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

Rubén Saborido y Francisco Chicano

Resumen— **En este trabajo se presenta una herramienta software, denominada Interactive SPS o iSPS, que 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 mostrar la utilidad de este enfoque preferencial sobre el problema SPS, se ilustra su aplicación con un ejemplo paso a paso en el que se aplica el algoritmo iWASFGA a una instancia concreta del problema.**

Palabras clave— **Planificación de Proyectos Software, herramienta software, optimización multi-objetivo, toma de decisiones multi-criterio.**

I. Introducción

Muchos problemas de la vida real requieren optimizar varios criterios u objetivos, como por ejemplo tareas complejas en el dominio de la Ingeniería [1], la Bioinformática [2] o la Economía [3]. Este tipo de problemas se denominan *problemas de optimización muti-objetivo* (MOP) [4]. 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 [5], que define un conjunto de soluciones óptimas donde ninguno de los valores objetivos puede ser mejorado sin empeorar al menos uno de los restantes. A este conjunto de soluciones se le denomina *conjunto óptimo de Pareto* y a su imagen en el espacio de objetivos *frente óptimo de Pareto*.

Entre las metodologías existentes para resolver un problema de optimización multi-objetivo, los dominios de *Multiple Criteria Decision Making* (MCDM) [6] y *Evolutionary Multi-objective Optimization* (EMO) [7] han contribuido con numerosas técnicas, aplicadas con éxito en problemas reales.

École Polytechnique de Montréal. E-mail: ruben.saborido-infantes@polymtl.ca.
Universidad de Málaga. E-mail: chicano@lcc.uma.es.

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.

Desde finales de la década de los 1990, hay cierto interés en hibridar la *Toma de Decisiones Multi-criterio* y los *Algoritmos Evolutivos* en la resolución de problemas de optimización multi-objetivo. 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* [8]. Existen varios enfoques de algoritmos EMO basados en preferencias. Algunos de ellos hacen uso de conceptos e ideas extraídas del ámbito MCDM [9], [10], [11], [12], otros modifican métodos existentes, como el *Non-dominated Sorting Genetic Algorithm* (NSGA-II) [13], para incorporar preferencias [9], [10], [14] y algunos modifican o complementan la relación de dominancia de Pareto [15], [16].

El *Problema de Planificación de Proyectos Software* (SPS, de *Software Project Scheduling*) es un problema de optimización combinatoria de asignación de personas a tareas, cuyos objetivos son minimizar la duración y el coste de un proyecto. Las personas tienen asociadas un salario, una dedicación y unas habilidades personales que les 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 (TPG, de *Task Precedence Graph*), lo que añade ciertas restricciones al problema. La planificación de proyectos software 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 los últimos años, con el uso de metaheurísticas se ha conseguido abordar problemas complejos de este tipo. Este problema ha sido resuelto en el pasado con técnicas metaheurísticas multi-objetivo [17].

En este trabajo, se presenta una herramienta software desarrollada en Java, denominada *Interactive SPS* (iSPS), que permite resolver, de forma interac-

tiva, 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 muestra un ejemplo práctico de uso de la aplicación sobre una instancia concreta del problema SPS.

El resto del artículo se organiza como sigue. En la Sección II se introducen los fundamentos teóricos en que está basada la herramienta. En la Sección III se describe el problema de Planificación de Proyectos Software. A continuación se detalla la herramienta desarrollada en la Sección IV y se muestra su utilidad con un ejemplo en la Sección V. Finalmente, se concluye el artículo en la Sección VI.

II. Fundamentos

En esta sección vamos a introducir algunos conceptos básicos de optimización multi-objetivo y algoritmos basados en preferencias.

A. Optimización Multi-objetivo

Un problema de optimización multi-objetivo se define matemáticamente como

$$\begin{array}{rl} \text{minimizar} & \mathbf{f}(x) = (f_1(x), f_2(x), \ldots, f_k(x))^T \\ \text{sujeto a} & x \in S \end{array} \tag{1}$$

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

Dados $x, x' \in S$, se dice que 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 soluciones no dominadas (eficientes) se conoce como *conjunto óptimo de Pareto*, mientras que la imagen por $\mathbf{f}$ de este conjunto define el *frente óptimo de Pareto*.

B. Decisión basada en preferencias

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 [18]. 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* [19]. 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*.

Una forma de medir la calidad de la aproximación de la región de interés teniendo en cuenta el punto de referencia es la propuesta en [20], basada a su vez en el *hipervolumen* [21]. 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 $y \in P$, se construye un hipercubo v_y acotado por un punto de referencia $\mathbf{R}$ y la propia solución y. El punto de referencia $\mathbf{R}$ puede fijarse al vector nadir.[1]

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

$$HV(P) = volumen\left(\bigcup_{i=1}^{|P|} v_i\right). \tag{2}$$

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:

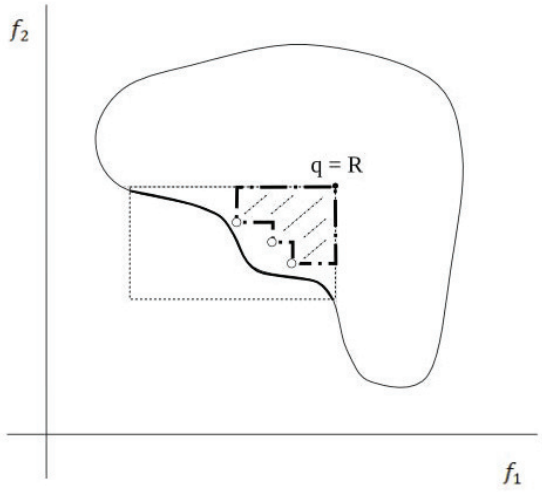
- Si $\mathbf{q}$ es alcanzable, entonces $\mathbf{R} = \mathbf{q}$.
- Si $\mathbf{q}$ es inalcanzable, entonces $\mathbf{R}$ se define como sigue. Supongamos que A es un frente de referencia, el cual es una buena aproximación del frente óptimo de Pareto. Entonces, $R_i = \max_{\mathbf{x} \in \mathbf{A_q}} f_i(\mathbf{x})$ para todo $i = 1, \ldots, k$, donde $A_{\mathbf{q}}$ es el subconjunto de soluciones de A que son dominadas por $\mathbf{q}$.

La Figura 1 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 $HV_{\mathbf{q}}$.

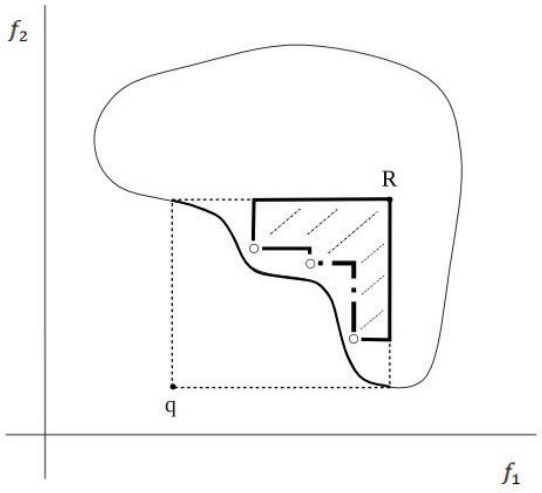
Los métodos de optimización multi-objetivo evolutivos basados en preferencias, pretenden incorporar las preferencias del decisor a un algoritmo evolutivo para aproximar la región de interés en el frente óptimo de Pareto. Esta información puede ser empleada para enfocar la búsqueda hacia la región del frente óptimo de Pareto de mayor interés, bajo el punto de vista del decisor. En este artículo nos centramos en un enfoque interactivo, donde el algoritmo interactúa con el decisor cada cierto número de generaciones para orientar gradualmente la búsqueda considerando sus preferencias.

Existen multitud de algoritmos EMO basados en preferencias. *Guided Multi-objective Evolutionary Algorithm* (G-MOEA) [22] modifica el criterio de dominancia para incorporar las preferencias del decisor mediante *trade-offs* aceptables. *Preference Based Evolutionary Algorithm* (PBEA) [23] usa un

[1] El punto o vector *nadir* es aquél que tiene como componentes los máximos de cada una de las funciones objetivos aplicadas al conjunto óptimo de Pareto.



(a)q alcanzable



(b)q inalcanzable

Fig. 1. Ilustración de la métrica $HV_{\mathbf{q}}$.

indicador de calidad basado en funciones escalarizadas de logro definidas para uno o varios puntos de referencia. En [12] proponen un EMO interactivo que se apoya en la generación de conos poliédricos, a partir de las preferencias proporcionadas por el decisor en cada interacción, para guiar el proceso de búsqueda de un algoritmo evolutivo. En [15] 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 decisor. Recientemente, en [16] 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) [14]. 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) [9] incorpora a NSGA-II una metodología extraída de MCDM denominada *dirección de referencia* [24]. A partir de un punto del espacio de objetivos y un punto de referencia proporcionado por el decisor 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* [25], ha sido utilizada en [10] con su integración en NSGA-II. *interactive MOEA/D* (iMOEA/D) es un enfoque interactivo de MOEA/D propuesto en [11]. Tras un número determinado de generaciones se muestra un conjunto de soluciones al decisor, 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.

III. El problema SPS

La formulación que usamos aquí del problema de planificación de proyectos software fue inicialmente presentada en [26]. Llamemos E 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): 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. Llamemos π al 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

$$done(\tau) = \left\{ t_j \in T \middle| \sum_{\tau'=0}^{\tau-1} \pi_j(\tau') \geq t_j^c \right\}, \quad (3)$$

y

$$active(\tau) = \{t_j \in T | \forall t_i, (t_i, t_j) \in A : t_i \in done(\tau - r_j)\} - done(\tau), \quad (4)$$

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:

$$\pi_j(\tau) = \begin{cases} \sum_{e_i \in E: \mathbf{Q}_{i,j} > 0} \frac{d_i \cdot \mathbf{P}_{i,j} \cdot \mathbf{Q}_{i,j}}{\sum_{t_k \in active(\tau)} \mathbf{Q}_{i,k}} & \text{si } t_j \in active(\tau), \\ 0 & \text{en otro caso.} \end{cases} \quad (5)$$

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:

$$makespan(x) = min\{\tau \in \mathbb{N} | done(\tau) = T\}. \quad (6)$$

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,

$$cost(x) = \sum_{e_i \in E} e_i^s \cdot d_i \cdot w_i, \quad (7)$$

con w_i definido como:

$$w_i = \left| \left\{ \tau \in \mathbb{N} | \exists t_j \in active(\tau) : \mathbf{Q}_{(i,j)} \cdot \mathbf{P}_{(i,j)} > 0 \right\} \right|. \quad (8)$$

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 vectorial objetivo

$$\mathbf{f}(x) = (cost(x), makespan(x)). \quad (9)$$

IV. Interactive SPS

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. Cuando el decisor no dispone de suficiente conocimiento previo sobre el problema, los métodos interactivos permiten adquirir, de forma iterativa, conocimiento sobre el mismo. Nuestra herramienta, denominada *Interactive SPS* (iSPS) 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.

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*.

A. Algoritmos integrados en iSPS

Entre los algoritmos EMO existentes basados en preferencias mediante punto de referencia, destacan g-NSGA-II [15], el algoritmo genético preferencial definido en [27], al que nos referiremos como P-MOGA, y WASF-GA [20]. Éstos permiten aproximar la región de interés del frente óptimo de Pareto considerando las preferencias del decisor. Todos ellos se han implementado dentro del *framework* jMetal [28], 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 [16]. 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 decisor 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 [29] 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 decisor, considerando un conjunto de vectores de pesos. El algoritmo pretende aproximar la región de interés con un conjunto de soluciones bien distribuida.

B. Arquitectura de iSPS

iSPS utiliza 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 manipulen la información suministrada por el decisor 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 con más detalle abajo. 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 2(a) muestra las principales clases 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.

C. Interfaz gráfica de usuario

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 2(b).

La interfaz se divide en 7 zonas bien diferenciadas, las cuales comentamos a continuación.

1) Configuración del algoritmo

Permite seleccionar el algoritmo a ejecutar para resolver el correspondiente problema SPS, así como modificar distintos parámetros del algoritmo escogido: número de soluciones, tamaño de la población, número de generaciones, probabilidad de recombinación y probabilidad de mutación.

2) Configuración del problema

Permite seleccionar la instancia del problema SPS a resolver. Por defecto se ofrecen seis instancias de entre 8 y 32 empleados y entre 64 y 512 tareas. Las instancias fueron generadas automáticamente con el generador de instancias del problema SPS usado en [26]. 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. El usuario puede añadir nuevas instancias. Para ello, basta copiar los archivos de las instancies en la carpeta anterior.

3) Punto de referencia

Este panel se usa para especificar las preferencias del decisor. Muestra la información de los puntos ideal[2] y 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 decisor puede determinar los niveles de aspiración para cada objetivo.

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 decisor 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 identificativa de cada solución, mientras que las dos columnas restantes contienen el valor del coste y de la duración del proyecto, respectivamente.

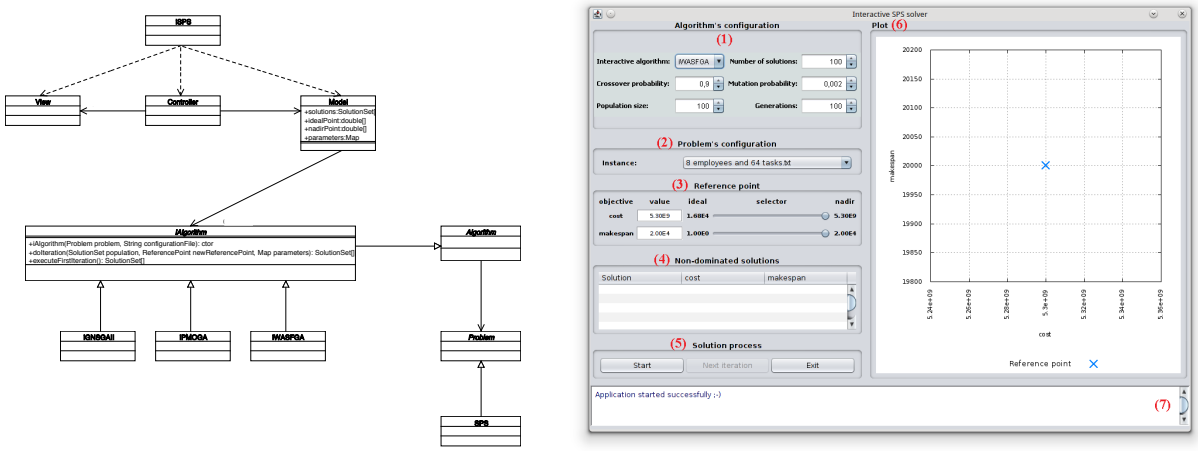
5) Proceso de resolución

Este panel contiene tres botones que permiten ejecutar el algoritmo escogido sobre la instancia del problema SPS seleccionada (*Start*), dar una nueva iteración partiendo de la aproximación de la iteración anterior (*Next iteration*) o finalizar la aplicación (*Exit*). Tras cada iteración, iSPS guarda en ficheros el conjunto de soluciones obtenidas, las variables asociadas a éstas y el punto de referencia usado. Además, tanto el conjunto de soluciones no dominadas como el punto de referencia usado son representados gráficamente. También se representa el conjunto de soluciones no dominadas de la iteración anterior para que el decisor pueda comparar visualmente la mejora obtenida.

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 junto con el punto de referencia empleado en cada iteración.

[2]El punto o vector *ideal* es aquél que tiene como componentes los mínimos de cada una de las funciones objetivos aplicadas al conjunto óptimo de Pareto.



(a)Arquitectura software de iSPS.

(b)Interfaz gráfica de usuario de iSPS.

Fig. 2. Arquitectura y GUI de la aplicación.

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.

V. Ejemplo práctico de uso de iSPS

Con la finalidad de mostrar el proceso natural de uso de la aplicación por parte del decisor, 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 2(b). Como el decisor 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 3.

Paso 3: iteración 1

En el paso 2 el decisor 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

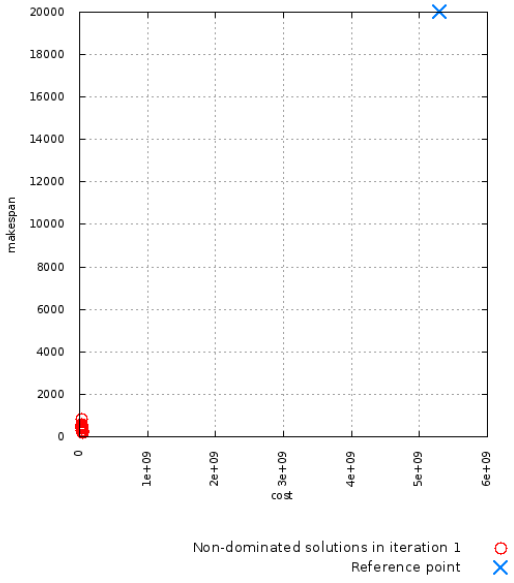


Fig. 3. Ejecución inicial de iWASFGA en la instancia *8 employees and 64 tasks*.

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. Las soluciones en gris representan a las soluciones no dominadas en la iteración anterior.

Paso 4: iteración 2

En el paso 3 el decisor 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 generaciones a 1000 y pulsa nuevamente sobre el botón *Next iteration*. La aproximación resultante se muestra en la Figura 5.

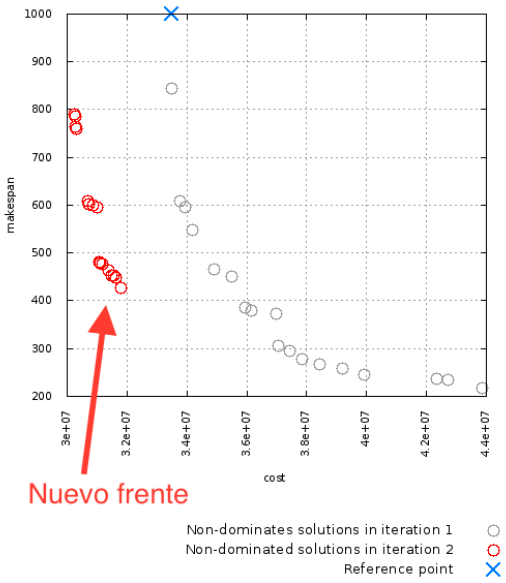


Fig. 4. Primera iteración de iWASFGA en la instancia *8 employees and 64 tasks*.

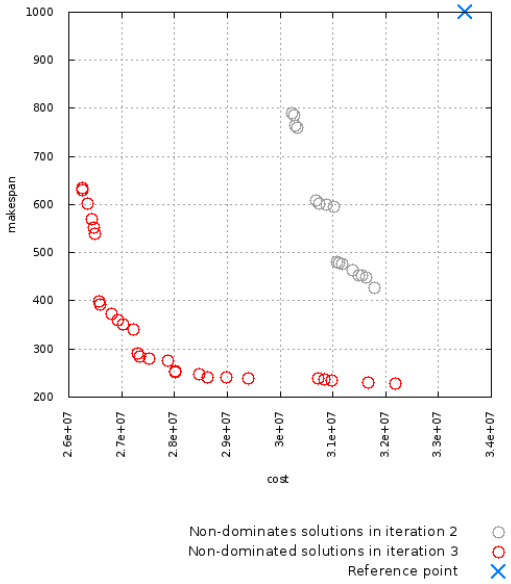


Fig. 5. Segunda iteración de iWASFGA en la instancia *8 employees and 64 tasks*.

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 decisor 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 decisor realiza una nueva iteración, pero fijando en iWASFGA el número máximo de soluciones deseadas a cinco. Como puede observarse en la Figura 6, se aproxima la región de interés con tres soluciones bien distribuidas.

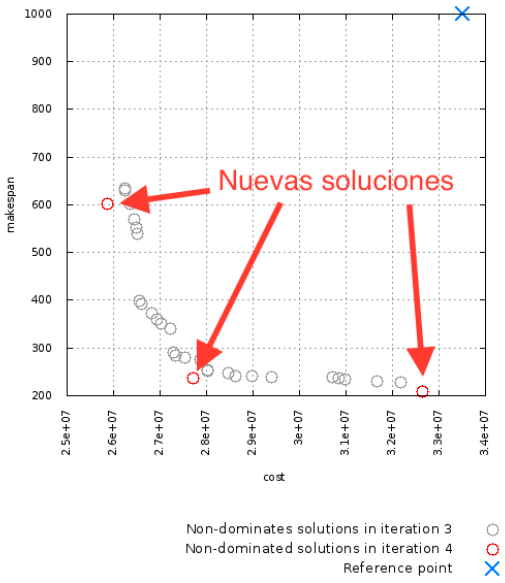


Fig. 6. Tercera iteración de iWASFGA en la instancia *8 employees and 64 tasks*.

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

VI. Conclusiones

El problema de planificación de proyectos software es un problema de optimización combinatoria de asignación de recursos a tareas, cuyos objetivos son minimizar la duración y el coste de un proyecto. Se trata de 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.

Como trabajo futuro podría ampliarse la herramienta para darle mayor flexibilidad, permitiendo, por ejemplo, seleccionar distintos operadores de

variación. De la misma forma, debería mejorarse el procesamiento de ficheros relacionados con las instancias y soluciones, con el objetivo de que el usuario pueda escoger una ubicación arbitraria del sistema de ficheros para ambos.

Agradecimientos

Este trabajo ha sido financiado por la Universidad de Málaga, Andalucía Tech, el Ministerio de Ciencia e Innovación y FEDER con contrato TIN2011-28194 (proyecto roadME) y la Universidad Técnica de Ostrava con contrato OTRI 8.06/5.47.4142.

Referencias

[1] 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.*, vol. 17, no. 4, pp. 527–544, Dec. 2009.

[2] 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*, vol. 9, no. 2, pp. 143–158, Apr. 2005.

[3] 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*, vol. 19, no. 1, pp. 213–231, 2011.

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

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

[6] 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.

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

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

[9] 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)*, 2007, pp. 781–788.

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

[11] 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*, 2011, GECCO '11, pp. 721–728.

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

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[14] K. Deb, J. Sundar, B. Ubay, and S. Chaudhuri, "Reference point based multi-objective optimization using evolutionary algorithm," *International Journal of Computational Intelligence Research*, vol. 2, no. 6, pp. 273–286, 2006.

[15] 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*, vol. 197, pp. 685–692, 2009.

[16] 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*, vol. 14, no. 5, pp. 801–818, 2010.

[17] Francisco Luna, David L. González-Álvarez, Francisco Chicano, and Miguel A. Rodríguez, "The software project scheduling problem: A scalability analysis of multi-objective metaheuristics," *Applied Soft Computing*, vol. 15, pp. 136 – 148, Jan-02-2014 2014.

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

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

[20] 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*, vol. in press, 2014.

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

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

[23] L. Thiele, K. Miettinen, P. Korhonen, and J. Molina, "A preference-based evolutionary algorithm for multi-objective optimization," *Evolutionary Computation*, vol. 17, no. 3, pp. 411–436, 2009.

[24] P. Korhonen and J. Laakso, "A visual interactive method for solving the multiple criteria problem," *European Journal of Operational Research*, vol. 24, no. 2, pp. 277–287, 1986.

[25] A. Jaszkiewicz and R. Slowiński, "The 'Light Beam Search' Approach – An Overview of Methodology and Applications," *European Journal of Operational Research*, vol. 113, no. 2, pp. 300–314, 1999.

[26] 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*, Berlin, Heidelberg, 2012, EvoApplications'12, pp. 497–507, Springer-Verlag.

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

[28] Juan J. Durillo and Antonio J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760 – 771, 2011.

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