Universidad de La Laguna
Escuela Superior de Ingeniería y Tecnología
Sección de Ingeniería Informática

225 ANIVERSARIO 1792 - 2017

Trabajo de Fin de Grado

Herramienta para la validación estadística de metaheurísticas

Tool for statistical validation of metaheuristics

Pedro Manuel Ramos Rodríguez

La Laguna, 6 de junio de 2017

D[a]. **Gara Miranda Valladares**, con N.I.F. 78.563.584-T profesora Ayudante Doctor adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

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

I N F O R M A (N)

Que la presente memoria titulada:

"*Herramienta para la validación estadística de metaheurísticas*"

ha sido realizada bajo su dirección por D. **Pedro Manuel Ramos Rodríguez**, con N.I.F. 54.049.267-A.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 6 de junio de 2017

Agradecimientos

Quisiera agradecer a mi tutora Gara Miranda Valladares por todos los consejos que me ha dado, la ayuda y la dedicación prestada. Gracias a sus consejos y a la paciencia que ha tenido en las explicaciones, ha sido posible realizar este trabajo. También, me gustaría agradecer a Eduardo Segredo González por la ayuda prestada y consejos para poder mejorar los resultados obtenidos en el trabajo.

Además, me gustaría agradecer el apoyo de mi hermana y mi madre ya que sin ellas posiblemente no estaría escribiendo estas palabras de agradecimiento. También y no menos importante, agradecer a los amigos y compañeros que han hecho posible que el camino hasta aquí haya sido más ameno.

Licencia

Resumen

El objetivo de este Trabajo de Fin de Grado ha sido el desarrollo de una aplicación web que facilite el proceso de experimentación involucrado en proyectos de programación científica que impliquen el análisis o comparación de metaheurísticas. Concretamente, se realizará una comparativa de metaheurísticas utilizadas para la resolución de problemas multi-objetivo.

Este tipo de metaheurísticas utilizan una serie de métricas para evaluar el rendimiento. Existen multitud de métricas para comparar el rendimiento entre las diferentes metaheurísticas que resuelven problemas multi-objetivo (hipervolumen, distancia generacional, familia epsilon, superficie de alcance, etc). En la aplicación desarrollada, se tiene en cuenta la métrica más utilizada para la comparación de técnicas de resolución de problemas de optimización multi-objetivo, el hipervolumen. Debido a la estocasticidad de los resultados obtenidos, se necesita corroborar los datos producidos por las diferentes metaheurísticas analizadas mediante una serie de tests estadísticos.

La herramienta proporcionará al usuario mecanismos para diseñar las pruebas a realizar, así como para aplicar un conjunto de técnicas estadísticas que permitan analizar, interpretar y representar, de forma apropiada, los resultados obtenidos permitiendo en cada caso comparar adecuadamente distintos métodos de resolución. Una vez el usuario ha realizado con éxito el diseño del análisis a realizar, la aplicación mostrará los resultados obtenidos mediante una serie de gráficas y tablas mostrando las conclusiones del experimento. Además, se permitirá al usuario descargar los resultados en formato imagen para los gráficos y en formato .tex, .txt y pdf para las tablas obtenidas.

Palabras clave: optimización multi-objetivo, metaheurísticas, métricas para la evaluación del rendimiento algorítmico, análisis estadístico.

Abstract

The objective of this Final Degree Project has been to develop a web application that help to facilitate the experiment process envolved in scientific development projects that compare and analyze metaheuristics. Specifically, a comparison of metaheuristics for multi-objective optimization will be performed.

This kind of metaheuristics, uses a series of metrics to evaluate the performance. There are a lot of metrics to compare the performance between the metaheuristics for resolving multi-objective problems (hypervolume, generational distance, epsilon family, attainment surface and so on).

In this web application, the most used metric for comparing multi-objective optimization methods was taken into account, the *hypervolume*. As the results are stochastic, it will be necessary to compare statistically the results of the different metaheuristics, for this purpose, it will be used a series of statistical tests. The application will provide to the users a series of tools to design the test to perform, will offer a statistical set to analyze, translate and represent the results obtained and compare the different methods of resolution.

Finally, when the analysis design is done, the application will show de results in charts and tables showing the conclusions of the experiment. In addition, will be possible download the charts in image file and the results of the tables in *.tex*, *.txt* and *pdf* file.

Keywords: *multi-objective optimization, metaheuristic, metrics for algorithm performance evaluation, statistical analysis.*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Antecedentes

Por un lado, es altamente deseable diseñar metaheurísticas de"propósito general", que no requieran de un conocimiento específico del problema y que se puedan aplicar fácilmente a un amplio abanico de problemas. Esta ha sido una línea de investigación puntera en las últimas décadas que ha dado como resultado la aparición de una serie de paradigmas metaheurísticos de propósito general: recocido simulado, algoritmos evolutivos, búsqueda tabú, colonia de hormigas, etc. Estos paradigmas tienen propósito general y parecen especialmente adecuados para los profesionales que están interesados en obtener una "buena" solución al problema sin invertir una gran cantidad de tiempo en la comprensión de las propiedades matemáticas del modelo ni en el diseño de algoritmos a medida.

Hay que tener en cuenta que este tipo de algoritmos, en general, parecen proporcionar un rendimiento variable dependiendo de la sensibilidad (habilidades, pericia, ingenio, etc.) del diseñador en la afinación algorítmica. Con el fin de maximizar el rendimiento algorítmico, puede ser necesaria una afinación fina de los parámetros involucrados en el algoritmo. Además, el no determinismo presente en este tipo de algoritmos produce una variabilidad en los mismos que provoca algunos problemas en cuanto a la reproducibilidad de los resultados. Por estos motivos, es difícil comparar y evaluar a fondo una metaheurística y su rendimiento. Según el *no free lunch theorem* ningún algoritmo supera a todos los demás para todo tipo de instancias o problemas, sin embargo, es interesante a nivel práctico obtener métodos de propósito general que sean capaces de resolver eficazmente un amplio espectro de problemas. Es decir, ningún método puede garantizar soluciones de alta calidad sobre todas las instancias posibles de una clase de problema dada pero, al menos, es deseable presentar un comportamiento robusto sobre un espectro de instancias pertenecientes a la misma clase de problema. De esta forma, se podrían identificar el tipo de métodos que mejor funcionan para según qué tipo de problemas.

Sin embargo, debido al comportamiento estocástico de algunas metaheurísticas y la falta de técnicas comúnmente aceptadas y adoptadas para la evaluación del rendimiento algorítmico, dados dos algoritmos diseñados para abordar la misma clase de problemas, no siempre es posible "clasificar" tales algoritmos en términos de desempeño. Una consecuencia directa es que todavía no se entiende claramente qué características son realmente exitosas y en qué circunstancias. En otras palabras, a menos que se definan estándares claros sobre metaheurísticas, será realmente difícil tener una comprensión total de, primero, qué algoritmos son mejores que otros y, en segundo lugar, cuál es la contribución real de cada característica del algoritmo sobre el conjunto de datos.

Por todo ello, en la actualidad, cuando los expertos diseñan y testean nuevas metaheurísticas o cuando un usuario quiere determinar la metaheurística que mejor se adapta al tipo de problema que le atiene, una práctica habitual durante la experimentación consiste en tener en cuenta los aspectos siguientes:

- Probar el método con un conjunto de problemas/instancias lo suficientemente amplio y consolidado en la literatura.
- Sintonizar adecuadamente los parámetros involucrados en el algoritmo.
- Identificar y testear el comportamiento de otros métodos consolidados en la literatura para la resolución del mismo tipo de problemas/instancias, afinando en caso necesario los parámetros involucrados en los mismos.
- Realizar un número suficientemente elevado de ejecuciones para cada uno de los métodos, problemas o instancias a evaluar.
- Finalmente, es necesario evaluar o comparar la validez o la eficiencia de los distintos métodos. Para realizar este tipo de comparación es necesario realizar un análisis estadístico de los datos (soluciones) obtenidos. Cabe destacar que para el caso de problemas de optimización multi-objetivo los mecanismos de análisis deberán tener en cuenta la dimensionalidad de las soluciones.

Para llevar a cabo este tipo de análisis existen diversos mecanismos estadísticos que facilitan la determinación de las diferencias estadísticas (significativas o no) entre los distintos algoritmos. Además, también existen herramientas software que implementan dichas funciones estadísticas. Sin embargo, para facilitar la integración de metaheurísticas en entornos reales, donde este marco experimental es desconocido por los usuarios, sería de gran interés poder contar con alguna herramienta que automatizara, guiara y simplificara en la medida de lo posible todo este proceso.

1.2. Optimización y resolución de problemas

La optimización combinatoria es una rama de la optimización en matemáticas aplicadas y en ciencias de la computación, relacionada con la investigación operativa, la teoría de algoritmos y la teoría de la complejidad computacional. El propósito de la optimización combinatoria es encontrar objetos matemáticos discretos que maximicen (o minimicen) una determinada función objetivo. Generalmente, estos objetos se denominan estados y al conjunto de todos los posibles estados candidatos se le denomina espacio de búsqueda. La naturaleza de los estados y del espacio de búsqueda viene determinada por el tipo de problema.

Aquellos problemas que se centran en la optimización de una única función objetivo reciben el nombre de problemas mono-objetivo. Sin embargo, en multitud de problemas reales en ingeniería es necesario optimizar simultáneamente un conjunto de funciones objetivo. A este tipo de problemas se les denomina problemas multi-objetivo.

1.2.1. Optimización mono-objetivo

Un problema de optimización mono-objetivo se define de la siguiente manera:

Definición. *Formalmente, se entiende por aquel problema de la forma:* $f(x)$ *sujeto a* $g_i(x) \leq 0,\ i = \{1,\ \ldots, m\}$, $h_j(x) = 0,\ j = \{1, \ldots, p\}$, *y con* $x \in \Omega$. *Una solución minimiza (o maximiza) el escalar* $f(x)$, *donde* x *es un vector n-dimensional de variables de decisión* $x = (x_1,\ \ldots, x_n)$ *del universo* Ω.

Destacar que $g_i(x) \leq 0$ y $h_j(x) = 0$ representan restricciones que deben ser cumplidas mientras se está optimizando $f(x)$. Ω contiene todas las posibles soluciones factibles x que pueden ser utilizadas para satisfacer una evaluación de $f(x)$ sus restricciones.

Así, $x = x_1,\ \ldots, x_n$ representa un vector de decisión, cuyos valores podrán ser contínuos o discretos. Por su parte, la función objetivo $f(x)$ también puede ser contínua o discreta.

En general, el mínimo global de un problema mono-objetivo es definido como [3, 8]:

Definición. *Dada una función* $f\colon\ \Omega \subseteq \Re^n \to \Re,\ \Omega \neq \emptyset$, *para* $x \in \Omega$ *el valor* $f(x^*) > -\infty$ *es llamado un* ***mínimo global*** *si y solo si*

$$\forall x \in \Omega\colon\ \ f(x^*) \leq f(x)$$

donde x^* *es por definición la solución global mínima,* f *es la función objetivo, y el conjunto* Ω *es la región factible de* x. *El objetivo de determinar la solución mínima global o las soluciones mínimas globales es llamado el* ***problema de optimización global*** *para un problema mono-objetivo.*

1.2.2. Multi-objetivo

Definición. *Formalmente, se entiende por problema de optimización multi-objetivo (Multi- Objective Optimization Problem, MOP) aquel problema de la forma:*

Maximizar o minimizar $F(x) = (f_1;\ f_2;\ \ldots;\ f_n)$ *sujeto a m restricciones de desigualdad:* $g_i(x) \leq 0; \quad i = 1;\ 2;\ \ldots;\ m$, *sujetos a p restricciones de igualdad:* $h_j(x) = 0; \quad j = 1;\ 2;\ \ldots;\ p$, *donde n es el número de funciones objetivo y r es el número de variables de decisión del problema:* $x = x_1,\ x_2,\ \ldots;\ x_r$.

Normalmente, en los problemas de optimización multi-objetivo, los objetivos presentan algún grado de conflicto entre ellos, por lo que puede no existir una solución que optimice todos y cada uno de los objetivos simultáneamente. En estos casos, se trata entonces de encontrar un conjunto de soluciones no dominadas lo más cercana posible al conjunto óptimo. Ninguna solución de ese conjunto es peor que otra si se tienen en cuenta todos los objetivos. En estos casos, el concepto de optimalidad es, por tanto, diferente al manejado para otro tipo de problemas, como por ejemplo, los problemas de optimización mono-objetivo. El término más comúnmente aceptado es el de optimalidad de Pareto.

Definición. *Se dice que un vector de variables de decisión* $x^* \in \mathcal{F}$ *es un óptimo de Pareto si no existe otro* $x \in \mathcal{F}$ *tal que* $f_i(x) \leq f_i(x^*)$ *para todo* $i = 1,\ \ldots,\ k$ *y* $f_j(X) < f_j(x^*)$ *para al menos un* j.

Es decir, x^* es un óptimo de Pareto si no existe otro punto factible $x \in \mathcal{F}$ que mejore alguno de los objetivos sin causar un empeoramiento en ninguno de los demás objetivos. Esto, tal y como se ha mencionado anteriormente, proporciona más de una solución. A este conjunto de soluciones se le denomina *óptimos de Pareto* o *soluciones no dominadas*. La imagen formada por este conjunto se denomina frente de Pareto.

En la optimización de problemas multi-objetivo, se podrían ponderar los distintos objetivos para obtener una única función objetivo y, por tanto, poder aplicar los métodos clásicos de resolución. Sin embargo, debido a la complejidad inherente en la mayoría de los MOPs (espacios de búsqueda muy grandes, incertidumbre, ruido, curvas de Pareto disjuntas, etc.), las técnicas tradicionales que tratan de transformarlos en problemas mono-objetivo no son convenientes para su resolución. Es por ello que existe un conjunto de algoritmos especialmente diseñados para afrontar la resolución de los MOPS como tales. Estos algoritmos tienen en cuenta que es posible que no exista una única solución optimizando todos y cada uno de los objetivos. En tal caso, lo que se obtiene es un conjunto de soluciones no inferiores o no dominadas. De entre este conjunto de soluciones, un experto podrá elegir la que más convenga en cada caso particular.

Los algoritmos evolutivos multi-objetivo (*Multi-Objective Evolutionary Algorithms*, MOEAs) son una de las metaheurísticas más ampliamente utilizadas pa-

ra resolver MOPs. Se ha demostrado que estos enfoques evolutivos son eficaces resolviendo MOPs. Los algoritmos evolutivos son una familia de técnicas metaheurísticas basadas en población e inspiradas en la evolución de las especies. Estas técnicas han sido utilizadas satisfactoriamente en numerosos problemas de gran complejidad en diferentes campos científicos y tecnológicos.

En la evolución natural, un entorno se encuentra habitado por una población limitada de individuos los cuales compiten por su supervivencia y reproducción. La selección natural, por tanto, es un mecanismo que evita el crecimiento descontrolado de las poblaciones y que favorece a aquellos individuos que mejor se adaptan al entorno que les rodea. Este fenómeno es también conocido como "*la supervivencia del más apto*". Los supervivientes, por tanto, serán capaces de reproducirse con el objetivo de perpetuar la especie, dando origen a individuos igual o mejor adaptados al entorno.

Trasladando este concepto a la resolución de problemas de optimización, en este tipo de problemas tenemos funciones objetivo a optimizar sujetas a unas determinadas restricciones. En los algoritmos evolutivos se hace uso de funciones de aptitud, o *fitness*, para averiguar cuan buena es una solución con respecto a los objetivos del problema. La calidad de este valor de aptitud determinará si un individuo es apto para la supervivencia en la población y su reproducción. El pseudocódigo general de estos algoritmos se muestra en el Algoritmo 1.

Algoritmo 1 Pseudocódigo genérico de un algoritmo evolutivo

1: Incialización de parámetros del algoritmo.
2: Generación de la población inicial.
3: Evaluación de la población inicial para obtener el valor de fitness de cada individuo.
4: **mientras** no se cumpla el criterio de parada **hacer**
5: **Selección de padres:** selección de individuos de la población actual para construir el mating pool o conjunto de individuos más aptos.
6: **Cruce y mutación:** aplicación del operador de cruce entre los individuos del mating pool para generar la nueva población, denominada offspring, y utilizar en ella el operador de mutación con la intención de dotar al individuo de características que no estaban presentes en los padres.
7: **Evaluación:** evaluación de los nuevos individuos para la asignación de los valores de fitness correspondientes.
8: **Selección de supervivientes:** selección de los mejores individuos, entre padres e hijos, como supervivientes para pasar a la siguiente generación.
9: **fin mientras**

1.3. Objetivos

Los objetivos a conseguir en este proyecto son: realizar una comparación de MOEAS (Multi-Objective Evolutionary Algorithms), validar de manera estadística los resultados obtenidos y mostrar de una forma visual las comparaciones entre los diferentes algoritmos analizados.

Para poder realizar este proceso en su conjunto, primero hay que realizar el proceso de experimentación. Este proceso, consta de una serie de pasos que el usuario debe realizar previamente para realizar las comparativas de los diferentes algoritmos utilizados que han sido descritos en la sección 1.1

Una vez el usuario ha realizado el experimento en sí, se producirán una serie de resultados de salida (ficheros en formato *.dat*, *.txt*, etc.) que posteriormente se utilizarán para realizar las comparativas de los resultados obtenidos.

Antes de realizar el desarrollo de la aplicación, se realizó un estudio del estado del arte para examinar aplicaciones existentes que cumplieran los objetivos propuestos en el mismo. Tras realizar este estudio, se comprobó que existen frameworks de algoritmos evolutivos para obtener soluciones a la resolución de problemas multi-objetivo, aplicaciones que muestran una comparativa de manera gráfica ó aplicaciones para realizar diferentes test estadísticos para comparar resultados.

Una vez se muestren los resultados de las diferentes metaheurísticas utilizadas, se procedería a realizar la validación estadística de los datos obtenidos. Esta validación, habitualmente consta de la realización de diferentes tests estadísticos para la aprobación de los datos.

El objetivo de la aplicación desarrollada en este proyecto es unificar todo el proceso. Es decir, que con una única aplicación se muestren los resultados obtenidos de los algoritmos seleccionando un tipo de métrica o un conjunto de ellas, y además, mostrar los resultados estadísticos del experimento realizado. Como conclusión al experimento, se le permitirá al usuario la posibilidad de guardar los resultados obtenidos en ficheros de imagen y tablas exportadas en formato Latex, Pdf y formato de texto.

Capítulo 2

Experimentación y metaheurísticas

2.1. Proceso experimentación

Como se indicó en la sección 1.1, para realizar una comparación entre diferentes metaheurísticas hay que realizar un proceso de experimentación. A continuación (ver Figura 2.1), se describe el proceso de resolución del problema y análisis de los resultados.

Como se aprecia en la imagen, para resolver un problema en concreto, primero hay que definir qué problema se quiere resolver, cuál o cuáles son las variables que posee este problema y las restricciones que posee el mismo. A continuación, se define el número de objetivos que se quiere resolver. Si el problema sólo posee una función objetivo, se le denominará problema mono-objetivo. Mientras que si el problema a resolver tiene un número igual o mayor a dos objetivos, se define el problema como un problema multi-objetivo.

Una vez el definido el problema y el número de objetivos a resolver, se realizará la optimización del mismo, es decir, obtener una solución al problema propuesto con los mejores resultados. En caso de que sea un problema mono-objetivo, se obtendrá la solución del mismo con una heurística en concreto destinada para su resolución. Por el contrario, si el problema a optimizar es multi-objetivo, hay que utilizar una metaheurística en concreto y realizar la ejecución de la misma en múltiples ocasiones para obtener una solución cercana a la óptima. La utilización de esta metaheurística, proporcionará un conjunto de soluciones donde unas dominarán sobre otras y donde el conjunto de soluciones óptimas para la resolución del problema propuesto se denomina frente óptimo.

Obtener este conjunto de resultados tan óptimos es una tarea complicada por tanto, el conjunto de soluciones que se elige para la resolución del problema es el conjunto más cercano a este frente y es lo que se denomina **Frente de Pareto**.

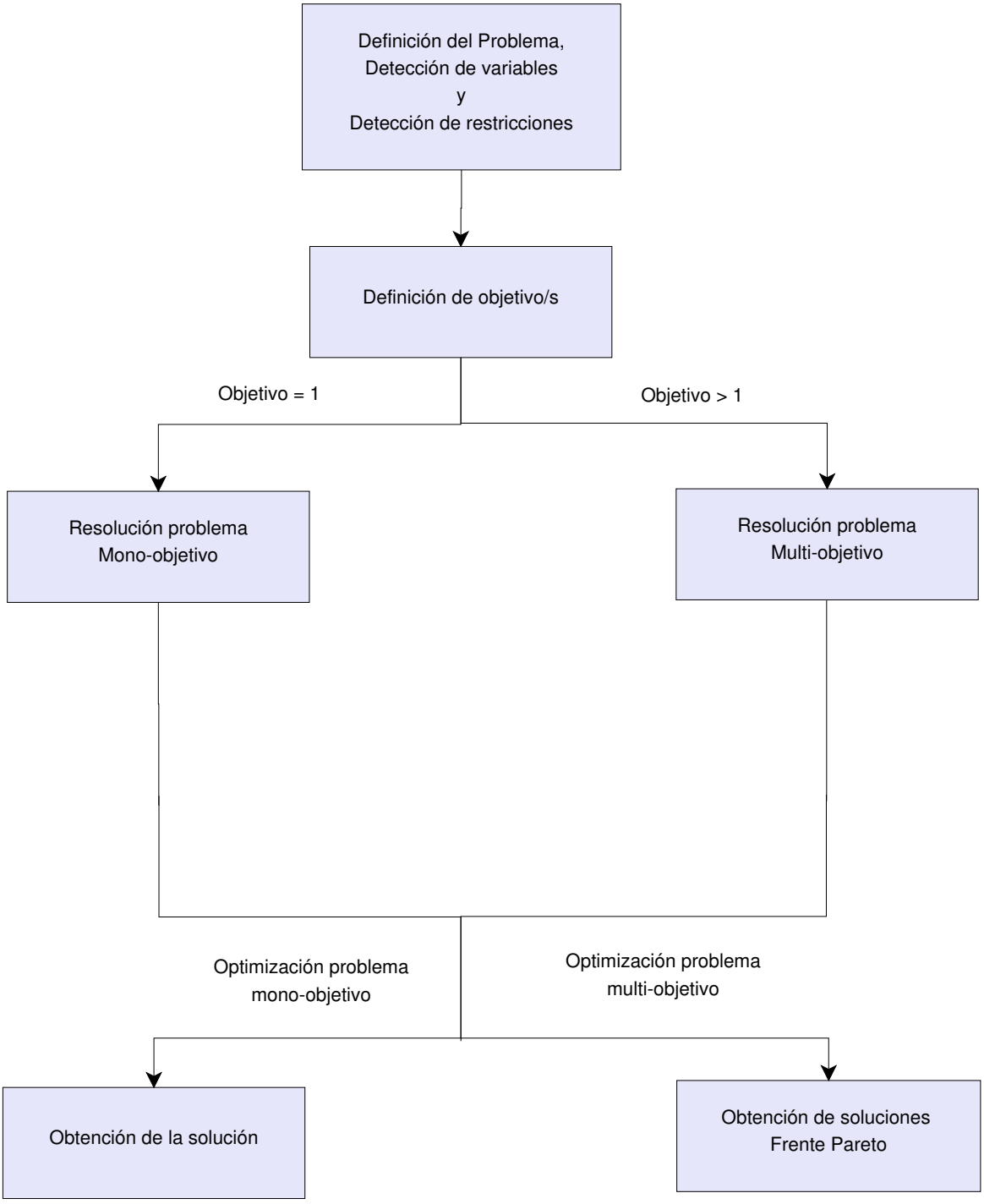

Figura 2.1: Proceso de análisis y resolución del problema

2.2. Métricas

Entre las numerosas métricas que existen para comprobar el rendimiento de los algoritmos utilizados para la optimización de problemas multi-objetivo, la más utilizada es la métrica del hipervolumen, o también conocida como *S-metric* o medida de Lebesgue. En Zitzler et al. [63, 64] se define la métrica del hipervolumen de la siguiente manera:

$$HV(S,R) = volume(\bigcup_{i=1}^{|S|} v_i)$$

Lo que quiere decir esta fórmula, es que por cada solucion $\vec{s_i} \in S$, un hipercubo v_i es construido con el conjunto de referencia y la solución $\vec{s_i}$ como la diagonal entre las esquinas de los hipercubos.

El hipervolumen es una métrica unitaria (recibe como parámetro un único conjunto A para ser evaluado) que mide cuánto del espacio objetivo es dominado por un conjunto no dominado. Para realizar el cálculo del hipervolumen y comprobar el espacio cubierto por el conjunto, se necesita un punto de referencia (ver Figura 2.2). Este punto de referencia, normalmente se calcula con la peor solución del conjunto para los diferentes objetivos, es decir, si el frente dado está normalizado (valores entre 0-1), el punto de referencia suele ser (1,1) en el caso de que sea un problema multi-objetivo con dos objetivos.

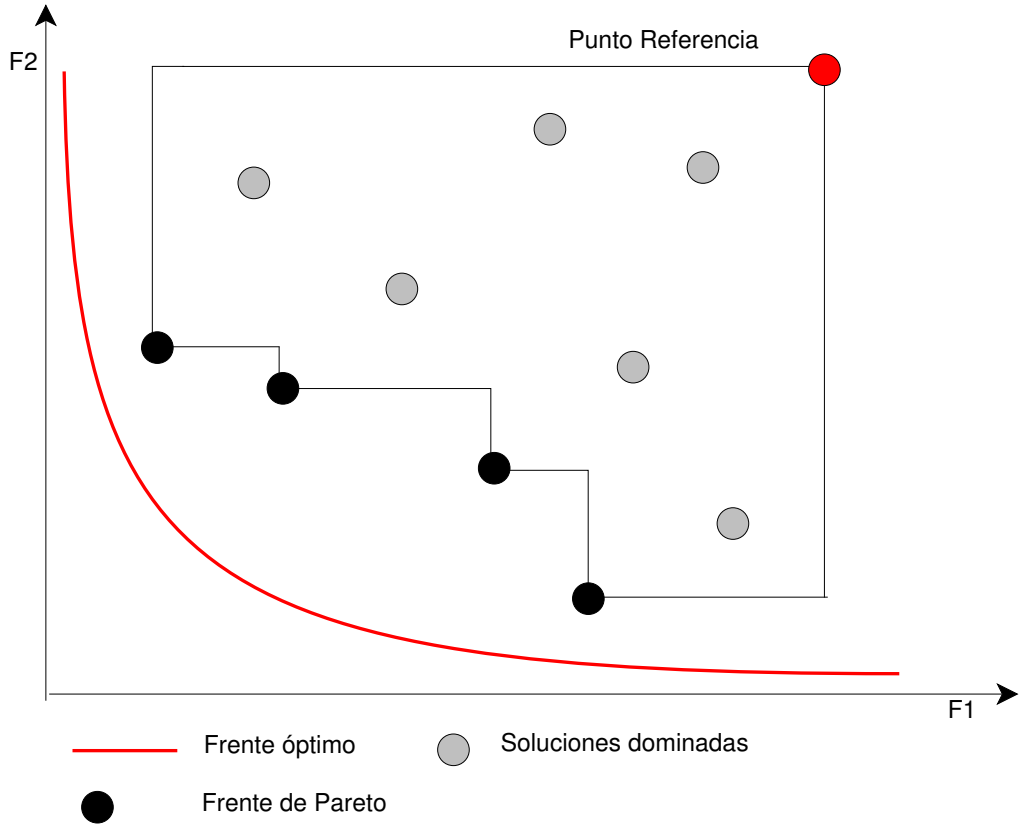

Figura 2.2: Representación del hipervolumen

Como se puede apreciar en la imagen, se posee una serie de soluciones dominadas, el punto de referencia, el frente de Pareto, el frente de Pareto óptimo y el hipervolumen resultante. El frente de Pareto es el conjunto de puntos o soluciones obtenidos, donde no existe otro punto que mejore el resultado de alguno de los objetivos sin empeorar el otro. Sin embargo, el frente de Pareto óptimo, sería el conjunto de soluciones más óptimas para resolver el problema dado. Llevando esto a la realidad es muy complicado conseguirlo y lo que se realiza es aproximar esta solución óptima, o lo que se denomina conjunto de aproximación [65].

2.2.1. Análisis estadístico

En este apartado, se hablará en detalle de cada uno de los tests estadísticos utilizados para dar veracidad a los datos obtenidos por las metaheurísticas. Para la realización de los diferentes test estadísticos (ver Figura 2.3), se utilizó un nivel de confianza por defecto del 95 % siendo este modificable por el usuario a la hora de realizar el experimento con la aplicación.

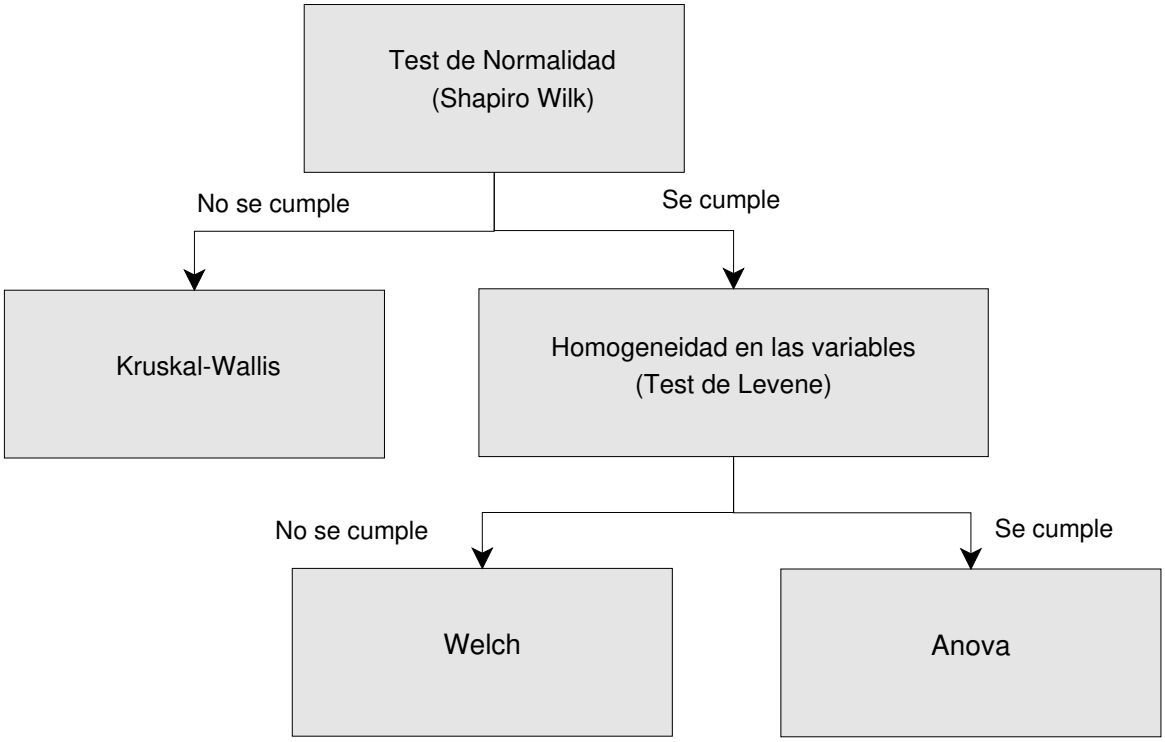

Figura 2.3: Proceso de análisis estadístico

Test Shapiro Wilk: esta prueba es un test de normalidad y se utiliza para comprobar si las variables se distribuyen normalmente como la población. El estadístico se denota como *Wc*. La variable se distribuye normalmente si se cumple $Wc = p \geqslant 0{,}05$. En caso contrario, las variables no se distribuyen normalmente.

Test Kruskal-Wallis: este análisis estadístico es utilizado para comprobar si un grupo de datos proviene de la misma población, no asumiendo la distribución normal. En este test, se tienen en cuenta las medianas de las variables. El estadístico se denota con la letra *k*. Lo deseable en este test estadístico es que las medianas sean diferentes o al menos una lo sea, en caso contrario, se dice que los datos no proceden de la misma población.

Test Levene: este test estadístico, es utilizado para comprobar la homogeneidad entre las varianzas de los grupos. El estadístico es denotado con la letra *F*. Si al hacer Levene existe homogeneidad entre las varianzas $F = p \geqslant 0{,}05$ se pasa a realizar el análisis de varianzas *ANOVA*, en caso contrario, se realizará la corrección a través de *Welch*.

$$\text{Existe homogeneidad:} \quad \sigma_1^2 = \sigma 2^2 \tag{2.1}$$

$$\text{No existe homogeneidad:} \quad \sigma_1^2 \neq \sigma 2^2 \tag{2.2}$$

Test Anova: probablemente es el test más utilizado en el ámbito de la estadística. Este test es utilizado para comprobar si existen diferencias significativas entre las varianzas del grupo de datos ó para comprobar si la variable independiente tiene un efecto significativo sobre la variable dependiente. El estadístico se denota con la letra *F*. Lo deseable para este test estadístico es que haya diferencias significativas entre las varianzas del grupo, en ese caso se dice que $F = p < 0{,}05$.

Test Welch: este test estadístico se utiliza cuando se cumple *Shapiro Wilk* pero no se cumple *Levene*. Se utiliza para corregir los grados de libertad de la variable para que exista homogeneidad entre las varianzas. De esta forma, una vez se han corregido los grados de libertad de la variable comprobar si siguen siendo las varianzas heterogéneas u homogéneas. El estadístico se denota con la letra *t*.

2.3. Aplicaciones existentes

Para la comparación del proyecto propuesto y otras herramientas, se ha basado principalmente en la búsqueda de herramientas que comparen metaheurísticas que resuelven problemas multi-objetivo. Este tipo de herramientas se pueden dividir en dos tipos: herramientas webs y herramientas de escritorio, donde utilizan frameworks para la optimización multi-objetivo con metaheurísticas. Cabe destacar, que al realizar la búsqueda de herramientas similares, se ha encontrado un número de aplicaciones reducidas que estén accesibles al público, ya que la gran mayoría son de uso docente o privativo.

- **Guimoo** [13]: es una aplicación libre que permite analizar los resultados en problemas de optimización multi-objetivo. Esta aplicación, permite seleccionar métricas para mostrar cuantitativamente y cualitativamente el rendimiento de las evaluaciones. Con la elección de estas métricas y los resultados proporcionados por parte del uso de frameworks de algoritmos evolutivos, es posible realizar el análisis y la comparativa de los algoritmos utilizados para la resolución del problema. Alguna de las métricas que permite Guimo evaluar son las siguientes:

  - **S-Metric o hipervolumen** [64]
  - **R-Metric** [19]
  - **Contribución** [38]
  - **Entropía** [14]
  - **Distancia generacional** [57]
  - **Espaciado** [51]
  - **Tamaño de espacio dominado**
  - **Cobertura de dos conjuntos y diferencia de cobertura** [20]

- **ParadisEO** [40] : es un framework white box [58] de C++ orientado a objetos que permite utilizar metaheurísticas para la resolución de problemas mono y multi-objetivos. Está compuesto por cinco componentes (EO, MO, MOEO, SMP y PEO).

  - **EO**: Es el componente que permite usar metaheurísticas basadas en poblaciones.
  - **MO**: Este componente permite realizar la resolución de problemas mediante el uso de metaheurísticas de resolución de problemas mono-objetivo.
  - **MOEO**: Con este componente se permite utilizar metaheurísticas en la optimización de problemas multi-objetivo.
  - **SMP y PEO**: permite utilizar metaheurísticas para la optimización de problemas en paralelo, distribuidos o mixtos.

  En [33] hablan en detalle del uso del framework ParadisEO y la comparación de éste con otros frameworks disponibles. Además, detallan que para realizar la comparación de diferentes metaheurísticas y mostrar los resultados de manera gráfica, se podría utilizar una herramienta libre dedicada al análisis de optimización multi-objetivo llamada **Guimoo**.

- **JMetal** [43]: es un framework Java para optimización multi-objetivo utilizando metaheurísticas. Este framework, permite la resolución de los problemas mediante el uso de los algoritmos multi-objetivo más conocidos. Debido a que la arquitectura que se usa en Jmetal es orientada a objetos, el framework puede ser utilizado para la implementación de un algoritmo propio integrandolo en esta librería. En [12] hablan con más detenimiento de este framework y de las capacidades que posee el mismo además de la comparativa que hacen de diferentes metaheurísticas utilizando Jmetal para ello.

  A partir de este framework, se ha creado otro plugin de Jmetal desarrollado para C++ llamado Jmetalcpp. En él, se integran las mismas características que posee la librería para Java, pero incluyéndolo en proyectos desarrollados para el lenguaje C++ [34].

  Alguno de los algoritmos que soporta esta librería son los siguientes:

  - **Metaheurísticas multi-objetivo** : NSGA-II [10], ssNSGA-II, GDE3, SMPSO, SMPSOhv, OMOPSO [54], PAES [30], SMS-EMOA, MOCHC y MOEA/D.
  - **Metaheurísticas mono-objetivo** : generational GA (gGA), steady-state GA (ssGA), DE, PSO, PSO, PSO y CMA-ES.

  Además, Jmetal permite cargar una serie de problemas para testear los algoritmos mono y multi-objetivo que se hayan diseñado. Algunos de estos problemas son los siguientes: Fonseca [15], Kursawe [31], Schaffer, Srinivas, Tanaka, Rastrigin, Rosenbrock, ZDT benchmark [60], DTLZ benchmark [29], LZ09 benchmark [32], CEC2005 benchmark.

- **METCO** [6]: Meta-heuristic-based Extensible Tool for Cooperative Optimisation es una herramienta tipo plugin para la optimización multi-objetivo que implementa algunos de los algoritmos evolutivos para la resolución de problemas multi-objetivos más conocidos (NSGA-II, SPEA2, IBEA [61], etc.). Está desarrollado para su uso en aplicaciones desarrolladas en C++ y permite la ejecución tanto en paralelo como de manera secuencial.

- **MOEAFramework** [9]: es una librería de Java Open Source que permite desarrollar aplicaciones experimentando con optimización multiobjetivo, mediante el uso de algoritmos evolutivos y otro tipo de algoritmos de optimización. Los algoritmos que soporta esta librería son los siguientes: NSGA-II [10], NSGA-III, ϵ-MOEA, GDE3, PAES [30], PESA2, SPEA2 [62], IBEA [41], SMS-EMOA, SMPSO, OMOPSO [54], CMA-ES y MOEA/D. En su web oficial, se puede descargar una aplicación demo en la que muestra los diferentes usos que se pueden realizar con esta librería

mediante diferentes programas desarrollados usando esta librería. El último de ellos, es una demostración que permite realizar una comparación de diferentes metaheurísticas eligiendo un tipo de métrica. Además de mostrar al usuario en tiempo real el rendimiento de los diferentes algoritmos, cuando finaliza la ejecución de los diferentes algoritmos compararados hay una opción para comprobar estadísticamente el rendimiento de los diferentes algoritmos seleccionados.

- **STATService**: es un conjunto de herramientas para el análisis estadístico, que permiten realizar de manera online o en una aplicación de escritorio mediante el uso de análisis estadísticos la resolución de ciertos problemas propuestos. En [1] detallan el uso de este tipo de herramientas para la comparación de metaheurísticas mediante los análisis estadísticos utilizando este tipo de herramientas. Además, analizan las diferencias entre utilizar este tipo de componentes y la implementación de una aplicación de escritorio con la utilización de frameworks de optimización con metaheurísticas.

- **Entorno web para la comparación de metaheurísticas inspiradas en inteligencia de enjambre** [4]: este proyecto realizado por la Universidad Complutense de Madrid, es una herramienta web que permite ejecutar distintas metaheurísticas, que se basan en la inteligencia de enjambre y mostrar comparaciones, mediante gráficas de los resultados obtenidos por las diferentes metaheurísticas. La similitud con esta aplicación está en la comparación de metaheurísticas. En este proyecto, la comparación de las metaheurísticas también se realizará mediante gráficas y tablas, además, también se mostrará la comparación de las mismas mediante los resultados obtenidos en los test estadísticos a los que será sometido cada una de las metaheurísticas.

Capítulo 3

Tecnologías, diseño e implementación

Para la realización del TFG se han analizado multitud de herramientas, de las cuales no se han utilizado todas para la implementación del mismo. A continuación, se detallan las diferentes tecnologías analizadas para el desarrollo del proyecto divididas en los diferentes ámbitos de uso.

3.1. Tecnologías para back-end

- **Django** [11]: es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como Modelo–vista–controlador. El código fue liberado en el año 2005.

- **NodeJs** [26]: es un entorno de ejecución para JavaScript en el lado servidor construido con el motor JavaScript V8 de Chrome [17]. Usa un modelo orientado a eventos que lo hace eficiente. El ecosistema que usa NodeJs es el gestor de paquetes NPM [23], que es el ecosistema más grande de librerías de código abierto en el mundo.

- **Python** [44]: es un lenguaje de programación interpretado, cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

- **R** [45]: es un lenguaje de programación enfocado al análisis estadístico. Es uno de los lenguajes más utilizados por la comunidad estadística para la investigación, siendo muy popular en el campo de minería de datos, bioinformática y matemáticas financieras. R, permite cargar gran cantidad de paquetes o librerías desarrolladas por la comunidad con multitud de funcionalidades.

3.2. Tecnologías para front-end

- **Javascript**: JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor.

- **MaterializeCss** [35]: framework web destinado al front-end que permite crear entornos webs responsivos, y adaptando el diseño de material design de Google a un entorno web.

- **React** [46]: librería Javascript creada por Facebook, para construir interfaces de usuario, que permiten crear aplicaciones SPA *Single Page Application* más eficientes y funciona tanto en el lado cliente como en el servidor, haciendo posible la creación de aplicaciones isomórficas.

- **Shiny** [49]: framework para aplicaciones webs desarrolladas con el lenguaje R. Shiny es un proyecto de Rstudio [48] el cual permite realizar una interfaz web sin necesidad de usar HTML, CSS o Javascript para el correcto funcionamiento de la aplicación. Permite con unos simples scripts en R [45] mostrar una interfaz web con la salida de los diferentes scripts ejecutados. Esta herramienta, se analizó para la posibilidad de integrar todo el proyecto en torno al lenguaje de programación R, ya que el proyecto se basa principalmente en utilizar una serie de tests estadísticos para la validación de las diferentes metaheurísticas a analizar y este lenguaje de programación es muy potente en este ámbito.

3.3. Tecnologías para la generación de documentación

- **Gitbook** [21]: Herramienta para crear de una manera fácil e intuitiva la documentación de nuestro proyecto al igual que permite publicarlo al público. Gitbook es una herramienta de línea de comando (y librería para NodeJs) para crear el libro de nuestro proyecto utilizando Github/git [22] y el lenguaje de marcado Markdown. Además, también permite crear el libro de manera local y alojarlo en su portal oficial.

- **JupyterNotebook** [28]: herramienta utilizada por la comunidad científica y en la ciencia de computadores, para crear cuadernillos propios donde

se puede implementar código (Python2.x o Python 3.x, R o Julia). La posibilidad de utilizar Jupyter en este proyecto, consistiría en la posibilidad de exportar los resultados obtenidos tras utilizar la aplicación a un cuadernillo propio, es decir, una vez el usuario utilice la aplicación y obtenga unos resultados, trasladar los mismos a un cuadernillo y tener gestionado mediante informes las conclusiones obtenidas por el usuario.

- **Mybinder** [5]: herramienta web que permite crear una dirección web, a partir de una dirección de repositorio de Github, para mostrar y ejecutar de manera online el Notebook de Jupyter creado.
- **Thrive Notebook** [56]: herramienta web que permite crear Notebooks de manera similar a Jupyter, con la diferencia de que si se quiere trabajar en un notebook creado por algún usuario de la comunidad, se crea una copia exacta del mismo al perfil y se trabaja directamente en el cuadernillo sin necesidad de tener que descargar un fichero *.ipynb*.

3.4. Librerías analizadas para el desarrollo de la aplicación

- **jQuery** [27]: es una biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.
- **Plotly** [42]: herramienta web que permite realizar, de manera online, el análisis de datos y mostrar los resultados mediante gráficos online. Además, Plotly está disponible en forma de librerías o paquetes para los lenguajes de programación como Python, R, Matlab, Perl, Julia, Arduino y REST.
- **ChartJs** [7]: es una librería open source creada por la comunidad para la visualización de datos mediante gráficos. Esta librería utiliza la etiqueta canvas de HTML5 para renderizar el contenido del gráfico.
- **Django-jchart** [37]: es una extensión de ChartJs para la renderización de los gráficos generados por esta librería en una plantilla desarrollada en el framework web Django. Permite crear los mismos tipos de gráficos que se crean en la librería ChartJs pero bajo código Python. Es compatible con versiones de Python 2.7 y 3.5 y versiones superiores a la 1.5 de Django.
- **Numpy** [52]: librería de Python que permite agregar un mejor soporte para el tratamiento de vectores y matrices.

- **Scipy.stats** [53]: módulo de la librería Scipy de Python que contiene un gran número de distribuciones de probabilidad además de numerosas funciones estadísticas.
- **Matplotlib** [36] librería de Python para generar gráficas a partir del tratamiento de datos almacenados en arrays o vectores.
- **Pandas** [39]: librería de Python que se utiliza para el análisis y tratamiento de los datos.

3.5. Tecnologías utilizadas

A continuación, se detallarán las tecnologías que finalmente fueron escogidas para el desarrollo de la aplicación.

3.5.1. Elección de lenguaje de programación

Una vez analizados los diferentes lenguajes de programación acordes para la realización del proyecto, finalmente, los seleccionados han sido el lenguaje de programación Python y Javascript. En concreto, la versión elegida de Python para la realización del proyecto completo ha sido la version 2.7. El motivo de la elección de este lenguaje, principalmente es por la facilidad de trabajar con test estadísticos, la capacidad de realizar aplicaciones webs mediante la utilización de frameworks y la posibilidad de integrar todo lo necesario para la aplicación a realizar con este mismo lenguaje de programación.

Sin embargo, para la conexión entre el lado cliente y el lado servidor de la aplicación y manipulación de la interfaz de usuario, se ha utilizado el lenguaje de programación Javascript utilizando en concreto características de ES6. La principal causa por la que se eligió este lenguaje de programación, se debe a que gracias a las librerías que posee, se permite realizar una interfaz de usuario agradable y dinámica además de permitir la conexión entre lado cliente y lado servidor de una manera fácil y potente.

3.5.2. Frameworks

Los frameworks utilizados para la realización de la aplicación han sido el framework web Django y el framework CSS MaterializeCss.

Debido a que el proyecto casi al completo se realizaría sobre el lenguaje de programación Python, era recomendable utilizar Django como framework web ya que permite crear aplicaciones webs utilizando este lenguaje de programación. Gracias al sistema de vistas, templates, formularios y modelos que posee Django, es posible crear una interfaz web muy potente y muy visual utilizando para ello sólo código Python y no depender tanto del lenguaje de marca HTML.

Para la creación de una aplicación con diseño responsivo y visual para el usuario, se utilizó el framework CSS MaterializeCss. Este framework, permite añadir características similares al material design de Google a aplicaciones webs, permitiendo hacer una interfaz usable y elegante al usuario.

3.5.3. Librerías seleccionadas

Las librerías seleccionadas para la realización de la aplicación están divididas en librerías de lado cliente y las librerías utilizadas en el lado servidor.

Para el lado cliente, se ha utilizado la librería Jquery de Javascript. Gracias a esta librería, es posible realizar animaciones en la interfaz de usuario además de permitir el envío de datos desde el lado cliente al lado servidor utilizando peticiones Ajax.

Para el lado servidor, se han utilizado las diferentes librerías científicas que se utilizan en el lenguaje de programación Python. Las librerías utilizadas han sido:

- **Numpy**
- **Scipy.stats**
  - **f_oneway**
  - **shapiro**
  - **levene**
  - **ttest_ind**
  - **kruskal**
- **Pandas**
- **Django-jchart**

3.5.4. Tecnologías para documentación

Debido a que todo el proyecto ha sido realizado con el control de versiones Git, se ha utilizado como herramienta de documentación del mismo la herramienta Gitbook. Con Gitbook, es posible documentar todo lo que se realiza en el proyecto además de permitir publicar al público el libro realizado con la herramienta. De esta manera, al tener todo enfocado a la utilización de Git y Gitbook, es posible obtener un feedback por parte de la comunidad en caso de producirse errores de implementación de la aplicación o falta documentación.

Capítulo 4

Metanalyze

4.1. Descripción

En este apartado, se explicará cómo se ha creado la aplicación y el funcionamiento de la misma. Se muestran las diferentes pantallas que aparecen mientras se está realizando un experimento con la aplicación y además, se añaden los resultados obtenidos tras realizar dicho experimento.

Figura 4.1: Logotipo Metanalyze

El uso de la aplicación está dividido en varias fases:

- **Descripción de los experimentos realizados**: en esta fase es donde se detallan los datos de los experimentos realizados, es decir, es donde se indica el nombre de los algoritmos utilizados para el experimento, el número de objetivos del problema resuelto, las ejecuciones realizadas, el paso de evaluación y la condición de parada.
- **Elección de la métrica**: una vez se han añadido los datos del experimento realizado, se indicará el tipo de métrica que se quiere utilizar para comparar el rendimiento de los algoritmos utilizados y el análisis a aplicar.
    - **Rendimiento mínimo**

  - **Rendimiento medio**
  - **Rendimiento máximo**

- **Visualización y descarga de los datos**: la última de las fases del uso de la herramienta será la de visualización y descarga de los resultados obtenidos.

Para la implementación de la aplicación, se ha utilizado el framework web Django. Con Django, es posible realizar una aplicación web desarrollada completamente con código Python. Además, gracias a las posibilidades que posee el lenguaje de programación en sí y la API que posee Django, es posible realizar aplicaciones webs potentes y sencillas.

4.2. Implementación de la herramienta

Para la creación del formulario principal se ha utilizado la *Apiforms* que posee Django. Mediante el uso de esta API que está integrada en Django, es posible crear un formulario con una simple clase y con sus atributos. Además, este formulario creado tendrá el mismo aspecto que un formulario desarrollado integramente en HTML. El formulario creado se llama **AlgorithmForm** y tiene una serie de atributos que van desde botones de radio hasta campos de texto.

Para gestionar los datos que se introducen en el formulario y realizar las diferentes operaciones para el correcto funcionamiento de la aplicación, se ha utilizado la API de Django de modelos. Estos modelos, al igual que con los formularios, es posible crearlos simplemente creando una clase y añadiendo los atributos con el tipo de dato que puede guardarse. Además, permite añadir la longitud máxima de caracteres permitido en caso de que sea un campo de texto. La API de modelos, al hacer una "migración" al gestor de base de datos que se vaya a utilizar, traduce los datos que contenga la clase del modelo a código SQL.

Los modelos que se han creado para la aplicación son los siguientes:

- **Configuration**. Modelo que guarda la configuración del experimento a realizar. Los campos que completan este modelo son los siguientes:
  - **nAlgorithms**: campo que almacenará el número de algoritmos a comparar.
  - **test**: campo que guardará si se realizarán los tests estadísticos.
  - **nObjectives**: campo que guarda el número de objetivos.
  - **nExecutions**: campo que guarda el número de ejecuciones realizadas con cada uno de los algoritmos a comparar.
  - **step**: tamaño del paso utilizado en los ficheros de datos para mostrar las soluciones obtenidas por los algoritmos.

  - **stopCondition**: campo que guarda el paso en el que se paró el algoritmo para la resolución del problema planteado.
  - **dataOutput**: campo que guarda la elección del usuario para la salida de los datos del experimento (tabla, gráficos o ambas).
  - **bound**: campo tipo lista que guardará el límite que quiere el usuario analizar (mínimo, media, máximo) para cada algoritmo.
  - **metric**: campo tipo lista que guardará la métrica o las métricas que el usuario quiera utilizar para analizar los algoritmos.

- **Algorithms**. Modelo que guardará la configuración de cada algoritmo a analizar en el experimento. Los campos que completan este modelo son los siguientes:

  - **algorithm**: este campo de texto guardará el nombre que se le asigne al algoritmo.
  - **idAlgorithm**: este campo será el que guarde el identificador que le asigne el usuario al algoritmo, es decir, un "alias" que el usuario quiera asignarle al algoritmo.
  - **fileName**: campo que guardará el nombre de fichero asociado al algoritmo.
  - **nVariablesAlgorithm**: campo que guardará el número de variable que posee el problema abordado por los algoritmos.
  - **configuration**: este es un campo que se le asigna el identificador de la configuración del análisis a realizar. Es decir, que para cada ejecución de la aplicación, se utilizará una configuración diferente con un número de algoritmos a comparar. Este identificador es el que indica a qué experimento pertenece este algoritmo. En caso de que se elimine esta configuración de la base de datos, se eliminarán los algoritmos que pertenecen a esta configuración del análisis.
  - **hypervolumeList**: es un campo tipo lista que guardará los valores de hipervolumen del algoritmo.

- **MinAvgMaxChartModel**. Este modelo guardará los datos que se necesitan para mostrar la salida del gráfico con el mínimo, la media y el máximo. Está compuesto por los siguientes campos:

  - **configuration**: este es un campo que se le asigna el id númerico de la configuración a realizar, siendo una clave foránea. En caso de que se elimine la configuración perteneciente a este algoritmo, se hace un borrado en cascada para no almacenar datos que no se utilizarán en ningún momento.

  - **listValues**: campo que guardará un diccionario con los datos de los gráficos divididos por los límites.

- **MinChartModel**. Modelo que guarda los datos que se necesitan para mostrar los datos del gráfico que muestran el rendimiento mínimo. Al igual que el modelo de `MinAvgMaxChartModel`, tiene los campos `listValues` y `configuration`.

- **AvgChartModel**. Modelo que guarda los datos que se necesitan para mostrar los datos del gráfico de medias. Al igual que el modelo de `MinAvgMaxChartModel`, tiene los campos `listValues` y `configuration`.

- **MaxChartModel**. Modelo que guarda los datos que se necesitan para mostrar los datos del gráfico de rendimiento máximo. Al igual que el modelo de `MinAvgMaxChartModel`, tiene los campos `listValues` y `configuration`.

- **StatisticDataframeTex**. Modelo el cual guardará los datos de la comparación estadística de los algoritmos para generar el fichero **.tex**. Al tener varios formatos de salida de datos diferentes, es necesario tener los datos separados en diferentes modelos. Al igual que los modelos anteriores, tiene los campos `listValues` y `configuration`.

- **StatisticDataframeTxt**. Modelo el cual guardará los datos para generar el fichero **.txt**. Al guardar los mismos datos que el fichero **.tex** pero con símbolos diferentes, está compuesto por los mismos campos que el modelo `StatisticDataframeTex`.

- **StatisticDataframeHtml**. Modelo el cual guardará los datos para generar la tabla que se mostrará en la vista de resultados con el resultado de la comparación estadística entre los diferentes algoritmos. Utiliza los mismos campos que los modelos anteriores.

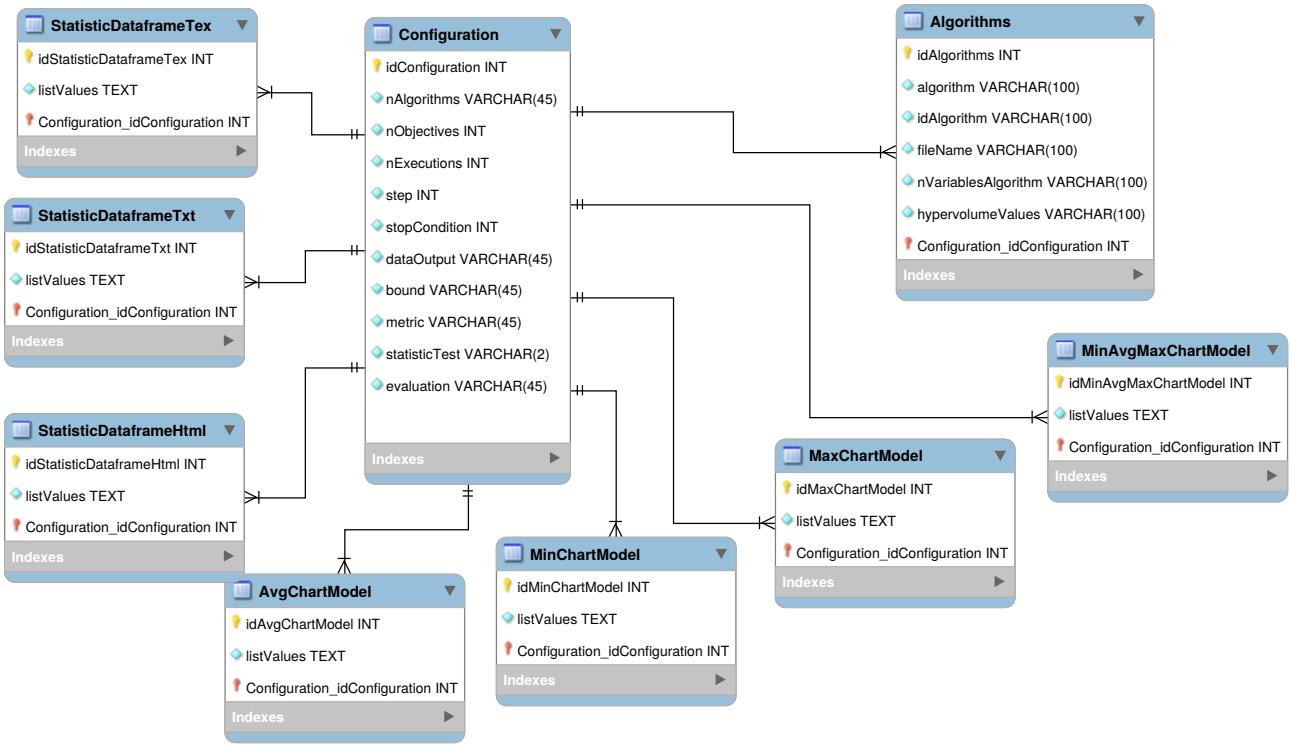

Figura 4.2: Modelos realizado para la aplicación

Debido a que Django sigue el modelo vista controlador, es necesario utilizar un sistema de vistas para renderizar el contenido de la aplicación. Para la creación de las mismas, se ha creado una clase por cada vista de la aplicación. Las vistas creadas son las siguientes:

- **index**: vista que se encarga de mostrar el formulario principal, donde el usuario añadirá la configuración pertinente al experimento a realizar (número de algoritmos, salida de datos, límites, steps, stop condition, etc). Para el correcto funcionamiento de esta vista, se creó una plantilla (archivo *.html*) con el contenido de la misma y se configuró el patrón de URL que posee Django.

- **about**: vista que muestra información acerca de la aplicación web que se está utilizando.

- **results**: esta es la vista en la que se muestran los resultados obtenidos por el experimento. En la misma, se muestran la/s salida/s de datos (gráfico,tabla o ambas) los botones para la descarga de los mismos, etc. Esta vista, es la más trabajada debido a que dependiendo del tipo de request o respuesta que se realice (POST o GET), se hará una operación u otra. Esta vista retorna en formato diccionario los elementos que haya seleccionado en el apartado de configuración (index) el usuario.

- **download**: esta es una vista que simplemente lo que hace es la posibilidad

de que el usuario pueda descargar los resultados de las tablas en formato Latex, Txt y Pdf.

4.3. Inicio

La pantalla inicial que aparece al cargar la web es la pantalla de configuración. En esta ventana, el usuario añade toda la configuración del experimento a realizar. Está compuesto por un formulario dinámico, donde se seleccionan los campos y se añaden las configuraciones que el usuario desee. La vista está compuesta por:

- **Configuración de algoritmos**
- **Selección de la métrica**
- **Elección de realizar los test estadísticos**
- **Configuración de la ejecución de los algoritmos**
- **Formato de salida de los datos**

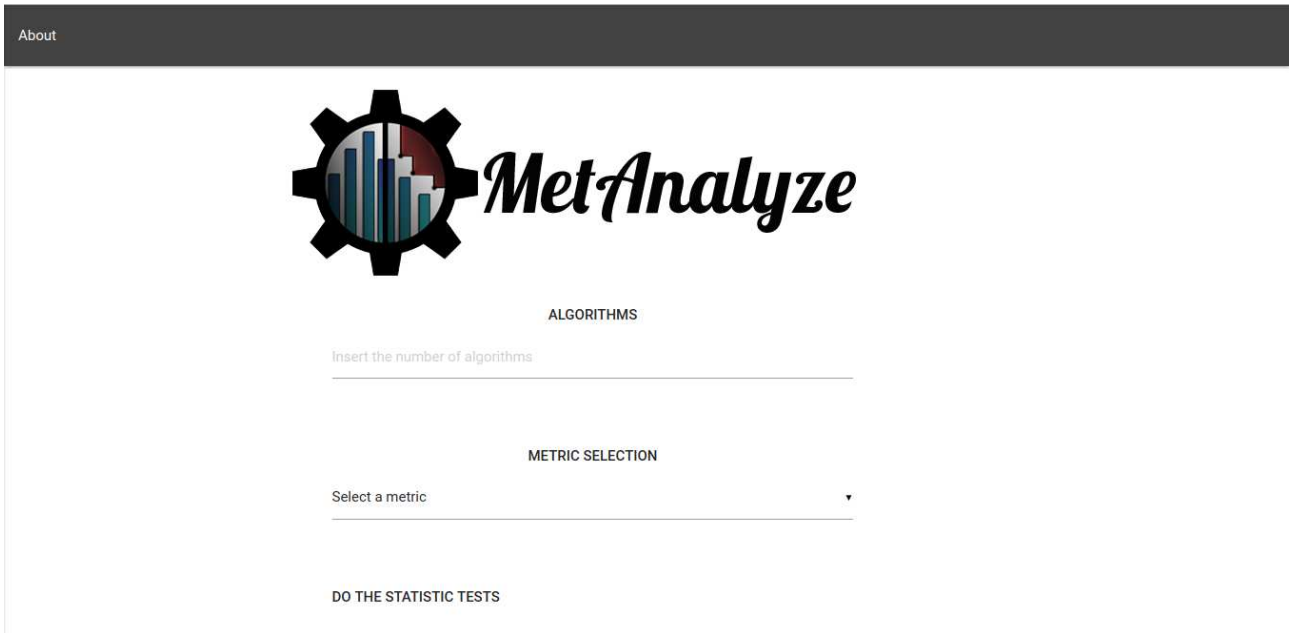

Figura 4.3: Pantalla de inicio de la aplicación

4.4. Algoritmos

En la primera sección de la vista se realizará la configuración del número de algoritmos a comparar. En este apartado, se añadirá el nombre de cada uno de los algoritmos a insertar, un identificador de los mismos y el fichero de configuración de los mismos. Los ficheros con el contenido de las evaluaciones de los diferentes algoritmos deben tener extensiones **.zip**, **.tar** y **.tar.gz**, cualquier otro formato no será admitido. Además, el formato del fichero **.dat** ó **txt**, o formato de texto que el usuario haya creado con las evaluaciones, tendrá que tener el siguiente formato:

- El paso debe ir indicado con un **#stepX**; donde X es el número del paso ó simplemente **XXX(XXX)** donde X sería el paso en forma numérica. Por ejemplo:

```
#Paso1
   0.59012207739 0.0432129750187
   0.0269166867501 0.917201716522
   0.446679651528 0.786439705731
   0.821910962048 0.746322048856
   0.75222895892 0.863565653673
   200(200)
   0.59012207739 0.0432129750187
   0.0269166867501 0.917201716522
   0.446679651528 0.786439705731
   0.821910962048 0.746322048856
   0.75222895892 0.863565653673
```

El usuario debe realizar esta operación para cada uno de los algoritmos que quiera realizar la comparación.

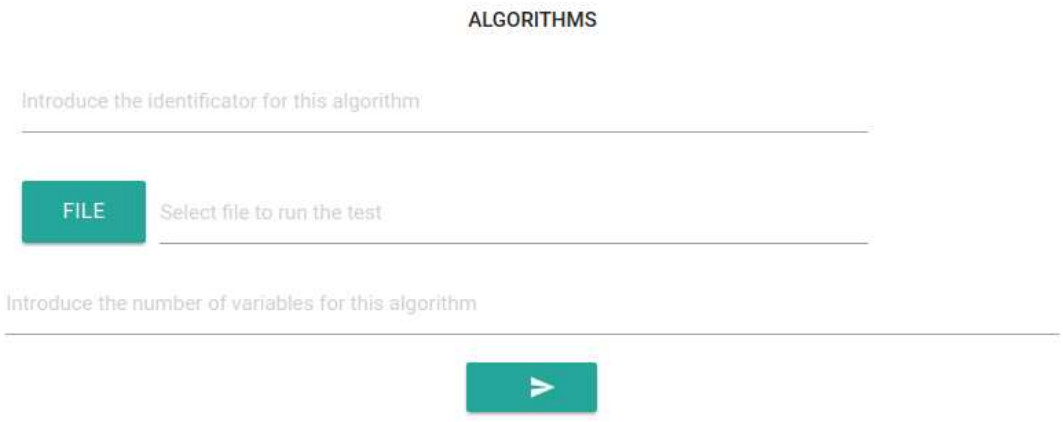

Figura 4.4: Configuración del algoritmo a comparar

4.4.1. Métrica

Una vez el usuario introduce la configuración de las metaheurísticas ejecutadas, deberá seleccionar la métrica a utilizar para comparar el rendimiento de los diferentes algoritmos. Una vez seleccionada la métrica, aparecerá una opción para calcular el rendimiento mínimo, el rendimiento medio o el rendimiento máximo de los diferentes algoritmos a comparar.

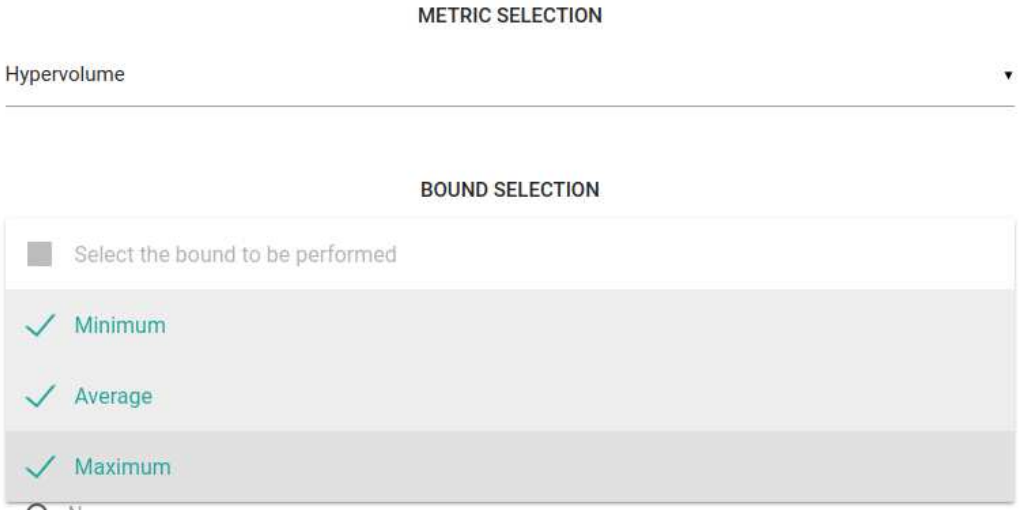

Figura 4.5: Selección del análisis de rendimiento del hipervolumen

A continuación, el usuario deberá seleccionar si desea realizar los test estadísticos o no. Como se explica en capítulos anteriores, estos tests estadísticos se utilizan para dar una veracidad a los datos obtenidos.

4.5. Configuración de la ejecución

En la sección de *configuración de la ejecución*, se introduce la información de cómo se realizó la ejecución de los diferentes algoritmos, para la resolución del problema. Este apartado del formulario está dividido en varias partes:

- **Número de objetivos**: en este campo de texto se introduce el número de objetivos que tenía el problema de optimización a resolver.
- **Número de ejecuciones**: se insertará el número de ejecuciones que se realizó para los diferentes algoritmos aplicados a la resolución del problema.
- **Número de pasos**: se introducirá el tamaño del paso que indica la frecuencia (medida en tiempo de evaluaciones o tiempo), donde el algoritmo volcó un conjunto de soluciones.

- **Condición de parada**: se introduce el paso en el que se paró la ejecución del algoritmo.

Finalmente, el usuario deberá seleccionar qué formato de salida quiere para la realización del experimento. Los formatos pueden ser tres:

- **Formato tabla**: se mostrarán los resultados del rendimiento de cada algoritmo en una tabla.
- **Formato gráfico**: los resultados se mostrarán por gráficos divididos por el tipo de rendimiento que el usuario haya seleccionado *min, max y avg*. En caso de que el usuario seleccione los tres, se mostrará también un gráfico en el que se muestra el rendimiento de todos los algoritmos comparados.
- **Tabla y gráfico**: el experimento se mostrará en formato tabla y en gráficos, dividiendo los resultados por el tipo de rendimiento que el usuario haya introducido.

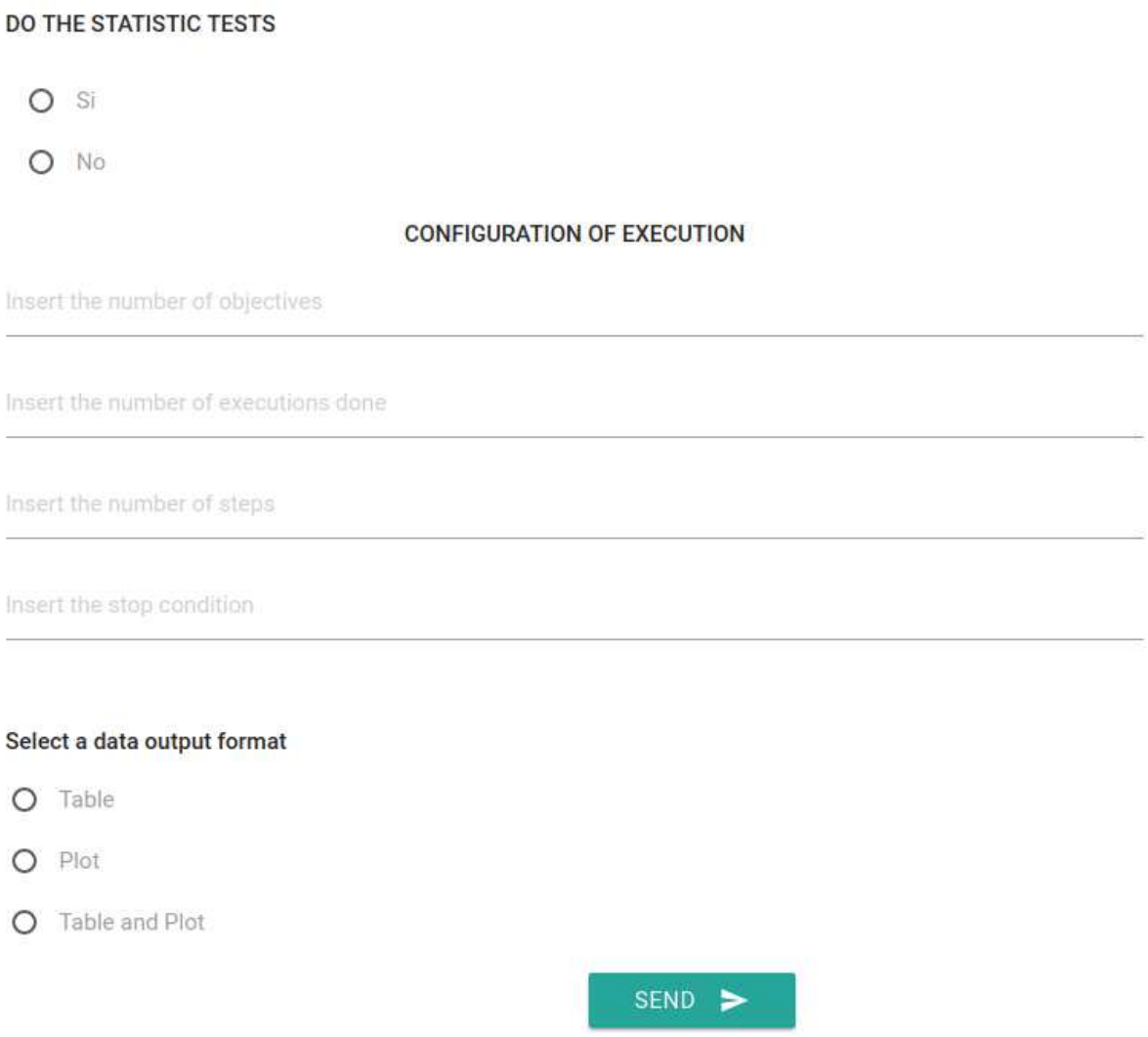

Figura 4.6: Configuración de la ejecución del algoritmo

4.6. Resultados obtenidos

Una vez el usuario ha introducido todos los campos de configuración de la realización del experimento y da a la opción de enviar, se subirán los datos al servidor. A continuación, se realizará el parseo de los ficheros, se guardarán los datos de los frentes divididos por algoritmos y por número de pasos. El siguiente paso, será calcular el punto de referencia para realizar el cálculo del hipervolumen a cada uno de los algoritmos (este punto de referencia es común para todos). Una vez se realiza el cálculo del hipervolumen, se normalizan los datos para mostrarlos en escala 0-1 y guardar los datos en los modelos. El siguiente paso, será realizar los tests estadísticos en caso de que el usuario haya seleccionado la opción. Una vez realizados, los datos se guardarán en una tabla para mostrar los resultados al usuario. Finalmente, con los datos de la métrica calculada de cada uno de los algoritmos, si el usuario ha seleccionado *gráfico* como salida de datos, se realizará la creación de los gráficos para mostrar los resultados. Sin embargo, si ha seleccionado *tabla* como salida de datos, se realizará la creación de las diferentes tablas para mostrarlas y generar los ficheros *txt*, *tex* y *pdf*.

Cuando se realizan todas estas operaciones, se mostrará una vista con la salida de datos que el usuario haya elegido.

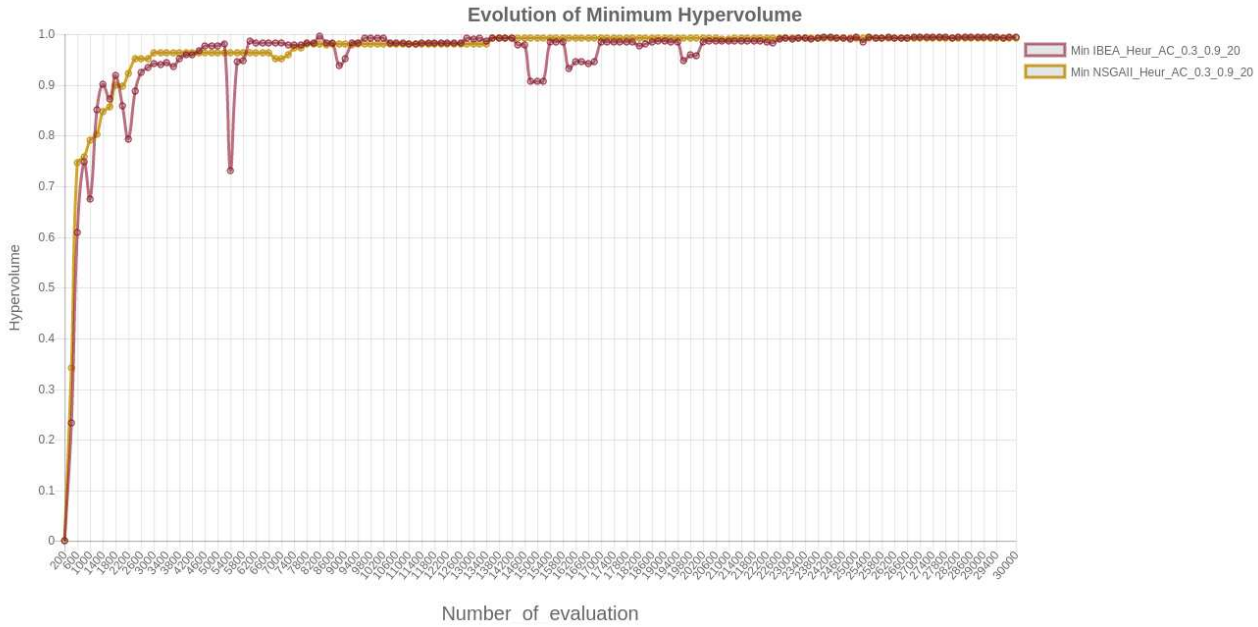

Figura 4.7: Gráfico de hipervolumen mínimo

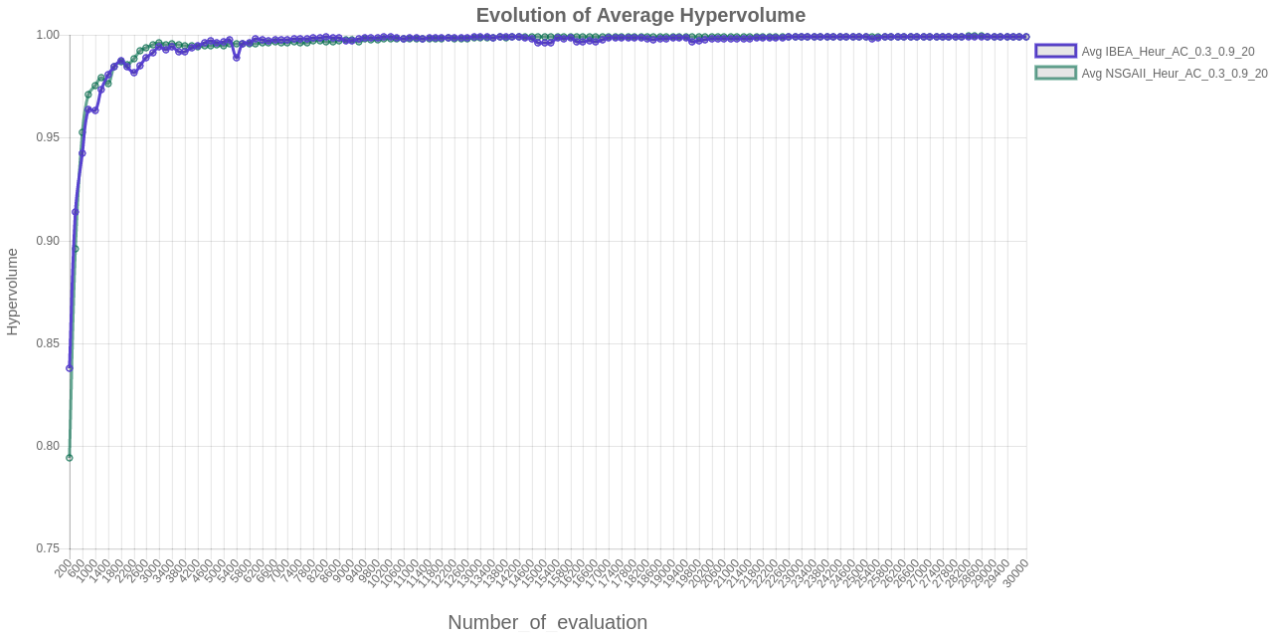

Figura 4.8: Gráfico de hipervolumen medio

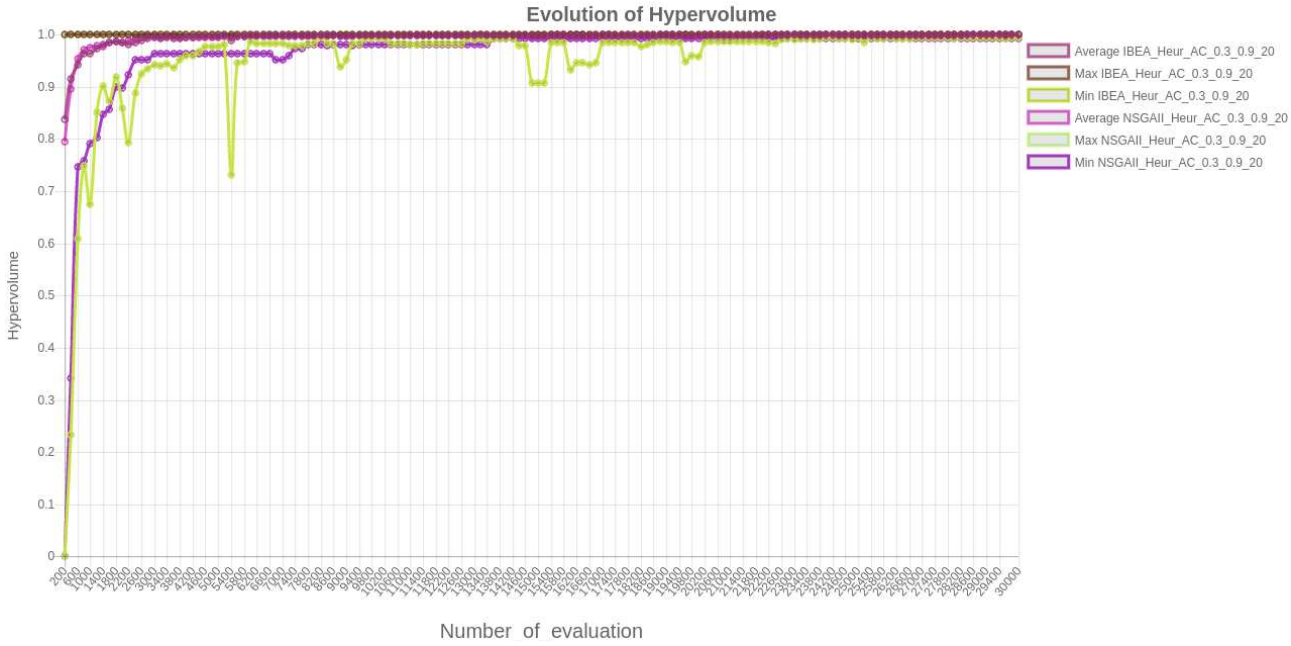

Figura 4.9: Gráfico de hipervolumen medio

	IBEA	NSGAII	SPEA
IBEA	↔	↑	*
NSGAII	↓	↔	↓
SPEA	*	↑	↔

Figura 4.10: Tabla de resultados estadísticos

La figura 4.7 muestra el gráfico resultante de la evolución del hipervolumen mínimo del experimento realizado. En esta ocasión, sólo se realizó la comparativa de dos algoritmos diferentes con la misma configuración y la misma heurística.

La figura 4.8 muestra el resultado gráfico de la evolución del hipervolumen medio del experimento.

Sin embargo, la figura 4.9 muestra el resultado de la comparativa de los dos algoritmos con la evolución del hipervolumen. En esta ocasión, se muestra tanto el hipervolumen mínimo, medio y máximo de los diferentes algoritmos comparados. Este tipo de gráfico sólo se mostrará en caso que el usuario seleccione que quiere realizar la evolución del hipervolumen mínimo, medio y el máximo.

Una función a destacar en los gráficos, es la posibilidad que tiene el usuario de eliminar la visualización de ciertos datos. Para ello, lo único que tiene que hacer es seleccionar en la leyenda el resultado que no quiere observar en la gráfica y añadirá una línea en la descripción de la leyenda y se borrarán los resultados en el gráfico.

La figura 4.10 muestra el resultado estadístico obtenido al realizar el experimento con la aplicación. Para la realización del mismo, todos los algoritmos han sido comparados estadísticamente entre ellos. Posteriormente, para mostrar los resultados obtenidos en una tabla se utilizan una serie de iconos. El significado de los iconos es el siguiente:

- ↔: este símbolo, se utiliza para demostrar que un algoritmo no es estadísticamente ni mejor ni peor al comparado. Además, este icono se utiliza también para la comparación del algoritmo consigo mismo ya que esta comparativa no se realiza.
- ↑: este símbolo, se utiliza para demostrar que un algoritmo es estadísticamente mejor que el algoritmo comparado.
- ↓: sin embargo, este símbolo es utilizado para mostrar que estadísticamente el algoritmo comparado tiene resultados peores.

- $*$: finalmente, este símbolo denota que la comparación entre los diferentes algoritmos es un caso anormal, es decir, no es un caso que se pueda definir ni mejor ni peor estadísticamente. Para la representación de este caso, se analizan tanto las medias y las medianas de los algoritmos a comparar. Mientras que para el resto de casos se utilizan únicamente las medianas de los algoritmos a comparar.

4.7. Guardar experimento

Cuando se muestran los resultados del experimento realizado, se le dará la oportunidad al usuario de poder descargar los datos obtenidos. El formato de los ficheros para los gráficos será de un fichero **.png** con el gráfico seleccionado. En caso de que quisiera descargar los resultados de las tablas, se descargará un fichero **.tar** con los resultados del experimento realizado. Este fichero **.tar**, estará compuesto por las tablas con los resultados divididos por algoritmo en formato **.tex** y **.pdf** y **.txt**.

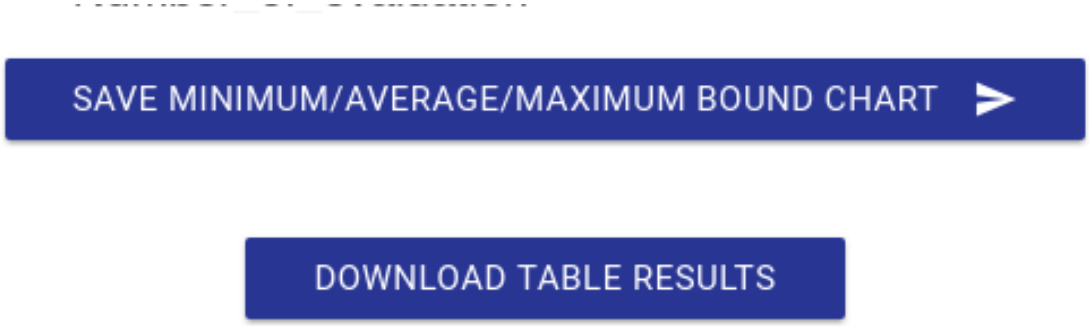

Figura 4.11: Botones de descarga de los gráficos y tablas

Capítulo 5

Conclusiones y líneas futuras

5.1. Conclusiones

En la actualidad, existe una gran cantidad de problemas que son difíciles de resolver. Gracias al rápido crecimiento de la tecnología, y sobre todo gracias al rápido avance en la ciencia de computadores, es posible resolver este tipo de problemas complejos utilizando las medidas pertinentes. Utilizando estas medidas para buscar soluciones óptimas a los problemas complejos de la humanidad, es posible hallar soluciones a problemas que antes era completamente imposible de encontrar.

Por ejemplo, utilizando algoritmos evolutivos, heurísticas para resolver ciertos problemas, etc., es posible obtener respuestas desde el campo de la ciencia de datos hasta resolver problemas que perjudiquen a la humanidad. Aunque el uso de este tipo de algoritmos no proporciona una solución esperada, es posible obtener un resultado muy similar al óptimo para la resolución de los problemas. Para llegar a obtener este tipo de soluciones, es necesario un tiempo de cómputo largo y de un gran análisis comparativo entre los resultados de los diferentes algoritmos.

Con la aplicación desarrollada, cualquier usuario con grandes o pequeños conocimientos en el ámbito de la optimización de problemas multi-objetivo, puede obtener en una misma aplicación si los resultados que se han obtenido por parte del uso de metaheurísticas son fiables o no. Además, es posible analizar y comparar los resultados obtenidos por las diferentes metaheurísticas para la resolución de un mismo problema mediante la contrastación de los datos estadísticamente.

Finalmente, con la posibilidad que se tiene de mostrar y guardar los resultados en formato tabla e imagen, es posible compartir los datos de una manera más rápida y fiable.

5.2. Líneas futuras

El trabajo futuro en esta aplicación está enfocado a la mejora de las características de la misma y a permitir realizar experimentos, tanto en problemas de optimización mono-objetivo y multi-objetivo, ya que actualmente sólo se enfoca en los problemas multi-objetivo con un número máximo de dos objetivos. Las propuestas para mejorar la aplicación serían las siguientes:

- **Añadir experimentos con problemas mono-objetivo**: actualmente, la aplicación sólo permite realizar una comparativa de metaheurísticas para la resolución de problemas multi-objetivo. Por tanto, sería adecuado permitir realizar una comparativa para algoritmos que sean utilizados para la resolución de problemas de optimización mono-objetivo.

- **Mejorar los resultados obtenidos**: los resultados obtenidos por el experimento sólo se pueden descargar en formato **.png** para los gráficos. Una opción a añadir o modificar la existente, sería la posibilidad de descargar los resultados gráficos en formato **.eps**, ya que este formato es más modificable y es más utilizado en documentos de investigación.

- **Añadir nuevas métricas**: actualmente, sólo se permite seleccionar la métrica del hipervolumen. Una modificación posible, sería la posibilidad de añadir nuevas métricas para realizar un experimento más completo.

- **Posibilidad de guardar resultados en Jupyter Notebook**: una línea futura podría ser la inclusión de esta opción al obtener los resultados del experimento. Esta opción, permitiría al usuario la posibilidad de compartir los resultados obtenidos al realizar el experimento con la aplicación desarrollada.

- **Añadir nuevas librerías de gráficos**: actualmente, se utiliza la librería django-jchart para la visualización de los gráficos. El tipo de gráficos que permite esta librería para mostrar los resultados está muy limitado, una propuesta de mejora sería la posibilidad de añadir una nueva librería que permitiera generar más tipos de gráficos que la actual.

- **Realizar una aplicación de escritorio**: asimismo, se podría realizar una aplicación de escritorio que tuviera las mismas características con una interfaz similar.

- **Desplegar la aplicación**: para que el uso de la aplicación sea accesible a todos los usuarios, sería recomendable desplegar la aplicación en un dominio y seguir actualizándola.

Capítulo 6

Summary and Conclusions

Nowadays, there are many problems that are difficult to solve. Thanks to the rapid growth of technology and especially with the growth of computer science, it is possible to resolve this kind of problems with a several actions. Using this actions to resolve complex problems, can be possible to find an optimal solutions to human problems that have never been solved.

For example, using evolutionary algorithms, heuristics, and so on, it is possible to get answer from data science to harmful human problems.

Although this type of algorithms does not provides an expected solution, it is possible to obtain a very similar result to the optimal one for the resolution of the problems. In order to obtain these types of solutions, a computation time is necessary and a large comparative analysis between the results of the different algorithms.

With the application developed, any user with advanced or basic knowledge in the multiobjective optimization problem solving environment, can obtain in the same application whether the results of metaheuristics are reliable or not. In addition, it is possible to analyse and compare the results of the metaheuristics used to solve a problem by contrasting the results with the statistical tests.

Finally, with the possibility of show and download the results in table and image format, it is possible to share the results faster and more reliable.

Capítulo 7

Presupuesto

Para la realización de este Trabajo de Fin de Grado, el trabajo ha sido dividido en una serie de fases, las cuales serán detalladas a continuación. Además, para la realización de cada una de estas fases en las que se ha dividido el trabajo, se han utilizado una serie de horas, siendo detalladas en el presupuesto del TFG. Las fases de realización del trabajo han sido:

- **Análisis del proceso de experimentación**: para la realización del Trabajo de Fin de Grado, era necesario entender las prácticas habituales a la hora de realizar comparativas de metaheurísticas. Para entender este proceso de experimentación, se prestó especial atención en las aplicaciones existentes y, a partir de ello, saber cuáles son los factores más importantes a implementar en la herramienta.

- **Identificación del proceso de análisis para algoritmos multi-objetivo**: para entender el proceso de análisis de los algoritmos para la resolución de problemas multi-objetivo, se realizó una recopilación de las diferentes métricas que se utilizan a la hora de comparar algoritmos que manejan soluciones multi-objetivo.

- **Estudio de tests estadísticos**: para entender los diferentes tests estadísticos que se implementarían en la herramienta, fue necesario estudiarlos y entender el paso a paso para realizarlos todos.

- **Documentación sobre el lenguaje de programación**: debido al escaso uso del lenguaje de programación utilizado, fue necesario documentarse para realizar el desarrollo de la aplicación. Al usar el lenguaje de programación Python y utilizar su framework web, fue necesario documentarse bastante para entender el correcto funcionamiento y de las posibilidades del mismo.

- **Documentación sobre las librerías utilizadas**: la documentación de las librerías de gráficos para el framework web Django son muy escasas. Por tanto, fue necesario invertir un gran número de horas para entender el

correcto funcionamiento de las mismas en Javascript y poder implementar sus funciones con el lenguaje de programación Python.

- **Desarrollo de la aplicación**: para el desarrollo de la aplicación se utilizó el lenguaje de programación Python, el framework web Django y las librerías para la interfaz de usuario y la demostración de los gráficos.
- **Documentación**: se realizó la memoria del TFG y la documentación extra que englobó al mismo.

Estipulando un precio total de 16 € por hora de trabajo, el presupuesto final queda dividido de la siguiente manera:

Fase del TFG	**Número de horas**
Análisis del proceso de experimentación	20
Proceso de análisis para algoritmos multi-objetivo	20
Estudio de test estadísticos	10
Documentación lenguaje de programación y framework	30
Documentación sobre las librerías utilizadas	10
Desarrollo de la aplicación	140
Documentación del TFG	30
Número total de horas realizadas	**260**
Equipamiento informático	**600 €**
Presupuesto total	**4760 €**

Tabla 7.1: Presupuesto y horas dedicadas

Bibliografía

[1] Parejo Jose Antonio, Jorge García, Antonio Ruiz-Cortés, and Jose Cristobal Riquelme. Statservice: Herramienta de análisis estadístico como soporte para la investigación con metaheurísticas. In *Actas del VIII Congreso Expañol sobre Metaheurísticas, Algoritmos Evolutivos y Bio-inspirados*, 2012.

[2] M. Arriaza. *Guía Práctica de análisis de datos.* 2006.

[3] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, Oxford, UK, 1996.

[4] Gómez Carrero Beatriz, Corrales Daniel Rodríguez, and García Ávila Federico. Entorno web para la comparación de metaheurísticas inspiradas en inteligencia de enjambre. Proyecto de Sistemas Informáticos (Facultad de Informática, Curso 2013-2014), 2014.

[5] Binder. Binder Project. http://mybinder.org/. Online; accedida 19 Enero 2017.

[6] León C., Miranda G., and Segura C. Metco: A parallel plugin-based framework for multi-objective optimization. *International Journal on Artificial Intelligence Tools*, 18(4):569–588, 2009.

[7] ChartJs. ChartJs Documentation. http://www.chartjs.org/docs/. Online; accedida 1 Mayo 2017.

[8] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[9] Hadka David and other contributors. http://moeaframework.org/. Online; accedida 16 Enero 2017.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Transaction Evolutionary Computation*, 6(2):182–197, April 2002.

[11] Django. DjangoProject Documentation. https://docs.djangoproject.com/en/1.10/. Online; accedida 16 Enero 2017.

[12] Juan J Durillo and Antonio J Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42:760–771, 2011.

[13] Talbi El-Ghazali, Tantar Emilia, and Van Den Hekke Ulrich. GUIMOO. http://guimoo.gforge.inria.fr/. Online; accedida 05 Abril 2017.

[14] A. Farhang-Mehr and S. Azarm. Diversity assessment of pareto optimal solution sets: an entropy approach. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 1, pages 723–728, May 2002.

[15] C. M. Fonseca and P. J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. ii. application example. *Trans. Sys. Man Cyber. Part A*, 28(1):38–47, January 1998.

[16] S. García M. *La guía definitiva de django*. 2015.

[17] Google. ChromeV8. https://developers.google.com/v8/. Online; accedida 10 Enero 2017.

[18] D. Greenfeld-Roy and A. Greenfeld-Roy. *Two Scoops of Django*. 2015.

[19] M. P. Hansen and A. Jaszkiewicz. *Evaluating the quality of approximations to the non-dominated set*. IMM, Department of Mathematical Modelling, Technical Universityof Denmark, 1998.

[20] T. Hiroyasu, M. Miki, and S. Watanabe. Divided range genetic algorithms in multiobjective optimization problems. *Proc. of IWES*, 99:57–65, 1999.

[21] Gitbook Inc. Gitbook. https://toolchain.gitbook.com/. Online; accedida 30 Marzo 2017.

[22] Github Inc. Github. https://github.com/. Online; accedida 10 Enero 2017.

[23] Npm Inc. Npmjs. https://www.npmjs.com/. Online; accedida 10 Enero 2017.

[24] Stack Exchange Inc. Stackoverflow. http://stackoverflow.com/. Online; accedida 10 Enero 2017.

[25] Siwei Jiang, Yew-Soon Ong, Jie Zhang, and Liang Feng. Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Trans. Cybernetics*, 44(12):2391–2404, 2014.

[26] Joyent. NodeJs. https://nodejs.org/es/docs/. Online; accedida 10 Enero 2017.

[27] Jquery. Jquery API. http://api.jquery.com//. Online; accedida 31 Enero 2017.

[28] Jupyter. JupyterNotebooks Documentation. https://jupyter.readthedocs.io/en/latest/index.html. Online; accedida 19 Enero 2017.

[29] Deb K, Thiele L, Laumanns M, and Zitzler E. Scalable test problems for evolutionary multi-objective optimization. zurich. Technical report, Switzerland, Tech. Rep. 112, 2001.

[30] J. Knowles and D. Corne. The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1, 1999.

[31] Frank Kursawe. *A variant of evolution strategies for vector optimization*, pages 193–197. Springer Berlin Heidelberg, Berlin, Heidelberg, 1991.

[32] Hui Li and Qingfu Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302, 2009.

[33] Arnaud Liefooghe, Matthieu Basseur, Laetitia Jourdan, and El-Ghazali Talbi. ParadisEO-MOEO: A Framework for Evolutionary Multi-objective Optimization. In *Evolutionary Multi-Criterion Optimization (EMO 2007)*, volume 4403 of *Lecture Notes in Computer Science (LNCS)*, pages 386–400, Matsushima, Japan, Feb 2007.

[34] Esteban López-Camacho, María Jesús García Godoy, Antonio J. Nebro, and José F. Aldana-Montes. jmetalcpp: optimizing molecular docking problems with a c++ metaheuristic framework. *Bioinformatics*, 30(3):437–438, 2014.

[35] Materialize. MaterializeCss. http://materializecss.com/. Online; accedida 16 Enero 2017.

[36] Matlab. Matplotlib Library. http://matplotlib.org/contents.html. Online; accedida 22 Febrero 2017.

[37] Heimensen Matthisk. django-jchart github repository. https://github.com/matthisk/django-jchart. Online; accedida 1 Mayo 2017.

[38] H. Meunier, E. G. Talbi, and P. Reininger. A multiobjective genetic algorithm for radio network optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation.*, volume 1, pages 317–324 vol.1, 2000.

[39] Pandas. Pandas Library. http://pandas.pydata.org/pandas-docs/version/0.19.1/. Online; accedida 22 Febrero 2017.

[40] ParadisEO. ParadisEO. http://paradiseo.gforge.inria.fr/. Online; accedida 04 Abril 2017.

[41] D. H. Phan and J. Suzuki. R2-ibea: R2 indicator based evolutionary algorithm for multiobjective optimization. In *2013 IEEE Congress on Evolutionary Computation*, pages 1836–1845, June 2013.

[42] Plotly. PlotlyChart Documentation. https://plot.ly/feed/. Online; accedida 19 Enero 2017.

[43] Maven project. Jmetal. http://jmetal.sourceforge.net/. Online; accedida 04 Abril 2017.

[44] Python. Phyton Documentation. https://docs.python.org/2/. Online; accedida 16 Enero 2017.

[45] R. R-project. https://www.r-project.org/other-docs.html. Online; accedida 16 Enero 2017.

[46] React. React Documentation. https://facebook.github.io/react/docs/hello-world.html. Online; accedida 10 Enero 2017.

[47] Nery Riquelme, Christian Von Lücken, and Benjamín Barán. Performance metrics in multi-objective optimization. In Hector Cancela, Alex Cuadros-Vargas, and Ernesto Cuadros-Vargas, editors, *2015 XLI Latin American Computing Conference (CLEI)*, pages 286–296, Arequipa-Peru, October 2015. CLEI, CLEI.

[48] RStudio. Rstudio. https://www.rstudio.com/. Online; accedida 16 Enero 2017.

[49] RStudio. ShinyProject. https://shiny.rstudio.com/reference/shiny/latest/. Online; accedida 16 Enero 2017.

[50] Cheng S., Shi Y., and Qin Q. On the performance metrics of multiobjective optimization. In *Tan Y.,Shi Y.,Ji Z.(eds) Advances in Swarm Intelligence*, volume 7331 of *ICSI 2012*, pages 504–512, Berlin, Heidelberg, 2012. Springer.

[51] J. R. Schott. Fault tolerant design using single and multicriteria genetic algorithm optimization. Technical report, DTIC Document, 1995.

[52] Scipy. Numpy Package. https://docs.scipy.org/doc/. Online; accedida 22 Febrero 2017.

[53] Scipy. Statisical Functions. https://docs.scipy.org/doc/scipy-0.18.1/reference/stats.html. Online; accedida 22 Febrero 2017.

[54] Margarita Reyes Sierra and Carlos A. Coello Coello. Improving pso-based multi-objective optimization using crowding, mutation and ϵ-dominance. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, EMO'05, pages 505–519, Berlin, Heidelberg, 2005. Springer-Verlag.

[55] Simpy. Simpy. http://simpy.readthedocs.io/en/latest/contents.html. Online; accedida 22 Febrero 2017.

[56] Thrive. Thrive Notebooks. https://notebook.thrive.to/. Online; accedida 31 Enero 2017.

[57] D. A. Van Veldhuizen. Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations. Technical report, Evolutionary Computation, 1999.

[58] WhiteBox. WhiteBox. http://s.sudre.free.fr/WBWhiteBoxIntro.html/. Online; accedida 04 Abril 2017.

[59] G. G. Yen and Z. He. Performance metric ensemble for multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 18(1):131–144, Feb 2014.

[60] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

[61] E. Zitzler and S. Künzli. Indicator-Based Selection in Multiobjective Search. In X. Yao et al., editors, *Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *LNCS*, pages 832–842. Springer, 2004.

[62] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. Technical report, Swiss Federal Institute of Technology (ETH), 2001.

[63] E. Zitzler and L. Thiele. *Multiobjective optimization using evolutionary algorithms — A comparative case study*, pages 292–301. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[64] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Nov 1999.

[65] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.