ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

ÁREA DE CIENCIA DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

TRABAJO FIN DE MÁSTER

METAHEURÍSTICAS APLICADAS AL PROBLEMA DE PLANIFICACIÓN DE CÉLULAS DE SOLDADURA ROBOTIZADAS

D. ARIAS ALVAREZ, DIANNE

TUTOR: D. PUENTE PEINADOR, JORGE

COTUTOR: D. VARELA ARIAS, RAMIRO

FECHA: JULIO 2021

Contenido

1. Introducción

El desarrollo de la industria del automóvil y el aumento de la fabricación de equipos industriales han hecho que el número de robots utilizados en los procesos de soldadura sea cada vez mayor. Cada vez se ven robots industriales con mayores prestaciones dinámicas y de precisión gracias a la mejora constante en los sistemas mecánicos, sensoriales y de control de los robots. Esto ha llevado a una expansión de las aplicaciones robotizadas de soldadura. La planificación de las órdenes llevadas a cabo por estos robots es cada vez más compleja por la necesidad de cumplir con los plazos impuestos por los clientes, así como la necesidad de reducir costes en cuando al uso de recursos.

Dentro de las múltiples empresas que se dedican a este sector se encuentra Soldavigil, empresa especializada desde sus orígenes en la fabricación de componentes para la industria de automoción. Los procesos implantados abarcan desde la fabricación de utillajes (conjunto de instrumentos y herramientas que optimizan la realización de las operaciones de proceso de fabricación necesarios para el correcto posicionamiento y fijación de todas las piezas a soldar) al suministro de componentes pintados y ensamblados. Se especializa en el curvado de tubos CNC, en la estampación y la soldadura robotizada de componentes para lo cual disponen de las secciones de matricería, estampación, mecanizado, curvado, plegado y soldadura robotizada. En las figuras 1 y 2 podemos observar una mesa de soldadura (con perforaciones milimétricas), con utillajes (piezas negras y amarillas atornilladas a la mesa).

Figura 1. Mesa de soldadura con utillajes.

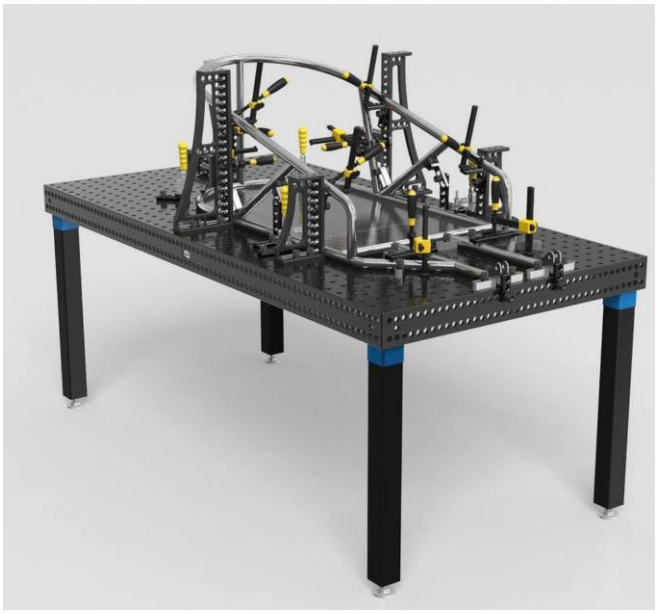


Figura 2. Mesa con utillaje.

La soldadura es el principal valor de la empresa utilizando las últimas técnicas y los equipos más avanzados para poder estar en vanguardia en materia de soldadura. Para una mejor comprensión del proceso de soldaduras se recomienda el siguiente video ilustrativo donde al inicio un operario fija la pieza con el utillaje, a partir de ese momento el robot sigue su programación para soldar la pieza: https://www.youtube.com/watch?v=t_UAyEIpKks

Actualmente todo el proceso de planificación de las órdenes dentro de los robots se realiza manualmente lo que puede ser un proceso tedioso, esto da pie a un problema de planificación multiobjetivo. La optimización multiobjetivo ha sido popular en la investigación y la aplicación y se han presentado muchos algoritmos evolutivos multiobjetivo. También han surgido varios framework de optimización multiobjetivo que simplifican mucho el trabajo de los investigadores tal como EVA2(Kronfeld, 2011), OPT4j (Lukasiewycz et al., 2011), MOEAT (Sağ & Çunkaş, 2009) y JMetal (Durillo & Nebro, 2011). En esta investigación se utiliza este último debido a las grandes ventajas que brinda.

Para trabajar con el framework es necesario contar con un problema, pero al inicio de la investigación no se tenía una definición del problema ni descripción alguna del mismo, por lo que para poder comenzar a trabajar se decide buscar algún problema conocido para comenzar a trabajar. El problema seleccionado es un problema que forma parte de un caso de aprendizaje basado en proyectos en el marco de un proyecto coordinado en el Máster Universitario en Ingeniería Informática de la Universidad de Oviedo (Usamentiaga et al., n.d.). La idea era resolver este problema multiobjetivo conocido utilizando el framework jMetal para sentar la base de la investigación realizada sobre el framework y luego aplicar todo lo aprendido en un problema real como lo es la planificación de células de soldadura robotizada. Se inicia la investigación comprendiendo el jMetal, todas sus funcionalidades, algoritmos y operadores. Luego se modela el problema de los drones y se crea un primer prototipo de interfaz donde dado un conjunto de datos se optimiza utilizando algunos de los algoritmos brindados por el framework y se muestran los

resultados obtenidos. Por otro lado, también se lleva a cabo una experimentación para ver las ventajas que brinda el framework en este sentido.

Solo hasta hace muy poco se nos presenta la oportunidad de tener un encuentro con la empresa y finalmente tener una primera aproximación al problema. Sobre esa base se modela el problema y se crea un segundo prototipo con la gestión y planificación de órdenes y por último se lleva a cabo una experimentación buscando la mejor configuración de parámetros para la resolución del problema.

Por todo esto la presente investigación está encaminada a la mejora de ese proceso de planificación de células de soldadura robotizada utilizando técnicas de optimización multiobjetivo y el framework JMetal y se divide de la siguiente forma: el apartado dos introduce la optimización multiobjetivo, luego en el apartado tres se habla sobre el framework JMetal dividiéndose en varios subapartados con sus características principales. En un cuarto apartado se explican los algoritmos y los diferentes operadores disponibles en dicho framework para configurar esos algoritmos. A continuación, se introduce el problema de las soldaduras con varios subapartados para explicar el modelo de resolución del problema mediante un algoritmo evolutivo. El apartado sexto explica el modelado del problema de los drones, ya que también fue objeto de estudio en este trabajo. Recogiendo ambos problemas, el apartado séptimo describe el prototipo software propuesto para su resolución. En los apartados octavo y noveno se presenta el presupuesto del proyecto, así como la planificación seguida. El apartado décimo recoge el plan de experimentación propuesto y se explican los cuatro experimentos realizados, así como los resultados obtenidos. Y por último los restantes apartados se dedican a las conclusiones del trabajo, la bibliografía utilizada y los anexos.

2. Problemas de optimización multiobjetivo

En la actualidad la mayoría de los problemas de optimización son de naturaleza multiobjetivo, lo que significa que resolverlos requiere optimizar dos o más funciones u objetivos contradictorios. Estos problemas se conocen como problemas de optimización multiobjetivo (MOP). El óptimo buscado de este tipo de problemas no es una solución única como en el caso mono objetivo, sino un conjunto de soluciones conocido como frente Pareto. Ningún elemento de este conjunto es mejor que los demás para todos los objetivos. Este conjunto se le entrega al experto o persona que toma las decisiones, quien tiene que elegir la mejor solución de compromiso de acuerdo con sus preferencias. Como cualquier problema de optimización, los MOP son difíciles de resolver, además si consideramos los problemas de ingeniería, encontramos con frecuencia que algunas de las funciones que los componen no son lineales y pueden ser computacionalmente costosas de evaluar. En esas situaciones, las técnicas exactas a menudo no son aplicables, por lo que los métodos aproximados son obligatorios. Como en la optimización mono objetivo, las metaheurísticas (Blum & Roli, 2003; Jean-Yvesspotvin, 2010) son algoritmos aproximados para resolver MOP. Entre ellos, los algoritmos evolutivos son muy populares (Coello et al., 2007; Deb, 2011), y algunos de los algoritmos más conocidos en este campo pertenecen a esta clase (por ejemplo, NSGA-II (Deb et al., 2002), PAES (J. D. Knowles & Corne, 2000), SPEA2 (Zitzler et al., 2001). Sin embargo, otras técnicas metaheurísticas también están ganando impulso, como la optimización del enjambre de partículas (Reyes-Sierra & Coello Coello, 2006).

3. Framework jMetal

Continuamente se proponen nuevas técnicas para hacer frente a entornos reales en las investigaciones relacionadas con la optimización multiobjetivo mediante metaheurísticas. En este contexto el uso de frameworks es una herramienta valiosa de apoyo en la implementación de nuevas ideas, facilitada por la reutilización de código de algoritmos implementados, así como la comprensión de técnicas ya existentes. Un ejemplo de esto es el framework jMetal (Durillo & Nebro, 2011) basado en Java diseñado para la optimización multiobjetivo utilizando metaheurísticas muy simple y fácil de usar, flexible y extensible. Existe otras herramientas de software, algunas basadas en Java como por ejemplo EVA2, OPT4j y ECJ u otras como MOEAT (Sağ & Çunkaş, 2009) implementadas en C#. La mayoría de esos framework se centran principalmente en algoritmos evolutivos, y muchos de ellos se centran en la optimización de un solo objetivo, ofreciendo solo extensiones al dominio multiobjetivo.

En el caso de jMetal cuenta con un conjunto de características que lo hacen único y muy interesante de utilizar como por ejemplo poner a nuestra disposición la implementación de una serie de algoritmos modernos de optimización multiobjetivo (Durillo et al., 2006). Por otro lado, incluye un amplio conjunto de problemas de prueba, desde algunos clásicos como Kursawe (Kursawe, 1990) y Schaffer (Schaffer, 1985) hasta algunos con restricciones como Srinivas (Srinivas & Deb, 1994) y Tanaka (Tanaka et al., 1995). También contiene la implementación de los indicadores de calidad más utilizados como por ejemplo Hipervolumen (Zitzler & Thiele, 1999), Spread (Deb et al., 2002), Distancia generacional (Veldhuizen & Lamont, 1998), Distancia generacional invertida (Veldhuizen & Lamont, 1998) y Epsilon (J. D. ; Knowles et al., 2006). Posee diferentes representaciones de variables: binaria, real, real codificada en binario, entero y permutación. Además, brinda soporte para la realización de estudios experimentales, incluida la generación automática de tablas LaTeX con los resultados después de aplicar indicadores de calidad, también una comparación estadística por pares utilizando la prueba de Wilcoxon (Demˇ, 2006) a los

resultados obtenidos, y en R (http://www.r-project.org/) boxplots que resuman esos resultados. Además, jMetal incluye la posibilidad de utilizar varios subprocesos para realizar este tipo de experimentos de tal forma que se puedan ejecutar varias ejecuciones independientes en paralelo utilizando modernas CPUs multinúcleo. Y por último provee un sitio web (http://jmetal.sourceforge.net) que contiene los códigos fuente, el manual del usuario y, entre otra información, los frentes de Pareto de los MOP incluidos, referencias a los algoritmos implementados y referencias a artículos que utilizan jMetal.

jMetal ha sido diseñado siguiendo una arquitectura orientada a objetos que facilita la inclusión de nuevos componentes y la reutilización de los existentes. Proporciona al usuario un conjunto de clases base que se pueden utilizar para diseñar nuevos algoritmos, implementar problemas u operadores complejos, etc.

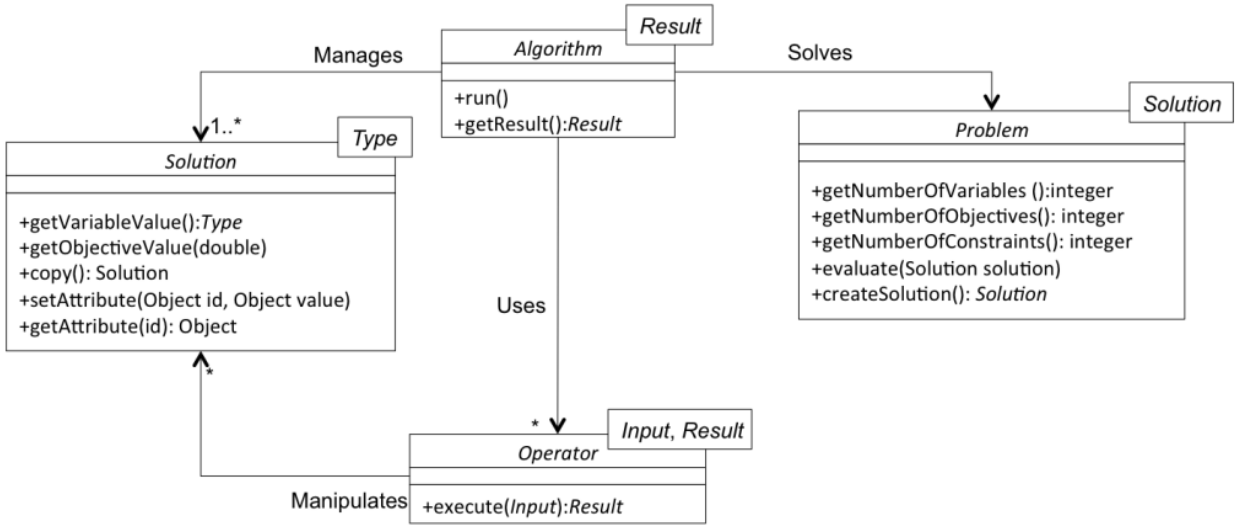


Figura 3. Diagrama de clases UML de jMetal clases del core (Lewis et al., 2017)

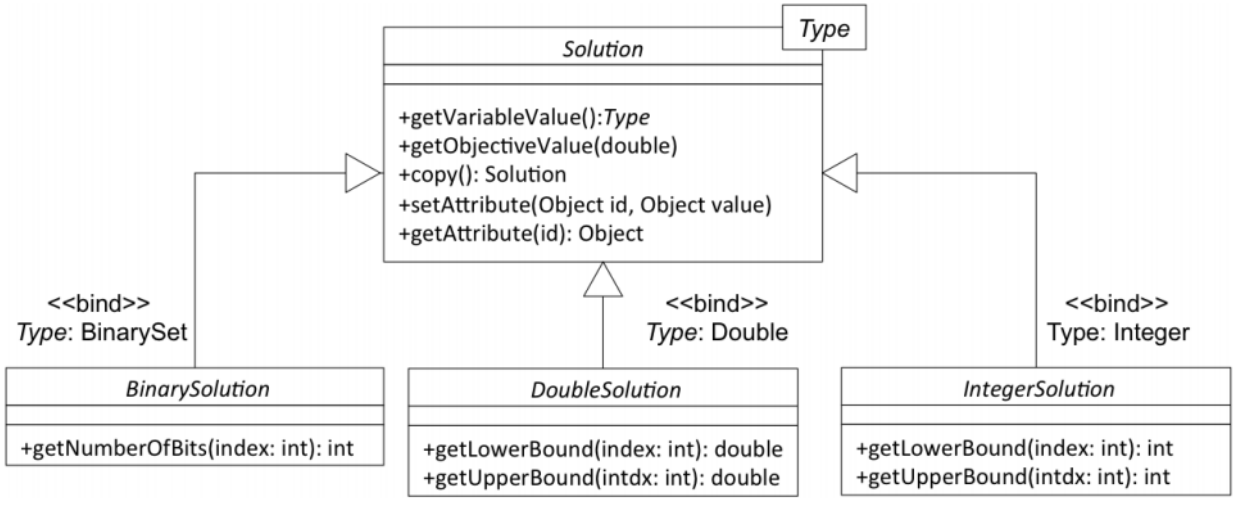


Figura 4. Diagrama de la clase Solution (Lewis et al., 2017)

jMetal es un proyecto de Maven lo cual lo hace más fácil de incorporar en cualquier proyecto de Maven (https://maven.apache.org/) y también de utilizar dependencias de terceros. Hasta el momento está dividido en 6 paquetes:

- jmetal-algorithm: Implementaciones de las metaheurísticas incluidas en el framework.
- jmetal-auto: Auto configuración
- jmetal-core: Clases de la arquitectura central más algunas utilidades, incluidos los indicadores de calidad.
- jmetal-example: Contiene un conjunto de ejemplos propuestos por el framework.
- jmetal-lab: Paquete que contiene un conjunto de clases para realizar estudios experimentales relacionados con la evaluación y comparación del desempeño de algoritmos multiobjetivo en un conjunto de problemas.
- jmetal-problem: Implementación de los problemas conocidos.

3.1. Codificaciones de soluciones

Al utilizar metaheurísticas una de las primeras decisiones que hay que tomar es definir cómo codificar o representar las soluciones tentativas del problema a resolver. La representación depende en gran medida del problema y determina las operaciones (p. Ej., Recombinación con otras soluciones, procedimientos de búsqueda local, etc.) que se pueden aplicar. Así, seleccionar una representación concreta tiene un gran impacto en el comportamiento de las metaheurísticas y, por tanto, en la calidad de los resultados obtenidos. La clase base de todas las codificaciones de la solución en jMetal es la interfaz de la solución, donde cualquier solución contiene una lista de variables de decisión, un vector de valores objetivos y un vector de valores de restricción (es decir, el grado de violación de restricción por cada una de las restricciones del lado del problema) y un diccionario de atributos). Los valores de las variables se asignan cuando se crea una solución (normalmente, cuando se invoca el método createSolution() de un problema), mientras que los valores de objetivo y restricción se establecen al evaluar una solución. jMetal proporciona actualmente las siguientes interfaces que representan codificaciones (todas ellas extendiendo la Solución): BinarySolution, IntegerSolution, DoubleSolution, PermutationSolution, CharSequenceSolution (define una implementación de solución que representa secuencias de caracteres) y CompositeSolution (una solución se compone de una lista de soluciones, lo que permite mezclar diferentes tipos de codificaciones en una única solución).

3.2. Problemas

Todos los problemas en jMetal implementan la interfaz Problem. Cada problema se caracteriza por el número de variables de decisión, el número de funciones objetivo y el número de restricciones, por lo que se deben definir métodos de obtención para devolver esos valores. El tipo genérico S permite determinar la codificación de las soluciones del problema. De esta manera, un problema debe incluir un método para evaluar cualquier solución de clase S, así como proporcionar un método createSolution() para crear una nueva solución. Se proporciona una clase llamada AbstractGenericProblem que contiene implementaciones de métodos setter y getter para los campos del problema. Si un problema tiene restricciones, se asume que el grado de restricción general de una solución dada se calcula dentro del método evaluate(). jMetal proporciona actualmente un conjunto de problemas conocido tanto multiobjetivos como monoobjetivos que pueden servir de guía abordar cualquier problema de optimización.

3.3. Manejo de restricciones

El mecanismo de manejo de restricciones adoptado en jMetal se basa en el utilizado en NSGA-II (descrito en (Deb et al., 2002), e involucra soluciones, problemas y comparadores. Cualquier solución puede tener un conjunto de restricciones, que se devuelven en un vector de valores dobles. Las restricciones generalmente se asignan cuando una solución es evaluada por un problema restringido; si el problema no tiene restricciones, se devolverá una matriz vacía. Se puede crear un método para evaluar las restricciones llamado evaluateConstraints() que se puede llamar luego de evaluar la solución o simplemente se puede realizar el cálculo de las restricciones dentro del método evaluate(). Existe un requisito para trabajar con restricciones el cual indica que cada restricción debe expresarse como una desigualdad de expresión de tipo >= 0.0. De esta manera, cuando la expresión < 0.0, se considera una violación de restricción. El grado de violación de restricción general de una solución se puede calcular mediante el método NumberOfViolatedConstraints() que pertenece a la clase estática ConstraintHandling.

3.4. Estudio Experimental

Cualquier algoritmo multiobjetivo en jMetal tiene como objetivo encontrar una aproximación al frente Pareto de un problema, que se devuelve al final de la búsqueda como una lista de soluciones. Dado un algoritmo que resuelve un problema continuo, el resultado de la optimización se puede obtener con el método getResult (). Los resultados pueden ser almacenados en archivos para su posterior utilización. Generalmente se utilizan los archivos “VAR.csv” y “FUN.csv” que contendrán los valores de las soluciones (variables) y las funciones objetivo, respectivamente, utilizando una coma como separador. La razón para separar los resultados de salida en dos archivos es que "FUN.csv" se puede trazar directamente con GnuPlot, R, etc. Un estudio experimental se utiliza para comparar una serie de algoritmos al resolver un conjunto de problemas. Implica la ejecución y posterior análisis de los resultados de R ejecuciones independientes de A algoritmos, cada uno de los cuales resolverá P problemas. Este proceso se lleva a cabo en tres pasos: ejecución de todas las configuraciones (es decir, todas las combinaciones de A, P y R), aplicar indicadores de calidad a los frentes obtenidos y evaluación del desempeño mediante el uso de pruebas estadísticas y la generación de material que resuma los resultados (por ejemplo, tablas de látex y gráficas). Existe una clase llamada Experiment (que tiene asociada una clase ExperimentBuilder), que se puede poblar con varios componentes. Los pasos mencionados son realizados por ExecuteAlgorithms (paso 1), ComputeQualityIndicators (paso 2), y el resto de componentes se pueden seleccionar para producir una variedad de elementos para analizar los resultados, como tablas, figuras (boxplots) y páginas HTML de Latex. Para calcular los indicadores de calidad es necesario tener un frente Pareto por problema; cuando se resuelven problemas de benchmarks estos frentes generalmente se conocen y pueden ser descargados dentro del propio proyecto de jMetal y posteriormente utilizados, pero este no es el caso cuando se trata de problemas reales. Para hacer frente a este problema jMetal incluye una clase llamada GenerateReferenceParetoFront que produce un frente Pareto de referencia a partir de todos los resultados arrojados por todas las ejecuciones de todos los algoritmos, además la clase GenerateReferenceParetoSetAndFrontFromDoubleSolutions, hace lo mismo si los problemas a resolver son continuos; en este caso, también se genera un frente Pareto de referencia, así como archivos con las soluciones aportadas por cada algoritmo a este conjunto.

3.5. Submódulo de Paralelismo

3.5.1. Paralelismo síncrono

Tradicionalmente, jMetal ha proporcionado soporte básico para evaluar la lista de soluciones (es decir, poblaciones y enjambres) en paralelo mediante el uso de núcleos de procesador. Se recibe la lista de soluciones a ser evaluadas por un problema y devuelve la lista de soluciones evaluadas. Se incluyen dos evaluadores, uno que evalúa secuencialmente y otro que evalúa las soluciones en paralelo utilizando varios núcleos del procesador. El modelo paralelo cuando se usan múltiples hilos es síncrono, lo que implica que el comportamiento del algoritmo en paralelo es el mismo que el secuencial, pero el rendimiento se ve afectado por el hecho de que el algoritmo alterna códigos paralelos (la evaluación de soluciones) y secuenciales (el resto del código del algoritmo), por lo que este modelo no se escala bien. En el submódulo jmetal-parallel se incluyen dependencias de los paquetes Spark ya que se utiliza un evaluador basado en Apache Spark (Barba-Gonzaléz et al., 2017).

3.5.2. Paralelismo asíncrono

Como se mencionó anteriormente, un inconveniente del paralelismo síncrono es que puede encontrar dificultades para escalar cuando el número de núcleos de procesadores es alto. Una alternativa es un modelo paralelo asincrónico, que básicamente consiste en que, cada vez que se tiene que evaluar una nueva solución, se envía inmediatamente para su evaluación de forma asincrónica, sin tener que esperar. Por tanto, este modelo puede escalar mejor que el síncrono, pero hay que modificar la metaheurística para adoptarlo, y el algoritmo resultante no se comporta exactamente como el secuencial. El submódulo jmetal-parallel contiene una interfaz de paralelismo asíncrono y una implementación multiproceso de la misma basada en el esquema master-worker, que se aplica en dos clases: AsynchronousMultiThreadedGeneticAlgorithm y AsynchronousMultiThreadedNSGAII.

3.6. Autoconfiguración de algoritmos evolutivos

Una tendencia actual en la optimización multiobjetivo es utilizar herramientas de configuración automática de parámetros para encontrar ajustes precisos de metaheurísticas para resolver eficazmente una serie de problemas. La idea es evitar el enfoque tradicional de realizar una serie de pruebas piloto, que normalmente se realizan sin seguir una estrategia sistemática. En este contexto, una configuración de algoritmo es una asignación completa de valores a todos los parámetros requeridos del algoritmo.

Una herramienta de autoconfiguración que se puede utilizar con el framework Irace (Nebro et al., 2019), un paquete de R que implementa un algoritmo elitista, donde las configuraciones del algoritmo se seleccionan al azar al principio, pero sesgadas hacia las mejores configuraciones encontradas en iteraciones posteriores. En cada iteración, las configuraciones generadas y las mejores de iteraciones anteriores se evalúan en instancias de problemas de entrenamiento. Se utiliza una prueba estadística para decidir qué configuraciones deben eliminarse. Cuando se termina una iteración, las configuraciones con mejores resultados son las que prevalecen.

3.7. Indicadores de rendimiento

En la optimización multiobjetivo no es aplicable observar el mejor valor obtenido por un algoritmo, sino que debemos calcular un conjunto aproximado al frente Pareto del problema, normalmente se requiere de dos propiedades, convergencia y diversidad uniforme. Anteriormente se mencionaron varios que podían utilizarse para medir esas dos propiedades tales como Hipervolumen (Zitzler & Thiele, 1999),

Distancia generacional (Veldhuizen & Lamont, 1998), Distancia generacional invertida (Veldhuizen & Lamont, 1998), Epsilon (J. D. ; Knowles et al., 2006), Spread (Deb et al., 2002) y Spread Generalizado (Zhou et al., 2006). Algunos de ellos pretenden medir solo la convergencia o la diversidad, mientras que otros contemplan ambos criterios. En la siguiente figura se muestra una clasificación de los indicadores en función del aspecto que miden

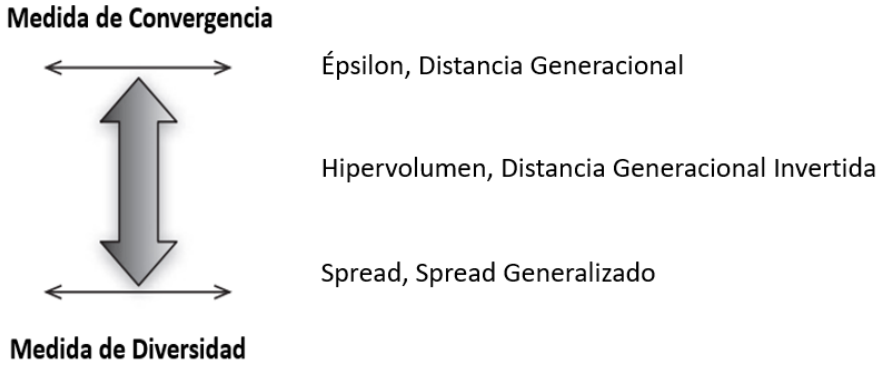


Figura 5. Clasificación de los indicadores de calidad (Durillo & Nebro, 2011)

3.7.1. Distancia Generacional

Fue introducido para medir la distancia entre los elementos de la aproximación calculada y los del frente de Pareto óptimo y se define como:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

donde *n* es el número de soluciones en la aproximación y d_i es la distancia euclidia (medida en el espacio objetivo) entre cada una de estas soluciones y el miembro más cercano en el frente de Pareto óptimo. Un valor de $GD = 0$ indica que todos los elementos generados están en el frente de Pareto (Veldhuizen & Lamont, 1998).

3.7.2. Distancia Generacional Invertida

Es una variante de la distancia generacional. Mide las distancias entre cada solución que compone el frente de Pareto óptimo y la aproximación calculada. Se puede definir como sigue:

$$IDG = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

siendo *n* el número de soluciones en el frente óptimo de Pareto y di es la distancia euclidia (medida en el espacio objetivo) entre cada punto de ese frente y la solución más cercana (Veldhuizen & Lamont, 1998).

3.7.3. Hipervolumen

Este indicador calcula el volumen, en el espacio objetivo cubierto por los miembros de un conjunto no dominado de soluciones Q, por ejemplo, la región encerrada en la línea discontinua de la figura 6, $Q = \{A, B, C\}$, para problemas en los que todos los objetivos deben ser minimizados. Matemáticamente, para

cada solución $i \in Q$, se construye un hipercubo v_i con un punto de referencia *W* y la solución *i* como las esquinas diagonales del hipercubo. El punto de referencia puede encontrarse simplemente construyendo un vector de los peores valores de la función objetivo. A continuación, se encuentra la unión de todos los hipercubos y se calcula su hipervolumen (HV) (Zitzler & Thiele, 1999):

$$HV = volumen\left(\bigcup_{i=1}^{|Q|} v_i\right)$$

Los algoritmos con valores más grandes de HV son deseables.

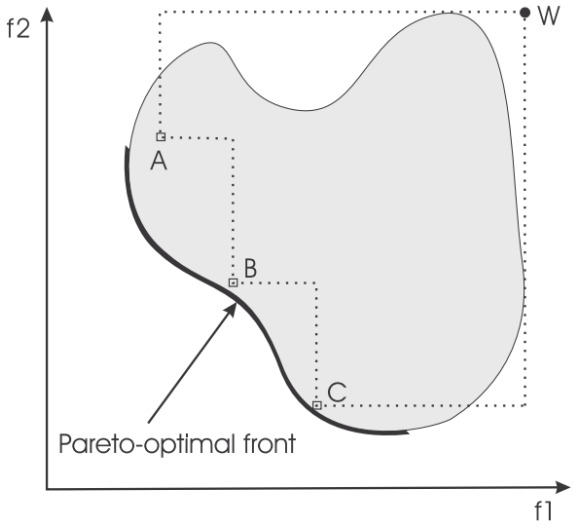


Figura 6. Hipervolumen delimitado por las soluciones no-dominadas A, B y C (Durillo & Nebro, 2011).

3.7.4. Epsilon

Dado un frente calculado para un problema A, este indicador es una medida de la menor distancia que se necesitaría para trasladar cada solución en A de manera que domine el frente de Pareto óptimo de este problema. Formalmente dado $z^{\vec{1}} = (z_1^1, \dots, z_n^1)$ y $z^{\vec{2}} = (z_1^2, \dots, z_n^2)$ donde *n* es el número de objetivos (J. D. ; Knowles et al., 2006).

$$I_{\epsilon+}^1(A) = inf_{\epsilon \in \mathbb{R}}\left\{\forall z^{\vec{2}} \in \mathcal{PF}^* \exists z^{\vec{1}} \in A : z^{\vec{1}} \prec_\epsilon z^{\vec{2}}\right\}$$

donde $z^{\vec{1}} \prec_\epsilon z^{\vec{2}}$ sí y solo si $\forall 1 \leq i \leq n: z_i^1 < \varepsilon + z_i^2$

3.7.5. Spread

Mide el grado de dispersión del conjunto de soluciones calculadas. Se define como:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1}|d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}}$$

donde d_i es la distancia euclídea entre soluciones consecutivas, $\bar{d}$ es la media de estas distancias, y d_f y d_l son las distancias euclídias del frente Pareto óptimo en el espacio objetivo. Este indicador toma un

valor cero para una distribución ideal, señalando una perfecta dispersión de las soluciones en el frente de Pareto (Deb et al., 2002).

3.7.6. Spread Generalizado

Es una extensión de la métrica Spread, donde se calcula la distancia de un punto dado a su vecino más cercano. Está basado en la métrica propuesta en (Zhou et al., 2006). Dado que estos indicadores no están libres de un escalado arbitrario de los objetivos, jMetal los aplica siempre después de normalizar los valores de la función objetivo.

4. Algoritmos evolutivos

4.1. Algoritmos

El framework jMetal ofrece un conjunto de algoritmo que se pueden utilizar para la resolución de diferentes problemas, en la figura 7 se muestran los utilizados para la resolución de los problemas planteados en esta investigación mediante permutaciones. Cada algoritmo cuenta con un conjunto de operadores.

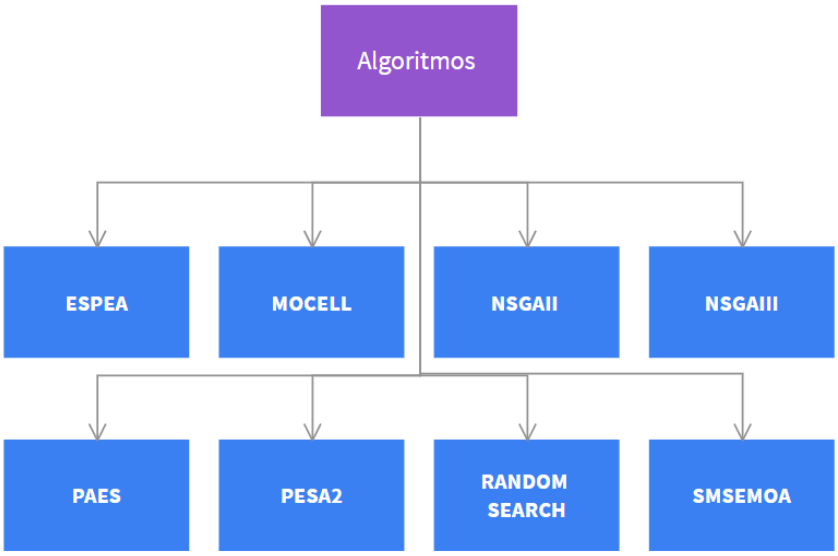


Figura 7. Algoritmos utilizados para la resolución de los problemas propuestos en la investigación mediante permutaciones.

4.1.1. ESPEA

Es un algoritmo evolutivo inspirado en la energía potencial electrostática, que utiliza una noción matemática para definir una distribución óptima de puntos según un criterio de diversidad global. Esta noción se puede combinar para impulsar la distribución hacia subconjuntos preferidos del frente de Pareto (Braun et al., 2015). Para aproximar el frente de Pareto se basa en el fenómeno físico de la energía potencial electrostática. En física, las partículas cargadas interactúan mediante fuerzas de Coulomb entre sí. En un sistema cerrado, estas fuerzas de Coulomb inducen una energía potencial electrostática. La energía que exhibe una partícula dada con respecto a otra partícula es igual al producto de ambas cargas multiplicado por la constante de Coulomb y dividido por la distancia euclidiana entre sí. La suma de todas

las energías por pares constituye la energía del sistema. Dado un conjunto finito de cargas, existe una distribución de partículas que minimiza la energía del sistema cerrado (Walker, 2010).

Parámetros:

- Problema
- Operador de mutación
- Probabilidad de mutación
- Operador de cruce
- Probabilidad de cruce
- Tamaño de la población
- Número de evaluaciones
- Operador de selección
  - Comparador de dominancia
- Método de Escalarización
- Normalización de objetivos: tipo de dato boolean en dependencia de si se quiere normalizar los objetivos o no

4.1.2. MOCELL

Algoritmo genético celular simple y elitista para resolver problemas de optimización multiobjetivo. Se caracteriza por utilizar un archivo externo para almacenar soluciones no dominadas y un mecanismo de retroalimentación en el que las soluciones de este archivo reemplazan aleatoriamente a los individuos existentes en la población después de cada iteración (Nebro et al., 2009).

Parámetros:

- Problema
- Operador de mutación
- Operador de cruce
- Tamaño de la población
- Número de evaluaciones
- Operador de selección
  - Comparador de dominancia

4.1.3. NSGA-II

Algoritmo evolutivo multiobjetivo basado en clasificación no dominada que soluciona las tres dificultades principales de los algoritmos evolutivos como lo son la complejidad computacional, un enfoque no basado en elitismo y la necesidad de especificar un parámetro compartido. Además, cuenta con un operador de selección que en la fase de reemplazamiento crea una población resultado de combinar las poblaciones de progenitores y descendientes y seleccionando las mejores soluciones, siendo por tanto elitista (Deb et al., 2002).

Parámetros:

- Problema
- Operador de mutación

- Operador de cruce
- Tamaño de la población
- Número de evaluaciones
- Operador de selección
  - Comparador de dominancia

4.1.4. NSGA-III

Algoritmo evolutivo de muchos objetivos (many-objective optimization) basado en puntos de referencia siguiendo el marco NSGA-II que enfatiza en los miembros de la población que no están dominados, pero cerca de un conjunto de puntos de referencia proporcionados. En problemas de dimensiones superiores, los algoritmos evolutivos multiobjetivo se enfrentan a una tarea cada vez más difícil de mantener la diversidad, así como de converger al frente de Pareto óptimo. El suministro de un conjunto de puntos de referencia y la metodología de niching eficiente de NSGA-III para encontrar una solución de Pareto óptimo asociada con cada punto de referencia hace posible la preservación de la diversidad de las soluciones obtenidas en un máximo de 15 objetivos (Deb & Jain, 2014).

Parámetros:

- Problema
- Operador de mutación
- Operador de cruce
- Tamaño de la población
- Número de evaluaciones
- Operador de selección
  - Comparador de dominancia
- Número de divisiones

4.1.5. PAES

Se basa en un esquema de evolución más simple para problemas multiobjetivo, llamado Estrategia de Evolución Archivada de Pareto (PAES). Se basa en dos ideas fundamentales, una es que debe limitarse estrictamente a la búsqueda local, o sea, utilizar un solo operador de variación de la población (mutación) y pasar de una solución actual a un vecino cercano y la otra idea es que debe ser un verdadero optimizador de pareto, tratando todas las soluciones no dominadas como si tuvieran el mismo valor. Lograr ambas ideas es algo complejo ya que al comparar dos soluciones ninguna solución dominará a la otra. PAES ofrece una solución manteniendo un archivo de soluciones no dominadas encontradas anteriormente, el cual luego es utilizado para como medio para estimar la clasificación de dominio real de un par de soluciones. Dicho esto, podemos ver a PAES como un algoritmo compuesto por tres partes: un generador de soluciones candidatas, una función de aceptación de soluciones candidatas y el archivo de soluciones no dominadas. Se puede decir que representa el enfoque no trivial más simple para un procedimiento de búsqueda local multiobjetivo, donde el generador de soluciones candidatas es similar a una simple escalada por mutación aleatoria, que mantiene una única solución actual y en cada iteración produce un único candidato nuevo a través de una mutación aleatoria. Dado que el objetivo de la búsqueda multiobjetivo es encontrar una variedad de soluciones no dominadas, PAES necesariamente debe

proporcionar una lista de soluciones no dominadas para mantener explícitamente un número limitado de ellas. (J. Knowles & Corne, 1999).

Parámetros:

- Problema
- Operador de mutación
- Tamaño del archivo
- Número de evaluaciones

4.1.6. PESA2

Es una versión del algoritmo PESA (D. W. Corne et al., 2000), que aplica una nueva técnica de selección para algoritmos evolutivos de optimización multiobjetivo en la que la unidad de selección es una hyperbox en el espacio objetivo. En esta técnica, en lugar de asignar una aptitud selectiva a un individuo, la aptitud selectiva se asigna a las hyperbox en el espacio objetivo que están ocupadas por al menos un individuo en la aproximación actual al frente de Pareto. De este modo, se selecciona una hyperbox y el individuo seleccionado resultante se elige al azar de esta. Es un método de selección más sensible para asegurar una buena distribución del desarrollo a lo largo del frente de Pareto que la selección basada en individuos (D. Corne et al., 2001).

Parámetros:

- Problema
- Operador de mutación
- Probabilidad de mutación
- Operador de cruce
- Probabilidad de cruce
- Tamaño de la población
- Número de evaluaciones
- Tamaño del archivo

4.1.7. RANDOM SEARCH

Algoritmo que implementa una búsqueda aleatoria simple.

Parámetros:

- Problema
- Número de evaluaciones

4.1.8. SMS-EMOA

La medida de hipervolumen (o métrica S) es una medida de calidad que se aplica con frecuencia para comparar los resultados de los algoritmos evolutivos de optimización multiobjetivo (EMOA). Este algoritmo se centra en la maximización del hipervolumen dominado dentro del proceso de optimización. Presenta un operador de selección basado en la medida de hipervolumen combinado con el concepto de clasificación no dominada. La población del algoritmo evoluciona a un conjunto de soluciones bien distribuidas, centrándose así en regiones interesantes del frente de Pareto. (Beume et al., 2007).

Parámetros:

- Problema
- Operador de mutación
- Probabilidad de mutación
- Operador de cruce
- Probabilidad de cruce
- Tamaño de la población
- Número de evaluaciones
- Operador de selección
  - Comparador de Dominancia

4.2. Operadores

4.2.1. Población inicial

- Random
  - Parámetros: el problema a resolver y el tamaño de la población.
- A partir de un fichero: si la cantidad de elementos en el fichero es menor que el tamaño de la población el resto de la población se completará de manera aleatoria. Por el contrario, si es mayor se lanzará una excepción.
  - Parámetros: el problema a resolver y el tamaño de la población.

4.2.2. Operadores de selección

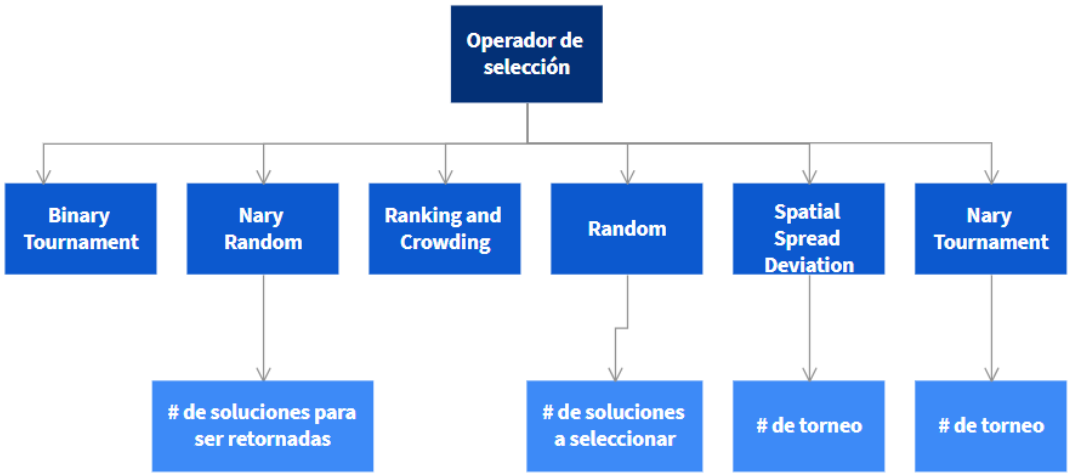


Figura 8. Operadores de selección.

- Binary Tournament Selection: aplica una selección de torneo binario (2:1) para devolver la mejor solución entre dos que se han elegido al azar de una lista de soluciones.
  - Parámetros: comparador de dominancia.
- Nary Random Selection: operador de selección aleatorio utilizado para seleccionar un número ***N*** de soluciones.
  - Parámetros: número de soluciones a ser retornadas.

- Random Selection: operador de selección aleatoria utilizado para seleccionar una solución de una lista.
- Ranking And Crowding Selection: las soluciones se toman de acuerdo a la media de los valores de ranking y su crowding distance.
    - Parámetros: comparador de dominancia y número de soluciones a ser retornadas.
- Spatial Spread Deviation Selection: utiliza un estimador de densidad SSD cuya complejidad es $O\ (n^2 \log n)$ que es ligeramente superior a la de crowding distance (CD). Sin embargo, es altamente competitiva con CD en problemas bidimensionales, y más adecuada para problemas tridimensionales.
    - Parámetros: comparador de dominancia y número de torneo.
- Nary Tournament Selection: aplica una selección de torneo para devolver la mejor solución entre ***N*** que se han elegido al azar de una lista de soluciones (N:M).
    - Parámetros: número de soluciones a ser retornadas ***M*** y comparador de dominancia.

4.2.3. Operadores de cruce

Para problemas que se resuelven mediante permutaciones el número de operadores de cruce es limitado, por lo que se utiliza el único operador disponible.

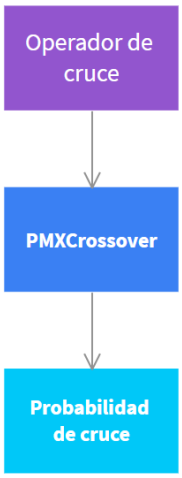


Figura 9. Operador de cruce.

- PMX Crossover
    - Parámetros: probabilidad de cruce.

Técnica de cruce crossover parcialmente mapeado. Comienza eligiendo dos puntos de corte aleatorios a lo largo de las cadenas principales. En el primer paso, los segmentos entre los puntos de corte se copian del padre 1 a la descendencia 2 y del padre 2 a la descendencia 1. En el siguiente paso, todos los elementos antes del primer punto de corte y después del segundo se copian del padre 1 y del padre 2 a la descendencia 1 y la descendencia 2, respectivamente (Attarmoghaddam et al., 2019).

4.2.4. Operadores de mutación

Tal y como pasa en el operador de mutación para problemas que se resuelven mediante permutaciones solamente hay un operador de mutación disponible.

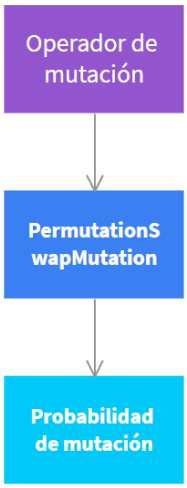


Figura 10. Operador de mutación.

- Permutation Swap Mutation
  - Parámetros: probabilidad de mutación, en caso de no establecer ningún valor será 1 dividido entre el número de variables.

4.2.5. Comparadores de dominancia

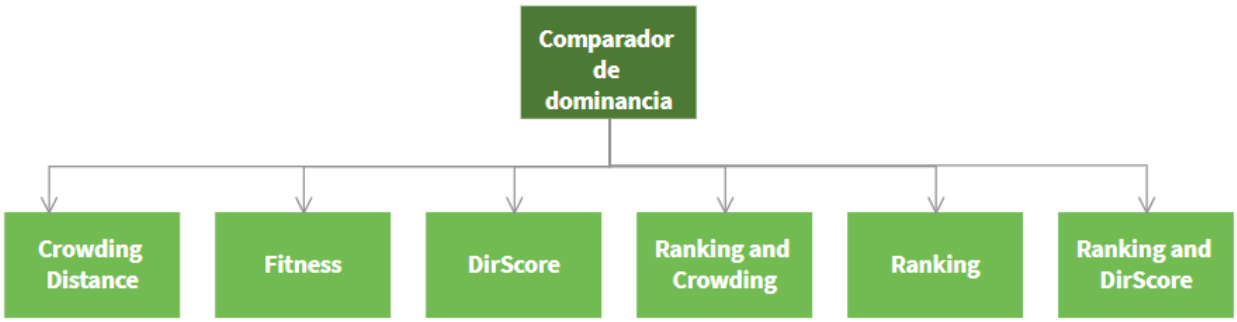


Figura 11. Operadores de comparación de dominancia.

- Crowding Distance Comparator: compara dos soluciones de acuerdo a la crowding distance, cuanto mayor sea la distancia mejor (Deb et al., 2002).
- Dir Score Comparator: comparador de diversidad basado en vectores de referencia (DIR) para estimar la diversidad de aproximaciones del frente Pareto (FP) para la optimización de muchos objetivos. En DIR, se genera un conjunto de vectores de referencia uniformes y generalizados. La cobertura de cada solución en el espacio objetivo se evalúa por el número de vectores de referencia representativos con los que está asociada. La diversidad (tanto la dispersión como la uniformidad) está determinada por la desviación estándar de la cobertura para todas las soluciones. Cuanto menor sea el valor de DIR, mejor será la diversidad de una aproximación de FP (Cai et al., 2018).
- Fitness Comparator: comparador mediante la función de evaluación.
- Ranking And Crowding Distance Comparator: comparador basado en el ranking de las soluciones, si este es el mismo se utiliza la crowding distance. Es el comparador utilizado en el algoritmo NSGA-II (Deb et al., 2002).

- Ranking Comparator: compara de acuerdo con el rank de las soluciones.
- Ranking And Dir Score Distance Comparator: comparador que combina el ranking y el comparador Dir Score.

4.2.6. Reemplazamiento

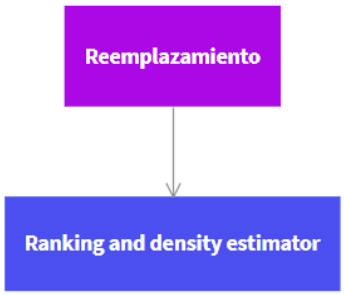


Figura 12. Operadores de reemplazamiento.

- Ranking And Density Estimator Replacement
  - Parámetros: multi comparador (ranking y el estimador de densidad).

4.2.7. Evaluaciones

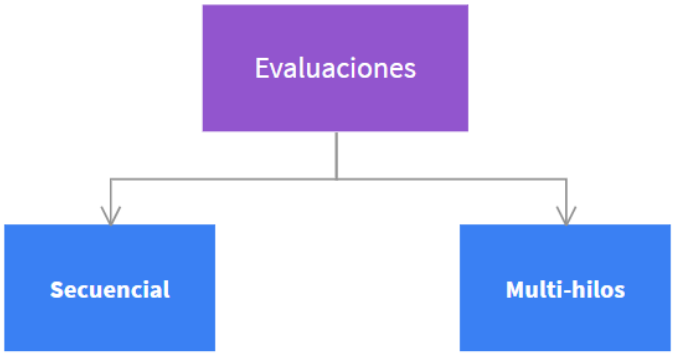


Figura 13. Tipos de evaluaciones

- Sequential
  - Parámetros: el problema a resolver.
- Multithreaded
  - Parámetros: el problema a resolver y el número de hilos.

4.2.8. Criterios de parada

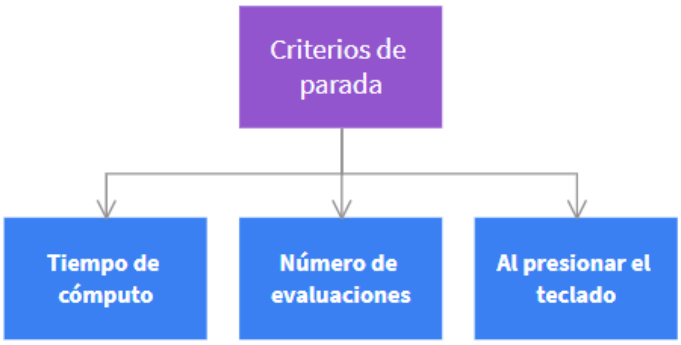


Figura 14. Criterios de parada.

- Termination By Computing Time: la condición de para es cuando el tiempo de cálculo de un algoritmo supera un umbral determinado.
  - Parámetros: tiempo computacional en milisegundos.
- Termination By Evaluations: la condición de parada es en base a un número máximo de evaluaciones indicadas.
  - Parámetros: número de iteraciones.
- Termination By Keyboard: la condición de parada es en base a la introducción de un carácter por

4.2.9. Escalarización

Tradicionalmente se utiliza un método de escalarización para resolver problemas de optimización multicriterio.

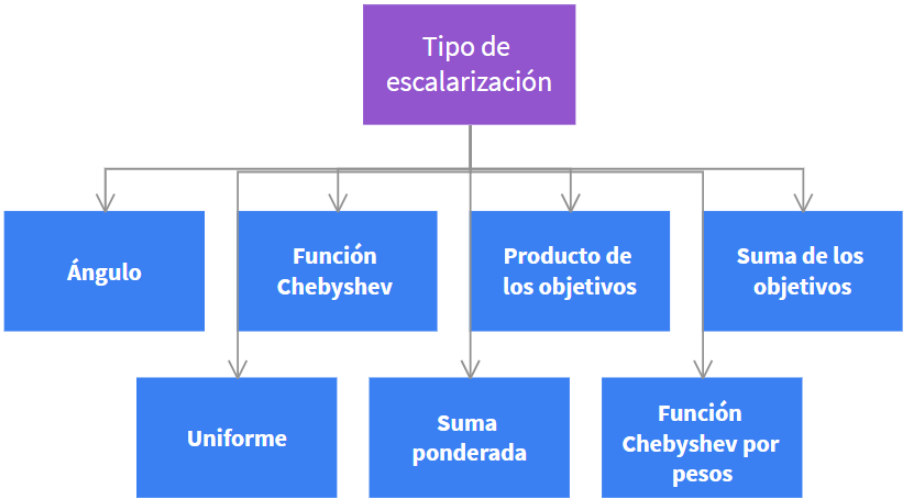


Figura 15. Tipos de escolarización.

- ANGLE UTILITY: Los valores de escalarización se basan en ángulos máximos a puntos extremos (Braun et al., 2017).
- CHEBYSHEV: utiliza la función de escalarización Chebyshev (Drugan & Nowe, 2014).

- PRODUCT_OF_OBJECTIVES: multiplica todos los objetivos.
- SUM_OF_OBJECTIVES: suma todos los objetivos.
- UNIFORM: A todas las soluciones se les asigna un valor de escalarización de 1.
- WEIGHTED_SUM: se utiliza la suma ponderada.
- WEIGHTED_CHEBYSHEV: utiliza la función de Chebyshev con pesos.

5. Problema de planificación de células de soldadura robotizada.

5.1. Definición del problema

Este problema consiste en la planificación de un conjunto de ordenes (o pedidos de soldadura de piezas) sobre un conjunto de celdas de soldadura (o robots de soldadura) de acuerdo con un conjunto de restricciones impuestas en cada orden:

- Número de unidades a ser fabricadas (soldadas).
- Su gama, o velocidad de fabricación estimada en unidades/hora.
- Número de turnos de trabajo de 8 horas diarios para su fabricación.
- De prioridad entre órdenes, de forma que las de mayor prioridad deben ser procesadas antes que las de prioridad menor.
- De compatibilidad entre órdenes y robots, en cuanto a que las piezas por sus dimensiones solo pueden ubicarse en las mesas de determinados robots.
- Temporales de:
  - Disponibilidad del robot, que debe de estar libre para poder procesar cualquier orden.
  - Disponibilidad de Materia prima que asegure un flujo continuo de piezas a ser soldadas.
  - Disponibilidad e utillajes (o anclajes de las piezas a soldar) que necesariamente han de estar instalados en la mesa del robot antes de poder ubicar y fijar las piezas a soldar.
  - Preparación del robot (lo que incluye la instalación de los utillajes en su mesa y la programación eficaz del brazo soldador.
  - Fecha máxima de entrega de todas las unidades pactada con el cliente.

Para resolver esta producción disponemos de un conjunto de robots de soldadura que ofrecen:

- Uno o más brazos de soldadura, la gama de fabricación de una pieza se multiplica directamente por el número de brazos del robot.
- Una fecha más temprana de disponibilidad, salvo avería o mantenimiento, esta vendrá condicionada por el tiempo necesario para finalizar la orden que se esté produciendo actualmente en el robot.

Como resultado debemos asignar a cada orden un robot y su posición en su cola de órdenes a ejecutar. Su fecha de inicio y finalización se fijará siguiendo todas las restricciones temporales anteriormente indicadas. En cada proceso de (re)planificación los expertos pueden cambiar o actualizar cualquiera de los parámetros de las ordenes existentes en el sistema, tanto las aún no iniciadas como las que están en producción (ej: se cambia el número de turnos de una orden para acelerar su producción, se actualiza la gama de una pieza a la vista de la producción del día anterior, se elimina un robot entre los compatibles con las órdenes debido a que está averiado, etc.).

El conjunto de ordenes (no iniciadas) se replanifica con cada nueva jornada, y también con la llegada de un pedido urgente (con prioridad alta) que deba ser planificado a la mayor brevedad. Esta planificación no debe ser solo correcta, sino que además es deseable que se cumplan los siguientes objetivos:

- Minimizar la fecha de finalización de todas las ordenes con respecto a su fecha de entrega pactada con el cliente.
- Minimizar el número de robots a emplear para la fabricación de todas las órdenes.

Ambos objetivos responden a un fin común, disponer de la mayor disponibilidad de recursos para responder ante incidencias sobre la producción: retrasos debidos a fallos de suministro de materia prima, llegada de otros pedidos urgentes, producción más lenta de lo previsto inicialmente, averías de robots, etc.

En los siguientes apartados se describirán en detalle cada una de estas entidades, los objetivos, restricciones y su modelado y resolución con un algoritmo evolutivo multiobjetivo.

5.2. Entidades relacionadas con el problema

5.2.1. Datos de entrada

Órdenes: Por orden entendemos cada uno de los pedidos que llegan a la Empresa. Cada orden lleva asociada una pieza o conjunto de piezas que deben ser soldadas en un número concreto de unidades. A continuación, se describen los distintos atributos que definen una orden y las restricciones materiales y temporales para su producción.

- Id: identificador único de la orden. Es la etiqueta con la que el equipo humano se refiere a este pedido.
- Fecha entrada: fecha en que la orden llega a la empresa. A partir de esa fecha sería posible producir las piezas de la orden, siempre y cuando se disponga de materia prima, de los utillajes necesarios y estos se hayan instalado y el robot programado correctamente para ello.
- Unidades (uds): número entero total de unidades a fabricar.
- Fecha del material: fecha en que la materia prima está lista para empezar a producir unidades de la orden. La Empresa dispone de los almacenes, transporte y operarios necesarios para suministrar al robot de soldadura la materia prima de forma continua desde una fecha concreta.
- Fecha de utillaje: fecha en que el utillaje está listo para ser instalado y empezar a producir las unidades de la orden. Para ello el equipo de ingeniería de la Empresa debe diseñar y fabricar el/los utillajes necesarios para el correcto emplazamiento de las piezas en la mesa de soldadura del robot.
- Prioridad: prioridad de la orden entre 1 y 5, a mayor valor mayor es la prioridad. La prioridad se puede deber a diversas cuestiones de la Empresa y sus clientes, pero se resume en una prelación de unos pedidos frente a otros y que se resumen en estos cinco valores.
- Tiempo de preparación: (también denominado setup) horas totales necesarias para la preparación de la orden. Esto incluye la instalación del utillaje, la programación del robot y el transitorio hasta alcanzar la velocidad de producción marcada por la gama. La gran parte de los robots son ciegos y se limitan a reproducir la secuencia de posicionamientos de su brazo equipado con una pistola de soldadura en su extremo, con lo que la precisión debe ser milimétrica.

- Gama: velocidad de fabricación de unidades por hora. Esta medida puede verse actualizada con el paso del tiempo con los datos de producción de la última jornada. Como resultado esta gama o velocidad se puede incrementar si la estimación inicial fue muy conservadora y los operarios y el robot son capaces de producir más unidades en el mismo tiempo, o por el contrario la velocidad puede reducirse si empiezan a fallar las soldaduras o los operarios no son capaces de suministrar con la suficiente celeridad las piezas al robot.
- Turnos: número de turnos empleados en la fabricación de las órdenes. Por lo general las órdenes se resolverán en uno o dos turnos de 8 horas, excepcionalmente algunos pedidos se pueden planificar a 3 turnos, siempre y cuando haya personal y esté suficientemente justificada su urgencia ya que supone un coste elevado debido a las horas extra de los trabajadores. El número de turnos será otro parámetro que podría cambiar de una jornada a otra.
- Fecha de entrega: fecha máxima de entrega del total de unidades a producir. Esta fecha estará preestablecida por el departamento de ventas de la Empresa en concordancia con el cliente. El instante de finalización de la soldadura de la última unidad no debe sobrepasar esta fecha.
- Robots compatibles: lista de robots compatibles con esta orden. Las dimensiones de las piezas y sus utillajes restringen los robots que pueden ser utilizados, ya que imponen unas restricciones en cuanto a las dimensiones que deben disponer las mesas sobre las que actúa el robot.
- Órdenes compatibles: lista de órdenes ya planificadas en un robot que son compatibles (en cuanto a compartir el robot donde está instalado) con esta nueva orden a planificar. Esto permitiría instalar más de una orden en un robot y que ambas órdenes se produjeran de forma simultánea, aunque esto afectaría negativamente a la gama de ambas piezas. Aunque este dato no se utilizará en la versión actual, la Empresa quiere que esté ya representada para futuras versiones.
- Robot preasignado: las órdenes cuya producción ya se ha iniciado en un robot no son expulsables, es decir el planificador debe respetar la asignación de robot actual para que finalice su producción antes de que el robot pueda comenzar el resto de las órdenes que le asigne el planificador.

Robots: En la actualidad la Empresa dispone de 11 celdas de soldadura, cada una de ellas dispone de uno o dos brazos robot equipados todos ellos con una pistola de soldadura. En la versión actual que un robot disponga de más de un brazo implica una reducción en los tiempos de fabricación, reduciendo el tiempo de soldadura (la gama) de forma directamente proporcional. En futuras versiones se estudiará también la posibilidad de que cada brazo suelde piezas de órdenes distintas.

- Id: identificador único del robot.
- Nombre: nombre del robot. Es la etiqueta con la que el equipo humano se refiere a este robot.
- Número de brazos: número de brazos de soldadura del robot. Dos brazos permiten duplicar la gama de fabricación de las piezas. En cada una de las celdas actuales a lo sumo hay dos brazos instalados.

5.2.2. Datos de salida

Órdenes planificadas: El objetivo del planificador es asignar a cada orden un robot y encolar la tarea dentro de su cola de tareas de manera que el proceso de soldadura de todas sus unidades se finalice dentro del plazo preestablecido con los clientes.

- Id del robot: identificador del robot.
- Id de la orden: identificador de la orden.

- Posición: posición ordinal dentro del conjunto de órdenes planificadas en el robot. Cada robot dispondrá de una cola secuencial de órdenes a ser producidas en dicho robot.
- Unidades producidas: número de unidades ya producidas de la orden. Esta información nos permitirá replanificar una orden a partir de la fecha actual y conociendo el número de unidades reales producidas hasta el momento.
- Turnos: número de turnos a emplear para la fabricación de la orden (1 -> mañana, 2 -> mañana y tarde, 3 -> mañana, tarde y noche). Aunque en la versión actual este dato es una entrada, se incorpora también a los datos de salida para que en futuras versiones el planificador pueda sugerir o fijar este dato.
- Fecha de inicio: fecha en que debe iniciarse la fabricación de esta orden. Esta fecha ya presupone la disponibilidad de los utillajes, la materia prima, así como el proceso de preparación finalizado: instalación de los utillajes en el robot, programación del brazo para la soldadura de la pieza y pruebas de ajuste de las soldaduras para comenzar a soldar piezas en los tiempos establecidos por la gama.
- Fecha fin: fecha final de la fabricación teniendo en cuenta el total de unidades a producir, número de unidades previamente producidas, la fecha de inicio, el tiempo de preparación, la gama y el número de turnos. Para estimar las fechas se tiene en cuenta los calendarios de festivos de la Empresa, así como los horarios de cada turno de los días laborables.

5.3. Restricciones

A raíz de las descripciones anteriores, podemos concluir que el problema cuenta con un conjunto de restricciones las cuales serán enumeradas a continuación:

- La fecha fin de cada orden no debe ser superior a la fecha de entrega.
- Una orden no puede comenzar antes de la mayor de sus fechas de material y utillaje.
- Si una orden tiene una predecesora en su robot asignado no podrá comenzar hasta que esta haya terminado.
- Para planificar las órdenes se debe tener en cuenta la prioridad. Una orden de menor prioridad no debe ser planificada antes de una de mayor prioridad.
- Se debe respetar el número de turnos establecidos para cada orden.
- Cada robot tiene una fecha de disponibilidad la cual puede ser utilizada para establecer fechas de mantenimiento, en caso de no ser establecida se tomará como fecha de disponibilidad la actual, lo cual quiere decir que el robot puede comenzar de inmediato.
- Las órdenes planificadas en un robot que tiene establecido una fecha de disponibilidad no podrán comenzar a producirse hasta esta fecha.
- Antes de planificar las órdenes se deben hacer un análisis de las órdenes que ya se están produciendo, para esto se verifica su fecha de inicio y si es igual o anterior a la fecha actual quiere decir que ya se comenzó a producir por lo cual no debe ser planificada y las órdenes que sean planificadas en este mismo robot no podrán comenzar hasta la fecha fin de esta orden que se está produciendo. En caso contrario se planificarán todas las órdenes. Es decir, las ordenes ya iniciadas no son expulsables por el planificador, esta operación solo se resolverá de manera manual.

5.4. Objetivos a optimizar

En este caso se tienen dos objetivos a optimizar. Uno de ellos es minimizar el lateness (fecha fin – fecha entrega) y el otro es minimizar el número de robots a utilizar para la producción de piezas en la fábrica.

- **Máximo Lateness** (a minimizar): Se entiende por Lateness de una orden la diferencia, en tiempo, entre la fecha en que se ha finalizado la fabricación de la última unidad del pedido y la fecha de entrega previamente acordada con el cliente. Si esta diferencia es negativa nos estamos retrasando más allá del margen de tiempo prefijado y puede consecuencias negativas para la Empresa, económicas derivadas directamente de la demora, o bien de pérdida de confianza del cliente con el riesgo de perderlo para futuros pedidos. Se tratará de minimizar el máximo Lateness del conjunto total de órdenes. Este objetivo será cero si todos los pedidos se entregan en su fecha, e idealmente será negativo si todos los pedidos finalizan su producción antes de su fecha de entrega al cliente, lo que proporcionaría a la Empresa una holgura a provechar en caso de imprevistos: aparición de pedidos urgentes, averías de robots, rotura de stocks de materia prima, etcétera.
- **Número de robots necesarios** (a minimizar): Si para cumplir las fechas de entrega de todas las órdenes podemos utilizar el menor número de robots ofreceremos a la Empresa una doble ventaja. Por un lado, una mayor robustez ante paralizaciones de robots debidas averías o mantenimientos programados, pudiendo derivar sus pedidos a otro robot libre sin que ello perjudique mayormente los plazos de entrega. Por otro lado, una mayor flexibilidad a la hora de resolver pedidos urgentes que en otro caso supondrían detener la producción de una orden de un robot con el consiguiente trastorno temporal que supone al equipo de producción. Esta métrica se corresponde con el número de robots respecto del total que tienen al menos una orden asignada, si un robot no tuviera asignada ninguna orden, significa que no sería necesario. Por tanto, nuestro objetivo será minimizar el número de robots necesarios, es decir, que tienen al menos una orden asignada.

5.5. Representación de las soluciones

Los individuos manejados por el algoritmo genético en este problema contienen la información de control necesaria para la asignación del conjunto de órdenes a los robots. Cabe destacar que puede que algunos de los robots no tengan ninguna orden asociada ya sea porque no son compatibles con ninguna de esas órdenes, o porque al evaluar la solución sin tenerlos en cuenta en la planificación se obtuvieron mejores resultados o porque el conjunto de órdenes era inferior a la cantidad de robots y por lo tanto no fue necesarios utilizarlos.

5.5.1. Genotipo

Para resolver el problema, se utilizará una representación basada en permutaciones, donde cada individuo cromosoma codificará una solución al problema como una cadena de enteros cada uno de los cuales representa una orden o un robot. Cada identificador de robot representa en el cromosoma un separador entre dos grupos de órdenes y la subcadena de identificadores de órdenes que tienen como robot compatible más cercano en el cromosoma al robot actual representa la secuencia de entregas que debe cumplir un robot durante los turnos de trabajo. En la figura 16 se muestra una posible representación de una solución para el problema con 10 órdenes y 4 robots. Si se encuentra en una solución 2 identificadores de robots que no están separados por ningún identificador de orden se entiende que el

robot no tiene órdenes asignadas y por lo tanto no será necesario utilizar todos los robots. Así, una posible representación de un genotipo para el problema podría ser:

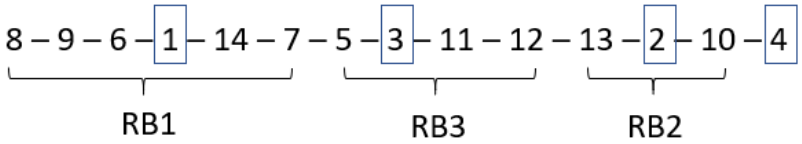


Figura 16. Representación de la solución.

En la figura 16 los robots son los representados con la numeración 1, 3, 2, 4 y los que están siendo agrupados de forma simbólica con RB1, RB3, RB2 son el conjunto de órdenes que serán producidas en el robot x, siendo RBx. En este caso el robot 4 no tiene asignado ninguna orden por lo cual no será necesario en la planificación.

5.5.2. Fenotipo

Por cada genotipo se puede obtener un fenotipo que representaría el conjunto de órdenes a realizar. En la asignación del robot no solo se agrega el identificador del robot y su posición sino también su instante de inicio y fin de acuerdo con las restricciones del problema.

5.6. Funciones objetivo

En consonancia con los objetivos a optimizar, se definen por el momento dos funciones objetivo:

- Lateness: a la hora de evaluar cada individuo, se calcula el lateness correspondiente a cada una de las órdenes que van a producir cada robot y se obtiene el mínimo de estos lateness. Este valor se minimizará.
- Número de robots: de manera paralela, a la hora de evaluar cada individuo, se calcula el número de robots dependiendo de los que tienen al menos una orden asignada.

5.7. Modelado del problema

El problema se modela de manera compatible para la resolución a partir de un Algoritmo Genético. Se utilizan los conceptos de “Robot” (cada uno de los robots que se encargan de la producción de las órdenes), además “Orden” (cada uno de los pedidos realizados por los clientes) y por último “Whelding” (conjunto de órdenes planificadas por el algoritmo). También se tienen en cuenta un conjunto de restricciones comentadas previamente a la hora de realizar la conversión del genotipo al fenotipo, es decir la decodificación de la solución a partir del cromosoma.

Para la decodificación de cualquier solución es necesario crear una clase problema, en este caso la clase creada es SchedulingProblem, que herede de una determinada clase abstracta dependiendo del tipo de codificación de la solución, que para este caso se hereda de la clase AbstractIntegerPermutationProblem. En esta clase SchedulingProblem se debe establecer el número de variables, objetivos, restricciones y el nombre del problema. Además, también es necesario sobrescribir dos métodos, getLength() donde se retornará el número máximo de soluciones, que el algoritmo utilizará para generar aleatoriamente las permutaciones y el método evaluate() donde se evaluarán los individuos generados por el algoritmo. Por otro lado, esta clase también es utilizada leer los ficheros, construir la solución e imprimir los resultados.

5.8. Software desarrollado para el problema de Soldavigil

El sistema para la planificación de células de soldadura robotizada brinda dos funcionalidades fundamentales. Una de ellas es la planificación manual por parte del cliente y la otra es la optimización de la planificación de órdenes utilizando algoritmos evolutivos.

5.8.1. Configuración general del sistema

La primera vez que el programa se lanza es necesario realizar una configuración del sistema, donde se establecen los días laborables, los horarios de inicio y fin de los turnos de trabajo, así como los días festivos. En la figura 17 se puede ver la ventana de configuración.

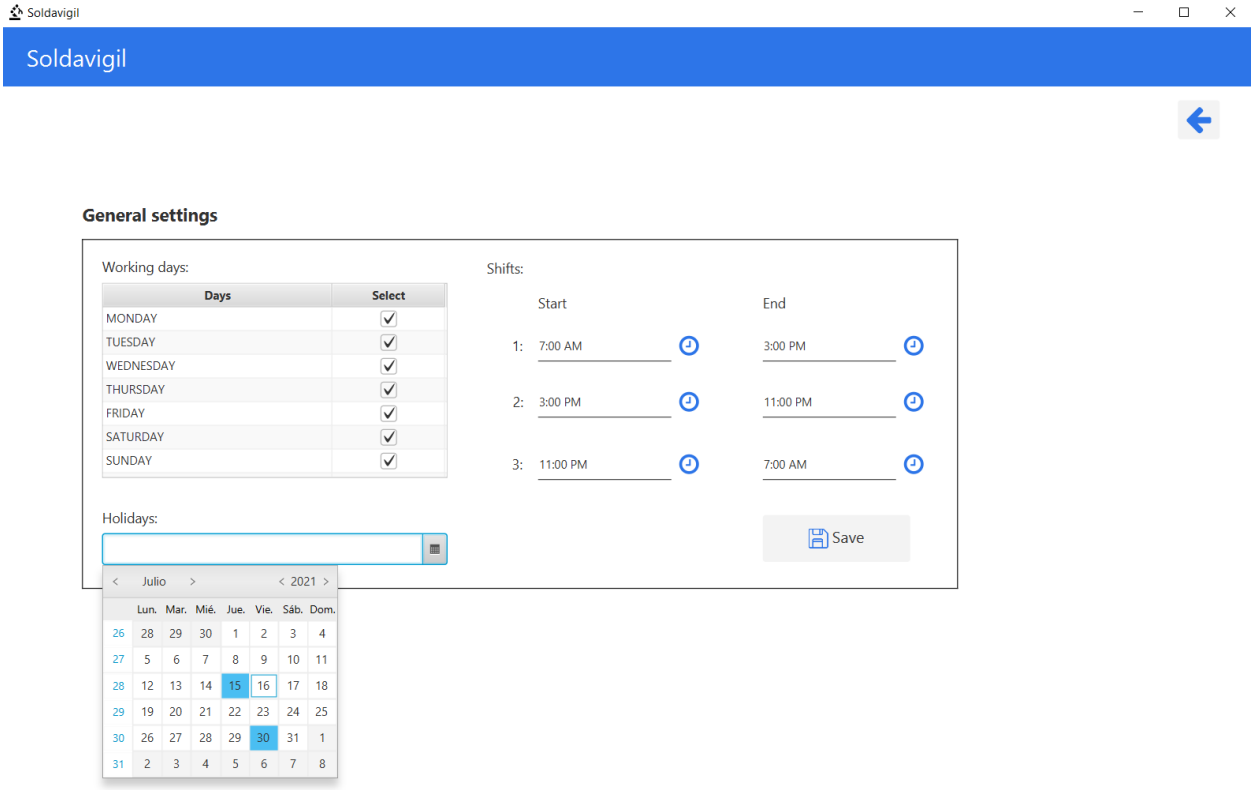


Figura 17. Ventana de configuración general del sistema.

5.8.2. Planificación de órdenes

En la ventana principal del software lo primero que se debe hacer es cargar el directorio de trabajo, donde se encuentran los archivos necesarios para planificar las órdenes, entre los cuales se encuentra orders (conjunto de órdenes a planificar), robots (conjunto de robots que serán utilizados para producir las órdenes) y planned (órdenes ya planificadas, en caso de que lo haya).

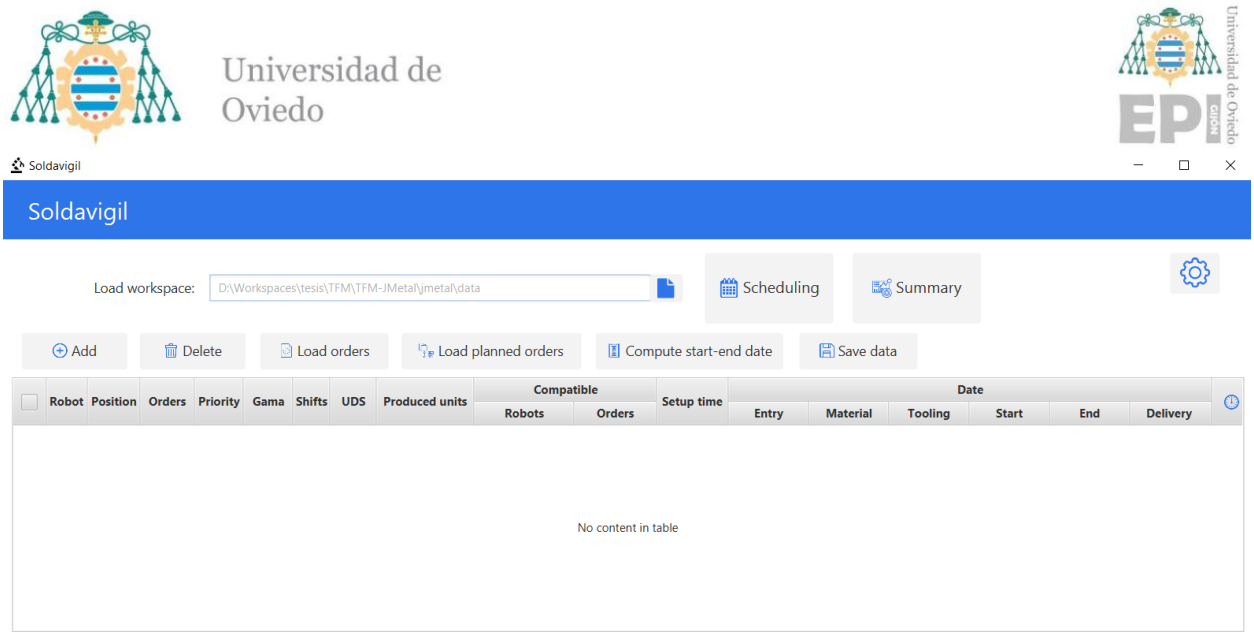


Figura 18. Ventana de planificación.

Desde esta ventana podemos acceder a un conjunto de funcionalidades que serán listadas a continuación:

- Add: agrega una nueva orden con unos valores por defecto que deben ser cubiertos por el cliente.
- Delete: elimina la orden seleccionada.
- Load orders: carga las órdenes desde el fichero orders ubicado en el directorio de trabajo.
- Load planned orders: carga las órdenes planificadas desde el fichero planned ubicado en el directorio de trabajo.
- Compute start-end date: calcula las horas de inicio y fin de todas las órdenes seleccionadas de la tabla.
- Save data: permite guardar los datos seleccionados de la tabla. Al presionar este botón sale una ventana modal para seleccionar si se quiere guardar las órdenes sin planificar o las planificadas. En la figura 19 podemos observar esta ventana modal.

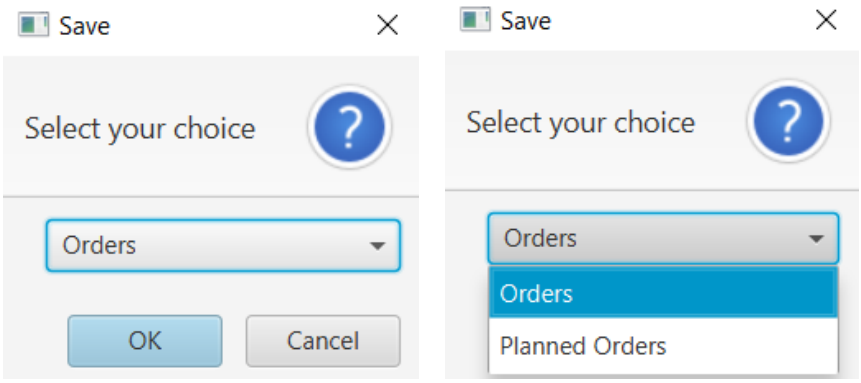
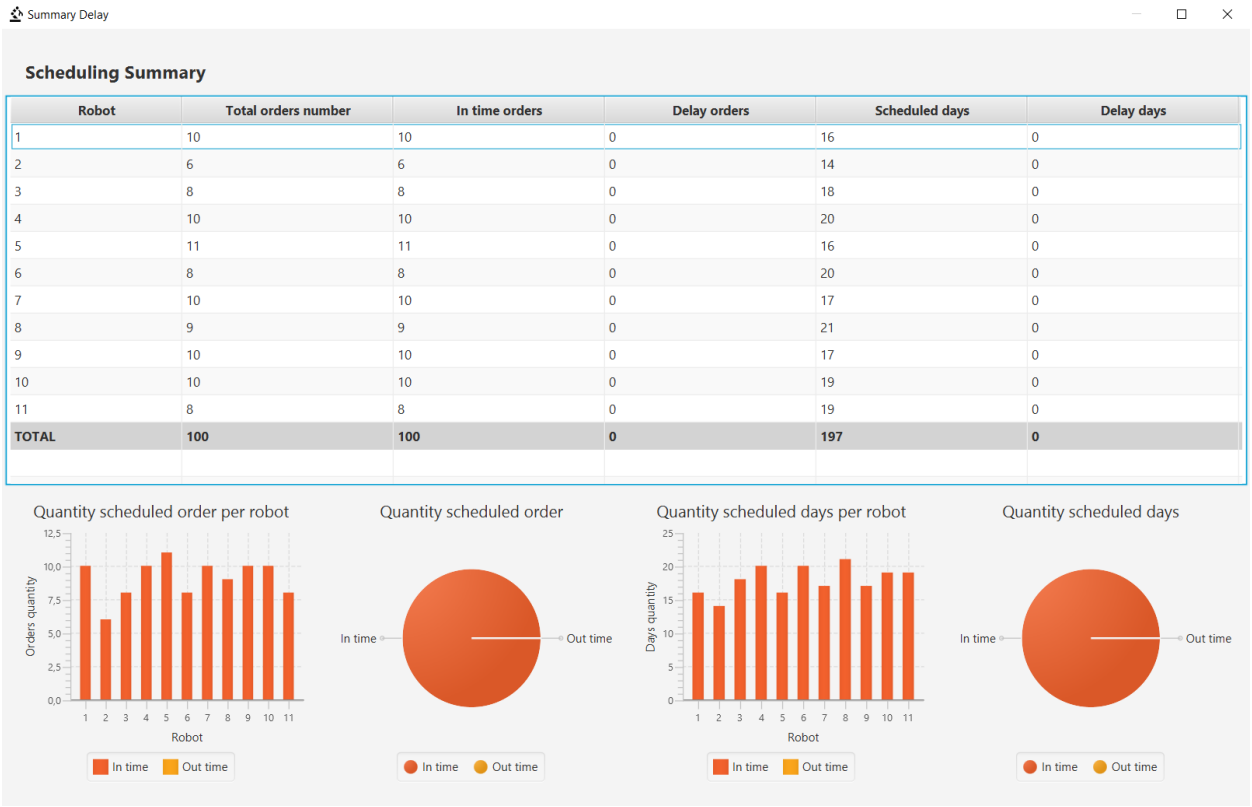


Figura 19. Ventana modal para seleccionar si guardar órdenes planificadas o sin planificar.

- Scheduling: planifica las órdenes utilizando el algoritmo.
- Summary: brinda un resumen por robot y general del número total de órdenes, dentro de estas la cantidad que están en tiempo y la cantidad que están retrasadas. Además, el total de días

planificados y el total de días de retraso. También brinda mediante gráficos un análisis de la cantidad de órdenes por robot, la cantidad total de órdenes planificadas, la cantidad de días planificados por robot y la cantidad total de días planificados. En la figura 20 podemos observar dicha ventana.



Robot	Total orders number	In time orders	Delay orders	Scheduled days	Delay days
1	10	10	0	16	0
2	6	6	0	14	0
3	8	8	0	18	0
4	10	10	0	20	0
5	11	11	0	16	0
6	8	8	0	20	0
7	10	10	0	17	0
8	9	9	0	21	0
9	10	10	0	17	0
10	10	10	0	19	0
11	8	8	0	19	0
TOTAL	**100**	**100**	**0**	**197**	**0**

Figura 20. Ventana resumen.

- Otra de las funcionalidades es la visualización de un diagrama de Gantt con la planificación. Este gráfico se carga al planificar las órdenes de forma automática o al hacerlo de forma manual y presionar en el botón de “compute start-end date”. Justo debajo de la tabla no solo aparece el diagrama de Gantt sino también un select con los diferentes robots que están participando en la planificación. Al desplegar dicho select y seleccionar el identificador del robot deseado, automáticamente se cargará la planificación para este robot. El gráfico brinda las funcionalidades de zoom in (sombreando el área del gráfico en el que queremos hacer zoom in de izquierda a derecha) y zoom out (sombreando el área del gráfico en el que queremos hacer zoom out de derecha a izquierda). Por último, haciendo click derecho sobre el gráfico podemos exportar el gráfico para un futuro análisis en los formatos de .PNG, .JPEG, .PDF, .SVG. En la figura 21 podemos observar la ventana con un diagrama de Gantt con la planificación para un robot. En el gráfico la barra de color azul significa el tiempo de producción de la pieza, para el cual se utiliza la medida de días, lo cual quiere decir que en caso de que la producción de una orden dure solo horas esa barra no se pintará. Por otro lado, tenemos la barra de color verde lo cual representa la holgura de esa hora.

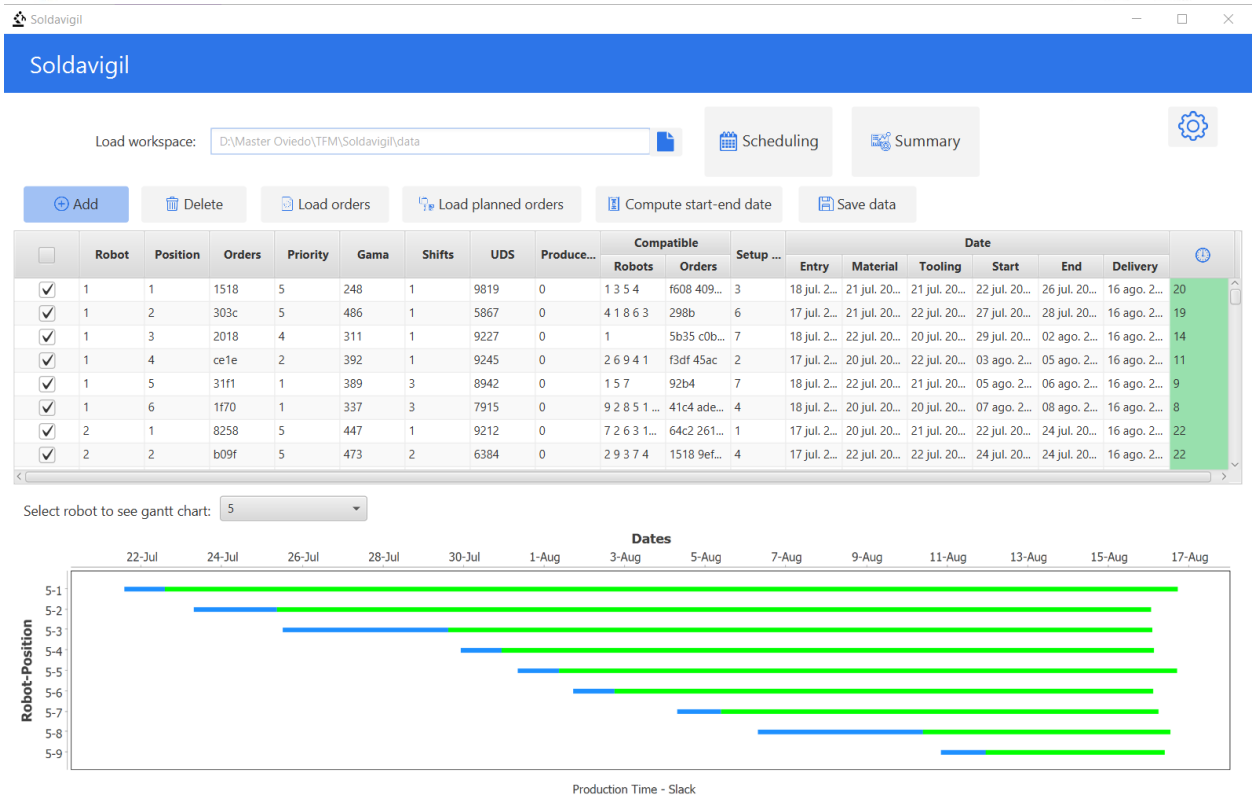


Figura 21. Ventana de planificación que muestra el diagrama de Gantt de la planificación del robot 1 para unas órdenes de ejemplo.

- En la tabla de la ventana de planificación se muestran los datos necesarios para realizar la planificación de las órdenes. No todos son obligatorios, ya que el algoritmo se encarga de dar como salida alguno de ellos en caso de ser una planificación automática entre estos tenemos el robot, la posición de la orden dentro del robot, la fecha de inicio y fin de la producción y el delay que es un campo calculado de acuerdo a la cantidad de días que hay entre la fecha fin de producción y la fecha de entrega de la orden, podría entenderse como la holgura de una orden, la cual será positivo y con fondo de color verde si se va a entregar en tiempo y negativo y de color rojo en caso contrario. Por otro lado, si la planificación se va a realizar de forma manual es necesario que el cliente llene todos los campos menos la fecha de inicio y fin de la producción y el delay.
- La tabla es editable en todos sus campos menos la fecha de inicio y fin de la producción y el delay debido a que son campos calculados.

6. Problema del cálculo de rutas de vuelo de drones en un sistema de supervisión de un área boscosa para la detección de posibles incendios.

6.1. Problema VRP (Vehicle Routing Problem)

Para introducir el problema de "cálculo de rutas de vuelo de drones en un sistema de supervisión de un área boscosa para la detección de posibles incendios" es conveniente introducir previamente el problema clásico de optimización de enrutamiento de flotas de vehículos denominado VRP (Vehicle Routing Problem) ya que nuestro problema comparte con él la representación/codificación/decodificación de sus soluciones cuando utiliza una metaheurística, como es el caso de los algoritmos genéticos, para su optimización.

6.1.1. Definición y Resolución del problema VRP

El problema de enrutamiento de vehículos (VRP) es un problema de optimización combinatoria y programación entera que busca dar servicio a un número de clientes con una flota de vehículos. Propuesto por (Dantzig 1959), el VRP es un problema importante en los campos del transporte, la distribución y la logística.

El Problema de Rutas de Vehículos (VRP) es un nombre genérico dado a toda una clase de problemas en los que hay que determinar un conjunto de rutas para una flota de vehículos con base en uno o varios depósitos para una serie de ciudades o clientes dispersos geográficamente. El objetivo del VRP es realizar las entregas a un conjunto de clientes con demandas conocidas mediante la generación de rutas de vehículos de coste mínimo con origen y destino en un depósito. En la figura siguiente podemos ver una imagen de una entrada típica para un problema VRP y una de sus posibles salidas:

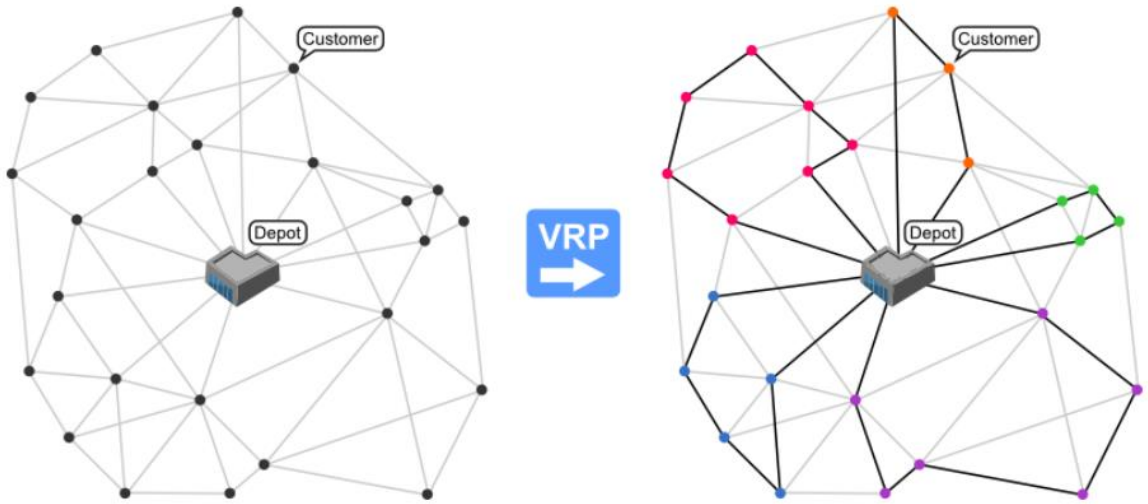


Figura 22. Una instancia de un VRP (izquierda) y su solución (derecha)

Para resolver el VRP con los algoritmos genéticos, es habitual representar a cada individuo con un solo cromosoma, que es una cadena de enteros, cada uno de los cuales representa un cliente o un vehículo.

Así, cada identificador de vehículo representa en el cromosoma un separador entre dos rutas diferentes, y una cadena de identificadores de clientes representa la secuencia de entregas que debe cubrir un vehículo durante su ruta. En la figura 23 podemos ver una representación de una posible solución para VRP con 10 clientes y 4 vehículos. Cada ruta comienza y termina en el depósito (se le asignará el número 0). Si en una solución encontramos dos identificadores de vehículo no separados por ningún identificador de cliente, entenderemos que la ruta está vacía y, por tanto, no será necesario utilizar todos los vehículos disponibles.

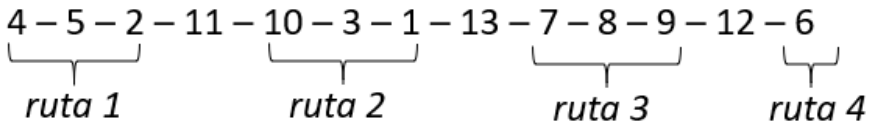


Figura 23. Representación de las soluciones

En el ejemplo, los vehículos son los etiquetados como 11, 12, 13, e implícitamente habría un cuarto vehículo al que le correspondería la ruta 4.

6.1.2. El problema de optimización de rutas de vuelos de drones para el control de incendios forestales

La Administración Pública Española lleva a cabo una tarea importante de protección del medio natural, así como instalaciones estratégicas. Los espacios naturales son protegidos especialmente del riesgo de incendio, que puede llegar a ser muy alto debido a las condiciones meteorológicas y la orografía del terreno. Las tareas de prevención y extinción de incendios son muy complejas y costosas tanto en tiempo como en recursos humanos y materiales, teniendo que realizarse, en muchas ocasiones, por distintos medios para que sean efectivas. Como ayuda a los efectivos contra incendios disponibles actualmente, y mejorar en la prevención de incendios en parques nacionales y naturales, de una o varias comunidades, se desea implantar un nuevo servicio que incluya drones como elementos de vigilancia tanto en la prevención como en la extinción de fuegos. Para el área donde se implantará el servicio, se establecerán distintas rutas de vuelos que los drones han de llevar a cabo. Estos drones podrán volar de manera automática siguiendo una ruta establecida. Los drones tomarán datos meteorológicos e imágenes del área que sobrevuelen. Esta información será transmitida, junto con información propia del dron, visualizada en tiempo real por los usuarios del sistema y almacenada para su análisis. El análisis incluirá también información procedente de fuentes externas de manera que se pueda localizar objetos o efectivos, detectar focos activos, prevenir situaciones de riesgo, proporcionar patrones de comportamiento y avance del fuego, y ayudar en la toma de decisiones.

Los drones llevan a cabo distintas rutas de vuelos para supervisar el área que les sea asignada. Mientras sobrevuelan el área toman datos meteorológicos e imágenes. En algunos casos los espacios a supervisar corresponden con superficies muy extensas, por lo que se dividen en diferentes áreas de un tamaño similar entre ellas, en las cuales se establecen Estaciones de Control que centralizan las operaciones a realizar en cada área y Bases desde donde los drones asignados despegan y aterrizan. Paralelamente, también se establecen los Barridos que cubrirán la totalidad de cada área. Para realizar una supervisión de un área determinada se debe establecer una flota de drones y sus rutas de vuelo.

6.2. Entidades relacionadas con el problema

La diferencia del problema de los drones con el VRP clásico pasa por un renombramiento de los conceptos. A partir de la descripción del problema, se definen las siguientes entidades:

Estación de Control: es la parte encargada de supervisar un área determinada (podría considerarse similar a un depósito). Todos los drones partirán de aquí y luego regresarán cuando hayan cumplido con su misión para recargar las baterías y así estar listos para una nueva misión. Esta se representa como "CS" en el fenotipo.

Bases: Son las zonas dentro de cada Estación de Control que se habilitan para aterrizar o despegar los drones vinculados con el área de la que es responsable.

Drones: vehículos aéreos que supervisan de forma autónoma las diferentes áreas a las que están asignados. Se encargan de detectar incendios mediante la realización de ciertos barridos que cubren el área a supervisar. Cada dron posee un identificador único (id = i).

Área a supervisar: Los drones sobrevuelan la zona a cubrir supervisando que no exista ningún incendio activo y comprobando las condiciones que les permita prever alguno, realizando para ello tomas fotográficas y mediciones de la temperatura del aire.

Rutas de vuelo: Son los recorridos que hará cada Dron. Está compuesta principalmente por barridos y los movimientos necesarios para posicionarse en los mismos partiendo desde la Estación de Control.

Barridos: Se trata de segmentos rectos que están paralelos al suelo y que recorrerá un dron a cierta altura, presumiblemente constante. La altura no se va a considerar en el ámbito de este estudio. Cada barrido se representará en el fenotipo por su número de secuencia.

6.3. Restricciones

- Los vuelos entre barridos consecutivos siempre se harán desde el punto de finalización del barrido actual al más próximo del siguiente barrido a recorrer.
- Las rutas siempre empiezan y acaban en una Estación de Control.

6.4. Objetivos a optimizar

En este caso se tienen dos objetivos a optimizar. Uno de ellos es minimizar el número de drones que supervisan la zona y el otro es minimizar la distancia máxima recorrida por eso drones.

- **Número de drones necesarios** (a minimizar): corresponde con el número de drones respecto del total que tienen barridos asignados; si un dron no los tuviera, significa que no es necesario.
- **Distancia máxima de la ruta** (a minimizar): la ruta más larga de entre todas las asignadas a los drones en la solución propuesta.

6.5. Representación de las soluciones

Los individuos manejados en este problema contienen la representación del conjunto de rutas asociadas a todos los drones (algunos de ellos con una ruta que no contenga ningún barrido, por lo que se consideraría ese dron como innecesario) que garantizan que se cubre toda el área a supervisar a estar todos los barridos incluidos exactamente una vez en alguna de las rutas.

6.5.1. Genotipo

El genotipo para este problema consiste en una secuencia de drones y puntos de ruta. Cada identificador de dron representa en el cromosoma un separador entre dos rutas diferentes, y una cadena de identificadores de rutas representa la secuencia de entregas que debe cubrir un dron durante su ruta. Una posible representación de un genotipo para un problema con "B" barridos y "D" drones podría ser:

Barrido 1	…	Barrido B	Dron 2	…	Dron D

Tabla 1. Genotipo del problema de FireDrone.

Como se puede apreciar no se representa el último dron, ya que implícitamente iría siempre al final del cromosoma. A partir de esta secuencia se haría una reordenación aleatoria para obtener una posible solución al problema.

Supongamos un problema donde tenemos 10 barridos, identificados por los enteros del 1 al 10, y 4 drones identificados por los enteros del 11 al 14. Un posible genotipo solución sería la permutación de la siguiente figura:

$$\underbrace{4-5-2}_{route1}-11-\underbrace{10-3-1}_{route2}-13-\underbrace{7-8-9}_{route3}-12-\underbrace{6}_{route4}$$

Figura 24. Ejemplo de cromosoma representando 10 barridos y 4 drones.

En ella podemos ver como los genes codifican una permutación de los números del 1 al 13, ya que el último dron iría implícitamente siempre después del último gen del cromosoma. La interpretación de este cromosoma supondría que hay 4 rutas que se corresponderían con los identificadores de barrido que anteceden a cada gen que codifica un dron. Así por ejemplo el primer dron (gen 11) tiene asignados los barridos: 4, 5 y 2; el segundo dron (gen 13) tiene asignados los barridos: 10, 3 y 1; el tercer dron (gen 12) tiene asignados los barridos: 7, 8 y 9; finalmente el último dron (gen 14) tiene asignado el barrido 6.

6.5.2. Fenotipo

Por cada genotipo se puede obtener un fenotipo que representaría el conjunto de rutas de puntos a recorrer por cada dron. Cada ruta siempre comienza en la posición de la estación de control y finaliza en este mismo punto. En el caso de que un dron no tenga asignado ningún barrido no será necesario hacer uso de él en esta solución.

Los elementos que componen una solución son los siguientes:

- La estación de control con sus coordenadas (x^{cs},y^{cs})
- Cada dron *Drone i*, con su identificador asociado (Id = i)
- Cada Swept (*barrido*) j, con las coordenadas de los dos puntos que lo definen ($x_1^j, y_1^j, x_2^j, y_2^j$)

Así para el ejemplo anterior tendríamos las rutas de puntos:

Drone 1: (x^{cs},y^{cs}) (x_1^4, x_2^4) (x_1^5, x_2^5) (x_1^2, x_2^2) (x^{cs},y^{cs})

Drone 2: (x^{cs},y^{cs}) (x_1^{10}, x_2^{10}) (x_1^3, x_2^3) (x_1^1, x_2^1) (x^{cs},y^{cs})

Drone 3: (x^{cs},y^{cs}) (x_1^{10}, x_2^{10}) (x_1^3, x_2^3) (x_1^1, x_2^1) (x^{cs},y^{cs})

Drone 4: (x^{cs},y^{cs}) (x_1^6, x_2^6) (x^{cs},y^{cs})

A partir de estas secuencias de puntos podríamos calcular la longitud de cada ruta como la suma de las distancias entre cada par de puntos consecutivos.

6.6. Funciones objetivo

En este caso se tienen dos objetivos a optimizar. Uno de ellos es minimizar el número de drones que supervisan la zona y el otro es minimizar la distancia máxima recorrida por eso drones.

- Número de drones necesarios (a minimizar): corresponde con el número de drones respecto del total que tienen barridos asignados; si un dron no los tuviera, significa que no es necesario.
- Distancia máxima de la ruta (a minimizar): la ruta más larga de entre todas las asignadas a los drones en la solución propuesta.

6.7. Modelado del problema

El problema se modela en base al Problema de Enrutamiento de Vehículos (VRP en inglés) que permite ser modelado de manera compatible para la resolución a partir de un Algoritmo Genético. Básicamente se han mantenido el concepto de “Vehículo” (que hará de “Dron”) y se ha adaptado el concepto de “Cliente a visitar” por “Barrido a recorrer” (Avellar et al., 2015).

Cada Barrido conllevará la realización de un recorrido interno lineal (desde el punto de origen al punto de fin, en línea recta o desde el fin al origen dependiendo del recorrido externo previo) que podrá ser 0 en aquellos casos donde los puntos de inicio y fin coincidan; y de un recorrido externo (aquel necesario para posicionarse en el punto más próximo de ambos extremos del barrido) que podrá ser 0 en caso de coincidir el punto de fin del anterior barrido y el inicio del actual.

7. Software desarrollado para los dos problemas planteados (JMetal-Benchmarking)

Esta es una aplicación que engloba los dos problemas desarrollados en este trabajo, Soldavigil y FireDrone, donde se puede optimizar probando un conjunto de algoritmos y parámetros para ambos problemas, realizar un estudio experimental o cargar un estudio experimental lanzado con anterioridad para su análisis. La ventana principal del sistema cuenta con un menú con 3 funcionalidades principales:

- Optimizar: en esta ventana se brinda la posibilidad de optimizar 2 problemas (planificación de células de soldadura robotizada - Soldavigil y optimización de rutas de vuelo de drones en un sistema de supervisión de un área boscosa para detección de posibles incendios - FireDrone).
- Experimentación: permite realizar la experimentación de los problemas antes mencionados utilizando un conjunto de algoritmos y parámetros de configuración de estos algoritmos.

- Análisis de la experimentación: brinda la posibilidad de cargar una experimentación anteriormente lanzada para su análisis.

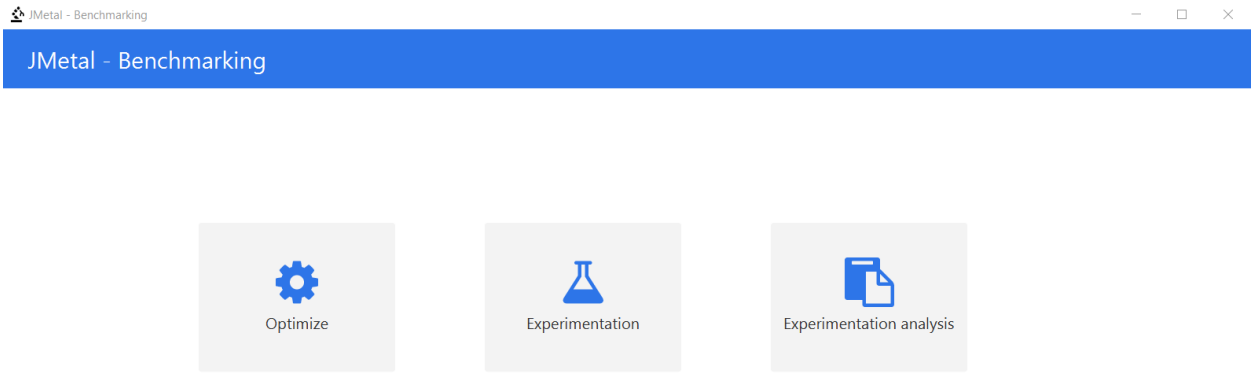


Figura 25. Menú de la aplicación

En la ventana de optimización se deben configurar los parámetros de los algoritmos. En el select del problema tenemos la opción de escoger entre el problema de Soldavigil y el de FireDrone(en caso de seleccionar este aparecería dos nuevos campos, uno para cargar el fichero con las rutas y otro para establecer el número de drones para resolver el problema). En el select de algoritmo se puede escoger entre los siguientes algoritmos: NSGAII, ESPEA, MOCELL, NSGAIII, PAES, PESA2, RANDOMSEARCH y SMSEMOA. En el caso del problema de Soldavigil solamente están disponibles los algoritmos de NSGAII y RANDOMSEARCH. En la figura 26 se muestra la ventana de configuración de parámetros.

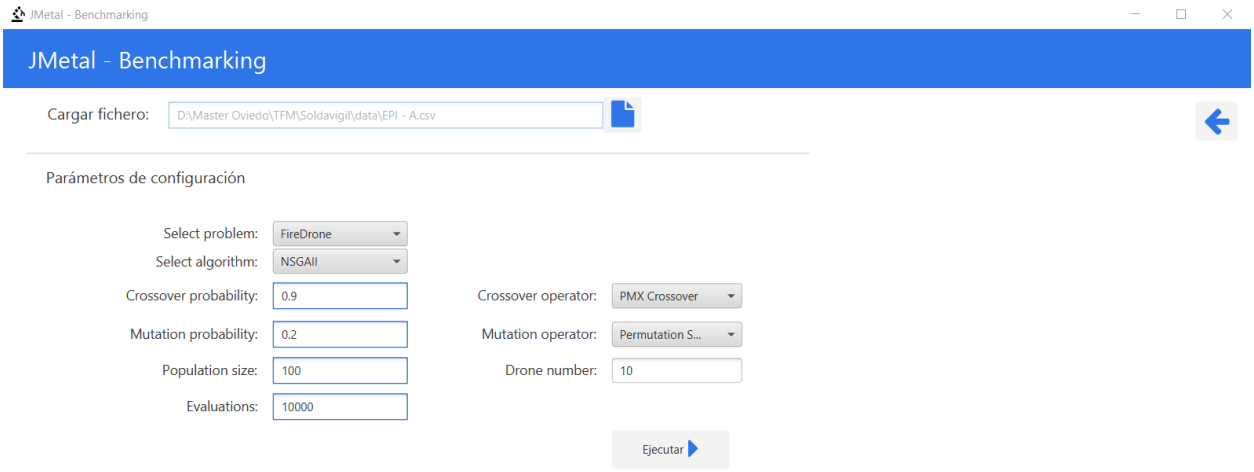


Figura 26. Ventana de configuración de parámetros.

Una vez se han seleccionado los parámetros y el algoritmo se muestra la ventana con los resultados de la optimización para el problema de los drones. En la figura 27 podemos observar dicha ventana donde en

la tabla superior se cargan las soluciones, con los barridos y los valores de las funciones objetivo. Además, se muestran dos gráficos, uno con el frente Pareto abajo a la derecha y el otro con los barridos abajo a la izquierda, este último dependiendo de la solución que esté seleccione en la tabla.

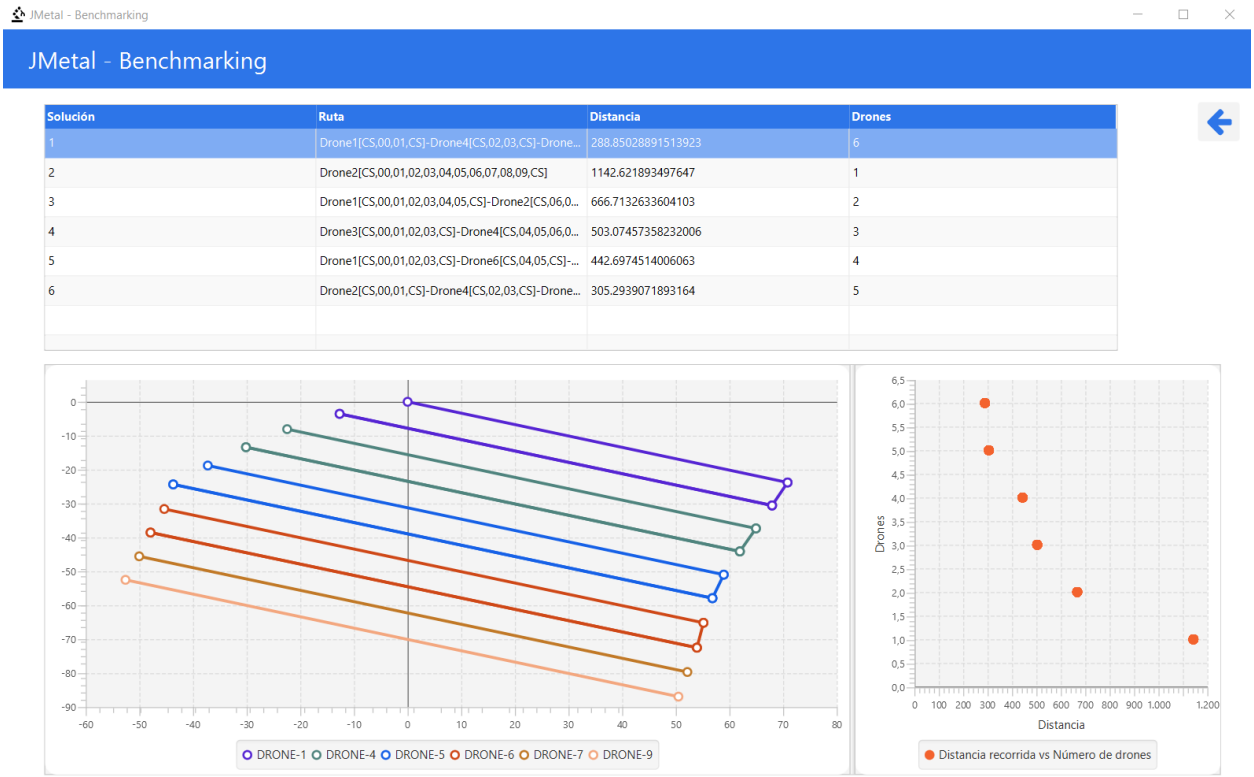


Figura 27. Ventana de solución del problema de FireDrone.

Otra de las opciones de menú es realizar un plan de experimentación. En la figura 28 se muestra la ventana donde se configuran los parámetros para realizar la experimentación. En esta ventana se debe especificar el problema que se va a resolver, el nombre con el que se guardará la experimentación, la cantidad de ejecuciones independientes y el número de cores que utilizará el software para lanzar esa experimentación. Por otro lado, cuenta con un conjunto de funcionalidades para gestionar los parámetros, adicionar una nueva configuración, eliminarla, salvar los parámetros que estén en la tabla, cargar un fichero con estas configuraciones previamente preparadas y por último lanzar la configuración (desde el botón de run). La tabla es editable en su totalidad donde se podrán realizar dos tipos de experimentaciones: un mismo algoritmo con diferentes parámetros para determinar con cuales se obtienen mejores soluciones y diferentes algoritmos con una misma configuración para decidir con que algoritmo evolutivo es más factible la resolución del problema. Para los dos problemas que actualmente están disponibles se deben cargar los ficheros con los datos y en el caso del problema de FireDrone se habilitará un campo de texto para establecer el número de drones disponibles para la supervisión del área boscosa. Para el problema de soldaduras (Soldavigil) actualmente están habilitados solamente los algoritmos de NSGAII y RandomSearch. Además, en la ventana aparece un checkbox que si está marcado habilitará la experimentación haciendo uso de una clase donde se combinarán todos los valores para dar

lugar a un conjunto de configuraciones de parámetros con los cuales será probado el algoritmo. Mientras más parámetros se agreguen mayor será el número de combinaciones por lo cual la experimentación tardará mucho más, la ventaja está en explorar todas las posibles combinaciones de una vez y no tener que hacer una experimentación incremental, o sea ir operador a operador encontrando el mejor valor para luego probar con el resto de operadores.

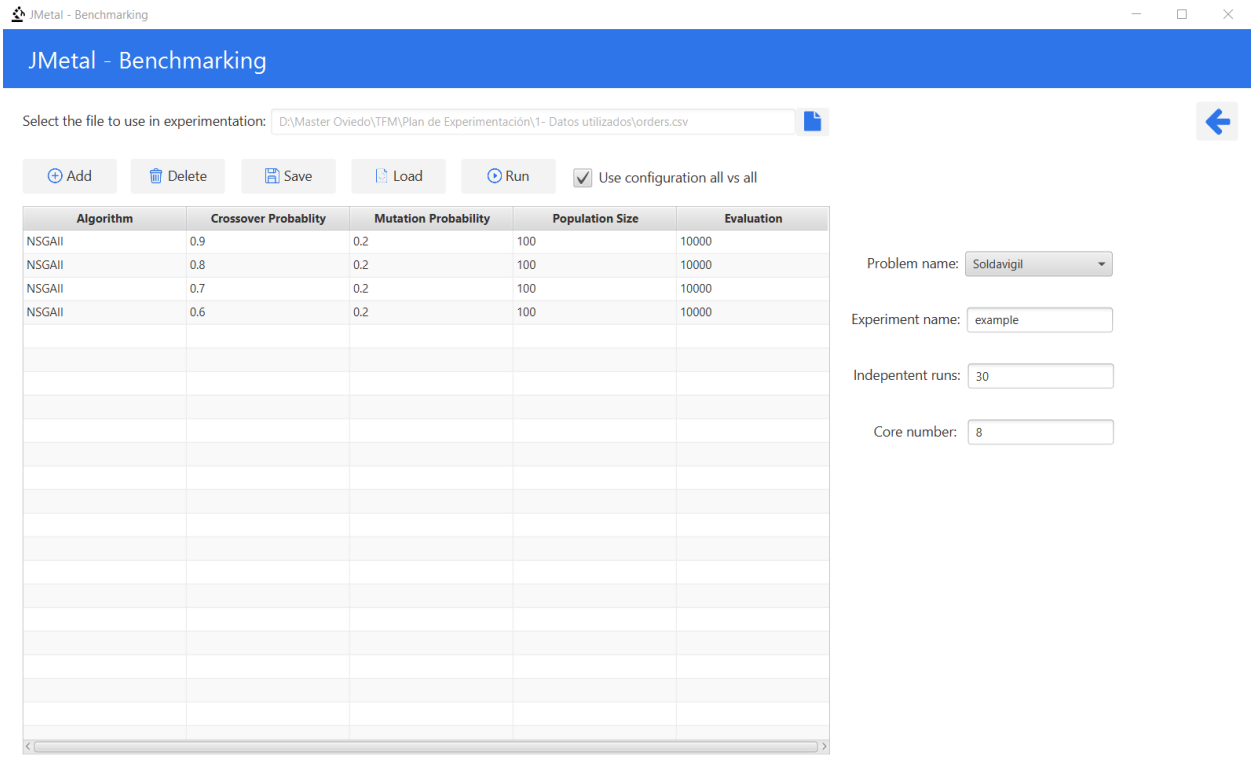


Figura 28. Ventana para preparar el plan de experimentación.

Una vez finalizada la experimentación se mostrará tal y como se ilustra en la figura 29 los distintos indicadores de calidad para ser analizados, al presionar sobre alguno de ellos se cargará un archivo html con un conjunto de métricas para la comparación entre diferentes algoritmos o también un mismo algoritmo con diferentes parámetros buscando la mejor combinación, una descripción más profunda se realizará en la sección de Experimentación.

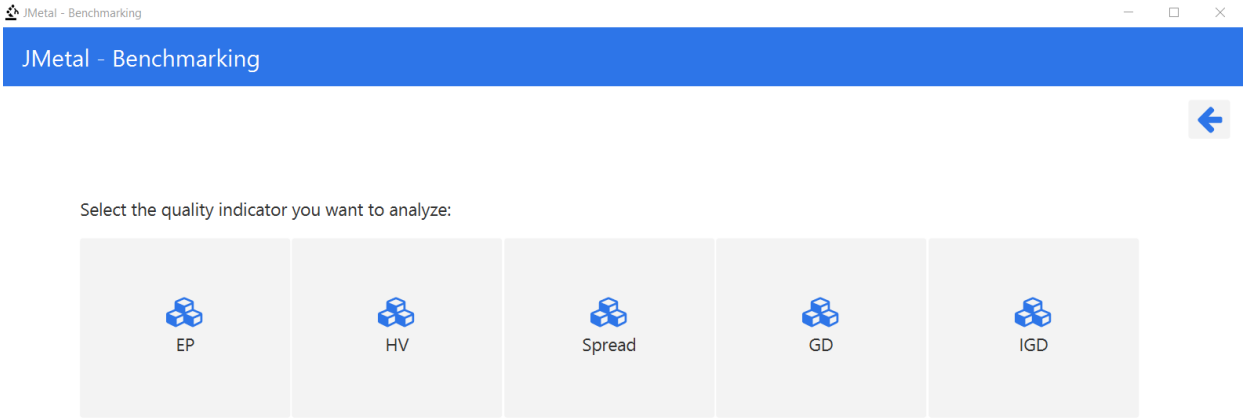


Figura 29. Selección de indicadores de calidad.

A esta misma ventana podremos acceder desde el menú en el botón de análisis de experimentación, donde debemos acceder al directorio donde se encuentra guardada nuestra experimentación y seleccionar dentro de esta el directorio de html lo cual cargará esa experimentación y nos mostrará la ventana de la figura 29 donde podremos seleccionar los indicadores y analizar los datos obtenido.

Como se mencionó anteriormente ha sido muy breve el tiempo para desarrollar el problema de soldaduras por lo cual no se cuenta con un conjunto de datos para probar el software. Por esta razón se decide implementar un generador de órdenes aleatorias para no solo probar los resultados de planificación sino también para poder experimentar con varias configuraciones buscando encontrar la mejor configuración para la resolución del problema mediante un NSGAII. En la figura 30 podemos observar esta ventana donde se deben establecer el número de órdenes a generar aleatoriamente y los rangos que se quieren utilizar.

Figura 30. Ventana de selección de rangos de parámetros.

8. Presupuesto

En esta sección se exponen a los costes asociados a la elaboración del proyecto. El presupuesto del proyecto está dado por el coste del material utilizado, el salario del personal implicado y la licencia de software utilizada. Dicha licencia se corresponde con el valor de la venta al público del proveedor. En la siguiente tabla se muestra el presupuesto del proyecto para los recursos utilizados para llevar a cabo el desarrollo del proyecto.

Elemento	Tipo de recurso	Tipo de unidad	Unidades	Precio por unidad	Costo
Equipo	Material	Unidad	1	600,00€	600,00€
Personal investigador	Personal	Jornada laboral	3	1.867,67€	5.603,01€
Windows 10 Pro	Software	Licencia	1	259,00€	259,00€
TOTAL					6.462,01€

Tabla 2. Costes directos.

Al total de costes directos del proyecto se le debe sumar los costes indirectos, entre los cuales tenemos los materiales de oficina y el consumo eléctrico. Sobre el presupuesto de ejecución debe aplicarse los porcentajes de Gastos Generales (13%) y Beneficio Industrial (6%). En la siguiente tabla se muestra el Presupuesto Base de Licitación sin IVA que asciende a la cantidad de 6.462,01€ y sobre este valor se le debe aplicar el IVA (21%) lo cual daría el Presupuesto Base de Licitación con IVA, en este caso asciende a 9.304,65€.

TOTAL DE EJECUCIÓN DEL PROYECTO	6.462,01€
Beneficio industrial 6%	387,72€
Gastos generales 13%	840,06€
PRESUPUESTO BASE SIN IVA	7.689,79€
IVA 21%	1.614,86€
PRESUPUESTO BASE CON IVA	9.304,65€

Tabla 3. Presupuesto base.

9. Planificación

En figura 31 se muestra un diagrama de Gantt con la planificación del proyecto. Esta planificación está dividida en varias partes.

- Estudio del framework: comienza con el análisis de los framework de optimización multiobjetivo. En esta fase se plantean una serie de alternativas, se hace una planificación de las tareas a realizar y se realiza un estudio de los algoritmos, operadores y funcionalidades del framework seleccionado.
- Análisis del problema de FireDrone: al no tener el problema de soldaduras, se decide utilizar el problema de los drones para aplicar los conocimientos adquiridos de la fase anterior. Para esta fase se realiza un análisis, modelado, diseño e implementación del problema utilizando el JMetal.
- Análisis del problema de Soldavigil: esta fase estuvo compuesta por el análisis, modelado, diseño e implementación del problema de soldaduras.
- Experimentación: se realiza un diseño de los experimentos, un plan de experimentación y un análisis de los resultados obtenidos.
- Desarrollo de la documentación: en paralelo al desarrollo del proyecto se redacta la documentación del proyecto.

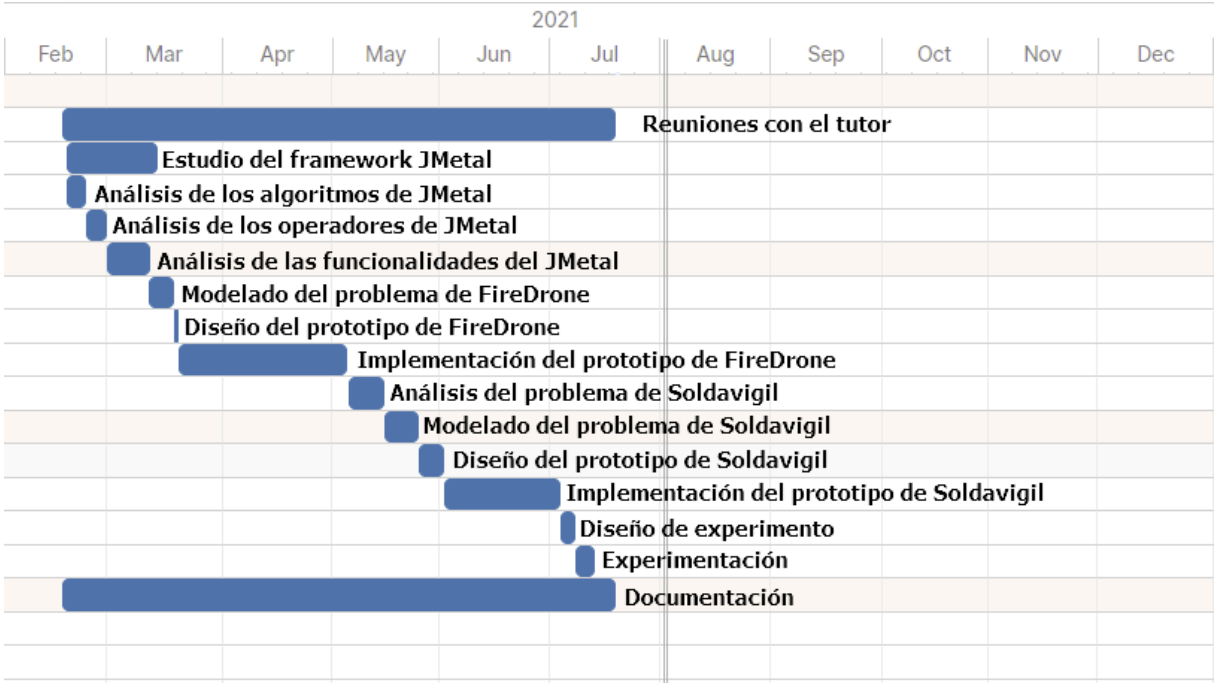


Figura 31. Diagrama de Gantt de la planificación del proyecto.

10. Experimentación.

Un estudio experimental se utiliza para comparar varias configuraciones del mismo algoritmo y/o distintos algoritmos al resolver una instancia de un problema, o un conjunto de diferentes instancias. Implica la ejecución y posterior análisis de los resultados de ***R*** ejecuciones independientes de ***A*** algoritmos, cada uno de los cuales resolverá ***P*** problemas. En este caso el proceso se lleva a cabo siguiendo los tres pasos siguientes:

1- Ejecución de todas las configuraciones (es decir, todas las combinaciones de ***A***, ***P*** y ***R***).
2- Aplicar indicadores de calidad a los frentes pareto obtenidos.
3- Evaluación del desempeño mediante el uso de pruebas estadísticas y la generación de material que resuma los resultados (por ejemplo, tablas de látex, gráficos).

Para calcular los indicadores de calidad, es necesario tener un frente de referencia por problema como se mencionó anteriormente pero este no es el caso de los problemas resueltos en esta investigación, planificación de rutas de vuelo y planificación de células de soldadura robotizada, ya que la representación estándar de sus soluciones se basa en permutaciones por lo que fue necesario realizar una modificación de dicha clase para contemplar problemas cuyas soluciones son codificadas mediante permutaciones.

Para realizar la experimentación es necesario primero configurar un conjunto de parámetros. El experimento se configura usando la clase **ExperimentBuilder** y la configuración incluye:

- El directorio base de la experimentación.
- Número de ejecuciones independientes para cada configuración.
- Nombre de la experimentación.

- El o los algoritmos con sus parámetros de configuración específicos y la lista de instancias a resolver.
- El directorio que contiene los frentes de referencia de las instancias.
- El prefijo predeterminado de los nombres de los archivos de salida que contienen las soluciones (VAR) y los valores de sus funciones objetivo (FUN). Para cada combinación de algoritmo e instancia, los archivos de salida serán FUN0.csv, FUN1.csv, . . ., FUN29.csv y VAR0.csv, VAR1.csv, . . ., "VAR29.csv". El número queda determinado por el número de ejecuciones independientes establecido.
- Una lista con los indicadores de calidad multiobjetivo que se desea utilizar.
- El número de núcleos. Un experimento puede requerir una gran cantidad de tiempo de cálculo. Este parámetro indica el número de núcleos que se utilizarán.

Luego de ejecutar el algoritmo se creará un directorio llamado datos que contendrá una carpeta por cada configuración, cada una de las cuales a su vez almacena una subcarpeta por instancia resuelta. Dentro de cada subdirectorio se encuentran los archivos antes mencionados FUNi.csv y VARi.csv, donde ***i*** toma valores en el rango de (0, ***N*** - 1), donde ***N*** es el número de ejecuciones antes comentado. Después de ejecutarse los indicadores de calidad seleccionados dentro del subdirectorio también se almacenarán los archivos referentes a los indicadores. Para poder explicar mejor la estructura de archivos tomaremos de referencia el indicador **epsilon**, los resultados del cálculo de este indicador para cada archivo FUNi.csv en un archivo llamado EP (el nombre corto del indicador), que contiene una línea por valor de indicador (en nuestro ejemplo, este archivo contiene ***N*** líneas teniendo en cuenta el número de ejecuciones seleccionado). También se crean, con la intención de saber qué frentes tienen los mejores valores o las medianas, los archivos: **BEST_EP_FUN.csv**, **BEST_EP_VAR.csv**, **MEDIAN_EP_FUN.csv** y **MEDIAN_EP_VAR.csv**. El mismo proceso se realiza para el resto de los indicadores de calidad seleccionados.

Otro archivo generado es el **QualityIndicatorSummary.csv**, el cual contiene un resumen de todos los valores del indicador de calidad. El encabezado contiene los siguientes datos:

- Nombre del algoritmo
- Nombre del problema
- Nombre del indicador
- Id de la ejecución (de 0 al número de ejecuciones menos 1)
- Valor del indicador

Los test estadísticos utilizados para la experimentación son el de Wilcoxon (Demˇ, 2006), además el de Friedman (Demˇ, 2006; Martin et al., 1993) y el test de Holm (Holm, 1979).

10.1. Generación de tablas en Latex

Luego de obtener los valores de los indicadores de calidad de los frentes obtenidos por todos los algoritmos sobre los problemas podemos realizar un análisis estadístico utilizando algunas de las funcionalidades que nos ofrece JMetal podremos generar archivos Latex que contienen datos estadísticos (media / mediana y desviación estándar, test de Friedman (Demˇ, 2006) y scripts R que producen Boxplots y tablas de Latex que contienen información sobre el test de Wilcoxon.

10.2. Generación de páginas html

A pesar de las grandes ventajas que nos ofrece las tablas Latex resulta un poco engorroso analizarlas todas por separado por lo que la funcionalidad de generar una página html que contenga todos los resultados obtenidos de la experimentación resulta muy interesante. Esta funcionalidad crea una carpeta llamada html y genera un archivo HTML por indicador de calidad (es decir, EP.html, HV.html, etc.). Cada página contiene lo siguiente:

- Tabla con los valores promedios.
- Tabla con los resultados del test de Wilcoxon.
- Tabla con la clasificación obtenida tras aplicar el test de Friedman y Holm.
- Un boxplot por instancia.
- Un gráfico con el frente Pareto

10.3. Tabla de valores promedios

Dentro del archivo html se genera una tabla donde las filas son los problemas que se están resolviendo y las columnas son los algoritmos utilizados para resolver dichos problemas. Por otro lado, cada celda se corresponde con los valores promedios del indicador de calidad para cada par de problema-algoritmo. Para cada problema se marcan los algoritmos que obtiene el mejor y el segundo mejor valor del indicador con diferentes tonos de verdes como color de fondo, siendo el más intenso el que obtiene un mejor valor (máximo o mínimo dependiendo del indicador en cuestión). Los resultados se ordenan de acuerdo a los valores de las funciones objetivo, si es maximizar de mayor a menor y si es minimizar de menor a mayor. Al tener varios resultados con el mismo valor se utiliza el indicador de hipervolumen para desempatar.

En la figura 32 se muestra un ejemplo de visualización para el problema de soldadura utilizando el algoritmo NSGAII con cuatro configuraciones de parámetros diferentes y se resaltan, tal y como se comentó anteriormente los mejores resultados obtenidos en verde.

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	5E-1	5,71429E-1	5,71429E-1	5,71429E-1

Figura 32. Ejemplo de visualización de la tabla de valores medianos para el problema de Soldavigil con el algoritmo NSGAII y cuatro configuraciones diferentes.

10.4. Tabla del test de Wilcoxon

Es una prueba de hipótesis estadística no paramétrica que permite hacer comparaciones por pares entre configuraciones para determinar si existen diferencias significativas entre ellas, se utiliza al no poder suponer la normalidad de cada conjunto de datos obtenidos (Demˇ, 2006). Estos resultados se resumen en una tabla después de aplicar la prueba a los valores del indicador de calidad a los frentes Pareto calculados por las diferentes configuraciones utilizadas en el estudio experimental. En cada celda cada problema se representa con uno o más símbolos, uno por cada instancia a resolver. Se utilizan tres símbolos diferentes: = indica que no hay diferencias estadísticas significativas entre el rendimiento de la

configuración de la fila y la configuración de la columna correspondiente, ⓤ significa que la configuración en la fila ha obtenido mejores resultados que la configuración de la columna y ⓓ se utiliza cuando la configuración en la columna es estadísticamente mejor que la configuración en la fila.

Este test puede encontrarse en la experimentación en dos formatos diferentes, uno en los archivos html en forma de tabla y otro en los archivos R, este último puede ser abierto utilizando el programa de R Studio (https://www.rstudio.com/). En la figura 33 se muestra un ejemplo de visualización en el archivo html.

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	=
NSGAII-1	=		=	=
NSGAII-2	=	=		=
NSGAII-3	=	=	=	

Figura 33. Ejemplo de visualización del test de Wilcoxon en la tabla mostrada en los archivos html de los diferentes indicadores.

10.5. Tabla del test de Friedman y Holm

Es una prueba estadística no paramétrica apropiada para probar la hipótesis de que los datos de ***k*** muestras emparejadas se extraen de la misma población o en situaciones en las que se obtienen múltiples medidas correlacionadas en los mismo sujetos (Martin et al., 1993) (Holm, 1979). Clasifica las configuraciones para cada conjunto de datos por separado, la configuración de mejor rendimiento obtiene el rango 1, el segundo mejor el rango 2 y así sucesivamente, en caso de empates se asignan rangos promedios. Los datos se organizan en una tabla bidireccional que tiene ***N*** filas y ***M*** columnas. Las filas corresponden a los bloques de configuraciones y las columnas representan los diferentes tratamientos (ranking, p-value, holm e hypothesis). Los valores en las filas son los rangos de las puntuaciones para las diferentes configuraciones bajo cada uno de los ***M*** tratamientos.

El test de Friedman y Holm se encuentra en dos formatos diferentes al igual que el test de Wilcoxon, en una tabla en el archivo html de cada indicador y otro como un archivo de latex (.tex). En la figura 34 se muestra un ejemplo de visualización de la tabla en el archivo html.

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-1	1	0E0	0	-
NSGAII-0	2	1,973E-9	0,05	Rejected
NSGAII-2	3,5	7,342E-51	0,025	Rejected
NSGAII-3	3,5	7,342E-51	0,017	Rejected

Figura 34. Ejemplo de visualización del Test de Friedman y Holm en la tabla mostrada en los archivos html de los diferentes indicadores.

10.6. Boxplots

Dentro de la funcionalidad de mostrar los resultados en una página html de la experimentación podemos contar con la visualización de los resultados en un boxplot por cada configuración. Este histograma no solo nos permite saber que configuración resulta mejor en promedio (mediana) sino también medir su robustez entre las diversas ejecuciones de la configuración: rango entre el primer y tercer cuartil (RIC), también entre el mínimo y el máximo, así como la existencia o no de valores atípicos. Así como saber que configuración es capaz de alcanzar el mejor valor (máximo o mínimo valor, dependiendo del tipo de indicador). En la figura 35 se muestra un ejemplo del gráfico.

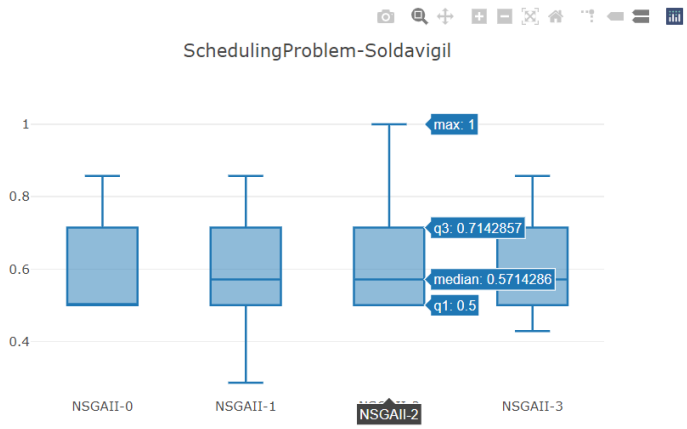


Figura 35. Ejemplo de visualización del boxplot

10.7. Gráfico de frente Pareto

Otra de las funcionalidades brindadas es el gráfico del frente Pareto. Dentro de la página html generada para cada indicador, se muestra primero la tabla de medias, la tabla que contiene los resultados de la prueba de Wilcoxon y la tabla con los resultados de la prueba Friedman y Holm, después por cada configuración se muestra un boxplot y los frentes Pareto para cada algoritmo. Luego de ejecutar la experimentación quedan guardados los mejores resultados obtenidos de todas las ejecuciones y son estos los mostrados en el frente Pareto de cada algoritmo. Cada frente muestra las mejores soluciones

obtenidas por cada función objetivo para cada instancia del problema. En la figura 36 se muestra un ejemplo de gráfico de frente Pareto.

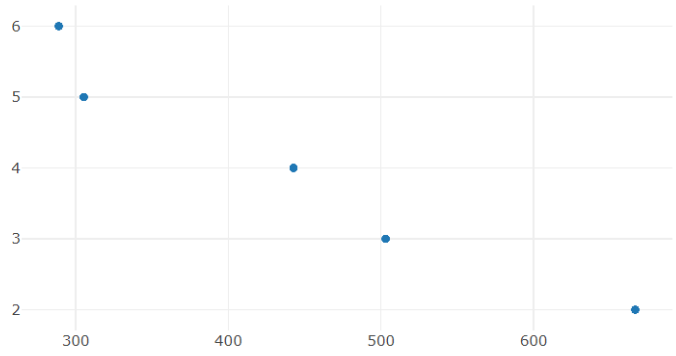


Figura 36. Ejemplo de visualización del frente Pareto.

10.8. Plan de experimentación

En este plan de experimentación se contemplan cuatro experimentos individuales tres de estos para el problema de Soldavigil y el cuarto para el problema de los drones FireDrone. El primer experimento es para determinar que parámetros de configuración del algoritmo NSGAII hace que su efectividad en la resolución del problema de Soldavigil sea mejor, luego de encontrar esos parámetros se realiza un segundo experimento con la intención de comprobar aumentando el número de evaluaciones el algoritmo es capaz de converger a mejores soluciones. Una vez determinado que parámetros de configuración se utilizará en el algoritmo NSGA se decide comparar este algoritmo con esas configuraciones contra un algoritmo Random Search. El último experimento se realiza sobre el problema de FireDrone utilizando dos instancias del problema, el área de la Epi y el otro el bosque de Muniellos, este último significativamente más extenso que la primera área. La idea es comprobar con una misma configuración cómo se comportan los algoritmos NSGAII, NSGAIII, ESPEA, PAES, PESA2, MOCELL, SMSEMOA y RANDOM SEARCH en la resolución de las dos instancias, con un mismo número de drones.

Para todos los experimentos se utilizan 30 ejecuciones independientes (Teghem, 2010), 8 núcleos del procesador y en el caso de FireDrone 50 drones. Las diferentes configuraciones que se establecen en la tabla del software para realizar los experimentos son nombrados como el nombre del algoritmo que se está seleccionado más la posición que ocupa en la tabla, de esta forma para dos configuraciones de NSGAII, se devolvería en el directorio del experimento NSGAII-0, NSGAII-1 y para dos configuraciones de NSGAII y ESPEA se devolvería NSGAII-0 y ESPEA-1.

10.8.1. Conjunto de datos utilizados en el problema de Soldavigil

Para la resolución del problema de Soldavigil se implementa una clase para la generación de datos aleatorios debido a que actualmente se cuenta con muy pocos ejemplos para tener una buena experimentación. En el generador de datos aleatorios el usuario debe establecer un conjunto de valores, siendo los rangos, entre los cuales se generarán los resultados. A continuación, se muestra en una tabla los rangos utilizados para generar un conjunto de 100 órdenes para la resolución del problema.

Número de órdenes	UDS	Gama	Tiempo de preparación	Prioridad	Turnos	Fecha			
						Entrada	Material	Utillaje	Entrega
100	5000-10000	200-500	1-10	1-5	1-3	16 jul.-19 jul.	20 jul.-23 jul.	20 jul.-23 jul.	15 ago.-16ago.

Tabla 4. Rangos establecidos para la generación de órdenes aleatorias.

10.8.2. Conjuntos de datos utilizados en el problema de FireDrone

Para la resolver el problema de los drones se utilizaron dos conjuntos de datos, uno del área de la EPI y el otro de área del bosque de Muniellos. A continuación, se muestran dos figuras de ambos conjuntos de datos. Cabe destacar que en cada conjunto de datos se muestra en una columna X e Y las coordenadas de la estación de control, mientras que X1, Y1 y X2, Y2 sería las coordenadas de inicio y fin de cada barrido.

X	Y		
83,840975	-58,859385		
X1	Y1	X2	Y2
0	0	70,831168	-23,785642
-12,676222	-3,547689	67,9401	-30,558503
-22,461375	-8,062929	64,937837	-37,331363
-30,133825	-13,384463	61,935574	-44,104224
-37,2503	-18,786626	58,933311	-50,957714
-43,699606	-24,350047	56,820608	-57,891834
-45,36753	-31,606683	55,152684	-65,14847
-47,925013	-38,540803	53,929539	-72,485736
-50,037717	-45,555551	52,150421	-79,661743
-52,5952	-52,489671	50,482497	-86,91838

Figura 37. Conjunto de datos del área de la EPI.

X	Y		
2000	-200		
X1	Y1	X2	Y2
0	0	0	5000
57,6	0	57,6	5000
115,2	0	115,2	5000
172,8	0	172,8	5000
230,4	0	230,4	5000
288	0	288	5000
345,6	0	345,6	5000
403,2	0	403,2	5000
460,8	0	460,8	5000
518,4	0	518,4	5000
576	0	576	5000
633,6	0	633,6	5000
691,2	0	691,2	5000
748,8	0	748,8	5000
806,4	0	806,4	5000
864	0	864	5000
921,6	0	921,6	5000
979,2	0	979,2	5000
1036,8	0	1036,8	5000
1094,4	0	1094,4	5000
1152	0	1152	5000
1209,6	0	1209,6	5000
1267,2	0	1267,2	5000
1324,8	0	1324,8	5000
1382,4	0	1382,4	5000
1440	0	1440	5000

1497,6	0	1497,6	5000
1555,2	0	1555,2	5000
1612,8	0	1612,8	5000
1670,4	0	1670,4	5000
1728	0	1728	5000
1785,6	0	1785,6	5000
1843,2	0	1843,2	5000
1900,8	0	1900,8	5000
1958,4	0	1958,4	5000
2016	0	2016	5000
2073,6	0	2073,6	5000
2131,2	0	2131,2	5000
2188,8	0	2188,8	5000
2246,4	0	2246,4	5000
2304	0	2304	5000
2361,6	0	2361,6	5000
2419,2	0	2419,2	5000
2476,8	0	2476,8	5000
2534,4	0	2534,4	5000
2592	0	2592	5000
2649,6	0	2649,6	5000
2707,2	0	2707,2	5000
2764,8	0	2764,8	5000
2822,4	0	2822,4	5000
2880	0	2880	5000
2937,6	0	2937,6	5000
2995,2	0	2995,2	5000
3052,8	0	3052,8	5000
3110,4	0	3110,4	5000

3168	0	3168	5000
3225,6	0	3225,6	5000
3283,2	0	3283,2	5000
3340,8	0	3340,8	5000
3398,4	0	3398,4	5000
3456	0	3456	5000
3513,6	0	3513,6	5000
3571,2	0	3571,2	5000
3628,8	0	3628,8	5000
3686,4	0	3686,4	5000
3744	0	3744	5000
3801,6	0	3801,6	5000
3859,2	0	3859,2	5000
3916,8	0	3916,8	5000
3974,4	0	3974,4	5000

Figura 38. Conjunto de datos del área del bosque de Muniellos.

10.8.3. Experimento 1: Comparar los resultados del NSGAII con diferentes configuraciones de parámetros en el problema de Soldavigil.

Objetivo: determinar con que configuración se obtienen los mejores resultados.

La idea de este experimento tal y como se comenta en el objetivo es saber con qué configuración el NSGAII es más eficiente en la resolución del problema planteado. Los parámetros que se tienen en cuenta son la probabilidad de cruce, probabilidad de mutación, tamaño de población y el número de evaluaciones. El análisis se hace probando un conjunto de valores para el primer parámetro y se selecciona el parámetro que arroja mejores valores en la mayor cantidad de indicadores, luego se mantiene ese valor en ese parámetro y se prueba con un conjunto de valores para el segundo parámetro y así sucesivamente hasta obtener la configuración final. Cabe destacar que es una tarea algo repetitiva y que pudiera ser automatizada, lo cual se realizó, pero queda pendiente de algunas pruebas en futuras versiones. En la siguiente tabla podemos observar los diferentes valores con los que se probó y resaltado en color amarillo el que fue seleccionado para cada parámetro, además se explicará el procedimiento seguido para la selección del parámetro de la probabilidad de cruce que es el mismo procedimiento seguido para el resto, en los anexos se puede observar los resultados de los experimentos para cada indicador validando el porqué de la selección del resto de los parámetros.

Probabilidad de cruce	Probabilidad de mutación	Tamaño de la población	Número de evaluaciones
0.9	0.1	100	10000
0.8	0.2	200	15000
0.7	0.3	300	20000
0.6	0.4	400	25000

Tabla 5. Parámetros utilizados en la experimentación de Soldavigil para encontrar la mejor configuración para el algoritmo NSGAII

Algorithm	Crossover Probablity	Mutation Probability	Population Size	Evaluation
NSGAII	0.9	0.2	100	10000
NSGAII	0.8	0.2	100	10000
NSGAII	0.7	0.2	100	10000
NSGAII	0.6	0.2	100	10000

Figura 39. Configuraciones utilizadas para probar que probabilidad de cruce obtenía los mejores resultados con el algoritmo NSGAII.

Una vez termina la experimentación probando con esas diferentes probabilidades se comienza a hacer un análisis por cada indicador de calidad. Por cada uno de estos el software muestra los valores medios, el test de Wilcoxon, el test de Friedman y Holm, así como un Boxplot que nos permiten tomar decisiones en base a que probabilidad obtiene los mejores resultados en la mayor cantidad de indicadores. En la figura 40 podemos observar los valores medios obtenidos en el indicador de calidad épsilon (EP).

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	5E-1	5,71429E-1	5,71429E-1	5,71429E-1

Figura 40. Median values obtenido en el indicador de calidad EP

Como se comentó en subapartados anteriores en los valores medio el mejor valor se seleccionaba con un color de fondo verde en un tono oscuro, el segundo mejor en un tono verde más claro y si todos obtenían el mismo resultado no se aplicaba ningún color de fondo en las casillas. En este caso tenemos un ganador que es la probabilidad de cruce de 0.9.

Otro de los test que brinda la experimentación es el de Wilcoxon que se explicó como analizarlo en un subapartado anterior. En este caso en la figura 41 se muestra que no hay diferencias significativas entre las configuraciones.

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	=
NSGAII-1	=		=	=
NSGAII-2	=	=		=
NSGAII-3	=	=	=	

Figura 41. Test de Wilcoxon obtenido en el indicador de calidad EP.

Además, se aplica el test de Friedman donde se ordenan las diferentes configuraciones en base a sus funciones objetivos y se establece un ranking, luego se coloca en la tabla el de menor ranking en primera posición y se aplica el test de Holm para ver si hay diferencias significativas entre el de ranking menor y el resto.

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-1	1	0E0	0	-
NSGAII-0	2	1,973E-9	0,05	Rejected
NSGAII-2	3,5	7,342E-51	0,025	Rejected
NSGAII-3	3,5	7,342E-51	0,017	Rejected

Figura 42. Test de Friedman y Holm obtenido en el indicador de calidad EP.

En la figura 43 se muestra el gráfico boxplot para la experimentación. En el gráfico podemos observar que las cuatro configuraciones difieren principalmente en los valores extremos (los bigotes del boxplot) y fijándose únicamente en él cabría escoger la segunda configuración, por tener la mediana igual a la primera y la tercera configuración, pero sus valores extremos tienen un mejor rendimiento.

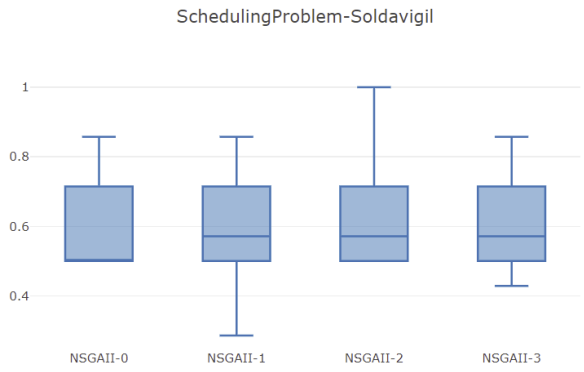


Figura 43. Boxplot obtenido en el indicador de calidad EP.

Como se mencionó anteriormente este mismo proceso se realiza para cada uno de los indicadores y la probabilidad de cruce que obtiene mejores resultados en más indicadores es el elegido. Y así se hace con cada uno de los parámetros dando como resultado la configuración seleccionada con la que en la siguiente sección se compara el NSGAII con un algoritmo Random Search.

10.8.4. Experimento 2: Comparar el desempeño del NSGAII con la configuración seleccionada con un mayor número de evaluaciones en la resolución del problema Soldavigil.

Objetivo: determinar si el algoritmo es capaz de converger hacia mejores soluciones con un mayor número de evaluaciones.

Mientras mayor número de población, evaluaciones o ejecuciones independientes tenga el algoritmo mayor será el tiempo que tomará para completar una experimentación. En algunos casos el algoritmo es capaz de encontrar mejores soluciones cuando tiene por ejemplo un mayor número de evaluaciones por lo cual sería interesante sacrificar quizás velocidad por mejores soluciones, en otros casos puede que, aunque tenga más evaluaciones siga convergiendo hacia las mismas soluciones y por lo tanto no sea capaz de arrojar una mejor solución. Por esta razón se decide hacer un experimento para comprobar ante cuál de los casos estamos y tomar la decisión de si las diferencias entre ambas configuraciones no son significativas quedarnos con la que alcance una solución suficientemente buena para la resolución del problema. En la siguiente tabla se muestra las configuraciones utilizadas para la experimentación.

Algoritmo	Probabilidad de cruce	Probabilidad de mutación	Tamaño de la población	Número de evaluaciones
NSGAII	0.9	0.4	100	25000
NSGAII	0.9	0.4	100	50000

Tabla 6. Resumen de configuración de parámetros utilizados con el algoritmo NSGAII.

Median values

	NSGAII-0	NSGAII-1
SchedulingProblem-Soldavigil	6,66667E-1	6,66667E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1
NSGAII-0		=
NSGAII-1	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-1	1	0E0	0	-
NSGAII-0	2	9,486E-15	0,05	Rejected

Figura 44. Valores medios, test de Wilcoxon, test de Friedman y Holm obtenido de la comparación de ambas configuraciones en el indicador EP.

Median values

	NSGAII-0	NSGAII-1
SchedulingProblem-Soldavigil	1,52976E-2	1,13217E-2

Wilcoxon Test

	NSGAII-0	NSGAII-1
NSGAII-0		=
NSGAII-1	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-1	1	0E0	0	-
NSGAII-0	2	9,486E-15	0,05	Rejected

Figura 45. Valores medios, test de Wilcoxon, test de Friedman y Holm obtenido de la comparación de ambas configuraciones en el indicador GD.

Median values

	NSGAII-0	NSGAII-1
SchedulingProblem-Soldavigil	1,79487E-1	2,05128E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1
NSGAII-0		=
NSGAII-1	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-1	1	0E0	0	-
NSGAII-0	2	9,486E-15	0,05	Rejected

Figura 46. Valores medios, test de Wilcoxon, test de Friedman y Holm obtenido de la comparación de ambas configuraciones en el indicador HV.

Median values

	NSGAII-0	NSGAII-1
SchedulingProblem-Soldavigil	2,88462E-1	2,78846E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1
NSGAII-0		=
NSGAII-1	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-1	1	0E0	0	-
NSGAII-0	2	9,486E-15	0,05	Rejected

Figura 47. Valores medios, test de Wilcoxon, test de Friedman y Holm obtenido de la comparación de ambas configuraciones en el indicador IGD+.

Median values

	NSGAII-0	NSGAII-1
SchedulingProblem-Soldavigil	1,40106E0	1,36423E0

Wilcoxon Test

	NSGAII-0	NSGAII-1
NSGAII-0		=
NSGAII-1	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-1	1	0E0	0	-
NSGAII-0	2	9,486E-15	0,05	Rejected

Figura 48. Valores medios, test de Wilcoxon, test de Friedman y Holm obtenido de la comparación de ambas configuraciones en el indicador SPREAD.

Como se puede comprobar en las figuras 44-48, en todos los indicadores a pesar que la configuración con 50000 evaluaciones obtiene los mejores resultados en los valores medianos, sin embargo, no existen

diferencias significativas entre ambas configuraciones por lo cual se mantiene la selección de la configuración resultante del experimento 1 para ser utilizada en el experimento 3.

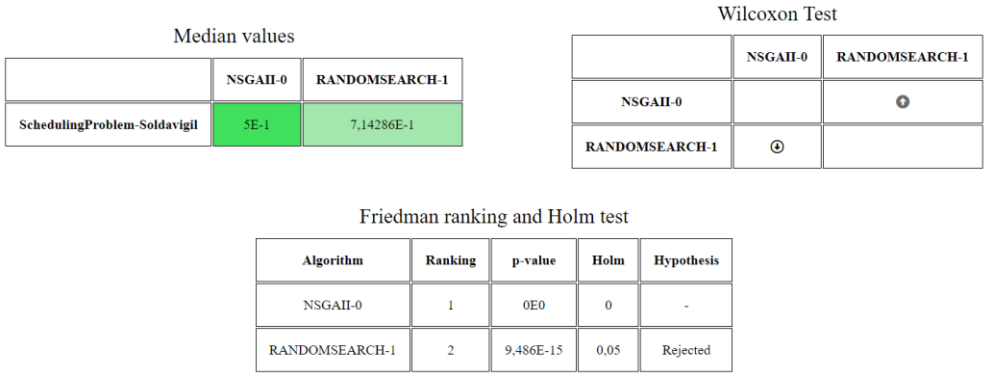
10.8.5. Experimento 3: Comparar el desempeño de los algoritmos NSGAII y RANDOMSEARCH con una misma configuración en el problema de Soldavigil.

Objetivo: determinar que algoritmo tiene mejor desempeño en la resolución del problema de Soldavigil.

A continuación, en la tabla 7 se muestra un resumen de las configuraciones utilizadas.

Algoritmo	Probabilidad de cruce	Probabilidad de mutación	Tamaño de la población	Número de evaluaciones
NSGAII	0.9	0.4	100	25000
Random Search	-	-	-	25000

Tabla 7. Resumen de configuración de parámetros para los algoritmos NSGAII y Random Search.



Median values

	NSGAII-0	RANDOMSEARCH-1
SchedulingProblem-Soldavigil	5E-1	7,14286E-1

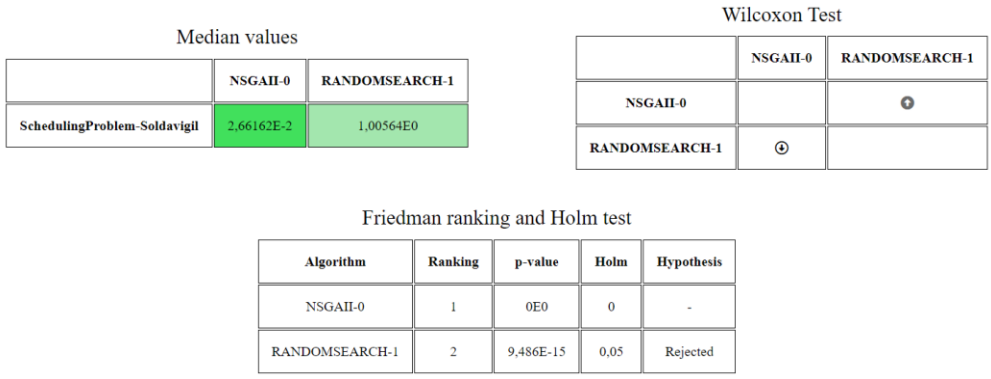
Wilcoxon Test

	NSGAII-0	RANDOMSEARCH-1
NSGAII-0		⊕
RANDOMSEARCH-1	⊖	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
RANDOMSEARCH-1	2	9,486E-15	0,05	Rejected

Figura 49. Valores medios, test de Wilcoxon, test de Friedman y Holm obtenido de la comparación de los algoritmos NSGAII y Random Search (EP)



Median values

	NSGAII-0	RANDOMSEARCH-1
SchedulingProblem-Soldavigil	2,66162E-2	1,00564E0

Wilcoxon Test

	NSGAII-0	RANDOMSEARCH-1
NSGAII-0		⊕
RANDOMSEARCH-1	⊖	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
RANDOMSEARCH-1	2	9,486E-15	0,05	Rejected

Figura 50. Valores medios, test de Wilcoxon, test de Friedman y Holm obtenido de la comparación de los algoritmos NSGAII y Random Search (GD)

Median values

	NSGAII-0	RANDOMSEARCH-1
SchedulingProblem-Soldavigil	7,14286E-2	0E0

Wilcoxon Test

	NSGAII-0	RANDOMSEARCH-1
NSGAII-0		⊕
RANDOMSEARCH-1	⊖	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
RANDOMSEARCH-1	2	9,486E-15	0,05	Rejected

Figura 51 Valores medios, test de Wilcoxon, test de Friedman y Holm obtenido de la comparación de los algoritmos NSGAII y Random Search (HV)

Median values

	NSGAII-0	RANDOMSEARCH-1
SchedulingProblem-Soldavigil	3,09524E-1	6,6453E-1

Wilcoxon Test

	NSGAII-0	RANDOMSEARCH-1
NSGAII-0		⊕
RANDOMSEARCH-1	⊖	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
RANDOMSEARCH-1	2	9,486E-15	0,05	Rejected

Figura 52. Valores medios, test de Wilcoxon, test de Friedman y Holm obtenido de la comparación de los algoritmos NSGAII y Random Search (IGD+)

Median values

	NSGAII-0	RANDOMSEARCH-1
SchedulingProblem-Soldavigil	1,55662E0	7,12591E-1

Wilcoxon Test

	NSGAII-0	RANDOMSEARCH-1
NSGAII-0		⊕
RANDOMSEARCH-1	⊖	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
RANDOMSEARCH-1	1	0E0	0	-
NSGAII-0	2	9,486E-15	0,05	Rejected

Figura 53. Valores medios, test de Wilcoxon, test de Friedman y Holm obtenido de la comparación de los algoritmos NSGAII y Random Search (SPREAD)

En las figuras 49 hasta 53 se pueden observar los resultados obtenidos en cada uno de los indicadores de calidad, donde se comprueba que el algoritmo NSGAII con la configuración resultante del experimento 1 es significativamente mejor que el algoritmo Random Search. En la tabla 8 se muestra un resumen de los mejores valores medianos encontrados en todos los indicadores.

Indicadores de calidad	Median values	NSGAII	Random Search
EP	5E-1	X	
GD	2.66162E-2	X	
HV	7.14286E-2	X	
IGD+	3.09524E-1	X	
SPREAD	7.12591E-1		X

Tabla 8. Mejores resultados obtenidos en los valores medianos de todos los indicadores de calidad.

En cuatro de los cinco indicadores de calidad el algoritmo NSGAII es significativamente mejor que el Random Search excepto en el indicador SPREAD. Por lo cual podemos llegar a la conclusión de que el algoritmo NSGAII con la configuración propuesta ofrece mejores resultados para la resolución del problema.

10.8.6. Experimento 4. Comparar el desempeño de los algoritmos NSGAII, NSGAIII, ESPEA, PESA2, PAES, MOCELL, SMSEMOA y RANDOMSEARCH en la resolución del problema de FireDrone con los conjuntos de datos de la EPI y el Bosque de Muniellos.

Objetivo: Comparar un conjunto de algoritmos para seleccionar el de mejor desempeño en la resolución del problema en los dos conjuntos de datos.

Se utiliza la configuración resultante del experimento 1 pero cabe destacar que, aunque esa configuración arrojara los mejores resultados en el problema de Soldavigil no tiene por qué ser igual en otro problema. La intención de este experimento es utilizar una configuración aleatoria con diferentes algoritmos para ver bajo las mismas condiciones cual obtiene los mejores resultados en dos conjuntos de datos con diferente tamaño. Luego de tener ese o esos algoritmos se podría realizar otra experimentación para encontrar los parámetros que mejoren el desempeño. A continuación, se muestra una tabla con las diferentes configuraciones utilizadas en la experimentación.

Algoritmo	Probabilidad de cruce	Probabilidad de mutación	Tamaño de la población	Número de evaluaciones
NSGAII	0.9	0.4	100	25000
NSGAII	0.9	0.4	100	25000
ESPEA	0.9	0.4	100	25000
PAES	0.9	0.4	100	25000
PESA2	0.9	0.4	100	25000
MOCELL	0.9	0.4	100	25000
SMSEMOA	0.9	0.4	100	25000
RANDOMSEARCH	0.9	0.4	100	25000

Tabla 9. Configuraciones utilizadas para varios algoritmos en la resolución del problema de FireDrone.

Al analizar los resultados obtenidos de la experimentación (Anexos figuras 76 a 90) se observa como el algoritmo NSGAII con la configuración propuesta aplicado al conjunto de datos de la EPI tiene un mejor desempeño que el resto de los algoritmos analizados en 4 de los 5 los indicadores de calidad utilizados.

Por otro lado, los resultados del experimento para el conjunto de datos del bosque de Muniellos (Anexos figuras 91 a 105) muestran que para este conjunto el algoritmo NSGAIII obtienen los mejores resultados en 3 de los 5 indicadores de calidad.

11. Conclusiones

11.1. Aportaciones

En esta investigación hemos abordado los problemas del cálculo de rutas de vuelo de drones en un sistema de supervisión de un área boscosa para la detección de posibles incendios y el de planificación de células de soldadura robotizada utilizando las ventajas que brinda el framework de optimización multiobjetivo JMetal. La resolución del primer problema sirvió de base para asentar los conocimientos de la investigación realizada sobre el framework para luego aplicarlos al segundo problema. Ambos problemas han sido modelados mediante permutaciones utilizando el algoritmo genético multiobjetivo NSGAII.

El objetivo de esta investigación era mejorar el proceso de planificación de células de soldadura robotizada, hasta ahora manual, por lo cual surge la necesidad de encontrar un conjunto de parámetros que arrojen los mejores resultados en la resolución del problema, para ello se realiza un estudio experimental con el objetivo de encontrar dichos parámetros y una vez se obtienen se realiza un segundo experimento comparando los resultados del NSGAII con el algoritmo Random Search para confirmar la calidad de convergencia hacia soluciones razonablemente mejores que las obtenidas den una búsqueda aleatoria.

Además, se implementan dos soluciones informáticas:

JMetal-Benchmarking: una herramienta encaminada a la investigación donde se incluyen los dos problemas tratados en esta investigación. Se puede dar solución a ambos problemas teniendo en cuenta diferentes algoritmos y configuraciones y por último comparar los resultados. También se brinda una funcionalidad de experimentación donde se pueden crear un conjunto de configuraciones para aplicar a los problemas.

Soldavigil: La otra es un primer prototipo para la empresa de Soldavigil donde se desarrollan las siguientes funcionalidades: gestión de órdenes, programación de órdenes de trabajo empleando algoritmo genético, visualización del diagrama de Gantt con la planificación arrojada por el mismo, resumen de las órdenes programadas por robot (las que están fuera de tiempo o no, los días programados en cada robot y los días con retraso en caso de existir). En ella se encuentra incrustado como algoritmo de planificación el método NSGA-II con la configuración resultante del plan de experimentación realizado con la primera de las aplicaciones (JMetal-Benchmarking).

Esta última herramienta servirá como primer prototipo de planificación automática, pero más importante aún como pieza clave en la comunicación con los expertos de la empresa y facilitar la elicitación de las restricciones reales del proceso de planificación para poder concluir con el desarrollo de una aplicación que pueda ser explotada como asistente a la planificación de las órdenes de soldadura.

Como resultado del trabajo desarrollado con el framework JMetal, ha sido necesario no solo hacer un uso correcto de su código sino también de su corrección, al ser este un proyecto vivo y de código abierto. se ha agregado una corrección al framework en la representación del test de Wilcoxon al escribir en los archivos R y html ya que nos mostraba una matriz simétrica, eliminando el último elemento de las filas y el primer elemento de las columnas lo cual hacía que no coincidieran los resultados obtenidos cuando se

analizaban desde las filas y las columnas. Por otro lado se implementan algunas clases inexistentes en el framework para las soluciones basadas en permutaciones como lo es la implementación de una clase para la creación de un frente Pareto de referencia, la funcionalidad de crear la población inicial a partir de una solución ya arrojada por alguno de los algoritmos utilizados y dadas un conjunto de configuraciones para realizar una experimentación realizar un todos contra todos para buscar los parámetros con los que los algoritmos obtenga los mejores resultados.

11.2. Trabajo futuro

Teniendo en cuenta el alcance de lo desarrollado en esta investigación se plantea dentro del ámbito de las metaheurísticas como trabajo futuro:

- Disponer de una base de instancias más amplia con históricos de pedidos de la empresa.
- Incluir nuevos operadores de cruce y mutación para representaciones mediante permutaciones ya que el framework solo ofrece uno de cada tipo. Para ello se deberá realizar un estudio del problema
- Incorporar la funcionalidad del JMetal de autoconfiguración de parámetros utilizando Irace. Este software permite la parametrización automática del algoritmo evolutivo mediante un mecanismo por decirlo así evolutivo de los valores de entrada de los parámetros de entrada.
- Implementar un algoritmo de búsqueda local al algoritmo evolutivo logrando así un algoritmo memético lo que permita obtener soluciones aún más eficientes en situaciones complejas de difícil resolución con solo un algoritmo evolutivo. Incorporando:
    - La posibilidad de planificar de manera concurrente más de una orden en cada robot en el caso de órdenes compatibles entre sí, es decir que puedan disponerse ambas piezas en las mesas de un mismo robot.
- Interrumpir las ordenes actualmente en producción, para intercalar otras órdenes, finalizando más tarde (en el mismo u otro robot) la orden interrumpida, si con ello se consigue servir todas las ordenes dentro de los plazos de entrega.

12. Bibliografía

Attarmoghaddam, N., Li, K. F., & Kanan, A. (2019). FPGA implementation of crossover module of genetic algorithm. *Information (Switzerland)*, *10*(6), 1–11. https://doi.org/10.3390/info10060184

Avellar, G. S. C., Pereira, G. A. S., Pimenta, L. C. A., & Iscold, P. (2015). Multi-UAV routing for area coverage and remote sensing with minimum time. *Sensors (Switzerland)*, *15*(11), 27783–27803. https://doi.org/10.3390/s151127783

Barba-Gonzaléz, C., García-Nieto, J., Nebro, A. J., & Aldana-Montes, J. F. (2017). *Multi-Objective Big Data Optimization with jMetal and Spark*. https://github.com/jMetal/jMetalSP

Beume, N., Naujoks, B., & Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, *181*(3), 1653–1669. https://doi.org/10.1016/j.ejor.2006.08.008

Blum, C., & Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison Development of a new metaheuristic algorithm for combinatorial optimization based on instance reduction View project SWARMANOID View project Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison Metaheuristics in Combinatorial Optimization. *ACM Computing Surveys*, *35*(3), 268–308. https://doi.org/10.1145/937503.937505

Braun, M., Shukla, P. K., & Schmeck, H. (2015). Obtaining optimal Pareto front approximations using scalarized preference information. *GECCO 2015 - Proceedings of the 2015 Genetic and Evolutionary Computation Conference*, 631–638. https://doi.org/10.1145/2739480.2754674

Braun, M., Shukla, P., & Schmeck, H. (2017). Angle-based preference models in multi-objective optimization. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10173 LNCS*, 88–102. https://doi.org/10.1007/978-3-319-54157-0_7

Cai, X., Sun, H., & Fan, Z. (2018). A diversity indicator based on reference vectors for many-objective optimization. *Information Sciences*, *430–431*, 467–486. https://doi.org/10.1016/j.ins.2017.11.051

Coello, C. A. C., Lamont, G. B., & Van Veldhuizen, D. A. (2007). *Second Edition Evolutionary Algorithms for Solving Multi-Objective Problems*. http://www.springer.com

Corne, D., Jerram, N., Knowles, J., Oates, M., & Martin, J. (2001). PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, *January*, 283–290. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.2194

Corne, D. W., Knowles, J. D., & Oates, M. J. (2000). The Pareto envelope-based selection algorithm for multiobjective optimization. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1917*(Mcdm), 839–848. https://doi.org/10.1007/3-540-45356-3_82

Deb, K. (2011). *Multi-Objective Optimization Using Evolutionary Algorithms: An Introduction*. http://www.iitk.ac.in/kangal/deb.htm

Deb, K., & Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. *IEEE*

Transactions on Evolutionary Computation, *18*(4), 577–601. https://doi.org/10.1109/TEVC.2013.2281535

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: {NSGA}-{II}. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017

Demˇ, J. (2006). *Statistical Comparisons of Classifiers over Multiple Data Sets*. *7*, 1–30.

Drugan, M. M., & Nowe, A. (2014). Scalarization based Pareto optimal set of arms identification algorithms. *Proceedings of the International Joint Conference on Neural Networks*, *2010*, 2690–2697. https://doi.org/10.1109/IJCNN.2014.6889484

Durillo, J. J., & Nebro, A. J. (2011). JMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, *42*(10), 760–771. https://doi.org/10.1016/j.advengsoft.2011.05.014

Durillo, J. J., Nebro, A. J., Luna, F., & Alba, E. (2006). jMetal : a Java Framework for Developing Multi-objective. *Science And Technology*, *01*(January), 1–12. http://www.iitk.ac.in/kangal/codes.shtml

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, *6*(2), 65–70. http://www.jstor.org/stable/4615733

Jean-Yvesspotvin, M. (2010). *Handbook of Metaheuristics Third Edition*. http://www.springer.com/series/6161

Knowles, J., & Corne, D. (1999). The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, *1*(December 2013), 98–105. https://doi.org/10.1109/CEC.1999.781913

Knowles, J. D. ;, Thiele, L. ;, & Zitzler. (2006). *A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers*. https://doi.org/10.3929/ethz-b-000023822

Knowles, J. D., & Corne, D. W. (2000). *Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy*. https://doi.org/10.1162/106365600568167

Kronfeld, M. (2011). *EvA2 Short Documentation*. http://www.ra.cs.uni-tuebingen.de/software/EvA2

Kursawe, F. (1990). A Variant of Evolution Strategies for Vector Optimization 1 Introduction 2 Shortcomings of Conventional Methods 3 Evolution Strategies ... *In International Conference on Parallel Problem Solving from Nature*, 193–197.

Lewis, B., Smith, I., Fowler, M., & Licato, J. (2017). The robot mafia: A test environment for deceptive robots. *28th Modern Artificial Intelligence and Cognitive Science Conference, MAICS 2017*, 189–190. https://doi.org/10.1145/1235

Lukasiewycz, M., Glaß, M., Reimann, F., & Teich, J. (2011). Opt4J - A modular framework for meta-heuristic optimization. *Genetic and Evolutionary Computation Conference, GECCO'11*, 1723–1730. https://doi.org/10.1145/2001576.2001808

Martin, L., Leblanc, R., & Toan, N. K. (1993). *Tables for the Friedman rank test*. *21*(1), 39–44.

Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., & Alba, E. (2009). MOCell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*, *24*(7), 726–746.

https://doi.org/10.1002/int.20358

Nebro, A. J., López-Ibáñez, M., Barba-González, C., & García-Nieto, J. (2019). *Automatic Configuration of NSGA-II with jMetal and irace*. https://doi.org/10.1145/3319619.3326832

Reyes-Sierra, M., & Coello Coello, C. A. (2006). Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. In *International Journal of Computational Intelligence Research* (Vol. 2, Issue 3). http://delta.cs.cinvestav.mx/

Sağ, T., & Çunkaş, M. (2009). A tool for multiobjective evolutionary algorithms. *Advances in Engineering Software*, *40*(9), 902–912. https://doi.org/10.1016/j.advengsoft.2009.01.001

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *The 1st International Conference on Genetic Algorithms*, *JANUARY 1985*, 93–100. http://dl.acm.org/citation.cfm?id=657079

Srinivas, N., & Deb, K. (1994). Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, *2*(3), 221–248. https://doi.org/10.1162/evco.1994.2.3.221

Tanaka, M., Watanabe, H., Furukawa, Y., & Tanino, T. (1995). *GA-based decision support system for multicriteria optimization*. http://escholarship.lib.okayama-u.ac.jp/industrialengineering/57

Teghem, J. (2010). Book Review. *European Journal of Operational Research*, *205*(2), 486–487. https://doi.org/10.1016/j.ejor.2009.12.010

Usamentiaga, R., Suarez, F. J., Tuya, P. J., Suárez-cabal, M. J., & Corcoba, V. (n.d.). *nes : un caso de aprendizaje basado en proyectos en el marco de un proyecto coordinado en un Máster Universitario en Ingeniería Informática Resumen*. *2018*, 1–12.

Veldhuizen, D. A. Van, & Lamont, G. B. (1998). Multiobjective Evolutionary Algorithm Research : A History and Analysis. *Technical Report TR-98-03, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio*, *1998*, 1–88.

Walker, J. D. (2010). *Fundamentals of physics extended. Wiley.* https://www.researchgate.net/publication/254582467_Fundamentals_of_Physics_Extended_9th_ed

Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., & Tsang, E. (2006). *Combining Model-based and Genetics-based Offspring Generation for Multi-objective Optimization Using a Convergence Criterion*.

Zitzler, E. ;, Laumanns, M. ;, Thiele, L., Zitzler, E., & Laumanns, M. (2001). *SPEA2: Improving the strength pareto evolutionary algorithm*. https://doi.org/10.3929/ethz-a-004284029

Zitzler, E., & Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. In *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION* (Vol. 3, Issue 4).

13. Anexos

Para lograr una mayor legibilidad de la investigación en el apartado del Plan de Experimentación solo se explican los resultados de uno de los indicadores de calidad por lo cual en este apartado de Anexos contamos con el resto de figuras utilizadas en la experimentación para cada problema y cada indicador de calidad.

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	4,08344E-2	4,1682E-2	4,20216E-2	3,96592E-2

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	=
NSGAII-1	=		=	=
NSGAII-2	=	=		=
NSGAII-3	=	=	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-3	1	0E0	0	-
NSGAII-0	2	1,973E-9	0,05	Rejected
NSGAII-1	3	3,553E-33	0,025	Rejected
NSGAII-2	4	1,948E-72	0,017	Rejected

Figura 54. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis de la probabilidad de cruce (GD)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	0E0	0E0	0E0	0E0

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	▲	▲
NSGAII-1	=		▲	▲
NSGAII-2	=	=		▲
NSGAII-3	=	=	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
NSGAII-1	2	1,973E-9	0,05	Rejected
NSGAII-2	3	3,553E-33	0,025	Rejected
NSGAII-3	4	1,948E-72	0,017	Rejected

Figura 55. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis de la probabilidad de cruce (HV)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	4,11431E-1	4,11431E-1	4,52381E-1	4,52381E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	=
NSGAII-1	=		=	=
NSGAII-2	=	=		=
NSGAII-3	=	=	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
NSGAII-1	2	1,973E-9	0,05	Rejected
NSGAII-3	3	3,553E-33	0,025	Rejected
NSGAII-2	4	1,948E-72	0,017	Rejected

Figura 56. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis de la probabilidad de cruce (IGD+)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	1,51181E0	1,47537E0	1,42466E0	1,38363E0

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	⊕
NSGAII-1	=		=	=
NSGAII-2	=	=		=
NSGAII-3	⊖	=	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-2	1	0E0	0	-
NSGAII-3	2	1,973E-9	0,05	Rejected
NSGAII-0	3	3,553E-33	0,025	Rejected
NSGAII-1	4	1,948E-72	0,017	Rejected

Figura 57. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis de la probabilidad de cruce (SPREAD)

Algorithm	Crossover Probablity	Mutation Probability	Population Size	Evaluation
NSGAII	0.9	0.1	100	10000
NSGAII	0.9	0.2	100	10000
NSGAII	0.9	0.3	100	10000
NSGAII	0.9	0.4	100	10000

Figura 58. Configuraciones utilizadas para probar que probabilidad de mutación obtienen los mejores resultados con el algoritmo NSGAII.

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	6,66667E-1	6,66667E-1	6,66667E-1	6,66667E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		▲	=	=
NSGAII-1	▲		=	=
NSGAII-2	▲	▲		=
NSGAII-3	▲	▲	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-2	1	0E0	0	-
NSGAII-3	2	1,973E-9	0,05	Rejected
NSGAII-1	3	3,553E-33	0,025	Rejected
NSGAII-0	4	1,948E-72	0,017	Rejected

Figura 59. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis de la probabilidad de mutación (EP)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	1,574E-2	1,59187E-2	1,45768E-2	1,3019E-2

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	=
NSGAII-1	=		=	=
NSGAII-2	=	=		=
NSGAII-3	=	=	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-3	1	0E0	0	-
NSGAII-1	2	1,973E-9	0,05	Rejected
NSGAII-2	3	3,553E-33	0,025	Rejected
NSGAII-0	4	1,948E-72	0,017	Rejected

Figura 60. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis de la probabilidad de mutación (GD)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	2,33333E-1	2,33333E-1	2,5E-1	2,41667E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	=
NSGAII-1	=		=	=
NSGAII-2	=	=		=
NSGAII-3	▲	=	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-2	1	0E0	0	-
NSGAII-3	2	1,973E-9	0,05	Rejected
NSGAII-0	3	3,553E-33	0,025	Rejected
NSGAII-1	4	1,948E-72	0,017	Rejected

Figura 61. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis de la probabilidad de mutación (HV)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	3,1875E-1	3E-1	3E-1	2,875E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	▼
NSGAII-1	=		=	=
NSGAII-2	=	=		=
NSGAII-3	▲	=	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-2	1	0E0	0	-
NSGAII-3	2	1,973E-9	0,05	Rejected
NSGAII-0	3	3,553E-33	0,025	Rejected
NSGAII-1	4	1,948E-72	0,017	Rejected

Figura 62. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis de la probabilidad de mutación (IGD+)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	1,28188E0	1,28747E0	1,32081E0	1,292E0

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	⊕	⊕
NSGAII-1	=		=	=
NSGAII-2	⊖	=		=
NSGAII-3	⊖	=	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
NSGAII-1	2	1,973E-9	0,05	Rejected
NSGAII-3	3	3,553E-33	0,025	Rejected
NSGAII-2	4	1,948E-72	0,017	Rejected

Figura 63. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis de la probabilidad de mutación (SPREAD)

Algorithm	Crossover Probablity	Mutation Probability	Population Size	Evaluation
NSGAII	0.9	0.4	100	10000
NSGAII	0.9	0.4	200	10000
NSGAII	0.9	0.4	300	10000
NSGAII	0.9	0.4	400	10000

Figura 64. Configuraciones utilizadas para probar que tamaño de población obtienen los mejores resultados con el algoritmo NSGAII.

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	5E-1	6,66667E-1	6,66667E-1	6,66667E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		⊕	=	⊕
NSGAII-1	=		=	=
NSGAII-2	=	=		=
NSGAII-3	=	=	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
NSGAII-2	2	1,973E-9	0,05	Rejected
NSGAII-1	3	3,553E-33	0,025	Rejected
NSGAII-3	4	1,948E-72	0,017	Rejected

Figura 65. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis del tamaño de la población (EP)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	1,00272E-2	5,93241E-3	1,22095E-2	4,85437E-2

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	⬆	⬆
NSGAII-1	=		⬆	⬆
NSGAII-2	⬇	⬇		⬆
NSGAII-3	⬇	⬇	⬇	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-1	1	0E0	0	-
NSGAII-0	2	1,973E-9	0,05	Rejected
NSGAII-2	3	3,553E-33	0,025	Rejected
NSGAII-3	4	1,948E-72	0,017	Rejected

Figura 66. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis del tamaño de la población (GD)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	2,45614E-1	2,2807E-1	2,2807E-1	2,10526E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		⬆	⬆	⬆
NSGAII-1	⬇		=	⬆
NSGAII-2	=	=		⬆
NSGAII-3	⬇	⬇	⬇	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
NSGAII-1	2	1,973E-9	0,05	Rejected
NSGAII-2	3	3,553E-33	0,025	Rejected
NSGAII-3	4	1,948E-72	0,017	Rejected

Figura 67. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis del tamaño de la población (HV)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	2,52193E-1	2,82895E-1	2,76316E-1	3,04825E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	⊕
NSGAII-1	=		=	⊕
NSGAII-2	=	=		⊕
NSGAII-3	⊖	⊖	⊖	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
NSGAII-2	2	1,973E-9	0,05	Rejected
NSGAII-1	3	3,553E-33	0,025	Rejected
NSGAII-3	4	1,948E-72	0,017	Rejected

Figura 68. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis del tamaño de la población (IGD+)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	1,40026E0	1,33405E0	1,31808E0	1,25309E0

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	⊖
NSGAII-1	=		=	⊖
NSGAII-2	=	=		⊖
NSGAII-3	⊕	⊕	⊕	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-3	1	0E0	0	-
NSGAII-2	2	1,973E-9	0,05	Rejected
NSGAII-1	3	3,553E-33	0,025	Rejected
NSGAII-0	4	1,948E-72	0,017	Rejected

Figura 69. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis del tamaño de la población (SPREAD)

Algorithm	Crossover Probablity	Mutation Probability	Population Size	Evaluation
NSGAII	0.9	0.4	100	10000
NSGAII	0.9	0.4	100	15000
NSGAII	0.9	0.4	100	20000
NSGAII	0.9	0.4	100	25000

Figura 70. Configuraciones utilizadas para probar que número de evaluaciones obtienen los mejores resultados con el algoritmo NSGAII.

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	6,66667E-1	5E-1	6,66667E-1	3,33333E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	=
NSGAII-1	=		▲	=
NSGAII-2	=	=		=
NSGAII-3	=	=	▲	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-3	1	0E0	0	-
NSGAII-1	2	1,973E-9	0,05	Rejected
NSGAII-0	3	3,553E-33	0,025	Rejected
NSGAII-2	4	1,948E-72	0,017	Rejected

Figura 71. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis del número de evaluaciones (EP)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	1,47796E-2	2,04199E-2	1,45774E-2	1,53825E-2

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	=
NSGAII-1	=		▼	▼
NSGAII-2	=	▲		=
NSGAII-3	=	▲	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-2	1	0E0	0	-
NSGAII-3	2	1,973E-9	0,05	Rejected
NSGAII-0	3	3,553E-33	0,025	Rejected
NSGAII-1	4	1,948E-72	0,017	Rejected

Figura 72. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis del número de evaluaciones (GD)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	2,33333E-1	2,5E-1	2,41667E-1	3,16667E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	⊕
NSGAII-1	=		=	⊕
NSGAII-2	=	=		⊕
NSGAII-3	⊖	⊖	⊖	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-3	1	0E0	0	-
NSGAII-1	2	1,973E-9	0,05	Rejected
NSGAII-2	3	3,553E-33	0,025	Rejected
NSGAII-0	4	1,948E-72	0,017	Rejected

Figura 73. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis del número de evaluaciones (HV)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	3,125E-1	3E-1	3,125E-1	2,52083E-1

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	=
NSGAII-1	=		=	=
NSGAII-2	=	=		⊕
NSGAII-3	=	=	⊖	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-3	1	0E0	0	-
NSGAII-1	2	1,973E-9	0,05	Rejected
NSGAII-2	3	3,553E-33	0,025	Rejected
NSGAII-0	4	1,948E-72	0,017	Rejected

Figura 74. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis del número de evaluaciones (IGD+)

Median values

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
SchedulingProblem-Soldavigil	1,30737E0	1,40724E0	1,28482E0	1,44606E0

Wilcoxon Test

	NSGAII-0	NSGAII-1	NSGAII-2	NSGAII-3
NSGAII-0		=	=	=
NSGAII-1	=		⊖	=
NSGAII-2	=	⊕		=
NSGAII-3	=	=	=	

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-2	1	0E0	0	-
NSGAII-3	2	1,973E-9	0,05	Rejected
NSGAII-0	3	3,553E-33	0,025	Rejected
NSGAII-1	4	1,948E-72	0,017	Rejected

Figura 75. Valores medios, test de Wilcoxon y test de Friedman y Holm del análisis del número de evaluaciones (SPREAD)

Median values

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
FireDroneProblem-FireDrone	4E-1	3E-1	5E-1	4E-1	6E-1	4E-1	6E-1	6E-1

Figura 76. Media values del indicador EP en el problema FireDrone con el conjunto de datos de la EPI.

Wilcoxon Test

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
NSGAII-0		=	⊕	=	⊕	⊕	⊕	⊕
NSGAIII-1	=		⊕	⊕	⊕	⊕	⊕	⊕
ESPEA-2	⊖	⊖		=	⊕	=	=	=
MOCell-3	=	=	⊕		⊕	=	⊕	⊕
PAES-4	⊖	⊖	⊖	⊖		⊖	⊕	⊕
PESA2-5	=	=	⊕	=	⊕		⊕	⊕
SMSEMOA-6	⊖	⊖	=	=	⊕	=		=
RANDOMSEARCH-7	⊖	⊖	=	⊖	⊕	⊖	=	

Figura 77. Test de Wilcoxon del indicador EP en el problema FireDrone con el conjunto de datos de la EPI.

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
NSGAIII-1	2	7,744E-6	0,05	Rejected
MOCell-3	3	3,744E-19	0,025	Rejected
PESA2-5	4	4,846E-41	0,017	Rejected
ESPEA-2	5	1,448E-71	0,013	Rejected
SMSEMOA-6	6	9,505E-111	0,01	Rejected
RANDOMSEARCH-7	7	1,339E-158	0,008	Rejected
PAES-4	8	3,996E-215	0,007	Rejected

Figura 78. Test de Friedman y Holm del indicador EP en el problema FireDrone con el conjunto de datos de la EPI.

Median values

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
FireDroneProblem-FireDrone	0E0	0E0	0E0	0E0	2,03853E-3	0E0	4,09305E-4	3,0047E-2

Figura 79. Media values del indicador GD en el problema FireDrone con el conjunto de datos de la EPI.

Wilcoxon Test

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
NSGAII-0		⬆	⬆	⬆	⬆	⬆	⬆	⬆
NSGAIII-1	⬆		⬆	⬆	⬆	⬆	⬆	⬆
ESPEA-2	=	=		⬆	⬆	⬆	⬆	⬆
MOCell-3	⬆	⬆	⬆		⬆	⬆	⬆	⬆
PAES-4	⬇	⬇	=	⬇		⬇	⬇	⬆
PESA2-5	⬆	⬆	⬆	⬆	⬆		⬆	⬆
SMSEMOA-6	⬇	⬇	=	=	⬆	=		⬆
RANDOMSEARCH-7	⬇	⬇	⬇	⬇	⬇	⬇	⬇	

Figura 80. Test de Wilcoxon del indicador GD en el problema FireDrone con el conjunto de datos de la EPI.

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
NSGAIII-1	2	7,744E-6	0,05	Rejected
SMSEMOA-6	3	3,744E-19	0,025	Rejected
PESA2-5	4	4,846E-41	0,017	Rejected
ESPEA-2	5	1,448E-71	0,013	Rejected
MOCell-3	6	9,505E-111	0,01	Rejected
PAES-4	7	1,339E-158	0,008	Rejected
RANDOMSEARCH-7	8	3,996E-215	0,007	Rejected

Figura 81. Test de Friedman y Holm del indicador GD en el problema FireDrone con el conjunto de datos de la EPI.

Median values

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
FireDroneProblem-FireDrone	5,09926E-1	5,49574E-1	4,35017E-1	5,09926E-1	3,00755E-1	5,09926E-1	3,60109E-1	3,56016E-1

Figura 82. Media values del indicador HV en el problema FireDrone con el conjunto de datos de la EPI.

Wilcoxon Test

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
NSGAII-0		=	⬆	=	⬆	⬆	⬆	⬆
NSGAIII-1	=		⬆	⬆	⬆	⬆	⬆	⬆
ESPEA-2	⬇	⬇		=	⬆	=	=	=
MOCell-3	=	=	⬆		⬆	=	⬆	⬆
PAES-4	⬇	⬇	⬇	⬇		⬇	⬇	⬇
PESA2-5	=	=	⬆	=	⬆		⬆	⬆
SMSEMOA-6	⬇	⬇	=	⬇	⬆	⬇		=
RANDOMSEARCH-7	⬇	⬇	=	⬇	⬆	⬇	=	

Figura 83. Test de Wilcoxon del indicador HV en el problema FireDrone con el conjunto de datos de la EPI.

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
NSGAIII-1	2	7,744E-6	0,05	Rejected
MOCell-3	3	3,744E-19	0,025	Rejected
PESA2-5	4	4,846E-41	0,017	Rejected
ESPEA-2	5	1,448E-71	0,013	Rejected
SMSEMOA-6	6	9,505E-111	0,01	Rejected
RANDOMSEARCH-7	7	1,339E-158	0,008	Rejected
PAES-4	8	3,996E-215	0,007	Rejected

Figura 84. Test de Friedman y Holm del indicador HV en el problema FireDrone con el conjunto de datos de la EPI.

Median values

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
FireDroneProblem-FireDrone	1E-1	7,27446E-2	1,5E-1	1E-1	2,5055E-1	1E-1	2E-1	2,03411E-1

Figura 85. Media values del indicador IGD+ en el problema FireDrone con el conjunto de datos de la EPI.

Wilcoxon Test

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
NSGAII-0		=	⊕	=	⊕	⊕	⊕	⊕
NSGAIII-1	=		⊕	⊕	⊕	⊕	⊕	⊕
ESPEA-2	⊖	⊖		=	⊕	=	=	=
MOCell-3	=	=	⊕		⊕	=	⊕	⊕
PAES-4	⊖	⊖	⊖	⊖		⊖	⊖	⊖
PESA2-5	=	=	⊕	=	⊕		⊕	⊕
SMSEMOA-6	⊖	⊖	=	⊖	⊕	⊖		=
RANDOMSEARCH-7	⊖	⊖	=	⊖	⊕	⊖	=	

Figura 86. Test de Wilcoxon del indicador IGD+ en el problema FireDrone con el conjunto de datos de la EPI.

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAII-0	1	0E0	0	-
NSGAIII-1	2	7,744E-6	0,05	Rejected
MOCell-3	3	3,744E-19	0,025	Rejected
PESA2-5	4	4,846E-41	0,017	Rejected
ESPEA-2	5	1,448E-71	0,013	Rejected
SMSEMOA-6	6	9,505E-111	0,01	Rejected
RANDOMSEARCH-7	7	1,339E-158	0,008	Rejected
PAES-4	8	3,996E-215	0,007	Rejected

Figura 87. Test de Friedman y Holm del indicador IGD+ en el problema FireDrone con el conjunto de datos de la EPI.

Median values

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
FireDroneProblem-FireDrone	1,41422E0	1,27583E0	6,6471E-1	6,03109E-1	1,34461E0	1,41422E0	1,3179E0	7,16243E-1

Figura 88. Media values del indicador SPREAD en el problema FireDrone con el conjunto de datos de la EPI.

Wilcoxon Test

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
NSGAII-0		⊕	⊕	⊕	⊕	=	⊕	⊕
NSGAIII-1	⊖		⊕	⊕	=	⊖	⊖	⊕
ESPEA-2	⊖	⊖		=	⊖	⊖	⊖	=
MOCell-3	⊖	⊖	⊖		⊖	⊖	⊖	⊖
PAES-4	⊖	=	⊕	⊕		⊖	⊕	⊕
PESA2-5	=	⊕	⊕	⊕	⊕		=	⊕
SMSEMOA-6	⊖	⊕	⊕	⊕	⊖	=		⊕
RANDOMSEARCH-7	⊖	⊖	=	⊕	⊖	⊖	⊖	

Figura 89. Test de Wilcoxon del indicador SPREAD en el problema FireDrone con el conjunto de datos de la EPI.

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
MOCell-3	1	0E0	0	-
ESPEA-2	2	7,744E-6	0,05	Rejected
RANDOMSEARCH-7	3	3,744E-19	0,025	Rejected
NSGAIII-1	4	4,846E-41	0,017	Rejected
PAES-4	5	1,448E-71	0,013	Rejected
SMSEMOA-6	6	9,505E-111	0,01	Rejected
PESA2-5	7	1,339E-158	0,008	Rejected
NSGAII-0	8	3,996E-215	0,007	Rejected

Figura 90. Test de Friedman y Holm del indicador SPREAD en el problema FireDrone con el conjunto de datos de la EPI.

Median values

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
FireDroneProblem-FireDrone	2E-1	1,12616E-1	1,42857E-1	1,71429E-1	2,85714E-1	1,71429E-1	2,28571E-1	3,71429E-1

Figura 91. Media values del indicador EP en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Wilcoxon Test

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
NSGAII-0		⊖	⊖	=	⊕	=	⊕	⊕
NSGAIII-1	⊕		⊕	⊕	⊕	⊕	⊕	⊕
ESPEA-2	⊕	⊖		⊕	⊕	⊕	⊕	⊕
MOCell-3	=	⊖	⊖		⊕	=	⊕	⊕
PAES-4	⊖	⊖	⊖	⊖		⊖	⊖	⊕
PESA2-5	=	⊖	=	=	⊕		⊕	⊕
SMSEMOA-6	⊖	⊖	⊖	⊖	⊕	⊖		⊕
RANDOMSEARCH-7	⊖	⊖	⊖	⊖	⊖	⊖	⊖	

Figura 92. Test de Wilcoxon del indicador EP en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAIII-1	1	0E0	0	-
ESPEA-2	2	7,744E-6	0,05	Rejected
MOCell-3	3	3,744E-19	0,025	Rejected
PESA2-5	4	4,846E-41	0,017	Rejected
NSGAII-0	5	1,448E-71	0,013	Rejected
SMSEMOA-6	6	9,505E-111	0,01	Rejected
PAES-4	7	1,339E-158	0,008	Rejected
RANDOMSEARCH-7	8	3,996E-215	0,007	Rejected

Figura 93. Test de Friedman y Holm del indicador EP en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Median values

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
FireDroneProblem-FireDrone	5,17638E-3	9,58995E-3	1,16753E-2	1,43009E-2	1,64722E-3	5,74655E-3	3,93082E-3	3,71116E-2

Figura 94. Media values del indicador GD en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Wilcoxon Test

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
NSGAII-0		⬆	⬆	⬆	⬇	=	⬇	⬆
NSGAIII-1	⬇		⬆	=	⬇	⬇	⬇	⬆
ESPEA-2	⬇	⬇		=	⬇	⬇	⬇	⬆
MOCell-3	⬇	=	=		⬇	⬇	⬇	⬆
PAES-4	⬆	⬆	⬆	⬆		⬆	⬆	⬆
PESA2-5	=	⬆	⬆	⬆	⬇		⬇	⬆
SMSEMOA-6	⬆	⬆	⬆	⬆	⬇	⬆		⬆
RANDOMSEARCH-7	⬇	⬇	⬇	⬇	⬇	⬇	⬇	

Figura 95. Test de Wilcoxon del indicador GD en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
PAES-4	1	0E0	0	-
SMSEMOA-6	2	7,744E-6	0,05	Rejected
NSGAII-0	3	3,744E-19	0,025	Rejected
PESA2-5	4	4,846E-41	0,017	Rejected
NSGAIII-1	5	1,448E-71	0,013	Rejected
MOCell-3	6	9,505E-111	0,01	Rejected
ESPEA-2	7	1,339E-158	0,008	Rejected
RANDOMSEARCH-7	8	3,996E-215	0,007	Rejected

Figura 96. Test de Friedman y Holm del indicador GD en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Median values

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
FireDroneProblem-FireDrone	6,5092E-1	7,18309E-1	6,76351E-1	6,55022E-1	5,67022E-1	6,55606E-1	6,30254E-1	4,7901E-1

Figura 97. Media values del indicador HV en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Wilcoxon Test

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
NSGAII-0		⊖	⊖	=	⊕	=	⊕	⊕
NSGAIII-1	⊕		⊕	⊕	⊕	⊕	⊕	⊕
ESPEA-2	⊕	⊖		⊕	⊕	⊕	⊕	⊕
MOCell-3	=	⊖	⊖		⊕	=	⊕	⊕
PAES-4	⊖	⊖	⊖	⊖		⊖	⊖	⊕
PESA2-5	=	⊖	⊖	=	⊕		⊕	⊕
SMSEMOA-6	⊖	⊖	⊖	⊖	⊕	⊖		⊕
RANDOMSEARCH-7	⊖	⊖	⊖	⊖	⊖	⊖	⊖	

Figura 98. Test de Wilcoxon del indicador HV en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAIII-1	1	0E0	0	-
ESPEA-2	2	7,744E-6	0,05	Rejected
MOCell-3	3	3,744E-19	0,025	Rejected
PESA2-5	4	4,846E-41	0,017	Rejected
NSGAII-0	5	1,448E-71	0,013	Rejected
SMSEMOA-6	6	9,505E-111	0,01	Rejected
PAES-4	7	1,339E-158	0,008	Rejected
RANDOMSEARCH-7	8	3,996E-215	0,007	Rejected

Figura 99. Test de Friedman y Holm del indicador HV en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Median values

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
FireDroneProblem-FireDrone	4,86018E-2	2,62701E-2	3,89795E-2	4,86802E-2	8,90701E-2	4,9393E-2	5,84134E-2	1,49788E-1

Figura 100. Media values del indicador IGD+ en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Wilcoxon Test

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
NSGAII-0		⊕	⊕	=	⊖	=	⊖	⊖
NSGAIII-1	⊖		⊖	⊖	⊖	⊖	⊖	⊖
ESPEA-2	⊖	⊕		⊖	⊖	⊖	⊖	⊖
MOCell-3	=	⊕	⊕		⊖	=	⊖	⊖
PAES-4	⊕	⊕	⊕	⊕		⊕	⊕	⊖
PESA2-5	=	⊕	⊕	=	⊖		⊖	⊖
SMSEMOA-6	⊕	⊕	⊕	⊕	⊖	⊕		⊖
RANDOMSEARCH-7	⊕	⊕	⊕	⊕	⊕	⊕	⊕	

Figura 101. Test de Wilcoxon del indicador IGD+ en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
NSGAIII-1	1	0E0	0	-
ESPEA-2	2	7,744E-6	0,05	Rejected
MOCell-3	3	3,744E-19	0,025	Rejected
NSGAII-0	4	4,846E-41	0,017	Rejected
PESA2-5	5	1,448E-71	0,013	Rejected
SMSEMOA-6	6	9,505E-111	0,01	Rejected
PAES-4	7	1,339E-158	0,008	Rejected
RANDOMSEARCH-7	8	3,996E-215	0,007	Rejected

Figura 102. Test de Friedman y Holm del indicador IGD+ en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Median values

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
FireDroneProblem-FireDrone	1,38015E0	9,371E-1	8,47144E-1	8,47159E-1	1,19648E0	1,36621E0	1,40225E0	8,09842E-1

Figura 103. Media values del indicador SPREAD en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Wilcoxon Test

	NSGAII-0	NSGAIII-1	ESPEA-2	MOCell-3	PAES-4	PESA2-5	SMSEMOA-6	RANDOMSEARCH-7
NSGAII-0		⊕	⊕	⊕	⊕	=	=	⊕
NSGAIII-1	⊖		⊕	⊕	⊖	⊖	⊖	⊕
ESPEA-2	⊖	⊖		=	⊖	⊖	⊖	⊕
MOCell-3	⊖	⊖	=		⊖	⊖	⊖	⊕
PAES-4	⊖	⊕	⊕	⊕		⊖	⊖	⊕
PESA2-5	=	⊕	⊕	⊕	⊕		=	⊕
SMSEMOA-6	=	⊕	⊕	⊕	⊕	=		⊕
RANDOMSEARCH-7	⊖	⊖	⊖	⊖	⊖	⊖	⊖	

Figura 104. Test de Wilcoxon del indicador SPREAD en el problema FireDrone con el conjunto de datos del bosque de Muniellos.

Friedman ranking and Holm test

Algorithm	Ranking	p-value	Holm	Hypothesis
RANDOMSEARCH-7	1	0E0	0	-
MOCell-3	2	7,744E-6	0,05	Rejected
ESPEA-2	3	3,744E-19	0,025	Rejected
NSGAIII-1	4	4,846E-41	0,017	Rejected
PAES-4	5	1,448E-71	0,013	Rejected
NSGAII-0	6	9,505E-111	0,01	Rejected
PESA2-5	7	1,339E-158	0,008	Rejected
SMSEMOA-6	8	3,996E-215	0,007	Rejected

Figura 105. Test de Friedman y Holm del indicador SPREAD en el problema FireDrone con el conjunto de datos del bosque de Muniellos.