Optimización evolutiva multiobjetivo distribuida mediante aplicación selectiva de operadores

Pablo García-Sánchez
Departamento de Ingeniería Informática
Universidad de Cádiz
Puerto Real, España
pablo.garciasanchez@uca.es

Julio Ortega, Jesús González, Pedro A. Castillo,
Juan Julián Merelo, Antonio M. Mora y Antonio Fernández-Ares
Departamento de Arquitectura y Tecnología de Computadores
Universidad de Granada
Granada, España
julio@ugr.es

***Resumen*—Este trabajo presenta un nuevo enfoque, llamado *aplicación selectiva de operadores*, para hacer frente a problemas no separables en el entorno de los algoritmos co-evolutivos multiobjetivo: en lugar de compartir secciones de individuos, cada procesador aplica los operadores de variación a un subconjunto específico de todo el individuo. Esto evita pasos adicionales para recomponer los individuos completos de otras islas antes de ser evaluados.**

En este trabajo se pretende demostrar que la decisión automática del tamaño de la sección solapada, es capaz de obtener mejores resultados que la utilización del mismo tamaño independientemente del número de islas. Para ello se ha comparado con otras técnicas evolutivas colaborativas considerando diferentes números de islas y problemas. El análisis de los resultados experimentales obtenidos, utilizando diferentes métricas, muestra que nuestro enfoque puede proporcionar mejoras estadísticamente significativas con respecto al algoritmo base en problemas multiobjetivo de alta dimensionalidad.

***Index Terms*—Algoritmos multiobjetivo, NSGA-II, modelo de islas, algoritmos evolutivos distribuidos**

I. Introducción

Los problemas de optimización multiobjetivo (MOP por sus siglas en inglés) son aquellos en los que varios objetivos tienen que ser optimizados a la vez [1]. Resolver un MOP implica optimizar una función compuesta por varias funciones de coste independientes, una por objetivo. En estos problemas el objetivo es obtener un conjunto de soluciones que sean mejores que el resto, considerando todos los objetivos; este conjunto se conoce como el Frente de Pareto (FP). Las soluciones en este conjunto son *no dominadas*, lo que significa que no hay otra solución que sea igual o mejor para todos los objetivos.

Este aspecto de la búsqueda de soluciones no dominadas, así como, en muchos casos, el tamaño del propio espacio de búsqueda, implica una alta demanda de tiempo computacional, lo que lleva a la propuesta de métodos paralelos y distribuidos para resolverlos [2], [3], uno de los cuales son los algoritmos evolutivos (EAs) [4].

Trabajo financiado por los proyectos SPIP2017-02116 (Dirección General de Tráfico), EphemeCH TIN2014-56494-C4-3-P, TIN2015-67020-P, DeepBio TIN2017-85727-C4-2-P y TEC2015-68752 (Ministerio de Economía y Competitividad y Fondos FEDER) y PR2018-056 (Programa de Fomento e Impulso de la Investigación y de la Transferencia de la Universidad de Cádiz 2018-2019).

Los EAs se han extendido a la optimización multiobjetivo a través de un buen número de Algoritmos Evolutivos Multiobjetivo (MOEAs, por sus siglas en inglés) [5]. Se han utilizado diferentes enfoques para paralelizar los EAs, ya que cada individuo puede ser considerado como una unidad independiente [6], [7]. Los métodos clásicos, como los EAs paralelos globales (Maestro-Esclavo), o los algoritmos espacialmente estructurados (Modelo de isla o EAs celulares) se han aplicado con éxito en el pasado [8]. Sin embargo, en el caso de los MOEAs, estos enfoques [2] necesitan ocuparse de todo el conjunto de soluciones, el FP. Esto implica el uso de diferentes mecanismos de distribución y compartición, ya que existe un equilibrio entre la mejora obtenida a partir de la paralelización y la necesidad de recombinar globalmente los resultados para identificar con precisión el FP [9].

Las MOPs del mundo real normalmente requieren un alto número de variables de decisión, lo que significa que los MOEAs necesitan tratar con individuos grandes y gastar un tiempo significativo adicional para el cruce, mutación y migración. Diferentes autores han propuesto métodos para dividir el espacio de decisión (el cromosoma) para mejorar el rendimiento y la calidad de las soluciones. En este aspecto, el modelo de co-evolución es un modelo distribuido por dimensiones en el que un problema de alta dimensionalidad se divide en otros de dimensiones más bajas [7], que evolucionan por separado. Un ejemplo de aplicación de esta técnica fue descrito en [10]. El método presentado en ese trabajo involucra a diferentes trabajadores que evolucionan subpoblaciones creadas y recombinadas por un proceso maestro, el cual realiza diferentes alternativas de recombinación de las partes devueltas por los procesos de los trabajadores.

Un enfoque similar utilizado para resolver este tipo de problemas es el de la aplicación de la Aplicación Selectiva de Operadores (ASO) presentado en este documento. En este caso, cada isla se ocupa de la totalidad del cromosoma, pero sólo modifica un fragmento en la fase de cruce y mutación en función del número de islas, utilizando todo el cromosoma para el cálculo del fitness. Esto permite hacer frente a problemas que no se pueden descomponer. En nuestro trabajo anterior [11] utilizamos este método de manera preliminar. En ese trabajo demostramos que la aplicación de los operadores de variación sólo sobre secciones específicas de todo el cromosoma mejora

la calidad de las soluciones en el mismo tiempo de cálculo para un algoritmo multi-objetivo basado en islas. Además, en lugar de hacer que cada isla se centre en un subconjunto disjunto del cromosoma, el uso de secciones solapadas (compartidas) del cromosoma puede mejorar la calidad de las soluciones cuando se aumenta el número de islas.

Esto nos motiva a continuar esta línea de investigación utilizando un entorno más adecuado: un cluster real con hasta 128 nodos y una parametrización más completa. Además de comparar los métodos anteriores en esta nueva configuración experimental, en este trabajo proponemos un nuevo método que automáticamente establece el tamaño del solapamiento. en función del número de islas disponibles, comparándolo con versiones anteriores.

El resto del documento está organizado de la siguiente manera: después del estado del arte en MOEAs distribuidos y co-evolutivos, la metodología utilizada y los algoritmos comparados se describen en la Sección III. Después se presentan los resultados de los experimentos (Sección V), Finalmente, se discuten las conclusiones y el trabajo futuro [1].

II. Estado del arte

Desde principios de la década de los 2000, los EAs distribuidos y paralelos se han utilizado principalmente para aprovechar sistemas como clusters o grids [12]. Pero en el caso de los MOEAs, la distribución y paralelización es más difícil que en los EAs con un solo objetivo. Esto se debe a que en diferentes pasos del algoritmo el conjunto completo de soluciones dependientes, el FP, debe ser gestionado como un todo, dedicando tiempo a reunir a todos los individuos de los diferentes procesadores o islas.

Para resolver este problema, algunos autores han propuesto el uso de enfoques Maestro-Esclavo. Por ejemplo, [13] comparó diferentes enfoques maestro-esclavo: síncrono generacional, asíncrono generacional y asíncrono estacionario, siendo esta última la opción más prometedora.

También se ha explorado otro tipo de enfoques. El trabajo de Deb et al. [14] fue uno de los primeros enfoques para MOEAs distribuidos (dMOEAs). En ese trabajo, el dominio de las soluciones se divide en las islas mediante una transformación de coordenadas. En ese trabajo, los autores concluyeron que dividir el espacio de búsqueda es una buena idea, aunque lograr esto no es trivial. La división del espacio de búsqueda ha sido explorada por otros investigadores, por ejemplo, dividiendo la población en élites y subpoblaciones de búsqueda [15], o separándola en procesadores por objetivo [16]. Otros autores, como [17] utilizan la migración para aceptar individuos basados en la diversidad, y emigran desde áreas no superpobladas.

Abordar este tipo de problemas usando co-evolución cooperativa también ha sido estudiado en varios trabajos con enfoques más cercanos al que aquí se presenta. El enfoque de centrarse en una porción del cromosoma, como en nuestro método de solapamiento, fue utilizado en primer lugar en el trabajo de Dorronsoro et al. [18], obteniendo un rendimiento superlinear en varios casos. Recientemente, este enfoque también ha sido probado utilizando un algoritmo de optimización de enjambre de partículas (PSO) [19], obteniendo también mejoras significativas en la velocidad y calidad de la solución.

El enfoque descrito por Dorronsoro et al. también ha sido utilizado por Kimovski et al. [10], pero implementando un método maestro-esclavo que divide la población en varios procesadores. Como en trabajos anteriores, cada nodo ejecuta un MOEA paralelo que sólo afecta a una porción de los individuos y el proceso maestro recibe todas las subpoblaciones a combinar cada cierto número de generaciones. Se utilizaron hasta 8 procesadores y se compararon varias alternativas de combinación. La principal diferencia de nuestro trabajo con respecto a los trabajos anteriores es que nuestro enfoque no difunde todas las soluciones a todas las islas para la recombinación, sino sólo una solución a una isla aleatoria, necesitando menos tiempo de comunicación. Además, los enfoques de Dorronsoro o Kimovsky limitaron el número máximo de islas a 8, mientras que en este documento hemos utilizado hasta 128 islas.

En nuestro trabajo anterior [11] utilizamos algunas de las ideas mencionadas anteriormente para comparar dos dMOEAs diferentes. El primero dividió el cromosoma en P secciones, siendo P el número de islas. Cada isla p sólo realizó la mutación y el cruce en esa parte (p_{th}) del cromosoma (la aplicación selectiva de operadores), mientras que el *fitness* se calculó usando el individuo entero. Después de un cierto número de generaciones, los individuos fueron migrados al azar a otras islas. Las métricas de rendimiento se calcularon al final de la ejecución. El segundo método, la aplicación selectiva de operadores con islas solapadas, usaba las secciones p_{th-1} y p_{th+1} de cada cromosoma, además de la parte p_{th}. Usando la misma cantidad de tiempo, ambos métodos obtuvieron mejores resultados que un algoritmo de *baseline* que se ocupaba de todo el cromosoma en cada isla para el cruce y la mutación. Descubrimos que el rendimiento utilizando uno u otro método depende del número de secciones de los individuos y del número de islas utilizadas. Esto nos motivó a encontrar un nuevo método automático para seleccionar este número de secciones del cromosoma a utilizar, dependiendo del número de islas. Además, se realizaron experimentos previos en un modelo de isla síncrona con un único procesador y con un número limitado de islas (8, 32 y 128). En este trabajo se han utilizado 8, 16, 32, 64 y 128 islas, y en esta ocasión los experimentos se han realizado en un cluster paralelo. Por lo tanto, al mismo tiempo, estamos proponiendo un nuevo método para dividir el espacio de búsqueda individual según el número de subpoblaciones que evolucionan, y también validando el enfoque anterior.

III. Metodología

El objetivo de esta sección es explicar la metodología que hemos seguido para comparar las diferentes versiones de la selección de secciones a modificar en cada isla.

[1] Nota: una versión extendida de este artículo está siendo revisada en la revista *Applied Soft-Computing*

En este trabajo analizamos varios algoritmos basados en ASO que hemos implementado con respecto un algoritmo *baseline* multiobjetivo paralelo. Los métodos propuestos, utilizando diferentes esquemas de solapamiento, se basan en NSGA-II, como casi todos los trabajos discutidos anteriormente [13]–[18]. Por lo tanto, hemos utilizado un algoritmo NSGA-II básico distribuido sin solapar secciones como *baseline* (B).

Este algoritmo básico distribuye la población entre P islas; después de un número fijo de generaciones, un individuo en una isla dada es migrado a otra isla al azar, evitando así sincronizar el FP global cada cierto número de generaciones como lo hacen otros métodos descritos en la Sección 2. Al final de cada ejecución, los FPs de todas las islas se agregan en uno nuevo y se evalúan las medidas de calidad.

Se pueden idear diferentes alternativas de ASO para evolucionar las subpoblaciones de acuerdo con el espacio de decisión a explorar por cada isla. Al igual que en la *baseline* (B), un individuo es migrado a otra isla al azar después de un número fijo de generaciones. En la nueva isla, este individuo será considerado como uno más en la isla, cruzado y mutado de la misma manera, dependiendo del identificador de la nueva isla (de 1 a P). Nótese que, a diferencia de otros trabajos como los descritos por Talbi et al. [12], todas las islas tratan con cromosomas completos para el cálculo del fitness, por lo que nuestro enfoque puede tratar con problemas separables y no separables.

Concretamente hemos comparado las siguientes versiones:

- **ASO con islas disjuntas (D)** En este enfoque, cada individuo del tamaño L se divide en trozos de P del tamaño L/P. Cada isla p sólo realiza crossover y mutación en la parte p_{th} de los individuos.
- **ASO con secciones solapadas (S)** Este enfoque es similar al anterior, pero cada isla también utiliza los trozos de $p+1$ y $p-1$ (usando el módulo) del individuo para el cruce y la migración. Por lo tanto, existe algún tipo de solapamiento de las partes cruzadas y mutadas entre las islas.
- **ASO automático con secciones solapadas (A)** Como en el método anterior, las secciones a tratar por los operadores se solapan, pero en lugar de usar una sección extra a cada lado de la sección p ($p+1$ y $p-1$), usa fragmentos c a cada lado ($p+c$ y $p-c$), siendo c un valor que depende del número de islas.

Como primera aproximación para calcular automáticamente este valor, se han utilizado los resultados mostrados en [11] como base para obtener c. En ese trabajo, el método de solapamiento (cuando $c = 1$) obtuvo mejores resultados si el número de islas era mayor de 8. Por el contrario, cuando el número de islas es pequeño, no es necesario solaparlas. Por lo tanto, hemos usado este conocimiento para proponer la fórmula $c = round(0{,}2 * P - 1)/2$ para calcular las secciones extra a solapar. Por lo tanto, cuando $P = 8$ entonces $c = 0$ (equivalente a D), cuando $P = 16$ entonces $c = 1$ (equivalente a S), y así sucesivamente. La figura 1 explica el método A, asumiendo que $c = 2$, por ejemplo.

IV. Experimentos

En esta subsección se describen los indicadores de calidad utilizados y los experimentos. Los indicadores de calidad elegidos son:

- Hipervolumen (HV): mide el área formada por todas las soluciones no dominadas encontradas con respecto a un punto de referencia. Los valores más altos implican una mejor calidad del FP.
- Distancia Generacional Invertida (IGD): calcula la distancia del conjunto de soluciones obtenidas al FP óptimo. En esta métrica, cuanto más bajo, mejor.
- Spread (S): Mide la dispersión entre soluciones, teniendo en cuenta la distancia euclídea entre soluciones consecutivas. Como en la métrica anterior, cuanto menor sea el valor, mejor, ya que implica soluciones distribuidas a lo largo de todo el FP.

Hemos elegido estas métricas no sólo porque se han utilizado ampliamente, especialmente en algunos de los documentos presentados en la Sección II, como [13], [15], [17], [18], sino también porque cubren diferentes criterios de calidad. La formulación matemática de estas métricas se puede encontrar en [18].

El tamaño del cromosoma (L) es de 2048. También se ha comparado un número diferente de islas (P): 8, 16, 32, 64 y 128. Este número máximo de islas también ha sido utilizado en un trabajo anterior en la literatura [17]. El cruce y la mutación elegidos, SBX y polinomial, también han sido utilizados previamente por otros autores en [13].

El benchmark ZDT [20] ha sido elegido debido a que es el más utilizado en esta área [13]–[15], [17]. Este benchmark incluye varias funciones, con diferentes características que son representativas de los problemas de optimización del mundo real. La formulación matemática de cada función está disponible en [20].

El criterio utilizado para terminar un experimento ha sido el tiempo de ejecución: 100 segundos por ejecución. Según Alba y Luque [21], otros criterios de parada como el número de evaluaciones necesarias para alcanzar una solución, pueden ser engañosos dependiendo del escenario estudiado. En nuestro caso hemos utilizado el tiempo en lugar del número de evaluaciones, en primer lugar porque nuestra hipótesis argumenta que el tiempo ahorrado en el cruce y la mutación se puede gastar en mejorar las subpoblaciones y se pueden lograr más operaciones y migraciones. Además, estamos utilizando un número diferente de islas (con diferentes tamaños de subpoblación) y eso podría llevar a diferentes tiempos de ejecución, por lo que sería difícil comparar diferentes tiempos y calidad de soluciones al mismo tiempo.

Hemos usado el framework ECJ [22] para ejecutar los experimentos. El código fuente usado puede descargarse de nuestro repositorio GitHub[2] bajo una licencia LGPL V3.

El modelo de isla se ha ejecutado de forma asíncrona, utilizando el modelo de intercambio distribuido de interpoblación de ECJ, en un cluster de 16 nodos, cada uno con 16

[2]https://github.com/hpmoon/hpmoon-islands

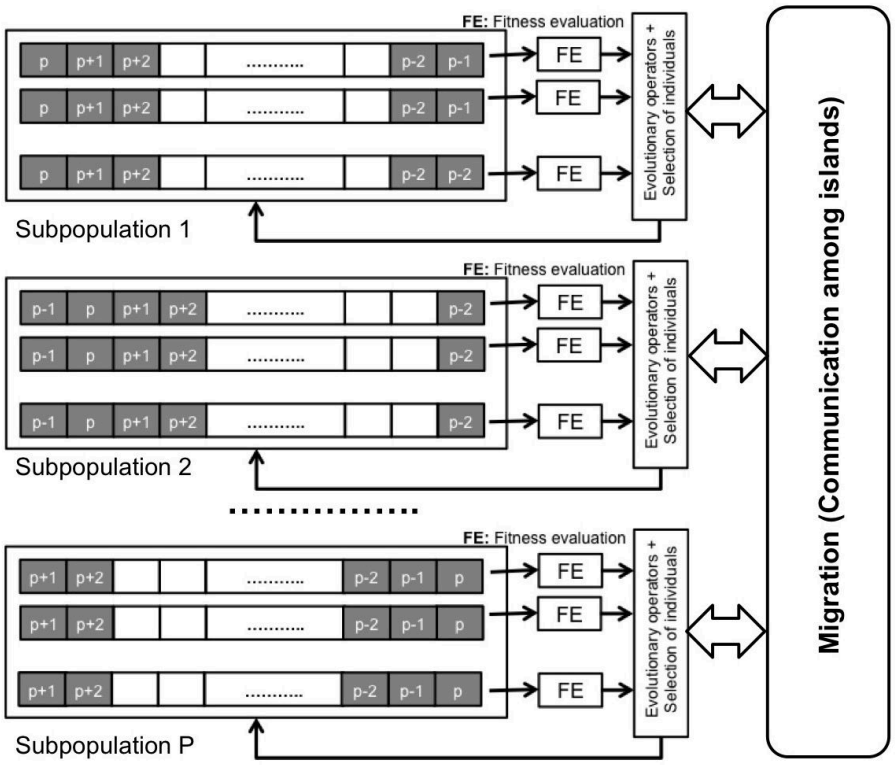

Figura 1. ASO solapado automáticamente (A): cada isla p modifica los $p + c$, p_{th} y $p - c$ componentes (en gris) de los individuos usando operadores genéticos (cruce y mutación). Además, cada isla evalúa a sus propios individuos usando el cromosoma completo. Después de un número dado de generaciones coopera con las otras islas a través de la migración. Se calcula c dependiendo del número de islas. En este caso, $c = 2$.

Nombre del parámetro	*Valor*
Tamaño global de la población (N)	1024
Selección	Torneo Binario
Tipo de reemplazo	Generacional
Tipo de crossover	SBX
Tipo de mutación	Polinomial
Probabilidad de mutación	$1/L$
Individuos por migración	1
Generaciones entre migración	5
Selección para migración	Torneo Binario
Ejecuciones por configuración	30
Número de islas (P)	8, 16, 32, 64 and 128
Tamaño del cromosoma (L)	2048
Tiempo de ejecución (s)	100

Tabla I
PARÁMETROS Y OPERADORES USADOS EN LOS EXPERIMENTOS.

procesadores Intel(R) Xeon(R) CPU E5520 @2.27GHz, 16 GB RAM, tarjetas de red Broadcom NetXtreme II BCM5716 1000Base-T (C0) PCI Express, CentOS 6.8 y Java Versión 1.8.0_80.

El conjunto total de parámetros usados se muestra en la Tabla I.

V. Resultados

Para calcular la calidad de los FPs obtenidos en cada configuración se han utilizado diferentes métricas, explicadas anteriormente. Como el HV requiere que se calcule un punto de referencia, hemos elegido el valor (1,9) ya que ninguna de las soluciones existentes en todos los frentes obtenidas durante todas las ejecuciones están dominadas por él.

Se ha realizado una prueba de significación de Kruskal-Wallis a las métricas de todas las ejecuciones de las configuraciones, ya que la prueba de Kolmogorov-Smirnov detectó distribuciones no normales. Los resultados medios de cada configuración se muestran en la Tabla II. Como se explicó anteriormente, cuando $P = 8$ los resultados de A son equivalentes a los obtenidos por D (porque $c = 0$), y cuando $P = 16$ son equivalentes a S ($c = 1$). Nos referimos a esta equivalencia con la palabra *Equiv-* en las tablas.

Los resultados muestran que la división del cromosoma produce una mejora en todos los indicadores de calidad utilizando la versión automática (A) (Tabla II), superando incluso los métodos solapado (S) y disjunto (D). Por lo tanto, existe algún tipo de punto límite en la longitud de los cromosomas donde un método será preferible a otro, además de depender del número de islas y del tamaño de la población.

Esto puede explicarse comparando el número de soluciones no dominadas de cada frente y el número medio de generaciones (Tabla III). Aumentar el número de islas implica más generaciones con todos los métodos (lógicamente, ya que hay menos individuos en cada isla). Pero también, los métodos ASO se acercan al número de generaciones con la *baseline* al aumentar el número de islas. Sin embargo el tiempo de

	HV				Spread				IGD			
#Island	B	D	S	A	B	D	S	A	B	D	S	A
ZDT1												
8	0.891	**0.953**	0.937	Equiv-D	**0.681**	**0.635** B	0.661 D	Equiv-D	0.015	**0.002**	0.005	Equiv-D
16	0.884	0.850	**0.942**	Equiv-S	**0.705**	0.908	**0.670** B	Equiv-S	0.016	0.022	**0.004**	Equiv-S
32	0.851	0.674	0.859 B	**0.900**	**0.754**	0.868	0.826 D	**0.763** B	0.023	0.062	0.020 B	**0.012**
64	**0.800**	0.608	0.697	**0.824** B	**0.808**	0.880	**0.861** B	**0.823** B	0.033	0.078	0.056	**0.027**
128	0.735	0.582	0.613	**0.745**	**0.841**	0.888	0.878 D	0.865 S	0.047	0.084	0.075	**0.043**
ZDT2												
8	0.832	**0.895**	0.869	Equiv-D	**0.849**	**0.886** B	0.853 D	Equiv-D	0.023	**0.006**	0.013	Equiv-D
16	0.831	0.833 B	**0.884**	Equiv-S	**0.810**	1.001	**0.802** B	Equiv-S	0.023	0.022 B	**0.009**	Equiv-S
32	0.800	0.628	0.800 B	**0.817**	**0.848**	0.974	0.983 D	0.908	0.031	0.082	0.032 B	**0.027**
64	**0.729**	0.491	0.623	0.716	**0.909**	0.967	0.979 D	**0.997** BD	**0.052**	0.121	0.084	**0.055** B
128	**0.630**	0.441	0.500	**0.614** B	**0.957**	0.989	0.978 D	**0.994** BS	**0.080**	0.136	0.119	**0.085** B
ZDT3												
8	0.917	**0.971**	0.960	Equiv-D	**0.843**	**0.854** B	0.868 D	Equiv-D	0.009	**0.001**	0.004	Equiv-D
16	0.911	0.876 B	**0.963**	Equiv-S	**0.864**	0.899 B	**0.837** B	Equiv-S	0.010	0.014	**0.003**	Equiv-S
32	0.884	0.710	0.883 B	**0.931**	**0.856**	**0.870** B	**0.842** B	**0.842** BDS	0.013	0.032	0.013 B	**0.008**
64	0.828	0.645	0.728	**0.854**	**0.878**	**0.896** B	**0.871** BD	**0.873** BDS	**0.019**	0.040	0.030	**0.016** B
128	**0.770**	0.620	0.651	**0.773** B	**0.887**	**0.901** B	**0.890** BD	**0.885** BDS	**0.026**	0.043	0.039	**0.025** B
ZDT6												
8	0.271	**0.398**	0.323	Equiv-D	**0.982**	**0.982** B	0.994	Equiv-D	0.171	**0.115**	0.149	Equiv-D
16	0.275	0.295 B	**0.354**	Equiv-S	**0.981**	**0.970** B	1.006	Equiv-S	0.170	0.161 B	**0.136**	Equiv-S
32	0.239	0.123	0.240	**0.254**	**0.989**	**0.991** B	**0.982** BD	**0.999** BD	0.186	0.235	0.185 B	**0.179**
64	**0.184**	0.068	0.125	**0.178** B	**0.985**	**0.982** B	**0.992** B	**0.995** BS	**0.209**	0.259	0.235	0.212
128	**0.128**	0.051	0.071	**0.124** B	**0.991**	**0.992** B	**0.988** BD	1.003	**0.233**	0.266	0.257	**0.235** B

Tabla II

MÉTRICAS DE CALIDAD MEDIA OBTENIDAS DESPUÉS DE 30 EJECUCIONES POR CONFIGURACIÓN, PARA LOS 4 MÉTODOS COMPARADOS: *baseline* (B), DISJUNTO (D), SOLAPADO (S) Y SOLAPADO AUTOMÁTICO (A). LOS ACRÓNIMOS QUE APARECEN JUNTO A LOS VALORES INDICAN QUE NO HAY DIFERENCIAS SIGNIFICATIVAS CON RESPECTO A ESE MÉTODO PARA ESE VALOR. LOS MEJORES VALORES ESTÁN MARCADOS EN NEGRITA. EQUIV-X IMPLICA QUE EL VALOR ES EL MISMO QUE EL DE EJECUTAR X, YA QUE AMBAS CONFIGURACIONES SERÍAN IGUALES.

	Average solutions per front				Generations			
#Island	B	D	S	A	B	D	S	A
ZDT1								
8	137.733	47.167	119.933 B	Equiv-D	175.067	225.667	207.933	Equiv-D
16	92.633	38.233	47.533 D	Equiv-S	204.867	238.533	232.900	Equiv-S
32	56.167	41.167	28.667	32.567 S	225.433	243.833	242.533	242.000 S
64	50.900	49.667 B	35.600	25.367	236.500	245.967	245.700	244.033 D
128	45.767	64.933	44.167 B	31.867	242.800	247.000	247.000 D	245.733
ZDT2								
8	64.267	22.733	56.867 B	Equiv-D	175.633	226.000	208.100	Equiv-D
16	52.300	10.733	20.067 D	Equiv-S	205.033	238.867	233.100	Equiv-S
32	34.400	10.367	10.433 D	17.900 S	225.600	244.267	242.633	242.000 S
64	24.600	10.900	9.267 D	12.867 DS	236.700	246.000	245.767	244.033 D
128	18.667	17.833 B	10.367	12.933 S	243.367	247.000	247.000 d	245.833
ZDT3								
8	163.767	83.367	119.467	Equiv-D	176.533	226.400	207.700	Equiv-D
16	108.467	49.433	47.700 D	Equiv-S	205.333	238.467	232.933	Equiv-S
32	72.133	44.333	31.833	36.533 S	225.100	243.900	242.433	242.000 S
64	54.867	57.600 B	40.633	29.300	236.333	245.933	245.733	244.000 D
128	50.600	72.133	49.033 B	34.933	243.200	247.000	247.000 D	245.833
ZDT6								
8	22.133	13.833	14.433 D	Equiv-D	175.733	226.100	207.767	Equiv-D
16	20.767	15.867	13.300 D	Equiv-S	205.033	238.433	232.933	Equiv-S
32	22.600	10.700	10.033 D	11.967 DS	225.833	243.667	242.500	242.000 S
64	19.033	11.267	10.533 D	10.567 DS	236.433	245.900	245.833	244.000 D
128	15.800	24.500	11.600	12.233 DS	243.800	247.000	247.000 D	245.533

Tabla III

PROMEDIO DE GENERACIONES Y PROMEDIO DE SOLUCIONES POR FRENTE, OBTENIDOS DESPUÉS DE 30 EJECUCIONES POR CONFIGURACIÓN, PARA LOS 4 MÉTODOS COMPARADOS: *baseline* (B), DISJUNTO (D), SOLAPADO (S) Y SOLAPADO AUTOMÁTICO (A). LOS ACRÓNIMOS QUE APARECEN JUNTO A LOS VALORES INDICAN QUE NO HAY DIFERENCIAS SIGNIFICATIVAS CON RESPECTO A ESE MÉTODO PARA ESE VALOR. EQUIV-X IMPLICA QUE EL VALOR ES EL MISMO QUE EL DE EJECUTAR X, YA QUE AMBAS CONFIGURACIONES SERÍAN IGUALES.

migración es lo suficientemente grande como para no mejorar el número de generaciones con respecto a la *baseline*, incluso en el menor número de islas. Por lo tanto, más generaciones no significan necesariamente mejorar la solución del FP global, sino centrarse en diferentes elementos del cromosoma. Como se ha dicho anteriormente, cada isla desconoce los FPs de las otras islas, y están tratando de optimizar sus soluciones independientemente. Con respecto al número promedio de soluciones por frente, hay una clara diferencia con la *baseline*, donde este valor es en la mayoría de los casos, menos de la mitad. El número de soluciones no dominadas también implica un mejor indicador de Spread, donde la *baseline* obtiene mejores (o no significativamente diferentes) valores en la mayoría de las configuraciones comparadas.

VI. Conclusiones

Los problemas que requieren un alto rendimiento y que tratan con un gran número de variables de decisión pueden aprovecharse de la división del espacio de decisión que proporcionan los algoritmos paralelos y distribuidos. Esto se puede hacer en dMOEAs mediante la aplicación selectiva de operadores (ASO), es decir, dividiendo el cromosoma en diferentes partes, cada una modificada por una isla diferente. Este trabajo compara un NSGA-II distribuido, como *baseline*, con tres estrategias diferentes para separar el cromosoma (partes disjuntas o solapadas), utilizando distintos números de islas. Los resultados muestran que estos métodos pueden lograr

métricas de mejor calidad que el *baseline* en la misma cantidad de tiempo.

Los resultados obtenidos también muestran que al aumentar el número de islas, el método de solapamiento automático (A) mejora significativamente los resultados con respecto a los métodos disjunto y solapado. El estudio de este factor con más tipos de problemas y nuevas configuraciones del tamaño de la población y la longitud de los cromosomas puede abordarse en el futuro. Por ejemplo, comparar diferentes maneras de calcular c usando funciones lineales, logarítmicas o exponenciales dependiendo del tamaño de la población, número de islas, tamaño del cromosoma, u otros valores. Además, se podría realizar un análisis de la diversidad de los individuos de cada isla durante la ejecución del algoritmo para comprender su influencia en los resultados.

Asimismo, podrían utilizarse implementaciones más distribuidas en varios sistemas (como GPUs o clusters heterogéneos) con diferentes cantidades de islas/procesadores para realizar un estudio de escalabilidad de los diferentes métodos, siendo el tiempo de transmisión entre islas un tema relevante a tratar. También pueden compararse otros MOEAs disponibles en la literatura, como SPEA o MOEA/D. Además, podrían utilizarse otros benchmarks y problemas reales para validar este enfoque.

Referencias

[1] A. M. Mora, P. García-Sánchez, J. J. Merelo-Guervós, and P. A. Castillo, "Pareto-based multi-colony multi-objective ant colony optimization algorithms: an island model proposal," *Soft Comput.*, vol. 17, no. 7, pp. 1175–1207, 2013.

[2] F. Luna and E. Alba, "Parallel multiobjective evolutionary algorithms," in *Springer Handbook of Computational Intelligence*, J. Kacprzyk and W. Pedrycz, Eds. Springer, 2015, pp. 1017–1031.

[3] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. C. Coello, "A survey of multiobjective evolutionary algorithms for data mining: Part I," *IEEE T. Evolut. Comput.*, vol. 18, no. 1, pp. 4–19, 2014.

[4] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Springer, 2015.

[5] E.-G. Talbi, "A unified view of parallel multi-objective evolutionary algorithms," *J. Parallel Distrib. Comput.*, pp.–, 2018, in press. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S074373151830279X

[6] E. Alba, G. Luque, and S. Nesmachnow, "Parallel metaheuristics: recent advances and new trends," *Int. T. Oper. Res.*, vol. 20, no. 1, pp. 1–48, 2013.

[7] Y. Gong, W. Chen, Z. Zhan, J. Zhang, Y. Li, Q. Zhang, and J. Li, "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art," *Appl. Soft Comput.*, vol. 34, pp. 286–300, 2015.

[8] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 5, pp. 443–462, 2002.

[9] J. Branke, H. Schmeck, K. Deb, and R. S. Maheshwar, "Parallelizing multi-objective evolutionary algorithms: cone separation," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2004, 19-23 June 2004, Portland, OR, USA*. IEEE, 2004, pp. 1952–1957.

[10] D. Kimovski, J. Ortega, A. Ortiz, and R. Baños, "Parallel alternatives for evolutionary multi-objective optimization in unsupervised feature selection," *Expert Syst. Appl.*, vol. 42, no. 9, pp. 4239–4252, 2015.

[11] P. García-Sánchez, J. Ortega, J. González, P. A. Castillo, and J. J. Merelo, "Addressing high dimensional multi-objective optimization problems by coevolutionary islands with overlapping search spaces," in *Applications of Evolutionary Computation - 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 - April 1, 2016, Proceedings, Part II*, ser. Lecture Notes in Computer Science, G. Squillero and P. Burelli, Eds., vol. 9598. Springer, 2016, pp. 107–117.

[12] E. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph, and C. A. C. Coello, "Parallel approaches for multiobjective optimization," in *Multiobjective Optimization, Interactive and Evolutionary Approaches [outcome of Dagstuhl seminars]*., ser. Lecture Notes in Computer Science, J. Branke, K. Deb, K. Miettinen, and R. Slowinski, Eds., vol. 5252. Springer, 2008, pp. 349–372.

[13] A. J. Nebro and J. J. Durillo, "A study of the parallelization of the multi-objective metaheuristic MOEA/D," in *Learning and Intelligent Optimization, 4th International Conference, LION 4, Venice, Italy, January 18-22, 2010. Selected Papers*, ser. Lecture Notes in Computer Science, C. Blum and R. Battiti, Eds., vol. 6073. Springer, 2010, pp. 303–317.

[14] K. Deb, P. Zope, and A. Jain, "Distributed computing of Pareto-optimal solutions with evolutionary algorithms," in *Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003, Faro, Portugal, April 8-11, 2003, Proceedings*, ser. Lecture Notes in Computer Science, C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, Eds., vol. 2632. Springer, 2003, pp. 534–549.

[15] W. Zhi-xin and G. Ju, "A parallel genetic algorithm in multi-objective optimization," in *Control and Decision Conference, 2009. CCDC '09. Chinese*, June 2009, pp. 3497–3501.

[16] N. Xiao and M. P. Armstrong, "A specialized island model and its application in multiobjective optimization," in *Genetic and Evolutionary Computation - GECCO 2003, Genetic and Evolutionary Computation Conference, Chicago, IL, USA, July 12-16, 2003. Proceedings, Part II*, ser. Lecture Notes in Computer Science, E. Cantú-Paz, J. A. Foster, K. Deb, L. Davis, R. Roy, U. O'Reilly, H. Beyer, R. K. Standish, G. Kendall, S. W. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. A. Dowsland, N. Jonoska, and J. F. Miller, Eds., vol. 2724. Springer, 2003, pp. 1530–1540.

[17] M. Märtens and D. Izzo, "The asynchronous island model and NSGA-II: study of a new migration operator and its performance," in *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013*, C. Blum and E. Alba, Eds. ACM, 2013, pp. 1173–1180.

[18] B. Dorronsoro, G. Danoy, A. J. Nebro, and P. Bouvry, "Achieving super-linear performance in parallel multi-objective evolutionary algorithms by means of cooperative coevolution," *Computers & OR*, vol. 40, no. 6, pp. 1552–1563, 2013.

[19] A. Atashpendar, B. Dorronsoro, G. Danoy, and P. Bouvry, "A scalable parallel cooperative coevolutionary PSO algorithm for multi-objective optimization," *J. Parallel Distrib. Comput.*, vol. 112, pp. 111–125, 2018.

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

[21] E. Alba and G. Luque, "Evaluation of parallel metaheuristics," in *Empirical Methods for the Analysis of Algorithms, Workshop EMAA 2006, Proceedings*, L. Paquete, M. Chiarandini, and D. Basso, Eds., Reykjavik, Iceland, 2006, pp. 9–14.

[22] S. Luke *et al.*, "ECJ: A Java-based Evolutionary Computation and Genetic Programming Research System," 2009, available at http://www.cs.umd.edu/projects/plus/ec/ecj.