eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

NUEVOS ALGORITMOS BIOINSPIRADOS PARA EL DESPLIEGUE DE INFRAESTRUCTURAS MÓVILES DE RADIOCOMUNICACIÓN EN DESASTRES NATURALES

Alumno/Alumna: de la Cruz, Farrán, Alberto

Director/Directora: Del Ser, Lorente, Javier

Curso: 2017-2018

Fecha: Martes, 24, julio, 2018

RESUMEN

Este proyecto consiste en el desarrollo e implementación de un software que optimice el despliegue de nuevos nodos de comunicación en áreas catastróficas, para de esta forma proporcionar una mayor área de cobertura a los equipos de rescate.

Para su desarrollo, se plantea un problema multi-objetivo (en el cual se busca el mínimo coste de despliegue y la máxima área cubierta) que dé como solución la posición de los nodos en un área extensa.

Para la resolución de este problema se utilizaran algoritmos bioinspirado, los cuales proporcionan una solución equilibrada para los 2 objetivos planteados. En este problema se ha optado por los algoritmos NSGAII, SMPSO y MOEAD.

Palabras clave: algoritmo, bioinspirado, comunicaciones, optimización

LABURPENA

Proiektu honek hondamendi eremuetan komunikazio-nodo berrien hedapena optimizatzen duen softwarea garatzea eta ezartzea du helburu, erreskate-taldeei estaldura-eremu handiagoa eskaintzeko.

Garapenean zehar arazo anitzeko helburua proposatzen da (gutxieneko inplementazio-kostu eta estalitako gehineko azalera bilatuz), irtenbide gisa nodoen kokalekua ematen duenak eremu zabalean.

Arazo honi aurre egiteko, proposatutako bi helburuak lortzeko bioinsipatuko algoritmoak erabiliko dira. Horretarako aukeratuko algoritmoak NSGAII, SMPSO eta MOEAD izan dira.

Hitz gakoak: algoritmo, bioinsiparatua, komunikazioak, optimizazioak

ABSTRACT

This project consists in the development and implementation of a software that optimizes the deployment of new communication nodes in catastrophic areas, in order to provide a larger area of coverage for rescue teams.

For its development, a multi-objective problem (in which the objectives are, provide the minimum cost of deployment and the maximum area covered) which gives as a result the position of the nodes in a large area.

For the resolution of this problem bio-inspired algorithms will be used, these algorithms provide a balanced solution for the 2 objectives. In this problem we have opted for the following algorithms: NSGAII, SMPSO and MOEAD.

Keywords: algorithm, nature-inspire, communications, optimization

DEFINICIONES

Heurística: manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas…

Meta-heurística: estrategias usadas para escribir algoritmos, que queden por encima de las heurísticas, y van algo más allá.

Redes neuronales: Sistema de computación compuesto por un gran número de elementos que procesan la información, se organizan en niveles.

Algoritmo: Secuencia lógica, finita y con instrucciones que forman una fórmula matemática o estadística para realizar el análisis de datos.

Algoritmo bioinspirado: simula el comportamiento de sistemas naturales para el diseño de métodos heurísticos no determinísticos de búsqueda/aprendizaje/comportamiento…

Población: conjunto de candidatos a ser solución que puede o no cumplir las restricciones del problema.

Mecanismo de evolución: Conjunto de procedimientos por los cuales se modifica la población.

Mecanismos de calificación: Conjunto de procedimientos por los cuales se asigna una calificación a cada elemento de la población. Normalmente es la función a optimizar.

Condiciones de parada: Mecanismo que controla si se ha encontrado una solución o número de operaciones aceptable.

Predictive Analytics: El análisis predictivo es la utilización de datos para predecir tendencias o eventos futuros.

Dataset: Un conjunto de muestras sobre un tema.

Muestras: en inglés “samples”. Suelen constar de características y etiquetas que las identifican.”

Big data: entre las múltiples definiciones que encontramos, la que más nos interesa en este proyecto es la de plataforma desarrollada para procesar una gran cantidad de datos y sacar analíticas de ellos.

REFERENCIAS DE CONTENIDO

Tablas

Ilustraciones

Acrónimos

IDE: Integrated Development Environment

NSGA-II: Non-Dominated Sorting Genetic Algorithm versión 2

NSGA-III: Non-Dominated Sorting Genetic Algorithm versión 3

MOEA/D: Multi-Objective Evolutionary Algorithm

IBEA: Indicator-based evolutionary algorithm

Epsilon-MOEA: Epsilon Multi-Objective Evolutionary Algorithm

SPEA2: Strength Pareto Evolutionary Algorithm

GDE3: Third Evolution Step of Generalized Differential Evolution

PSO: Particle Swarm Optimization Algorithm

OMOPSO: Multi-Objective PSO

SMPSO: Speed-constrained Multi-objective PSO

Epsilon-NSGA-II: Epsilon Non-Dominated Sorting Genetic Algorithm versión 2

Índice

1 Introducción

Este proyecto tiene como meta el desarrollo y despliegue de un software que permita resolver un problema de optimización (de la forma más rápida y precisa posible) y realizando todo esto en tiempo real, esto es así por la naturaleza del tema para el que se desarrolla el software. En muchas situaciones de emergencia el tiempo y la rapidez de actuación son vitales, por ello es necesario que se pueda modificar de forma sencilla los datos de entrada y recalcular las soluciónes de nuevo.

Su principal función es el cálculo de la posición óptima en la que desplegar los nodos de comunicación en el área de la catástrofe. Relacionado con esto, se busca que sea lo más sencillo posible de escalar, tanto en cuanto al número de nodos, como en cuanto al tipo.

Cabe mencionar, que este proyecto está directamente relacionado con otro proyecto (Modelo predictivo para la estimación de trayectorias de recursos de vigilancia y contingencia de desastres naturales), formando de esta forma un sistema completo que permite por un lado predecir la trayectoria de las tropas desplegadas y por otro optimizar el despliegue de los nodos de comunicación en base a las posiciones calculadas por el otro proyecto.

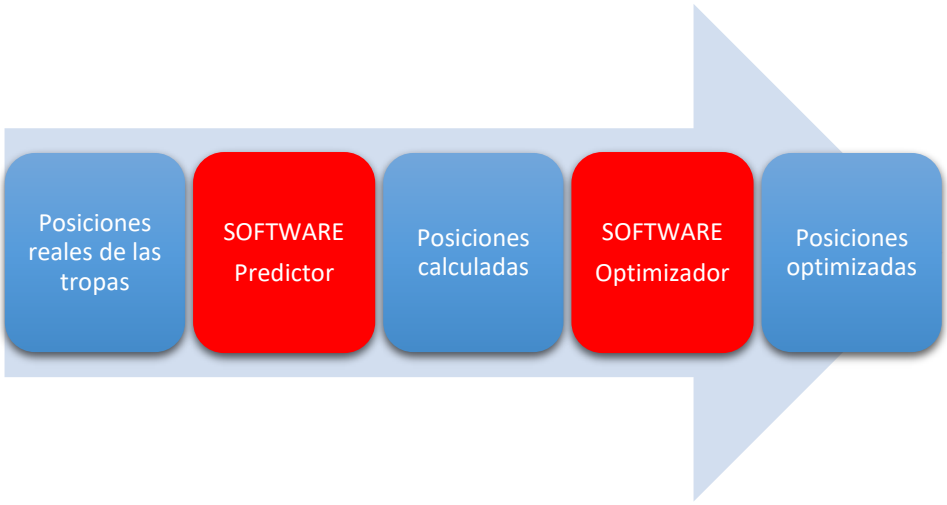

Ilustración 1. Estructura completa del sistema

Como podemos ver en la Ilustración 1, el sistema consta de:

- Un flujo de datos procedente de un dataset con muestras GPS de distintas actuaciones que han realizado las brigadas [4].
- Un software predictor que analizando ese flujo de datos saca la predicción del siguiente punto en el que se encuentra la brigada.
- Este nuevo flujo se introduce en el siguiente software, el cual optimiza en base a estas muestras, para obtener la posición más óptima en la que desplegar un nodo de comunicaciones.

- Finalmente, este ciclo empieza de nuevo, realimentándose con los datos obtenidos de ambos software en la salida.

Para poder resolver este problema, en el proyecto se proponen las bases y principios teóricos en los que se basa para plantear el problema y desarrollar una solución al mismo. Entre estas soluciones destacan la teoría y el estado del arte de los algoritmos "multi-objetivo" los cuales están basados en sistemas "bioinspirados". También se mostraran ejemplos de uso e implementaciones de estos algoritmos.

En posteriores apartados se comentará el alcance y objetivos buscados en la resolución de este problema, ya que debido a su complejidad, es necesario establecer límites, tanto en su planteamiento (número de objetivos buscados, cantidad de recursos disponibles…), como en su solución (formato de los datos de salida, representación de estos datos…).

Además, se incluye una sección de metodología donde se detalla el proceso de diseño e implementación de software, junto con todos los documentos y detalles técnicos del mismo.

El proyecto incluirá diferentes apartados relacionados con la gestión, coste del desarrollo y su implementación:

- Análisis de riesgos: se analizan los diferentes riesgos que se dan al realizar un proyecto, estos casos pueden ser tanto riesgos internos (software, problemas desarrollo…) como externos (fluctuación del mercado, crisis económica…).
- Planificación: se indicaran los hitos y pasos que sigue el desarrollo mediante un diagrama de Gantt.
- Presupuesto: consiste en un análisis desde el punto de vista económico del proyecto, el cual se separa en partidas. Mediante diferentes indicadores económicos es posible ver si el proyecto es viable.

Todos estos documentos tienen una gran importancia a la hora de desarrollar la idea y de definir el alcance y objetivos del proyecto.

Finalmente se incluirán anexos, en los que se adjuntarán las partes del código que sean importantes, por un lado todo lo relacionado con la gestión y adaptación de los datos con los que operar, y por otro el algoritmo que procesa y devuelve los datos.

2 Contexto

Hoy en día, la frecuencia y magnitud de las catástrofes naturales han sufrido un importante incremento. Además, debido a los recortes de presupuesto sufridos por los gobiernos desde el comienzo de la crisis y a otros factores como el cambio climático, está convirtiéndose en una tarea difícil para los equipos desplegados en las áreas catastróficas el cumplimiento de su misión.

Solamente en Europa se han producido más de 2700 incendios en el año 2016 [1], siendo unos de los más graves el ocurrido en Portugal, más concretamente en Pedrógão Grande, en el cual se quemaron más de 40000 hectáreas y fallecieron 64 personas [2]. Las comunicaciones en el área se vieron afectadas, y un grupo de personas que intentaban huir de las llamas, se quedó atrapado en una zona de difícil acceso y alto peligro. A pesar del esfuerzo de las brigadas de bomberos, fue demasiado tarde y no pudieron hacer nada por ellos.

Otro incendio destacable es el ocurrido en Estados Unidos, más concretamente en el estado de California. En este incendio, tras 22 días de intensas llamas y más de 113000 hectáreas quemadas fue controlado por los bomberos. Estos incendios costaron la vida a 44 personas y destruyeron cerca de 8900 edificios [6].

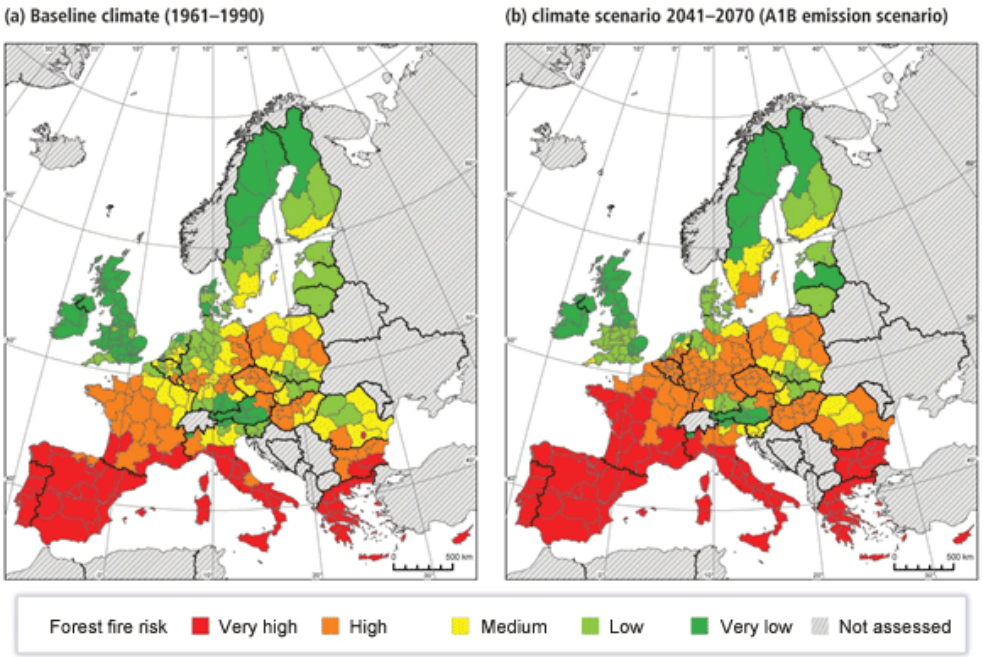

Ilustración 2. Mapa de riesgo de incendios en Europa

Como podemos ver en la Ilustración 2, el número de incendios por toda Europa ha sufrido un importante aumento, debido principalmente al cambio climático. En todos los países del sur de Europa se está produciendo un aumento de la desertización, que provoca un incremento de las temperaturas y por consiguiente un mayor riesgo de incendio.

La intensidad y rápida propagación de este tipo de incendios, ha llevado a la comunidad científica a la investigación de nuevas medidas para la predicción, prevención, detección y extinción de incendios, siendo la predicción uno de los puntos con mayor área para la investigación.

El uso de algoritmos para la predicción de catástrofes naturales tiene un amplio recorrido y es muy probable que en el futuro se vea una importante inversión de recursos, tanto económicos como humanos, en el desarrollo de mejores algoritmos, eficientes y con poca probabilidad de fallo.

Además, hay que tener en cuenta que una de las principales estrategias para la extinción de incendios es la intervención humana directa (con el riesgo asociado que lleva), es decir, el despliegue de brigadas. Esto, tradicionalmente ha sido efectivo en áreas de extensión pequeña o media, pero la cantidad de peligros que conlleva para las brigadas (desorientación, estrés por el calor, fatiga, humo, polvo, quemaduras…) y de situaciones que han terminado en tragedia [3] cuestionan su eficacia.

Por desgracia, a día de hoy, todavía es necesaria la intervención humana en las tareas de extinción, por ello para prevenir situaciones peligrosas y reducir el riesgo, es crucial el despliegue de redes de comunicaciones en las áreas de la catástrofe (generalmente en estas áreas el número de nodos de comunicación es bastante limitado, y dependiendo de la naturaleza de la catástrofe pueden verse afectados en mayor o menor medida).

Por ello, para garantizar la seguridad del personal de emergencias, es necesario el despliegue de nodos en posiciones estratégicas, para de esta forma, asegurarse de que los equipos de emergencias están informados en todo momento de la posición del resto de quipos, además de obtener información en tiempo real de cómo se desarrolla la catástrofe. Todo esto buscando reducir el coste del despliegue y maximizando el área cubierta por los nodos.

Este despliegue se puede llevar a cabo de manera fácil y eficiente usando algoritmos bioinspirado, es decir aquellos algoritmos que basan sus resultados en modelos inspirados en la naturaleza.

3 Estado del arte

Este tipo de problemas multi-objetivo, en su mayor parte son de optimización combinatoria, donde las métricas involucradas no son fácilmente tratables con las técnicas clásicas de optimización, es decir no es posible obtener una solución exacta al problema, solamente una aproximación.

Por ello, tiene un gran interés entender cómo se empezó a desarrollar este tipo de problemas y las técnicas que se aplican para su resolución.

3.1 Algoritmos

Actualmente múltiples empresas en el ámbito de las telecomunicaciones y de la tecnología están desarrollando algoritmos que suplan aquellas tareas repetitivas y complicadas de realizar. Entre estos podemos encontrar algoritmos de optimización, de ruta óptima, de recopilación de datos, de reconocimiento facial, de predicción…

El potencial de estos programas está en auge, por ello además de su uso en áreas tan específicas como son las comunicaciones o el Big data. Puede ser de interés su uso en otras como son la medicina, la industria, el procesamiento de imagen…

3.2 Heurística

La heurística es una palabra griega que significa “descubrir”, “encontrar”. Se puede definir como un conjunto de reglas que se realizan de forma metodológica, y las cuales no tienen por qué estar definidas, que sugieren o establecen como proceder en la elaboración de hipótesis para la resolución de un problema. Su uso se da en múltiples áreas y situaciones, aunque el principal campo en el que se usa es el de la ingeniería.

En la ingeniería, una heurística es un método basado en la experiencia que puede utilizarse para la resolución de una amplia variedad de problemas de manera eficiente. Los problemas en los que se puede utilizar varían desde problemas de diseño y optimización, hasta problemas de organización y planteamiento de recursos.

En relación con la informática y el campo de la computación, su objetivo principal es encontrar algoritmos con buenos tiempos de ejecución y soluciones que, no siendo las óptimas, cumplan de manera adecuada los objetivos del problema.

Cabe mencionar que los problemas que utilizan algún algoritmo heurístico también se pueden resolver utilizando algoritmos específicos, pero debido a que estos en ocasiones implican un gasto de recursos y tiempo demasiado alto (e incluso en ocasiones son inviables, ya que no es posible obtener una solución exacta) se opta por el uso de los primeros.

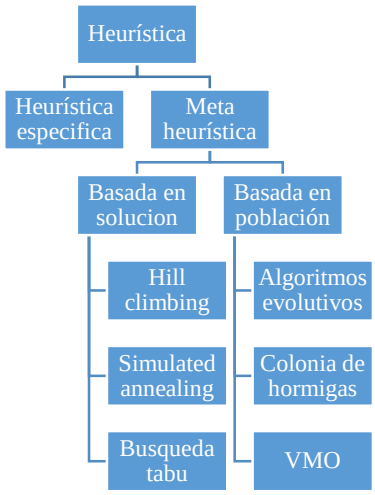

Ilustración 3. Clasificación de heurística

Esto quiere decir que se hace uso de la heurística para encontrar las soluciones que siendo válidas, pueden no ser exactas, pero si lo bastante próximas a la solución como para poder obtener una conclusión correcta.

Este tipo de soluciones permiten reducir el tiempo que se tarda en obtener una solución, además de permitir abordar problemas con unas dimensiones difíciles de manejar por otros algoritmos (gran cantidad de condiciones, cantidad de datos de entrada muy alta…).

En función de cómo se obtiene los resultados y se realiza la validación de la solución, estos se pueden clasificar como algoritmos basados en:

- **Métodos de descomposición**: como su nombre indica, se divide el problema inicial en problemas más fáciles de resolver.
- **Métodos inductivos**: se plantea una versión del problema más sencilla y tras obtener una solución, se aplica al problema completo. Estas soluciones más sencillas son aplicables al problema completo.
- **Métodos de reducción**: consiste en reducir la complejidad del problema, para ello se reduce el espacio de soluciones del problema. El riesgo principal es dejar fuera del espacio soluciones válidas.
- **Métodos constructivos**: consiste en construir paso a paso una solución del problema, es decir dar una solución determinista del problema. Suele basarse en elegir la mejor solución de cada iteración
- **Métodos de búsqueda local**: estos métodos comienzan con una solución del problema y en cada iteración la mejoran. Este método finaliza cuando para una solución no existe una solución siguiente que mejore la anterior.

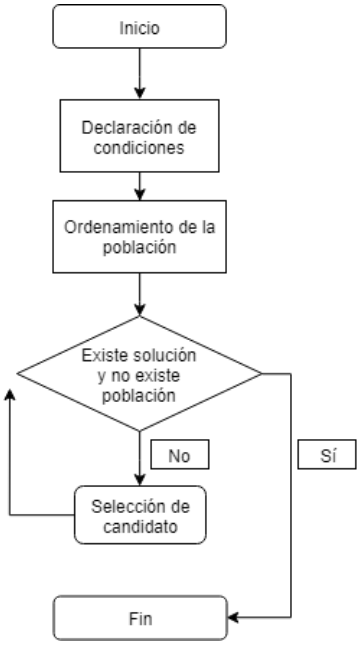

Ilustración 4. Diagrama de flujo de un algoritmo heurístico específico

Cuando a un algoritmo heurístico, se le modifica uno o varios parámetros para adaptarlos a problemas de distinta naturaleza entramos en el campo de la meta-heurística.

Si bien todos los métodos anteriormente indicados han servido para la resolución y desarrollo de una gran cantidad de problemas, a día de hoy la base de los algoritmos meta-heurísticos son los métodos constructivos y los métodos de búsqueda local.

3.3 Meta-heurística

La meta-heurística se usa generalmente para hacer referencia a un subcampo muy importante dentro de la optimización estocástica (también llamada heurística).

Su principal uso se da en problemas:

- De los que no se dispone inicialmente mucha información para su resolución.
- En los que no se dispone del formato exacto que tiene que tener la solución del mismo.
- No existe un método establecido de resolución y en caso de existirlo en inviable (ya sea por tiempo o por recursos) su aplicación.
- En los que la búsqueda por fuerza bruta no es una opción.
- En los que el espacio del problema es muy grande y difícil de trabajar con él.

A pesar de todas estas aparentes limitaciones que pueda tener un problema, la solución que nos proporciona un algoritmo de este tipo se puede analizar y sacar conclusiones valiosas.

Cabe mencionar, como se ha dicho en ocasiones anteriores, que la solución obtenida en este tipo de algoritmos no es exacta (y en ocasiones no es válida), por ellos en necesario analizar y validar los resultados.

Los principales algoritmos en los que se hace uso la meta-heurística (y en los cuales se basan una importante cantidad de algoritmos más complejos) son la "Random Search" (búsqueda aleatoria) y el "Hill Climbing" (escalada simple).

En la "búsqueda aleatoria" se calcula las soluciones al problema de forma aleatoria durante un cierto periodo, y se excluyen aquellas menos adecuadas, quedando los mejores resultados. Este algoritmo es bastante poco optimo es cuanto a recursos y tiempo, por ello es más habitual el uso del algoritmo "Escalada simple".

Este algoritmo comienza con una solución arbitraria del problema y luego de forma iterativa e incrementalmente vuelve a calcular la solución con una variación de un parámetro. Si el cambio produce una mejor solución, se vuelve a cambiar ese parámetro, así hasta que ya no se produzcan mejoras en la solución.

3.4 Algoritmos bioinspirado

Estos algoritmos tratan de imitar el comportamiento de sistemas biológicos. Su funcionamiento consiste en elegir de una población de resultados aquellos que sean los más fuertes (función fitness) y en función de estos resultados generar nuevos resultados. En resumen, en el proceso iterativo se tiene en cuenta el crecimiento, desarrollo, reproducción, selección y supervivencia de las soluciones que mejor se adapten a los objetivos.

Tabla 1. Ejemplos de diferentes sistemas biológicos

Redes neuronales
Optimización basada en colonias de hormigas
Optimización basada en enjambre
Algoritmos basados en la evolución (genéticos)
Algoritmos inmunológicos

Dentro de los algoritmos bioinspirado, los algoritmos genéticos son los que mayor uso tienen dentro de la resolución de problemas.

Un algoritmo genético es un método para resolver tanto problemas con restricciones como sin ellas que están basados en la selección natural, es decir la evolución biológica.

El algoritmo genético modifica de forma repetida una población de soluciones individuales. En cada paso el algoritmo genético elige soluciones aleatorias de la población como padres y generan un hijo para la siguiente generación, el cual tendrá mejores características que sus padres, ya que se combinan sus resultados.

A lo largo de las sucesivas generaciones, la población evoluciona hacia una solución óptima.

Su funcionamiento consiste en:

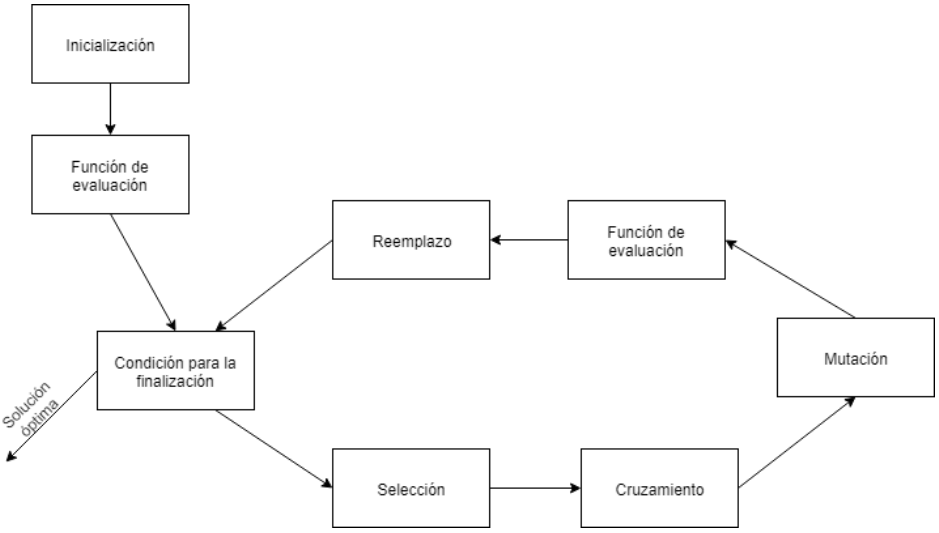

Ilustración 5. Funcionamiento algoritmo genético

- **Inicialización**: Generar aleatoriamente la población inicial, siendo cada una de estas soluciones posibles resultados para el problema planteado. A estas soluciones se las conoce como cromosomas.
- **Función de evaluación**: a cada uno de los cromosomas se le aplica la función de evaluación para obtener un indicador de como de correcta es la solución.
- **Condición para la finalización**: existen múltiples implementaciones, generalmente se opta por limitar a un número de iteraciones máximo el algoritmo, o detenerlo cuando no haya cambios en la población. En cada iteración se realiza:
    - Selección: Después de obtener un indicador para cada cromosoma se eligen aquellos que serán cruzados en la siguiente generación. Los cromosomas que tienen un índice mejor tienen una probabilidad más alta de ser seleccionados.
    - Cruzamiento: este es el principal operador (el que más influencia tiene en las muestras), su funcionamiento consiste en operar sobre dos cromosomas a la vez para generar nuevos descendientes (los cuales se combinan para tener características).
    - Mutación: se realiza una modificación al azar de una característica de la población (de esta forma disminuye el espacio de búsqueda).
    - Reemplazo: una vez aplicados los operadores, se seleccionan los mejores individuos de la población, y se aplican a la generación siguiente.

3.5 Optimización multi-objetivo

En ocasiones es necesario optimizar más de una función (y en muchos casos dichas funciones son completamente opuestas, y por lo tanto imposibles de satisfacer al completo).

Por ello en este tipo de optimización lo que se busca no es que se optimice al máximo los objetivos, si no que se busca la opción más equilibrada. De todo el conjunto de soluciones siempre existe una que es dominante sobre el resto de soluciones.

Es decir si esa solución es igual de buena que la otra en todas sus características, y superior en al menos 1 esa solución será dominante. En caso de que no ocurriera esto se dice que las soluciones son no dominantes, y por lo tanto estas son las mejores soluciones del problema.

4 Beneficios del proyecto

Como todo proyecto desarrollado en el campo de la ingeniería es interesante analizar los beneficios económicos, técnicos y sociales que tiene. Esto puede ser un indicador útil de cara a decidir si un proyecto es viable económicamente, si es necesario realizarlo por las mejoras técnicas que puede aportar a la industria o si por otro lado no es del todo viable económicamente, pero puede aportar una mejora cualitativa a la sociedad. Por ello, a continuación se comentan los 3 puntos de vista principales desde los que analizar el proyecto:

4.1 Económicos

Desde el punto de vista económico a primera vista puede parecer caro [Apartado 9], pero analizando en más detalle el proyecto podemos decir que:

- El despliegue más eficiente permitiría un coste menor de todos los medio, tanto aéreos como terrestres.
- Se reduciría el área afectada, al ser más rápida la llegada de los medios. Además también repercutiría en los gastos de recuperación de la zona afectada, ya que al ser menor área estos serán menores.
- Todo lo anterior repercutirá de forma positiva en las arcas pública, y por consiguiente en el contribuyente. Debido a esto, también influirá en el resto de beneficios este punto.

Como vemos el punto de vista económico no es de los más destacados pero tiene un peso importante dentro de los beneficios.

4.2 Técnicos

Desde el punto de vista técnico, el uso de algoritmos para situaciones de emergencia no es nuevo [Apartado 5], a pesar de ello en este proyecto:

- Este sería uno de los beneficios donde mayor influencia tendría, su uso cambiara las estrategias y normas a seguir en situaciones de emergencia. Ya que al colocarse los medios de forma más óptima, se pueden desplegar menos brigadas y estas tener mayor efecto.
- Este tipo de algoritmos, pueden ser exportables a otras áreas de forma sencilla y con un resultado sobresaliente. Por ejemplo en el diseño de las líneas de montaje, en el despliegue de los recursos a lo largo de la misma…. Por ello la industria puede ser uno de los grandes sectores beneficiados:

Sin duda el mayor impacto sería en los beneficios técnicos, ya que todavía tiene mucho potencial por explorar estos algoritmos, y su uso en el área de la prevención de desastres naturales es bastante bajo.

4.3 Sociales

Dentro del impacto social su principal punto sería en el desplegué de los nodos, cuando se busca la mayor cobertura posible se evitarían situaciones en las que algún equipo se quede aislado del resto y por lo tanto se pueda evitar que se produzcan víctimas.

5 Alcance y objetivos

A continuación, se procederá a realizar un análisis y descripción de los objetivos de este proyecto, además de indicar el alcance del mismo. Para esto se indicaran las áreas del conocimiento que se estudian, haciendo hincapié en las soluciones y analizando si estas son válidas, o no cumplen con los objetivos propuestos.

La definición del alcance de un proyecto, tiene una gran importancia en el mismo, independientemente de que metodología se elija y aplique al proyecto.

Los principales puntos a tener en cuenta para establecer un alcance son:

- Establecer un plazo de tiempo para la finalización del proyecto.
- Estudiar y establecer los objetivos del proyecto.
- Indicar los procedimientos que se van a realizar para la realización del proyecto.
- Distribuir de forma adecuada los recursos disponibles.

Concretamente, en este proyecto los objetivos serían:

Tabla 2. Objetivos y alcance del proyecto

Adaptar el problema	Muestras del predictor
Plantear un problema	Optimización multi-objetivo
Medir la calidad de los resultados	Validez de las muestras obtenidas
Validar el proyecto	Comprobar su funcionamiento conjunto

En primer lugar, es necesario **adaptar** las muestras que se reciben del algoritmo predictor, ya que el formato es diferente a las utilizadas por el algoritmo optimizador.

Tras esto, se tiene como objetivo **plantear y desarrollar un problema de optimización** multi-objetivo, donde se tienen que definir los diferentes parámetros críticos que se quieren optimizar y los resultados que se quieren conseguir.

En este caso, las muestras obtenidas en la solución representan la posición idónea para desplegar el equipo de comunicaciones móviles. Esta solución debe proporcionar la mayor área de cobertura posible, a la vez que reduce al máximo el presupuesto de desplegar los mismos. Es decir, como se puede ver los parámetros que busca optimizar son el área de cobertura total y el coste de desplegarlos.

Otro objetivo es **medir la calidad** de los resultados obtenidos, ya que al ser estos las posiciones de los equipos móviles, es necesario comprobar si se cumplen los objetivos y parámetros indicados previamente. Para ello, se utiliza los algoritmos bioinspirado, los cuales proporcionan las soluciones que mejor se adaptan a la definición del problema. Es decir las mejores soluciones por objetivo, en este caso:

- Las soluciones que den como resultado el mínimo coste en el despliegue.

- Las soluciones que den como resultado el mínimo riesgo de los equipos (o máxima área de cobertura).
- Las soluciones que den como resultado un punto medio, el equilibrio entre riesgo y coste.

Finalmente, una vez finalizada la fase de desarrollo es necesario **validar,** mediante un conjunto de pruebas el correcto funcionamiento y ejecución del mismo.

6 Descripción de la metodología

6.1 Planteamiento del problema

Para entender de forma sencilla el planteamiento del problema, se presenta el siguiente esquema del mismo.

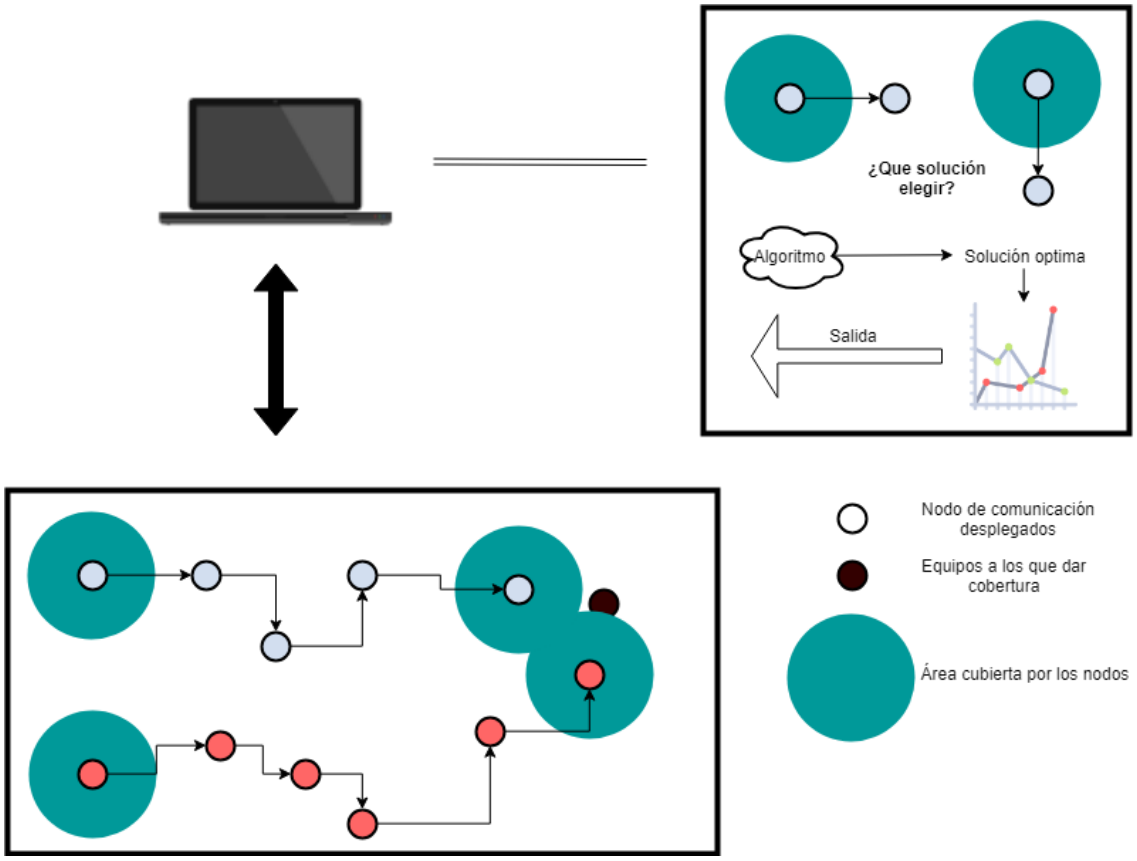

Ilustración 6. Esquema del problema

Supongamos un conjunto de equipos (enjambre) de tamaño $\mathcal{N}$, siendo este conjunto de un tamaño $|\mathcal{N}| = N$ y siendo las posiciones de este conjunto ${p_n}^{\Delta,t}$, las cuales son dependientes del tiempo $\{{p_n}^{\Delta,t}\}_{n=1}^{N} = \{({x_n}^{\Delta,t}, {y_n}^{\Delta,t})\}_{n=1}^{N}$ (donde t hace referencia a esta variación del tiempo) sobre un área $S^{\blacksquare}$ siendo esta área $S^{\blacksquare} = [x_{min,}, x_{max}]X[y_{min,}, y_{max}]$.

Además de este conjunto de equipos, se introduce en el área otro conjunto, esta vez de nodos de comunicación, $\mathcal{M}$ siendo el tamaño de este conjunto $|\mathcal{M}| = M$. Además las coordenadas de estos equipos son $\{{p_m}^{\blacksquare,t}\}_{m=1}^{M} = \{({x_m}^{\blacksquare,t}, {y_m}^{\blacksquare,t})\}_{m=1}^{M}$.

Además, los nodos introducidos tendrán diferentes características, por ejemplo:

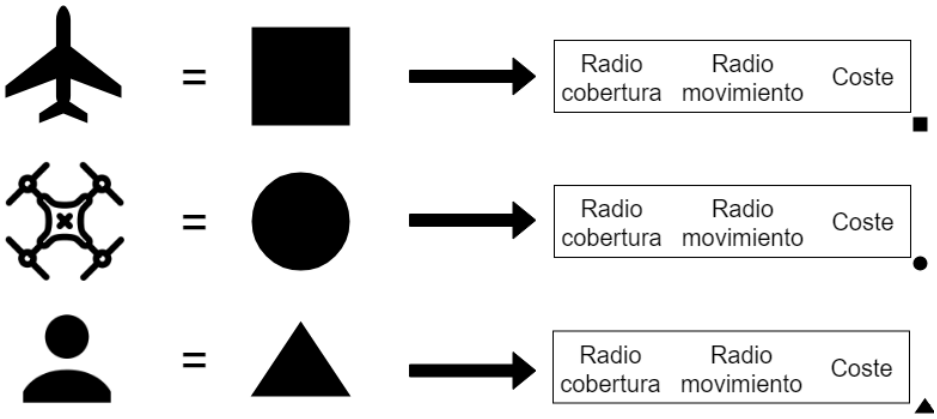

Ilustración 7. Ejemplos de posibles nodos a desplegar

Cada nodo desplegado está equipado con una antena que les permite dar cobertura a un área de $\{S_m^{\blacksquare}\}_{m=1}^{M}$ alrededor de su propia localización.

Esta área en la que se puede dar cobertura se representa haciendo las siguientes suposiciones:

- El área de cobertura se supondrá circular.
- El radio se representará como $R_m^{\blacksquare}$.

Quedando el área $S_m = \{(x,y)\, \epsilon\, S^{\blacksquare} : \left(x - x_m^{\blacksquare,t}\right)^2 + \left(y - y_m^{\blacksquare,t}\right)^2 \leq R_m^2\}$ (área indicada con ● en la Ilustración 6)

Un nodo $m\, \epsilon\, \mathcal{M}$ dará cobertura a un equipo $n\, \epsilon\, \mathcal{N}$ cuando su distancia $d_{m,n}^t$ no supere el límite indicado por D_{max} que indica el radio máximo de cobertura de ese nodo desplegado:

$$d_{m,n} = \sqrt{(x_n^{\Delta,t} - x_m^{\blacksquare,t})^2 + (y_n^{\Delta,t} - y_m^{\blacksquare,t})^2} \leq D_{max}$$

Además el movimiento de los nodos involucra un consumo de recursos y por lo tanto un aumento del coste del despliegue. El coste de un nodo se representa con $C_m^{\blacksquare}$:

$$C_m^{\blacksquare} = d_m * c_m^{\blacksquare}$$

Donde $c_m^{\blacksquare}$ hace referencia al coste del nodo por unidad de distancia.

Con estas definiciones, se puede plantear los objetivos del problema, en este caso máxima cobertura de equipos y mínimo coste de los nodos desplegados.

Matemáticamente, la cobertura de los equipos a los que pueden dar cobertura M nodos, se puede modelar como

$$Cobertura = U\left(\blacksquare,\ x_n^{t+t'},\ y_n^{t+t'},\ x_m^{\blacksquare,t},\ y_m^{\blacksquare,t}\right) = \frac{1}{M}\sum_{i=1}^{M} I$$

Siendo ■ el tipo de nodo que se ha desplegado, $x_n^{t+t'}$ la coordenada 'x' donde se supone que se encontrará el equipo desplegado en un instante de tiempo t', $y_n^{t+t'}$ la coordenada 'y' donde se supone que se encontrará el equipo desplegado en un instante de tiempo t', $x_m^{\blacksquare,t}$ la coordenada 'x' donde se encuentra el nodo desplegado e y_m^t la coordenada 'y' donde se encuentra el nodo desplegado.

I es una función de indicación binaria tal que I=1 si se cumple que $d_{m,n} \leq D_{max}$ siendo 0 en cualquier otro caso.

Por lo tanto el objetivo que se busca para este proyecto es:

$$maximize\{Cobertura\}$$

Matemáticamente, el coste de desplegar M nodos se puede modelar como:

$$Coste = \sum_{\blacksquare} C_m^{\blacksquare}$$

Siendo $C_m^{\blacksquare}$ el coste de desplegar un nodo. Y quedando por lo tanto, el otro objetivo como:

$$minimize\{Coste\}$$

6.2 Planteamiento de la solución

Para la correcta resolución del problema planteado se propone aplicar algoritmos bioinspirado, en este caso se opta por una comparativa de los algoritmos NSGAII, SMPSO y MOEAD para poder comprobar así cuál es el mejor en diferentes situaciones.

Estos algoritmos meta-heurísticos centralizados son capaces de balancear de la manera más óptima posible los 2 objetivos planteados en el problema, siendo detallados a continuación.

Al estar basados en fenómenos biológicos, la implementación de los 3 algoritmos es similar a la indicada en el [Apartado 3.4].

6.2.1 NSGAII (Non-Dominated Sorting Genetic Algorithm version 2)

El NSGAII es uno de los algoritmos evolutivos más populares de optimización multi-objetivo. Está basado en la ordenación no dominada de muestras, es decir se divide una población en diferentes fronteras, y así capa tras capa se llega a una frontera que no es dominada por ninguna otra. En este caso, esas fronteras contendrán las soluciones con las rutas de los nodos desplegados.

Su funcionamiento consiste en:

1. Ordenar la población de candidatos basándose en cuál de ellos domina sobre el resto, creándose de esta forma los rankings o fronteras.

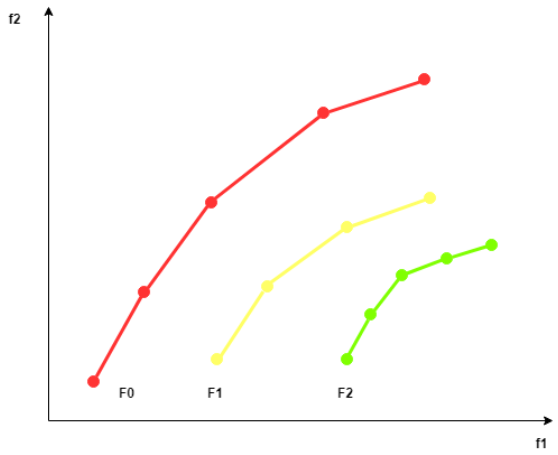

Ilustración 8. Ejemplo de fronteras obtenidas con NSGAII

2. Tras obtener las fronteras, y para buscar las soluciones óptimas, se busca la distancia más corta de cada solución, y se van acumulando.

6.2.2 SMPSO (Speed-constrained Multi-objective PSO)

Para entender el SMPSO primero es necesario explicar el algoritmo PSO (Particle Swarm Optimization), el cual es un algoritmo meta-heurístico, basado en fenómenos biológicos el cual tiene un gran desarrollo e implementación en la resolución de problemas multi-objetivo.

PSO tiene una serie de limitaciones en algunos tipos de problemas, que produce que las soluciones obtenidas en estas situaciones sean incoherentes.

Debido a esto en una de las múltiples implementaciones que existen de este algoritmo, la SMPSO, se añade una solución a estos problemas, en forma de restricción extra la cual permite obtener resultados que pueden competir con los obtenidos en el uso del NSGAII.

6.2.3 MOEAD (Multi-Objective Evolutionary Algorithm)

El MOEAD en un algoritmo meta-heurístico multi-objetivo, el cual se basa en la descomposición del problema. De esta forma se dispone de un subconjunto de problemas (con un único objetivo) que se optimizan de forma paralela.

Su principal característica es que tiene menor complejidad que NSGAII, consiguiendo unos resultados parecidos en cuanto a calidad de las mismas.

Su uso mayoritario es en problemas de 3 objetivos, siendo este caso en el cual da los mejores resultados.

6.3 Solución propuesta

En este punto se detalla la arquitectura seguida para la resolución del problema, desde una descripción general de las herramientas utilizadas, bloques del proyecto… hasta un análisis más en profundidad del código de cada bloque.

Para ilustrar de forma sencilla el desarrollo se plantea el siguiente diagrama:

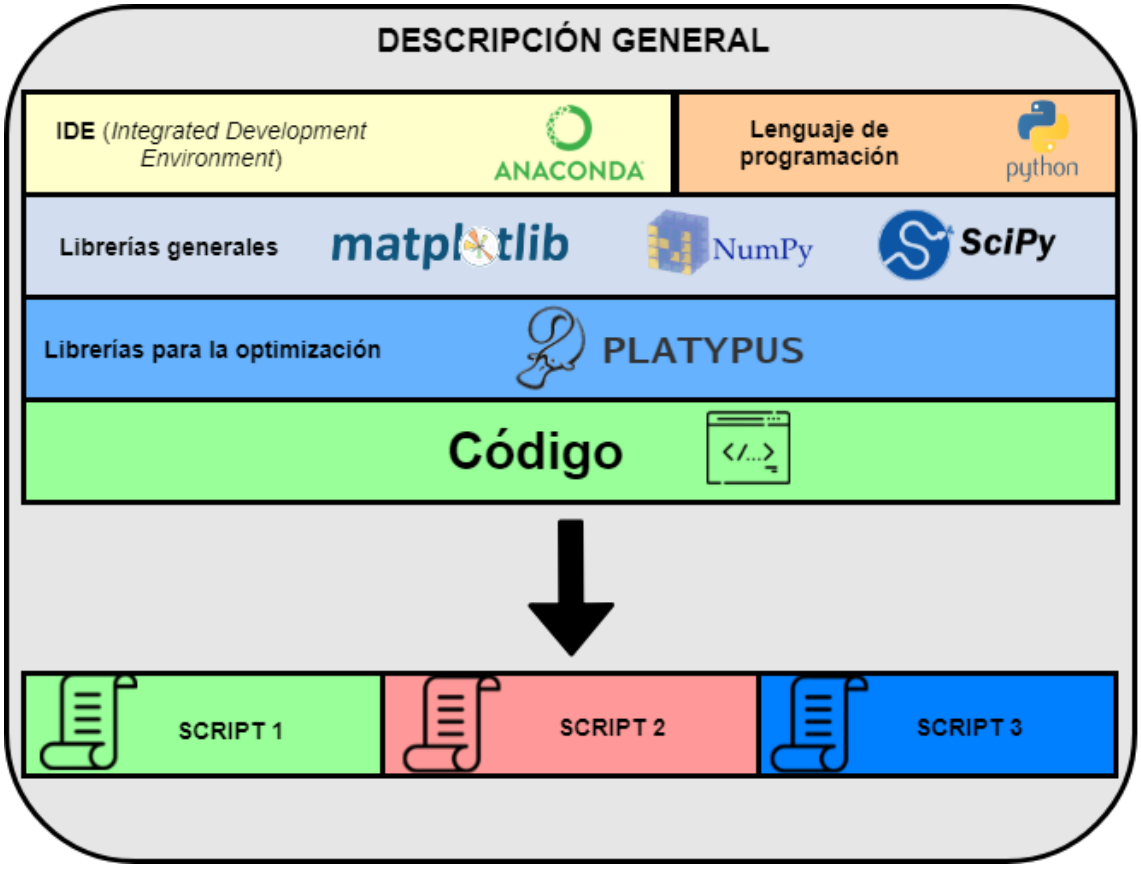

Ilustración 9. Descripción general de la solución

6.3.1 Descripción general de la solución

El proyecto está desarrollado íntegramente en Python y es de carácter puramente software. Como se puede apreciar en la Ilustración 9, la base del proyecto está desarrollado con Python y con el IDE Anaconda, el cual incluye una gran cantidad de librerías como:

- SciPy: librería de código abierto con una gran cantidad de herramientas para la representación de gráficas y realización de operaciones matemáticas complejas.

  Los principales paquetes de los que se hace uso en este proyecto son:

  - Matplotlib: es un paquete de Python que permite dibujar figuras en 2D en una amplia variedad de formatos, además de permitir ejecutar de forma interactiva las gráficas. Otro punto interesante es que permite

incorporar de forma sencilla estas gráficas a scripts de Python sin necesidad de añadir código extra.

Se puede generar entre otras cosas, gráficas, histogramas, scatterplots…

- NumPy: es el paquete científico fundamental para procesamiento de datos en Python. Entre otras cosas se utiliza en:
  - Operaciones con matrices N-dimensionales.
  - Generación y manejo de funciones predefinidas.
  - Algebra lineal, operaciones con transformadas de Fourier y generación de datos aleatorios.

Además de estas librerías, en el proyecto ha sido necesario el uso de una librería para la optimización de resultados, siendo esta:

- Platypus: es un framework de computación evolucionaria en Python, que se centra en la resolución de problemas multi-objetivo mediante el uso de algoritmos evolucionarios (MOEAs). Se diferencia de otras librerías de optimización existentes, por ejemplo: PyGMO, Inspyred, DEAP y Scipy, en que provee de los algoritmos de optimización y de herramientas para el análisis de los resultados. Actualmente soporta la mayoría de algoritmos existentes: NSGA-II, NSGA-III, MOEA/D, IBEA, Epsilon-MOEA, SPEA2, GDE3, OMOPSO, SMPSO, y Epsilon-NSGA-I

En la Ilustración 9, también se puede observar una parte que hace referencia al código. Como metodología seguida, se ha optado por desarrollar un código genérico a partir del cual (realizando pequeñas modificaciones) se puedan obtener 3 scripts (mirar anexos) que representen dentro de los objetivos planteados distintas situaciones:

- Un script que muestre los resultados obtenidos de minimizar al máximo el coste del despliegue.
- Un script que muestre los resultados obtenidos de maximizar el área de cobertura del despliegue (es decir reducir el riesgo al máximo).
- Un script que muestre resultados intermedios entre coste y área cubierta.

6.3.2 Descripción de los módulos

A continuación se procede a explicar brevemente la estructura y los módulos en los que se divide el código, para más detalles de la planificación de estos mirar [Apartado 8]:

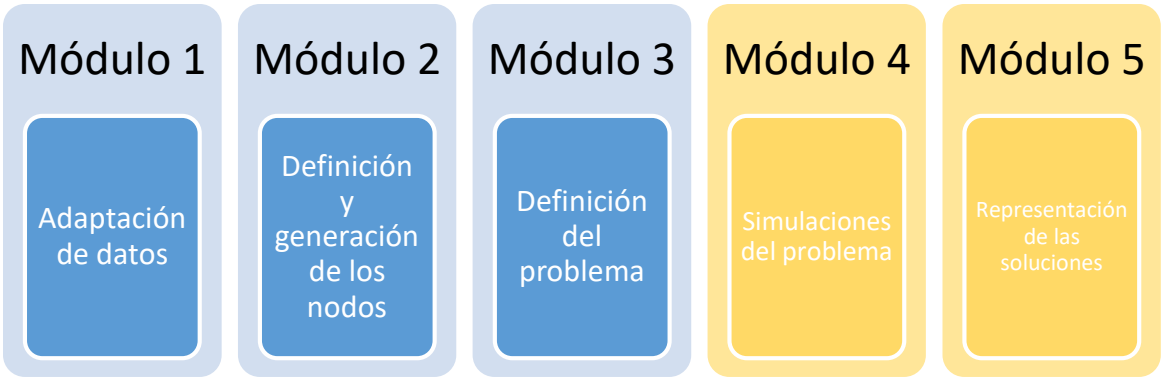

Ilustración 10. Módulos del proyecto

- **Módulo 1 – Adaptación de datos**: para este problema, es necesario el tratamiento de los datos recibidos del software predictor [Apartado 1] ya que el formato es diferente. Además también se pueden leer estos datos de un dataset y luego transformarlos para su uso en la definición del problema, siendo esta la forma elegida en el proyecto. Para más detalles mirar el Anexo I: Código Script 1, 2, 3 - Adaptar datos del dataset.

- **Módulo 2 – Definición y generación de los nodos**: es necesario indicar las características [Ilustración 7] que tendrán los diferentes nodos que se desplegaran, además de indicar cuantos nodos de cada tipo se quieren disponer en el área. Para más detalles mirar el Anexo II: Código Script 1, 2, 3– Generar nodos desplegados.

- **Módulo 3 – Definición del problema**: es necesario definir el número de variables que tiene que optimizar el algoritmo (en este caso un módulo que representa la distancia recorrida por el nodo y un argumento que indica en qué dirección se desplaza). Además se tienen que definir los objetivos a optimizar, en este caso 2, máxima área cubierta y mínimo coste. Para más información mirar el Anexo III: Código Script 1, 2, 3 – Definir problema multi-objetivo.

- **Módulo 4 – Simulaciones del problema**: tras definir el problema es necesario realizar las simulaciones suponiendo diferentes situaciones (por ejemplo buscando minimizar el coste, buscando soluciones intermedias o buscando maximizar el área). Además de comprobar cuál es el algoritmo genético que mejor se adapta al problema (NSGAII, SMPSO, MOEAD). Para más información mirar Anexo IV: Código Script 1 – Obtener soluciones del problema, Anexo VI: Código Script 2 – Obtener soluciones del problema y Anexo VIII: Código Script 3 – Obtener soluciones del problema

- **Módulo 5 – Representación de las soluciones**: finalmente con los datos de las simulaciones se obtienen diferentes indicadores gráficos que permiten analizar las soluciones. Para más información mirar Anexo V: Código Script 1 – Representar soluciones del problema multi-objetivo, Anexo VII: Código Script 2

– Representar soluciones del problema multi-objetivo y Anexo IX: Código Script 3 – Representar soluciones del problema multi-objetivo.

6.3.3 Herramientas para el análisis de los resultados

Para el análisis de los resultados se utiliza:

- Una representación 3D de todas las soluciones (es decir para cada slot se dibujan todos los puntos que se han obtenido de solución) pudiéndose observar los frentes de cada slot.

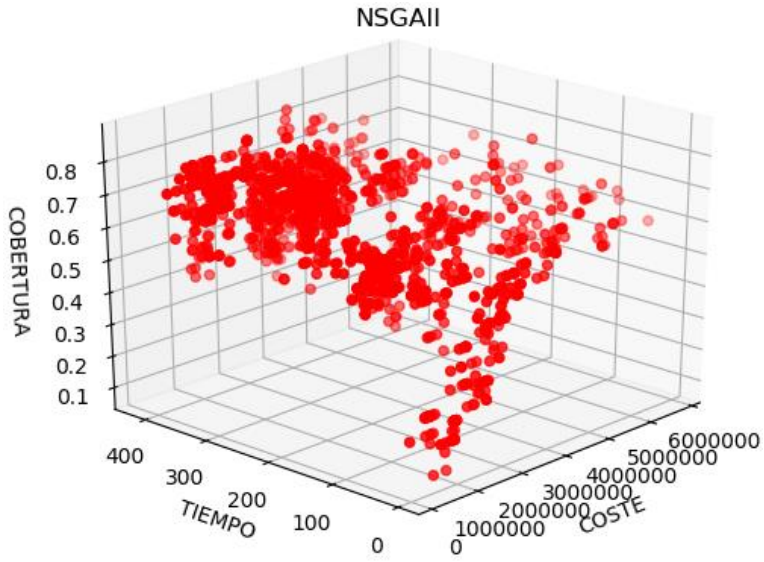

Ilustración 11. Ejemplo de la representación 3D

- Una representación 2D de todos los puntos, pudiéndose así observar que puntos dominan a otros.

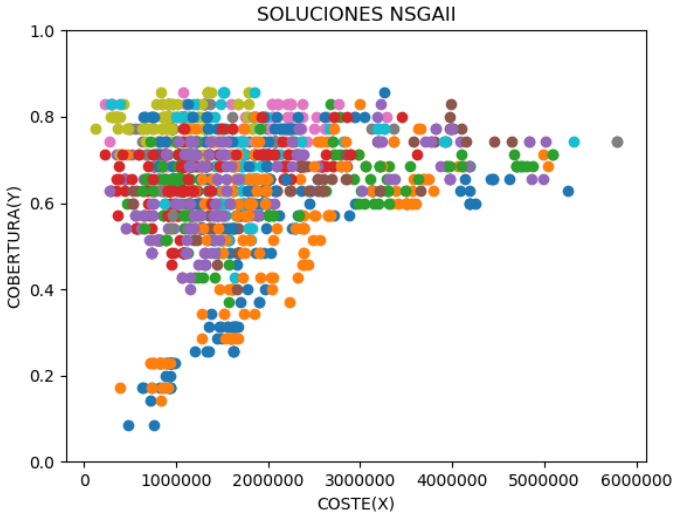

Ilustración 12. Ejemplo de la representación 2D

- La trayectoria seguida por los nodos desplegados y su mapa de cobertura en el instante en que se está cubriendo al máximo número de equipos. En rojo y como cruces se representan las posiciones de los equipos desplegados a lo largo del tiempo, en gris y con líneas discontinuas se representan las trayectorias seguidas por los nodos desplegados.

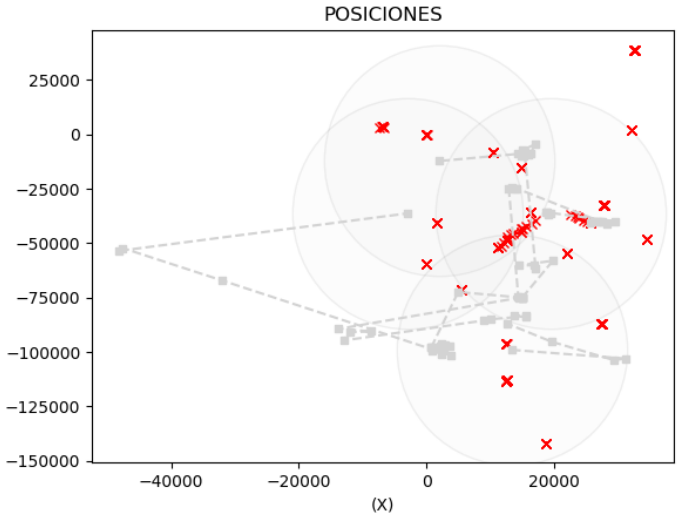

Ilustración 13. Ejemplo de la trayectoria de los nodos

- Y el hypervolumen, el cual es un indicador a lo largo del tiempo, de cuanta área del espacio total está cubierta por ese frente. El objetivo de este indicador es indicar el frente que maximiza su valor [7]. Es decir:

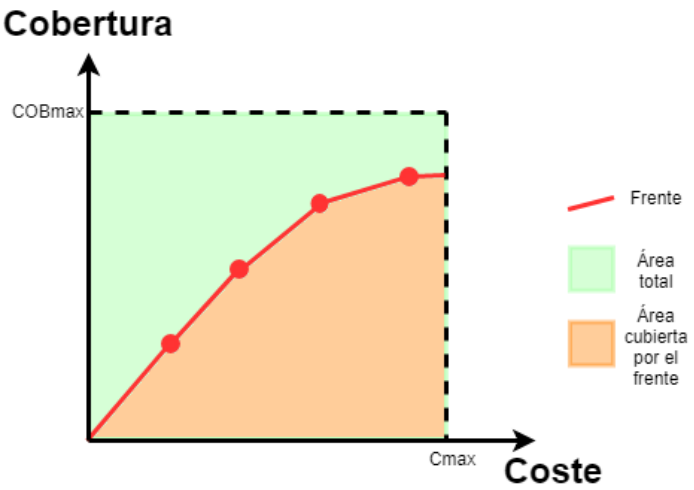

Ilustración 14. Explicación hypervolumen

Y de forma matemática se calcula:

$$HYPsol = \frac{Área\ CUBIERTA}{Área\ TOTAL}$$

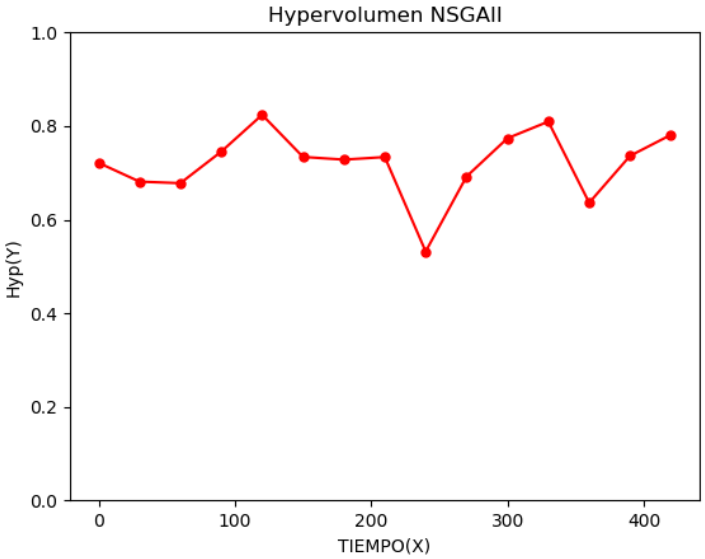

Ilustración 15. Ejemplo Hypervolumen

6.4 Descripción de los resultados

En caso de no indicarse lo contrario, en las simulaciones se han utilizado los siguientes datos iniciales como referencia:

- Número de nodos desplegados:

Tabla 3. Número de nodos utilizados en la simulación

Tipo de nodo	Número
Dron	1
Persona	5
Coche	2
Total	**8**

- Características de los nodos:

Tabla 4. Características de los nodos simulados

Tipo de nodo	Radio de cobertura (unidades)	Radio de movimiento (unidades/s)	Coste (€/unidad)	Posición inicial (x,y)
Dron	20000	1950	40	15000,-75000
Persona	5000	1250	10	15000,-75000
Coche	15000	1400	3	15000,-75000

Y se han realizado las siguientes suposiciones:

- El tiempo que pueden moverse los nodos se divide en slots, en las simulaciones se utilizan 15 slots.
- La duración de cada slot es de 30 segundos, es decir los nodos pueden moverse en intervalos de 30 segundos (la duración total de la simulación es de 450 segundos). Esto es así por cómo se han muestreado los datos del dataset, siendo su periodo de actualización de 30 segundos.
- Para facilitar las simulaciones se indica el número total de equipos al comienzo de la misma
- Los datos del dataset no están normalizado, por ello las características indicadas en la Tabla 4 son 'unidades'.
- La posición inicial de los equipos se han elegido en función de en qué zonas se encuentra la mayor cantidad de nodos, para que con el tiempo total disponible se pueda realizar la simulación.

6.4.1 Resultados simulación 1

Tabla 5. Datos de la simulación 1

Script	1
Objetivo	Mínimo coste
Algoritmo	NSGAII

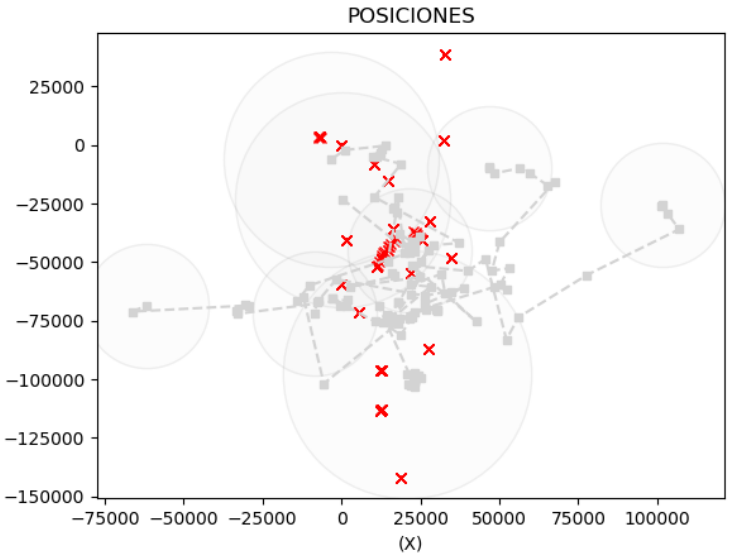

Ilustración 16. Simulación 1: Rutas de los nodos

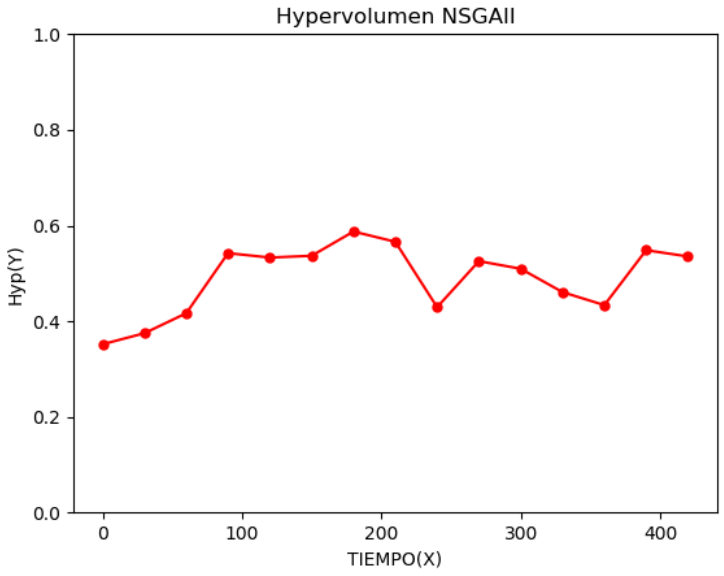

Ilustración 17. Simulación 1: Hypervolumen

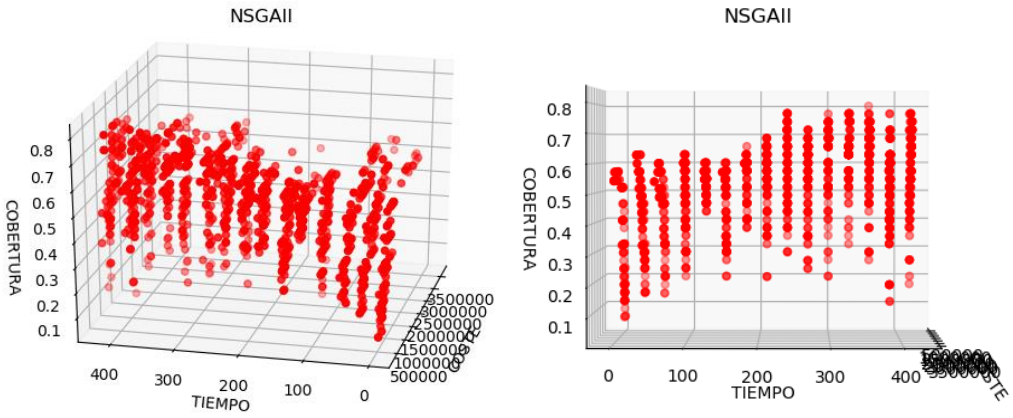

Ilustración 18. Simulación 1: Representación 3D de las soluciones

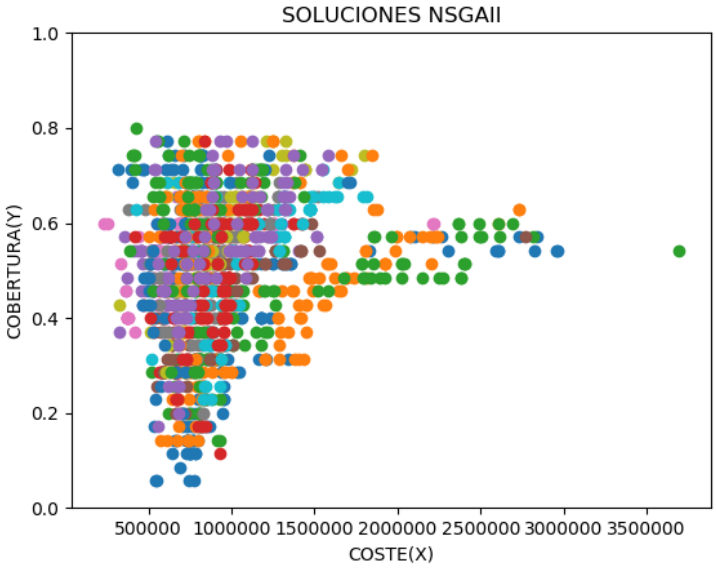

Ilustración 19. Simulación 1: Representación 2D de las soluciones

6.4.2 Resultados simulación 2

Tabla 6. Datos de la simulación 2

Script	1
Objetivo	Mínimo coste
Algoritmo	SMPSO

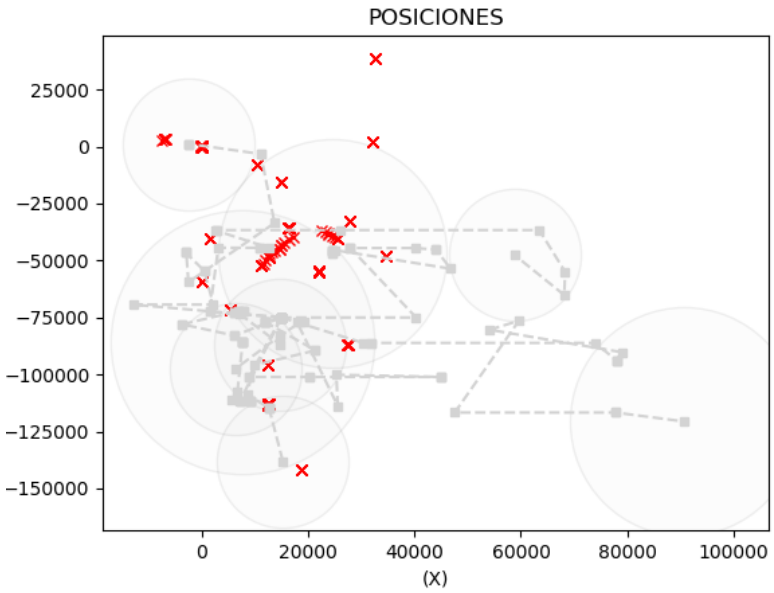

Ilustración 20. Simulación 2: Rutas de los nodos

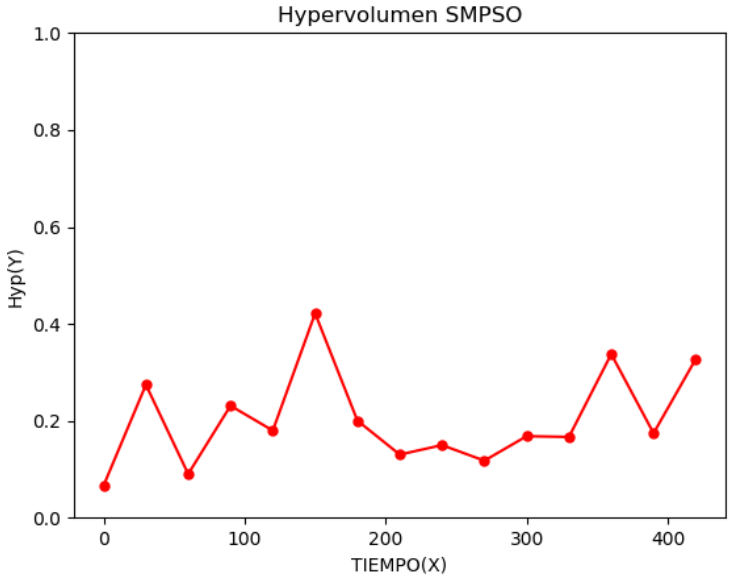

Ilustración 21. Simulación 2: Hypervolumen

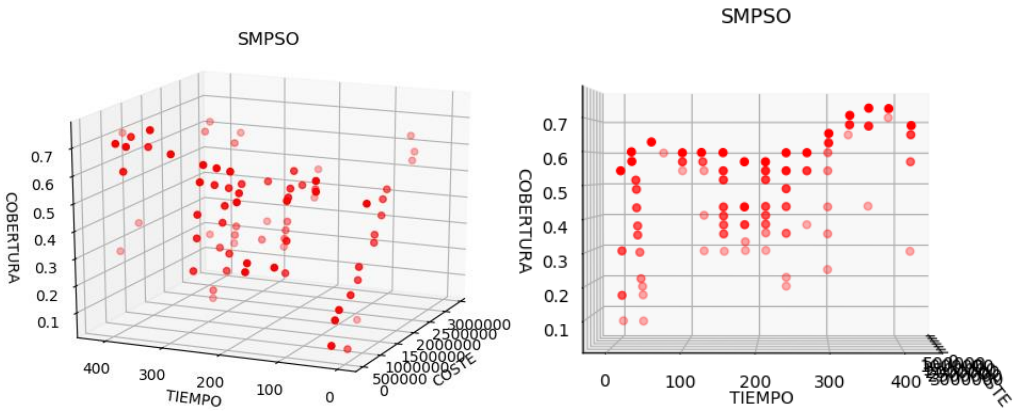

Ilustración 22.Simulación 2: Representación 3D de las soluciones

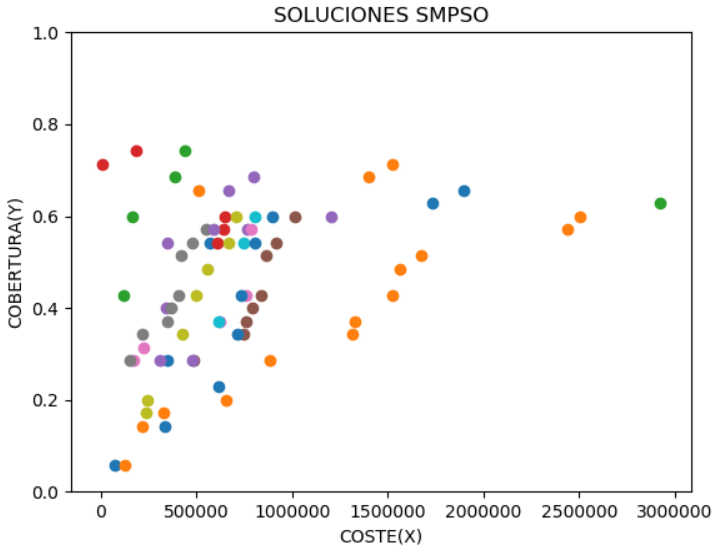

Ilustración 23. Simulación 2: Representación 2D de las soluciones

6.4.3 Resultados simulación 3

Tabla 7. Datos de la simulación 3

Script	1
Objetivo	Mínimo coste
Algoritmo	MOEAD

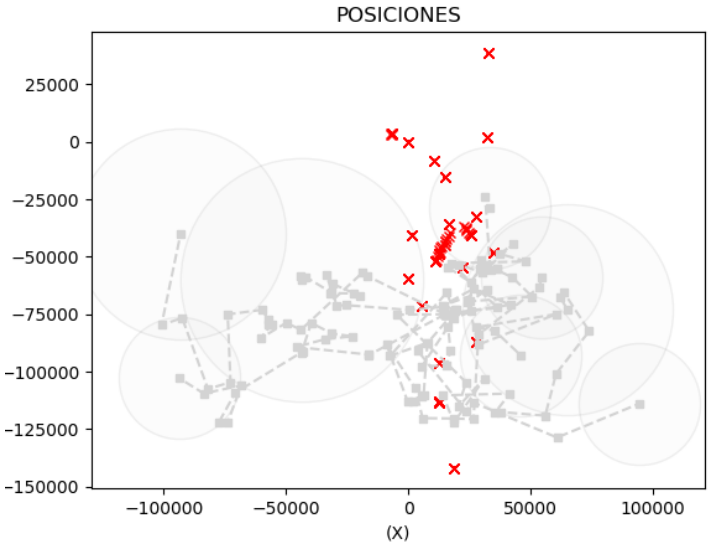

Ilustración 24. Simulación 3: Rutas de los nodos

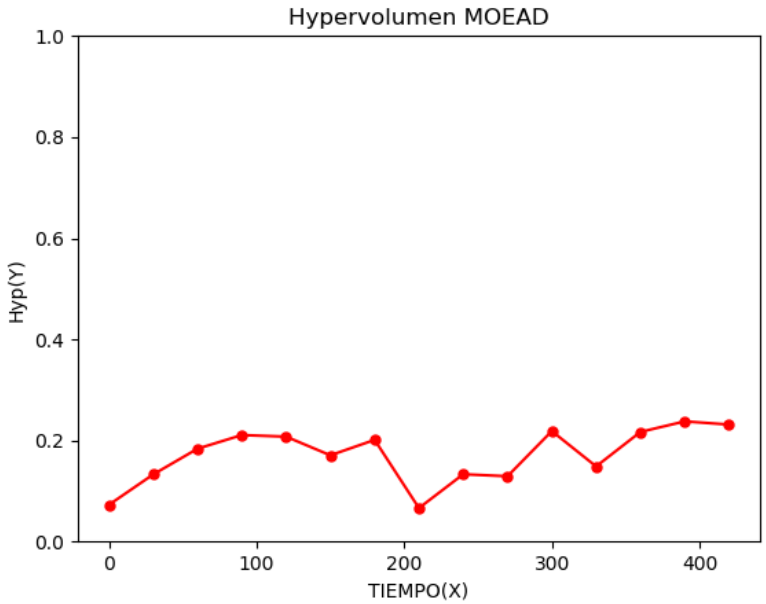

Ilustración 25. Simulación 3: Hypervolumen

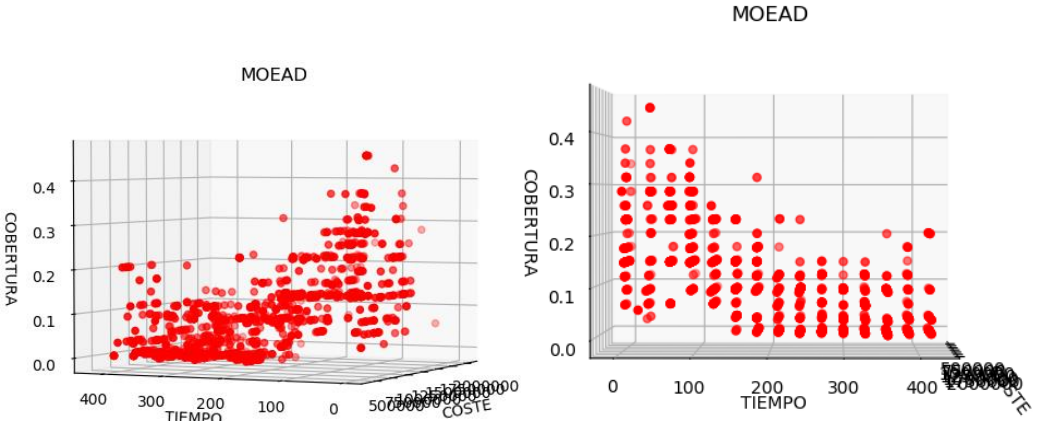

Ilustración 26. Simulación 3: Representación 3D de las soluciones

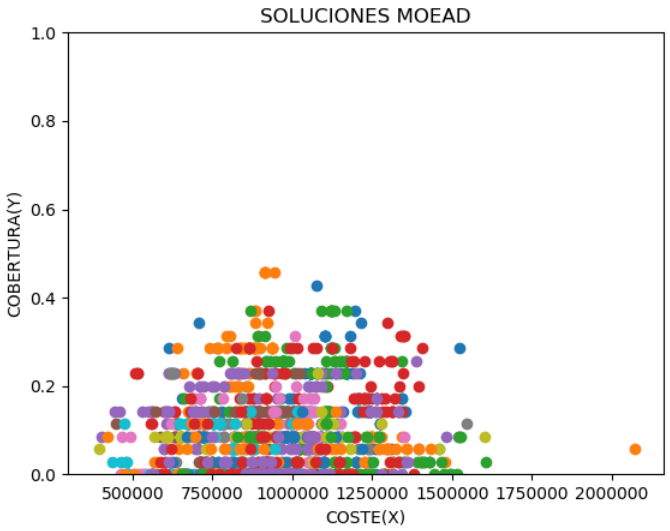

Ilustración 27. Simulación 3: Representación 2D de las soluciones

6.4.4 Análisis de los resultados: Script 1

Tabla 8. Resultados Script 1

	Cumple el objetivo	Converge rápido	Cobertura máxima conseguida
Simulación 1	Sí	No	0.8
Simulación 2	Sí	Sí	0.8
Simulación 3	A medias	No	0.4

El script cumple su objetivo en 2 de las 3 simulaciones, en el caso de NSGAII mirando la Ilustración 19, se puede ver que los costes medios de mover un nodo se mantienen entre 500000 y 1500000, es decir cada slot que se mueve un equipo su coste está en ese intervalo.

En el SMPSO, el coste se mantiene entre 500000 y 1000000 como vemos en Ilustración 23, siendo el que menor coste tiene de las 3 simulaciones.

En el caso de MOEAD los resultados de coste son parecidos a NSGAII (Ilustración 27).

En el caso de la cobertura tanto NSAGII como SMPSO cumplen consiguiendo ofrecer un índice muy alto, en cambio MOEAD solamente llega a 0.4.

Finalmente en cuanto a la convergencia, el SMPSO es mucho más rápido, consiguiendo resultado similares a los obtenidos por NSGAII pero en menor tiempo.

Por lo tanto, en esta batería de simulaciones, la mejor opción para el objetivo de minimizar el coste sería el uso del algoritmo SMPSO o NSGAII, los cuales proporcionan la mejor cobertura con el menor coste. El MOEAD se descarta ya que en coste tiene resultados similares, pero en cobertura es bastante inferior al resto.

6.4.5 Resultados simulación 4

Tabla 9. Datos de la simulación 4

Script	2
Objetivo	Equilibrio
Algoritmo	NSGAII

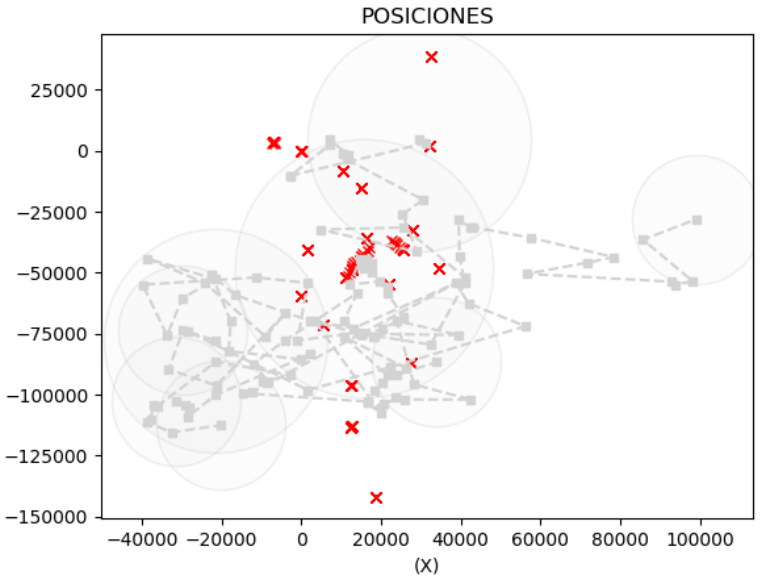

Ilustración 28. Simulación 4: Rutas de los nodos

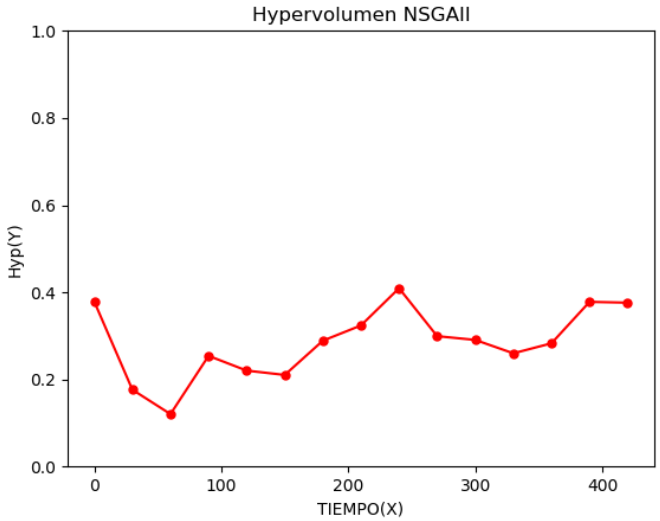

Ilustración 29. Simulación 4: Hypervolumen

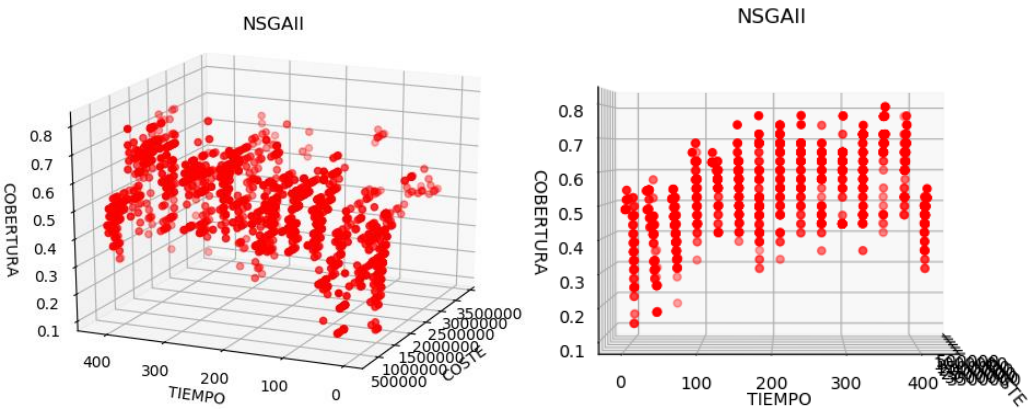

Ilustración 30. Simulación 4: Representación 3D de las soluciones

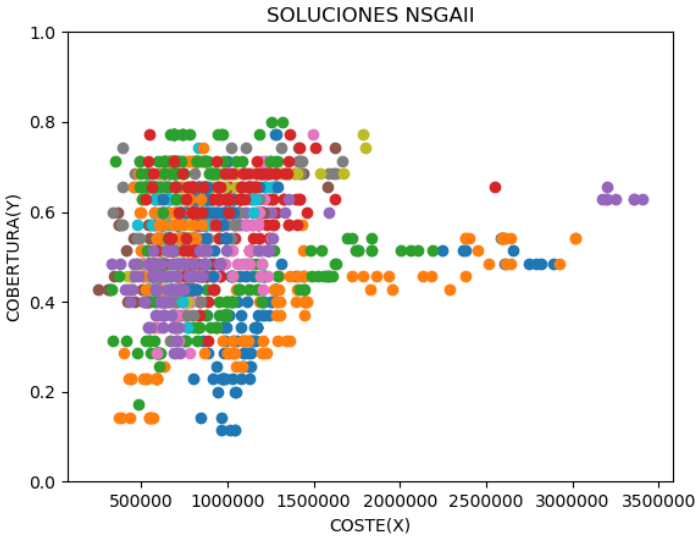

Ilustración 31. Simulación 4: Representación 2D de las soluciones

6.4.6 Resultados simulación 5

Tabla 10. Datos de la simulación 5

Script	2
Objetivo	Equilibrio
Algoritmo	SMPSO

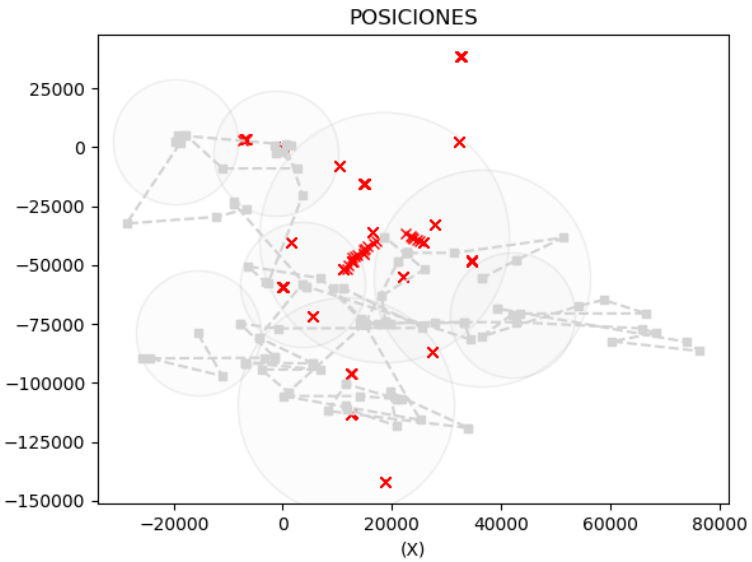

Ilustración 32. Simulación 5: Rutas de los nodos

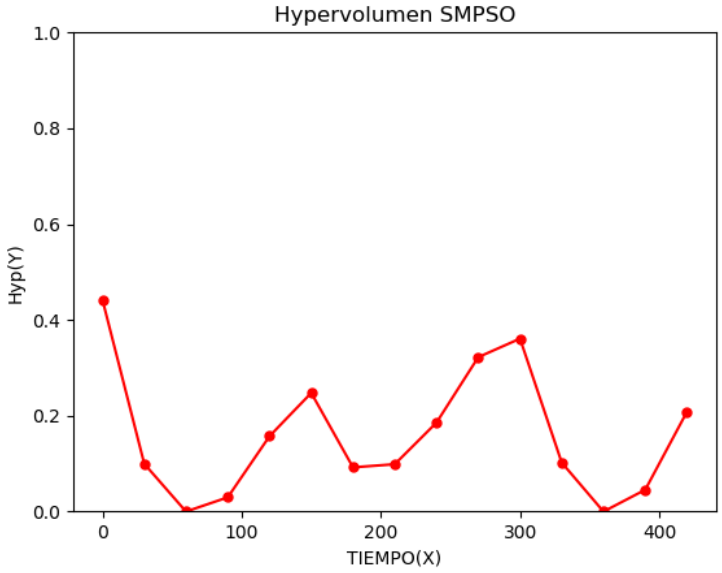

Ilustración 33. Simulación 5: Hypervolumen

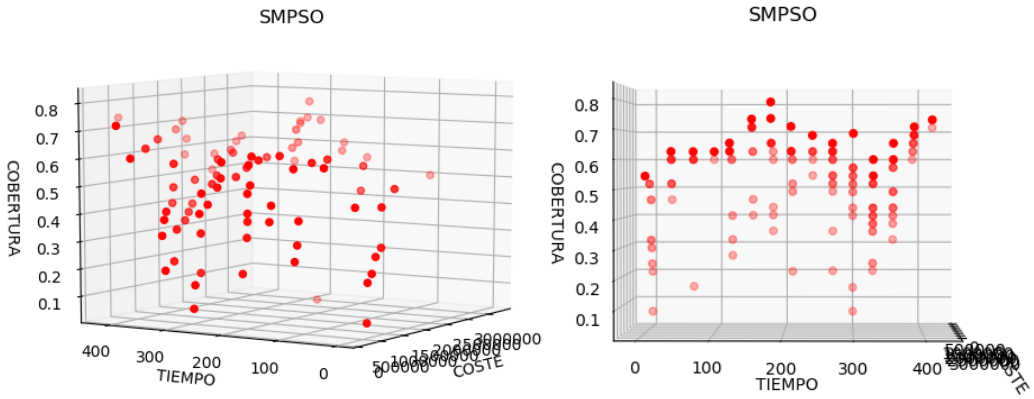

Ilustración 34. Simulación 5: Representación 3D de las soluciones

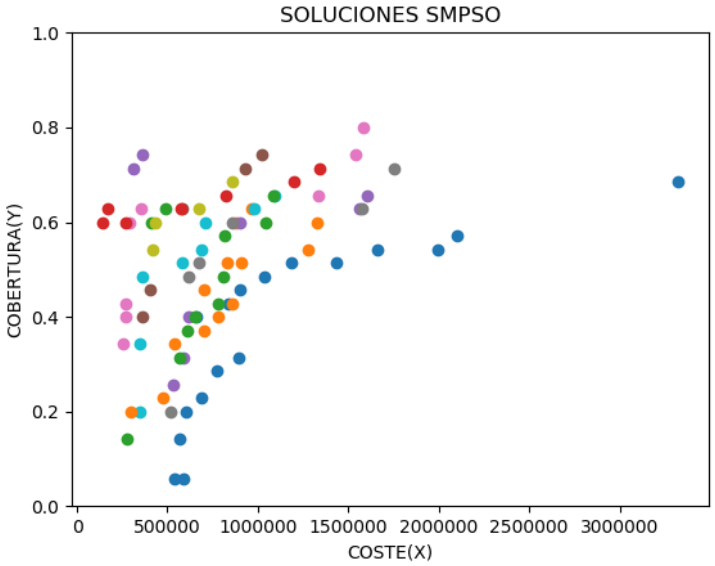

Ilustración 35. Simulación 5: Representación 2D de las soluciones

6.4.7 Resultado simulación 6

Tabla 11. Datos de la simulación 6

Script	2
Objetivo	Equilibrio
Algoritmo	MOEAD

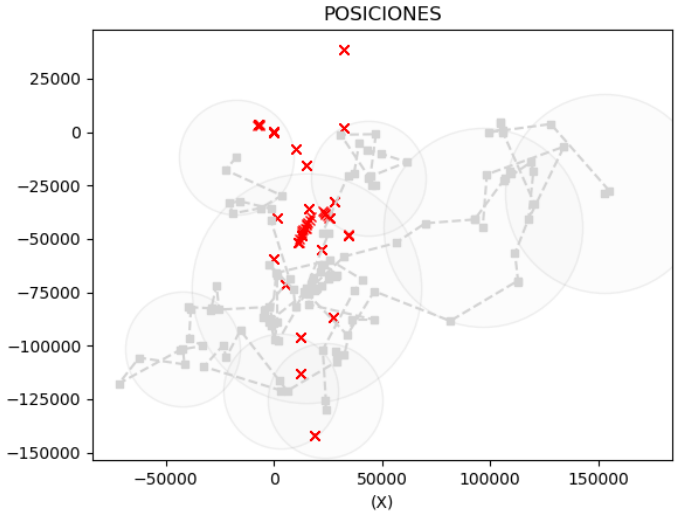

Ilustración 36. Simulación 6: Rutas de los nodos

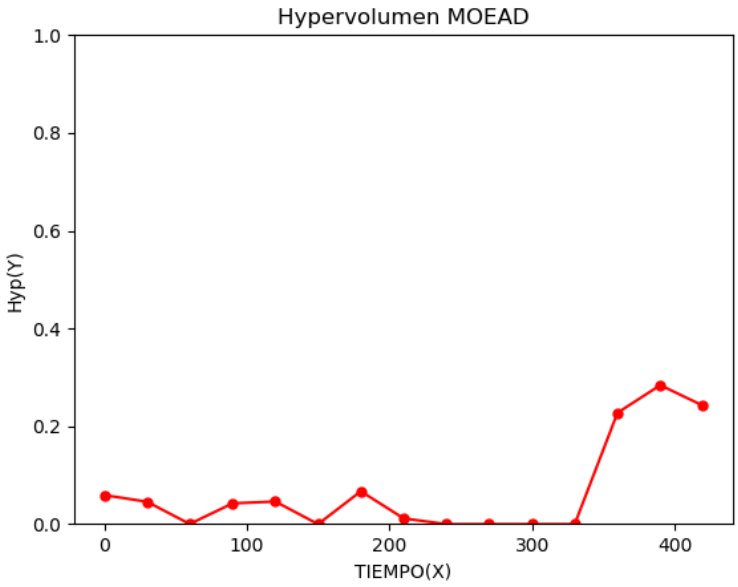

Ilustración 37. Simulación 6: Hypervolumen

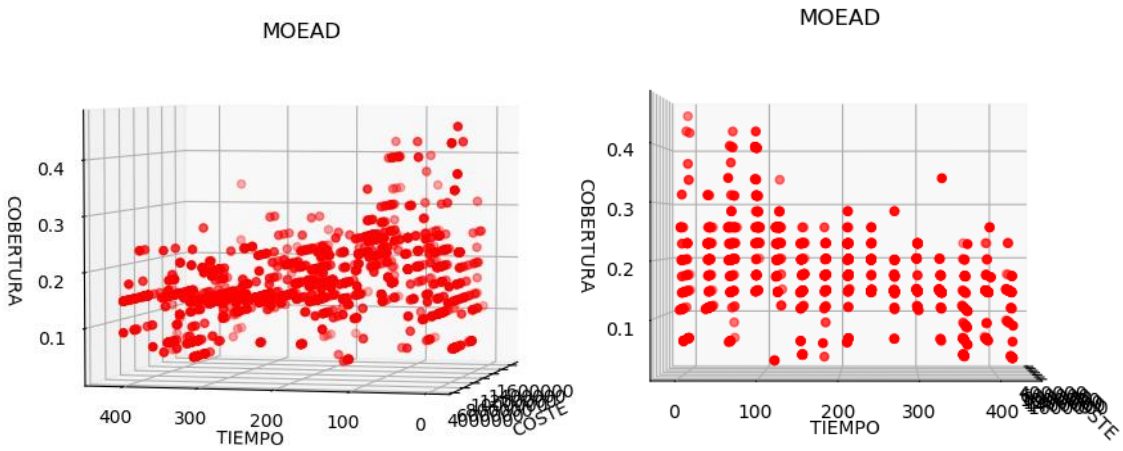

Ilustración 38. Simulación 6: Representación 3D de las soluciones

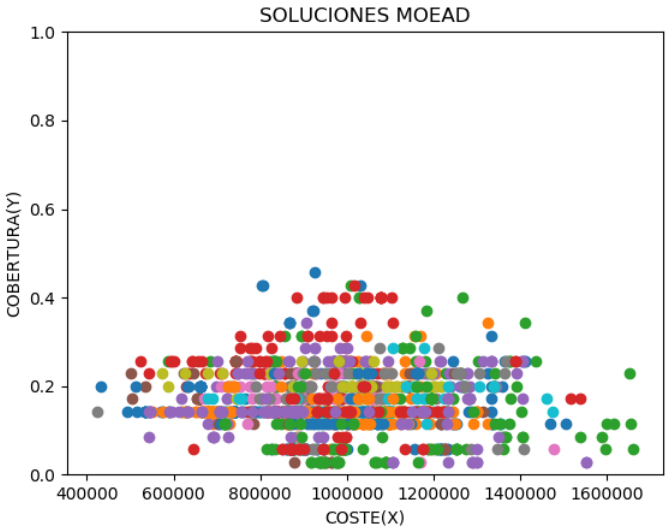

Ilustración 39. Simulación 6: Representación 2D de las soluciones

6.4.8 Análisis de los resultados: Script 2

Tabla 12. Resultado Script 2

	Cumple el objetivo	Converge rápido	Cobertura máxima conseguida
Simulación 4	Sí	No	0.8
Simulación 5	Sí	Sí	0.8
Simulación 6	No	Sí	0.4

El script cumple su objetivo en 2 de las 3 simulaciones, en el caso de NSGAII mirando la Ilustración 31, se puede ver que los costes medios de mover un nodo se mantienen entre 500000 y 1500000, en este caso están menos concentrados, por lo tanto hay más variedad de costes. Esto se debe s a que en este caso se busca una solución equilibrada entre coste y cobertura.

En el SMPSO, el coste se mantiene entre 500000 y 1000000 como vemos en Ilustración 35.

En el caso de MOEAD los resultados de coste son parecidos a NSGAII (Ilustración 39).

En el caso de la cobertura tanto NSAGII como SMPSO cumplen consiguiendo ofrecer un índice muy alto, en cambio MOEAD solamente llega a 0.4.

Finalmente en cuanto a la convergencia, el SMPSO es mucho más rápido, consiguiendo resultado similares a los obtenidos por NSGAII pero en menor tiempo. Esto se puede ver en las rutas que siguen los nodos a lo largo del tiempo (Ilustración 28).

Por lo tanto, en esta batería de simulaciones, la mejor opción para el objetivo de conseguir resultados equilibrados sería el uso del algoritmo SMPSO o NSGAII, los cuales proporcionan la mejor cobertura con el menor coste. El MOEAD se descarta ya que en coste tiene resultados similares, pero en cobertura es bastante inferior al resto.

6.4.9 Resultado simulación 7

Tabla 13. Datos de la simulación 7

Script	3
Objetivo	Máxima cobertura
Algoritmo	NSGAII

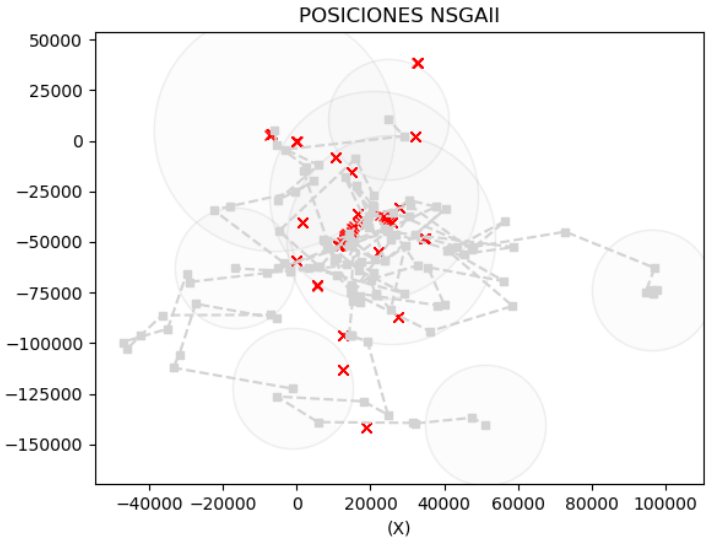

Ilustración 40. Simulación 7: Rutas de los nodos

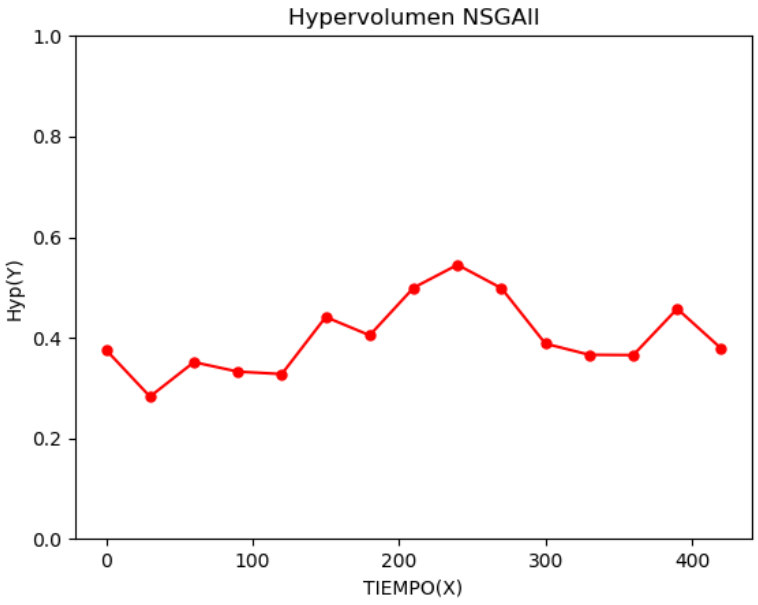

Ilustración 41. Simulación 7: Hypervolumen

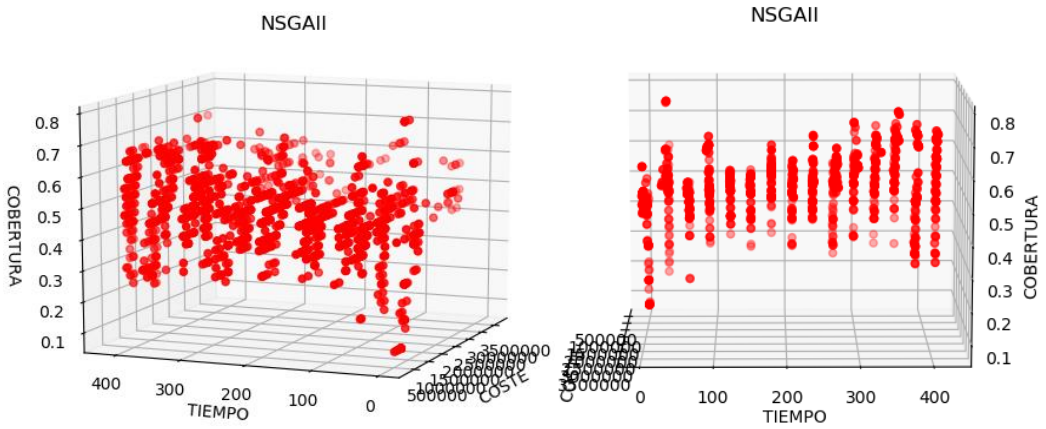

Ilustración 42. Simulación 7: Representación 3D de las soluciones

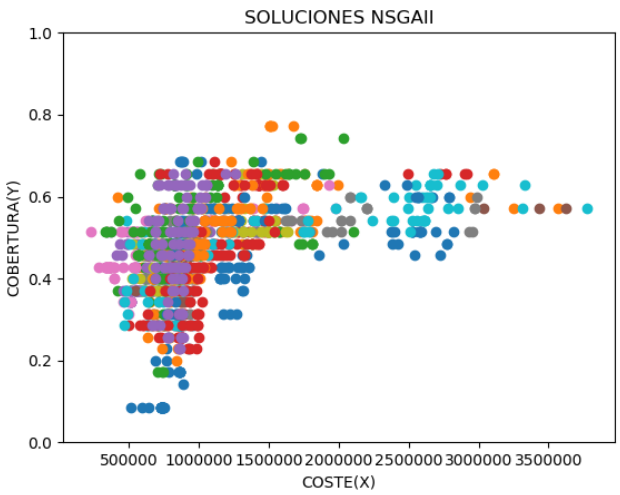

Ilustración 43. Simulación 7: Representación 2D de las soluciones

6.4.10 Resultado simulación 8

Tabla 14. Datos de la simulación 8

Script	3
Objetivo	Máxima cobertura
Algoritmo	SMPSO

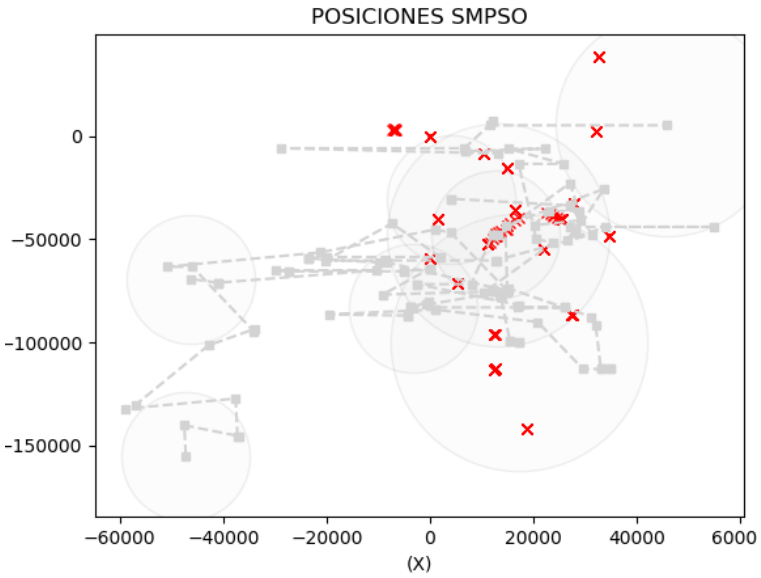

Ilustración 44. Simulación 8: Rutas de los nodos

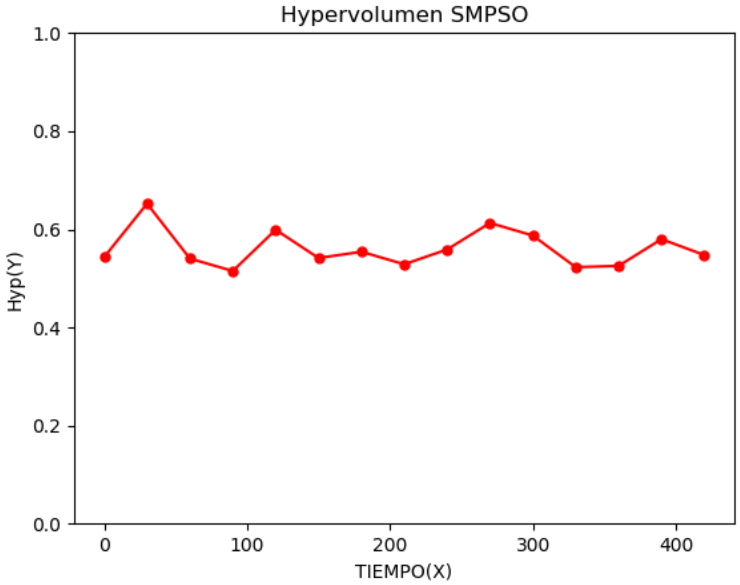

Ilustración 45. Simulación 8: Hypervolumen

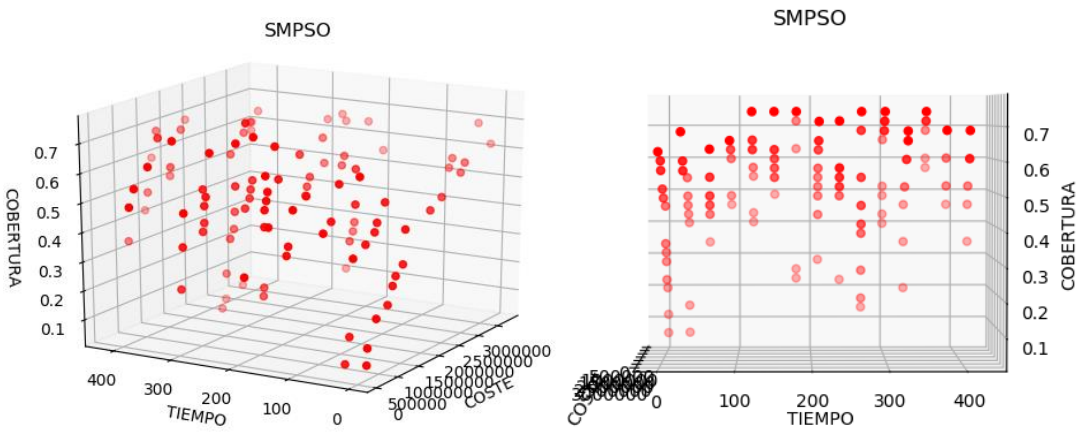

Ilustración 46. Simulación 8: Representación 3D de las soluciones

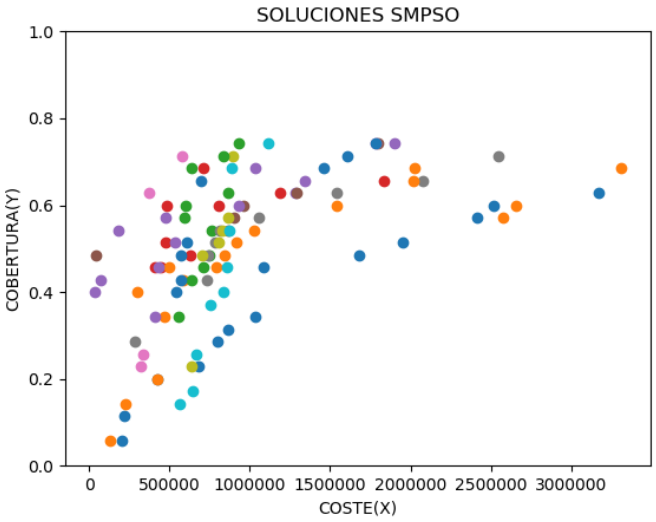

Ilustración 47. Simulación 8: Representación 2D de las soluciones

6.4.11 Resultado simulación 9

Tabla 15. Datos de la simulación 9

Script	3
Objetivo	Máxima cobertura
Algoritmo	MOEAD

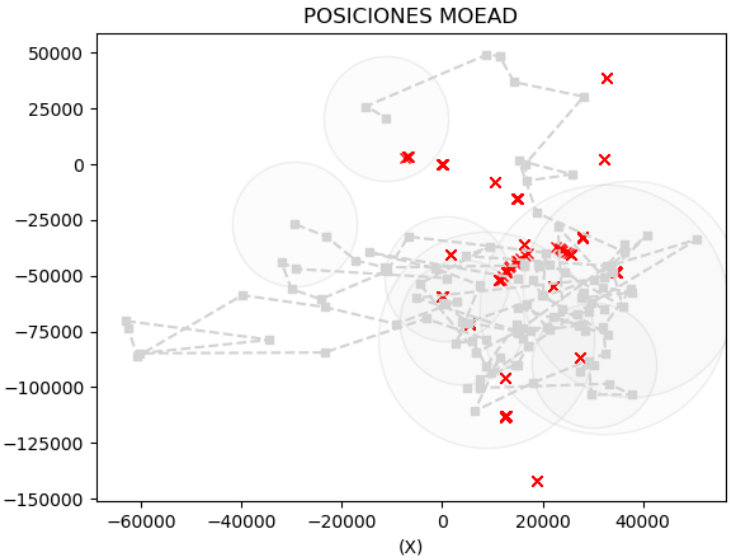

Ilustración 48. Simulación 9: Rutas de los nodos

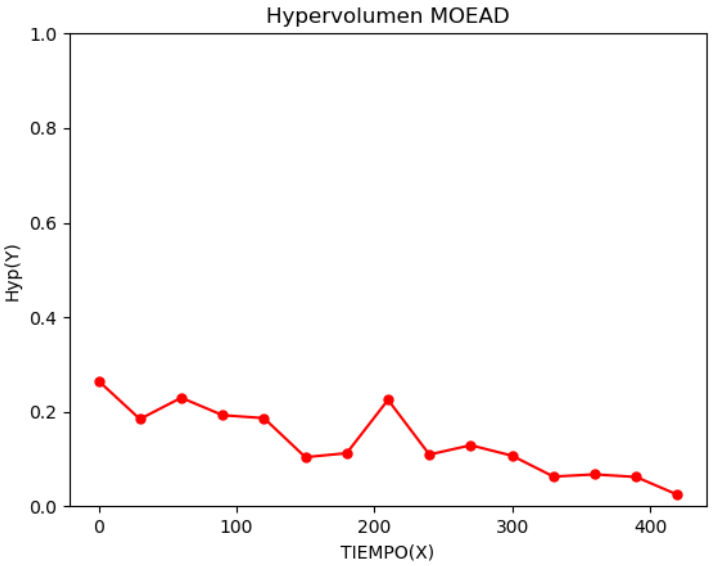

Ilustración 49. Simulación 9: Hypervolumen

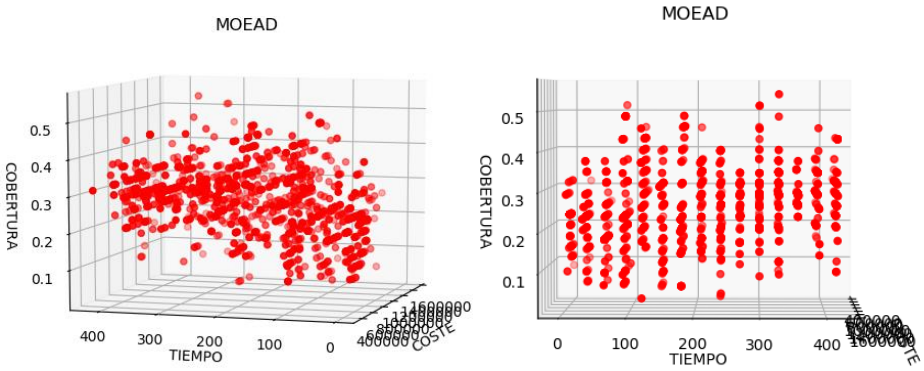

Ilustración 50. Simulación 9: Representación 3D de las soluciones

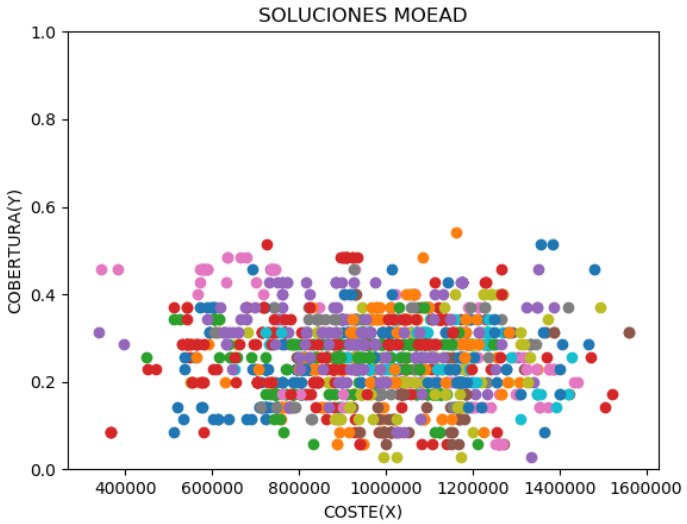

Ilustración 51. Simulación 9: Representación 2D de las soluciones

6.4.12 Análisis de los resultados: Script 3

Tabla 16. Resultados evaluados

	Cumple el objetivo	Converge rápido	Cobertura máxima conseguida
Simulación 7	Sí	No	0.8
Simulación 8	Sí	No	0.8
Simulación 9	No	No	0.4

Los resultados son similares a los obtenidos en el script 2, lo más destacable es en el caso del MOEAD, el cual respecto a las anteriores pruebas tiene un índice ligeramente mayor.

6.4.13 Observaciones adicionales

Para el caso de la cobertura máxima, los resultados obtenidos son parecidos en muchas simulaciones, esto se debe a las posiciones iniciales en las que se han desplegado los nodos, ya que al estar tan centrados respecto a los equipos desplegados, en la primera iteración la cobertura se mantiene y en las posteriores la mejora es mínima.

Esto es necesario porque en caso de colocarlos demasiado lejos de los equipos, con las características actuales no serían capaces de dar cobertura (Ilustración 52) sin variar alguna de las características de los nodos (velocidad de desplazamiento) o de la simulación (número de slots).

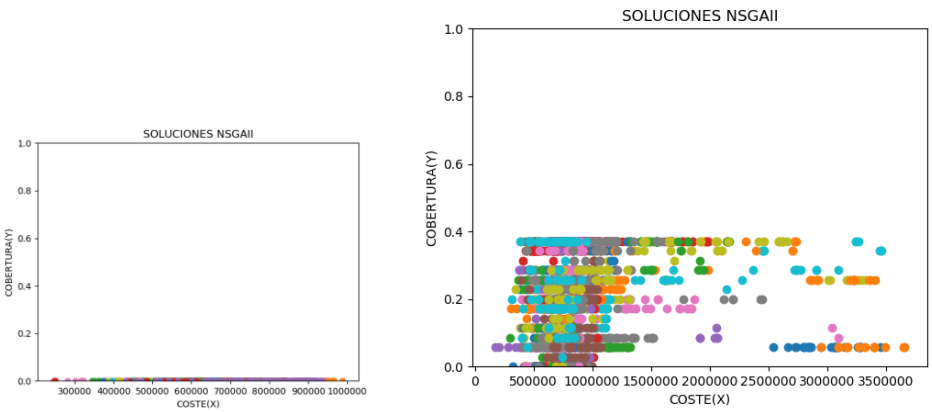

Ilustración 52. Cobertura suponiendo una zona alejada de despliegue (izq.) y el mismo caso variando el número de slots (der.)

En cuanto a la velocidad de convergencia, el algoritmo que cumple en casi todas las simulaciones es el SMPSO, ya que consigue la misma cobertura en una cantidad inferior de pasos (evitando pasos innecesarios).

Por lo tanto en esta implementación del proyecto se optaría por el NSGAII si se busca obtener resultados más precisos pero con una convergencia más lenta, o el SMPSO en caso de buscar una convergencia más rápida con menos pasos redundantes.

7 Análisis de riesgos

En esta parte se evaluaran los peligros potenciales y sus posibles consecuencias en un proyecto, con el objetivo de establecer medidas de prevención y de protección. De esta forma se puede analizar las causas y posibles consecuencias que pueden retrasar o hacer fallar el proyecto.

La metodología consistirá en 3 partes, identificación de los riesgos, análisis de estos y la elaboración de un plan en caso de que se den los mismos.

Para el análisis posterior de los riesgos se usará una herramienta disponible para este propósito, la matriz probabilidad-impacto. Esta herramienta nos permite establecer prioridades en cuanto a posibles riesgos de nuestro proyecto considerando tanto la probabilidad de que ocurran dichos riesgos como los efectos de los mismos en el proyecto.

7.1 Identificación de los riesgos

A continuación se comentan los riesgos que pueden afectar al proyecto:

- Exigencias de los clientes/usuarios
- Errores de diseño
- Superar el presupuesto inicial
- Mala gestión de recursos

7.2 Análisis de los riesgos

PROBABILIDAD	IMPACTO				
	Muy bajo (0, 05)	Bajo (0,1)	Moderado (0,2)	Alto (0,4)	Muy Alto (0,8)
Raramente (0,1)	Bajo 0,005	Bajo 0,01	Bajo 0,02	Moderado 0,04	Moderado 0,08
Difícilmente (0,3)	Bajo 0,015	Bajo 0,03	Moderado 0,06	Moderado 0,12	Alto 0,24
Posible (0,5)	Bajo 0,025	Moderado 0,05	Moderado 0,1	Alto 0,2	Alto 0,4
Probable (0,7)	Bajo 0,035	Moderado 0,07	Moderado 0,14	Alto 0,28	Alto 0,56
Casi seguro (0,9)	Moderado 0,045	Moderado 0,09	Alto 0,18	Alto 0,36	Alto 0,72

Ilustración 53. Matriz Probabilidad – Impacto

Tabla 17. Tabla de riesgos del proyecto

	Riesgos	Probabilidad	Impacto	Resultado
A	Exigencias de los clientes/usuarios	0.1 (Raramente)	0.2 (Moderado)	0.02 (Bajo)
B	Errores de diseño	0.5 (Posible)	0.2 (Moderado)	0.1 (Moderado)
C	Superar el presupuesto inicial	0.7 (Probable)	0.4 (Alto)	0.25 (Alta)
D	Mala gestión de recursos	0.1 (Raramente)	0.4 (Alto)	0.04 (Moderado)

- Exigencia de los clientes/usuarios (A):
    - **Riesgo**: los clientes piden explicaciones de forma constante y con unos plazos muy ajustados.
    - **Probabilidad**: la probabilidad de que ocurra este riesgo es muy rara.
    - **Impacto**: este riesgo tendría un impacto moderado en el proyecto.
- Errores de diseño (B):
    - **Riesgo**: se codifica de forma errónea alguno de los módulos del programa, lo cual puede provocar error a posteriori en otra parte del código.
    - **Probabilidad**: la probabilidad de que ocurra este riesgo es moderada.
    - **Impacto**: este riesgo tendría un impacto medio en el proyecto.
- Superar el presupuesto inicial (C):
    - **Riesgo**: según avanza el proyecto, la organización se da cuenta de que se va a superar la inversión inicial.
    - **Probabilidad**: la probabilidad de que ocurra este riesgo es probable.
    - **Impacto**: este riesgo tendría un impacto alto en el proyecto.
- Mala gestión de recursos (D):
    - **Riesgo**: durante el desarrollo del proyecto se realiza una mala gestión de los recursos de la empresa y se producen perdidas.
    - **Probabilidad**: la probabilidad de que ocurra este riesgo es rara.
    - **Impacto**: este riesgo tendría un impacto alto en el proyecto.

7.3 Plan de contingencia

Para poder realizar el plan de contingencia es necesario ordenar los riesgos en función de su necesidad de atención: C→B→D→A

Para evitar el riesgo C (Superar el presupuesto inicial) la forma más sencilla de evitarlo es añadir una partida de imprevistos en la partida final. Además de esto, se puede reforzar si se incluyen suficientes medios para revisar el código y así evitar errores posteriores.

Para evitar el riesgo B (Errores de diseño) se pueden realizar revisiones exhaustivas de la planificación de módulos antes de comenzar el desarrollo. Una vez comenzado el desarrollo, conviene realizar revisiones periódicas para comprobar el estado y detectar posibles fallos.

Para evitar el riesgo D (Mala gestión de recursos) bastaría con llevar un mayor control del libro diario de las cuentas, para de esa forma evitar que se realicen gastos injustificados.

Para evitar el riego A (Exigencia de los clientes/usuarios) es recomendable acordar los criterios y especificaciones desde el inicio del proyecto firmando un contrato con su alcance y los objetivos que tiene que conseguir.

8 Planificación

La planificación es uno de los momentos más importantes del desarrollo de un proyecto. Sin una buena planificación cualquier proyecto por muy bueno que sea andará a la deriva y sin rumbo (y probablemente nunca llegue a ningún resultado ni desarrollo).

A la hora de realizar la planificación de un proyecto se puede elegir entre 2 diagramas, por un lado el diagrama de GANTT y por otro el PERT.

En este proyecto se opta por el primero, ya que al ser sencillo el uso del GANTT es suficiente, en caso de tener una gran complejidad con muchas tareas y una duración muy larga en el tiempo es mejor el uso del PERT.

8.1 Descripción de las tareas

En este caso, las tareas han sido divididas según la arquitectura y los módulos que conforman el software, además de tener una parte relacionada con la creación y revisión de la documentación a entregar.

DIAGRAMA DE GANTT

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		**Estudio inicial del proyecto**	**15 días**	**vie 22/12/17**	**jue 11/01/18**	
2		Eleccion de tema	2 días	vie 22/12/17	lun 25/12/17	
3		Definición objetivos y alcance del proyecto	4 días	mar 26/12/17	vie 29/12/17	2
4		Eleccion de material para la formación	2 días	lun 01/01/18	mar 02/01/18	3
5		Analisis de problemas	3 días	mié 27/12/17	vie 29/12/17	2
6		Desarrollo plan de contingencia	4 días	vie 05/01/18	mié 10/01/18	4;5
7		**Diseño y desarrollo de los modulos del proyecto**	**28 días**	**vie 12/01/18**	**mar 20/02/18**	**1**
8		Analisis de alternativas y eleccion del lenguaje de programación	11 días	mar 16/01/18	mar 30/01/18	1
9		Análisis de alternativas y elección del entorno de desarrollo	8 días	mar 23/01/18	jue 01/02/18	1
10		Diseño documentación de los módulos	9 días	jue 08/02/18	mar 20/02/18	8;9
11		**Implementación de los módulos**	**85 días**	**mié 21/02/18**	**mar 19/06/18**	**7**
12		Pruebas y elección de algoritmos bioinspirados	5 días	mié 21/02/18	mar 27/02/18	7
13		Implementacion del módulo de optimización	63 días	mié 21/02/18	vie 18/05/18	7
14		Pruebas finales del programa	17 días	lun 28/05/18	mar 19/06/18	13
15		**Desarrollo de la documentación**	**25 días**	**mié 20/06/18**	**mar 24/07/18**	**11**
16		Desarrollo del informe final	16 días	mié 20/06/18	mié 11/07/18	
17		Revisión de la documentación a entregar	7 días	jue 12/07/18	dom 22/07/18	16
18		Entrega documentacion	2 días	lun 23/07/18	mar 24/07/18	17

Ilustración 54. Tareas del proyecto

Como podemos ver en la Ilustración 54, se ha dividido en las siguientes tareas el proyecto:

- Estudio inicial del proyecto: en esta tarea se **elige el tema** del proyecto, se consulta, compra o pide la **documentación** necesaria para poder formarse en la

materia. Además se **definen los objetivos** y en función de estos se realiza un plan de contingencia para en caso de que suceda alguno de estos problemas se pueda evitar o disminuir en la medida de lo posible como afecte al proyecto.

- Diseño y desarrollo de los módulos del proyecto: en esta tarea se definen la **documentación de los módulos** que forman parte del proyecto, además se elige entre otras cosas el **lenguaje de programación** en el que se realiza la implementación de los módulos, así como la elección del **entorno de desarrollo** del mismo.

- Implementación de los módulos: esta es la tarea principal dentro de todo el desarrollo, en ella se realiza todo la parte de programación de los módulos y tras su finalización las pruebas necesarias para comprobar si se ha realizado correctamente la implementación.

- Desarrollo de la documentación: finalmente, se realiza y revisa la documentación necesaria relacionada con el proyecto, desde los anexos hasta los resultados de las pruebas y demás partes.

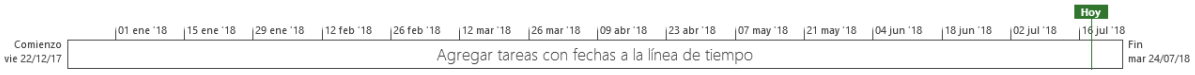

Ilustración 55. Duración total del proyecto

Como vemos en la Ilustración 55 el proyecto empezó el 22 de diciembre de 2017 y finalizo el 24 de julio de 2018, siendo su duración de 153 días.

8.2 Diagrama de GANTT

Se ha optado por utilizar un diagrama de GANTT, la principal labor de esta herramienta es mostrar la duración programada, así como los momentos de inicio y finalización de cada una de las tareas que componen el proyecto. Principalmente está formado por 2 ejes:

- El eje horizontal es un calendario en el que se representan las unidades temporales (puede ser en días, meses, años…)

- El eje vertical representa las tareas a realizar mediante barras horizontales que representan la duración de la tarea.

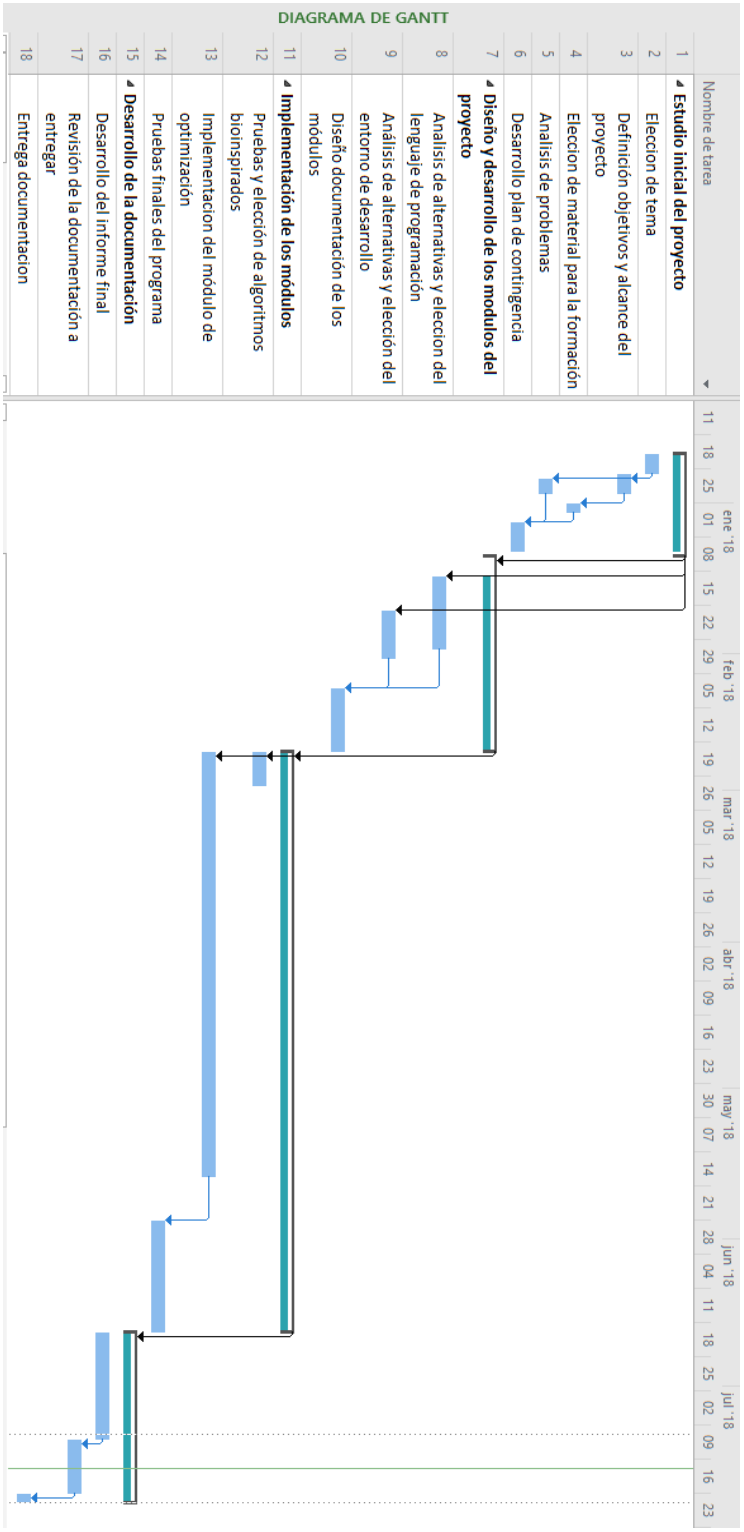

Ilustración 56. Diagrama Gantt

9 Presupuesto y coste

A continuación se incluirá el análisis de los costes del proyecto, tanto del desarrollo técnico como del despliegue del mismo. Debido a esto se incluirán 2 presupuestos diferenciados, es interesante esta diferenciación debido al carácter del proyecto. Al ser un software polivalente que puede ser usado en múltiples situaciones, hace que al disponer de un presupuesto aparte para la implementación permite que sea más sencilla su lectura.

Por un lado, se analizan los medios utilizados (tanto materiales como inmateriales) entre ellos equipos hardware, posibles amortizaciones…

Por otro, se tienen en cuenta diferentes gastos que son básico en cualquier empresa como por ejemplo, sueldos y salarios, alquiler de edificio, imprevistos…

9.1 Presupuesto del desarrollo

Esta parte del proyecto tiene una duración de 8 meses, desde diciembre hasta su finalización a finales de julio. El número de horas invertido se ha calculado en función del valor que tiene el proyecto en créditos ECTS, en este caso 12 créditos que equivalen a 300 horas de trabajo.

9.1.1 Horas internas

El personal de desarrollo de esta parte del proyecto constaría de 1 ingeniero junior el cual se encargaría del desarrollo de la solución y la creación del documento final.

Tabla 18. Presupuesto: Horas internas I

Concepto	Número	Coste horario (€/h)	Tiempo (h)	Coste (€)
Ingeniero de telecomunicaciones junior	1	35	300	10500
Subtotal				10500

9.1.2 Amortizaciones

A continuación, se evaluarán los equipos hardware que se han utilizado para el desarrollo, tanto aquellos que han sido necesario comprar para este proyecto en concreto, como aquellos que ya estaban en uso de otros proyectos.

Tabla 19. Presupuesto: Amortizaciones I

Concepto	Coste inicial (€)	Vida útil (h)	Tiempo de uso (h)	Coste (€)
Ordenador portátil Toshiba Satellite S50-B-151 (i5, 8GB, 240GB)	700	43800	300	4.8
Ordenador de sobremesa 2014 (i5, 6GB, 8TB)	500	43800	50	0.6
Licencia Microsoft Office 2013	100	8760	250	2.85
Subtotal				8.25

9.1.3 Gastos

En la siguiente partida se detallan los gastos generales del proyecto.

Tabla 20. Presupuesto: Gastos I

Concepto	Coste (€)
Abono de transporte público	263
Material para formación	30
Cursos online	10
Subtotal	303

9.1.4 Coste total

Finalmente, tras realizar todas las partidas, estas se agrupan en el resumen final del proyecto y se obtiene el total del mismo.

Tabla 21. Presupuesto: Coste total I

Concepto	Coste (€)
Subtotal Horas Internas	10500
Subtotal Amortizaciones	8.25
Subtotal Gastos	303
Subtotal	10811.25
Imprevistos (6%)	648.675
Total	11459.925

9.2 Presupuesto del despliegue

En esta parte se detallara el despliegue del proyecto, teniendo en cuenta tanto los equipos hardware en los que se implementara, como las pruebas y calibraciones que serán necesarias realizar para comprobar su correcta implementación.

La duración total de esta parte es de 2 meses incluyendo pruebas y formación para el uso del software.

9.2.1 Horas internas

El personal de esta parte constará de 3 ingenieros junior para la implementación y de 2 encargados de la formación.

Tabla 22. Presupuesto: Horas internas II

Concepto	Número	Coste horario (€/h)	Tiempo (h)	Coste (€)
Ingeniero de telecomunicaciones junior	3	35	320	33600
Encargado de la formación	2	20	128	5120
Subtotal				38720

9.2.2 Amortizaciones

A continuación se evaluara el hardware de los equipos necesarios para la implementación del software del proyecto.

Tabla 23. Presupuesto: Amortizaciones II

Concepto	Coste inicial (€)	Vida útil (h)	Tiempo de uso (h)	Coste (€)
Ordenador portátil ToughBook CF-31 (i5, 16GB, 500GB, Resistencia completa a altas temperaturas y situaciones climatológicas adversas)	3200	87600	1000	36.53
Subtotal				36.53

9.2.3 Subcontrataciones

En esta partida se incluirán todos los servicios que impliquen delegar en empresas externas parte de la implementación del programa.

En este caso, las pruebas de error para comprobar si se ha instalado todo correctamente se subcontratan.

Tabla 24. Presupuesto: Subcontrataciones I

Concepto	Coste (€)
Pruebas implementación	5000
Subtotal	5000

9.2.4 Gastos

Los gastos asociados a la implementación son:

Tabla 25. Presupuesto: Gastos II

Concepto	Coste (€)
Material formación	50
Desplazamientos	200
Subtotal	250

9.2.5 Coste total

Finalmente el coste total de esta parte del proyecto sería:

Tabla 26. Presupuesto: Coste total II

Concepto	Coste (€)
Subtotal Horas Internas	38720
Subtotal Amortizaciones	36.53
Subtotal Subcontrataciones	5000
Subtotal Gastos	250
Subtotal	44006.53

Imprevistos (6%)	2640.4
Total	46646.9

10 Conclusiones

En este proyecto se ha realizado un análisis sobre las diferentes familias de algoritmos bioinspirado y del estado del arte de los mismos. Detallando diferentes áreas en las que su uso se está extendiendo de forma muy importante.

Además se ha desarrollado un software optimizador en Python el cual implementa alguno de estos algoritmos (NSGAII, SMPSO, MOEAD) para ser usado en el despliegue de nodos de comunicación en situaciones de emergencia.

Para la elección del algoritmo que se implementará en el software final se han realizado simulaciones que mediante resultados gráficos han permitido ver en qué situaciones es mejor cada uno de los algoritmos previamente indicados.

Entre los futuros desarrollos de este proyecto, destaca la ampliación del estudio a nuevos algoritmos y su implementación conjunta con otro proyecto (Modelo predictivo para la estimación de trayectorias de recursos de vigilancia y contingencia de desastres naturales) para así formar un sistema completo que permita por un lado predecir el movimiento de las tropas y por el otro optimizar el despliegue de los nodos en función de donde vayan a estar los equipos.

Como conclusión final, tras la elaboración de este proyecto se puede deducir que el desarrollo de algoritmos bioinspirado va a tener un importante impulso en el futuro, tanto en el ámbito en el que está enfocado este trabajo (optimización de nodos de comunicación en catástrofes naturales) como en otros ámbitos tecnológicos.

11 Agradecimientos

En este apartado quiero dar las gracias a las personas que me han ayudado en la elaboración de este proyecto.

En primer lugar a **Javier del Ser** por ayudarme y guiarme a lo largo de estos meses, sin su ayuda esto no habría sido posible. Colaborando en todo lo posible, poniendo a mi disposición todos los medios a su alcance y por brindarme la oportunidad de desarrollar e investigar sobre un tema de tanto interés y prestigio.

En segunda instancia me gustaría agradecer a **Miren Nekane** su ayuda en el desarrollo del presente documento, dándome las indicaciones y recomendaciones para su publicación y desarrollo escrito. Ofreciendo en todo momento su disposición y tiempo para corregirme y ayudarme.

Finalmente me gustaría agradecer (por su gran ayuda e inestimable amistad) a mi compañera **Leticia Ortiz**. Siendo su cooperación crucial para la implementación conjunta de ambos TFGs en el sistema final. Además de clave en muchos puntos comunes del desarrollo software.

12 Referencias y bibliografía

Recursos de texto

[1] Jesús San-Miguel-Ayanz, Tracy Durrant, Roberto Boca, Giorgio Libertà, Alfredo Branco, Daniele de Rigo, Davide Ferrari, Pieralberto Maianti, Tomàs Artés Vivancos, Ernste Schulte, Peter Loffler; Forest Fires in Europe, Middle East and North Africa 2016. EUR 28707 EN, Publications Office, Luxembourg, 2017

[2] La vanguardia 'Europa entra en una nueva era de incendios forestales' 2017

[3] La vanguardia 'Identificadas las 11 víctimas del incendio de Guadalajara', 2005

[4] Regional Fire Department of Asturias, Spain 'Asturies-er-1year' 2016

[5] Agencia Sinc 'Un nuevo simulador predice el comportamiento de los incendios forestales', 2012

[6] La vanguardia, 'Declarado el estado de desastre en California por incendios', 2018

[7] Dimo Brockhoff , Tobias Friedrich , and Frank Neumann, 'Analyzing Hypervolume Indicator Based Algorithms', 2008

Recursos web

[W1] https://github.com/Project-Platypus/Platypus

[W2] https://github.com/manuparra/moeac-solver

[W3] https://python-para-impacientes.blogspot.com/2015/05/operaciones-con-archivos-csv.html

[W4] https://stackoverflow.com/questions/21519203/plotting-a-list-of-x-y-coordinates-in-python-matplotlib

[W5] http://chris35wills.github.io/courses/PythonPackages_matplotlib/matplotlib_scatter/

[W6] https://stackoverflow.com/questions/14827650/pyplot-scatter-plot-marker-size

13 Anexo I: Código Script 1, 2, 3 - Adaptar datos del dataset

```
import matplotlib.pyplot as plt
import numpy as np
from platypus import NSGAII, SMPSO, MOEAD, experiment, Hypervolume, display,
calculate
from platypus.core import Problem
from platypus.types import Real
from mpl_toolkits.mplot3d import Axes3D
from scipy.interpolate import griddata
from matplotlib import cm

#####ADAPTAR DATOS
#####Datos definidos por Leticia (por ahora se obtienen del .csv)

#Slots de TIEMPO
num_slot = 0
slot = 15
#Tiempo del que disponen los equipos en cada slot para moverse
t_equipos = 30
#Tiempo en SEGUNDOS de todos los slots [Se USA para obtener los equipos que estan ya
#desplegados]
t = slot * t_equipos


Datos_excel = open('C:\Users\Alberto de la Cruz\Desktop\TFG\datos.csv')

LIM_MAX=1048576
index = 0
diccionario_datos_brutos = {}
LIM = 2.5e5
instante_t = 3000

for line in Datos_excel:
    if index>0:
        if index>LIM:
            break
        lineParsed = line.strip().split(' ')
        if ( (int(lineParsed[0])) <= (instante_t + t) and (int(lineParsed[0]) ) >= instante_t ):
            if lineParsed[1] not in diccionario_datos_brutos.keys():
                diccionario_datos_brutos[lineParsed[1]] = []


diccionario_datos_brutos[lineParsed[1]].append([int(lineParsed[0]),float(lineParsed[2]),float(
lineParsed[3])])
    index = index + 1
```

```
Datos_excel.close()
del lineParsed
del line
del index

#Adaptamos el formato al que SUPONGO que va a usar Leticia
pos_ini_X = []
pos_ini_Y = []
pos_prediccion_X = []
pos_prediccion_Y = []

for i in diccionario_datos_brutos.keys():
    longitud_X = [x[1] for x in diccionario_datos_brutos[i]] #Longitud
    pos_ini_X.append(longitud_X[0])
    pos_prediccion_X.append(longitud_X)
    latitud_Y = [x[2] for x in diccionario_datos_brutos[i]] #Latitud
    pos_ini_Y.append(latitud_Y[0])
    pos_prediccion_Y.append(latitud_Y)

del longitud_X, latitud_Y, i, x

#Adaptamos los datos, normalizamos los datos de predicción
tam_max = 0

for i in range(0,len(pos_prediccion_X)):
    tam = len(pos_prediccion_X[i])
    if tam>tam_max:
        tam_max = tam
del i,tam


#Igualamos el tamaño del array de datos de la prediccion a los slots que tengamos
for i in range(0,len(pos_prediccion_X)):
    tam = len(pos_prediccion_X[i])
    dif = tam_max -tam
    if dif != 0:
        for j in range(0,dif):
            pos_prediccion_X[i].append(pos_prediccion_X[i][tam - 1])
            pos_prediccion_Y[i].append(pos_prediccion_Y[i][tam - 1])

del i, tam, dif
```

14 Anexo II: Código Script 1, 2, 3– Generar nodos desplegados

```
#####Definimos los TIPOS de EQUIPOS que tenemos disponible para desplegar
######Definimos una  clase MEDIO que hace referencia a los medios que vamos a desplegar

class Medio(object):
    #Tipos de vehiculo, radio de cobertura [metros], radio de movimiento [metros/t], coste por
#metro[€/metros], posicion inicial, posicion final
    def __init__(self, tipo, r_cobertura, r_movimiento, coste, pos_ini_x, pos_ini_y):
        self.tipo = tipo
        self.r_cobertura = r_cobertura
        self.r_movimiento = r_movimiento
        self.coste = coste
        self.pos_ini_x = pos_ini_x
        self.pos_ini_y = pos_ini_y
        self.posiciones_x = []
        self.posiciones_y = []
        self.posiciones_x.append(pos_ini_x)
        self.posiciones_y.append(pos_ini_y)

#Definimos un array con objetos de la clase MEDIO, siendo estos los NUEVOS MEDIOS
#que vamos a desplegar
N_DRONES = 1
N_PERSONAS = 5
N_COCHES = 2
N_OTROS = 0
N_ASSETS = N_COCHES + N_DRONES + N_PERSONAS + N_OTROS

lista_equipos = []

#Tipos de vehiculo, radio de cobertura [metros], radio de movimiento [metros/t], coste por
#metro[€/metros], posicion inicial, posicion final

for i in range(0, max(N_DRONES,N_COCHES,N_OTROS,N_PERSONAS)):
    if i < N_DRONES:
        lista_equipos.append(Medio("Dron", 20000, 1950, 40, 15000, -75000))
    if i < N_PERSONAS:
        lista_equipos.append(Medio("Persona", 5000, 1250, 10, 15000, -75000))
    if i < N_COCHES:
        lista_equipos.append(Medio("Coche", 15000, 1400, 3, 15000, -75000))
    if i < N_OTROS:
        lista_equipos.append(Medio("Otros", 0, 0, 0, 0, 0))
del i
```

15 Anexo III: Código Script 1, 2, 3 – Definir problema multi-objetivo

```
#####Definimos el problema MULTI-OBJETIVO

class allocation(Problem):
 #Especificamos: [numero de variables, numero de objetivos, numero de restricciones]
   def __init__(self):
      #__init__(numeroVariablesDecision, numeroDeObjetivos, numeroDeRestricciones)
      super(allocation, self).__init__(N_ASSETS * 2, 2, 0)

      #Variables decision---> modulo del desplazamiento del vehiculo en el tiempo de
#prediccion (t_equipos)
      #A mayor t_equipos mayor coste, pero mayor cobertura
      #El ARRAY tiene el formato [Modulo,Angulo, Modulo,Angulo, ...]
      aux = []
      for i in range(0, N_ASSETS):
         aux.append(Real(0, lista_equipos[i].r_movimiento * t_equipos))
         aux.append(Real(0, 2 * np.pi))
      self.types[:] = aux
      del i , aux

      #Los objetivos (en este caso 2 coste y area cubierta) que tienen que cumplir
      self.directions[:] = [Problem.MINIMIZE, Problem.MAXIMIZE]

   def evaluate(self, solution):

      #Definimos los objetivos (en este caso 2):

      #-->Coste en función de la distancia recorrida C=modulo(m)*coste(€/m)
      solution.objectives[0] = sum([solution.variables[2*i] * lista_equipos[i].coste for i in
range(0, N_ASSETS)])
      indice_cobertura = np.zeros(len(diccionario_datos_brutos))

      #-->Area cubierta por el equipo
      for i in range(0,N_ASSETS):
         #Sacamos el modulo/argumento del desplazamiento optimizado-->distancia que tiene
#que recorrer
         mod = solution.variables[2*i]
         ang = solution.variables[2*i + 1]

         #Convertimos a cartesianas
         pos_x = mod * np.cos(ang)
         pos_y = mod * np.sin(ang)

         #Hallamos la posicion final del equipo desplegado (Sumando distancia recorrida y
#posicion inicial)
         pos_x_fin = lista_equipos[i].pos_ini_x + pos_x
```

```
        pos_y_fin = lista_equipos[i].pos_ini_y + pos_y


        for j in range(0,len(pos_ini_X)):
            #Calculamos la distancia a la que se encuentran los bomberos, equipos de rescate...
            X = (pos_x_fin - pos_prediccion_X[j][num_slot])**2
            Y = (pos_y_fin - pos_prediccion_Y[j][num_slot])**2
            dis = np.sqrt( X + Y )

            #Guardamos en 1 array si un bombero se encuentra cubierto por al menos un
#equipo nuevo
            if dis <= lista_equipos[i].r_cobertura:
                indice_cobertura[j] = 1

    solution.objectives[1] = float(sum(indice_cobertura))/len(pos_ini_X)
```

16 Anexo IV: Código Script 1 – Obtener soluciones del problema

```
#####CONCLUSIONES
#Soluciones optimas en cads SLOT

copia_equipos = lista_equipos
lista_solucion = []
#Elige el algoritmo
opc = 0

if opc == 0:
    algoritmo = NSGAII
    nom_algoritmo = "NSGAII"
if opc == 1:
    algoritmo = SMPSO
    nom_algoritmo = "SMPSO"
if opc == 2:
    algoritmo = MOEAD
    nom_algoritmo = "MOEAD"

for i in range(0, slot):
    num_slot = i
    lista = []
    algorithm = algoritmo(allocation())
    algorithm.run(500)

    for sol in algorithm.result:
        lista.append(sol)
    lista_solucion.append(lista)

    #En funcion de lo que queramos MIN coste
    #Guardamos como POS INICIAL la POS FINAl anterior que cumpla esas condiciones

    for j in range(0, N_ASSETS):
        mod = lista_solucion[i][0].variables[2*j]
        ang = lista_solucion[i][0].variables[2*j + 1]

        pos_x = mod * np.cos(ang)
        pos_y = mod * np.sin(ang)


        #De la solución que sea la óptima guardamos la posición en la variable con la que
#volvemos a iterar
        lista_equipos[j].pos_ini_x = lista_equipos[j].pos_ini_x + pos_x
        lista_equipos[j].pos_ini_y = lista_equipos[j].pos_ini_y + pos_y

        #Punto a punto por donde se desplaza el equipo
        lista_equipos[j].posiciones_x.append(lista_equipos[j].pos_ini_x)
```

```
        lista_equipos[j].posiciones_y.append(lista_equipos[j].pos_ini_y)

del i, j, lista, pos_x, pos_y, mod, ang
```

17 Anexo V: Código Script 1 – Representar soluciones del problema multi-objetivo

```
#####DIBUJAMOS resultados en 2D
#DIBUJAMOS la POSICION de los equipos en la zona en 2D
plt.figure(1)
plt.title("POSICIONES")
plt.xlabel("(X)")
plt.ylabel("(Y)")

#Equipos en el área
for i in range(0,len(pos_prediccion_X)):
    plt.plot(pos_prediccion_X[i], pos_prediccion_Y[i],'xr-')
del i

#Nodos desplegados
for i in range(0,N_ASSETS):
    #Radio de cobertura de los nodos
    plt.scatter(lista_equipos[i].posiciones_x[(len(lista_equipos[i].posiciones_x) - 1)],
lista_equipos[i].posiciones_y[len(lista_equipos[i].posiciones_y) - 1],
color='lightgrey',marker='o',s=lista_equipos[i].r_cobertura,edgecolors='black',alpha=0.06)
    #Posicion de los nodos
    plt.plot(lista_equipos[i].posiciones_x, lista_equipos[i].posiciones_y, color='lightgrey',
linestyle='dashed', marker='s', markerfacecolor='lightgrey', markersize=5)
del i

plt.show()

#Dibujamos en 2D los frentes de pareto
plt.figure(2)
plt.ylim(0,1)
plt.title("SOLUCIONES " + nom_algoritmo)
plt.xlabel("COSTE(X)")
plt.ylabel("COBERTURA(Y)")

#Dibujamos la solucion de cada slot de tiempo
for i in range(0,slot):
    plt.scatter([s.objectives[0] for s in lista_solucion[i]],[s.objectives[1] for s in
lista_solucion[i]])
del i

plt.show()

#####FRENTES de PARETO 3D
#Dibujamos las soluciones de cada slot de tiempo en 3D
fig = plt.figure(3)
ax = fig.add_subplot(111, projection='3d')
ax.set_title(nom_algoritmo)
```

```
ax.set_xlabel('COSTE')
ax.set_ylabel('TIEMPO')
ax.set_zlabel('COBERTURA')

for i in range(0,slot):
    ax.scatter([s.objectives[0] for s in lista_solucion[i]], 30 * i, [s.objectives[1] for s in
lista_solucion[i]], c='r', marker='o')
del i

plt.show()

#####Hypervolumen
#Elegimos el ALGORITMO
lista_equipos = copia_equipos
lista_solucion = []
lista_hyp = []
c_max = 0
d_max = 0

for i in range(0, slot):
    num_slot = i
    results = experiment(algoritmo, allocation(), nfe=500, seeds=1)

    lista_solucion = results[nom_algoritmo]["allocation"]

    #En funcion de lo que queramos MIN coste
    #Guardamos como POS INICIAL la POS FINAl anterior que cumpla esas condiciones

    for j in range(0, N_ASSETS):
        mod = lista_solucion[0][0].variables[2*j]
        ang = lista_solucion[0][0].variables[2*j + 1]

        pos_x = mod * np.cos(ang)
        pos_y = mod * np.sin(ang)

        #De la solución que sea la óptima guardamos la posición en la variable con la que
#volvemos a iterar
        lista_equipos[j].pos_ini_x = lista_equipos[j].pos_ini_x + pos_x
        lista_equipos[j].pos_ini_y = lista_equipos[j].pos_ini_y + pos_y

        #Calculamos el COSTE maximo

        for k in range(0,len(pos_prediccion_X)):
            X = (lista_equipos[j].pos_ini_x - pos_prediccion_X[j][num_slot])**2
            Y = (lista_equipos[j].pos_ini_y - pos_prediccion_Y[j][num_slot])**2
            dis = np.sqrt( X + Y )
            if d_max < dis:
                d_max = dis
```

```
            c_max = dis*lista_equipos[j].coste

        #Punto a punto por donde se desplaza el equipo
        lista_equipos[j].posiciones_x.append(lista_equipos[j].pos_ini_x)
        lista_equipos[j].posiciones_y.append(lista_equipos[j].pos_ini_y)

    #Calculamos el Hypervolumen
    hyp = Hypervolume(minimum=[0, 0], maximum=[c_max, 1])
    hyp_result = calculate(results, hyp)
    lista_hyp.append(hyp_result[nom_algoritmo]["allocation"]["Hypervolume"][0])

del i, j, pos_x, pos_y, mod, ang, c_max, results, hyp_result, hyp, k, d_max, dis

#Dibujamos el HYPERVOLUMEN
plt.figure(5)
plt.ylim(0,1)
plt.title("Hypervolumen " + nom_algoritmo)
plt.xlabel("TIEMPO(X)")
plt.ylabel("Hyp(Y)")

lista_t = []
for i in range(0,slot):
    lista_t.append(30*i)
del i

plt.plot(lista_t, lista_hyp, color='r', marker='o', linestyle='-', markerfacecolor='red',
markersize=5)

plt.show()
```

18 Anexo VI: Código Script 2 – Obtener soluciones del problema

```
#####CONCLUSIONES
copia_equipos = lista_equipos
#Lista con las SOLUCIONES a lo largo del NUMERO de slots indicados
lista_solucion = []

#Elige el algoritmo
opc = 0

if opc == 0:
    algoritmo = NSGAII
    nom_algoritmo = "NSGAII"
if opc == 1:
    algoritmo = SMPSO
    nom_algoritmo = "SMPSO"
if opc == 2:
    algoritmo = MOEAD
    nom_algoritmo = "MOEAD"


for i in range(0, slot):
    num_slot = i
    lista = []
    algorithm = algoritmo(allocation())
    algorithm.run(500)

    for sol in algorithm.result:
        lista.append(sol)
    lista_solucion.append(lista)

    #En funcion de lo que queramos MIN coste
    #Guardamos como POS INICIAL la POS FINAl anterior que cumpla esas condiciones

    for j in range(0, N_ASSETS):

        mod = lista_solucion[i][int(len(lista_solucion[i])/2) - 1].variables[2*j]
        ang = lista_solucion[i][int(len(lista_solucion[i])/2) - 1].variables[2*j + 1]

        pos_x = mod * np.cos(ang)
        pos_y = mod * np.sin(ang)


        #De la solución que sea la óptima guardamos la posición en la variable con la que volvemos a iterar
        lista_equipos[j].pos_ini_x = lista_equipos[j].pos_ini_x + pos_x
        lista_equipos[j].pos_ini_y = lista_equipos[j].pos_ini_y + pos_y
```

```
        #Punto a punto por donde se desplaza el equipo
        lista_equipos[j].posiciones_x.append(lista_equipos[j].pos_ini_x)
        lista_equipos[j].posiciones_y.append(lista_equipos[j].pos_ini_y)


del i, j, lista, pos_x, pos_y, mod, ang
```

19 Anexo VII: Código Script 2 – Representar soluciones del problema multi-objetivo

```
#POSICION de los equipos en la zona 2D
plt.figure(1)
plt.title("POSICIONES")
plt.xlabel("(X)")
plt.ylabel("(Y)")

for i in range(0,len(pos_prediccion_X)):
   plt.plot(pos_prediccion_X[i], pos_prediccion_Y[i],'xr-')
del i

#Nodos desplegados
for i in range(0,N_ASSETS):
   #Radio de cobertura de los nodos
   plt.scatter(lista_equipos[i].posiciones_x[(len(lista_equipos[i].posiciones_x) - 1)],
lista_equipos[i].posiciones_y[len(lista_equipos[i].posiciones_y) - 1],
color='lightgrey',marker='o',s=lista_equipos[i].r_cobertura,edgecolors='black',alpha=0.06)
   #Posicion de los nodos
   plt.plot(lista_equipos[i].posiciones_x, lista_equipos[i].posiciones_y, color='lightgrey',
linestyle='dashed', marker='s', markerfacecolor='lightgrey', markersize=5)
del i

plt.show()

#Dibujamos en 2D los frentes de pareto
plt.figure(3)
plt.ylim(0,1)
plt.title("SOLUCIONES " + nom_algoritmo)
plt.xlabel("COSTE(X)")
plt.ylabel("COBERTURA(Y)")

#Dibujamos la solucion de cada slot de tiempo
for i in range(0,slot):
   plt.scatter([s.objectives[0] for s in lista_solucion[i]],[s.objectives[1] for s in
lista_solucion[i]])
del i


#####FRENTES de PARETO 3D
#Dibujamos las soluciones de cada slot de tiempo en 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_title(nom_algoritmo)
ax.set_xlabel('COSTE')
ax.set_ylabel('TIEMPO')
ax.set_zlabel('COBERTURA')
```

```
for i in range(0,slot):
   ax.scatter([s.objectives[0] for s in lista_solucion[i]], 30 * i, [s.objectives[1] for s in lista_solucion[i]], c='r', marker='o')
del i

plt.show()


#####Hypervolumen
#Elegimos el ALGORITMO
lista_equipos = copia_equipos
lista_solucion = []
lista_hyp = []
c_max = 0
d_max = 0

for i in range(0, slot):
   num_slot = i
   results = experiment(algoritmo, allocation(), nfe=500, seeds=1)

   lista_solucion = results[nom_algoritmo]["allocation"]

   #En funcion de lo que queramos MIN coste
   #Guardamos como POS INICIAL la POS FINAl anterior que cumpla esas condiciones

   for j in range(0, N_ASSETS):
      mod = lista_solucion[0][int(len(lista_solucion[0])/2)].variables[2*j]
      ang = lista_solucion[0][int(len(lista_solucion[0])/2)].variables[2*j + 1]

      pos_x = mod * np.cos(ang)
      pos_y = mod * np.sin(ang)

      #De la solución que sea la óptima guardamos la posición en la variable con la que #volvemos a iterar
      lista_equipos[j].pos_ini_x = lista_equipos[j].pos_ini_x + pos_x
      lista_equipos[j].pos_ini_y = lista_equipos[j].pos_ini_y + pos_y

      #Calculamos el COSTE maximo

      for k in range(0,len(pos_prediccion_X)):
         X = (lista_equipos[j].pos_ini_x - pos_prediccion_X[j][num_slot])**2
         Y = (lista_equipos[j].pos_ini_y - pos_prediccion_Y[j][num_slot])**2
         dis = np.sqrt( X + Y )
         if d_max < dis:
            d_max = dis
            c_max = dis*lista_equipos[j].coste
```

```
        #Punto a punto por donde se desplaza el equipo
        lista_equipos[j].posiciones_x.append(lista_equipos[j].pos_ini_x)
        lista_equipos[j].posiciones_y.append(lista_equipos[j].pos_ini_y)

    #Calculamos el Hypervolumen
    hyp = Hypervolume(minimum=[0, 0], maximum=[c_max, 1])
    hyp_result = calculate(results, hyp)
    lista_hyp.append(hyp_result[nom_algoritmo]["allocation"]["Hypervolume"][0])

del i, j, pos_x, pos_y, mod, ang, c_max, results, hyp_result, hyp, k, d_max, dis

#Dibujamos el HYPERVOLUMEN
plt.figure(5)
plt.ylim(0,1)
plt.title("Hypervolumen " + nom_algoritmo)
plt.xlabel("TIEMPO(X)")
plt.ylabel("Hyp(Y)")

lista_t = []
for i in range(0,slot):
    lista_t.append(30*i)
del i

plt.plot(lista_t, lista_hyp, color='r', marker='o', linestyle='-', markerfacecolor='red',
markersize=5)

plt.show()
```

20 Anexo VIII: Código Script 3 – Obtener soluciones del problema

```
#####CONCLUSIONES
copia_equipos = lista_equipos
#Lista con las SOLUCIONES a lo largo del NUMERO de slots indicados
lista_solucion = []

#Elige el algoritmo
opc = 0

if opc == 0:
    algoritmo = NSGAII
    nom_algoritmo = "NSGAII"
if opc == 1:
    algoritmo = SMPSO
    nom_algoritmo = "SMPSO"
if opc == 2:
    algoritmo = MOEAD
    nom_algoritmo = "MOEAD"


for i in range(0, slot):
    num_slot = i
    lista = []
    algorithm = algoritmo(allocation())
    algorithm.run(500)

    for sol in algorithm.result:
        lista.append(sol)
    lista_solucion.append(lista)

    #En funcion de lo que queramos MIN coste
    #Guardamos como POS INICIAL la POS FINAl anterior que cumpla esas condiciones

    for j in range(0, N_ASSETS):
        mod = lista_solucion[i][len(lista_solucion[i]) - 1].variables[2*j]
        ang = lista_solucion[i][len(lista_solucion[i]) - 1].variables[2*j + 1]

        pos_x = mod * np.cos(ang)
        pos_y = mod * np.sin(ang)


        #De la solución que sea la óptima guardamos la posición en la variable con la que
#volvemos a iterar
        lista_equipos[j].pos_ini_x = lista_equipos[j].pos_ini_x + pos_x
        lista_equipos[j].pos_ini_y = lista_equipos[j].pos_ini_y + pos_y
```

```
        #Punto a punto por donde se desplaza el equipo
        lista_equipos[j].posiciones_x.append(lista_equipos[j].pos_ini_x)
        lista_equipos[j].posiciones_y.append(lista_equipos[j].pos_ini_y)


del i, j, lista, pos_x, pos_y, mod, ang
```

21 Anexo IX: Código Script 3 – Representar soluciones del problema multi-objetivo

```
#POSICION de los equipos en la zona 2D
plt.figure(1)
plt.title("POSICIONES " + nom_algoritmo)
plt.xlabel("(X)")
plt.ylabel("(Y)")

for i in range(0,len(pos_prediccion_X)):
    plt.plot(pos_prediccion_X[i], pos_prediccion_Y[i],'xr-')
del i

#Nodos desplegados
for i in range(0,N_ASSETS):
    #Radio de cobertura de los nodos
    plt.scatter(lista_equipos[i].posiciones_x[(len(lista_equipos[i].posiciones_x) - 1)],
lista_equipos[i].posiciones_y[len(lista_equipos[i].posiciones_y) - 1],
color='lightgrey',marker='o',s=lista_equipos[i].r_cobertura,edgecolors='black',alpha=0.06)
    #Posicion de los nodos
    plt.plot(lista_equipos[i].posiciones_x, lista_equipos[i].posiciones_y, color='lightgrey',
linestyle='dashed', marker='s', markerfacecolor='lightgrey', markersize=5)
del i

plt.show()

#Dibujamos en 2D los frentes de pareto
plt.figure(3)
plt.ylim(0,1)
plt.title("SOLUCIONES " + nom_algoritmo)
plt.xlabel("COSTE(X)")
plt.ylabel("COBERTURA(Y)")

#Dibujamos la solucion de cada slot de tiempo
for i in range(0,slot):
    plt.scatter([s.objectives[0] for s in lista_solucion[i]],[s.objectives[1] for s in
lista_solucion[i]])
del i



###FRENTES de PARETO 3D
#Dibujamos las soluciones de cada slot de tiempo en 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_title(nom_algoritmo)
ax.set_xlabel('COSTE')
ax.set_ylabel('TIEMPO')
```

```
ax.set_zlabel('COBERTURA')

for i in range(0,slot):
    ax.scatter([s.objectives[0] for s in lista_solucion[i]], 30 * i, [s.objectives[1] for s in
lista_solucion[i]], c='r', marker='o')
del i

plt.show()


###Hypervolumen
#Elegimos el ALGORITMO
lista_equipos = copia_equipos
lista_solucion = []
lista_hyp = []
c_max = 0
d_max = 0

for i in range(0, slot):
    num_slot = i
    results = experiment(algoritmo, allocation(), nfe=500, seeds=1)

    lista_solucion = results[nom_algoritmo]["allocation"]

    #En funcion de lo que queramos MIN coste
    #Guardamos como POS INICIAL la POS FINAl anterior que cumpla esas condiciones

    for j in range(0, N_ASSETS):
        mod = lista_solucion[0][len(lista_solucion[0]) - 1].variables[2*j]
        ang = lista_solucion[0][len(lista_solucion[0]) - 1].variables[2*j + 1]

        pos_x = mod * np.cos(ang)
        pos_y = mod * np.sin(ang)

        #De la solución que sea la óptima guardamos la posición en la variable con la que
#volvemos a iterar
        lista_equipos[j].pos_ini_x = lista_equipos[j].pos_ini_x + pos_x
        lista_equipos[j].pos_ini_y = lista_equipos[j].pos_ini_y + pos_y

        #Calculamos el COSTE maximo

        for k in range(0,len(pos_prediccion_X)):
            X = (lista_equipos[j].pos_ini_x - pos_prediccion_X[j][num_slot])**2
            Y = (lista_equipos[j].pos_ini_y - pos_prediccion_Y[j][num_slot])**2
            dis = np.sqrt( X + Y )
            if d_max < dis:
                d_max = dis
                c_max = dis*lista_equipos[j].coste
```

```
        #Punto a punto por donde se desplaza el equipo
        lista_equipos[j].posiciones_x.append(lista_equipos[j].pos_ini_x)
        lista_equipos[j].posiciones_y.append(lista_equipos[j].pos_ini_y)

    #Calculamos el Hypervolumen
    hyp = Hypervolume(minimum=[0, 0], maximum=[c_max, 1])
    hyp_result = calculate(results, hyp)
    lista_hyp.append(hyp_result[nom_algoritmo]["allocation"]["Hypervolume"][0])

del i, j, pos_x, pos_y, mod, ang, c_max, results, hyp_result, hyp, k, d_max, dis

#Dibujamos el HYPERVOLUMEN
plt.figure(5)
plt.ylim(0,1)
plt.title("Hypervolumen " + nom_algoritmo)
plt.xlabel("TIEMPO(X)")
plt.ylabel("Hyp(Y)")

lista_t = []
for i in range(0,slot):
    lista_t.append(30*i)
del i

plt.plot(lista_t, lista_hyp, color='r', marker='o', linestyle='-', markerfacecolor='red',
markersize=5)

plt.show()
```