(replace worst). La combinación de hill-climbing con la exploración de un vecindario bien definido permite refinar y ajustar los individuos generados mediante el cruce, lo que puede conducir a soluciones más óptimas y eficientes.

Este proceso iterativo de cruce, búsqueda local y selección se repite a lo largo de múltiples generaciones hasta que se cumple un criterio de terminación, como alcanzar

un número máximo de generaciones o no observar una mejora significativa en la calidad de las soluciones. De esta manera, el algoritmo memético busca encontrar soluciones óptimas o cercanas a óptimas para el problema dado. Se ha planteado un pseudocódigo descrito en el Algoritmo 5, en el que se puede ver paso a paso lo explicado anteriormente.

Algoritmo 5: Fases de un algoritmo memético

```
population = create_initial_population();
while not termination_condition() do
    evaluate(population);
    selected_individuals = selection(population);
    offspring = crossover(selected_individuals);
    evaluate(offspring);
    for individual in offspring do
        individual = local_search(individual);
    end
    population = select_next_generation(population, offspring);
end
best_individual = get_best_individual(population);
return best_individual;
```

Capítulo 5

Experimentos y resultados

En este apartado mostraremos las diferentes pruebas realizadas tanto con las tres heurísticas explicadas anteriormente, como con el algoritmo memético desarrollado. Las pruebas realizadas abarcan ejecuciones de 100 y 1000 iteraciones. En cada una de ellas se mostrará el número de pruebas, el mejor fitness obtenido, una tabla con las masas mínimas obtenidas, otra donde se muestran los puntos seleccionados entre todos los disponibles y un resumen que muestra algunos datos relevantes. Finalmente, se llevará a cabo un análisis de los resultados obtenidos, así como las conclusiones extraídas.

Los experimentos se han ejecutado en Google Colaboratory, aprovechando su capacidad para ejecutar código en la nube y el acceso a recursos computacionales avanzados. Se puede acceder al cuaderno con el código fuente de los experimentos a través del siguiente enlace de Google Colaboratory Notebook [11].

5.1. Algoritmo de Búsqueda Aleatoria (BA)

Se ha ejecutado la BA descrita en la sección 4.1 con 100 y 1000 iteraciones y en esta sección se presentan y discuten sus resultados. Primero, se mostrarán las mejores soluciones obtenidas en ambos casos y, por último, un boxplot comparativo de las dos pruebas realizadas para comparar sus resultados.

Para 100 iteraciones el mejor fitness mínimo obtenido fue de $fitness = -0{,}5649$. Podemos observar en la Figura 5.1 como se muestran en color azul las distintas asignaciones de los asteroides a la estación correspondiente a lo largo de los 80 días disponibles, mientras que en rojo se muestran todas las oportunidades disponibles por estación. Al mismo tiempo, en la Figura 5.2 se puede observar las cantidades de materiales que se han recolectado en cada estación, concluyendo que la BA no consigue un reparto equitativo entre todas las estaciones, pues las estaciones 8 y 11 acumulan mucho material respecto a otras como la 4, 5 o 6 donde apenas se consigue recolectar. Por último en la Tabla 5.1 se hace un resumen cuantitativo de la mejor solución obtenida, donde se indica la cantidad de cada material recolectado en cada estación, el espacio de tiempo mínimo entre estaciones, que en este caso es de 1.122 días, y en caso de que hubiera, los identificadores de asteroides, los tiempos de llegada y las asignaciones de asteroides no válidas (para una solución factible, estos tres valores deben ser iguales a cero). Finalmente, se muestra el fitness (-0.5649) asociado a dicha solución.

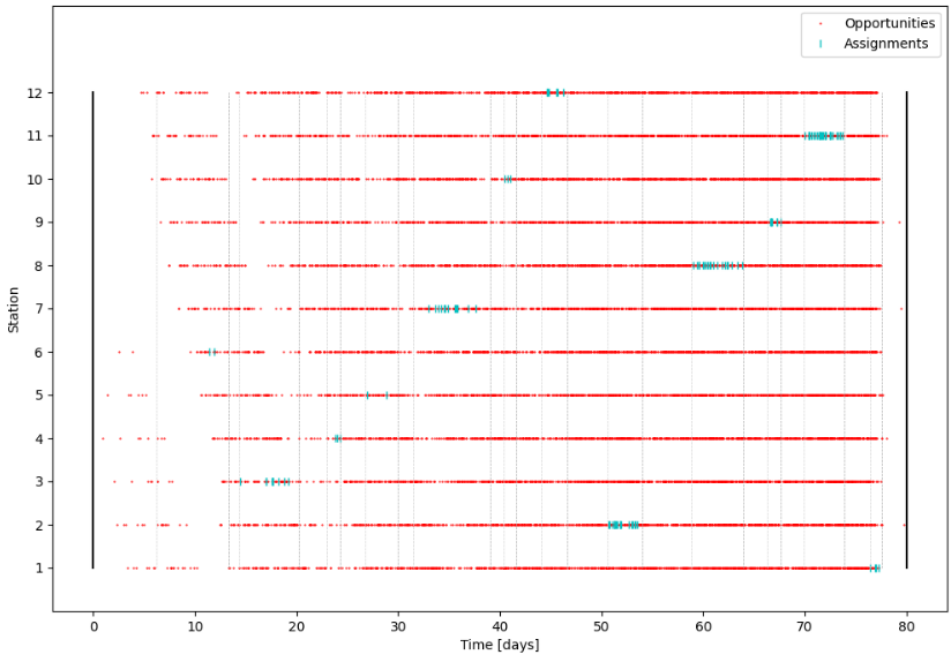


Figura 5.1: Cronograma de oportunidades y asignaciones realizadas en cada estación teniendo en cuenta la mejor solución obtenida

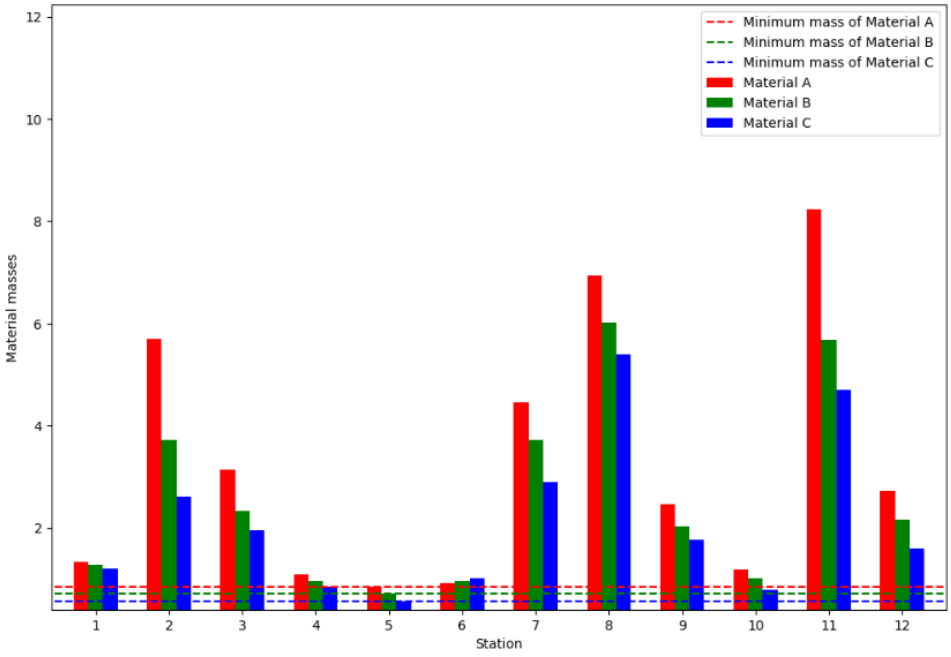


Figura 5.2: Cantidades obtenidas de cada material en cada estación teniendo en cuenta la mejor solución obtenida

Con pruebas de 1000 iteraciones el mejor fitness mínimo mejora significativamente respecto a la prueba de 100, consiguiendo un resultado de $fitness = -1{,}2263$. En la Figura 5.3 vemos en azul las asignaciones respecto a todas las oportunidades disponible en rojo y en la Figura 5.4, el reparto de materiales en cada estación, donde vemos que la estación 9 acumula gran parte de ellos. Por último, comprobamos que se trata de una solución factible en la Tabla 5.2 donde vemos el reparto total de todos los materiales en

Tabla 5.1: Análisis cuantitativo de la mejor solución obtenida con 100 iteraciones de la BA.

Características	**Material A**	**Material B**	**Material C**
Estación 1	1.328343	1.274389	1.196254
Estación 2	5.692302	3.724591	2.607414
Estación 3	3.145913	2.329938	1.957613
Estación 4	1.091324	0.958663	0.836543
Estación 5	0.842101	0.712337	0.564898
Estación 6	0.914900	0.963659	1.008336
Estación 7	4.463397	3.722995	2.891196
Estación 8	6.930291	6.022630	5.399001
Estación 9	2.467891	2.038908	1.773383
Estación 10	1.192427	1.014233	0.797679
Estación 11	8.241422	5.685335	4.698738
Estación 12	2.719867	2.167707	1.598983
IDs de asteroides no válidos	0 de 340		
Tiempos de llegada no válidos	0		
Brecha inter-estación mínima	1.122		
Asignaciones de asteroides no válidas	0		
Fitness	-0.5648		

cada estación, encontrando la masa mínima obtenida, que es lo que da lugar al fitness resultante, en la estación 2 en la recolecta del material C.

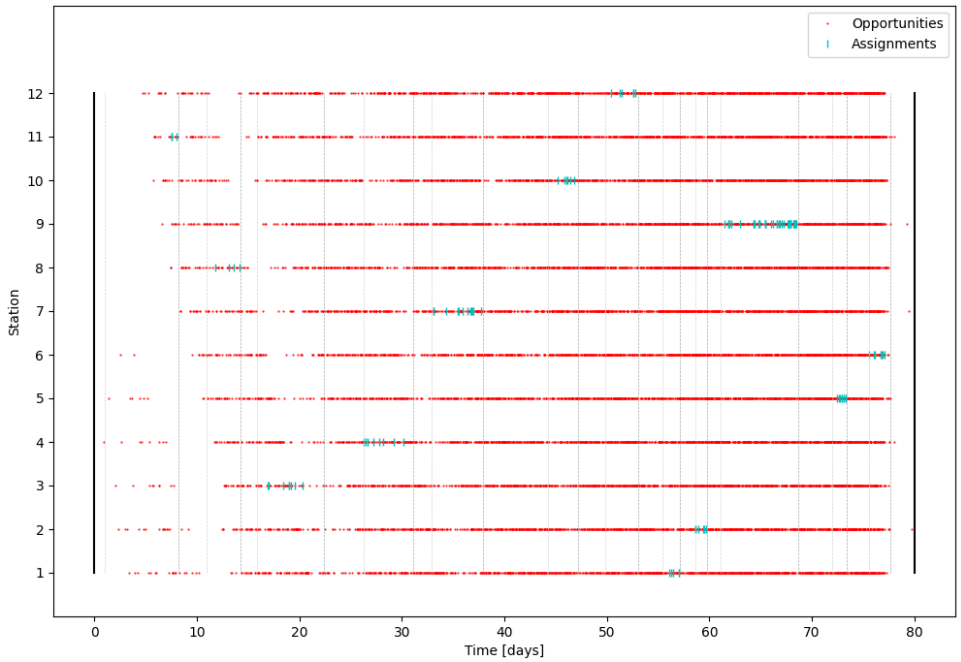


Figura 5.3: Cronograma de oportunidades y asignaciones realizadas en cada estación teniendo en cuenta la mejor solución obtenida

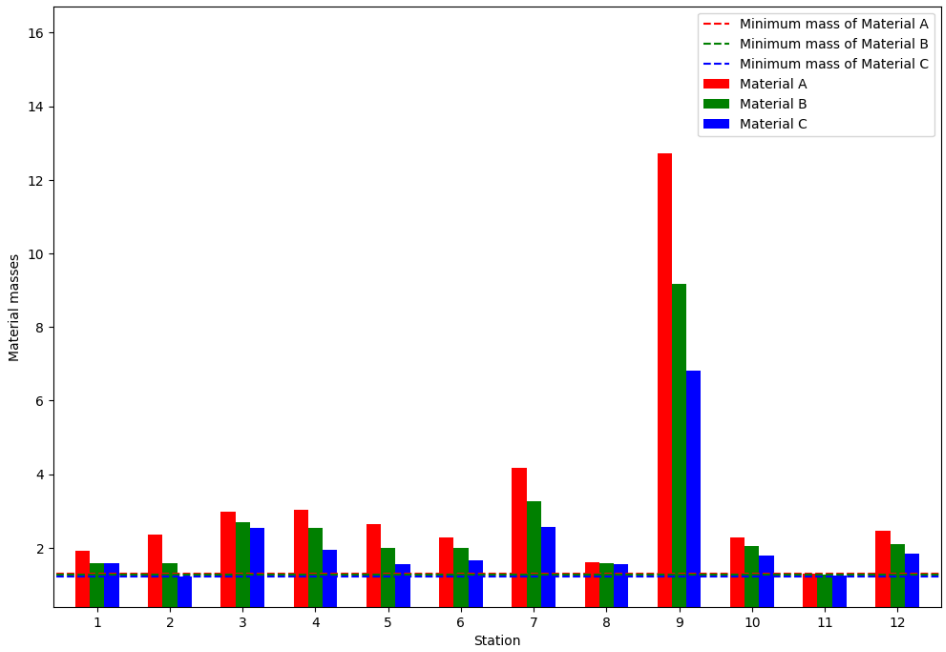


Figura 5.4: Cantidades obtenidas de cada material en cada estación teniendo en cuenta la mejor solución obtenida

Tabla 5.2: Análisis cuantitativo de la mejor solución obtenida con 1000 iteraciones de la BA

Característica	**Material A**	**Material B**	**Material C**
Estación 1	1.918712	1.589026	1.578072
Estación 2	2.351104	1.595654	1.226325
Estación 3	2.974428	2.695525	2.536973
Estación 4	3.040916	2.536696	1.951525
Estación 5	2.634661	2.010168	1.552792
Estación 6	2.285676	2.002017	1.653890
Estación 7	4.178710	3.279723	2.573054
Estación 8	1.605346	1.584994	1.546272
Estación 9	12.715818	9.160215	6.819995
Estación 10	2.286186	2.051636	1.781463
Estación 11	1.298257	1.279291	1.246016
Estación 12	2.470188	2.110618	1.839537
IDs de asteroides no válidos	0 de 340		
Tiempos de llegada no válidos	0		
Brecha inter-estación mínima	1.255		
Asignaciones de asteroides no válidas	0		
Fitness	-1.2263		

En la Figura 5.5 se muestra la distribución de los valores de fitness para 100 y 1000 iteraciones del algoritmo de BA. Para 100 iteraciones, el rango intercuartílico (IQR) se extiende desde aproximadamente -0.4362 hasta -0.3110, con un valor atípico en torno a -0.1021. Esto sugiere que la mayoría de las soluciones se encuentran dentro de este rango, con un mínimo absoluto de -0.5649. Por otro lado, en el caso de 1000 iteraciones, el IQR se amplía significativamente, desde aproximadamente -0.7355 hasta -0.5332, con un valor atípico en torno a -0.3705. Además, el mínimo absoluto disminuye considerablemente a -1.2263. Podemos deducir que, si bien la distribución de los valores de fitness es más amplia en el caso de 1000 iteraciones, también hay una mayor probabilidad de encontrar soluciones de mejor calidad, como se evidencia por el mínimo absoluto más bajo.

Por otro lado en la Tabla 5.3 se muestran los valores de fitness promedio, mediana, mínimo, máximo, primer cuartil y tercer cuartil para 100 y 1000 iteraciones del algoritmo de BA. Se puede observar que para 1000 iteraciones, la media, mediana y los cuartiles son más bajos en comparación con 100 iteraciones, lo que indica una tendencia hacia soluciones de mejor calidad. Además, el valor mínimo absoluto es más bajo para 1000 iteraciones, lo que confirma la capacidad del algoritmo para encontrar mejores soluciones con un mayor número de iteraciones.

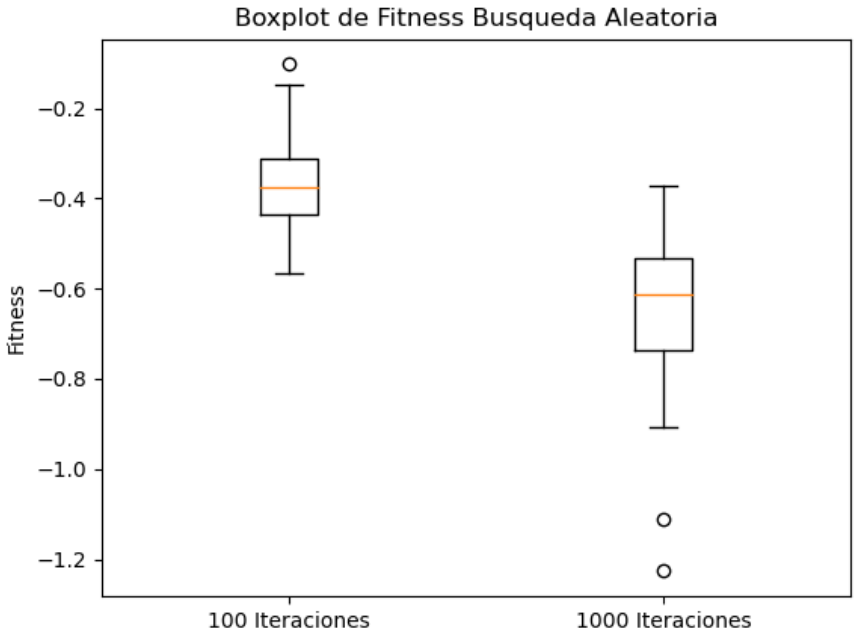


Figura 5.5: Boxplot comparativo del fitness de las soluciones obtenidas en 30 repeticiones de la BA ejecutada con 100 y 1000 iteraciones

Tabla 5.3: Valores de fitness para los diferentes criterios de parada (iteraciones) de la BA

Métrica	**100 Iteraciones**	**1000 Iteraciones**
Media	-0.35885	-0.66344
Mediana	-0.37522	-0.61309
Mínimo	-0.56490	-1.22633
Máximo	-0.10210	-0.37051
Primer Cuartil	-0.43624	-0.73547
Tercer Cuartil	-0.31097	-0.53318

5.2. Algoritmo heurístico *First-Window* (FW)

Se ha ejecutado la heurística FW descrita en la sección 4.2 con 100 y 1000 iteraciones y en esta sección se presentan y discuten sus resultados. Primero se mostrarán las mejores soluciones obtenidas en ambos casos y por último un boxplot comparativo de las dos pruebas realizadas para comparar sus resultados.

Para esta heurística el mejor fitness mínimo obtenido fue $fitness = -3{,}1411$ en 100 iteraciones. Podemos observar en la Figura 5.6 cómo se distribuyen las asignaciones de los asteroides a las estaciones a lo largo de los 80 días disponibles. En azul se muestran las asignaciones realizadas, mientras que en rojo se representan todas las oportunidades disponibles por estación. Además, en la Figura 5.7 se puede apreciar la cantidad de materiales recolectados en cada estación donde se logra un mayor reparto de los recursos entre las estaciones, en comparación con la BA, aunque sigue produciéndose acumulación desproporcionada de material en estaciones individuales como se puede ver en la estación 2. Aún así, esto sugiere una mayor eficiencia en la asignación de recursos y una mejor optimización de la planificación de las entregas respecto a la BA.

En la Tabla 5.4, se presenta un resumen cuantitativo de la mejor solución obtenida con la heurística FW. Se proporciona información detallada sobre la cantidad de cada material recolectado en cada estación, el espacio de tiempo mínimo entre estaciones, y la validación de las restricciones de asteroides, tiempos de llegada y asignaciones de asteroides. Además, se muestra el fitness asociado a la solución, el cual refleja un mejor nivel de eficiencia y optimización en la asignación de recursos.

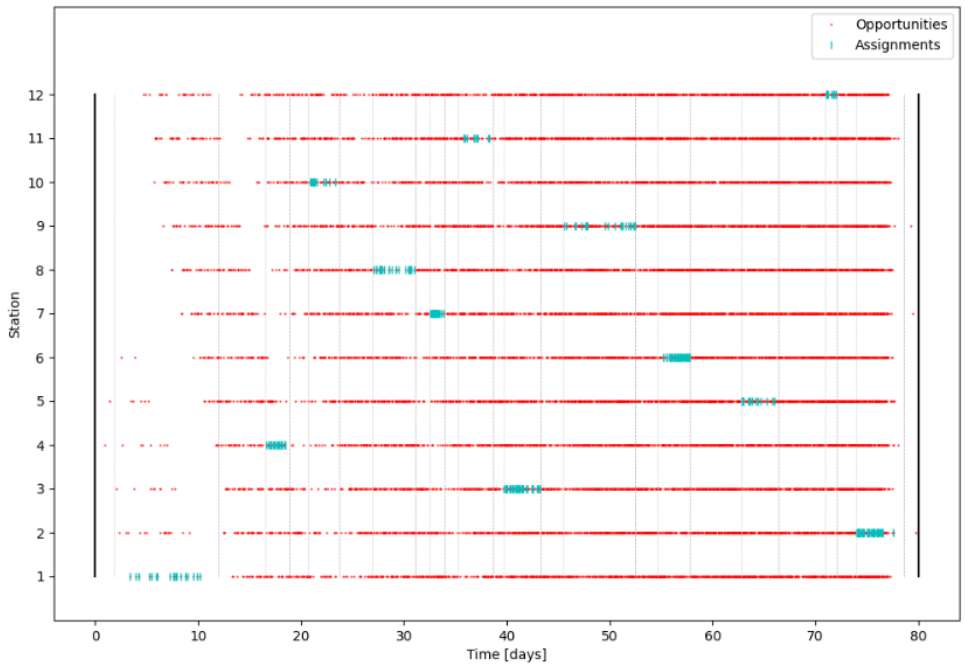


Figura 5.6: Cronograma de oportunidades y asignaciones realizadas en cada estación teniendo en cuenta la mejor solución obtenida

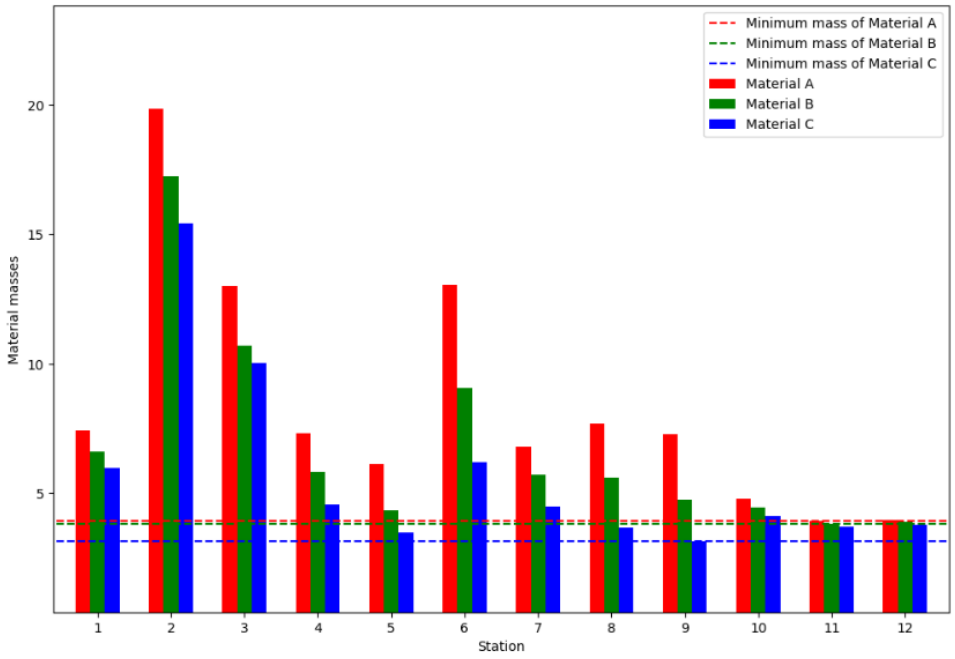


Figura 5.7: Cantidades obtenidas de cada material en cada estación teniendo en cuenta la mejor solución obtenida

Tabla 5.4: Análisis cuantitativo de la mejor solución obtenida con 100 iteraciones de la FW.

Características	**Material A**	**Material B**	**Material C**
Estación 1	7.422615	6.600207	5.962767
Estación 2	19.842022	17.261744	15.415266
Estación 3	12.987315	10.696440	10.044975
Estación 4	7.300875	5.828414	4.568531
Estación 5	6.104398	4.337260	3.494545
Estación 6	13.036783	9.074953	6.194965
Estación 7	6.793674	5.708023	4.486759
Estación 8	7.683923	5.590263	3.681178
Estación 9	7.256894	4.736047	3.141100
Estación 10	4.772611	4.445456	4.108051
Estación 11	3.929253	3.817134	3.711272
Estación 12	3.953081	3.886096	3.772348
Asteroid IDs inválidos	0 de 340		
Tiempos de llegada inválidos	0		
Brecha inter-estación mínima	1.035		
Asignaciones inválidas	0		
Fitness total	-3.1410		

Con pruebas de 1000 iteraciones, la heurística FW logra mejorar ligeramente el fitness mínimo en comparación con la prueba de 100 iteraciones, obteniendo un resultado de $fitness = -3{,}2121$. En la Figura 5.8 se muestra en azul las asignaciones realizadas en relación con todas las oportunidades disponibles en rojo, lo que proporciona una visualización clara de cómo se distribuyen los recursos en las diferentes estaciones a lo largo del tiempo. En la Figura 5.9, se presenta el reparto de materiales en cada estación, destacando la acumulación de materiales en la estación 9.

Al analizar la Tabla 5.5, que muestra el reparto total de todos los materiales en cada estación, se confirma que se trata de una solución factible. Se observa que la masa mínima obtenida, que determina el fitness resultante, se encuentra en la estación 10 en la recolección del material C. Esto sugiere que la heurística FW logra una distribución más equitativa de recursos entre las estaciones en comparación con la BA, lo que resulta en una mejora significativa en el fitness mínimo alcanzado.

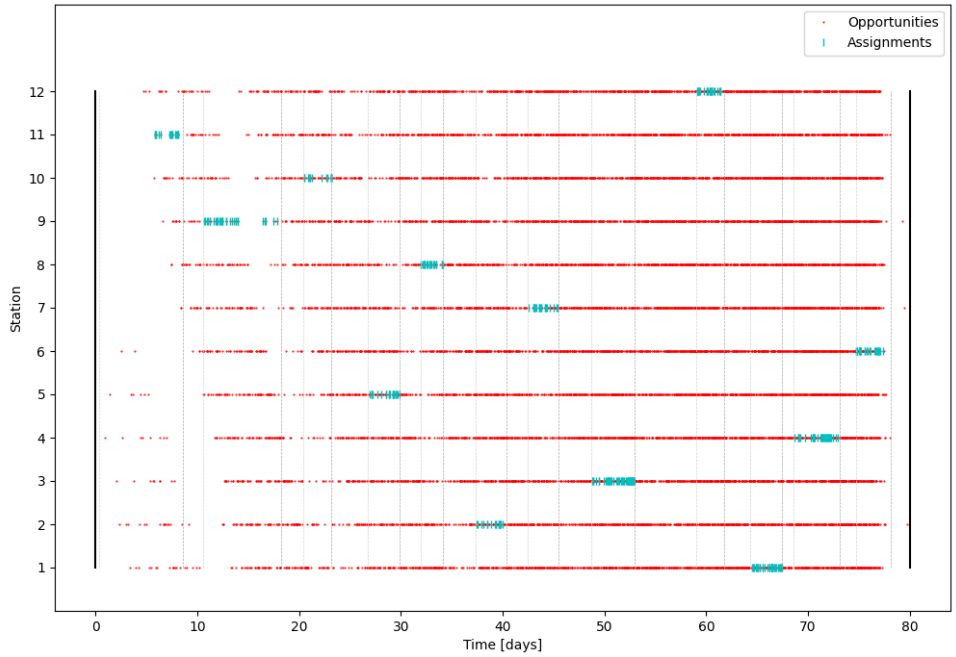


Figura 5.8: Cronograma de oportunidades y asignaciones realizadas en cada estación teniendo en cuenta la mejor solución obtenida

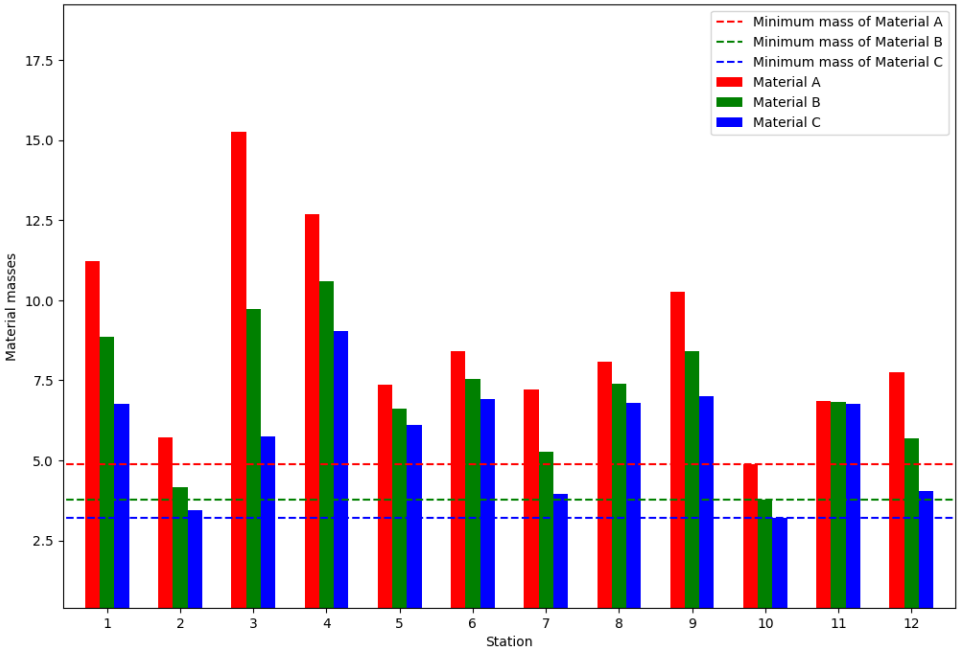


Figura 5.9: Cantidades obtenidas de cada material en cada estación teniendo en cuenta la mejor solución obtenida

Tabla 5.5: Análisis cuantitativo de la mejor solución obtenida con 1000 iteraciones de la FW.

Características	**Material A**	**Material B**	**Material C**
Estación 1	11.216160	8.853864	6.759517
Estación 2	5.730217	4.153108	3.430474
Estación 3	15.242882	9.735989	5.736066
Estación 4	12.693324	10.593594	9.022802
Estación 5	7.351074	6.598454	6.102926
Estación 6	8.407373	7.543877	6.922760
Estación 7	7.202169	5.275223	3.958934
Estación 8	8.080552	7.388930	6.802435
Estación 9	10.248211	8.394232	6.987536
Estación 10	4.889227	3.779686	3.212115
Estación 11	6.858551	6.818578	6.773471
Estación 12	7.753186	5.689729	4.027176
Asteroid IDs inválidos	0 de 340		
Tiempos de llegada inválidos	0		
Brecha inter-estación mínima	1.136		
Asignaciones inválidas	0		
Fitness total	-3.2121		

La Figura 5.10 muestra la distribución de los valores de fitness para 100 y 1000 iteraciones del algoritmo heurístico firstwindow. Para 100 iteraciones, el IQR se extiende desde aproximadamente -1.9106 hasta -1.4016, con un mínimo absoluto de -3.1411, tratándose de un valor atípico respecto al resto. Por otro lado, en el caso de 1000 iteraciones, el IQR se amplía significativamente, desde aproximadamente -2.5774 hasta -2.3398, con valores atípicos por ambos lados del boxplot. Además, el mínimo absoluto disminuye considerablemente a -3.2121. Esto indica que, si bien la distribución de los valores de fitness es más amplia en el caso de 1000 iteraciones, también hay una mayor probabilidad de encontrar soluciones de alta calidad, como se evidencia por el mínimo absoluto más bajo.

La Tabla 5.6 muestra los valores de fitness promedio, mediana, mínimo, máximo, primer cuartil y tercer cuartil para 100 y 1000 iteraciones del algoritmo heurístico firstwindow. Se puede observar que para 1000 iteraciones, la media, mediana y los cuartiles son más bajos en comparación con 100 iteraciones, lo que indica una tendencia hacia soluciones de mejor calidad según se amplia el número de iteraciones. Además, el valor mínimo absoluto es más bajo para 1000 iteraciones, lo que confirma la capacidad del algoritmo para encontrar soluciones mejores con un mayor número de iteraciones.

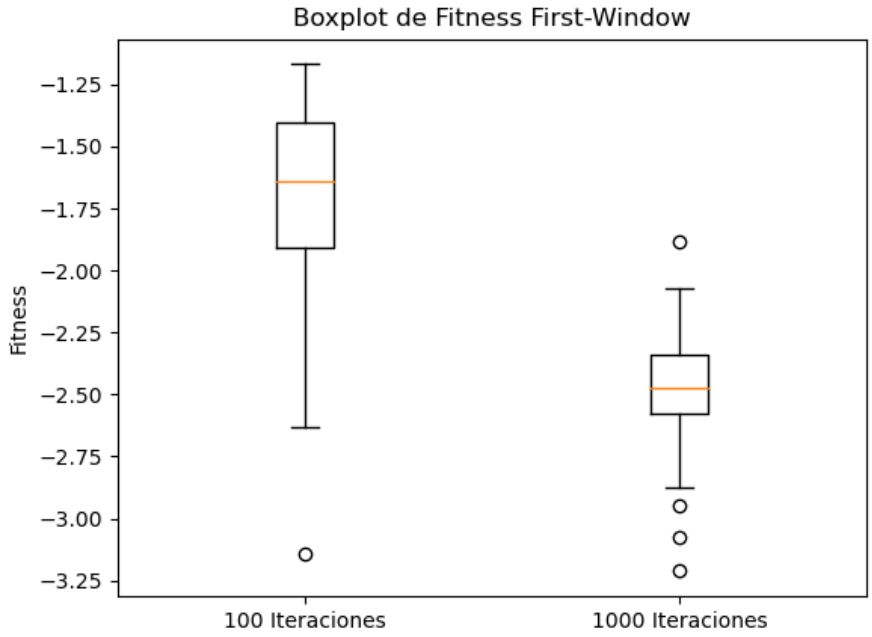


Figura 5.10: Boxplot comparativo del fitness de las soluciones obtenidas en 30 repeticiones de la heurística FW con 100 y 1000 iteraciones

Tabla 5.6: Valores de fitness para los diferentes criterios de parada (iteraciones) de la heurística FW

Métrica	**100 iteraciones**	**1000 iteraciones**
Media	-1.7167	-2.4957
Mediana	-1.6392	-2.4735
Mínimo	-3.1411	-3.2121
Máximo	-1.1681	-1.8861
Primer Cuartil	-1.9106	-2.5774
Tercer Cuartil	-1.4016	-2.3398

5.3. Algoritmo heurístico *Best-Fit* (BF)

Se ha ejecutado la heurística BF descrita en la sección 4.3 con 100 y 1000 iteraciones y en esta sección se presentan y discuten sus resultados. Primeo se mostrarán las mejores soluciones obtenidas en ambos casos y por último un boxplot comparativo de las dos pruebas realizadas para comparar sus resultados.

Para la heurística BF, el mejor fitness mínimo obtenido fue $fitness = -5{,}0302$ en 100 iteraciones. Este valor indica una alta eficiencia en la asignación de recursos, ya que representa una mayor cantidad de masa mínima recolectada en cada estación para cada uno de los materiales. Al analizar visualmente en la Figura 5.11 la distribución de las asignaciones de asteroides a lo largo de los 80 días disponibles, se observa una distribución más equitativa de los recursos en las estaciones. Además, al examinar la cantidad de materiales recolectados en cada estación, representada en la Figura 5.12, no se observa una acumulación desproporcionada de material en una estación específica. Esto sugiere una optimización en la planificación de las entregas y una alta eficiencia en la asignación de recursos lo que confirma una mayor masa mínima recolectada en comparación con otras heurísticas. Esta distribución más equitativa de los recursos indica una mayor optimización en la asignación de recursos y una mayor eficiencia en la recolección de materiales. En resumen, en la Tabla 5.7 vemos un resumen completo del fitness mínimo obtenido con la heurística BF reflejando una alta eficiencia y optimización en la asignación de recursos, lo que sugiere una alta calidad en la solución encontrada.

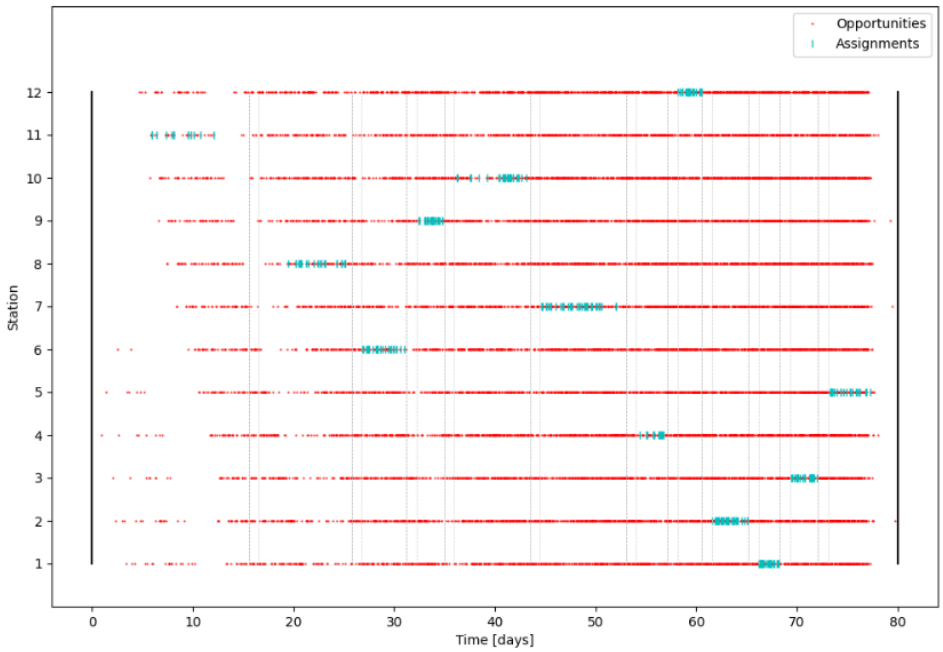


Figura 5.11: Cronograma de oportunidades y asignaciones realizadas en cada estación teniendo en cuenta la mejor solución obtenida

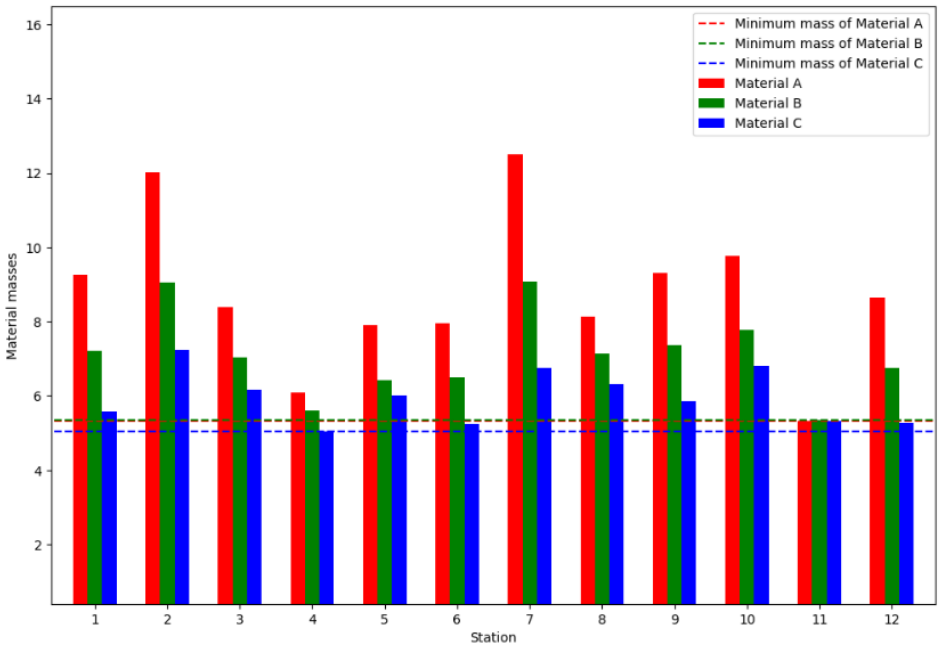


Figura 5.12: Cantidades obtenidas de cada material en cada estación teniendo en cuenta la mejor solución obtenida

Tabla 5.7: Análisis cuantitativo de la mejor solución obtenida con 100 iteraciones de la BF

Características	**Material A**	**Material B**	**Material C**
Estación 1	9.252890	7.219661	5.575423
Estación 2	12.006173	9.041555	7.241113
Estación 3	8.378714	7.039138	6.176136
Estación 4	6.082121	5.594009	5.030248
Estación 5	7.906640	6.424165	5.999012
Estación 6	7.963800	6.503120	5.244519
Estación 7	12.489784	9.087005	6.743601
Estación 8	8.138409	7.139156	6.310543
Estación 9	9.306509	7.366068	5.852296
Estación 10	9.753881	7.777988	6.797779
Estación 11	5.331439	5.358862	5.310008
Estación 12	8.653279	6.759249	5.279202
Asteroid IDs inválidos	0 de 340		
Tiempos de llegada inválidos	0		
Brecha inter-estación mínima	1.0		
Asignaciones inválidas	0		
Fitness total	-5.0302		

Con pruebas de 1000 iteraciones, la heurística BF logra mejorar ligeramente el fitness mínimo en comparación con la prueba de 100 iteraciones, obteniendo un resultado de $fitness = -5{,}4256$. En la Figura 5.13 se muestra en azul las asignaciones realizadas en relación con todas las oportunidades disponibles en rojo, lo que proporciona una visualización clara de cómo se distribuyen los recursos en las diferentes estaciones. En la Figura 5.14, se presenta el reparto de materiales en cada estación, destacando el reparto equitativo del material C, que es el que marca el fitness máximo, entre todas las estaciones. Al analizar la Tabla 5.8, que muestra el reparto total de todos los materiales en cada estación, se confirma que se trata de una solución factible. Además, se observa que la masa mínima obtenida, que determina el fitness resultante, se encuentra en la estación 8 en la recolección del material C. Esto sugiere que la heurística BF logra una distribución más equitativa de recursos entre las estaciones en comparación con las otras técnicas empleadas, lo que resulta en una mejora significativa en el fitness mínimo alcanzado.

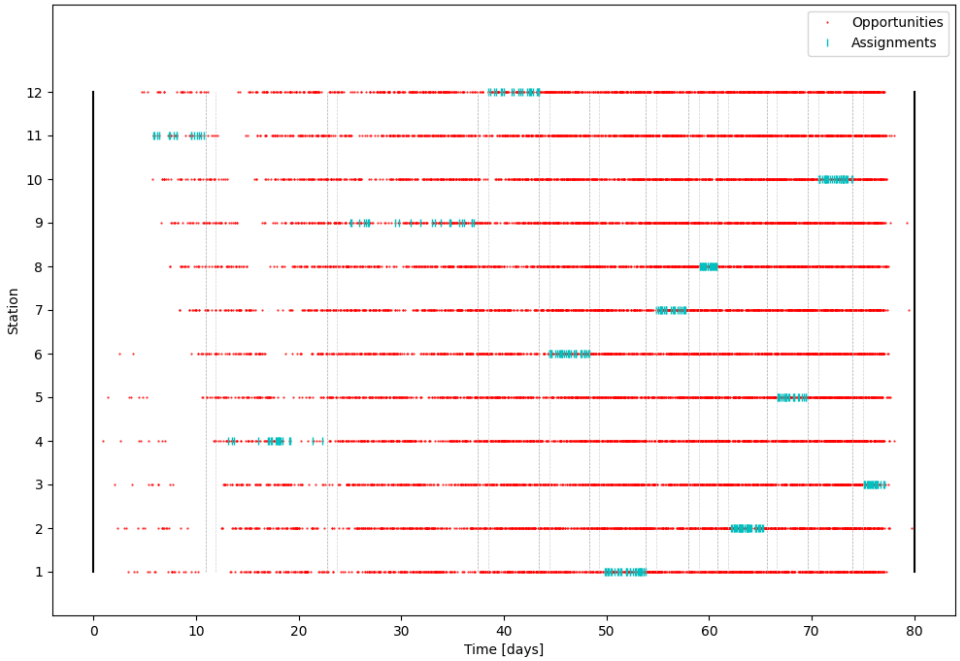


Figura 5.13: Cronograma de oportunidades y asignaciones realizadas en cada estación teniendo en cuenta la mejor solución obtenida

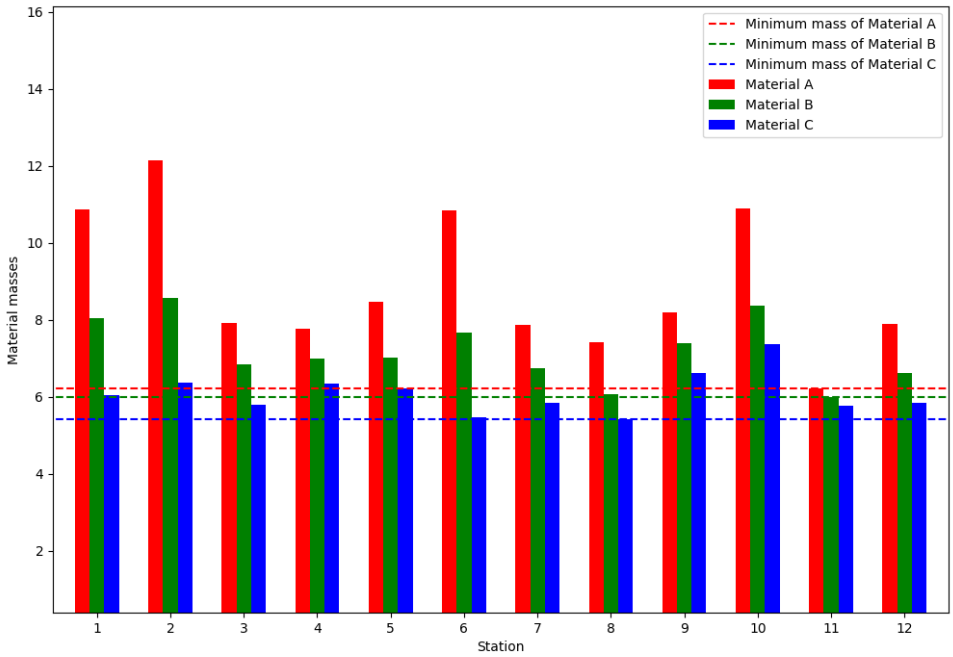


Figura 5.14: Cantidades obtenidas de cada material en cada estación teniendo en cuenta la mejor solución obtenida

Tabla 5.8: Análisis cuantitativo de la mejor solución obtenida con 1000 iteraciones de la BF

Características	**Material A**	**Material B**	**Material C**
Estación 1	10.876402	8.040901	6.030458
Estación 2	12.147476	8.571295	6.356070
Estación 3	7.924935	6.835299	5.804109
Estación 4	7.754937	6.997028	6.343551
Estación 5	8.455366	7.008623	6.209657
Estación 6	10.852691	7.670581	5.476881
Estación 7	7.862020	6.738608	5.851915
Estación 8	7.410038	6.068587	5.425628
Estación 9	8.197607	7.390873	6.622204
Estación 10	10.885350	8.364541	7.372621
Estación 11	6.224037	6.001586	5.763084
Estación 12	7.896580	6.610575	5.833964
Asteroid IDs inválidos	0 de 340		
Tiempos de llegada inválidos	0		
Brecha inter-estación mínima	1.0		
Asignaciones inválidas	0		
Fitness total	-5.4256		

La Figura 5.15 muestra la distribución de los valores de fitness para 100 y 1000 iteraciones del algoritmo heurístico BF. Para 100 iteraciones, el IQR se extiende desde aproximadamente -4.6271 hasta -4.2265, con un mínimo absoluto de -5.0302. En el caso de 1000 iteraciones, el IQR se amplía significativamente, desde aproximadamente -5.1935 hasta -4.9041. Además, el mínimo absoluto disminuye considerablemente a -5.4256. Esto indica que, al igual que en el caso de FW, la distribución de los valores de fitness es más amplia en el caso de 1000 iteraciones, lo que sugiere una mayor probabilidad de encontrar soluciones de alta calidad. En la Tabla 5.9 muestra los valores de fitness promedio, mediana, mínimo, máximo, primer cuartil y tercer cuartil para 100 y 1000 iteraciones del algoritmo heurístico best fit. Se puede observar que para 1000 iteraciones, la media, mediana y los cuartiles son más bajos en comparación con 100 iteraciones, lo que indica una tendencia hacia soluciones de mejor calidad a medida que se amplía el número de iteraciones. Además, el valor mínimo absoluto es más bajo para 1000 iteraciones, lo que confirma la capacidad del algoritmo para encontrar mejores soluciones con un mayor número de iteraciones.

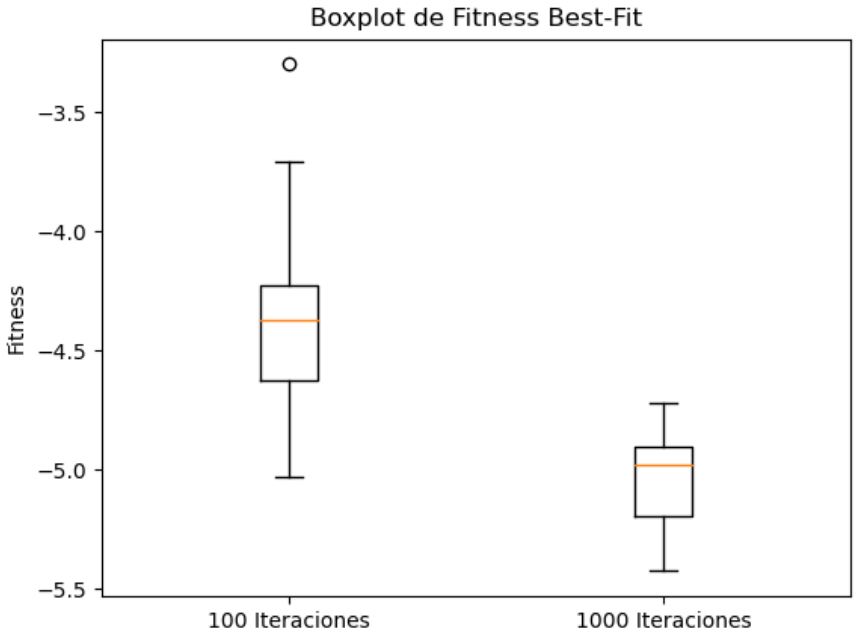


Figura 5.15: Boxplot comparativo del fitness de las soluciones obtenidas en 30 repeticiones de la heurística BF con 100 y 1000 iteraciones

Tabla 5.9: Valores de fitness para los diferentes criterios de parada (iteraciones) de la heurística BF

Métrica	**100 iteraciones**	**1000 iteraciones**
Media	-4.3825	-5.0335
Mediana	-4.3756	-4.9809
Mínimo	-5.0302	-5.4256
Máximo	-3.2990	-4.7224
Primer Cuartil	-4.6271	-5.1935
Tercer Cuartil	-4.2265	-4.9041

5.4. Análisis comparativo de la Búsqueda Aleatoria y las heurísticas implementadas

La evolución de los resultados desde la BA a la heurística FW y, luego, a la heurística BF, sugiere un progreso significativo en la calidad de las soluciones obtenidas. Este avance se puede atribuir a varias características específicas de cada heurística.

La BA se basa en un enfoque donde inicialmente genera soluciones de manera aleatoria, lo que resulta en una distribución de fitness con una media relativamente cercana a cero y una dispersión moderada. Sin embargo, debido a la naturaleza aleatoria del proceso, las soluciones óptimas son poco probables y el fitness mínimo obtenido es limitado.

La heurística FW mejora el enfoque aleatorio al introducir un criterio de selección más inteligente. En lugar de asignar asteroides aleatoriamente a estaciones, esta heurística selecciona la primera estación disponible que cumple con los requisitos de capacidad y tiempo. Esto conduce a una distribución de fitness con una media y mediana más bajas, lo que indica una tendencia hacia soluciones de mejor calidad. Además, el mínimo absoluto obtenido es considerablemente más bajo que el de la BA, lo que sugiere una mejora significativa en la eficiencia de la asignación de recursos.

Finalmente, la heurística best fit representa una mejora adicional al introducir un criterio de selección aún más riguroso. En lugar de simplemente seleccionar la primera estación disponible, esta heurística busca la estación que maximiza la masa mínima recolectada en la etapa actual. Esto permite una asignación más eficiente de recursos, lo que se refleja en un fitness mínimo aún más bajo. Además, la distribución de fitness es más concentrada alrededor de valores más bajos, lo que indica una mayor consistencia en la calidad de las soluciones obtenidas.

En resumen, mientras que la BA muestra resultados menos consistentes y menos óptimos, tanto FW como BF muestran una mejora significativa en la calidad de las soluciones. Best Fit, en particular, proporciona los mejores resultados en términos de fitness mínimo, indicando una mayor eficiencia en la asignación de recursos y la optimización de las soluciones. La mejora de un algoritmo a otro se debe principalmente a la introducción de criterios de selección más inteligentes y rigurosos, lo que permite una asignación más eficiente de recursos y una mejor optimización en la planificación de las entregas.

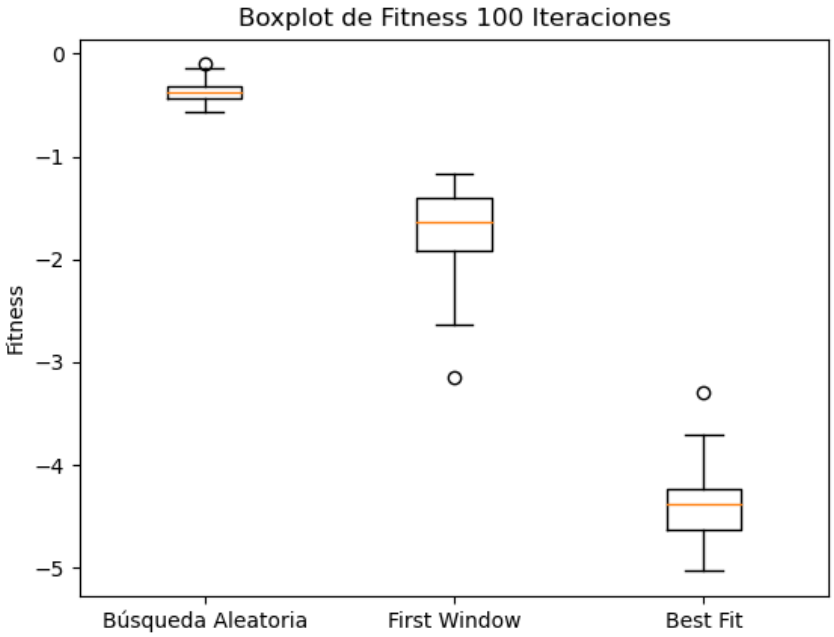


Figura 5.16: Boxplot comparativo del fitness de las soluciones obtenidas en 30 repeticiones de 100 iteraciones con cada uno de los algoritmos implementados

Figura 5.17: Boxplot comparativo del fitness de las soluciones obtenidas en 30 repeticiones de 1000 iteraciones con cada uno de los algoritmos implementados

5.5. Algoritmo Memético (AM)

Se ha ejecutado el AM descrito en la sección 4.4 durante 100 generaciones usando población de tamaño 100 y 200, con probabilidad de cruce de 1 y de 0.8. En la Tabla 5.10 vemos las distintas configuraciones hechas y el nombre que recibirán de aquí en adelante.

Tabla 5.10: Configuraciones del AM

Nombre	**Tamaño de población**	**Probabilidad de cruce**
AM_100_1	100	1
AM_100_0.8	100	0.8
AM_200_1	200	1
AM_200_0.8	200	0.8

Primero se mostrarán las mejores soluciones obtenidas en cada uno de los casos probados y por último un boxplot comparativo de las diferentes pruebas realizadas para comparar sus resultados y realizar un breve análisis de los mismos.

En las ejecuciones de la configuración **AM_100_1**, se logró obtener una solución con $fitness = -7{,}2498$. Este resultado representa la eficacia de la búsqueda en asignar los recursos de manera eficiente en las diferentes estaciones a lo largo del tiempo. La Figura 5.18 muestra la distribución de asignaciones de asteroides en cada estación a lo largo de los 80 días disponibles. En color azul se representan las asignaciones realizadas, mientras que en rojo se muestran todas las oportunidades disponibles por estación. Se observa una distribución equilibrada de asteroides en la mayoría de las estaciones, lo que indica una planificación efectiva de las entregas para maximizar la recolección de materiales. Además, en la Figura 5.19 se muestra la cantidad de cada material recolectado en cada estación. Se puede observar que se logra un reparto equitativo de los recursos entre las estaciones, lo que indica una asignación eficiente de los recursos para maximizar la masa recolectada. La Tabla 5.11 proporciona un resumen cuantitativo detallado de la mejor solución obtenida. Se incluyen las cantidades de cada material recolectado en cada estación, la validez de las restricciones de asteroides, tiempos de llegada y asignaciones, así como el fitness asociado a la solución. Este análisis cuantitativo respalda la eficiencia y optimización lograda por el AM en la asignación de recursos.

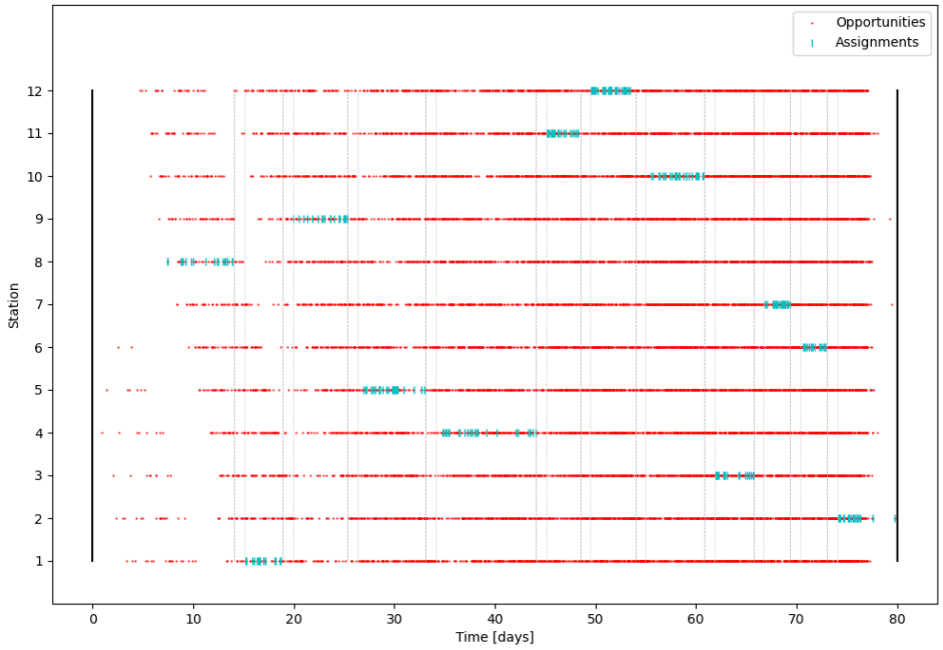


Figura 5.18: Cronograma de oportunidades y asignaciones realizadas en cada estación teniendo en cuenta la mejor solución obtenida con la AM_100_1

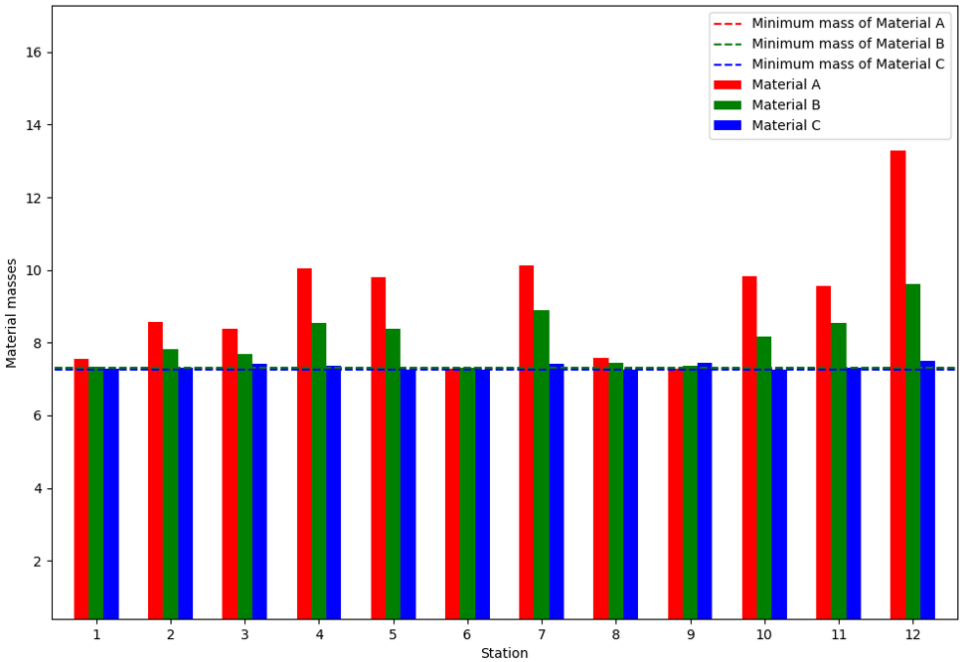


Figura 5.19: Cantidades obtenidas de cada material en cada estación teniendo en cuenta la mejor solución obtenida con la AM_100_1

Tabla 5.11: Análisis cuantitativo de la mejor solución obtenida con AM_100_1

Características	**Material A**	**Material B**	**Material C**
Estación 1	7.548486	7.348243	7.287492
Estación 2	8.572250	7.822083	7.312037
Estación 3	8.383270	7.693236	7.421336
Estación 4	10.052753	8.537780	7.365785
Estación 5	9.802404	8.389439	7.262418
Estación 6	7.283437	7.312996	7.249843
Estación 7	10.119471	8.882730	7.409130
Estación 8	7.568230	7.444552	7.252830
Estación 9	7.285222	7.372781	7.438369
Estación 10	9.824941	8.169204	7.267901
Estación 11	9.570982	8.548967	7.324753
Estación 12	13.278230	9.603387	7.502637
IDs de asteroides no válidos	0 de 340		
Tiempos de llegada no válidos	0		
Brecha inter-estación mínima	1.0		
Asignaciones de asteroides no válidas	0		
Fitness	-7.2498		

En las ejecuciones de la configuración **AM_100_0.8** se logró alcanzar una solución con $fitness = -7{,}4927$. Este resultado demuestra la capacidad del algoritmo para distribuir eficazmente los recursos en las diversas estaciones a lo largo del tiempo. En la Figura 5.20 se muestra la disposición de las asignaciones de asteroides en cada estación durante los 80 días disponibles. Las asignaciones realizadas se muestran en azul, mientras que en rojo se muestran todas las oportunidades disponibles por estación. Además, en la Figura 5.21 se presenta la cantidad de cada material recolectado en cada estación. Se observa una distribución equitativa de los recursos entre las estaciones, lo que indica una asignación eficiente de los recursos para maximizar la masa recolectada. La Tabla 5.12 proporciona un análisis detallado de la mejor solución obtenida. Se incluyen las cantidades de cada material recolectado en cada estación, así como la validez de las restricciones de asteroides, tiempos de llegada y asignaciones. Este análisis cuantitativo resalta la eficacia del AM_200_0.8 en la asignación de recursos para maximizar la recolección de materiales.

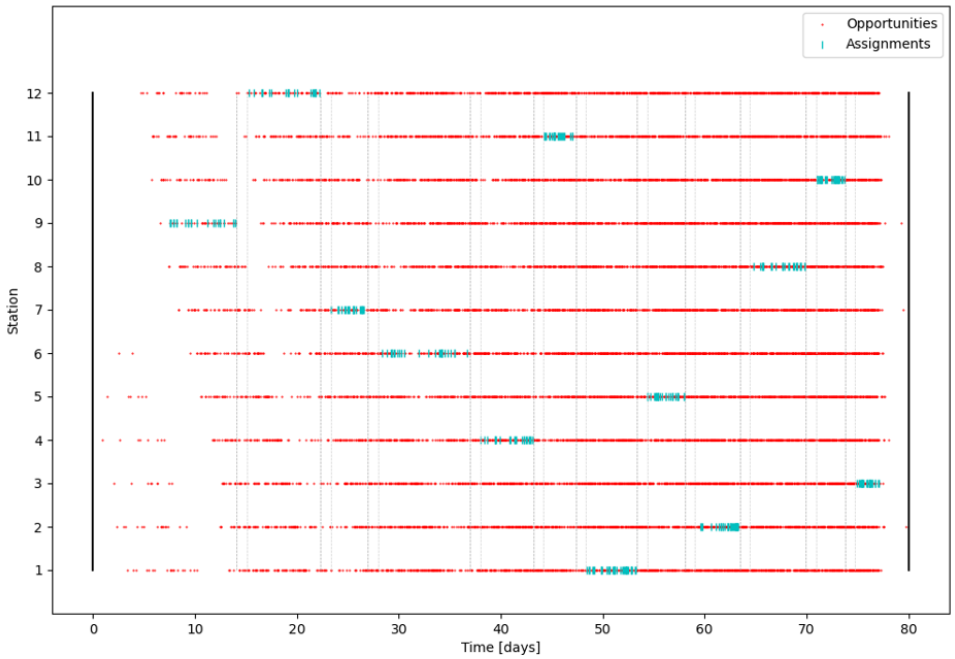


Figura 5.20: Cronograma de oportunidades y asignaciones realizadas en cada estación teniendo en cuenta la mejor solución obtenida con AM_100_0.8

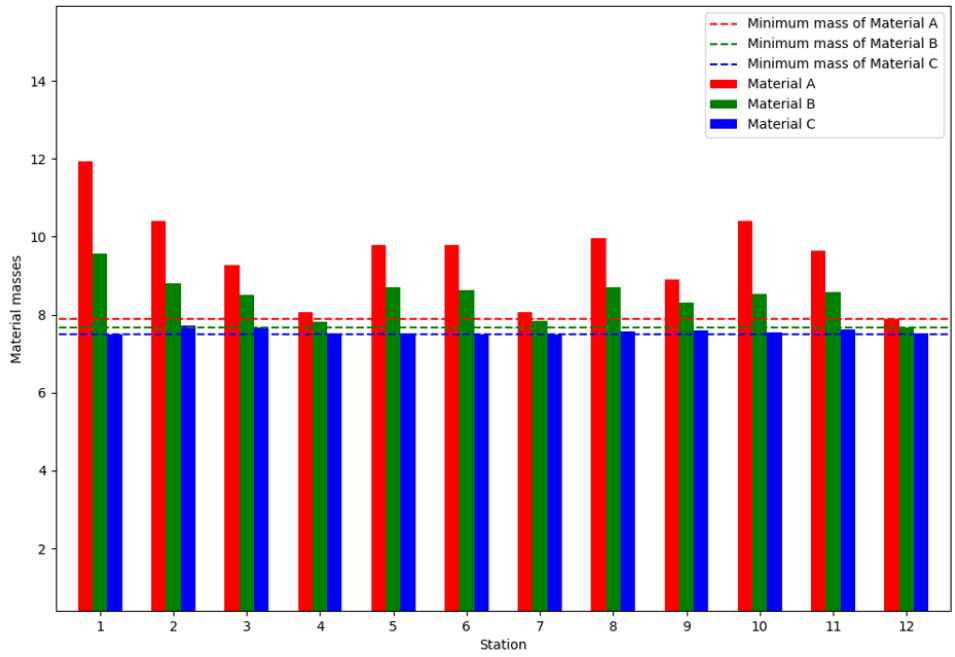


Figura 5.21: Cantidades obtenidas de cada material en cada estación teniendo en cuenta la mejor solución obtenida con AM_100_0.8

Tabla 5.12: Análisis cuantitativo de la mejor solución obtenida con AM_100_0.8

Características	**Material A**	**Material B**	**Material C**
Estación 1	11.923098	9.558737	7.501507
Estación 2	10.406732	8.808976	7.715578
Estación 3	9.275815	8.500732	7.656438
Estación 4	8.073243	7.810196	7.533123
Estación 5	9.792040	8.692202	7.517901
Estación 6	9.791092	8.617523	7.508031
Estación 7	8.055183	7.829515	7.492683
Estación 8	9.957927	8.711815	7.566883
Estación 9	8.889446	8.317076	7.586979
Estación 10	10.411874	8.531770	7.533289
Estación 11	9.639239	8.576111	7.626219
Estación 12	7.883390	7.679544	7.521817
IDs de asteroides no válidos	0 de 340		
Tiempos de llegada no válidos	0		
Brecha inter-estación mínima	1.0		
Asignaciones de asteroides no válidas	0		
Fitness	-7.4926		

Para la ejecución **AM_200_1**, se logró obtener una solución con $fitness = -7{,}5020$. En la Figura 5.22 que muestra la distribución de asignaciones de asteroides en cada estación a lo largo de los 80 días disponibles, se observa una distribución equilibrada de asteroides en la mayoría de las estaciones. Asimismo, en la Figura 5.23 que muestra la cantidad de cada material recolectado en cada estación, se puede apreciar un reparto equitativo de los recursos entre las estaciones. Esto indica una asignación eficiente de los recursos para maximizar la masa recolectada y una mejor optimización en la distribución de los materiales. La Tabla 5.13 proporciona un resumen cuantitativo detallado de la mejor solución obtenida. Se incluyen las cantidades de cada material recolectado en cada estación, así como la validez de las restricciones de asteroides, tiempos de llegada y asignaciones. El fitness asociado a la solución refleja un nivel de eficiencia y optimización mejorado en la asignación de recursos en comparación con ejecuciones anteriores del AM.

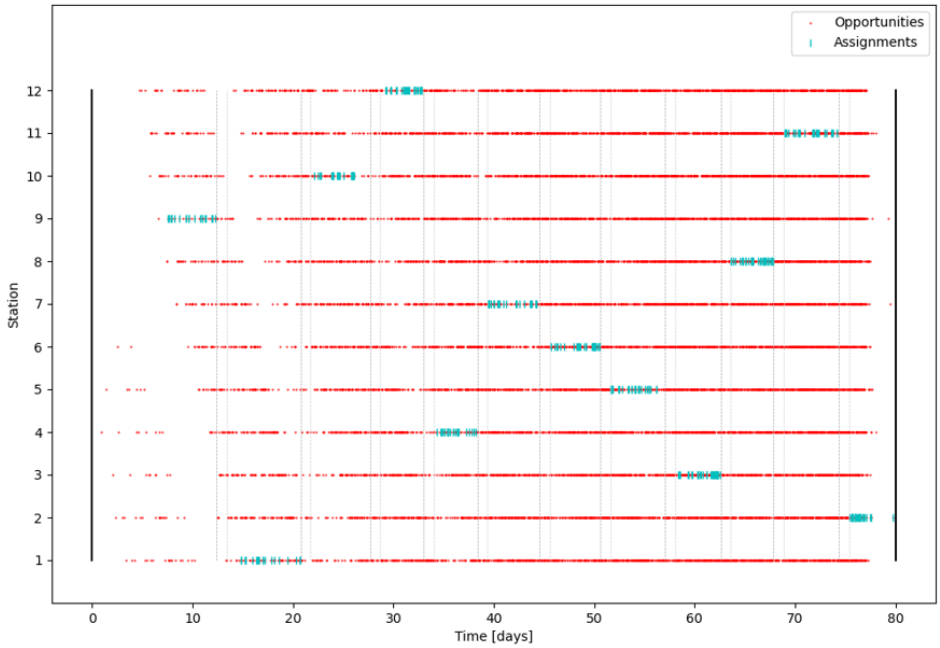


Figura 5.22: Cronograma de oportunidades y asignaciones realizadas en cada estación teniendo en cuenta la mejor solución obtenida con AM_200_1

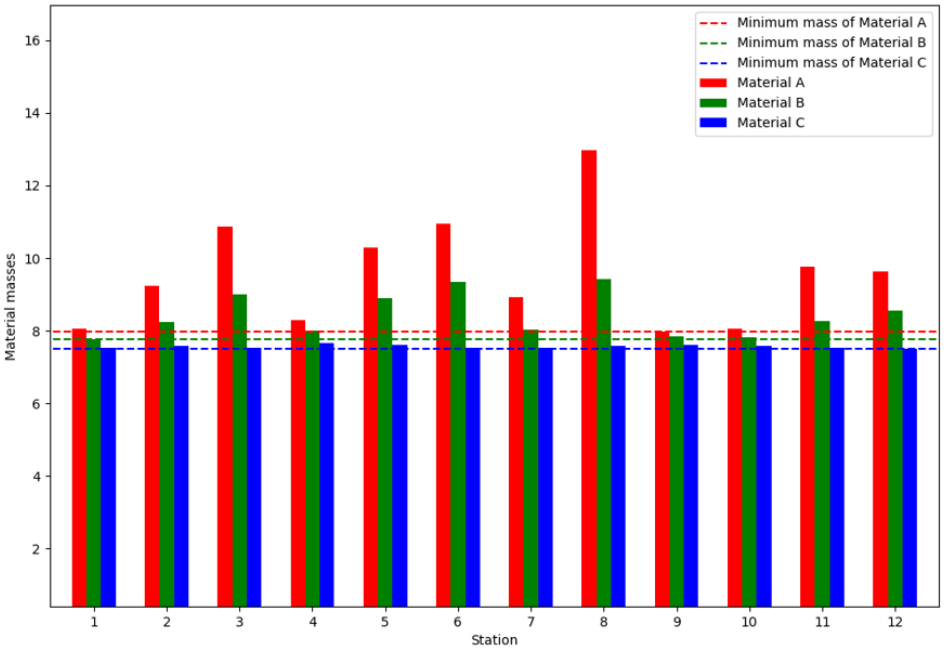


Figura 5.23: Cantidades obtenidas de cada material en cada estación teniendo en cuenta la mejor solución obtenida con AM_200_1

Tabla 5.13: Análisis cuantitativo de la mejor solución obtenida con AM_200_1

Características	**Material A**	**Material B**	**Material C**
Estación 1	8.046321	7.775111	7.532949
Estación 2	9.227783	8.230316	7.590764
Estación 3	10.874599	8.988476	7.519783
Estación 4	8.281012	8.013100	7.647136
Estación 5	10.284822	8.904211	7.595638
Estación 6	10.958044	9.354672	7.522395
Estación 7	8.932361	8.038478	7.517582
Estación 8	12.971123	9.416030	7.568798
Estación 9	7.967291	7.854652	7.599009
Estación 10	8.044622	7.824775	7.586047
Estación 11	9.751421	8.254064	7.532709
Estación 12	9.625073	8.551536	7.501975
IDs de asteroides no válidos	0 de 340		
Tiempos de llegada no válidos	0		
Brecha inter-estación mínima	1.0		
Asignaciones de asteroides no válidas	0		
Fitness	-7.5019		

Para la última ejecución con **AM_200_0.8** se logró obtener una solución con $fitness = -7{,}6291$. Este resultado indica una asignación efectiva de los recursos en las diferentes estaciones a lo largo del tiempo, demostrando la eficiencia del algoritmo en encontrar soluciones de alta calidad para el problema de asignación de recursos. En la Figura 5.25 la distribución de asteroides en cada estación muestra una asignación equilibrada de recursos, esto es fundamental para garantizar una eficiente extracción de recursos en cada estación. En cuanto a la cantidad de cada material recolectado en cada estación, se observa en la Figura 5.25 un reparto equitativo de los recursos entre las estaciones, lo que refleja una asignación eficiente para maximizar la masa recolectada. La Tabla 5.14 proporciona un resumen cuantitativo detallado de la mejor solución obtenida. En resumen, la solución obtenida mediante el AM presenta un alto nivel de eficiencia y optimización en la asignación de recursos. Esto sugiere que el algoritmo es capaz de encontrar soluciones de alta calidad de manera consistente, lo que lo convierte en una herramienta efectiva para abordar problemas de asignación de recursos en diferentes contextos.

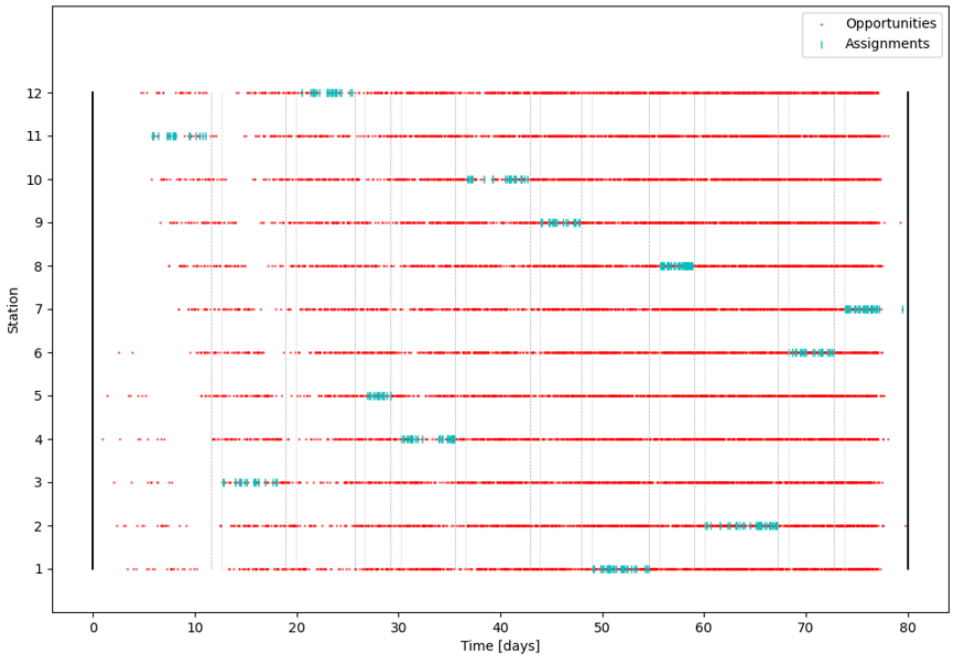


Figura 5.24: Cronograma de oportunidades y asignaciones realizadas en cada estación teniendo en cuenta la mejor solución obtenida con AM_200_0.8

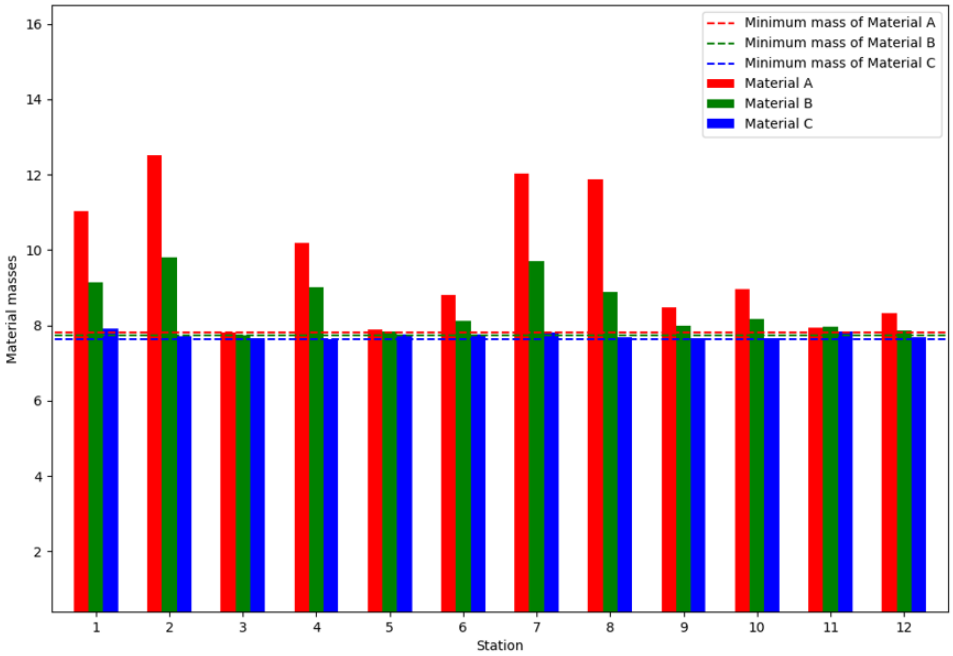


Figura 5.25: Cantidades obtenidas de cada material en cada estación teniendo en cuenta la mejor solución obtenida con AM_200_0.8

Estas fueron las mejores soluciones que se obtuvieron en cada una de las cuatro pruebas distintas utilizando el AM para abordar el problema de asignación de recursos en un contexto específico. Cada prueba se realizó con parámetros diferentes, como el tamaño de la población, el número de generaciones y la probabilidad de cruce. El objetivo de este análisis es comparar los resultados obtenidos en cada prueba y examinar las diferencias entre ellas. Se prestará especial atención al impacto de los diferentes parámetros en la

Tabla 5.14: Análisis cuantitativo de la mejor solución obtenida con AM_100_0.8

Características	**Material A**	**Material B**	**Material C**
Estación 1	11.032163	9.134636	7.921740
Estación 2	12.504532	9.797759	7.721788
Estación 3	7.813846	7.738967	7.655601
Estación 4	10.201278	9.024582	7.629131
Estación 5	7.883275	7.845102	7.756285
Estación 6	8.812129	8.124252	7.768585
Estación 7	12.040651	9.707068	7.814979
Estación 8	11.879074	8.887334	7.687751
Estación 9	8.482506	7.988521	7.664062
Estación 10	8.959116	8.167849	7.664159
Estación 11	7.953741	7.961158	7.840655
Estación 12	8.321507	7.854415	7.674332
IDs de asteroides no válidos	0 de 340		
Tiempos de llegada no válidos	0		
Brecha inter-estación mínima	1.0		
Asignaciones de asteroides no válidas	0		
Fitness	-7.6291		

calidad de las soluciones encontradas por el AM. Mediante esta comparación y análisis, se buscará identificar qué configuración de parámetros produce los mejores resultados en términos de eficiencia y optimización en la asignación de recursos.

Basándonos en los datos de la Tabla 5.15 podemos concluir los siguiente sobre las distintas pruebas realizadas. La **AM_100_1** exhibe una distribución de datos con una mediana ligeramente menor que la media. Esta discrepancia sugiere una posible asimetría hacia la izquierda en la distribución. Sin embargo, el IQR es relativamente estrecho, indicando una concentración de datos alrededor de la mediana. Aunque hay cierta dispersión, la mayoría de los datos se encuentran dentro del IQR. En contraste, la **AM_100_0.8** muestra una distribución con una mediana similarmente inferior a la media. No obstante, los valores mínimo y máximo indican una mayor dispersión de los datos en comparación con la primera prueba. Esto sugiere una distribución potencialmente más asimétrica, con una cola más larga hacia los valores más bajos. La **AM_200_1** revela una distribución de datos con una mediana muy cercana a la media, indicando una simetría en la distribución. Aunque la dispersión es moderada, el IQR es estrecho, lo que sugiere una concentración de datos alrededor de la mediana. La mayoría de los datos se encuentran dentro del IQR, lo que sugiere una distribución más uniforme. Por último, la **AM_200_0.8**, muestra una distribución de datos con una mediana ligeramente inferior a la media, similar a la segunda prueba. Sin embargo, los valores mínimo y máximo indican una dispersión considerablemente mayor de los datos en comparación con las otras pruebas. Esto sugiere una distribución más asimétrica, con una cola más larga hacia los valores más bajos.

En términos de diferencias y apoyándonos en la Figura 5.26, se observa que el valor de cruce influye en la dispersión de los datos, con un valor más bajo (0.8) asociado a una mayor dispersión en comparación con un valor de 1. Además, las pruebas con un tamaño de muestra más grande tienden a tener una distribución más simétrica en comparación con las pruebas de tamaño más pequeño. En conclusión, el tamaño de la muestra y el valor de cruce parecen influir en la distribución de los datos. Muestras más grandes tienden a mostrar una menor dispersión, mientras que un valor de cruce más bajo puede estar asociado a una distribución más asimétrica.

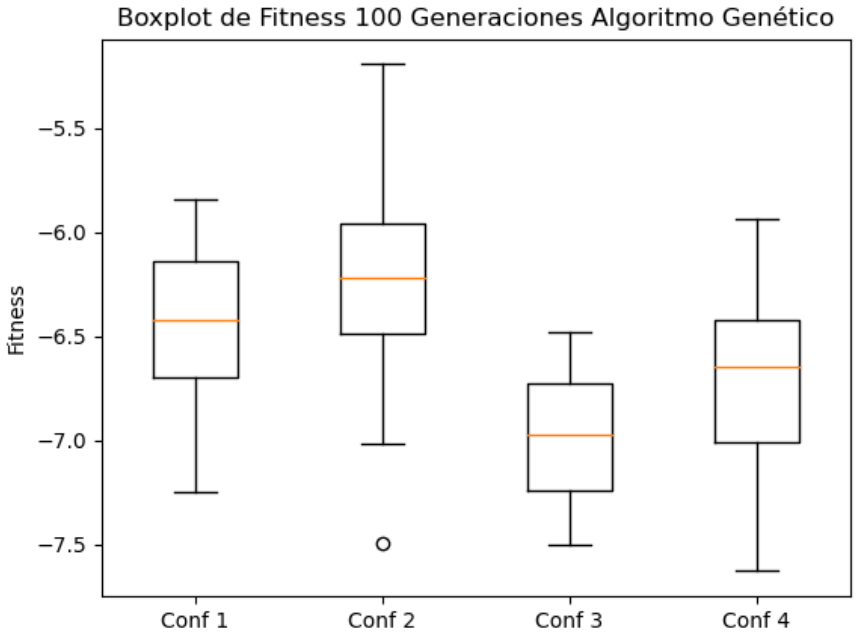


Figura 5.26: Boxplot comparativo del fitness de las soluciones obtenidas en 30 repeticiones del AM con las diferentes pruebas realizadas

Tabla 5.15: Análisis de métricas para diferentes tamaños y valores de cruce

Métrica	**AM_100_1**	**AM_100_0.8**	**AM_200_1**	**AM_200_0.8**
Media	-6.4414	-6.2252	-6.9853	-6.6666
Mediana	-6.4191	-6.2178	-6.9759	-6.6467
Mínimo	-7.2498	-7.4927	-7.5020	-7.6291
Máximo	-5.8385	-5.1923	-6.4782	-5.9377
Primer Cuartil	-6.6957	-6.4842	-7.2389	-7.0090
Tercer Cuartil	-6.1380	-5.9556	-6.7270	-6.4222

Después de haber analizado inicialmente las distribuciones de los datos a través de boxplots y tablas comparativas, se procedió a realizar pruebas estadísticas para una evaluación más precisa de las diferencias entre las distintas configuraciones del AM. El análisis de las pruebas T-test [3] proporcionó una visión más detallada de las comparaciones entre las configuraciones de cruces y tamaños de iteración. A partir de los resultados obtenidos que podemos ver en la Tabla 5.16, se destacan varias conclusiones importantes que complementan el análisis previo:

- **Impacto del tamaño de la población:**: Se observó una mejora significativa en el rendimiento al aumentar el tamaño de la población de 100 a 200. Las comparaciones entre configuraciones con la misma probabilidad de cruce pero diferente tamaño de población (por ejemplo, AM_100_1 vs AM_200_1 y AM_100_0.8 vs AM_200_0.8) revelaron diferencias significativas que indican un mejor desempeño con una población más grande.
- **Influencia de la probabilidad de cruce**: Al analizar las comparaciones entre las probabilidades de cruce, se observa que los valores de T son en su mayoría positivos,

indicando que las medias de las muestras con una probabilidad de cruce del 1 son mayores que las de una probabilidad de cruce del 0.8. A pesar de eso se encontraron mejores soluciones con la probabilidad de cruce del 0.8 respecto a la otra. Podemos decir que no se encontraron diferencias significativas entre configuraciones con misma población y distinta probabilidad de cruce pues con probabilidad de cruce 0.8 se pueden llegar a conseguir mejores soluciones pero el IQR es más amplia que las pruebas con probabilidad de cruce 1.

- **Elección de la mejor configuración**: Aunque la mejor configuración puede variar dependiendo de los criterios específicos del problema y las preferencias del usuario, en términos generales, la configuración que ofrece una mejora constante y significativa en las métricas de desempeño es aquella con una probabilidad de cruce de 1 y un tamaño de población de 200. A pesar de que no siempre es la que muestra las mejores soluciones de forma aislada, esta configuración se destaca por su consistencia en mejorar el rendimiento del AM en comparación con otras configuraciones

Estos hallazgos son fundamentales para la optimización de algoritmos meméticos, ya que permiten identificar las configuraciones que maximizan el rendimiento del algoritmo para un problema específico. Sin embargo, es importante tener en cuenta que estos resultados pueden variar según las características y restricciones del problema en cuestión.

Tabla 5.16: Resumen de las pruebas T-test entre diferentes configuraciones de probabilidad de cruce

Comparación	**T-value**	**p-value**	**Interpretación**
AM_100_1 vs AM_100_0.8	-1.92	0.059	No significativo
AM_100_1 vs AM_200_1	2.28	0.030	Significativo
AM_100_1 vs AM_200_0.8	2.07	0.043	Significativo
AM_100_0.8 vs AM_200_1	2.60	0.014	Significativo
AM_100_0.8 vs AM_200_0.8	3.65	0.001	Significativo
AM_200_1 vs AM_200_0.8	-1.15	0.260	No significativo

Además, se presenta la Figura 5.27 que muestra los resultados obtenidos por la mejor configuración del AM y por la heurística BF. Se observa que el AM supera consistentemente a BF en términos de calidad de las soluciones encontradas. Al considerar la Tabla 5.17, se destaca que el AM_200_1 logra una media de -6.98 con una mediana de -6.98, mientras que BF obtiene una media de -5.03 y una mediana de -4.98. Estas diferencias en los valores medios y medianos indican una mejora sustancial en el rendimiento del AM en comparación con BF. Además, para evaluar la significancia estadística de estas diferencias, se realizó un test t que arrojó un valor de $t = 13{,}666$ con $df = 31$ y $p < 0{,}001$, lo que confirma que la diferencia entre las dos configuraciones es estadísticamente significativa. Estos resultados respaldan aún más la superioridad del AM sobre BF en la resolución del problema de optimización considerado.

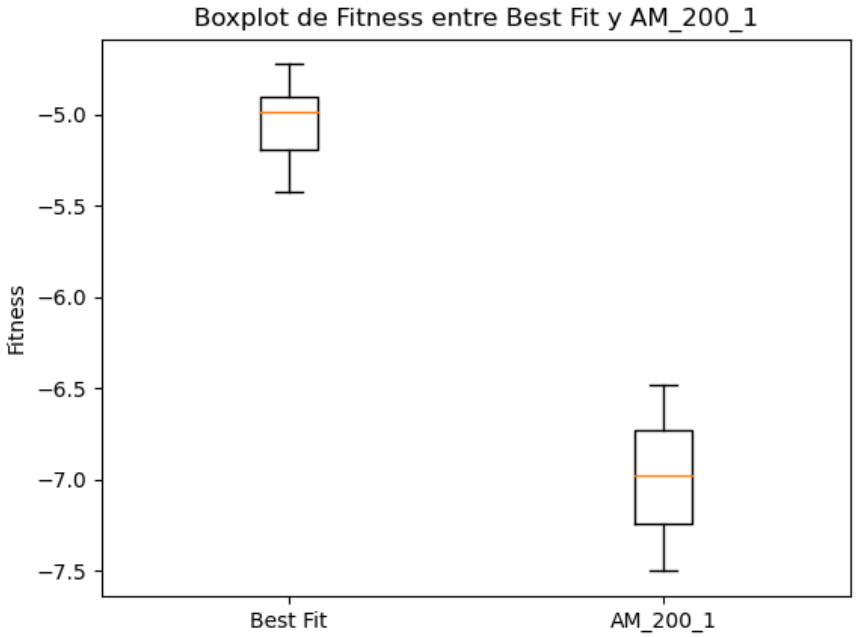


Figura 5.27: Boxplot comparativo del fitness de las soluciones obtenidas con la mejor heurística y la mejor configuración del AM

Tabla 5.17: Resultados Comparativos entre la mejor heurística y la mejor configuración del AM

Métrica	**Best Fit**	**AM_200_1**
Media	-5.0335	-6.9853
Mediana	-4.9809	-6.9759
Mínimo	-5.4256	-7.5020
Máximo	-4.7224	-6.4782
Primer Cuartil	-5.1935	-7.2389
Tercer Cuartil	-4.9041	-6.7270

Capítulo 6

Conclusiones y líneas futuras

En este trabajo se comenzó explorando técnicas simples como la búsqueda aleatoria para abordar el problema de la planificación de entrega de asteroides. Aunque la búsqueda aleatoria proporcionó una base inicial para la exploración del espacio de soluciones, se identificó rápidamente que no era suficiente para alcanzar soluciones de alta calidad debido a su naturaleza no dirigida.

Posteriormente, se desarrollaron y aplicaron heurísticas diseñadas a medida para mejorar el rendimiento y la calidad de las soluciones encontradas. Estas heurísticas ofrecieron mejoras significativas sobre la búsqueda aleatoria, proporcionando puntos de partida más razonables y eficaces para la optimización. Sin embargo, las heurísticas por sí solas todavía presentaban limitaciones en cuanto a la capacidad para explorar exhaustivamente el espacio de soluciones.

Finalmente, se implementó un algoritmo memético, que combina las fortalezas de los algoritmos evolutivos con un procedimiento de mejora de soluciones. Este enfoque ha demostrado ser el más eficaz, logrando soluciones de mayor calidad y consistencia. La capacidad del algoritmo memético para realizar una exploración global del espacio de soluciones y una explotación local ha resultado en un rendimiento superior en comparación con las técnicas previas.

En conclusión, aunque las técnicas simples y las heurísticas ofrecen una mejora progresiva en el rendimiento de la solución del problema, el algoritmo memético se ha destacado como la mejor opción al combinar exploración y explotación de manera efectiva. Los resultados muestran que el algoritmo memético proporciona soluciones de mayor calidad que el resto de técnicas algoritmicas evaluadas para la entrega de recursos en el espacio.

En futuras investigaciones, se propone incorporar criterios de optimización y restricciones adicionales, evaluar el algoritmo en escenarios más realistas y dinámicos, y explorar enfoques híbridos que combinen algoritmos evolutivos y meméticos con técnicas de aprendizaje automático e inteligencia artificial. El algoritmo memético propuesto, es sencillo desde el punto de vista de los operadores de variación utilizados, o desde el procedimiento de mejora utilizado. La propuesta de operadores más complejos y sofisticados que lleven a mejores soluciones es otra línea de interés que habrá que seguir. Por último, se recomienda validar los modelos con datos reales de misiones espaciales y desarrollar simulaciones avanzadas del entorno espacial para ajustar y mejorar la aplicabilidad práctica de los algoritmos propuestos.

Capítulo 7

Summary and conclusions

In this work, we began by exploring simple techniques such as random search to tackle the asteroid delivery problem. Although random search provided an initial basis for exploring the solution space, it quickly became apparent that it was insufficient for achieving high-quality solutions due to its undirected nature.

Subsequently, customized heuristics were developed and applied, designed to improve the performance and quality of the solutions found. These heuristics offered significant improvements over random search, providing more reasonable and effective starting points for optimization. However, heuristics alone still presented limitations in terms of exhaustively exploring the solution space.

Finally, a memetic algorithm was implemented, combining the strengths of evolutionary algorithms with a solution improvement procedure. This approach has proven to be the most effective, achieving higher quality and more consistent solutions. The memetic algorithm's ability to perform global exploration of the solution space and local exploitation has resulted in superior performance compared to previous techniques.

In conclusion, although simple techniques and heuristics offer a progressive improvement in solving the problem, the memetic algorithm has emerged as the best option by effectively combining exploration and exploitation. The results show that the memetic algorithm provides higher quality solutions compared to the other algorithmic techniques evaluated for resource delivery in space.

In future research, it is proposed to incorporate additional optimization criteria and constraints, evaluate the algorithm in more realistic and dynamic scenarios, and explore hybrid approaches that combine evolutionary and memetic algorithms with machine learning and artificial intelligence techniques. Furthermore, the memetic algorithm makes use of simple variation and improvement operators. The particular improvement of those operators could lead to even better results, so the above could be a line of research worth being explored. Finally, it is recommended to validate the models with real data from space missions and develop advanced simulations of the space environment to fine-tune and enhance the practical applicability of the proposed algorithms.

Capítulo 8

Presupuesto

Se propone el siguiente presupuesto solamente realizando un desglose de los costes de recursos humanos, pues no se ha tenido que realizar ningún coste tecnológico a lo largo del proyecto.

Tabla 8.1: Resumen de tipos

Tipos	**Duración (horas)**	**Coste**	**Total**
Estudio del estado del arte	**60**	**25€/h**	**1500€**
Trabajo de desarrollo	**180**	**30€/h**	**5400€**
Análisis de requerimientos	30	30€/h	900€
Diseño de la arquitectura	40	30€/h	1200€
Implementación de las heurísticas	50	30€/h	1500€
Implementación del algoritmo memético	40	30€/h	1200€
Pruebas y depuración	20	30€/h	600€
Experimentación y análisis	**70**	**35€/h**	**2450€**
Total	**310**		**9350€**

Bibliografía

[1] Advanced Concepts Team. Acoteam - spoc - delivery scheduling, 2022. Available: https://indico.esa.int/event/426/contributions/7126/attachments/4732/7220/ACOteam-SpOC-DeliveryScheduling.pdf.

[2] Adam J. Burgasser and Eric E. Mamajek. On the age of the trappist-1 system. *The Astrophysical journal*, 845(2):110, 2017.

[3] Ciencia de Datos. T-test con python, n.d. Available: https://cienciadedatos.net/documentos/pystats10-t-test-python.

[4] Charles Darwin. *On the origin of species by means of natural selection, or The preservation of favoured races in the struggle for life*. John Murray, Albemarle Street, 1859.

[5] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Natural computing series. Springer Berlin Heidelberg, 2015.

[6] European Space Agency (ESA). SPOC: Delivery scheduling, 2022. Available: https://optimize.esa.int/challenge/spoc-delivery-scheduling/p/delivery-scheduling.

[7] GECCO 2022. Gecco 2022 | competitions, 2022. Available: https://gecco-2022.sigevo.org/Competitions.

[8] GECCO 2022. Gecco 2022 | homepage, 2022. Available: https://gecco-2022.sigevo.org/HomePage.

[9] IBM. ¿qué es el modelado de optimización? Available: https://www.ibm.com/es-es/topics/optimization-model#:~:text=El%20modelado%20de%20optimizaci%C3%B3n%20es,cuenta%20restricciones%20y%20objetivos%20espec%C3%ADficos.

[10] Panos M. Pardalos, Valentina Rasskazova, and MichaelÑ. Vrahatis. *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, volume 170. Springer Nature, Cham, 1st edition 2021 edition, 2021.

[11] Alberto Rios. Cuaderno de google colaboratory, 2024. Available: https://colab.research.google.com/drive/1IF8bxEvSDCZAn1vyMaJQ41I9Fy3hi8OY?usp=sharing.

[12] Space Optimisation Competition (SPOC) workshop. Indico at ESA / ESTEC (Indico), September 16 2022. Available: https://indico.esa.int/event/426/timetable/#27-iiia-csic-delivery-scheduli.

[13] Advanced Concepts Team. Gecco 2022 space optimisation competition (spoc), March 2022. Available: https://www.esa.int/gsp/ACT/projects/gecco-2022-competition/.