ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DE COMPUTADORES

Optimización Multi-objetivo Basada en Preferencias para la Planificación de Proyectos Software

(Preference-based Multi-objective Optimization for Software Project Scheduling)

Realizado por

Rubén Saborido Infantes

Tutorizado por

Dr. José Francisco Chicano García

Departamento

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JULIO DE 2014

Fecha de defensa:

El Secretario del Tribunal

Resumen: En este trabajo se realiza una propuesta software denominada Interactive SPS o iSPS. Ésta, permite resolver, de forma interactiva, instancias del problema de Planificación de Proyectos Software (SPS) haciendo uso de algoritmos evolutivos basados en punto de referencia. La finalidad de la herramienta es ayudar al director de proyectos software en la toma de decisiones, teniendo en cuenta sus preferencias para aproximar una región concreta del frente óptimo de Pareto. Para validar la utilidad de este enfoque preferencial sobre el problema SPS, se realiza una comparativa entre los algoritmos implementados en iSPS y NSGA-II. Según los experimentos realizados consideramos que los algoritmos analizados basados en preferencias permiten guiar el proceso de búsqueda hacia una región concreta del espacio de objetivos, considerando las preferencias del decisor.
Palabras claves: Planificación de Proyectos Software, Optimización Multi-objetivo Evolutiva, Toma de Decisiones Multi-criterio, Pareto Optimalidad.

Abstract: This work proposes a software application called Interactive SPS or iSPS. It allows to solve, interactively, instances of the Software Project Scheduling (SPS) problem, by using preference-based evolutionary algorithms. The aim of this proposal is to help the project manager in the decision making, taking into account her/his preferences to approximate a region of the Pareto optimal front. To check the utility of this preference-based approach over the SPS problem, a comparative between the algorithms implemented in iSPS and NSGA-II is done. Considering the results obtained, we confirm the utility of the preference-based evolutionary algorithms to guide the search process toward a concrete region in the objective space, taking into account the decision makers' preferences.
Keywords: Software Project Schedulling, Evolutionary Multi-objective Optimization, Multiple Criteria Decision Making, Pareto optimality.

Índice general

Capítulo 1

Introducción

Muchos problemas de la vida real requieren optimizar varios criterios u objetivos, como por ejemplo tareas complejas en el dominio de la Ingeniería [40], la Bioinformática [34] o la Economía [30]. Este tipo de problemas se encuadran dentro de la *programación multi-objetivo* (MOP) [37], que es una rama de la Toma de Decisiones Multi-criterio [18]. En general, los objetivos presentan algún tipo de conflicto entre ellos, lo que imposibilita la búsqueda de una solución donde todos los criterios alcancen su valor óptimo individual. Además, normalmente disponen de una o varias restricciones que determinan el conjunto factible de soluciones. Al no poder resolverse calculando el valor óptimo de cada objetivo individualmente, es necesario establecer mecanismos matemáticos que posibiliten la comparación de soluciones. Un concepto clave en MOP es la relación de dominancia de Pareto [36], que define un conjunto de soluciones óptimas donde ninguno de los valores objetivo puede ser mejorado sin empeorar al menos uno de los restantes. A este conjunto de soluciones Pareto óptimas se le denomina *conjunto óptimo de Pareto* y a su imagen en el espacio de objetivos *frente óptimo de Pareto*.

Entre las metdologías existentes para resolver un problema de optimización multi-objetivo, *Multiple Criteria Decision Making* (MCDM) [18, 25] y *Evolutionary Multi-objective Optimization* (EMO) [3, 5] han contribuido con numerosas técnicas, aplicadas con éxito en problemas reales. Los métodos EMO, en general, pretenden encontrar un conjunto distribuido de soluciones Pareto óptimas que aproximen el frente óptimo de Pareto, mientras que MCDM considera las preferencias de un decisor para determinar un conjunto reducido de soluciones eficientes.

El *Problema de Planificación de Proyectos Software* (o problema SPS por sus siglas en

inglés de *Software Project Scheduling*) es un problema de optimización combinatoria de asignación de recursos a tareas, cuyos objetivos son minimizar la duración y el coste de un proyecto. En el ámbito de los proyectos software, los empleados o desarrolladores son considerados los recursos del proyecto y tienen asociado un salario, una dedicación y unas habilidades personales que le permiten intervenir en un determinado tipo de tareas en su jornada laboral. Por otro lado, las tareas a ejecutar poseen un orden lógico de ejecución, el cuál suele ser modelado mediante un Grafo de Precedencia de Tareas (o TPG, acrónimo inglés de *Task Precedence Graph*), lo que añade ciertas restricciones extras al problema. La planificación de proyectos software o SPS es una de las actividades de mayor importancia y, al mismo tiempo, de mayor complejidad en la dirección de proyectos de desarrollo de software. En este ámbito, el proceso de desarrollo difiere de otros procesos de producción por diversas razones. Por ejemplo, solventar un error en una tarea concreta, realizada por un desarrollador, podría llevar mucho más tiempo a otra persona. En este sentido, la reasignación de desarrolladores para resolver tareas que no han sido tratadas por ellos, puede tener efectos en la duración y, por tanto, en el coste.

En general, la solución a un problema de optimización multi-objetivo, como el SPS, consiste en obtener un conjunto de soluciones no dominadas. En los últimos años, con el uso de metaheurísticas se ha conseguido abordar, de forma económica, problemas complejos de este tipo. En concreto, las metaheurísticas poblacionales, como los Algoritmos Evolutivos, han demostrado adaptarse de forma aceptable a problemas de esta índole [23].

Desde finales de la década de los 90, hay cierto interés en hibridar la *Toma de Decisiones Multi-criterio* y los *Algoritmos Evolutivos* en la resolución de problemas de optimización multi-objectivo. Parece lógico combinar ambas disciplinas, introduciendo las preferencias de un decisor como parte del propio método EMO para ahorrar esfuerzos centrándose en soluciones deseadas y, al mismo tiempo, obviando soluciones que no serán útiles, dando lugar a lo que se conoce en la literatura como *preference-based EMO* [4].

Existen varios enfoques de algoritmos EMO basados en preferencias. Algunos de ellos hacen uso de conceptos e ideas extraídas del ámbito MCDM [6, 7, 17, 35], otros modifican métodos existentes, como el *Non-dominated Sorting Genetic Algorithm* (NSGA-II) [10], para incorporar preferencias [6, 7, 11] y algunos modifican o complementan la relación de dominancia de Pareto [28, 33].

En este trabajo, se propone una herramienta software desarrollada en Java, deno-

minada *Interactive SPS* (o *iSPS*), que permite resolver, de forma interactiva, cualquier instancia del problema SPS proporcionada por el usuario , haciendo uso de diferentes algoritmos evolutivos preferenciales basados en punto de referencia. Estos algoritmos tienen en cuenta las preferencias del decisor para aproximar la región de interés del frente óptimo de Pareto. Además, se realiza un estudio empírico sobre las prestaciones de los algoritmos evolutivos basados en preferencias, integrados en iSPS, sobre una instancia concreta del problema SPS.

El resto del documento se organiza de la forma siguiente. En el Capítulo 2 se definen conceptos de utilidad en el ámbito de la optimización multi-objetivo y se define formalmente el problema SPS. En el Capítulo 3 se describen algunas de las técnicas existentes en MCDM, EMO y preference-based EMO para la resolución de problemas de optimización multi-objetivo. En el Capítulo 4 se presenta la propuesta de este trabajo, describiendo el diseño, forma de uso y utilidad del software iSPS. En el Capítulo 5 se analiza el rendmimiento de los algoritmos evolutivos basados en preferencias disponibles en iSPS, sobre una instancia del problema SPS, haciendo uso de un indicador basado en hipervolumen para cuantificar la calidad de la aproximación de la región de interés. Finalmente, el Capítulo 6 concluye comentando las virtudes y defectos de las propuesta realizada y considera posibles líneas de trabajo futuras.

Capítulo 2

Optimización multi-objetivo

Este capítulo describe conceptos básicos en MOP, como Pareto optimalidad, conjunto Pareto óptimo y frente óptimo de Pareto. En todas estas definiciones, así como en el resto de este trabajo, se asume, sin pérdida de generalidad, la minimización de todos los objetivos. Además de lo anterior, se describe formalmente el problema SPS.

2.1 Conceptos y base teórica

Un problema de optimización multi-objetivo se define matemáticamente como

$$\begin{aligned} \text{minimizar} \quad & \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\} \\ \text{sujeto a} \quad & \mathbf{x} \in S \end{aligned} \tag{2.1}$$

donde $f_i : S \to \mathbb{R}$, para $i = 1, \ldots, k$ $(k \geq 2)$, son las *funciones objetivos* a minimizar simultáneamente, siendo $S \subset \mathbb{R}^n$ el *conjunto factible* de *variables de decisión* $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$. La imagen del conjunto factible en el *espacio de objetivos* $\mathbb{R}^k$ se denomina *región objetivo factible* $Z = \mathbf{f}(S)$, compuesta por los *vectores objetivos* $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x}))^T$, para $x \in S$.

No siempre es posible encontrar una solución donde todos los objetivos alcancen su valor óptimo individual, pero existe un conjunto de soluciones matemáticamente equivalentes en las que ninguno de los objetivos puede ser mejorado sin empeorar otro. En base a esto, se establece la *relación de dominancia de Pareto* según la cual, dados $x, x' \in S$, x *domina* a x' (denotado por $x \prec x'$) si $f_i(x) \leq f_i(x')$ para todo $i = 1, \ldots, k$ y existe un $j \in \{1, \ldots, k\}$ tal que $f_j(x) < f_j(x')$. Se dice que $x \in S$ es *eficiente* o *Pareto óptimo*

si no existe $x' \in S$ tal que $x' \prec x$. El conjunto de vectores de decisión no dominados se conoce como *conjunto Pareto óptimo*, mientras que el conjunto de vectores objetivos no dominados define el *frente óptimo de Pareto* o *conjunto de soluciones no dominadas*.

Al ser todas las soluciones Pareto óptimas igualmente deseables desde el punto de vista matemático, puede ser útil considerar las preferencias concretas de un decisor. Para ello existen múltiples mecanismos [25]. Uno de los más frecuentes, tanto en MCDM como en preference-based EMO, consiste en especificar el valor deseable para cada función objetivo, determinando un *punto de referencia* [41]. Éste se define como $\mathbf{q} = (q_1, \ldots, q_k)^T$, siendo q_i un valor de aspiración para la función objetivo f_i, para todo $i = 1, \ldots, k$. Un punto de referencia se dice que es *alcanzable* si existe alguna solución factible que lo iguale o mejore simultáneamente en todos los objetivos. En caso contrario, se dice que el punto de referencia es *inalcanzable*.

Muchos de los métodos existentes en MCDM y preference-based EMO que incorporan preferencias mediante uno o varios puntos de referencia, hacen uso del concepto de *función escalarizada de logro* [26]. Una de las más utilizadas fue propuesta por Wierzbicki [42], y su definición, basada en la distancia L_∞, viene dada por la expresión (2.2).

$$s(\mathbf{q}, \mathbf{f}(\mathbf{x}), \mu) = \max_{i=1,\ldots,k} \{ \mu_i(f_i(\mathbf{x}) - q_i) \} + \rho \sum_{i=1}^{k} \mu_i(f_i(\mathbf{x}) - q_i), \tag{2.2}$$

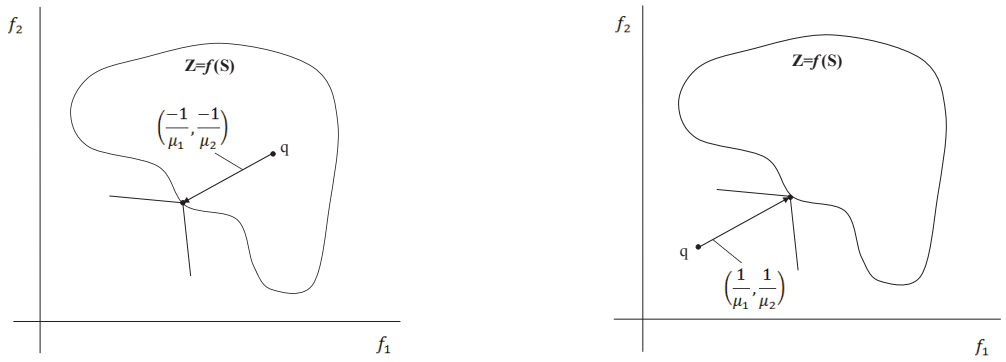
la cual debe ser minimizada sobre S:

$$\begin{aligned} \text{minimizar} \quad & s(\mathbf{q}, \mathbf{f}(\mathbf{x}), \mu) \\ \text{sujeto a} \quad & \mathbf{x} \in S \end{aligned} \tag{2.3}$$

Siendo $\mathbf{q} = (q_1, \ldots, q_k)^T$ un punto de referencia y $\mu = (\mu_1, \ldots, \mu_k)$ un vector de pesos con $\mu_i > 0$ para todo $i = 1, \ldots, k$, usado como coeficiente normalizador [32] o preferencial [24]. El parámetro ρ debe ser estrictamente positivo y se usa para asegurar que la solución de (2.2) es Pareto óptima para el problema (2.1) [27]. Normalmente suele fijarse a un valor constante de 0.001.

En esencia, la minimización de esta función escalarizada de logro supone proyectar el punto de referencia sobre el frente óptimo de Pareto, en la dirección determinada por el inverso de los pesos, tal y como se muestra en la Figura 2.1.

En ocasiones es útil conocer el rango de variación de las funciones objetivo ya que permite definir dos vectores en el espacio de objetivos denominados *ideal* y *nadir*, que



(a) Punto de referencia alcanzable (b) Punto de referencia inalcanzable

Figura 2.1: Solución de (2.3) para un problema de optimización de dos objetivos.

representan, respectivamente, al mejor y al peor valor de cada objetivo en el conjunto de vectores de decisión no dominados, el cual se define como PS.

El ideal, denotado por $\mathbf{z}^\star$, se obtiene minimizando cada función objetivo individualmente en la región factible. Formalmente, se define como:

$$\mathbf{z}^\star = (z_1^\star, \ldots, z_k^\star)^T \in \mathbb{R}^k, z_i^\star = \min_{\mathbf{x} \in S} f_i(\mathbf{x}) = \min_{\mathbf{x} \in PS} f_i(\mathbf{x}) \; \forall i = 1, \ldots, k$$

El nadir, denotado por $\mathbf{z}^{\text{nad}}$, es definido como:

$$\mathbf{z}^{\text{nad}} = (z_1^{\text{nad}}, \ldots, z_k^{\text{nad}})^T \in \mathbb{R}^k, z_i^{\text{nad}} = \max_{\mathbf{x} \in PS} f_i(\mathbf{x}) \; \forall i = 1, \ldots, k$$

Normalmente el vector objetivo nadir es díficil de calcular. En esos casos puede realizarse una estimación [8, 9, 13, 38].

2.2 El problema SPS

Una novedosa formulación del problema SPS se define en [2]. E representa al conjunto de personas involucradas en un proyecto software, donde cada empleado se denota con e_i y su correspondiente salario por e_i^s, variando i desde 1 a $|E|$. El conjunto de tareas del proyecto y cada tarea individual son definidas por T y $t_j \in T$, respectivamente, variando j desde 1 a $|T|$. El coste en persona-hora de cada tarea t_j se denota por t_j^c. Cada tarea requiere que se completen otras para poder comenzar. Estas precedencias se expresan con un grafo de precedencia de tareas (TPG). Éste, es un grafo acíclico dirigido $G(T, A)$ cuyos nodos representan las tareas y un arco $(t_i, t_j) \in A$ indica la precedencia de la tarea t_i sobre

la tarea t_j. Cada instancia del problema incluye una matriz $\mathbf{P}$ de tamaño $|E| \times |T|$ donde el elemento $\mathbf{P}_{i,j} \in [0, 1]$ es un número real positivo que determina la productividad del empleado e_i en la tarea t_j.

Una solución al problema SPS se denota por $\mathbf{x} = (\mathbf{d}, \mathbf{r}, \mathbf{Q})$ con $\mathbf{d} \in \mathbb{R}^{|E|}$, $\mathbf{r} \in \mathbb{N}^{|T|}$ y $\mathbf{Q} \in \mathbb{N}^{|E|\times|T|}$. Donde $\mathbf{d}$ contiene la dedicación de la jornada diaria laboral de cada empleado en las tareas del proyecto, $\mathbf{r}$ el retraso en horas en cada tarea y $\mathbf{Q}$ las tareas realizadas por cada empleado.

Dado que la dedicación de un empleado a una tarea es dependiente del tiempo, para calcular la duración del proyecto hace falta conocer la mano de obra usada en cada instante en cada tarea. El vector π es un vector dependiente del tiempo que indica la mano de obra empleada en una tarea en cada hora.

El conjunto de tareas finalizadas (*done*) y activas (*active*) en un instante de tiempo τ, basado en los valores de $\pi(\tau')$ para $\tau' < \tau$, viene definido por

$$done(\tau) = \left\{ t_j \in T \middle| \sum_{\tau'=0}^{\tau-1} \pi_j(\tau') \geq t_j^c \right\}, \tag{2.4}$$

y

$$active(\tau) = \{t_j \in T | \forall t_i, (t_i, t_j) \in A : t_i \in done(\tau - r_j)\} - done(\tau), \tag{2.5}$$

donde A representa el conjunto de arcos del TPG.

El vector $\pi_j(\tau)$ se puede calcular para cada paso de tiempo $\tau = 1, 2, ...$ iterativamente de la forma siguiente:

$$\pi_j(\tau) = \begin{cases} \sum\limits_{e_i \in E: \mathbf{Q}_{i,j} > 0} \frac{d_i \cdot \mathbf{P}_{i,j} \cdot \mathbf{Q}_{i,j}}{\sum_{t_k \in active(\tau)} \mathbf{Q}_{i,k}} & \text{si } t_j \in active(\tau), \\ \\ 0 & \text{en otro caso.} \end{cases} \tag{2.6}$$

Dado que la duración (*makespan*) del proyecto viene determinada por la cantidad de tiempo requerido en completar todas las tareas, su cálculo puede realizarse de la forma siguiente:

$$makespan(x) = min\left\{\tau \in \mathbb{N} | done(\tau) = T\right\}. \tag{2.7}$$

El coste (*cost*) del proyecto se calcula multiplicando el salario por hora de cada empleado por su dedicación a cada una de las tareas del proyecto. Según esto,

$$cost(x) = \sum_{e_i \in E} e_i^s \cdot d_i \cdot \left|\left\{\tau \in \mathbb{N} | \exists t_j \in active(\tau) : \mathbf{Q}_{(i,j)} \cdot \mathbf{P}_{(i,j)} > 0\right\}\right|. \tag{2.8}$$

Considerando las expresiones matemáticas de la duración y del coste de un proyecto, el problema SPS puede ser modelado como un problema de optimización bi-objectivo, con función objetivo

$$f(x) = (cost(x), makespan(x)). \tag{2.9}$$

Capítulo 3

Antecedentes

Muchos problemas de decisión y planificación tratan con múltiples objetivos en conflicto que deben ser considerados simultáneamente. En estos problemas no existe una solución única así que el interés recae en identificar un conjunto de soluciones no dominadas, eficientes o Pareto óptimas.

3.1 Optimización multi-objetivo en MCDM

En el ámbito MCDM, la idea de resolver un problema de optimización se basa principalmente en ayudar a un decisor o *decision maker* (DM) a considerar simultáneamente múltiples objetivos para encontrar la solución Pareto óptima que mejor se adapte a sus preferencias. En general, el DM es una persona con cierto conocimiento sobre el problema a resolver, por lo que es capaz de proporcionar información preferencial sobre los objetivos o soluciones de interés. Considerando el rol del DM en la resolución de problemas de optimización multi-objetivo, MCDM realiza una clasificación en tres categorías bien diferenciadas:

- Métodos *a priori*. Obtienen una solución eficiente a partir de la información preferencial proporcionada por el DM antes del proceso de optimización. Tienen como principal inconveniente que el DM no tiene por qué conocer las limitaciones del problema de antemano, pudiendo ser demasiado optimista o pesimista en sus preferencias.

- Métodos *a posteriori*. Generan una aproximación del conjunto de soluciones Pareto óptimas, sobre la que el DM selecciona, en base a sus preferencias, la solución de interés. Con este enfoque el DM adquiere conocimiento sobre las diferentes soluciones disponibles pero, si hay más de dos objetivos en el problema, resulta complicado su análisis. Además, generar una buena aproximación del frente óptimo de Pareto puede ser computacionalmente caro en problemas complejos.

- Métodos *interactivos*. Muestran determinada información del problema cada cierto número de iteraciones, lo que proporciona conocimiento al DM para que ajuste sus preferencias si lo considera necesario.

3.2 Optimización multi-objetivo evolutiva (EMO)

Los Algoritmos Evolutivos (AE) son métodos de optimización basados en los postulados de la evolución biológica de las especies. A partir de un conjunto inicial de soluciones, éstas se mezclan y compiten entre sí para evolucionar a mejores soluciones durante la ejecución de un proceso iterativo. De forma que las soluciones más aptas son las que, con mayor probabilidad, prevalecen a lo largo del tiempo.

Este tipo de algoritmos son especialmente indicados para resolver problemas donde el espacio de búsqueda es muy extenso y otros métodos de búsqueda no son capaces de encontrar soluciones aceptables en una magnitud de tiempo razonable.

Siguiendo la terminología de la teoría de la evolución biológica, cada solución del problema se denomina *individuo*, y al conjunto de éstos *población*. Los individuos son modificados durante el proceso evolutivo por operadores genéticos. La *recombinación* consiste en la mezcla de la información de dos o más individuos. El operador de mutación realiza un cambio, normalmente aleatorio, en un individuo. Finalmente, el operador de *selección* se encarga de escoger a aquellos individuos que sobrevivirán y conformarán la siguiente generación. Puesto que los individuos que representan las soluciones más adecuadas al problema tienen más posibilidades de sobrevivir, la población va mejorando gradualmente en un proceso iterativo.

El Algoritmo 1 muestra el pseudocódigo de un AE genérico. Como se describe, el proceso de optimización comienza con la generación de una población de soluciones, usualmente

aleatoria, dentro del intervalo determinado por la cota inferior y superior de cada variable de decisión. Mientras no se verifique un criterio de parada, se actualiza la población mediante operadores de *selección*, *recombinación* y *mutación*. Con la finalidad de determinar la calidad de las soluciones en cada iteración o *generación*, suele hacerse uso de alguna relación de orden o dominancia que permita distinguir buenas y malas soluciones numéricamente, considerando el valor de los objetivos y las restricciones del problema.

Algoritmo 1 Pseudocódigo de un AE genérico.

```
1: P(0) ← GenerarPoblaciónInicial()
2: EvaluarObjetivos(P(0))
3: PF ← CrearFrentePareto()
4: g ← 0
5: mientras not (CondiciónParada()) hacer
6:    padres ← Selección(P(g))
7:    descendientes ← Recombinación(padres)
8:    descendientes ← Mutación(descendientes)
9:    EvaluarObjetivos(descendientes)
10:   P(g + 1) ← ActualizarPoblación(P(g), descendientes)
11:   ActualizarFrontera(PF, P(g + 1))
12:   g ← g + 1
13: fin mientras
```

El uso de AE para la resolución de problemas de optimización multi-objetivo brinda varias ventajas, respecto a metodologías de optimización clásicas. Al estar basados en población, tratan con multitud de soluciones potenciales en cada generación. A pesar del coste computacional que esto conlleva, éste puede ser reducido drásticamente mediante técnicas de paralelización. En cualquier caso, la finalidad de EMO es encontrar un conjunto bien distribuido de soluciones Pareto óptimas que posibilite la aproximación del frente óptimo de Pareto.

Algunos de los EMO's existentes con mayor impacto en publicaciones y aplicaciones reales son *Non-dominated Sorting Genetic Algorithm* (NSGA-II) [10], *Pareto Archived Evolution Strategy* (PAES) [21], *Strength Pareto Evolutionary Algorithm* (SPEA2) [44] y *Multi-objective Optimization Evolutionary Algorithm Based on Decomposition* (MOEA/D) [43].

Varios algoritmos de los nombrados anteriormente son utilizados para resolver el problema de optimización SPS, descrito en la Sección 2.2. En [2] se realiza un exhaustivo estudio y análisis comparativo entre distintos AE aplicados al problema SPS.

3.3 Preference-based EMO

Los métodos de optimización multi-objetivo evolutivos basados en preferencias, pretenden incorporar las preferencias del DM a un algoritmo evolutivo para aproximar la región de interés en el frente óptimo de Pareto. Esto es factible cuando el DM posee cierto conocimiento sobre el problema, lo que permite delimitar el conjunto de soluciones preferidas. Esta información puede ser empleada para enfocar la búsqueda hacia la región del frente óptimo de Pareto de mayor interés, bajo el punto de vista del decisor.

Existen múltiples formas de aportar preferencias en EMO [4, 29]. Un enfoque frecuente consiste en definir uno o varios puntos de referencia que determinan las aspiraciones del DM para cada criterio u objetivo [41], delimitando la región de interés del frente óptimo de Pareto. Otras alternativas definen algunos de los objetivos como restricciones [4], determinan tasas de intercambio o *trade-offs* entre criterios (cuántas unidades son permisibles en el empeoramiento de un objetivo a cambio de mejorar a otro), establecen o modifican la relación de preferencia entre soluciones [14] o adaptan la función objetivo para tener en cuenta las consideraciones del DM.

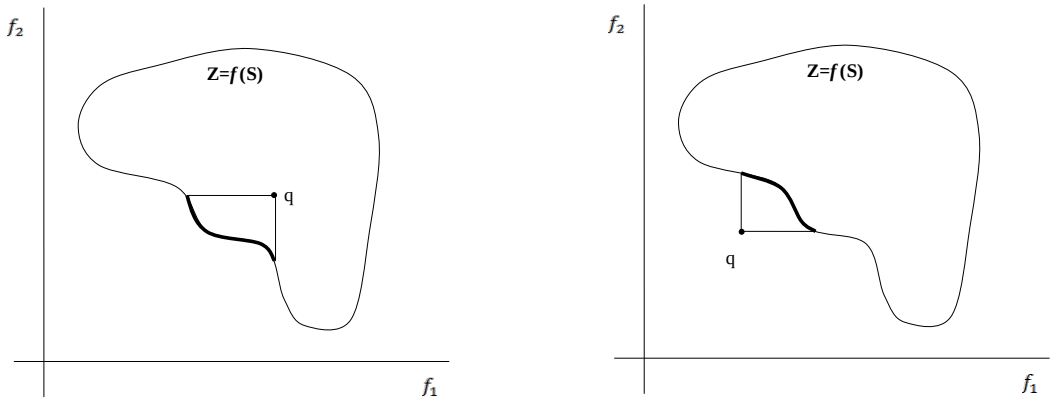
La adaptación a EMO de los enfoques a posteriori, interactivo y a priori de MCDM, es directa. Los métodos a posteriori consideran las preferencias del DM sobre la población resultante de un algoritmo evolutivo para obtener la solución Pareto óptima más apropiada. El enfoque interactivo interactúa con el DM cada cierto número de generaciones para orientar gradualmente la optimización del problema considerando sus preferencias. Por el contrario, los métodos a priori parten de la información del DM que usan durante el proceso de optimización para guiar a un algoritmo evolutivo sin la interacción del decisor.

Actualmente existen multitud de algoritmos EMO basados en preferencias. *Guided Multi-objective Evolutionary Algorithm* (G-MOEA) [1] modifica el criterio de dominancia para incorporar las preferencias del DM mediante trade-offs aceptables. *Preference Based Evolutionary Algorithm* (PBEA) [39] usa un indicador de calidad basado en funciones escalarizadas de logro definidas para uno o varios puntos de referencia. En [35] propo-

nen un EMO interactivo que se apoya en la generación de conos poliédricos, a partir de las preferencias proporcionadas por el DM en cada interacción, para guiar el proceso de búsqueda de un algoritmo evolutivo. En [28] definen una relación de dominancia, denominada *g-dominance*, que da prioridad a las soluciones que dominen o sean dominadas por un punto de referencia, determinado por el DM. Recientemente, en [33] definen otra relación de dominancia, llamada *r-dominance*, que complementa la relación de dominancia de Pareto considerando la distancia euclídea a un punto de referencia cuando dos soluciones son Pareto equivalentes. Otro enfoque basado en puntos de referencia es *Reference point based NSGA-II* (R-NSGA-II) [11]. Éste, modifica la forma de clasificar las soluciones de la población del NSGA-II en la última frontera considerando la proximidad de cada solución a cada punto de referencia. *Reference Direction based NSGA-II* (RD-NSGA-II) [6] incorpora a NSGA-II una metodología extraída de MCDM denominada *dirección de referencia* [22]. A partir de un punto del espacio de objetivos y un punto de referencia proporcionado por el DM se define una dirección de referencia, considerando la diferencia entre ambos. Sobre ésta se definen puntos de referencia equidistantes que son proyectados sobre el frente óptimo de Pareto mediante la función escalarizada de logro. Otra idea extraída de MCDM, denominada *Light Beam Search* [19], ha sido utilizada en [7] con su integración en NSGA-II. Un enfoque interactivo de MOEA/D es propuesto por *interactive MOEA/D* (iMOEA/D) [17]. Tras un número determinado de generaciones se muestran un conjunto de soluciones al DM, que especifica sus preferencias sobre éstas. El conjunto de pesos usado en MOEA/D para optimizar múltiples funciones de logro es acotado al vecindario de las soluciones determinadas como preferidas. Así, el proceso de búsqueda se orienta progresivamente hacia la región de interés del frente óptimo de Pareto.

3.3.1 Región de interés definida por un punto de referencia q

En determinadas situaciones, bien por el interés del DM o por exigencias técnicas, el conjunto de soluciones preferidas de un problema de optimización multi-objetivo puede restringirse a una región de interés. Cuando el DM dispone de cierto conocimiento del problema, puede proporcionar sus preferencias mediante un punto de referencia, denotado por $\mathbf{q}$. Éste, determina el valor deseable para cada función objetivo o criterio. Según esto, el DM puede limitar el conjunto de soluciones deseadas a aquellas que dominan o son dominadas por el punto de referencia, tal y como se indica en la Figura 3.1.



(a) Punto de referencia alcanzable (b) Punto de referencia inalcanzable

Figura 3.1: Región de interés en un problema con dos objetivos.

Hipervolumen de la región de interés definida por q

Para medir la calidad de la aproximación de la región de interés, se hace uso de la métrica propuesta en [31], basada en el *hipervolumen* [16, 45].

Para un conjunto de soluciones P, su hipervolumen, denotado por $HV(P)$, es el volumen (en el espacio de objetivos) cubierto por las soluciones de P. Formalmente, para cada solución $i \in P$, se construye un hipercubo v_i acotado por un punto de referencia $\mathbf{R}$ y la propia solución i. El punto de referencia $\mathbf{R}$ puede fijarse al vector nadir.

La unión de todos los hipercubos de las soluciones de P determinan el valor de $HV(P)$, tal y como indica la expresión siguiente:

$$HV(P) = volumen\left(\bigcup_{i=1}^{|P|} v_i\right). \tag{3.1}$$

La Figura 3.2 muestra la idea gráfica de la métrica HV para un problema bi-objetivo. Los puntos representan las soluciones pertenecientes a la aproximación del frente P.

Para un punto de referencia $\mathbf{q}$ y un conjunto de soluciones P, denotamos por $HV_{\mathbf{q}}(P)$ al *hipervolumen de P en la región de interés definida por* $\mathbf{q}$. Para calcular $HV_{\mathbf{q}}(P)$, el punto de referencia $\mathbf{R}$ se determina de la forma siguiente:

- Si $\mathbf{q}$ es alcanzable, entonces $\mathbf{R} = \mathbf{q}$, esto es, $r_i = q_i$ para todo $i = 1, \ldots, k$.
- Si $\mathbf{q}$ es inalcanzable, entonces $\mathbf{R}$ se define como sigue. Supongamos que A es un frente de referencia, el cual es una buena aproximación del frente óptimo de Pareto. Entonces, $r_i = \text{máx}_{\mathbf{x} \in A_{\mathbf{q}}} f_i(\mathbf{x})$ para todo $i = 1, \ldots, k$, donde $A_{\mathbf{q}}$ es el subconjunto de soluciones en A que pertenecen a la región de interés definida por $\mathbf{q}$.

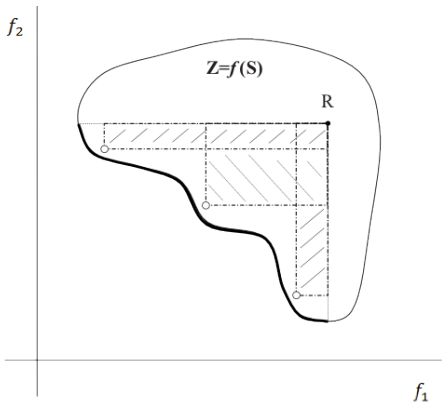
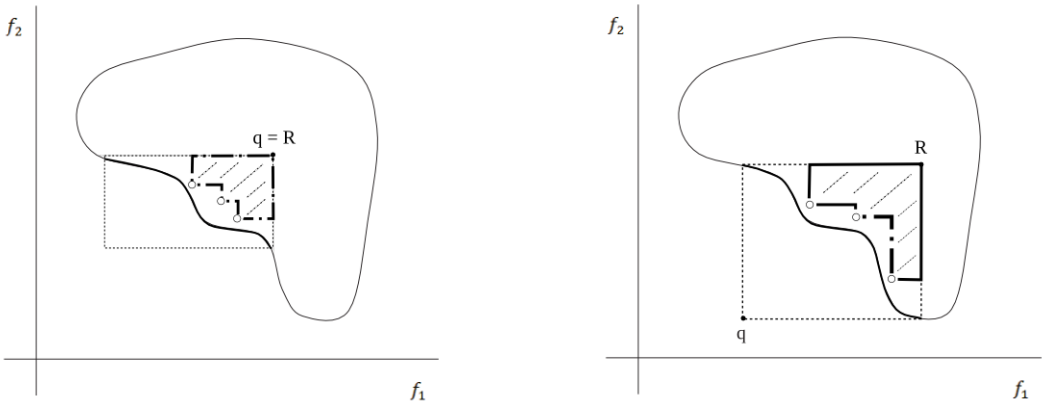


Figura 3.2: Idea gráfica de la métrica HV en un problema bi-objetivo.

La Figura 3.3 da una idea gráfica del área calculada por la métrica $HV_{\mathbf{q}}$, para un problema bi-objetivo. Se muestra la región de interés definida para el punto de referencia $\mathbf{q}$ y un conjunto de puntos o soluciones que representa una posible aproximación de la región de interés. El área rayada corresponde al $HV_{\mathbf{q}}$.



(a) Punto de referencia alcanzable (b) Punto de referencia inalcanzable

Figura 3.3: Idea gráfica de la métrica $HV_{\mathbf{q}}$ en un problema bi-objetivo.

Capítulo 4

iSPS: un software de resolución interactivo

El enfoque preferencial basado en punto de referencia resulta interesante en métodos interactivos. Éstos, cada cierto número de iteraciones, muestran las soluciones encontradas e interactúan con el decisor o *decision maker* (DM) solicitando sus preferencias. Para ayudar en la toma de decisiones al director de proyectos software, se ha desarrollado una herramienta que permite ejecutar diferentes algoritmos de optimización multi-objetivo basados en preferencias sobre instancias del problema SPS. Como se explica en el caso práctico descrito en este capítulo, la herramienta iSPS puede ser de utilidad para optimizar la planificación de proyectos software considerando las preferencias del director de proyectos. Cuando el DM no dispone de suficiente conocimiento previo sobre el problema, los métodos interactivos permiten adquirir, de forma iterativa, conocimiento sobre el mismo.

Interactive SPS o iSPS es un software que permite la resolución, mediante un enfoque interactivo, de instancias del problema SPS, haciendo uso de diferentes algoritmos evolutivos basados en punto de referencia existentes en la literatura. Éstos se describen en la Sección 4.1 de este capítulo.

La herramienta propuesta ha sido desarrollada en Java para permitir su ejecución multi-plataforma. Además, hace uso de una librería adicional que permite la elaboración y representación de gráficas a través del software libre *GNUPlot*. Según esto, los requisitos de ejecución de la aplicación son los siguientes:

1. Máquina virtual de Java en su versión 1.7 o superior.

2. *GNUPlot* en su versión 4.3 o superior, incluido en el *PATH* del Sistema Operativo.

4.1 AE basados en preferencias integrados en iSPS

Entre los algoritmos EMO existentes basados en preferencias mediante punto de referencia, destacan g-NSGA-II [28], el algoritmo genético preferencial definido en [15], al que nos referiremos como P-MOGA, y WASF-GA [31]. Éstos permiten aproximar la región de interés del frente óptimo de Pareto considerando las preferencias del DM, tal y como se expone en la Figura 3.1. Todos ellos se han implementado dentro del *framework* jMetal [12], el cual ofrece un conjunto de clases Java que facilitan el desarrollo de metaheurísticas para la resolución de problemas de optimización multi-objetivo, beneficiándose de las propiedades del paradigma de la programación orientada a objetos.

g-NSGA-II define una nueva relación de dominancia denominada *g-dominance*. Ésta, prefiere soluciones que dominan o son dominadas por el punto de referencia. La ventaja principal es que esta relación de dominancia puede ser incorporada en cualquier metaheurística, ya que no modifica la estructura del algoritmo. Por el contrario, se ha demostrado que *g-dominance* no preserva la dominancia de Pareto, por lo que soluciones dominadas pueden ser preferidas a la solución que las domina [33]. g-NSGA-II es el algoritmo resultante de modificar la relación de dominancia de Pareto de NSGA-II por la relación *g-dominance*.

P-MOGA trata las preferencias del DM modificando la forma en la que las soluciones son comparadas durante la ejecución de un algoritmo genético. Para ello, permite la degradación de los objetivos que mejoran al punto de referencia a cambio de mejorar las componentes que lo empeoran. Esto hace posible decantarse por una solución u otra considerando el punto de referencia, incluso cuando ambas soluciones son no dominadas.

WASF-GA hace uso de la función escalarizada de logro de Wierzbicki, definida por la expresión (2.2), para clasificar los individuos de cada generación en diferentes fronteras. Esta clasificación se realiza en base a los valores que la función de logro alcanza para el punto de referencia proporcionado por el DM, considerando un conjunto de vectores de pesos. El algoritmo pretende aproximar la región de interés con un conjunto de soluciones bien distribuida.

En todos los algoritmos anteriores se ha utilizado el operador *BinaryTournament* para

la selección de individuos, *TwoPointsCrossover* para la recombinación y *RandomMutation* para la mutación.

El operador de selección *BinaryTournament*, escoge una solución realizando una comparación directa entre dos soluciones considerando, normalmente, alguna relación de dominancia entre ellas.

La *recombinación en dos puntos* o *TwoPointsCrossover* es un operador de recombinación que determina dos puntos aleatorios e intercambia el valor de las variables de decisión existentes entre los dos puntos de recombinación creando dos soluciones nuevas. Una idea gráfica del funcionamiento de este operador de recombinación se expone en la Figura 4.1.

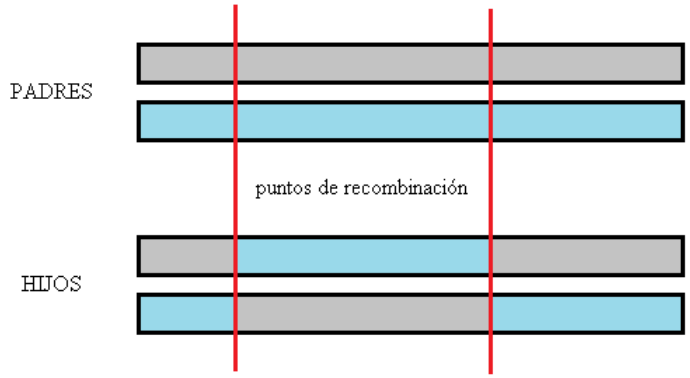


Figura 4.1: Idea gráfica del operador de recombinación *TwoPointsCrossover*.

El operador de mutación *RandomMutation* modifica, con una probabilidad P_m, cada una de las variables de decisión de una solución. Cuando se muta una variable i, se asigna un nuevo valor aleatorio $x_i' \in [ci, cs]$, siendo ci y cs las cotas inferior y superior, respectivamente, de esa variable.

4.2 Arquitectura software de iSPS

jMetal [12] es un *framework* basado en Java que permite desarrollar y analizar la utilidad de metaheurísticas en la resolución de problemas de optimización multi-objetivo. Además, proporciona una gran cantidad de algoritmos existentes en la literatura científica, un extenso conjunto de problemas de prueba o *benchmarks*, así como un conjunto de métricas que permiten cuantificar la calidad de la aproximación de los algoritmos. Además, proporciona mecanismos para realizar estudios experimentales que permiten comparar el

rendimiento de diferentes algoritmos sobre distintos problemas.

jMetal se distribuye como software libre y ha sido desarrollado haciendo uso del paradigma de orientación a objetos. Este hecho permite que sus funcionalidades puedan ser utilizadas por terceros o, incluso, ser extendidas y personalizadas por otros desarrolladores.

iSPS hace uso de jMetal ya que proporciona un conjunto de clases que facilitan enormemente la implementación de nuevas metaheurísticas, mediante la reutilización del código existente.

La Figura 4.2 muestra el diagrama de clases UML de jMetal. Como puede observarse, modela algoritmos, conjuntos de soluciones, problemas y operadores. Además, implementa diferentes algoritmos y problemas de prueba.

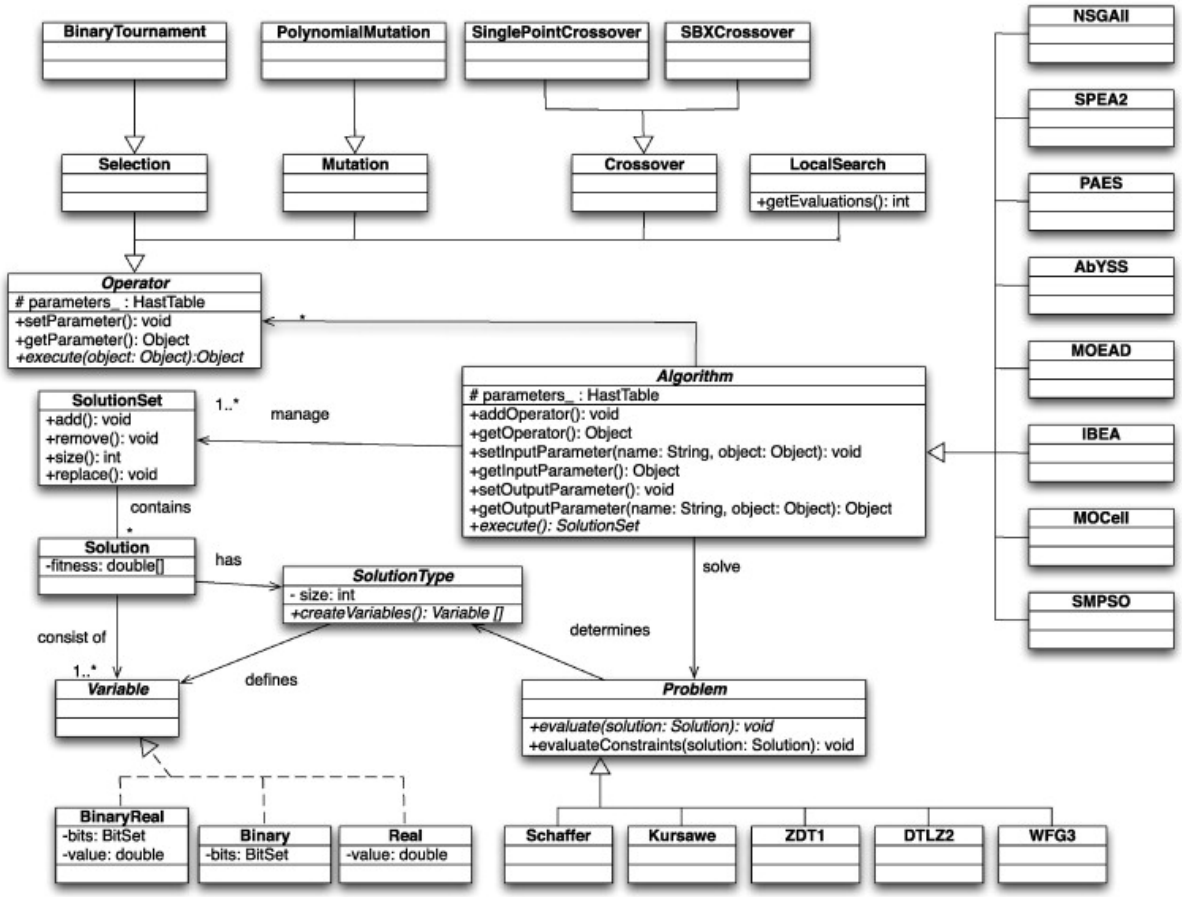


Figura 4.2: Diagrama de clases de jMetal.

iSPS hace uso de parte del conjunto de clases del núcleo de jMetal para definir e implementar los algoritmos evolutivos iGNSGAII, iPMOGA e iWASFGA. Como estos algoritmos poseen un enfoque interactivo, deben proporcionarse mecanismos que manipu-

len la información suministrada por el DM en cada interacción. Para ello, se ha definido una clase abstracta *iAlgorithm* que hereda de la clase abstracta *Algorithm* de jMetal. iGNSGAII, iPMOGA e iWASFGA se implementan como algoritmos interactivos, así que extienden la clase abstracta *iAlgorithm*. Para facilitar el proceso de interacción con el decisor, iSPS ofrece una interfaz gráfica de usuario, la cual se comenta en profundidad en la Sección 4.3. Con la finalidad de separar los datos de la interfaz gráfica, iSPS se ha implementado siguiendo el patrón de arquitectura software *Modelo-Vista-Controlador* (MVC). Éste, propone la construcción de tres componentes distintos que definen la información o lógica de la aplicación, la apariencia gráfica de la misma y la interacción con el usuario, respectivamente. La Figura 4.3 muestra la arquitectura software de iSPS. Aunque no se detalle en el diagrama, al igual que jMetal, iSPS define otras clases secundarias. Éstas, son de utilidad, por ejemplo, para modelar los conceptos de punto de referencia y función escalarizada de logro.

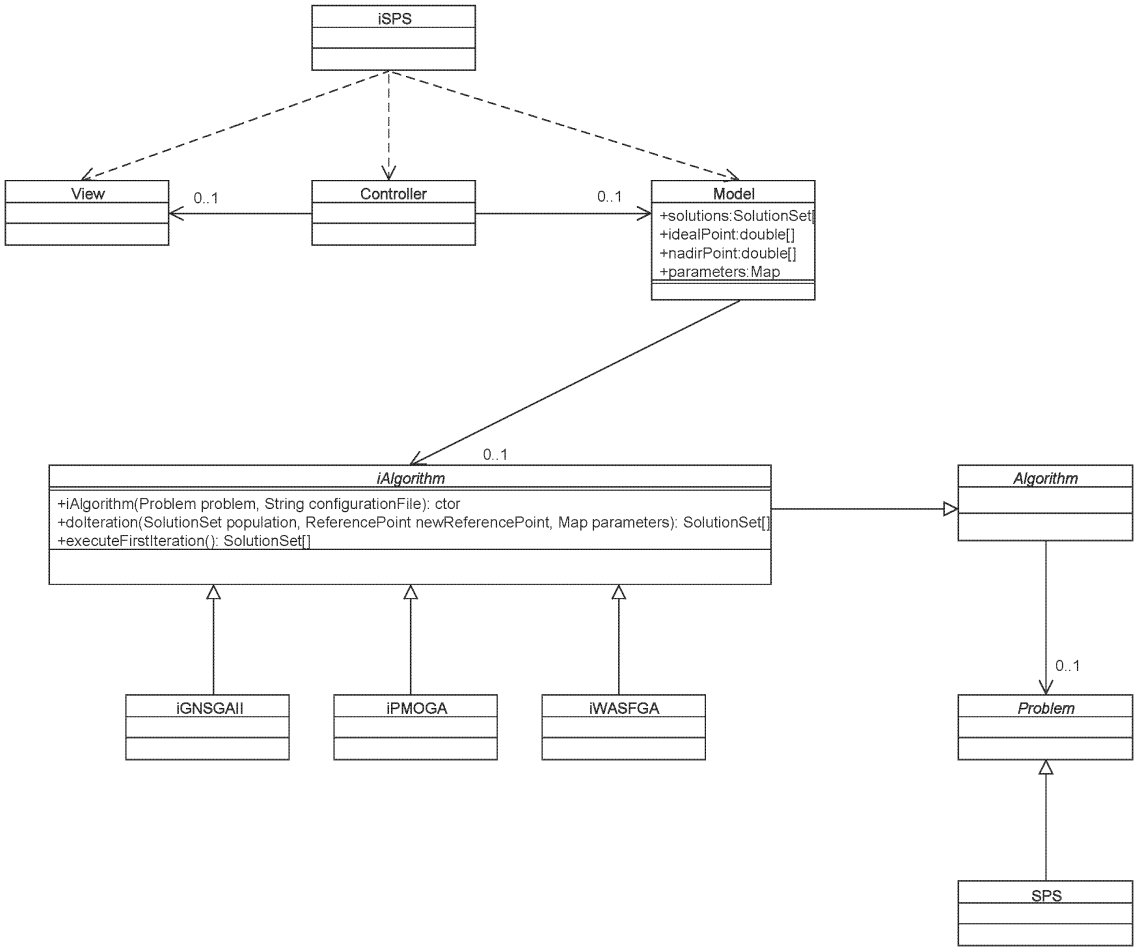


Figura 4.3: Arquitectura software de iSPS.

4.3 Interfaz gráfica de usuario de iSPS

La interfaz gráfica de usuario de iSPS es simple, pero ofrece todo lo necesario para cargar instancias del problema SPS, determinar el algoritmo a usar para su resolución incorporando preferencias mediante punto de referencia y visualizar los resultados obtenidos gráficamente. La apariencia inicial de la aplicación se muestra en la Figura 4.4. Como puede observarse, la interfaz se divide en 7 zonas bien diferenciadas, las cuales son comentadas a continuación:

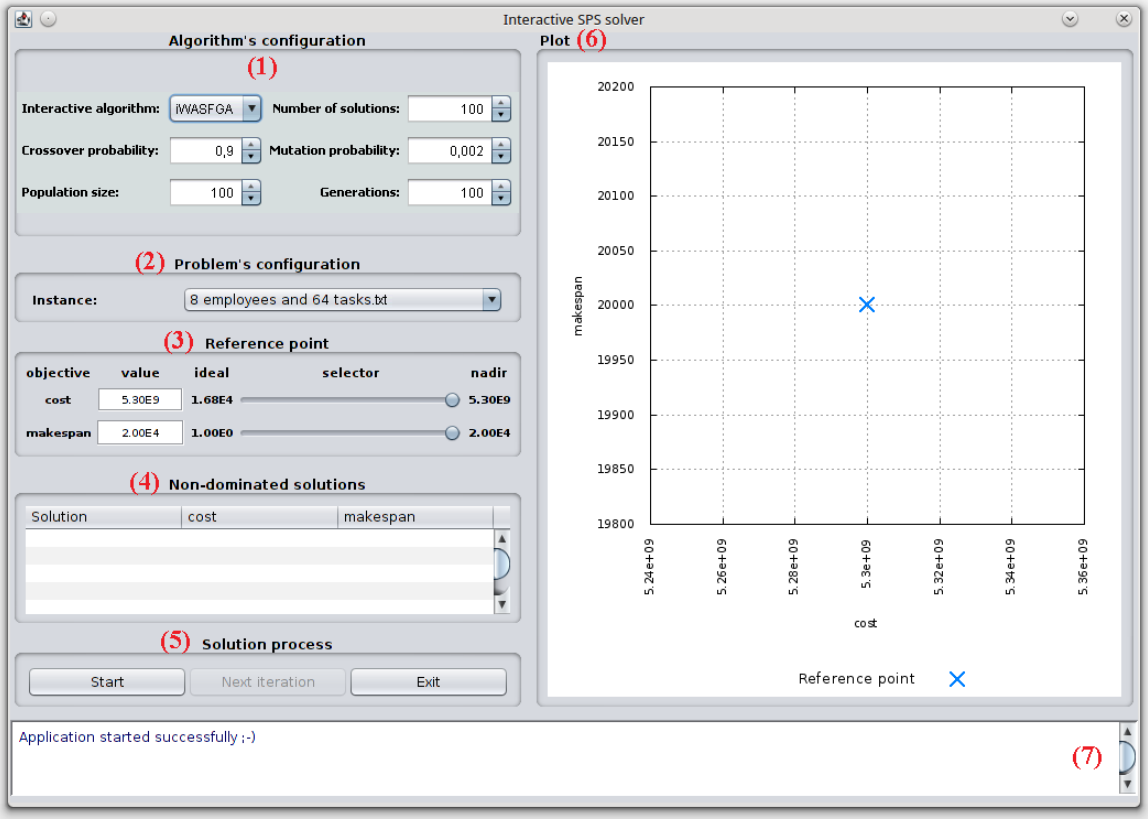


Figura 4.4: Interfaz gráfica de usuario de iSPS.

1) Configuración del algoritmo

Permite seleccionar el algoritmo a ejecutar para resolver el correspondiente problema SPS, así como modificar distintos parámetros del algoritmo escogido.

– *Algoritmo interactivo.*

  Permite seleccionar el algoritmo basado en punto de referencia a utilizar para la resolución del problema SPS escogido. Este algoritmo puede ser iGNSGAII, iPMOGA o iWASFGA, que corresponden a versiones interactivas de los algoritmos g-NSGA-II, P-MOGA y WASF-GA, respectivamente. Estos han sido desarrollados e integrados en jMetal tal y como se describe en la Sección 4.2.

– *Número de soluciones.*

  Determina el número máximo de soluciones para iWASFGA. Este campo se desactiva para los otros algoritmos, ya que en ellos el número máximo de soluciones final viene determinado por el tamaño de la población.

– *Tamaño de la población.*

  Especifica el número máximo de soluciones que devuelven los algoritmos iGNSGAII y iPMOGA. Para iWASFGA determina el conjunto de soluciones que internamente usa el algoritmo pero, en éste, el número máximo de soluciones finales es determinado por el parámetro *Número de soluciones.*

– *Número de generaciones.*

  Este parámetro, junto al anterior, concreta el número de evaluaciones de la función objetivo usado como criterio de parada del algoritmo seleccionado. Cuanto mayor sea su valor, mayor será el tiempo de cómputo requerido pero mejor debería ser la aproximación del frente obtenida.

– *Probabilidad de recombinación.*

  Fija la probabilidad del operador de recombinación. Este operador se usa para crear nuevas soluciones a partir de soluciones anteriores. Esta probabilidad determina con qué frecuencia se generan soluciones descendientes a partir de soluciones padres. Dependiendo del tipo de operador usado, valores altos próximos a uno pueden mejorar la convergencia del algoritmo.

– *Probabilidad de mutación.*

  Fija la probabilidad del operador de mutación. Este operador se usa para modificar soluciones existentes en la población, modificando el valor de las variables de deci-

sión. Esta probabilidad determina con qué frecuencia se muta cada solución. Suelen usarse valores bajos, ya que valores altos pueden perjudicar a la convergencia del algoritmo. Por defecto, el valor propuesto es de $1/n$, siendo n el número de variables de decisión de la instancia del problema SPS seleccionada.

2) Configuración del problema

Permite seleccionar la instancia del problema SPS a resolver. Por defecto se ofrecen las instancias siguientes, las cuales han sido generadas automáticamente con el generador de instancias del problema SPS usado en [2]:

- 8 empleados y 64 tareas.txt.
- 8 empleados y 128 tareas.txt.
- 8 empleados y 256 tareas.txt.
- 32 empleados y 128 tareas.txt.
- 32 empleados y 256 tareas.txt.
- 32 empleados y 512 tareas.txt.

Cada instancia disponible corresponde a un fichero de texto plano, existente en la carpeta */data/instances* de la aplicación, que contiene la configuración propia del problema instanciado. Otras instancias pueden ser añadidas por el usuario. Para ello, basta copiar los archivos creados por un generador de instancias en la carpeta anterior.

3) Punto de referencia

Este panel se usa para especificar las preferencias del DM. Muestra la información del ideal y del nadir estimados para la instancia del problema SPS seleccionada, lo que proporciona al decisor información de interés sobre el rango de variación de cada objetivo. Así, haciendo uso del campo *selector*, o bien introduciendo directamente el valor deseado sobre la caja de texto, el DM puede determinar los niveles de aspiración para cada objetivo (*cost* y *makespan*).

Suponiendo que el decisor dispone de poco conocimiento sobre el problema para determinar sus preferencias inicialmente, los valores de referencia de cada objetivo se fijan

por defecto al nadir. Así, la región de interés definida por el punto de referencia cubre todo el espacio de objetivos. Esto permite al DM visualizar una aproximación inicial del frente óptimo de Pareto y, por consiguiente, obtener conocimiento sobre la distribución de soluciones en el espacio de objetivos.

4) Soluciones no dominadas

En esta rejilla se muestra el conjunto de soluciones no dominadas obtenidas por el algoritmo seleccionado para la instancia del problema SPS especificada. La primera columna es una etiqueta indentificativa de cada solución, mientras que las dos columnas restantes contienen el valor del coste y de la duración del proyecto, respectivamente.

5) Proceso de resolución

Este panel contiene tres botones que permiten ejecutar el algoritmo escogido sobre la instancia del problema SPS seleccionada, dar una nueva iteración partiendo de la aproximación de la iteración anterior o finalizar la aplicación.

iSPS hace uso de una carpeta llamada *iSPSStudy/SPS/instance*, siendo *instance* el nombre de la instancia seleccionada, donde almacena información relevante para el DM. En concreto, guarda, para cada iteración, el conjunto de soluciones obtenidas, las variables asociadas a éstas y el punto de referencia usado.

El botón *Start* ejecuta el algoritmo seleccionado sobre una instancia concreta, partiendo de un conjunto de soluciones iniciales aleatorias. Al finalizar su ejecución, se crean los archivos *solutions_in_1.txt*, *non_dominated_solutions_in_1.txt*, *variables_in_1.txt* y *reference_points.txt*. Éstos, contienen el conjunto de soluciones obtenidas tras la finalización del algoritmo, el conjunto de soluciones no dominadas, las variables de decisión asociadas a cada solución y el punto de referencia utilizado, respectivamente. Además, tanto el conjunto de soluciones no dominadas como el punto de referencia usado son representados gráficamente.

Con el botón *Next iteration* se hace uso de la población resultante en la iteración anterior, cargando la nueva configuración del algoritmo seleccionado y la información del punto de referencia, en caso de haber sido modificada. Tras la ejecución de la nueva iteración, se crean los ficheros correspondientes asociados a esta iteración. Además, el conjunto de soluciones resultante se representa gráficamente junto al conjunto de soluciones no do-

minadas de la iteración anterior y al punto de referencia usado en la iteración actual. De esta forma el DM puede comparar visualmente la mejora obtenida respecto a la iteración previa.

6) Gráfico

Esta zona de la interfaz gráfica de usuario se emplea para representar el conjunto de soluciones no dominadas obtenidas en la iteración actual y en la inmediatamente anterior, tras la ejcución del algoritmo seleccionado. Además, también muestra el punto de referencia empleado en cada iteración.

7) Registro de acciones

Este componente tiene como finalidad mostrar información, a modo de histórico, sobre cada una de las acciones realizadas por el usuario.

4.4 Ejemplo práctico de uso de iSPS

Con la finalidad de mostrar el proceso natural de uso de la aplicación por parte del DM, se describe, paso a paso, una posible secuencia de acciones realizadas durante el proceso de resolución de la instancia *8 employees and 64 tasks*, usando el algoritmo iWASFGA. Suponemos que el decisor no tiene, a priori, un conocimiento exhaustivo sobre el problema seleccionado.

Paso 1: configuración del algoritmo y punto de referencia

Tanto la configuración del algoritmo como el punto de referencia seleccionado se muestran en la Figura 4.4. Como el DM no posee conocimiento previo sobre la instancia escogida, el punto de referencia usado es el determinado por defecto, que corresponde al nadir de la instancia seleccionada.

Paso 2: ejecución inicial

Tras pulsar sobre el botón *Start*, iWASFGA genera, a partir de una población inicial aleatoria, un conjunto de soluciones que dominan al punto de referencia. La aproximación obtenida se muestra en la Figura 4.5.

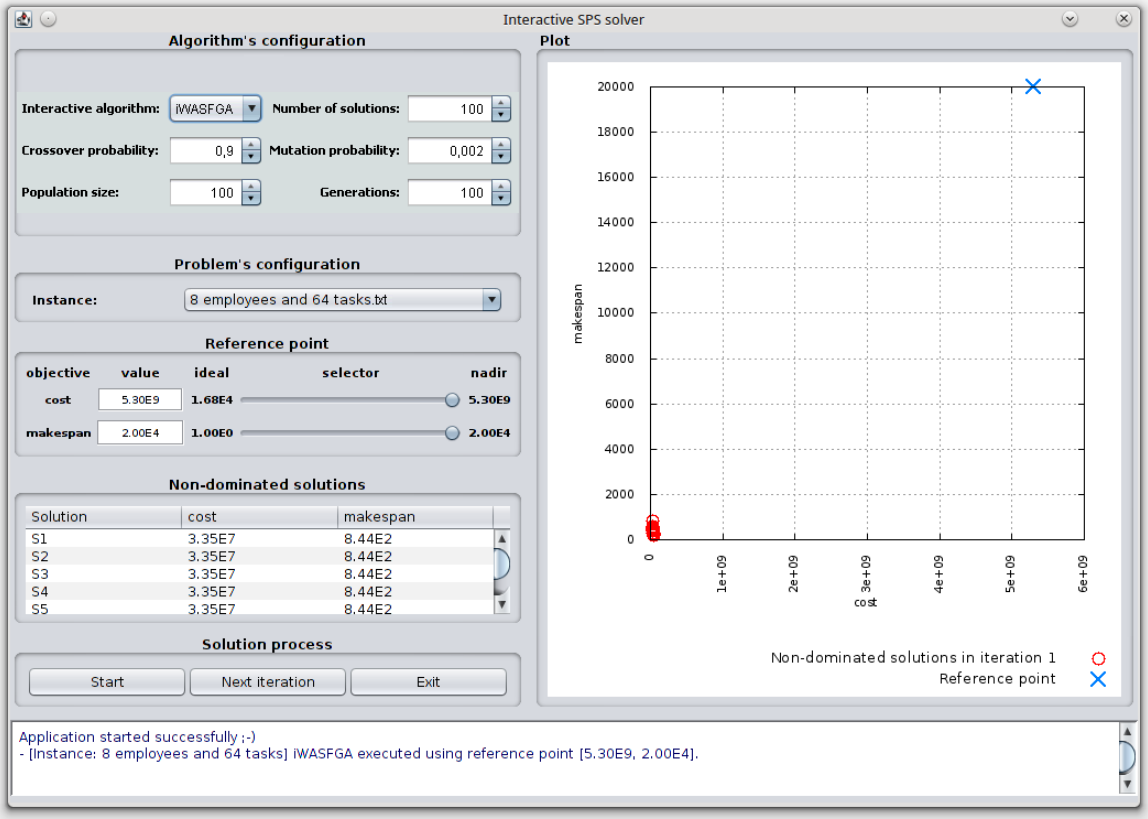


Figura 4.5: Ejecución inicial de iWASFGA en la instancia *8 employees and 64 tasks*.

Paso 3: iteración 1

En el Paso 2 el DM obtiene una primera aproximación de la distribución de soluciones en el espacio de objetivos. Según esto, ya dispone de cierta información para incorporar sus preferencias. Supongamos que el decisor no está interesado en aquellas soluciones donde $cost > 3.35 \cdot 10^7$ y, además, prefiere que $makespan < 1000$. Según esto, el punto de referencia se define a los valores anteriores y, al pulsar sobre el botón *Next iteration*, se obtiene una primera aproximación de la región de interés definida por el punto de referencia $\mathbf{q} = (3.35 \cdot 10^7, 1000)$. Esto puede apreciarse en la Figura 4.6. Las soluciones en gris representan a las soluciones no dominadas en la iteración anterior.

Paso 4: iteración 2

En el Paso 3 el DM ha obtenido un conjunto de soluciones que cumplen sus preferencias. En este momento, decide darle más tiempo al algoritmo para obtener una mejor aproximación de la región de interés definida por el punto de referencia. Para ello, aumenta el número de

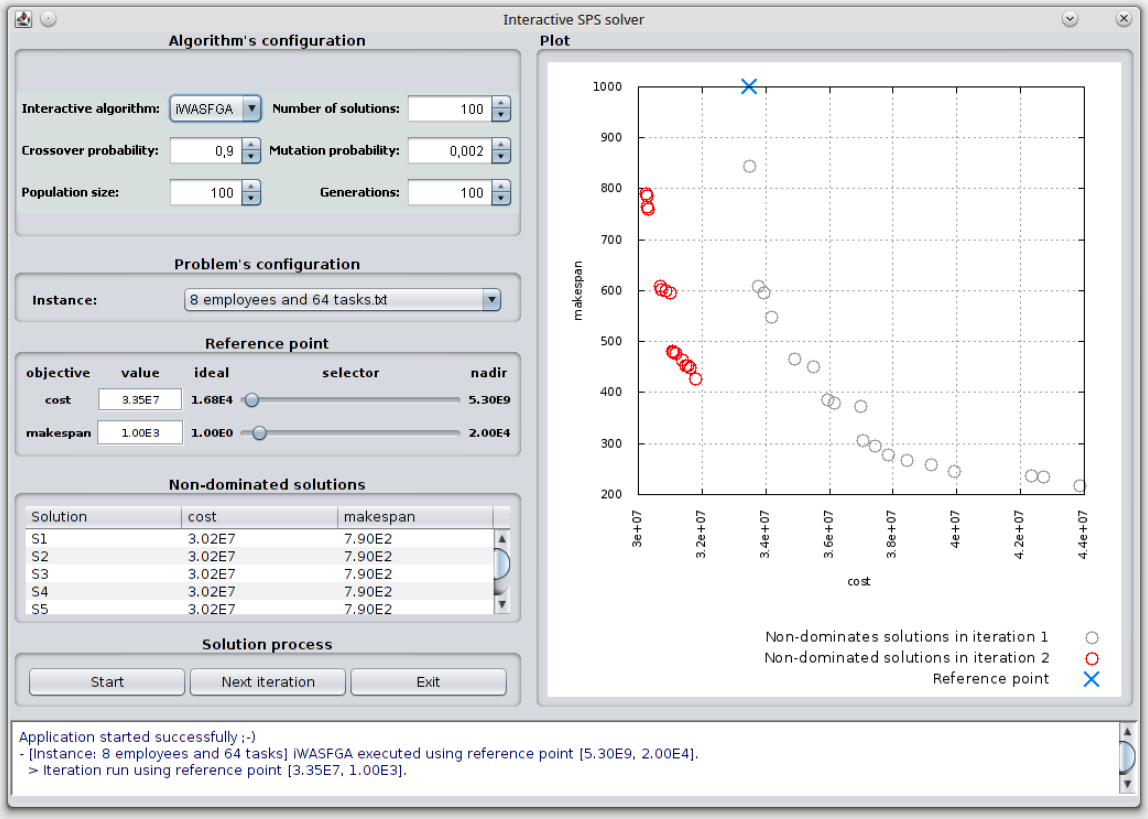


Figura 4.6: Primera iteración de iWASFGA en la instancia *8 employees and 64 tasks*.

generaciones a 1000 y pulsa nuevamente sobre el botón *Next iteration*. La aproximación resultante se muestra en la Figura 4.7.

Paso 5: iteración 3

En el Paso 4 las soluciones generadas mejoran a las obtenidas en la iteración anterior. No obstante, éstas pueden no ser Pareto óptimas o eficientes, aunque sí son no dominadas entre ellas. En cualquier caso, el DM desea un número reducido de soluciones, ya que decidir qué solución es la mejor entre múltiples soluciones requiere un gran esfuerzo cognitivo. Por este motivo, el DM realiza una nueva iteración, pero fijando en iWASFGA el número máximo de soluciones deseadas a cinco. Como puede observarse en la Figura 4.8, se aproxima la región de interés con tres soluciones bien distribuidas.

El conjunto de soluciones generadas durante el proceso anterior completo se almacena en disco en la carpeta de trabajo *iSPSStudy* de la aplicación. A partir de este punto, el

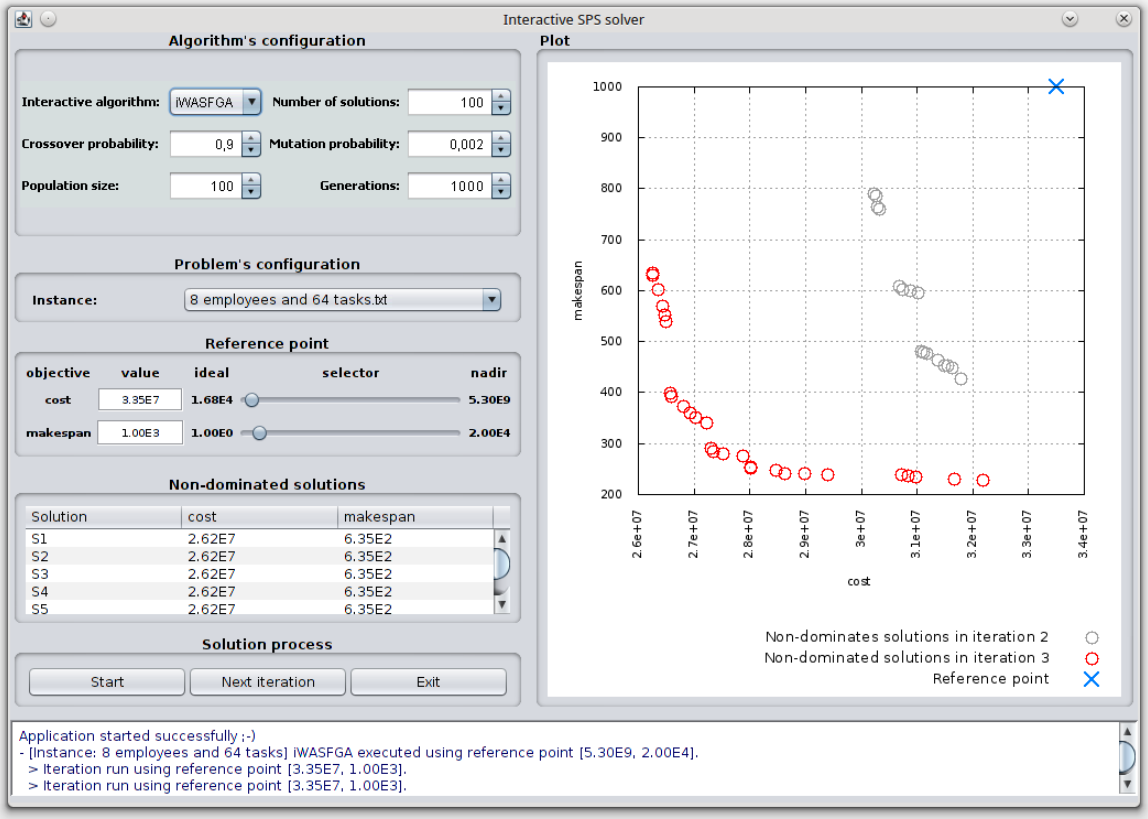


Figura 4.7: Segunda iteración de iWASFGA en la instancia *8 employees and 64 tasks*.

DM podría intentar mejorar las soluciones obtenidas, modificar sus preferencias o incluso comenzar el proceso usando otro algoritmo.

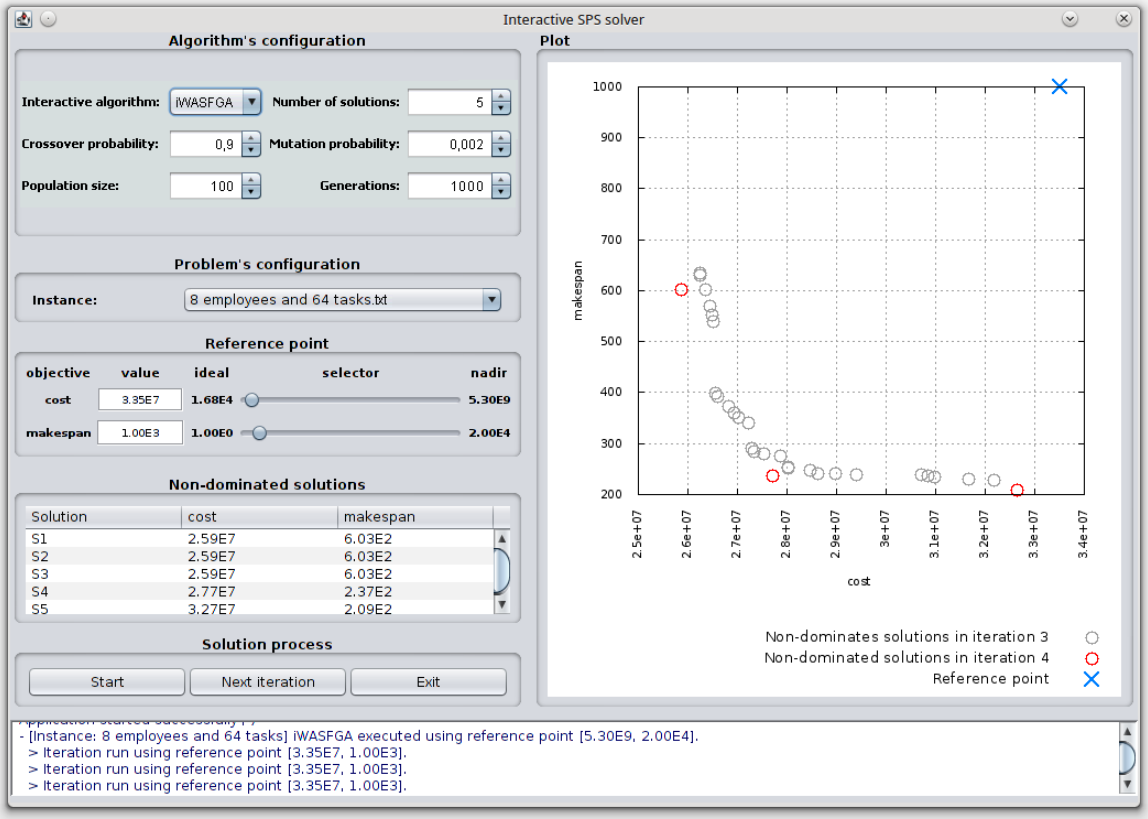


Figura 4.8: Tercera iteración de iWASFGA en la instancia *8 employees and 64 tasks*.

Capítulo 5

Estudio experimental

En este capítulo, con la finalidad de validar la utilidad de algoritmos de optimización multi-objetivo basados en preferencias en la resolución del problema SPS, se realiza una fase de experimentación sobre una instancia concreta del problema SPS. Para ello se realiza la comparativa de diferentes algoritmos genéticos de optimización multi-objetivo basados en preferencias, con NSGA-II [10], un algoritmo genético de optimización multi-objetivo que ha demostrado un buen funcionamiento en el problema SPS.

NSGA-II, es un algoritmo de optimización multi-objetivo muy popular en la literatura EMO. Éste, permite obtener un conjunto de soluciones no dominadas que aproximan el frente óptimo de Pareto del problema resuelto, pero no hace uso de información preferencial, por lo que no es posible centrar la búsqueda en una región concreta del espacio de objetivos.

Parece interesante comprobar si los algoritmos basados en preferencias permiten generar una mejor aproximación de la región de interés, en comparación a algoritmos que aproximan el frente Pareto óptimo de forma global, como NSGA-II. Para ello, se realiza un experimento donde los algoritmos g-NSGA-II, P-MOGA, WASF-GA y NSGA-II son comparados en la resolución de una instancia concreta[1] del problema SPS, con 8 empleados y 64 tareas. Todos lo algoritmos anteriores han sido implementados por el autor, excepto NSGA-II, que está disponible en el *framework* de jMetal.

[1]Definida de forma automática por un software generador de instancias, según se indica en [2]. Para ello, se establecen el número de empleados, sus habilidades, el número de tareas y el grafo de precedencias.

5.1 Configuración del experimento

Los cuatro algoritmos comparados hacen uso de operadores de selección, de recombinación y de mutación. En todos los casos, se han utilizado los operadores *BinaryTournament*, *TwoPointsCrossover* y *RandomMutation*, respectivamente. La probabilidad de recombinación se ha fijado a 0.9 y la de mutación a $1/n$, siendo n el número variables de decisión del problema a resolver. El tamaño de la población es de 100 individuos en cada algoritmo y el criterio de parada se ha establecido a 100000 evaluaciones de la función objetivo.

Tomando el papel del DM (gestor de proyectos software), se han determinado aleatoriamente dos puntos de referencia: uno alcanzable, $\mathbf{q}^a = (2.87 \cdot 10^7, 400)$, y otro inalcanzable, $\mathbf{q}^u = (2.7 \cdot 10^7, 200)$.

Dado que estos algoritmos tienen un comportamiento estocástico, se han realizado 30 ejecuciones independientes para obtener un valor medio representativo de sus rendimientos, haciendo uso de la métrica basada en hipervolumen definida en la Sección 3.3.1.

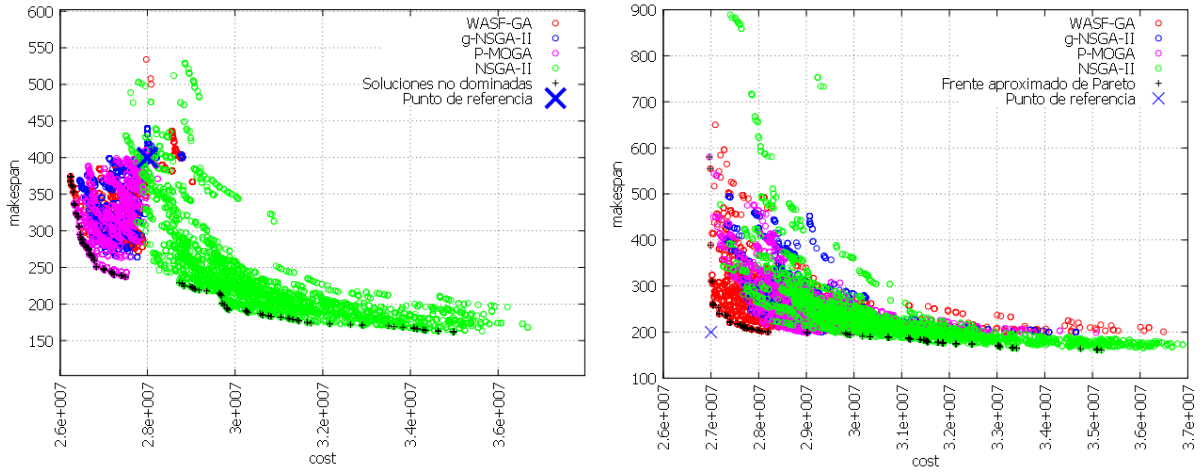
5.2 Resultados experimentales

La Tabla 5.1 recoge, para punto de referencia alcanzable e inalcanzable, respectivamente, el $HV_{\mathbf{q}}(P)$ medio en tanto por uno y su desviación típica. Donde P es el conjunto de soluciones no dominadas de cada ejecución y algoritmo. Se representa en negrita el valor de hipervolumen del algoritmo que obtiene una mejor aproximación de la región de interés.

Tabla 5.1: Media y desviación típica del $HV_{\mathbf{q}}$ para cada algoritmo, en 30 ejecuciones.

$\mathbf{q}$	WASF-GA	g-NSGA-II	P-MOGA	NSGA-II
$\mathbf{q}^a$	$1.81 \cdot 10^{-1}_{(1.96 \cdot 10^{-1})}$	$2.23 \cdot 10^{-1}_{(1.99 \cdot 10^{-1})}$	$\mathbf{2.81 \cdot 10^{-1}_{(2.46 \cdot 10^{-1})}}$	$3.95 \cdot 10^{-3}_{(1.40 \cdot 10^{-2})}$
$\mathbf{q}^u$	$\mathbf{8.91 \cdot 10^{-1}_{(7.02 \cdot 10^{-2})}}$	$8.09 \cdot 10^{-1}_{(6.47 \cdot 10^{-2})}$	$8.03 \cdot 10^{-1}_{(7.32 \cdot 10^{-2})}$	$7.59 \cdot 10^{-1}_{(7.93 \cdot 10^{-2})}$

La Figura 5.1 muestra el conjunto de soluciones obtenidas en cada ejecución para los distintos algoritmos, para punto de referencia alcanzable e inalcanzable, respectivamente. Como puede observarse, para los algoritmos g-NSGA-II, P-MOGA y WASF-GA la mayoría de las soluciones obtenidas se encuentran en la región de interés.



(a) Punto de referencia alcanzable (b) Punto de referencia inalcanzable

Figura 5.1: Soluciones obtenidas en 30 ejecuciones.

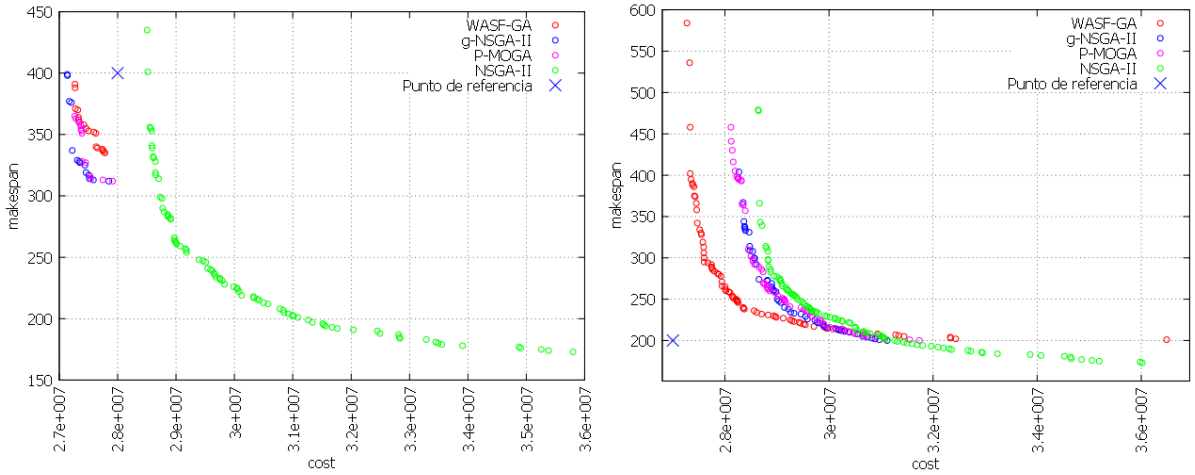
Para observar el rendimiento esperado de un algoritmo, así como su variabilidad, se puede hacer uso de la *empirical attainment function* (EAF) [20]. EAF es una función α que estima para una solución, la probabilidad de ser dominada por la aproximación del frente de Pareto de una ejecución individual de un algoritmo concreto. A partir de a aproximaciones del frente de Pareto, obtenidas en a ejecuciones, la EAF se define como:

$$\alpha(\mathbf{z}) = \frac{1}{a}\sum_{i=1}^{a} I(A^i \preceq \mathbf{z}), \tag{5.1}$$

donde A^i es la aproximación del frente de Pareto obtenida por el algoritmo en la ejecución número i e I es una función lógica que vale 1 cuando su predicado es cierto, y 0 en caso contrario. El predicado $A^i \preceq \mathbf{z}$ significa que A^i Pareto domina a $\mathbf{z}$.

A partir de la EAF, es posible definir el concepto de $k\,\%$-*attainment surface*, o $k\,\%$-AS, como la curva de nivel con valor $k/100$ para α. Según esto, se puede entender que $50\,\%$-AS es equivalente a la mediana del conjunto de frentes de Pareto aproximados por el algoritmo en cada ejecución. La Figura 5.2 muestra el $50\,\%$-AS de las 30 ejecuciones, para cada algoritmo y punto de referencia.

Según los resultados expuestos podemos afirmar que, para los puntos de referencia utilizados y considerando la métrica $HV_{\mathbf{q}}$, los algoritmos basados en preferencias obtienen una mejor aproximación de la región de interés en esta instancia del problema SPS. Atendiendo a la métrica $50\,\%$-AS, puede observarse que los algoritmos basados en preferencias consiguen superar, en promedio, el punto de referencia, mientras que NSGA-II no.



(a) Punto de referencia alcanzable (b) Punto de referencia inalcanzable

Figura 5.2: 50 %-*AS* en 30 ejecuciones.

Capítulo 6

Conclusión

El problema SPS es un problema de optimización combinatoria de asignación de recursos a tareas, cuyos objetivos son minimizar la duración y el coste de un proyecto. La planificación de proyectos software o SPS es una de las actividades de mayor importancia y, al mismo tiempo, de mayor complejidad en la dirección de proyectos de desarrollo software.

En este trabajo se ha realizado una propuesta software desarrollada en Java, denominada Interactive SPS o iSPS, que permite resolver, de forma interactiva, cualquier instancia del problema SPS, haciendo uso de diferentes algoritmos evolutivos basados en punto de referencia. Estos algoritmos tienen en cuenta las preferencias del decisor para aproximar una región concreta del frente óptimo de Pareto, definida indirectamente por el punto de refencia proporcionado por el usuario. La finalidad de la herramienta es ayudar al director de proyectos software en la toma de decisiones. Por este motivo la interfaz gráfica de usuario de iSPS es simple, pero ofrece todo lo necesario para cargar instancias del problema SPS, determinar el algoritmo a usar para su resolución incorporando preferencias mediante punto de referencia y visualizar los resultados obtenidos gráficamente.

Aparte de la herramienta software iSPS, se ha realizado un breve estudio para comprobar las prestaciones de algunos algoritmos evolutivos basados en preferencias sobre una instancia concreta del problema SPS. En concreto, los algoritmos g-NSGA-II, P-MOGA y WASF-GA se han comparado con NSGA-II, un algoritmo genético de optimización multi-objetivo que ha demostrado un buen funcionamiento en el problema SPS. Según los experimentos realizados y la métrica de rendimiento utilizada, basada en hipervolumen, concluimos que los algoritmos basados en preferencias permiten guiar la búsqueda de soluciones hacia una región concreta del espacio objetivo, lo que debe repercutir positiva-

mente en el tiempo de ejecución del algoritmo y en la calidad de la soluciones obtenidas. No obstante, únicamente se han realizado experimentos sobre una instancia concreta del problema SPS. Parece interesante repetir esta fase de experimentos sobre un conjunto más amplio de instancias y considerando un mayor número de puntos de referencia.

Indice de tablas

Indice de figuras

Bibliografía

[1] J. Branke, T. Kaussler, and H. Schemeck. Guidance in evolutionary multi-objective optimization. *Advances in Engineering Software*, 32:499–507, 2001.

[2] Francisco Chicano, Alejandro Cervantes, Francisco Luna, and Gustavo Recio. A novel multiobjective formulation of the robust software project scheduling problem. In *Proceedings of the 2012 European Conference on Applications of Evolutionary Computation*, EvoApplications'12, pages 497–507, Berlin, Heidelberg, 2012. Springer-Verlag.

[3] C. A. C. Coello, G. B. Lamont, and D.A.V. Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems. Second Edition*. Springer, New York, 2007.

[4] C.A.C. Coello. Handling Preferences in Evolutionary Multiobjective Optimization: A Survey. In *IEEE Congress on Evolutionary Computation*, pages 30–37, 2000.

[5] K. Deb. *Multi-objective Optimization using Evolutionary Algorithms*. Wiley, Chichester, 2001.

[6] K. Deb and A. Kumar. Interactive Evolutionary Multi-Objective Optimization and Decision-Making using Reference Direction Method. In *9th Annual Conference on Genetic and Evolutionary Computation (GECCO-2007)*, pages 781–788, 2007.

[7] K. Deb and A. Kumar. Light beam search based multi-objective optimization using evolutionary algorithms. In *IEEE Congress on Evolutionary Computation (CEC-2007)*, pages 2125–2132, 2007.

[8] K. Deb and K. Miettinen. Nadir point estimation using evolutionary approaches: Better accuracy and computational speed through focused search. In M. Ehrgott,

B. Naujoks, T.J. Stewart, and J. Wallenius, editors, *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems*, pages 339–354, Berlin, Heidelberg, 2010. Springer-Verlag.

[9] K. Deb., K. Miettinen, and S. Chaudhuri. Towards an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches. *IEEE Transactions on Evolutionary Computation*, 14(6):821–841, 2010.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[11] K. Deb, J. Sundar, B. Ubay, and S. Chaudhuri. Reference point based multi-objective optimization using evolutionary algorithm. *International Journal of Computational Intelligence Research*, 2(6):273–286, 2006.

[12] Juan J. Durillo and Antonio J. Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760 – 771, 2011.

[13] M. Ehrgott and D.Tenfelde-Podehl. Computation of ideal and nadir values and implications for their use in MCDM methods. *European Journal of Operational Research*, 151:119–139, 2003.

[14] Carlos M. Fonseca and Peter J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms-part i: A unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 28:26–37, 1998.

[15] C.M. Fonseca and P.J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In *5th International Conference on Genetic Algorithms*, pages 416–423, 1993.

[16] C.M. Fonseca, L. Paquete, and M. Lopez-Ibanez. An improved dimension-sweep algorithm for the hypervolume indicator. In *IEEE Congress on Evolutionary Computation (CEC-2006)*, pages 1157 – 1163, 2006.

[17] M. Gong, F. Liu, W. Zhang, L. Jiao, and Q. Zhang. Interactive MOEA/D for multi-objective decision making. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 721–728, 2011.

[18] C. L. Hwang and A. S. M. Masud. *Multiple Objective Decision Making – Methods and Applications: A State-of-the-Art Survey*. Springer-Verlag, Berlin, 1979.

[19] A. Jaszkiewicz and R. Slowiński. The 'Light Beam Search' Approach – An Overview of Methodology and Applications. *European Journal of Operational Research*, 113(2):300–314, 1999.

[20] J. Knowles. A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers. In *Intelligent Systems Design and Applications, 2005. ISDA '05. Proceedings. 5th International Conference on*, pages 552–557, Sept 2005.

[21] J. D. Knowles and D. W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[22] P. Korhonen and J. Laakso. A visual interactive method for solving the multiple criteria problem. *European Journal of Operational Research*, 24(2):277–287, 1986.

[23] Francisco Luna, J. Francisco Chicano, and Enrique Alba. Robust solutions for the software project scheduling problem: a preliminary analysis. *IJMHeur*, 2(1):56–79, 2012.

[24] M. Luque, K. Miettinen, P. Eskelinen, and F. Ruiz. Incorporating preference information in interactive reference point methods for multiobjective optimization. *Omega*, 37(2):450–462, 2009.

[25] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, 1999.

[26] K. Miettinen and M.M. Mäkelä. On scalarizing functions in multiobjective optimization. *OR Spectrum*, 24(2):193–213, 2002.

[27] K. Miettinen, M.M. Mäkelä, and K. Kaario. Experiments with classification-based scalarizing functions in interactive multiobjective optimization. *European Journal of Operational Research*, 175(2):931–947, 2006.

[28] J. Molina, L. V. Santana, A. G. Hernandez-Diaz, C. A. Coello, and R. Caballero. g-dominance: Reference point based dominance for multiobjective metaheuristics. *European Journal of Operational Research*, 197:685–692, 2009.

[29] L. Rachmawati and D. Srinivasan. Preference Incorporation in Multiobjective Evolutionary Algorithms. In *IEEE Congress on Evolutionary Computation*, pages 3385–3391, 2006.

[30] Rafael Rodríguez, Mariano Luque, and Mercedes González. Portfolio selection in the spanish stock market by interactive multiobjective programming. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 19(1):213–231, 2011.

[31] A.B. Ruiz, R. Saborido, and M. Luque. A preference-based evolutionary algorithm for multiobjective optimization: The weighting achievement scalarizing function genetic algorithm. *Journal of Global Optimization*, in press, 2014.

[32] F. Ruiz, M. Luque, and J.M. Cabello. A classification of the weighting schemes in reference point procedures for multiobjective programming. *Journal of the Operational Research Society*, 60(4):544–553, 2009.

[33] L. Ben Said, S. Bechikh, and K. Ghedira. The r-dominance: A new dominance relation for interactive evolutionary multcriteria decision making. *IEEE Transactions on Evolutionary Computation*, 14(5):801–818, 2010.

[34] Soo-Yong Shin, In-Hee Lee, Dongmin Kim, and Byoung-Tak Zhang. Multiobjective evolutionary optimization of dna sequences for reliable dna computing. *Trans. Evol. Comp*, 9(2):143–158, April 2005.

[35] A. Sinha, P.J. Korhonen, J. Wallenius, and K. Deb. An Interactive Evolutionary Multi-objective Optimization Method Based on Polyhedral Cones. In C. Blum and R. Battiti, editors, *Learning and Intelligence Optimization*, volume 6073 of *Lecture Notes in Computer Science*, pages 318–332. Springer, 2010.

[36] W. Stadler. A survey of multicriteria optimization or the vector maximum problem, part i: 1776–1960. *Journal of Optimization Theory and Applications*, 29:1–52, 1979. 10.1007/BF00932634.

[37] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application.* John Wiley, New York, 1986.

[38] M. Szczepanski and A. P. Wierzbicki. Application of multiple criterion evolutionary algorithm to vector optimization, decision support and reference point approaches. *Journal of Telecommunications and Information Technology*, 3:16–33, 2003.

[39] L. Thiele, K. Miettinen, P. Korhonen, and J. Molina. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary Computation*, 17(3):411–436, 2009.

[40] K. Weinert, A. Zabel, P. Kersting, T. Michelitsch, and T. Wagner. On the use of problem-specific candidate generators for the hybrid optimization of multi-objective production engineering problems. *Evol. Comput.*, 17(4):527–544, December 2009.

[41] A. P. Wierzbicki. Basic properties of scalarizing functionals for multiobjective optimization. *Mathematische Operationsforschung und Statistik*, 8:55–60, 1977.

[42] A. P. Wierzbicki. The use of reference objectives in multiobjective optimization. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making, Theory and Applications*, pages 468–486. Springer-Verlag, Berlin, 1980.

[43] Q. Zhang and H. Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[44] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In K. C. Giannakoglou et al., editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100. International Center for Numerical Methods in Engineering (CIMNE), 2002.

[45] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms - A comparative case study. In A. Eiben, T. Bäck, M. Schoenauer, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–301. Springer-Verlag, Berlin, 1998.