XIII Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (XIII MAEB)

MAEB 5: Sesión Especial: Algoritmos MultiObjetivo

Organizadores:

Enrique Alba y Mariano Luque

An Improvement Study of the Decomposition-based Algorithm Global WASF-GA for Evolutionary Multiobjective Optimization*

*Note: The full contents of this paper have been published in the volume *Lecture Notes in Artificial Intelligence 11160* (LNAI 11160)

Sandra Gonzalez-Gallardo, Rubén Saborido, Ana B. Ruiz, Mariano Luque
Department of Applied Economics (Mathematics)
University of Málaga
Málaga, Spain
{sandragg, rsain, abruiz, mluque}@uma.es

***Abstract*—The convergence and the diversity of the decomposition-based evolutionary algorithm Global WASF-GA (GWASF-GA) relies on a set of weight vectors that determine the search directions for new non-dominated solutions in the objective space. Although using weight vectors whose search directions are widely distributed may lead to a well-diversified approximation of the Pareto front (PF), this may not be enough to obtain a good approximation for complicated PFs (discontinuous, non-convex, etc.). Thus, we propose to dynamically adjust the weight vectors once GWASF-GA has been run for a certain number of generations. This adjustment is aimed at re-calculating some of the weight vectors, so that search directions pointing to overcrowded regions of the PF are redirected toward parts with a lack of solutions that may be hard to be approximated. We test different parameters settings of the dynamic adjustment in optimization problems with three, five, and six objectives, concluding that GWASF-GA performs better when adjusting the weight vectors dynamically than without applying the adjustment.**

***Index Terms*—Evolutionary multiobjective optimization, Decomposition-based algorithm, GWASF-GA, Weight vector**

Pruning Dominated Policies in Multiobjective Pareto Q-learning*

*Note: The full contents of this paper have been published in the volume *Lecture Notes in Artificial Intelligence 11160* (LNAI 11160)

Lawrence Mandow, José-Luis Pérez-de-la-Cruz
Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga
Málaga, Spain
{lawrence, perez}@lcc.uma.es

***Abstract*—The solution for a Multi-Objetive Reinforcement Learning problem is a set of Pareto optimal policies. MPQ-learning is a recent algorithm that approximates the whole set of all Pareto-optimal deterministic policies by directly generalizing Q-learning to the multiobjective setting. In this paper we present a modification of MPQ-learning that avoids useless cyclical policies and thus improves the number of training steps required for convergence.**

***Index Terms*—**

Estudio de escalabilidad de implementaciones multiobjetivo en problemas con muchos objetivos

Aurora Ramírez, Rafael Barbudo, José Raúl Romero, Sebastián Ventura
Dpto. Informática y Análisis Numérico, Universidad de Córdoba
{aramirez, rbarbudo, jrromero, sventura}@uco.es

***Resumen*—Los problemas de optimización de varios objetivos aparecen con frecuencia en aplicaciones reales, lo que ha llevado a la adaptación de la mayoría de las metaheurísticas existentes. El creciente interés en los problemas que presentan muchos objetivos también ha propiciado la aparición de nuevos métodos especializados. Para conseguir una mayor adopción de estas técnicas por parte de usuarios no familiarizados con el uso de metaheurísticas, es necesario que los nuevos desarrollos sean incluidos en librerías software que faciliten su configuración y adaptación. No obstante, la librería escogida puede repercutir en el rendimiento computacional, pues cada una sigue unos principios de diseño diferentes. Este trabajo analiza algunas de las implementaciones multiobjetivo disponibles en seis librerías con el propósito de estudiar su comportamiento frente a configuraciones de creciente complejidad y que incluyen problemas de hasta 50 objetivos. La experimentación realizada confirma que distintas implementaciones de un mismo algoritmo pueden presentar diferencias considerables en cuanto al tiempo de ejecución y la memoria consumida.**

***Index Terms*—Optimización de muchos objetivos, metaheurísticas, librerías software**

I. Introducción

Los problemas multiobjetivo han sido objeto frecuente de estudio en el campo de las metaheurísticas, pues son habitualmente aplicables en escenarios reales [1]. Este tipo de problemas requiere la definición de un conjunto de variables cuyo valor ha de determinarse, un conjunto de restricciones a cumplir y, al menos, dos objetivos independientes a optimizar [2]. La existencia de varios objetivos, a menudo contrapuestos, implica que la solución a un problema multiobjetivo no es única, sino que debe encontrarse un conjunto de soluciones donde cada una alcanza un compromiso distinto entre los objetivos. El estudio de enfoques bioinspirados para resolver problemas multiobjetivo ha dado lugar a algoritmos ya clásicos como NSGA-II y SPEA2, de los que es posible encontrar implementaciones en distintos lenguajes de programación [3].

Recientemente, la investigación en este campo se ha centrado en abordar la resolución de problemas con un número elevado de objetivos. Según la literatura, los denominados "problemas de muchos objetivos" (en inglés, *many-objective optimisation problems*) son aquellos que presentan cuatro o más objetivos [4]. Su resolución presenta nuevos retos, tales como el crecimiento exponencial del número de soluciones no dominadas o la ineficacia de operadores genéticos y técnicas de preservación de la diversidad [5]. De hecho, existen estudios que demuestran el bajo rendimiento de algoritmos clásicos a la hora de resolver este tipo de problemas, así como el incremento en el coste computacional [6]. Estos hechos han propiciado el desarrollo de métodos especializados [7], así como la realización de estudios comparativos [8].

A pesar de que algunas librerías software están incorporando algunos de los algoritmos para optimización de muchos objetivos que van adquiriendo mayor relevancia, lo cierto es que aún no existe un gran número de implementaciones disponibles. Este hecho no solo dificulta su uso con fines comparativos, sino que limita su aplicación por parte de usuarios con menos experiencia en el campo de las metaheurísticas. Por otro lado, características como el lenguaje de programación y el diseño estructural de estas librerías podría suponer ciertas diferencias en el uso de los recursos computacionales, como el tiempo de ejecución y la memoria, que también deben considerarse ante la creciente complejidad tanto de los problemas como de los algoritmos.

En este contexto, este trabajo presenta un estudio preliminar del rendimiento de algoritmos multiobjetivo implementados en diferentes librerías software. Tras realizar un análisis de las implementaciones disponibles, se han elegido seis librerías Java: ECJ, EvA, JCLEC-MO, jMetal, MOEA Framework y Opt4J. El objetivo de la comparación es por tanto analizar el rendimiento desde una perspectiva puramente computacional, realizando mediciones del tiempo de ejecución y la memoria RAM consumida. Para ello se han seleccionado varios algoritmos y problemas de entre los disponibles en las librerías. Además, se ha estudiado la escalabilidad de las distintas implementaciones de NSGA-II respecto al número de individuos, generaciones y objetivos. Los resultados muestran que existen pequeñas diferencias entre las implementaciones cuando se consideran parámetros estándar, y que éstas se incrementan a medida que se consideran valores más extremos.

El resto del artículo se estructura como sigue. La sección II describe los algoritmos y problemas más utilizados en optimización multiobjetivo, mientras que la sección III recopila las principales librerías software que los implementan. La sección IV detalla la metodología diseñada para el estudio comparativo, cuyos resultados son analizados en la sección V. Finalmente, la sección VI presenta las conclusiones.

II. Optimización con múltiples objetivos

Esta sección introduce los principales tipos de algoritmos multiobjetivo existentes, destacando aquellos especializados en

Trabajo financiado por los ministerios de Economía (TIN2017-83445-P) y de Educación (FPU13/01466), y fondos FEDER.

la optimización de muchos objetivos. Después se describen algunos de los problemas comúnmente utilizados para evaluar el rendimiento de estos algoritmos.

II-A. *Familias de algoritmos bioinspirados*

Desde la publicación de VEGA, considerado el primer algoritmo evolutivo multiobjetivo [2], la aparición de nuevos algoritmos o mejoras sobre los ya propuestos ha sido constante [9]. En los inicios del área, los algoritmos iban siendo clasificados en generaciones [10]. Así, la conocida como *primera generación* se basa fuertemente en el concepto de dominancia y el uso de estrategias de nichos. La incorporación de mecanismos de elitismo es la principal característica de la *segunda generación*, a la que pertenecen SPEA2 y NSGA-II.

En el ámbito de la optimización de muchos objetivos, los algoritmos se clasifican atendiendo al enfoque que siguen para abordar los retos mencionados anteriormente [7]. En primer lugar, los algoritmos basados en descomposición definen un conjunto de pesos para establecer diferentes direcciones de búsqueda. MOEA/D es el algoritmo más conocido dentro de esta familia. Por otro lado, los algoritmos basados en indicadores utilizan una medida de rendimiento, como el hipervolumen, para guiar la búsqueda. IBEA fue el primer algoritmo de esta familia, donde también destacan HypE o SMS-EMOA. Otro conjunto de algoritmos se caracterizan por la adopción de criterios de dominancia relajados, como la ϵ-dominancia utilizada tanto en algoritmos evolutivos (ϵ-MOEA) como de optimización de partículas (OMOPSO). El uso de un conjunto de puntos de referencia es una técnica habitual para garantizar la diversidad en espacios de objetivos multidimensionales. En este grupo se encuentran algoritmos como NSGA-III y RVEA, entre otros. Finalmente, los métodos basados en preferencias, como WASF-GA o PICEA-g, permiten restringir la búsqueda a una zona de interés para el usuario. Los detalles de estos algoritmos pueden encontrarse en artículos de referencia como [7], [9].

II-B. *Problemas para el estudio del rendimiento*

A la hora de comparar el rendimiento de distintos algoritmos, es frecuente utilizar problemas tipo o *benchmarks*. Por ejemplo, problemas tan conocidos como el de la mochila (*Knapsack Problem*, KP) o el viajante de comercio (*Travelling Salesman Problem*, TSP) han sido extendidos respecto a sus formulaciones originales para un único objetivo. Otro *benchmark* a destacar es LOTZ (*Leading Ones, Trailing Zeros*), un problema simple de codificación binaria con dos objetivos contrapuestos: maximizar el número de unos al comienzo del genotipo y maximizar el número de ceros al final.

Para codificación real existen varias familias de problemas cuyos frentes de Pareto óptimo presentan diferentes propiedades (discontinuidad en el frente, proximidad de óptimos locales, etc.). Además, algunas permiten establecer el número de objetivos que se desea optimizar, por lo que son especialmente útiles para estudiar la escalabilidad. Entre ellas cabe destacar ZDT, un conjunto de seis problemas con dos objetivos cada uno [11], y DTLZ, un conjunto de nueve problemas cuyo número de objetivos es adaptable [12].

III. Implementaciones disponibles en librerías

La popularidad de las técnicas metaheurísticas ha llevado a la creación de librerías y *frameworks* software que dan soporte a la investigación y facilitan su aplicación práctica [3]. Este tipo de herramientas no solo contemplan la configuración y ejecución de los algoritmos y sus componentes, sino que también permiten adaptarlos a nuevos problemas.

La Tabla I muestra un listado de algunas librerías que incluyen soporte para la optimización multiobjetivo, indicando la última versión y su año de lanzamiento, el lenguaje de programación y los algoritmos que proporcionan[1]. Como puede observarse, entre ellas hay herramientas de propósito general muy conocidas, como HeuristicLab y ECJ. A pesar de su madurez, el número de implementaciones multiobjetivo que ofrecen se limita a los algoritmos evolutivos más populares. Otras librerías Java son Opt4J y EvA, las cuales disponen de mayor número y variedad de algoritmos. Más recientemente, han ido apareciendo librerías en Python, como DEAP (especializada en algoritmos distribuidos) y PyGMO (especializada en algoritmos paralelos), si bien el número de algoritmos multiobjetivo en ambas es aún reducido.

ParadisEO-MOEO y JCLEC-MO se asemejan en cuanto a su estructura, pues ambas librerías disponen de un módulo específico para optimización multiobjetivo. Por un lado, ParadisEO ha sido posiblemente la librería más popular de entre las desarrolladas en C++, si bien lleva tiempo sin ser actualizada. Por otro lado, JCLEC-MO ofrece un mayor conjunto de algoritmos, incluyendo algunos especialmente indicados para problemas de muchos objetivos.

Finalmente, también es posible encontrar librerías especializadas en optimización multiobjetivo. La primera de ellas fue PISA, si bien se trata de un proyecto ya abandonado. jMetal y MOEA Framework, ambas desarrolladas en Java, destacan por sus continuas actualizaciones y su amplio catálogo de algoritmos. Dos proyectos muy recientes y activos son PlatEMO y Platypus, desarrollados en Matlab y Python, respectivamente.

IV. Metodología experimental

Para el estudio comparativo de implementaciones multiobjetivo se han seleccionado las seis librerías Java detalladas en la Tabla I. La elección de un único lenguaje de programación facilita la comparación pues evita posibles diferencias debidas al proceso de compilación. Además, estas librerías cuentan con, al menos, dos algoritmos en común, lo cual reduce la posibilidad de alterar el rendimiento debido a la diferencia de implementación entre desarrollos propios y nativos. Tan solo ha sido necesario implementar algunos problemas para disponer de un conjunto lo suficientemente variado, así como operadores genéticos estándar.

Tras analizar las herramientas, se han planteado dos experimentos. El primero tiene como objetivo estudiar distintas

[1]Las posibles variantes de dichos algoritmos no son mostradas por motivos de espacio. El lector es remitido a la documentación de cada herramienta.

Tabla I
Librerías y algoritmos disponibles para optimización multiobjetivo

Librería	Versión	Año	Lenguaje	Principales algoritmos
DEAP	1.2.2	2017	Python	CMAES-MO, NSGA-II, SPEA2
ECJ	26	2018	Java	NSGA-II, NSGA-III, SPEA2
EvA	2.2	2015	Java	CMAES-MO, MOGA, NSGA, NSGA-II, PESA, PESA2, SPEA, SPEA2
HeuristicLab	3.3.15	2018	C#	CMAES-MO, NSGA-II
JCLEC-MO	1.0	2018	Java	ϵ-MOEA, GrEA, HypE, IBEA, MOCHC, MOEA/D, OMOPSO, NSGA-II, NSGA-III, PAES, PAR, RVEA, SMS-EMOA, SMPSO, SPEA2
jMetal	5.4	2017	Java	AbYSS, CellDE, dMOPSO, GDE3, GWASF-GA, IBEA, MOCell, MOCHC, MOEA/D, MOEADD, MOMBI, MOMBI-II, NSGA-II, NSGA-III, OMOPSO, PAES, PESA2, SMS-EMOA, SMPSO, SPEA2, WASF-GA
MOEA Framework	2.12	2017	Java	CMAES-MO, DBEA, ϵ-MOEA, GDE3, IBEA, MOEA/D, MSOPS, NSGA-II, NSGA-III, OMOPSO, PAES, PESA2, RVEA, SMPSO, SMS-EMOA, SPEA2, VEGA
Opt4J	3.1.4	2015	Java	NSGA-II, OMOPSO, SMS-EMOA, SPEA2
PaGMO/PyGMO	2.8	2018	C++/Python	MOEA/D, NSGA-II
ParadisEO-MOEO	2.0.1	2012	C++	MOGA, IBEA, NSGA, NSGA-II, SPEA2
PlatEMO	1.5	2017	Matlab	AGE-II, dMOPSO, ϵ-MOEA, GDE3, GrEA, HypE, I-DBEA, IBEA, LMEA, MOEA/D, MOMBI-II, MOPSO, MSOPS-II, NSGA-II, NSGA-III, PICEA-g, PSEA-II, RPEA, RVEA, SMPSO, SMS-EMOA, SPEA2, Two_Arch2
Platypus	1.0.2	2018	Python	ϵ-MOEA, GDE3, IBEA, MOEA/D, NSGA-II, NSGA-III, OMOPSO, SMPSO, SPEA2.
PISA	-	2009	C	ϵ-MOEA, FEMO, HypE, IBEA, MSOPS, NSGA-II, SEMO2, SHV, SPAM, SPEA2

combinaciones de algoritmos y problemas, donde estos últimos varían en cuanto al tipo de codificación y al número de objetivos. La Tabla II muestra la configuración propuesta, donde el número entre paréntesis representa el número de problema dentro de la familia correspondiente. Para las tres librerías que disponen de un número menor de algoritmos se ha seleccionado un número mayor de problemas. En cuanto a las tres últimas librerías, se han seleccionado algoritmos representativos de distintas familias. En el caso de MOEA/D, únicamente se han considerado problemas de optimización real debido a que es el tipo de problema asumido en jMetal. Igualmente, la generación automática de los pesos se limita a problemas biobjetivo. Por ser un algoritmo de partículas, OMOPSO también queda restringido a problemas de optimización real. La Tabla III recoge los valores de los parámetros, tanto generales como específicos de cada algoritmo.

Tabla II
Algoritmos y problemas seleccionados

Algoritmo	Problema		
	2 obj.	4 obj.	6 obj.
ECJ, EvA, Opt4J			
NSGA-II	LOTZ, ZDT (1, 4)	KP	DTLZ (1, 2, 4)
SPEA2	LOTZ, ZDT (1, 4)	KP	DTLZ (1, 2, 4)
JCLEC-MO, jMetal, MOEA Framework			
IBEA	LOTZ	KP	DTLZ (1, 2)
MOEA/D	ZDT (1, 4)		DTLZ (1, 2)
OMOPSO	ZDT (1, 4)		DTLZ (1, 2)
NSGA-II	LOTZ	KP	DTLZ (1, 2)

El segundo experimento pretende analizar la escalabilidad de las implementaciones en base a tres factores: el tamaño de la población (100, 500, 1000), el número de generaciones (100, 500, 1000) y el número de objetivos (2, 5, 10, 25, 50). Estas configuraciones serán identificadas como Px-Gy-Oz, donde P representa el tamaño de la población, G, las generaciones y O, los objetivos. El algoritmo escogido es NSGA-II, ya que está presente en todas las librerías y no presenta parámetros adicionales. Puesto que se desea variar el número de objetivos, se ha seleccionado DTLZ1 como problema a resolver. Los operadores genéticos y sus probabilidades se configuran de acuerdo a los valores mostrados en la Tabla III.

Tabla III
Configuración de parámetros

Parámetro	Valor
Parámetros comunes	
Tamaño de la población	100
Número de generaciones	100
Probabilidad de cruce	0,9
Probabilidad de mutación	0,1 (1/longitud_genotipo)
Operador de cruce	*1-Point* (binario), SBX (real)
Operador de mutación	*Bit flip* (binario), *Polynomial* (real)
SPEA2	
Tamaño de la población (P)	50
Tamaño del archivo (A)	50
Selector de padres	Torneo binario
Parámetro k	$\sqrt[2]{P+A}$
MOEA/D	
Tamaño de vecindad (τ)	10
Máx. Núm. Reemplazos (nr)	2
Modo de evaluación	*Tchebycheff*
Generación de pesos	Uniforme
IBEA	
Selector de padres	Torneo binario
Indicator para evaluación	Hipervolumen
Parámetro κ	0,05
Parámetro ρ	2
OMOPSO	
Tamaño del archivo	100
Parámetro ϵ	0,0075

En ambos experimentos se ha utilizado Syrupy[2] como herramienta *profiler* para medir el tiempo de ejecución y el consumo de memoria a intervalos regulares. Dichos intervalos se han establecido en base a ejecuciones previas con el fin de

[2]https://github.com/jeetsukumaran/Syrupy

obtener un mínimo de 30 muestras para cada configuración. La experimentación se ha realizado en una máquina con Debian 8, ocho núcleos Intel Core i7-2600, CPU a 3,4 GHz y 16 GB de memoria RAM. En el caso del primer experimento, cada configuración ha sido ejecutada cinco veces en un único núcleo para evitar posibles interferencias. Además, se ha alternado el orden de ejecución de cada librería para mantener condiciones similares respecto al arranque de la JVM. En el segundo experimento, cada configuración ha sido lanzada seis veces utilizando dos núcleos para que el experimento no tuviese una duración excesiva. No obstante, las configuraciones también se planificaron de forma que no siempre las mismas librerías estuviesen ejecutando en paralelo a fin de reducir el sesgo lo más posible. Tras obtener el tiempo medio de ejecución de cada configuración, se ha aplicado el test de Friedman para analizar si existen diferencias significativas entre aquellas librerías para las que se ejecutaron los mismos algoritmos y problemas. Es decir, para el primer experimento se analizan dos grupos por separado, uno formado por ECJ, EvA y Opt4J, y otro formado por JCLEC-MO, jMetal y MOEA Framework. En caso de encontrar diferencias significativas, se aplica el post-procedimiento de Holm. Además, se ha ejecutado el test Cliff's Delta para analizar el tamaño del efecto, esto es, la magnitud de la diferencia observada.

V. Resultados y discusión

En esta sección se presentan los resultados obtenidos para cada uno de los experimentos planteados. Por motivos de espacio, solo los resultados más relevantes son discutidos.

V-A. Comparación de algoritmos y problemas

Las Tablas IV y V muestran los tiempos medios de ejecución y la desviación estándar para los dos grupos de librerías. La ejecución más rápida (0,24 s) se corresponde con la resolución de DTLZ2 con MOEA/D en jMetal. Por el contrario, el problema de la mochila con NSGA-II en ECJ es la combinación que más tiempo ha necesitado (4,88 s).

En el primer grupo de librerías (Tabla IV), ECJ es la que requiere menos tiempo, empleando más de un segundo en solo cuatro configuraciones. EvA es la librería que reporta tiempos de ejecución más altos, siempre por encima de un segundo, mientras que Opt4J mantiene tiempos más estables, entre 0,5 y 1,5 s. Para confirmar si estas diferencias son significativas, se ha aplicado el test de Friedman, donde la hipótesis nula establece que todas las librerías tienen un rendimiento similar. El valor del estadístico de *Iman and Daveport*, z, es 30,71, mientras que el valor crítico de acuerdo a una distribución F con 2 y 30 grados de libertad es igual a 5,39 ($\alpha = 0,01$). Dado que z es mayor que el valor crítico, la hipótesis nula puede ser rechazada. El test de Holm devuelve un $p - value$ igual a 0,05, lo que confirma que ECJ es significativamente superior a EvA, pero equivalente a Opt4J. En cuanto al efecto del tamaño, el test Cliff's Delta indica que la diferencia en tiempo entre las tres librerías es siempre amplia (*large*).

En el segundo grupo de librerías (JCLEC-MO, jMetal y MOEA Framework), IBEA es el algoritmo más costoso, requiriendo varios segundos de ejecución. La diferencia con respecto al resto de algoritmos es más evidente en JCLEC-MO y jMetal, especialmente cuando se resuelven problemas combinatorios. Para este conjunto de librerías, el test de Friedman no encuentra diferencias significativas. De hecho, el test Cliff's Delta siempre cataloga las diferencias entre cada pareja de librerías como insignificante (*negligible*), salvo entre jMetal y JCLEC-MO, que es pequeña (*small*).

Dado que todas estas configuraciones son simples y se ejecutan en pocos segundos, tan solo se pueden realizar algunas apreciaciones respecto a la memoria RAM consumida. Las seis librerías Java analizadas consumen un mínimo de 12.000 KB en cada ejecución. EvA y Opt4J son las dos librerías que muestran un comportamiento más estable, con un máximo de 56.459 KB y 126.084 KB de media, respectivamente. Entre el resto, el uso máximo de memoria es más disperso llegando a presentar "picos" de más de 100.000 KB en JCLEC-MO (76.130 KB de media), jMetal (88.326 KB) y MOEA Framerwork (89.005 KB), y más de 500.000 KB en ECJ (149.265 KB) cuando se resuelve el problema de la mochila.

Valorando los resultados del primer experimento en global,

Tabla IV
Tiempo de ejecución en segundos para el primer grupo de librerías (experimento 1)

Configuración	ECJ	EvA	Opt4J
NSGA-II – LOTZ(2)	$0,245 \pm 0,014$	$1,443 \pm 0,037$	$0,773 \pm 0,050$
NSGA-II – ZDT1(2)	$0,299 \pm 0,015$	$1,827 \pm 0,019$	$0,785 \pm 0,056$
NSGA-II – ZDT4(2)	$0,295 \pm 0,017$	$1,537 \pm 0,014$	$0,765 \pm 0,068$
NSGA-II – ZDT6(2)	$0,284 \pm 0,034$	$1,466 \pm 0,023$	$0,758 \pm 0,058$
NSGA-II – KP(4)	$4,879 \pm 0,022$	$1,587 \pm 0,041$	$1,142 \pm 0,076$
NSGA-II – DTLZ1(6)	$0,344 \pm 0,021$	$1,658 \pm 0,043$	$0,896 \pm 0,067$
NSGA-II – DTLZ2(6)	$0,336 \pm 0,022$	$1,803 \pm 0,040$	$0,900 \pm 0,043$
NSGA-II – DTLZ4(6)	$0,351 \pm 0,034$	$1,834 \pm 0,039$	$0,922 \pm 0,110$
SPEA2 – LOTZ(2)	$0,444 \pm 0,017$	$1,445 \pm 0,022$	$0,609 \pm 0,038$
SPEA2 – ZDT1(2)	$0,368 \pm 0,028$	$2,684 \pm 0,032$	$0,999 \pm 0,087$
SPEA2 – ZDT4(2)	$0,291 \pm 0,022$	$1,380 \pm 0,029$	$1,020 \pm 0,060$
SPEA2 – ZDT6(2)	$0,300 \pm 0,021$	$1,385 \pm 0,047$	$0,735 \pm 0,032$
SPEA2 – KP(4)	$3,219 \pm 0,033$	$1,736 \pm 0,105$	$1,173 \pm 0,065$
SPEA2 – DTLZ1(6)	$0,959 \pm 0,050$	$1,646 \pm 0,107$	$1,633 \pm 0,236$
SPEA2 – DTLZ2(6)	$1,442 \pm 0,023$	$2,702 \pm 0,038$	$1,583 \pm 0,070$
SPEA2 – DTLZ4(6)	$1,435 \pm 0,088$	$2,897 \pm 0,059$	$1,699 \pm 0,103$
Friedman (ranking)	**1,250**	2,875	1,875
Holm (α/i)	-	0,025	0,050

Tabla V
Tiempo de ejecución en segundos para el segundo grupo de librerías (experimento 1)

Configuración	JCLEC-MO	jMetal	MOEA Fram.
NSGA-II – LOTZ(2)	$0,410 \pm 0,009$	$0,624 \pm 0,023$	$0,328 \pm 0,018$
NSGA-II – KP(4)	$0,519 \pm 0,027$	$0,558 \pm 0,042$	$0,374 \pm 0,016$
NSGA-II – DTLZ1(6)	$0,537 \pm 0,028$	$0,540 \pm 0,066$	$0,444 \pm 0,011$
NSGA-II – DTLZ2(6)	$0,587 \pm 0,064$	$0,595 \pm 0,055$	$0,389 \pm 0,008$
IBEA – LOTZ(2)	$4,754 \pm 0,037$	$1,288 \pm 0,039$	$0,844 \pm 0,019$
IBEA – KP(4)	$3,012 \pm 0,036$	$1,678 \pm 0,142$	$1,036 \pm 0,029$
IBEA – DTLZ1(6)	$1,829 \pm 0,030$	$2,556 \pm 0,034$	$1,254 \pm 0,011$
IBEA – DTLZ2(6)	$1,843 \pm 0,018$	$2,538 \pm 0,032$	$1,285 \pm 0,007$
MOEA/D – ZDT1(2)	$0,429 \pm 0,018$	$0,337 \pm 0,020$	$0,458 \pm 0,008$
MOEA/D – ZDT4(2)	$0,413 \pm 0,030$	$0,277 \pm 0,010$	$0,414 \pm 0,010$
MOEA/D – DTLZ1(2)	$0,380 \pm 0,022$	$0,260 \pm 0,013$	$0,434 \pm 0,023$
MOEA/D – DTLZ2(2)	$0,492 \pm 0,035$	$0,236 \pm 0,004$	$0,412 \pm 0,016$
OMOPSO – ZDT1(2)	$0,556 \pm 0,034$	$0,412 \pm 0,050$	$1,269 \pm 0,040$
OMOPSO – ZDT4(2)	$0,530 \pm 0,043$	$0,284 \pm 0,010$	$1,117 \pm 0,032$
OMOPSO – DTLZ1(2)	$0,663 \pm 0,009$	$0,484 \pm 0,042$	$1,510 \pm 0,016$
OMOPSO – DTLZ2(2)	$0,715 \pm 0,035$	$1,466 \pm 0,086$	$2,139 \pm 0,072$
Friedman (ranking)	2,125	**1,938**	**1,938**

todas las librerías presentan unos requisitos de tiempo y memoria asumibles ante un escenario de uso normal. Aún así, y a pesar de que las configuraciones probadas son poco exigentes, se perciben ciertas diferencias. En concreto, no todas las librerías responden igual ante la ejecución de un mismo algoritmo en problemas de distinta naturaleza (combinatorios frente a optimización real). En este sentido, los problemas combinatorios son más costosos, aún cuando el número de objetivos es inferior al considerado para los problemas DTLZ. Esto puede deberse a que la evaluación requiere la interpretación del genotipo, frente al uso directo de las variables reales. Por otro lado, la formulación concreta de los diferentes problemas ZDT no parece tener una gran influencia en el tiempo, mientras que sí se observan algunas diferencias respecto a los problemas DTLZ (véase por ejemplo OMOPSO en jMetal y MOEA Framework, o SPEA2 en ECJ y EvA). Respecto a los algoritmos, NSGA-II suele ser el más rápido en la mayoría de librerías. La única excepción parece ser la implementación de MOEA/D en JCLEC-MO y jMetal, si bien hay que considerar que este algoritmo solo ha sido probado con problemas biobjetivo de codificación real.

V-B. Test de escalabilidad

El segundo experimento, donde se ejecutan configuraciones con hasta 50 objetivos, permite detectar mayores diferencias entre las librerías. La Tabla VI muestra los tiempos mínimos y máximos medios, con su desviación estándar, para cada tamaño de población. También se incluye el ranking obtenido con el test de Friedman. Cabe señalar que los tiempos mínimos y máximos corresponden, por lo general, con las configuraciones P100-G100-O2 y P1000-G5000-050. Las excepciones son algunos tiempos mínimos en Opt4J, JCLEC-MO y jMetal, que se consiguen para cinco objetivos. No obstante, los valores son muy similares a los obtenidos para dos objetivos, por lo que pueden deberse a pequeñas fluctuaciones en la carga de la máquina. En base a estos resultados, ECJ es la librería que mejor responde ante configuraciones más exigentes, seguida de MOEA Framework, si bien el test de Holm no indica la existencia de diferencias significativas entre ambas. Les siguen jMetal y JCLEC-MO, si bien cabe señalar que JCLEC-MO responde algo mejor que jMetal ante el aumento del número de objetivos. Lo mismo ocurre si se comparan las dos librerías más lentas según este estudio, pues aunque Opt4j y EvA reportan tiempos mínimos similares, Opt4J escala mejor.

Por otro lado, los datos extraídos muestran que el incremento en el tiempo de ejecución no depende únicamente del número de generaciones. Por ejemplo, configuraciones con 1000 generaciones y dos objetivos pueden requerir menos tiempo de cómputo que configuraciones con 500 generaciones y 50 objetivos. Aunque hay algunas excepciones, sí se aprecia un incremento lineal cuando dos de los parámetros son fijados y el otro aumenta progresivamente. Los dos parámetros más determinantes son el número de generaciones y el número de objetivos, especialmente cuando se consideran valores superiores a 500 y 10, respectivamente. A modo de ejemplo, la Figura 1 muestra el tiempo medio de ejecución para las distintas combinaciones de objetivos y generaciones, fijando el tamaño de población en 1000 (el valor más alto probado). Para facilitar la comparación entre librerías, se ha mantenido la misma escala respecto al tiempo. A excepción de EvA, las librerías presentan una tendencia similar y no suelen requerir más de 30 minutos por ejecución.

Con respecto al consumo de memoria, el alto número de combinaciones probadas permite obtener mayores diferencias. A modo de resumen, la Tabla VII recoge el valor mínimo y máximo de memoria registrado para cada librería. Como puede verse, no existen grandes diferencias en cuanto a los valores mínimos, pero sí en los máximos alcanzados. También se muestra el porcentaje de veces que cada librería obtiene el mínimo y máximo global para las 60 combinaciones probadas. En este caso, ECJ y Opt4J son las librerías que menor y mayor memoria consumen, respectivamente.

Este test de escalabilidad confirma que las implementaciones Java actuales pueden sufrir una degradación considerable en su rendimiento si se las somete a configuraciones extremas. Aunque quizás no sean los valores más habituales en la práctica, también debe considerarse que NSGA-II no es de los algoritmos más costosos y que el problema es sintético. En una aplicación real, donde fuese necesario ejecutar algoritmos específicos para problemas de muchos objetivos cuya evaluación fuese costosa, la elección de una implementación u otra podría ser determinante. Entre las librerías analizadas, EvA y Opt4J presentan mayores problemas para controlar el tiempo y la memoria, respectivamente. Por el contrario, ECJ y MOEA Framework presentan el mejor rendimiento considerando ambos factores a la vez, ya que tanto jMetal como JCLEC-MO experimentan mayores oscilaciones de memoria.

VI. Conclusiones

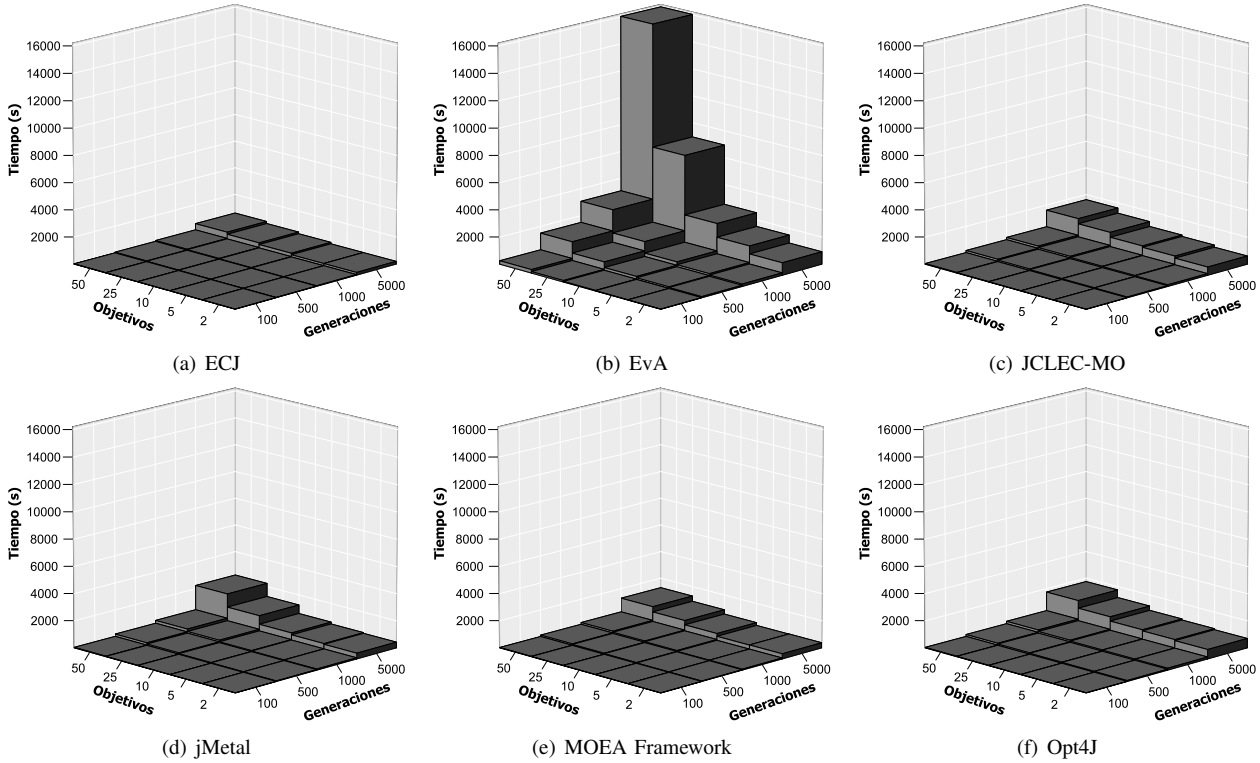
Este trabajo ha presentado una comparativa experimental de implementaciones de algoritmos multiobjetivo disponibles en seis librerías Java con el fin de analizar sus requisitos de tiempo y memoria. Los distintos algoritmos han sido ejecutados considerando configuraciones que difieren en el tipo de problema, el número de objetivos, el número de generaciones y el tamaño de la población. Los dos experimentos realizados indican que estos parámetros afectan en mayor o menor medida a todas las librerías. Para un uso estándar, las diferencias son poco apreciables y están condicionadas a la complejidad del algoritmo o del problema a resolver. Sin embargo, si se necesitan valores elevados para los parámetros anteriores, es conveniente analizar la forma en que cada librería implementa el algoritmo, pues las diferencias pueden ser notables. En este sentido, este estudio puede extenderse para abarcar un mayor número de librerías, algoritmos y problemas. Asimismo, sería interesante analizar la capacidad de las distintas librerías para paralelizar la evaluación de soluciones.

Referencias

[1] T. Stewart, O. Bandte, H. Braun, N. Chakraborti, M. Ehrgott, M. Göbelt, Y. Jin, H. Nakayama, S. Poles, and D. Di Stefano, "Real-World Applications of Multiobjective Optimization," in *Multiobjective Optimization*, vol. 5252 of *LNCS*, pp. 285–327, springer Berlin Heidelberg, 2008.

Tabla VI
Tiempos mínimos y máximos en segundos (experimento 2)

Librería	P = 100		P = 500		P = 1000		Ranking
	Min.	Máx.	Min.	Máx.	Min.	Máx.	
ECJ	$0,34 \pm 0,04$	$14,37 \pm 0,92$	$0,79 \pm 0,05$	$170,61 \pm 2,44$	$2,73 \pm 0,09$	$589,33 \pm 3,58$	1,00
EvA	$1,79 \pm 0,02$	$226,77 \pm 2,25$	$5,74 \pm 0,62$	$3830,01 \pm 94,69$	$16,38 \pm 0,49$	$16030,96 \pm 512,88$	6,00
JCLEC-MO	$0,64 \pm 0,07$	$30,55 \pm 0,58$	$3,25 \pm 0,10$	$475,77 \pm 4,83$	$11,86 \pm 0,07$	$1611,35 \pm 91,90$	4,11
jMetal	$0,58 \pm 0,06$	$30,74 \pm 1,21$	$2,64 \pm 0,07$	$576,81 \pm 3,86$	$9,01 \pm 0,17$	$2237,61 \pm 30,34$	3,35
MOEA Framework	$0,37 \pm 0,01$	$19,26 \pm 0,21$	$1,43 \pm 0,08$	$339,96 \pm 1,59$	$4,27 \pm 0,14$	$1284,00 \pm 7,17$	2,02
Opt4J	$0,93 \pm 0,08$	$39,18 \pm 0,68$	$4,12 \pm 0,09$	$533,96 \pm 6,95$	$13,98 \pm 0,31$	$1743,74 \pm 12,82$	4,52



(a) ECJ (b) EvA (c) JCLEC-MO

(d) jMetal (e) MOEA Framework (f) Opt4J

Figura 1. Variación en el tiempo de ejecución de NSGA-II para una población de 1000 individuos (experimento 2)

Tabla VII
Resumen del consumo de memoria (experimento 2)

Librería	Valores (KB)		Porcentajes	
	Min.	Máx.	Min.	Máx.
ECJ	123960	2064160	26,67	0,00
EvA	111080	1676000	15,00	0,00
JCLEC-MO	123840	6450680	13,33	0,00
jMetal	124080	14485480	23,33	23,33
MOEA Framework	123760	2589000	15,00	0,00
Opt4J	124480	3909920	6,67	76,67

[2] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2nd ed., 2007.

[3] J. A. Parejo, A. Ruiz-Cortés, S. Lozano, and P. Fernández, "Metaheuristic optimization frameworks: a survey and benchmarking," *Soft Comput.*, vol. 16, no. 3, pp. 527–561, 2012.

[4] S. Chand and M. Wagner, "Evolutionary many-objective optimization: A quick-start guide," *Surv. Oper. Res. Manag. Sci.*, vol. 20, no. 2, pp. 35–42, 2015.

[5] A. López Jaimes and C. A. Coello Coello, *Springer Handbook of Computational Intelligence*, ch. Many-Objective Problems: Challenges and Methods, pp. 1033–1046. Springer Berlin Heidelberg, 2015.

[6] V. Khare, X. Yao, and K. Deb, "Performance Scaling of Multi-objective Evolutionary Algorithms," in *Proc. 2nd Int. Conf. Evolutionary Multi-Criterion Optimization*, pp. 376–390, 2003.

[7] B. Li, J. Li, K. Tang, and X. Yao, "Many-Objective Evolutionary Algorithms: A Survey," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 13:1–35, 2015.

[8] J. Maltese, B. M. Ombuki-Berman, and A. P. Engelbrecht, "A Scalability Study of Many-Objective Optimization Algorithms," *IEEE T Evolut. Comput.*, vol. 22, no. 1, pp. 79–96, 2018.

[9] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evolut. Comput.*, vol. 1, no. 1, pp. 32–49, 2011.

[10] C. A. Coello Coello, "Evolutionary Multiobjective Optimization: Current and Future Challenges," in *Advances in Soft Computing*, pp. 243–256, Springer London, 2003.

[11] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolut. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.

[12] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, *Scalable Test Problems for Evolutionary Multiobjective Optimization*, ch. Evolutionary Multiobjective Optimization, pp. 105–145. Springer London, 2005.

Algoritmo Evolutivo con División del Espacio de los Objetivos en base a la Solución Nadir: Un Estudio Comparativo sobre el Problema de la Mochila 0/1 Bi-Objetivo

1st Máximo Méndez
Instituto Universitario de Sistemas Inteligentes (SIANI)
Universidad de Las Palmas de Gran Canaria (ULPGC)
España
maximo.mendez@ulpgc.es

2nd Daniel Alejandro Rossit
Departamento de Ingeniería
CONICET, Universidad Nacional del Sur (UNS)
Argentina
daniel.rossit@uns.edu.ar

3rd Mariano Frutos
Departamento de Ingeniería
CONICET, Universidad Nacional del Sur (UNS)
Argentina
mfrutos@uns.edu.ar

4th Begoña González
Instituto Universitario de Sistemas Inteligentes (SIANI)
Universidad de Las Palmas de Gran Canaria (ULPGC)
España
bego.landin@ulpgc.es

***Resumen*—Este trabajo presenta un algoritmo evolutivo multiobjetivo (AEMO) el cual divide el espacio de los objetivos, en varias regiones utilizando la solución Nadir calculada en cada generación del algoritmo. Para la clasificación de las soluciones de las distintas regiones en frentes no-dominados, se utilizan diferentes estrategias de optimización de las funciones objetivo. La idea es intensificar la diversidad del frente de soluciones no-dominadas alcanzado. El algoritmo propuesto (NSGA-II/OSD) se implementa sobre el algoritmo NSGA-II y se ensaya sobre el Problema de la Mochila 0/1 Bi-Objetivo (MOKP-0/1) bien conocido en la comunidad multiobjetivo. Con dos objetivos, este problema es de difícil resolución para un AEMO dado el elevado número de soluciones superpuestas (overlapping solutions) que se generan durante su evolución. El método propuesto ofrece muy buen desempeño cuando es comparado con los algoritmos NSGA-II y MOEA/D muy reconocidos ambos en la literatura especializada.**

***Index Terms*—Optimización, Algoritmos Evolutivos Multiobjetivo, Dominancia de Pareto, MOKP0/1, NSGA-II, MOEA/D.**

I. Introducción

Un problema de optimización multiobjetivo (POM) es aquel que corresponde a una cierta realidad industrial, económica o de otra índole y sobre el que un decisor desea optimizar varios objetivos usualmente en conflicto entre sí. Cuando se resuelve un POM de complejidad difícil, métodos metaheurísticos son muy apropiados. Estos métodos no grantizan obtener el frente exacto de soluciones no-dominadas, pero sí un conjunto aproximado.

La segunda generación de algoritmos evolutivos multiobjetvo (AEMOs), ha demostrado obtener excelentes resultados resolviendo POM [1]. De naturaleza estocástica, los AEMOs están basados en el concepto de población de soluciones lo que les proporciona gran destreza para encontrar múltiples soluciones no-dominadas en espacios de soluciones de diversa naturaleza. NSGA-II [2] y MOEA/D [12] son dos AEMOs muy reconocidos en la comunidad científica multiobjetivo. El primero, utiliza una clasificación de rangos por dominancia como mecanismo de convergencia, mientras que crowding-distance es el mecanismo usado para diversificar las soluciones. El segundo, funciona descomponiendo el POM en un número de subproblemas escalares o mono-objetivos que son resueltos todos a la misma vez mediante la evolución de una población de soluciones. Algunos trabajos recientes describen dificultades en cuanto a la especificación del punto de referencia en MOEA/D [6], [7], [9], [11]. Trabajos con AEMOs que incluyan un sub-división del espacio de los objetivos como la propuesta en este trabajo son muy escasos, sólo hemos encontrado [8], [10].

El problema de la mochila multiobjetivo en variables binarias (MOKP/0-1) es un problema combinatorio bien conocido y ampliamente manejado en la comunidad multi-objetivo. Un considerable número de métodos exactos y metaheurísticos, ver por ejemplo [3], [5], [12], intentan resolver este problema. En [4], se señala que se genera un elevado número de soluciones superpuestas (overlapping solutions) cuando los AEMOs se aplican a problemas combinatorios con muchas variables de decisión y baja dimensionalidad. La aparición de soluciones superpuestas cuando el Problema de la Mochila 0/1 Bi-Objetivo se resuelve con un AEMO, tiene un significativo impacto negativo sobre la diversidad del frente final de soluciones no-dominadas alcanzado.

Se propone un híbrido de NSGA-II, que subdivide el espacio de los objetivos en varias regiones utilizando la solución Nadir

calculada en cada generación del algoritmo. Las distintas subdivisiones utilizan diferentes estrategias de optimización de las funciones objetivo. La idea que subyace es intensificar la diversidad de soluciones del frente alcanzado, permitiendo que soluciones dominadas por los extremos del frente no-dominado alcanzado en cada generación del algoritmo puedan entrar en la población N no-dominada del algoritmo.

El resto del trabajo se organiza de la siguiente manera. En la sección II se presentan algunos conceptos básicos para entender mejor este trabajo. En la sección III se describe el método propuesto. Los resultados experimentales se detallan en la sección IV. Por último, en la sección V se presentan las conclusiones.

II. Conceptos básicos

En términos de minimización, un POM puede ser definido de la siguiente forma:

$$\begin{array}{l} Min.\ \ F(x) = f_1(x), f_2(x), ..., f_n(x) \\ \quad s.a. \\ \quad g_k(x) \leq 0 \qquad k \in (1,2,...,p) \\ \quad h_l(x) = 0 \qquad l \in (1,2,...,q) \end{array} \tag{1}$$

donde $F(x)$ es el vector de objetivos a minimizar y $n \geq 2$ el número de objetivos. Las ecuaciones $g_k(x) \leq 0$ y $h_l(x) = 0$ representan respectivamente p restricciones de desigualdad y q restricciones de igualdad. Los valores de x que satisfacen el conjunto de las $(p+q)$ restricciones, definen el espacio realizable *S*. El vector $x = (x_1, x_2, ..., x_m) \in S$ es un vector solución de m variables de decisión.

El conjunto de las imágenes de cada solución realizable en el espacio de la decisión, conforma el conjunto de soluciones realizables en el espacio de los objetivos $Z = f(S)$ definido como $Z = \{z_1 = f_1(x), ..., z_n = f_n(x),\ \forall x \in S\}$ y donde $z = (z_1, ..., z_n) \in R^n$ representa una solución realizable en el espacio de los objetivos.

Una solución $z^t = (z_1^t, z_2^t, ..., z_n^t) \in Z$ domina una solución $z^u = (z_1^u, z_2^u, ..., z_n^u) \in Z$ y se le conoce como Pareto-óptima si se verifican las siguientes condiciones:

1. $z_j^t \leq z_j^u \qquad \forall j \in (1,2,...,n)$
2. $\exists j \in (1,2,...,n)\ \ tal\ \ que\ \ z_j^t < z_j^u$

Al conjunto de soluciones Pareto-óptimas en el espacio de soluciones se le llama conjunto óptimo de Pareto y a su imagen en el espacio de objetivos frente óptimo de Pareto.

II-A. MOEA/D con especificación de Zhang del punto de referencia

Básicamente, MOEA/D trabaja descomponiendo un POM en un número finito de sub-problemas escalares y resolviéndolos simultáneamente mediante la evolución de una población de soluciones. Algunos enfoques para convertir un POM en un sub-problema escalar se pueden consultar en [12]. El enfoque de Tchebycheff se expresa como sigue:

$$\begin{array}{l} Min.\ \ g^{te}(x|\lambda, z^*) = \max\limits_{1 \leq i \leq n} \{\lambda_i f_i(x) - z_i^*\} \\ \quad s.a. \quad x \in X \end{array} \tag{2}$$

donde $z^* = (z_1^*, ..., z_n^*)$ es el punto de referencia (PR) especificado en este trabajo de la siguiente forma:

$$z_i^* = \alpha * min\{f_i(x) | x \in X\}\ paracada\ i = 1, ..., n \tag{3}$$

y donde $\alpha \geq 1$.

II-B. MOEA/D con especificación de Ishibuchi del punto de referencia

Ishibuchi et al. proponen en [6] usar en (2) la siguiente especificación para el PR:

$$z^R = z_i^* - \alpha_t(z_i^{max} - z_i^*) \tag{4}$$

donde $z_i^* = (z_1^*, ..., z_n^*)$ corresponde al mínimo valor en la función objetivo i, $z_i^{max} = (z_1^{max}, ..., z_n^{max})$ corresponde al máximo valor de la función objetivo i en la generación en curso y α_t es un parámetro que decrece en cada generación del algoritmo. De (4) es obvio que cuando $\alpha_t = 0$ los valores z^R y z^* tienen los mismos valores.

Para hacer que el valor z^R se aproxime gradualmente al valor del punto de referencia z^* durante la ejecución del algoritmo, el valor α_t es modificado en cada generación según:

$$\alpha_t = \alpha\ (t_{max} - t)/(t_{max} - 1) \tag{5}$$

donde α es un valor inicial de α_t, t_{max} es el número máximo de generaciones del algoritmo y t es la generación actual. Nótese que el valor final de α_t en la última generación es cero, i.e., z^R tiende a z^* durante la ejecución del algoritmo.

III. Método propuesto

El algoritmo propuesto en este trabajo consiste en una versión modificada de NSGA-II, en los siguientes términos:

1. En cada generación t del algoritmo: una población $R_t = P_t + Q_t$ es construida con P_t la población de padres de tamaño N y Q_t la población de hijos (tamaño N) generados de las operaciones de cruzamiento y mutación a partir de la población N construida en la generación t-1.
2. A partir de la población R_t la solución Nadir es identificada mediante algún método.
3. En base a la solución Nadir el espacio de los objetivos se divide en tres regiones independientes R_1, R_2 y R_3 tal como se indica en la Fig. 1 izquierda.
4. Las soluciones de cada región R_1, R_2 y R_3 son clasificadas (complejidad computacional O(n2N)) en diferentes frentes no-dominados (F_1, F_2,...,F_n). Para la clasificación de las soluciones, se usan diferentes criterios de optimización de las funciones objetivo f_1 y f_2.
   - *a*) Región R1: Las dos funciones objetivo f_1 y f_2 son siempre maximizadas (ver Fig. 1 derecha).
   - *b*) Región R2: La función f_1 es siempre maximizada. La función f_2 es maximizada si se cumple la condición t≥nt, en caso contrario f_2 es minimizada (ver Fig. 1 derecha). El valor t es la generación actual y nt un parámetro (número de generaciones

que se maximiza o minimiza la función objetivo) que se define como:

$$nt = \alpha\ t_{max} \quad 0 \leq \alpha \leq 1 \tag{6}$$

c) Región R3: La función f_2 es siempre maximizada. La función f_1 es maximizada si se cumple la condición t≥nt, en caso contrario f_1 es minimizada (ver Fig. 1 derecha).

Finalizado el paso 4, toda la población R_t queda clasificada en frentes de no-dominación (F_1, F_2,...,F_n).

5. Para completar la población P_{t+1} hasta alcanzar un tamaño N, como en NSGA-II, las soluciones con menor rango primero, y mayor crowding-distance segundo, son escogidas.

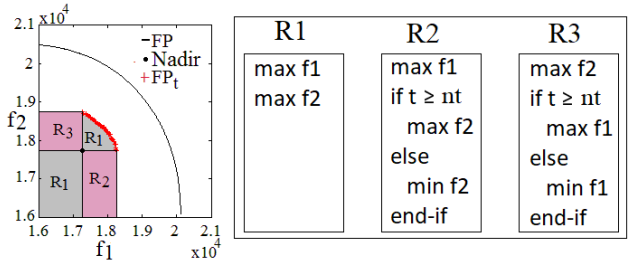


Figura 1. División propuesta del espacio de los objetivos en base a la solución Nadir (izquierda). Criterios de optimización de las funciones objetivo para cada región R_1, R_2 y R_3 (derecha).

IV. Experimentos y comparaciones

IV-A. Problema de la Mochila Multiobjetivo MOKP/0-1

El problema de la mochila multiobjetivo en variables binarias (MOKP/0-1) consiste en una serie de objetos o ítems con un peso, un beneficio asociado a cada uno de ellos y un límite de capacidad para cada mochila. Así, la tarea radica en encontrar el subconjunto de objetos que maximicen los beneficios totales de cada mochila y que puedan ser colocados en dichas mochilas sin exceder sus límites de capacidad. El problema es de complejidad NP-difícil y puede ser usado para modelar cualquier aplicación real que se ajuste al modelo descrito en (7).

El MOKP/0-1 puede ser definido formalmente como sigue:

$$\begin{cases} max. \quad f_i(x) = \sum_{j=1}^{m} b_{ij} x_j \quad i = 1, ..., n \\ s.a. \qquad \sum_{j=1}^{m} w_{ij} x_j \leq c_i \qquad x_j \epsilon \{0, 1\} \end{cases} \tag{7}$$

donde: m=número de objetos, x_j=variable de decisión, n=número de sacos, b_{ij}=beneficio objeto j según saco i, w_{ij}=peso objeto j según saco i y c_i = capacidad saco i.

IV-B. Configuración de parámetros y métricas

En este trabajo, las implementaciones de NSGA-II y MOEA/D se han realizado según [2] y [12]. Ambos algoritmos son aplicados sobre el 500-MOKP/0-1 con dos objetivos, los datos del problema y frente óptimo pueden descargarse de: http://www.tik.ee.ethz.ch/ sop/download/supplementary/ testProblemSuite/.

En todos los experimentos se utilizó codificación binaria, cruce uniforme de probabilidad 0.8 y probabilidad de mutación (bit a bit) de 1/500. Un número de 400000 evaluaciones de la función objetivo se usó como condición de parada. Tres tamaños de la población N=50, N=100 y N=200 soluciones fueron utilizadas en los tres algoritmos NSGA-II/OSD, NSGA-II y MOEA/D. Los valores examinados para el parámetro α definido en (6) de balance convergencia/diversidad de NSGA-II/OSD fueron α= 0.0, 0.1, 0.2, 0.3 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 y 1.0. Con MOEA/D, los valores ensayados de α en el cálculo del punto de referencia z^* según la especificación de Zhang definida en (3) fueron α= 1.0, 1.05, 1.1, 1.15, 1.2, 1.25, 1.3; en la versión de Ishibuchi para el cálculo del punto de referencia z^R definido en (4) y (5), los valores se fijaron en α= 0.0, 0.05, 0.1, 0.5, 1.0, 1.1, 2.0, 3.0, 4.0, 5.0 y 10.0.

Para la comparación de los resultados obtenidos por los algoritmos, se emplearon la S-metric o hipervolumen H (punto de referencia usado el 0,0) sugerida en [13] y visualizaciones de los frentes de soluciones de Pareto alcanzados de valor H más próximo al valor medio final calculado en 30 ejecuciones independientes.

IV-C. Análisis previo: NSGA-II/OSD y efectos de variar el parámetro α

En esta subsección se analizan las influencias en la convergencia y diversidad del frente de soluciones alcanzado, cuando se varía el parámetro α en la ecuación (6) en el algoritmo propuesto NSGA-II/OSD. En la Fig. 2 (izquierda), se muestran los valores del hipervolumen medio al modificar los valores del parámetro α. Se puede observar que los mejores valores de H se obtienen para valores intermedios del parámetro α. Para mejor interpretar estos resultados, la Fig. 2 (centro) muestra los efectos sobre la distribución de soluciones de los frentes de Pareto obtenidos cuando N=200. Los frentes de Pareto de menor diversidad (mayor convergencia) y mayor diversidad (menor convergencia) se obtienen cuando α=0.0 y α=1.0 respectivamente. El frente con el mejor valor del hipervolumen se alcanza cuando α=0.5, ver la Fig. 2 (derecha).

IV-D. Resultados comparativos: NSGA-II/OSD vs NSGA-II

En esta subsección se compara el algoritmo NSG-II/OSD propuesto en este trabajo con el algoritmo NSGA-II de Deb et al. [2]. La Fig. 3 muestra los frentes de soluciones de Pareto obtenidos por ambos algoritmos. Se observa que el frente de soluciones alcanzado con el método propuesto (α=0.5) tiene bastante más diversidad que el alcanzado por NSGA-II, aunque en términos de convergencia parece que NSGA-II obtiene mejor valor. La Fig. 4 (izquierda) muestra que en términos de la métrica H, el frente de soluciones logrado por NSGA-II/OSD es mejor que el obtenido por NSGA-II; también la Fig. 4 (derecha) señala para NSGA-II/OSD un menor valor de la desviación estándard del hipervolumen. En la Fig. 5 (izquierda) se aprecia que durante la evolución de

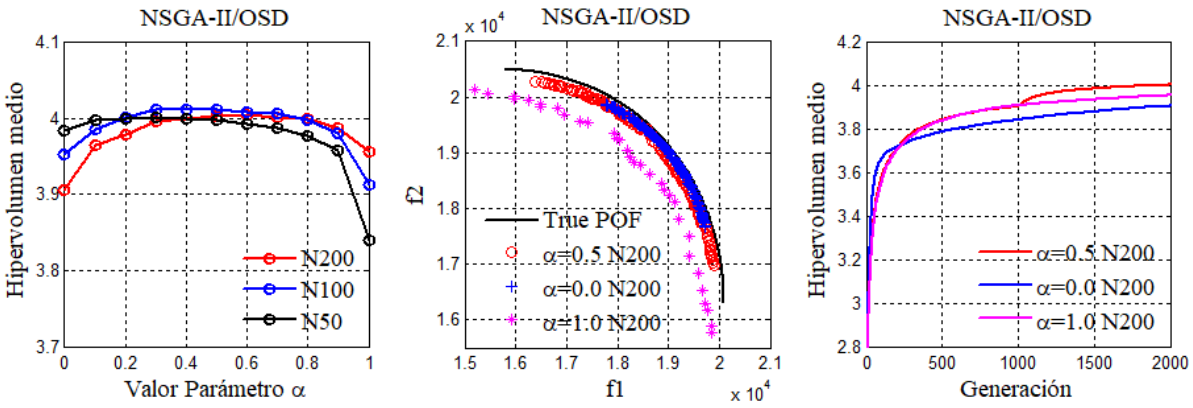


Figura 2. Valores H al variar el parámetro α en el algoritmo propuesto NSGA-II/OSD (izquierda). Gráficas de soluciones no-dominadas con el valor H más próximo al valor medio final en 30 ejecuciones (centro). Evolución del hipervolumen medio de los frentes mostrados en la gráfica central (derecha).

los algoritmos, el porcentaje de soluciones diferentes en la población N de soluciones es siempre mejor para el algoritmo propuesto, lo que justifica la mayor diversidad del frente de soluciones logrado por NSGA-II/OSD. También, el número de reparaciones por ensayo realizadas a los cromosomas de las soluciones de la población para que sean factibles, es menor con el método propuesto tal como se aprecia en la Fig. 5 (derecha).

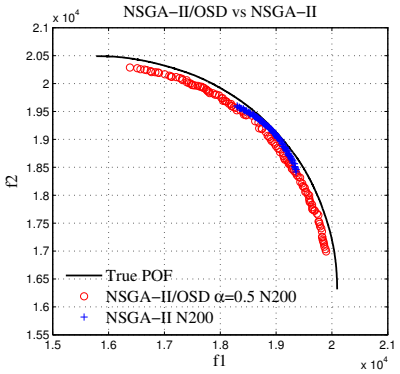


Figura 3. NSGA-II/OSD versus NSGA-II: gráficas de soluciones no-dominadas con el valor H más próximo al valor medio final en 30 ejecuciones.

IV-E. Análisis previo: MOEA/D e importancia de la especificación del punto de referencia PR

Dos especificaciones del punto de referencia -la de Zhang y la de Ishibuchi descritas en los apartados II-A y II-B- se comparan en esta subsección. La Fig. 6 (izquierda) muestra los valores del hipervolumen medio H al variar el valor α en la especificación de Zhang del punto de referencia en MOEA/D según la ecuación (3). El valor H crece hasta un valor máximo (cuando α =1.2) para luego volver a decrecer. También se pueden observar, en la Fig. 7 (izquierda), los efectos sobre la diversidad de los frentes de soluciones alcanzados cuando α=1.0 (mínima diversidad) y α=1.2 (máxima diversidad y máximo valor de H).

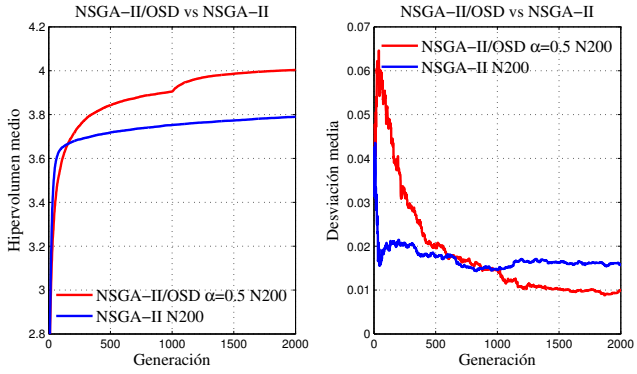


Figura 4. NSGA-II/OSD versus NSGA-II: Evolución de los valores de los hipervolumenes medio (iquierda) y desviaciones del hipervolumen (derecha).

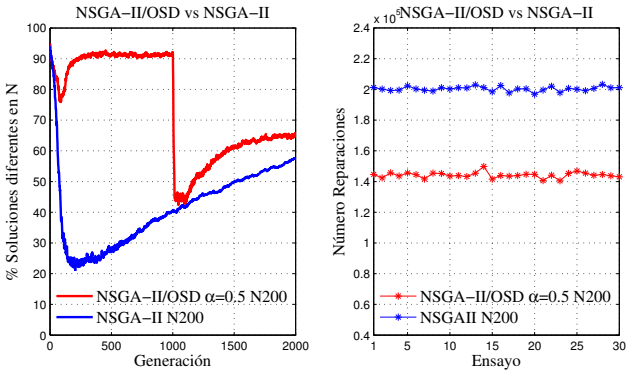


Figura 5. NSGA-II/OSD versus NSGA-II: Evolución de los porcentajes medios de soluciones diferentes en las poblaciones N (izquierda) y de los números de reparaciones de los cromosomas por ensayo (derecha).

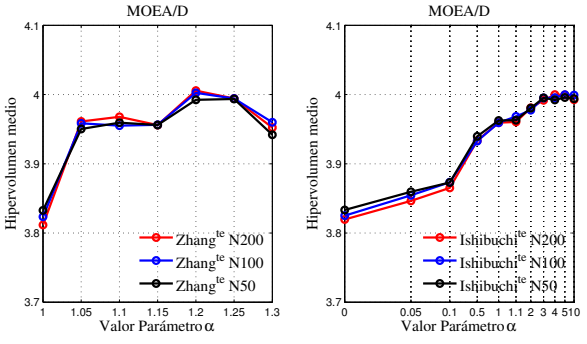


Figura 6. Valores H al variar el parámetro α en la especificación de Zhang del PR en MOEA/D (izquierda). Valores H al variar el parámetro α en la especificación de Ishibuchi del PR en MOEA/D (derecha).

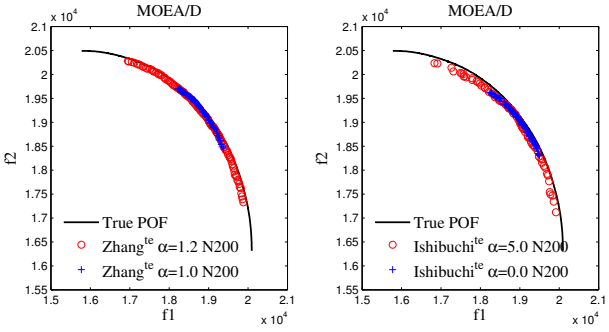


Figura 7. Gráficas de soluciones no-dominadas con el valor H más próximo al valor medio final en 30 ejecuciones: especificación de Zhang del PR (izquierda), especificación de Ishibuchi del PR (derecha).

Utilizando ahora en MOEA/D la especificación de Ishibuchi del punto de referencia según las ecuaciones (4) y (5), también al variar el valor α se obtienen diferentes valores del hipervolumen H. La Fig. 6 (derecha) señala que el valor H crece hasta un valor máximo (cuando $\alpha \geq 3$) a partir del cual permanece estable.

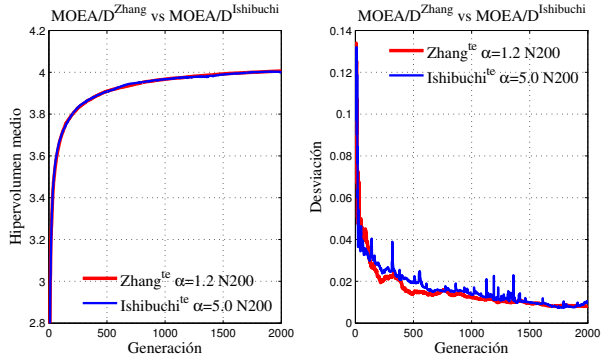


Figura 8. Evolución de los hipervolumenes medio (iquierda) y desviaciones del hipervolumen (derecha) usando las especificaciones de Zhang e Ishibuchi del PR en MOEA/D.

Una comparación de MOEA/D usando las especificaciones de Zhang con el valor α=1.2 e Ishibuchi con valores $\alpha \geq 3$, muestra valores parecidos de la métrica H (Fig. 8 izquierda)

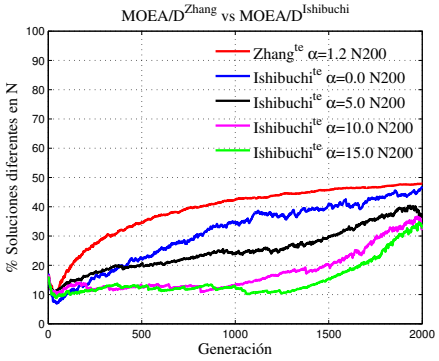


Figura 9. Evolución de los prcentajes medios de soluciones diferentes en las poblaciones N.

y de la desviación de H (Fig. 8 derecha) a lo largo de la evolución del algoritmo. No obstante, hay que señalar que el frente de soluciones logrado por MOEA/D con la especificación de Ishibuchi es más pobre (ver Fig. 7 derecha) en número de soluciones diferentes en la población N que el logrado usando la especificación de Zhang. En la Fig. 9 se muestra que utilizando la especificación de Zhang, MOEA/D consigue siempre en toda la evolución del algoritmo un mayor número de soluciones diferentes en la población N; por otro lado, en dicha gráfica también se observa que incrementando el valor α usando la especificación de Ishibuchi en MOEA/D, hay un decremento del número de soluciones diferentes en N.

IV-F. Resultados comparativos: NSGA-II/OSD vs MOEA/D

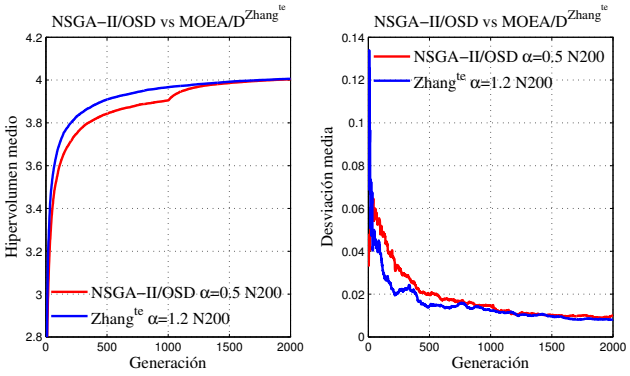


Figura 10. NSGA-II/OSD versus MOEA/D: Evolución de los valores de los hipervolumenes medio (iquierda) y desviaciones del hipervolumen (derecha).

En esta subsección comparamos el algoritmo NSGA-II/OSD propuesto en este trabajo y el algoritmo MOEA/D con especificación de Zhang del PR, utilizando en ambos algoritmos aquel valor α que proporcionó el mejor valor del hipervolumen medio H tras 30 ejecuciones independientes y que según los experimentos descritos en anteriores apartados fue α=1.2 para ambos algoritmos. En la Fig. 10 (izquierda) se observa que durante la primera mitad de la evolución de los algoritmos, MOEA/D tiene un mejor valor de H luego, los valores H obtenidos por ambos algoritmos se igualan justo a partir de

que las estrategias usadas con NSGA-II/OSD en las regiones R2 y R3 (Fig. 1) son a maximizar ambos objetivos. Idéntica conclusión se obtiene con las desviaciones del hipervolumen en la Fig. 10 (derecha).

De otra parte, en la Fig. 11 se observa que el frente de soluciones alcanzado con el método propuesto obtiene más diversidad que el alcanzado por MOEA/D con especificación de Zhang del PR, aunque en términos de convergencia es MOEA/D quien parece tener mejor valor. Estas diferencias se pueden explicar porque NSGA-II/OSD presenta siempre un mejor porcentaje de soluciones diferentes en la poblacion N durante la evolución del algoritmo tal como se ve en la Fig. 12 (izquierda). Finalmente, la Fig. 12 (derecha) muestra que NSGA-II/OSD realiza un menor número de reparaciones de cromosomas para factibilizar las soluciones.

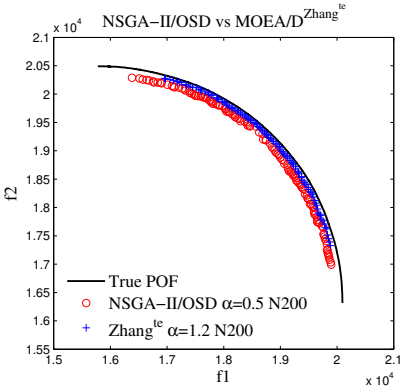


Figura 11. NSGA-II/OSD versus MOEA/D: gráficas de soluciones no-dominadas con el valor H más próximo al valor medio final en 30 ejecuciones.

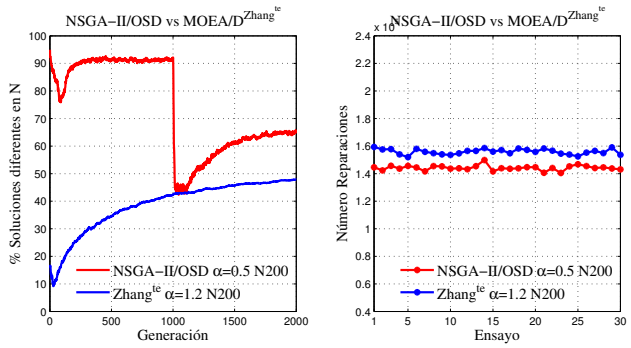


Figura 12. NSGA-II/OSD versus MOEA/D: Evolución de los porcentajes medios de soluciones diferentes en las poblaciones N (izquierda) y de los números de reparaciones de los cromosomas por ensayo (derecha).

V. Conclusiones

En este trabajo se propone un algoritmo evolutivo multi-objetivo (NSGA-II/OSD) que hibridiza NSGA-II. El algoritmo subdivide el espacio de los objetivos en varias regiones utilizando la solución Nadir calculada en cada generación del algoritmo. Las distintas subdivisiones utilizan diferentes estrategias de optimización de las funciones objetivo. El algoritmo NSGA-II/OSD propuesto permite definir un parámetro $\alpha \in [0, 1]$ muy fácil de usar, que permite balancear la diversidad/convergencia del frente de soluciones no-dominadas alcanzado por el algoritmo; para el valor α=1.2 el algoritmo logra su mejor valor de hipervolumen medio.

El algoritmo propuesto obtiene muy buen desempeño y mejores resultados cuando es comparado con NSGA-II. Cuando NSGA-II/OSD es comparado con MOEA/D con la especificación del punto de referencia de Zhang según (3), en términos de la métrica del hipervolumen medio, ambos algoritmos obtienen valores similares. No obstante, NSGA-II/OSD logra una mayor diversificación de soluciones en el frente no-dominado final alcanzado, mientras MOEA/D parece alcanzar mejor convergencia.

Agradecimientos

Agradecemos el soporte económico del Departamento de Informática y Sistemas de la Universidad de Las Palmas de Gran Canaria.

Referencias

[1] Carlos A. Coello Coello, Gary B. Lamont, David A. Van Veldhuizen. Evolutionary algorithms for solving multi-objective problems. Springer-Verlag Berlin, Heidelberg, 2006.

[2] K. Deb, A. Pratap, S. Agrawal, T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm : NSGA-II. IEEE Transactions on Evolutionary Computation, Vol. 6, n2, pages 182-197, 2002.

[3] X. Gandibleux, A. Freville. Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: the two objectives case. J. of Heuristics, 6(3): 361-383, 2000.

[4] Hisao Ishibuchi, Kaname Narukawa, Yusuke Nojima. Handling of Overlapping Objective Vectors in Evolutionary Multiobjective Optimization. International Journal of Computational Intelligence Research 1(2005): 1-18, 2005

[5] Hisao Ishibuchi, Naoya Akedo, Yusuke Nojima. Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. IEEE Transactions on Evolutionary Computation 19(2), 2015.

[6] Hisao Ishibuchi, Ken Doi, Yusuke Nojima. Reference Point Specification in MOEA/D for Multi-Objective and Many-Objective Problems. 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2016.

[7] R. Saborido, A.B. Ruiz, M. Luque. Global WASF-GA: An Evolutionary Algorithm in Multiobjective Optimization to approximate the whole Pareto Optimal Front. Evolutionary Computation 25(2): 309-349, 2017.

[8] Mingzhao Wang, Yuping Wang, Xiaoli Wang. A Space Division Multiobjective Evolutionary Algorithm Based on Adaptive Multiple Fitness Functions. International Journal of Pattern Recognition and Artificial Intelligence 30(3), 2016.

[9] Rui Wang, Jian Xiong, Hisao Ishibuchi, Guohua Wu, Tao Zhang. On the effect of reference point in MOEA/D for multi-objectiveoptimization. Applied Soft Computing 58(2017): 25-34, 2017.

[10] Yuping Wang, Chuangyin Dang. Improving Multiobjective Evolutionary Algorithm by Adaptive Fitness and Space Division. In: Wang L., Chen K., Ong Y.S. (eds) Advances in Natural Computation. ICNC 2005. LNCS, vol 3612. Springer, Berlin, Heidelberg, 2005.

[11] Zhenkun Wang, Qingfu Zhang, Hui Li, Hisao Ishibuchi, Licheng Jiao. On the use of two reference points in decomposition based multiobjective evolutionary algorithms. Swarm and Evolutionary Computation 34(2017): 89-102, 2017.

[12] Q. Zhang and H. Li. MOEA/D: A multi-objective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput., 11(6): 712-731, 2007.

[13] E. Zitzler, L. Thiele, *Multiobjective optimization using evolutionary algorithms - A comparative case study*, In A.E. Eiben et al. (Eds.): Parallel Problem Solving from Nature, LNCS, Volume 1498, pages 292-301, Springer, Heidelberg, 1998.

Algoritmo Evolutivo Multi-Objetivo para la Toma de Decisiones Interactiva en Optimización Dinámica

Cristóbal Barba-González
Lenguajes y Ciencias de la Computación
Universidad de Málaga
Málaga, España
cbarba@lcc.uma.es

José M. García-Nieto
Lenguajes y Ciencias de la Computación
Universidad de Málaga
Málaga, España
jnieto@lcc.uma.es

Ana B. Ruiz
Economía Aplicada (Matemáticas)
Universidad de Málaga
Málaga, España
abruiz@uma.es

Antonio J. Nebro
Lenguajes y Ciencias de la Computación
Universidad de Málaga
Málaga, España
antonio@lcc.uma.es

Mariano Luque
Economía Aplicada (Matemáticas)
Universidad de Málaga
Málaga, España
mluque@uma.es

José F. Aldana-Montes
Lenguajes y Ciencias de la Computación
Universidad de Málaga
Málaga, España
jfam@lcc.uma.es

***Resumen*—Debido al creciente interés en el análisis de datos en streaming en entornos Big Data para la toma de decisiones, cada vez es más común la aparición de problemas de optimización dinámica que involucran dos o más objetivos en conflicto. Sin embargo, los enfoques que combinan optimización dinámica multi-objetivo con la articulación de preferencias para la toma de decisiones son todavía escasos. En este artículo, proponemos un nuevo algoritmo de optimización dinámica multi-objetivo llamado InDM2, que permite incorporar preferencias del experto (humano) de cara a la toma de decisiones para guiar el proceso de búsqueda. Con InDM2, el decisor no solo puede expresar sus preferencias mediante uno o más puntos de referencia (que definen la región de interés deseada), sino que estos puntos también se pueden modificar de manera interactiva. La propuesta incorpora métodos para mostrar gráficamente las diferentes aproximaciones de la región de interés obtenidas durante el proceso de optimización. El decisor puede así inspeccionar y cambiar, en tiempo de optimización, la región de interés de acuerdo con la información mostrada. Las principales características de InDM2 son descritas y se analiza su funcionamiento mediante casos de uso académicos.**

***Index Terms*—Optimización Multi-Objectivo Dinámica, Toma de Decisiones, Selección de Preferencias, Algoritmos Interactivos**

I. Introducción

En la actualidad, aunque la mayoría de los esfuerzos se centran en abordar problemas de optimización multi-objetivo (MOP) estáticos, existe un creciente interés en aquellos escenarios en los que el problema cambia de alguna manera con el tiempo, es decir, MOP dinámicos o DMOPs [1].

Recientemente, han ido apareciendo varias propuestas de metaheurísticas para la resolución de DMOPs [2]–[5]. Al tratar con un DMOP, siempre que existan cambios en el entorno que afecten a las soluciones del problema (es decir, en la población o en el conjunto de soluciones en el frente de Pareto), el algoritmo de optimización debe reaccionar para adaptar la búsqueda a las nuevas características del problema [6]. Esto significa que las metaheurísticas de optimización dinámica multi-objetivo deben poder detectar cuándo cambia el problema y aplicar una estrategia para adaptarse a los cambios.

Sin embargo, analizar soluciones preferidas en el frente de Pareto aproximado y compararlas al mismo tiempo puede ser una tarea difícil en el proceso de toma de decisiones, especialmente en presencia de muchos objetivos. De hecho, puede ser computacionalmente costoso generar una gran cantidad de soluciones que aproximen todo el frente de Pareto, particularmente cuando se tratan problemas reales de la industria. Incluso, puede suponer un esfuerzo innecesario si el decisor está únicamente interesado en un subconjunto de soluciones ubicadas en un región particular.

Una forma de aliviar estos problemas es incorporar información sobre las preferencias del decisor en la metaheurística multi-objetivo, para enfocar progresivamente el proceso de búsqueda sobre aquel subconjunto de soluciones que corresponden a estas preferencias, conocido como la *región de interés*. Por lo tanto, en lugar de aproximar todo el frente de Pareto, la idea es aproximar sólo la región de interés.

Mientras que el manejo de preferencias ha sido ampliamente estudiado en su aplicación para MOPs estáticos en el campo de la toma de decisiones multi-criteria (*Multiple Criteria Decision Making* - MCDM) [7]–[9], apenas se ha explorado en la optimización dinámica de múltiples objetivos. En ese sentido, en un contexto donde el frente de Pareto puede cambiar con el tiempo, la especificación de preferencias que determinan la región de interés puede ser difícil debido al hecho de que el problema en sí cambia constantemente. Por lo tanto, una metaheurística de optimización dinámica multi-objetivo que incorpore preferencias, debería prestar atención especial a estas dos características simultáneamente: (1) visualización, para proporcionar al decisor una imagen gráfica de las aproximaciones de la región o regiones de interés de los frentes de Pareto que se encuentran durante el proceso de optimización; y (2) interactividad, para permitir que el decisor pueda indicar nueva información si quiere redefinir sus preferencias (es decir,

indicar una nueva región de interés) en base a su experiencia previa y al conocimiento que progresivamente adquiere sobre el problema al interactuar con el proceso de optimización.

Con esta motivación, proponemos en este artículo InDM2 (*Interactive Dynamic Multi-Objective Decision Making*), una nueva propuesta algorítmica que cumple con estos requisitos. InDM2 está diseñado para actuar como framework algorítmico bajo el que se puede incorporar cualquier metaheurística de optimización multi-objetivo basada en puntos de referencia, para la articulación de preferencias, delegando así en él el proceso de resolución del DMOP. Un punto de referencia está formado por valores de aspiración que el decisor desea alcanzar para cada una de las funciones objetivo.

En particular, hemos considerado una versión dinámica de dos algoritmos evolutivos basados en puntos de referencia: WASF-GA [10] y R-NSGA-II [11]. El proposito de WASF-GA es aproximar la región de interés determinada por el punto de referencia mediante el uso de una función escalarizada de logro [12] y un conjunto de vectores de pesos. Para ello, se clasifican los individuos en fronteras en base a los valores que alcanzan para dicha función. El algoritmo R-NSGA-II considera como información preferencial uno o varios puntos de referencia y modifica la distancia crowding y el operador niching de NSGA-II [13] para enfatizar las soluciones cercanas al/los punto(s) de referencia dado(s) por el decisor.

InDM2 incorpora un mecanismo que permite al decisor (si así lo desea) actualizar interactivamente el punto de referencia al resolver un DMOP. Además, ofrece un mecanismo para visualizar "sobre la marcha" las aproximaciones de la región de interés que se generan a lo largo del proceso de solución. Además, InDM2 se ha implementado con jMetalSP [14], por lo que todas las características de esta librería algorítmica están incorporadas en la propuesta, incluida la posibilidad de utilizar tecnologías Big Data, como Apache Spark. Esto permite, por ejemplo, procesar en streaming los datos almacenados en HDFS (el sistema de archivos Hadoop [15]), además de usar toda la potencia de los clústeres Spark.

Así, las principales contribuciones de este artículo son:

- Se presenta InDM2, un novedoso algoritmo que combina optimización dinámica multi-objetivo, toma de decisiones multi-criterio e interactividad. Una característica clave en InDM2 es el mecanismo para visualizar, en tiempo de optimización, las aproximaciones de la región de interés que se generan, junto con la posibilidad de indicar puntos de referencia para guiar estas aproximaciones.
- La propuesta se ha implementado dentro del framework jMetalSP, por lo que se puede utilizar para optimización en contextos Big Data, aprovechando las características del procesamiento de datos en streaming distribuido de Apache Spark. Esta implementación está disponible como código abierto en repositorio GitHub de jMetalSP[1].
- Se presentan dos casos de estudio para ilustrar el funcionamiento de InDM2 en la resolución de problemas dinámicos del benchmark FDA para DMOPs [6]. En concreto, se utilizan las metaheurísticas interactivas WASF-GA y R-NSGA-II como optimizador base para los dos ejemplos sobre el problema FDA2.

El resto del artículo se organiza de la siguente forma. La Sección II está dedicada a describir conceptos previos necesarios para la explicación de la propuesta. En la Sección III, se describen la estructura y los principales componentes de InDM2. La Sección IV explica los experimentos realizados a modo de caso de uso. Finalmente, la Sección V recoge las principales conclusiones y trabajo futuro.

II. Conceptos Previos

Una forma común de expresar preferencias consiste en indicar valores deseables para las funciones objetivo, que constituyen las componentes de lo que se conoce como punto de referencia. Este concepto se introdujo en [12] para MOP estáticos, aunque se puede adaptar fácilmente a DMOP.

Para un DMOP con k funciones objetivo, en un instante de timpo t, un *punto de referencia* viene dado por $\mathbf{q}_t = (q_{t,1}, \ldots, q_{t,k})^T$, donde $q_{t,i}$ es un nivel de aspiración para la función objetivo $f_i(\cdot, t)$ proporcionada por el decisor ($i = 1, \ldots, k$). Por lo general, se dice que $\mathbf{q}_t$ es *alcanzable* si existe, al menos, una solución factible del problema t cuyo vector objetivo iguala o domina $\mathbf{q}_t$. De lo contrario, se dice que el punto de referencia es *inalcanzable*, es decir, no todos los niveles de aspiración se pueden alcanzar o dominar simultáneamente.

Para un punto de referencia $\mathbf{q}_t$, se puede minimizar la *función escalarizada de logro* propuesta en [12] sobre el conjunto factible, para encontrar la solución óptima de Pareto que mejor satisfaga las expectativas del decisor en el instante t. Dado un vector de pesos en el instante t, $\mu_t = (\mu_{t,1}, \ldots, \mu_{t,k})^T$ con $\mu_{t,i} > 0$ ($i = 1, \ldots, k$), y un valor real pequeño $\rho \geq 0$, la función de logro propuesta en [12] viene dada por:

$$s(\mathbf{q}_t, \mathbf{f}(\mathbf{x}, t), \mu_t) = \max_{i=1,\ldots,k} \{ \mu_{t,i}(f_i(\mathbf{x}, t) - q_{t,i}) \} + \rho \sum_{i=1}^{k} \mu_{t,i}(f_i(\mathbf{x}, t) - q_{t,i}).$$

III. Propuesta Algorítmica: InDM2

El componente principal de InDM2 es una metaheurística multi-objetivo dinámica basada en preferencias, que actúa como motor de optimización. Una descripción más detallada de la implementación se puede encontrar en [16]. Para este trabajo, se han implementado nuevas versiones dinámicas de los algoritmos WASF-GA y R-NSGA-II de la siguente manera.

La arquitectura orientada a objetos de jMetalSP permite reutilizar la plantilla de algoritmos evolutivos proporcionada por jMetal[2]. Esta plantilla tiene un método `run()` que codifica el pseudo-código de un algoritmo evolutivo genérico. Las implementaciones de las versiones dinámicas de WASF-GA y R-NSGA-II siguen esta plantilla, por lo

[1] https://github.com/jMetal/jMetalSP

[2] https://jmetal.github.io/jMetal/

que el desarrollo de InDM2 sobre WASF-GA y R-NSGA-II requiere simplemente la redefinición de dos métodos. En primer lugar, se debe adaptar el comportamiento del método `isStoppingConditionReached()`, ya que en lugar de simplemente terminar el algoritmo, este método debe hacer que la aproximación de la región de interés esté disponible para los métodos consumidores de datos de InDM2. De esta forma, el algoritmo de optimización subyacente (basado en WASF-GA o R-NSGA-II) pueda comenzar de nuevo teniendo en cuenta esta información. En segundo lugar, al final de cada iteración, el método `updateProgress()` incrementa un contador de evaluación.

Para permitir que el decisor pueda actualizar el(los) punto(s) de referencia "sobre la marcha", InDM2 lee el(los) nuevo(s) punto(s) de referencia desde teclado (aunque podrían usarse otros métodos más sofisticados). Este evento se detecta mediante el método `updateProgress()`. Para ayudar al decisor a dar un nuevo punto de referencia de acuerdo con sus preferencias y teniendo en cuenta los resultados obtenidos, se ha incluido un consumidor de datos gráfico. Por lo tanto, las aproximaciones de la región de interés generadas por InDM2 se muestran cuando se producen, por lo que el decisor puede visualizarlas y cambiar los puntos de referencia como desee.

Por defecto, InDM2 siempre está funcionando. Esto significa que, si el DMOP no cambia durante un período de tiempo, el algoritmo lo resolverá una y otra vez, utilizando exactamente la misma configuración del problema. Como consecuencia, si el algoritmo converge la mayoría de las veces, la misma aproximación de la región de interés (o una muy similar, dado que estamos usando enfoques estocásticos) se mostrará constantemente. Esto puede ser confuso para el decisor, ya no percibirá cambios evidentes por un período de tiempo. Para evitar esta situación, el consumidor de datos gráficos solo muestra una nueva aproximación en caso de que ésta haya cambiado significativamente, en comparación con la anterior mostrada. Para verificar esto, se calcula internamente el indicador distancia generacional invertida (I_{IGD}) [17] desde la aproximación de la región de interés actual, hasta la anterior. La nueva aproximación se mostrará únicamente si el valor de este indicador es inferior a un umbral τ.

Todos los DMOP disponibles en jMetalSP tienen un método para averiguar si el problema ha cambiado o no. Por lo tanto, InDM2 solo tiene que llamar a este método al final de cada iteración (es decir, en el método `updateProgress()`), para saber si el parámetro del problema ha cambiado.

Una vez que se ha detectado un cambio en el problema y/o punto(s) de referencia, un aspecto clave en InDM2 es la estrategia de reinicio. Hemos adoptado un enfoque flexible que consiste en definir el reinicio de acuerdo con otras dos subestrategias: una para eliminar soluciones de la población y otra para llenar la población con nuevas soluciones. Para llenar la población, dado un valor $N' > 0$, la estrategia seguida es crear al azar N' nuevas soluciones. Para eliminar las soluciones, se pueden configurar las siguentes estrategias:

- Eliminar las primeras N' soluciones.
- Eliminar N' soluciones aleatorias.
- Eliminar las peores N' soluciones de acuerdo a la distancia crowding.
- Eliminar las peores N' soluciones de acuerdo a su contribución en hypervolumen con respecto a la última aproximación de la región de interés.

El pseudo-código del InDM2 se muestra en el Algoritmo 1. Después de la fase de inicialización (líneas 1-14), el algoritmo inicia el ciclo infinito (línea 15) que lleva a cabo múltiples rondas consecutivas de optimización dinámica y consumo de datos resultado (visualización). Cada ronda de optimización (líneas 16-25) implica un número máximo de iteraciones (G_{max}) en las que se ejecuta el algoritmo evolutivo basado en preferencias (es decir, WASF-GA o R-NSGA-II, línea 17), y se invocan los procedimientos de reinicio cada vez que se detecta un cambio en el punto de referencia (línea 18) o en el estado del problema (línea 20). Al final de cada iteración, la aproximación de la región de interés encontrada se envía a los consumidores de datos (línea 25) y el ciclo comienza de nuevo.

Algorithm 1 Pseudo-código InDM2

1: N ; // Tamaño de población
2: N' ; // Número de soluciones a reemplazar
3: G_{max} ; // Número máximo de generaciones
4: c, m ; // Operadores genéticos (cruze y mutación)
5: $t \leftarrow 0$; // Contador de generación
6: A_t ; // Estado del problema de optimización
7: $\mathbf{q}_t$; // Punto(s) de referencia inicial(es)
8: P ; // Algoritmo de optimización basado en preferencias
9: φ_q ; // Estrategia de reinicio ante cambios del punto de referencia
10: φ_p ; // Estrategia de reinicio ante cambios del problema
11: $M \leftarrow$ {WASF-GA, R-NSGA-II} ; // Optimizadores base
12: $P_t \leftarrow initializePopulation(N)$;
13: $evaluate(P_t, A_t)$;
14: $E_t \leftarrow initializeParetoSet(P_t)$;
15: **while** true **do**
16: **while** $t < G_{max}$ **do**
17: $(P_{t+1}, E_{t+1}) \leftarrow compute(M, \mathbf{q}_t, c, m, P_t, A_t)$;
18: **if** $\mathbf{q}_{t+1} \neq \mathbf{q}_t$ **then**
19: $P'_{t+1} \leftarrow restart(P_{t+1}, \varphi_q, \mathbf{q}_{t+1}, N')$;
20: **else if** $A_{t+1} \neq A_t$ **then**
21: $P'_{t+1} \leftarrow restart(P_{t+1}, \varphi_p, A_{t+1}, N')$;
22: **end if**
23: $t \leftarrow t + 1$;
24: **end while**
25: $C : (E_{t+1}, \tau)$; // Notifica aproximación a consumidores
26: $t \leftarrow 0$;
27: **end while**

IV. Experimentos

El objetivo de este estudio no es tanto la realización de comparativas exhaustivas de rendimiento algorítmico, sino el de ilustrar cómo funciona InDM2 en la práctica, cuando interactúa con un decisor, para así mostrar los beneficios potenciales de la propuesta desde el punto de vista de la toma de decisiones.

En este sentido, se ha probado InDM2 con el problema FD2, introduciendo manualmente las preferencias (es decir, los puntos de referencia) durante el tiempo de ejecución. La

Tabla I: Configuración de InDM2.

Parámetros	
Tamaño de población	50
Probabilidad de cruze	0.9
Probabilidad de mutación	$1/L$ (L = número de variables)
Número máximo de evaluaciones	25,000
Reinicio por punto de referencia	Eliminar 100 % de las soluciones Generar 100 % de soluciones aleatoriamente
Reinicio por cambio de problema	Eliminar 50 % de soluciones (hypervolumen) Generar 50 % de soluciones aleatoriamente
Comsumidores de datos	
Visualizador gráfico	Muestra frentes y puntos de referencia
Front writer	Carga los frentes generados en ficheros
Streaming runtime	
Mecanismo	*Java threads*

idea es observar cómo InDM2 puede adaptar el proceso de optimización a los cambios (tanto en la configuración del problema como en las preferencias), mostrando así el efecto de estos cambios en las aproximaciones de los frentes de Pareto obtenidas. Nos referimos a las dos configuraciones de InDM2 para los optimizadores base WASF-GA y R-NSGA-II como InDM2$_W$ y InDM2$_R$, respectivamente.

El problema FD2 es un DMOPs de representación continua perteneciente al benchmark FDA [6]. Cada problema de esta familia tiene dependencia con el tiempo t, que se define mediante la ecuación $t = \frac{1}{n_t}\left\lfloor \frac{\sigma}{\sigma_T} \right\rfloor$, en la que σ es un contador de generaciones, σ_T es el número de generaciones en las que t permanece fijo y n_t es la cantidad de pasos distintos en t. Para un número real a, $\lfloor a \rfloor$ denota el entero más grande no mayor a a. En [6], los autores recomiendan establecer $\sigma_T = 5$ y $n_t = 10$.

La formulación de FDA2 se puede encontrar en [6], por lo que siguiendo lo recomendado en este trabajo, se han utilizado soluciones de 31 variables con $x_I = x_1$, $|x_{II}| = |x_{III}| = 15$. Este problema se ha seleccionado debido a que las aproximaciones de frentes de Pareto y sus conjuntos de Pareto cambian en el tiempo (tipo II [6]). Por lo tanto, los cambios dinámicos en las aproximaciones causadas por cambios en el punto de referencia se pueden visualizar fácilmente.

La Tabla I muestra la configuración de parámetros utilizada para InDM2. La condición de parada utilizada para informar de los resultados a los procesos consumidores es alcanzar un máximo de 25,000 evaluaciones. Los operadores de variación son SBX como operador de cruce y el de mutación polinómica. El operador de selección es torneo binario. Para la estrategia de reinicio al detectar un cambio en el problema, en primer lugar, se elimina el 50 % de las soluciones de acuerdo con la contribución de hipervolumen; en segundo lugar, se llena la población con nuevas soluciones creadas aleatoriamente. La estrategia de reinicio cuando se proporciona un nuevo punto de referencia es similar, pero en este caso se eliminan todas las soluciones de la población (100 %).

Para generar los eventos temporales que conducirán a cambios en el problema FDA2, hemos utilizado una fuente simple de transmisión de datos que cuenta cada segundo, lo que implica que, con un valor predeterminado de σ_T, el problema se actualiza con una frecuencia de 5 segundos. Además, hemos utilizado dos consumidores de datos, uno para

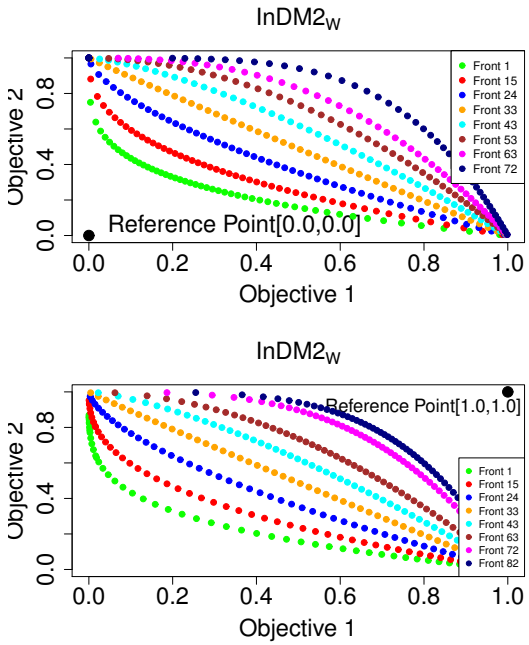


Figura 1: Aproximaciones de región de interés encontradas por InDM2$_W$ para el problema FD2, utilizando los puntos de referencia (0.0, 0.0) (arriba) y (1.0, 1.0) (abajo).

mostrar los frentes de Pareto aproximados que se generan y otra para guardar estas aproximaciones en archivos. En este estudio, solo se considera una fuente de datos, es decir, no hay ninguna ventaja de utilizar Apache Spark, por lo que hemos configurado el tiempo de ejecución en hebras de manera predeterminada.

El problema de optimización dinámica FDA2 se define con dos objetivos y frentes de Pareto que cambian de forma convexa a cóncava y vice-versa, con conjuntos de Pareto que también cambian con el tiempo.

Para tener una idea de los posibles valores de aspiración (deseables) que cada objetivo puede tener durante el proceso de búsqueda, se ha ejecutado inicialmente InDM2$_W$ utilizando dos puntos de referencia: el punto ideal (0.0, 0.0) (inalcanzable) y el punto nadir (1.0, 1.0) (alcanzable). Las aproximaciones producidas por InDM2$_W$ para las regiones de interés de ambos puntos de referencia se muestran en la Figura 1. En esta figura, las leyendas "*Front*$_i$" marcan las aproximaciones de la región de interés que se muestran. Cada una de estas leyendas indica implícitamente en qué iteración i se detectó una mejora significativa en la aproximación generada, en comparación con la anterior mostrada (como se explicó en la Sección III, una nueva aproximación se muestra gráficamente solo en caso de que se haya detectado una mejora significativa con respecto al indicador I_{IGD}).

En la Figura 1, podemos observar que cuando la región de interés aproximada es convexa, la distribución de las soluciones en las aproximaciones encontradas es ligeramente más densa en las regiones centrales que en las extremas,

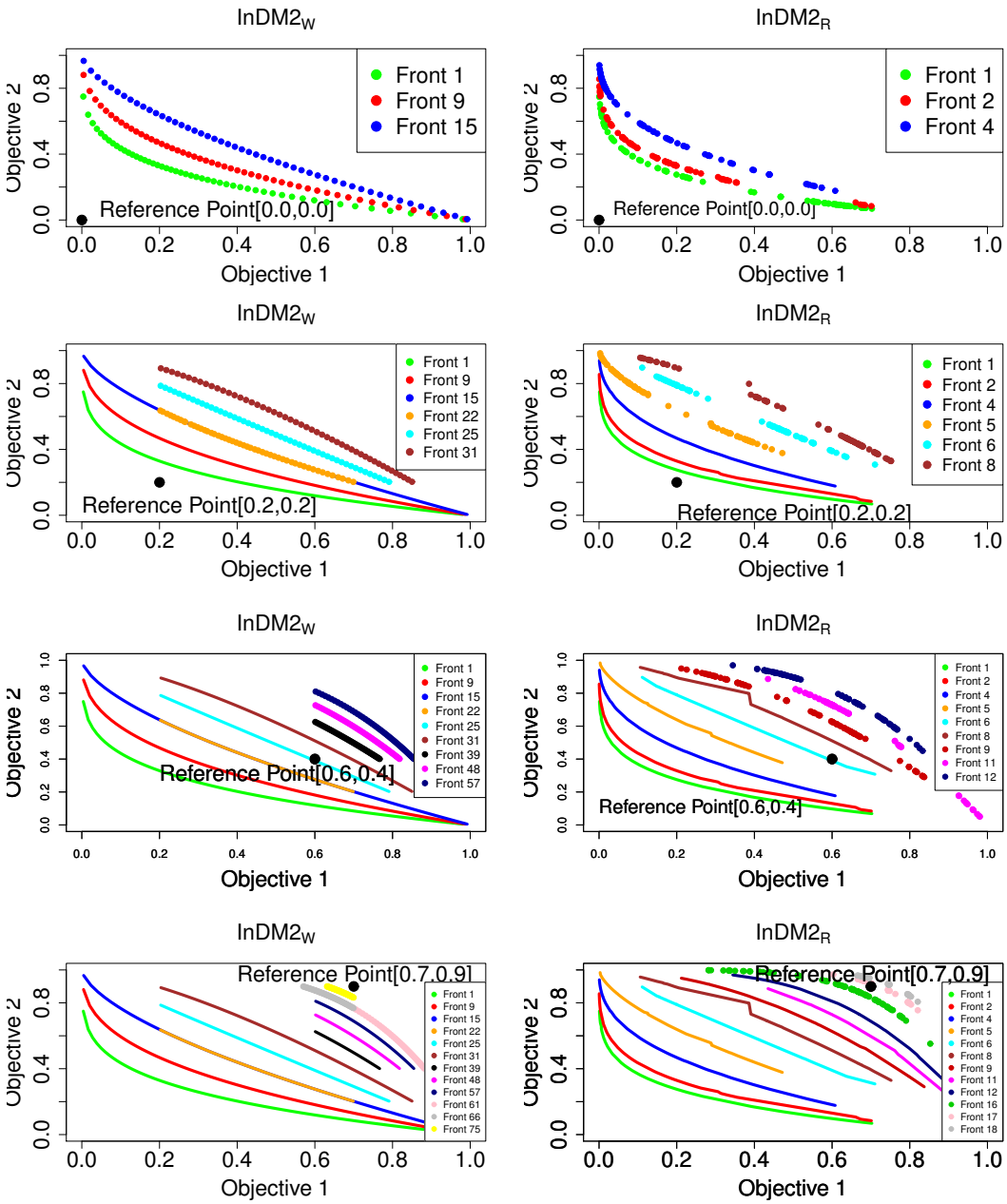


Figura 2: Aproximaciones de la región de interés encontradas por InDM2$_W$ (izquierda) e InDM2$_R$ (derecha) para el problema FDA2, utilizando diferentes puntos de referencia.

extendiéndose de manera uniforme cuando la región aproximada tiende a ser más lineal. Esto viene determinado por la distribución de las direcciones de proyección (es decir, vectores de peso) utilizadas internamente en WASF-GA, lo que influye en la distribución de las soluciones encontradas. Para ambos puntos de referencia, InDM2$_W$ ha podido aproximar la región de interés (que, como se puede ver, es el frente de Pareto completo en ambos casos) al adaptar el proceso de búsqueda a los cambios en el problema.

A continuación, se realizaron pruebas con InDM2$_W$ e InDM2$_R$ para FDA2, pero en este caso, se simuló un escenario realista en el que el decisor modifica sobre la marcha el punto de referencia para indicar la región deseada, varias veces durante el proceso de optimización. La Figura 2 muestra las aproximaciones generadas para los puntos de referencia indicados a lo largo del proceso de optimización de InDM2, para los dos algorithmos base: InDM2$_W$ y InDM2$_R$.

En el proceso interactivo de optimización seguido por InDM2$_W$, suponiendo que los rangos del frente de Pareto para problema FDA2 son desconocidos, el decisor seleccionó inicialmente (0.0, 0.0) como primer punto de referencia. Después de observar las tres primeras aproximaciones devueltas por

InDM2$_W$ en las iteraciones 5, 9 y 15 (véase la gráfica superior izquierda en la Figura 2), el decisor decidió dar un nuevo punto de referencia para reducir la región de interés aproximada y establecerlo en (0.2, 0.2). Podemos observar las nuevas aproximaciones generadas en el segundo gráfico a la izquierda en la Figura 2. En este gráfico, las primeras aproximaciones generadas para el punto de referencia inicial (0.0, 0.0) se muestran con líneas continuas, para poder distinguirlas de las nuevas aproximaciones encontradas para el nuevo punto de referencia (0.2, 0.2), que se muestran con líneas punteadas. Más tarde, el decisor volvió a ajustar la región de interés cambiando el punto de referencia dos veces, indicando (0.6, 0.4) como tercer punto de referencia y (0.7, 0.9) como el cuarto. Respectivamente, los resultados obtenidos para estos dos nuevos puntos de referencia se muestran en las gráficas tercera y cuarta a la izquierda en la misma figura.

Respecto a los frentes obtenidos por InDM2$_R$ en las gráficas de la derecha en la Figura 2, se puede observar cómo la aproximación de las regiones de interés producidas por esta variante tienen peor diversidad que las generadas por InDM2$_W$. Nótese que las soluciones generadas por InDM2$_W$ siempre pertenecen a la región de interés delimitada por los puntos de referencia, mientras que algunas de las encontradas por InDM2$_R$ también aproximan zonas fuera de esta región. Esto es consecuencia de las capacidades de búsqueda (convergencia y diversidad con respecto a la región de interés) de WASF-GA y R-NSGA-II, lo que evidencia el impacto que puede tener el algoritmo evolutivo basado en preferencias utilizado como base en InDM2.

Según la Figura 2, podemos ver que tres de los puntos de referencia dados en el proceso interactivo de optimización son inalcanzables: (0.0, 0.0), (0.2, 0.2) y (0.6, 0.4); y uno de ellos es alcanzable (0.7, 0.9). Cabe señalar que, en la práctica, la alcanzabilidad de los puntos de referencia no puede conocerse de antemano, y solo se puede saber si son alcanzables o no cuando el algoritmo genera soluciones y se verifica su relación de dominancia con respecto al punto de referencia utilizado.

V. Conclusiones

En este trabajo se ha descrito InDM2, un nuevo framework algorítmico de optimización dinámica que permite incorporar cualquier metaheurística multi-objetivo basada en puntos de referencia para la articulación de preferencias. Se ha mostrado su funcionamiento mediante la resolución de un problema de optimización dinámica multi-objetivo de benchmarking (FD2). A partir de una serie de visualizaciones, se ha demostrado cómo InDM2 es capaz de generar aproximaciones ajustadas a las preferencias del decisor en tiempo real, tanto si cambia la configuración del problema como si se indican nuevos puntos de referencia (es decir, cambia la región de interés).

Una característica importante de InDM2 viene heredada de su implementación en el framework de jMetalSP, por lo que dispone de procesamiento en streaming mediante Spark de manera transparente. Esto nos permite incorporar multiples fuentes de datos mediante el uso de proveedores, como sockets, archivos en un directorio, Kafka, Flume o Kinesis. Otra consecuencia del uso de Spark es la ventaja de trabajar con tecnologías Big Data, lo que nos permite desplegar InDM2 en clusters Hadoop y acceder a datos en HDFS.

Como trabajo futuro, se pretende estudiar el rendimiento de InDM2 en problemas complejos del mundo real, usando preferencias y cambiándolas interactivamente por un decisor experto en el dominio de aplicación.

Agradecimientos

Este trabajo ha sido financiado por los proyectos TIN2017-86049-R, TIN2014-58304-R, ECO2017-88883-R (MECD) y P12-TIC-1519 (PAIDI). J. M. García-Nieto y A. B. Ruiz disfrutan una beca Post-Doctoral "Captación de Talento para la Investigación", Universidad de Málaga. C. Barba-González dispone de un beca pre-doctoral BES-2015-072209.

Referencias

[1] C. Raquel and X. Yao, "Dynamic multi-objective optimization: A survey of the state-of-the-art," in *Evolutionary Computation for Dynamic Optimization Problems (2013)*, S. Yang and X. Yao, Eds. Springer, 2013, pp. 85–106.

[2] S. Jiang and S. Yang, "A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 65–82, 2017.

[3] X. Chen, D. Zhang, and X. Zeng, "A stable matching-based selection and memory enhanced MOEA/D for evolutionary dynamic multiobjective optimization," in *Int. Conf. on Tools with AI*, 2015, pp. 478–485.

[4] Y. Jin, C. Yang, J. Ding, and T. Chai, "Reference point based prediction for evolutionary dynamic multiobjective optimization," in *Proc. of the IEEE Cong. on Evol. Comp*, 2016, pp. 3769–3776.

[5] A. Muruganantham, K. C. Tan, and P. Vadakkepat, "Evolutionary dynamic multiobjective optimization via kalman filter prediction," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2862–2873, 2016.

[6] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: test cases, approximations, and applications," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 5, pp. 425–442, 2004.

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

[8] A. Jaszkiewicz and J. Branke, "Interactive multiobjective evolutionary algorithms," in *Multiobjective Optimization, Interactive and Evolutionary Approaches*, ser. LNCS, J. Branke, K. Deb, K. Miettinen, and R. Slowinski, Eds., vol. 5252. Springer, 2008, pp. 179–193.

[9] M. Helbig, K. Deb, and A. Engelbrecht, "Key challenges and future directions of dynamic multi-objective optimisation," in *Proc. of the IEEE Cong. on Evol. Comp.*, 2016, pp. 1256–1261.

[10] 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. 62, no. 1, pp. 101–129, 2015.

[11] K. Deb, J. Sundar, B. Ubay, and S. Chaudhuri, "Reference point based multi-objective optimization using evolutionary algorithm," *Int. Jour. of Comp. Intel. Res.*, vol. 2, no. 6, pp. 273–286, 2006.

[12] 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. Springer, 1980, pp. 468–486.

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

[14] C. Barba-Gonzalez, J. M. Garcia-Nieto, A. J. Nebro, J. A. Cordero, J. J. Durillo, I. Navas-Delgado, and J. F. Aldana-Montes, "jMetalSP: A framework for dynamic multi-objective big data optimization," *Applied Soft Computing,*, vol. to appear, 2017.

[15] T. White, *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 2009.

[16] A. J. Nebro, A. B. Ruiz, C. Barba-González, J. García-Nieto, M. Luque, and J. F. Aldana-Montes, "InDM2: Interactive dynamic multi-objective decision making using evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 40, pp. 184 – 195, 2018.

[17] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, *Modified Distance Calculation in Generational Distance and Inverted Generational Distance*. Springer International Publishing, 2015, pp. 110–125.