Optimización Multi-Objetivo Paralela con Islas Coevolutivas y Solapamiento de Espacios de Búsqueda

Pablo García-Sánchez, Julio Ortega, Jesús González,
Pedro A. Castillo, and Juan J. Merelo[1]

Dept. de Arquitectura y Tecnología de Computadores, Universidad de Granada

Resumen Los problemas de optimización multi-objetivo a gran escala con muchas variables de decisión han atraído recientemente la atención de los investigadores, debido a que muchas aplicaciones de minería de datos que incluyen patrones de alta dimensionalidad pueden aprovechar su uso. Las arquitecturas paralelas y distribuidas actuales pueden proporcionar las capacidades de computación necesarias para hacer frente a estos problemas una vez que los procedimientos eficientes estén disponibles. En este artículo se propone un método para el modelo de isla coevolutiva cooperativa basado en la ejecución paralela de las sub-poblaciones, cuyos individuos exploran diferentes dominios del espacio de variables de decisión. Específicamente, los individuos que pertenecen a la misma subpoblación (isla) exploran el mismo subconjunto de variables de decisión. Se han considerado y comparado tres alternativas para la distribución de las variables de decisión entre las diferentes sub-poblaciones. En el primer enfoque, los individuos en diferentes sub-poblaciones exploran subconjuntos disjuntos de variables de decisión (es decir, los cromosomas se dividen en subconjuntos disjuntos). Por otra parte, en la segunda y tercera alternativa, hay cierta superposición entre las variables exploradas por los individuos en diferentes sub-poblaciones. El análisis de los resultados experimentales obtenidos mediante el uso de diferentes métricas muestra que los enfoques coevolutivos proporcionan mejoras estadísticamente significativas con respecto al algoritmo base, siendo la relación del número de islas (sub-poblaciones) y la longitud del cromosoma (número de variables de decisión) un factor relevante para determinar la alternativa más eficiente de distribuir las variables de decisión.

Keywords: Algoritmos multi-objectivo, NSGA-II, Modelo de islas, algoritmos evolutivos distribuidos

1. Introducción

Los Algoritmos Evolutivos (AE) son inherentemente paralelizables, ya que cada individuo puede ser considerado como una unidad independiente [1], y por lo tanto, el rendimiento computacional se puede mejorar con respecto a sus versiones no paralelas. Esto también se puede aplicar a los algoritmos evolutivos multi-objetivo (Multi-objective Evolutionary Algorithms, MOEAs). Además, como este tipo de algoritmos puede ser computacionalmente costoso, se han propuesto

varios métodos de paralelización [2]. Sin embargo, debido a que los MOEAs trabajan sobre conjuntos completos de soluciones, llamados frentes de Pareto (FPs), se deben abordar diferentes mecanismos de distribución y de intercambio, ya que no existe una relación entre el aumento de velocidad de paralelización alcanzable y la necesidad de recombinar resultados globalmente, para identificar con precisión el FP [3]. Los enfoques clásicos, tales como los AEs paralelos globales (Master-Slave), o los algoritmos espacialmente estructurados (modelo AE isla o celular) han sido aplicados con éxito en el pasado [4,5].

Los MOEAs han ganado la atención en los últimos años, sobre todo debido a su aplicación en problemas reales [2,6]. Por lo general, estos problemas requieren un mayor número de variables de decisión, y estos individuos más grandes requieren un tiempo extra significativo para cruce, mutación y migración. Por lo tanto, dividiendo el espacio de decisión (es decir, el cromosoma) se puede mejorar el rendimiento y calidad de la solución. En este aspecto, el modelo de coevolución es un modelo de dimensión distribuida en el que un problema de alta dimensionalidad se divide en problemas de dimensiones inferiores [7], que evolucionan por separado. Además, la aplicación de técnicas de paralelismo en los AE no sólo implica una reducción en el tiempo de ejecución, sino en el desarrollo de nuevos y más eficientes modelos de búsqueda [2]. La idea de dividir el espacio de decisión ha sido estudiado en [8], donde los procesos esclavos evolucionan diferentes sub-poblaciones, creadas y recombinadas por un proceso maestro, que realiza diferentes alternativas de recombinación de los fragmentos devueltos por los procesos de trabajo. En [9], se utilizó un modelo de islas coevolutivo distribuido. En ambos trabajos se han alcanzado importantes aceleraciones, aunque se utilizó un bajo número de nodos (8). Sin embargo, al aumentar el número de islas, la división de cada sección del cromosoma se hace más pequeña, y se puede afectar la escalabilidad por la obtención de soluciones de menor calidad en la misma cantidad de tiempo.

La hipótesis de este trabajo consiste en que el uso de secciones solapadas de los cromosomas en un algoritmo multi-objetivo coevolutivo basado en islas puede mejorar la calidad de las soluciones en el mismo tiempo de cálculo cuando el número de islas aumenta. Para demostrar esto, se compara un nuevo esquema de superposición de las islas con una versión base de un algoritmo NSGA-II [10] distribuido, con un enfoque coevolutivo basado en conjuntos disjuntos, con diferentes problemas y con diferentes números de islas. Los resultados muestran que esta nueva técnica puede mejorar diferentes indicadores de calidad en la misma cantidad de tiempo.

El resto del trabajo se estructura de la siguiente manera: después de un estudio de la literatura sobre la paralelización en MOEAs, se describen los algoritmos a comparar y la metodología utilizada (Sección 3). A continuación, se presentan los resultados de los experimentos (Sección 4), seguidos de las conclusiones y sugerencias para futuras líneas de trabajo.

2. Estado del arte

Los algoritmos evolutivos paralelos y distribuidos se han utilizado desde la década de los 2000 [11] para aprovechar las capacidades de las nuevas arquitecturas de computadores paralelos, tales como los clusters o GRIDs. Sin embargo, la distribución de MOEAs no es tan fácil como lo es en los EA de un solo objetivo, ya que tienen que hacer frente a toda una serie de soluciones dependientes, el Frente de Pareto, que debe ser gestionado en varios pasos del algoritmo. Algunos autores se han centrado en aproximaciones Maestro-Esclavo para paralelizar este tipo de EA. Por ejemplo, Durillo et al. [12] comparó diferentes métodos maestro-esclavo para ahorrar tiempo cuando se ejecuta el AE. Por otro lado, método propuesto por Hiroyasu [13], genera descendencia en función de la potencia de cálculo.

Los primeros enfoques para MOEAs distribuidos (dMOEAs) fueron estudiados en [14]. En ese trabajo, la dominancia de las soluciones se divide en las islas utilizando una transformación de coordenadas. Los autores concluyeron que dividir el espacio de búsqueda es una buena idea, aunque lograrlo no es trivial. Otras ideas para dividir el espacio de búsqueda incluyen la separación de objetivos en diferentes procesadores [15], o dividir en dos poblaciones: una para la élite y otra para búsqueda [16]. Martens, en [17] genera una red Barabasi-Albert como la topología de la isla, y usa un método de selección basado en la migración de individuos de una zona no-crowded y aceptación basada en la diversidad.

Enfoques más similares al que se presenta en este artículo se estudiaron en [9,8], utilizando también la coevolución cooperativa para problemas de alta dimensionalidad. Dorronsoro et al. en [9] obtuvo un rendimiento super-lineal en varios casos, centrándose cada isla en una porción del cromosoma. Sin embargo, en algunos casos la versión paralela no siempre mejoró la versión secuencial. Kimovski et al. [8] propuso un método maestro-esclavo que divide la población en varios procesadores, cada uno ejecutando un MOEA paralelo que sólo afecta a una parte de los individuos. Después de un cierto número de generaciones, el proceso maestro recibe todas las sub-poblaciones, que se combinan. Se compararon diferentes alternativas de combinación en este paso y se usaron hasta 8 procesadores en la configuración experimental. Nuestro enfoque en este artículo no transmite todas las soluciones a todas las islas para la recombinación, como en los trabajos anteriores, sino sólo una solución a una isla al azar, necesitando así menos tiempo de comunicación. Además, el número máximo de islas en los enfoques de Dorronsoro o Kimovsky fue de 8, mientras que en este trabajo se han utilizado hasta 128 islas.

Recientemente, algunos investigadores [18] se han centrado en trabajar sobre el espacio de soluciones, dividiendo el frente de Pareto en diferentes grupos con el fin de modelarlo. Este enfoque *divide y vencerás* es fácilmente paralelizable y, en el sentido de dar la responsabilidad de diferentes partes del espacio de búsqueda usando diferentes módulos, es similar a la presentada en este trabajo. Una versión preliminar usando un solo procesador y un número más limitado de islas y métodos fue presentada por los autores en [19]. A continuación describiremos

cómo hemos probado nuestro nuevo enfoque en la siguiente sección, usando un clúster real, y un mayor número de islas.

3. Metodología y configuración experimental

En esta sección se describen los indicadores de calidad utilizados y los métodos comparados. Los indicadores de calidad elegidos han sido:

- Hipervolumen (HV): mide el área formada por todas las soluciones no dominadas encontradas con respecto a un punto de referencia. Valores altos implican una mejor calidad del FP.
- Distancia generacional invertida (DGI): calcula la distancia del conjunto obtenido de soluciones al FP óptimo. Sin embargo, esta métrica requiere conocer el FP óptimo encontrado en la literatura, o al teórico. En este indicador, valores más bajos son mejores.
- Dispersión (Spread, S): Mide la dispersión entre las soluciones, teniendo en cuenta la distancia euclídea entre las soluciones consecutivas. Al igual que en métrica anterior, valores más bajos son mejores, ya que implica soluciones distribuidas a lo largo de todo el FP.

Estas métricas han sido usadas ampliamente, y descritas con más profundidad en algunos de los artículos descritos anteriormente en la sección 2.

3.1. Algoritmo base (B)

Este algoritmo es un algoritmo normal NSGA-II distribuido a lo largo de un número P de islas. Después de un número fijo de generaciones, un individuo se migra a otra isla al azar. Al final de la ejecución, todos los FP de todas las islas se agrupan en una nueva para calcular las medidas de calidad.

3.2. Algoritmos coevolutivos comparados

Se han utilizado tres métodos coevolutivos diferentes. En estos métodos, cada isla sólo realiza cruce y mutación en secciones específicas de los individuos.

Al igual que en el algoritmo base, cada cierto número de generaciones, un individuo se migra a otra isla al azar. En la nueva isla, este individuo será considerado al igual que los demás, cruzado y mutado de la misma manera, en función del identificador de isla. Hay que tener en cuenta que, al contrario de otros trabajos como los que se describen en [11] todas las islas tratan con los cromosomas completos para el cálculo del fitness, por lo que nuestro enfoque puede hacer frente con problemas no descomponibles.

Coevolución con islas disjuntos (D) En este enfoque, cada individuo de tamaño L se divide en P trozos de tamaño de L/P. Cada isla p sólo realiza cruce y mutación en la p_{th} parte de los individuos. La Figura 1 (a) describe este enfoque.

Optimización MOP con Islas Coevolutivas y Solapamiento de Espacios

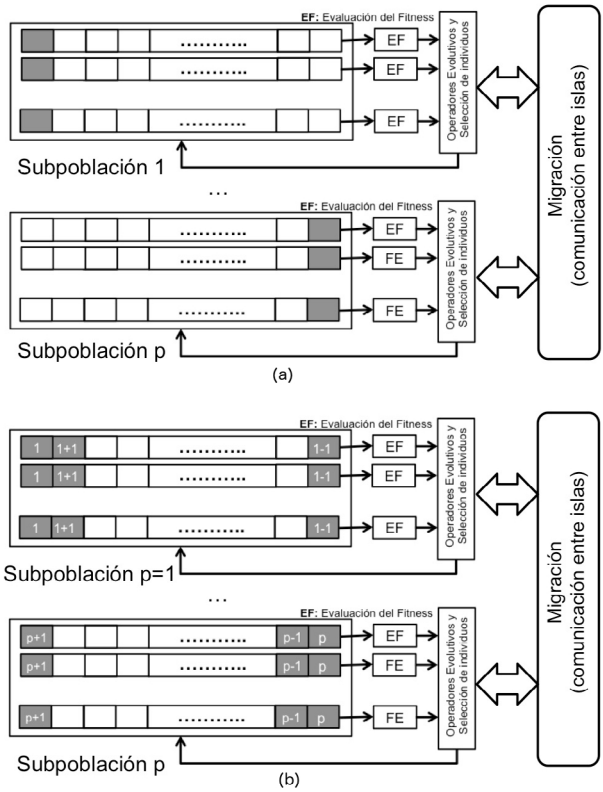

Figura 1: En el algoritmo disjunto (a) cada isla p solo modifica el fragmento p_{th} (en gris) de los individuos, mientras que en el algoritmo solapado adaptativo (b) modifica el fragmento p y los c a su izquierda y derecha. En este caso $c = 1$, por lo que es equivalente al algoritmo solapado (S).

Coevolución con islas solapadas (S) Este enfoque es similar al anterior, pero cada isla también utiliza las secciones $p+1$ y $p-1$ (módulo el tamaño) del individuo para el cruce y la mutación. Por lo tanto, existe cierto tipo de solapamiento de las secciones cruzadas y mutadas entre las islas.

Coevolución con islas solapadas con tamaño adaptativo (A) Al igual que el anterior, las secciones a abordar están solapadas, pero en lugar de usar una sección extra a cada lado ($p+1$ y $p-1$), utiliza c fragmentos más a cada lado ($p+c$ y $p-c$), siendo el valor máximo de c un valor dependiente del tamaño del cromosoma. La figura 1 (b) muestra las partes afectadas de los individuos en cada isla.

4. Experimentos y resultados

Los cuatro métodos han sido probados con una longitud de cromosomas (L) de 2048. También se han comparado con distintos números de islas (P): 8, 16, 32, 64 y 128. Este número máximo de islas también se ha utilizado en trabajos anteriores en la literatura [17]. El cruce y mutación elegidos, SBX y polinomial, también se han utilizado anteriormente por otros autores en [12].

Se ha elegido ZDT [20] como benchmark, ya que es el más utilizado en este área [14,17,16,12]. La distribución óptima de los FPs utilizados para la comparativa de las métricas de calidad tiene 1000 soluciones [1].

El criterio de parada utilizado es el tiempo de ejecución, con límite en 100 segundos. Hemos utilizado el tiempo en lugar del número de evaluaciones en primer lugar porque nuestra hipótesis sostiene que el tiempo ahorrado en el cruce y la mutación se puede emplear en la mejora de las sub-poblaciones y se pueden lograr más operaciones y migraciones. Además, estamos utilizando diferentes números de islas (con diferentes tamaños de sub-población) que podrían dar lugar a diferentes tiempos de ejecución, por lo que sería difícil comparar diferentes tiempos y calidad de soluciones al mismo tiempo.

Para el cálculo de c en el método A se ha utilizado como base los resultados de los métodos D y S, que como se verá a continuación, reducían su calidad cuando aumenta el número de islas, por lo que se ha usado la fórmula $c = round(0{,}2 * P - 1)/2$ para tener una proporción de secciones a modificar equilibrada.

Se ha utilizado el framework ECJ [21] para realizar los experimentos. Se han desarrollado operadores específicos como nuevos módulos para ECJ, disponibles en nuestro repositorio GitHub bajo una licencia LGPL v3 [2]. El modelo de la isla se ha ejecutado de forma asíncrona, utilizando el sistema de intercambio de ECJ, en un clúster de 16 nodos, cada uno con con 16 Intel(R) Xeon(R) CPU E5520 @2.27GHz, Java Versión 1.7.0_51.

Se han utilizado diversas métricas, explicadas en la sección anterior, para calcular la calidad de los FPs obtenidos en cada configuración. Como algunas

[1] Los FP óptimos están disponibles en: `http://www.tik.ee.ethz.ch/sop/download/supplementary/testproblems/`

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

Optimización MOP con Islas Coevolutivas y Solapamiento de Espacios

Nombre del parámetro	*Valor*
Número de islas (P)	8, 16, 32, 64 y 128
Fragmentos extra a cada lado en A (c)	0, 1, 3, 6 y 12
Tamaño del cromosoma (L)	2048
Tiempo de ejecución (s)	100
Tamaño global de la población (N)	1024
Tipo de selección	Selección por torneo binario
Tipo de remplazo	Generacional
Tipo de cruce	SBX
Tipo de mutación	Polinomial
Probabilidad de mutación	$1/L$
Generaciones entre migraciones	5
Individuos por migración	1
Selección para migración	Torneo binario
Ejecuciones por configuración	30

Tabla 1: Parámetros y operadores usados en los experimentos.

de las métricas requieren un punto de referencia para ser calculadas (como el HV), se ha elegido el punto (1,9) como referencia, ya que ninguno de los FP generados en todas las ejecuciones están dominados por él. Las métricas se han normalizado con respecto a ese punto. El test de significancia de Kruskall-Wallis se ha utilizado para comparar todas las métricas de todas las ejecuciones de cada configuración, ya que el test de Kolmogorov-Smirnov detectó distribuciones no normales. Los resultados promedio para cada configuración se muestran en la tabla 2.

Los resultados muestran que existe una mejora significativa respecto al algoritmo base para las métricas con números pequeños de islas (8 y 16) usando los métodos D y S, que son equivalentes a A según la fórmula de cálculo de c. A partir de 32 islas el algoritmo A también obtiene los mejores valores de todos los métodos, aunque en algunos casos y dependiendo de la instancia del problema, no existan diferencias significativas.

Además, los resultados muestran que todos los indicadores de calidad disminuyen cuando se aumenta el número de islas, ya que las subpoblaciones son más pequeñas. Esto es consistente con la afirmación de Durillo en [9], donde se comprobó que los MOEAs coevolutivos cooperativos funcionan mejor con poblaciones grandes (más de 100 individuos).

5. Conclusiones

Los problemas de alto rendimiento que requieren un gran número de variables de decisión pueden aprovechar la división del espacio de decisión que los algoritmos paralelos y distribuidos ofrecen. Esto se puede realizar en dMOEAs dividiendo el cromosoma en diferentes partes, siendo cada una modificada por

	HV				Dispersión				DGI			
#Islas	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 2: Métricas de calidad obtenidas después de 30 ejecuciones por configuración, para los cuatro métodos comparados: algoritmo base (B), disjunto (D), solapado (S) y solapado adaptativo (A). Los acrónimos junto a los valores indican que no hay diferencia significativa con respecto a ese método para ese valor. Los mejores valores están marcados en negrita.

una isla diferente. Este trabajo compara un algoritmo distribuido NSGA-II como base respecto a 3 estrategias diferentes para separar los cromosomas (secciones disjuntas, superpuestas y superpuestas adaptativas), utilizando un gran número de islas. Los resultados muestran que estos métodos pueden lograr mejores métricas de calidad respecto al algoritmo base en la misma cantidad de tiempo. Esto puede deberse a la reducción en el tiempo de cruce y mutación en los cromosomas, lo que permite ejecutar un mayor número de generaciones y más modificaciones de las soluciones del FP en cada sub-población, y por lo tanto mejora la calidad del FP global.

Los resultados también muestran que al aumentar el número de islas, los métodos con solapamiento mejoran significativamente los resultados con respecto al método disjunto, siendo el solapamiento adaptativo el más indicado conforme el número de islas aumenta. Por lo tanto, la relación entre el número de islas utilizado, el tamaño de la población, y la longitud del cromosoma es un factor clave para decidir si se utiliza el método disjunto o los de solapamiento, por lo que el estudio de este factor con más tipos de problemas, y nuevas configuraciones de tamaño y longitudes de cromosomas y población se abordarán en el futuro.

Además, se planea probar estas implementaciones distribuidas en otros sistemas (como por ejemplo, clústers heterogéneos) con distintos números de islas/procesadores, para llevar a cabo un estudio de escalabilidad más completo de los diferentes métodos, siendo el tiempo de transmisión entre las islas un

tema relevante para abordar. También se plantea abordar otros benchmarks y problemas reales para validar el enfoque propuesto en este trabajo.

Agradecimientos

Trabajo financiado en parte por los proyectos TIN2014-56494-C4-3-P y TIN2015-67020-P (Ministerio de Economía y Competitividad), PROY-PP2015-06 (Plan Propio 2015 UGR), y MSTR (PRY142/14, Fundación Pública Andaluza Centro de Estudios Andaluces en la IX Convocatoria de Proyectos de Investigación). Los autores desean agradecer a los revisores sus comentarios, que han permitido mejorar la calidad de este artículo.

Referencias

1. Alba, E., Luque, G., Nesmachnow, S.: Parallel metaheuristics: recent advances and new trends. International Transactions in Operational Research **20**(1) (2013) 1–48
2. Luna, F., Alba, E.: Parallel multiobjective evolutionary algorithms. In Kacprzyk, J., Pedrycz, W., eds.: Springer Handbook of Computational Intelligence. Springer (2015) 1017–1031
3. Branke, J., Schmeck, H., Deb, K., Maheshwar, R.S.: 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) 1952–1957
4. Folino, G., Pizzuti, C., Spezzano, G.: A scalable cellular implementation of parallel genetic programming. IEEE Trans. Evolutionary Computation **7**(1) (2003) 37–53
5. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Trans. Evolutionary Computation **6**(5) (2002) 443–462
6. Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., Coello, C.A.C.: A survey of multiobjective evolutionary algorithms for data mining: Part I. IEEE Trans. Evolutionary Computation **18**(1) (2014) 4–19
7. Gong, Y., Chen, W., Zhan, Z., Zhang, J., Li, Y., Zhang, Q., Li, J.: Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. Appl. Soft Comput. **34** (2015) 286–300
8. Kimovski, D., Ortega, J., Ortiz, A., Baños, R.: Parallel alternatives for evolutionary multi-objective optimization in unsupervised feature selection. Expert Syst. Appl. **42**(9) (2015) 4239–4252
9. Dorronsoro, B., Danoy, G., Nebro, A.J., Bouvry, P.: Achieving super-linear performance in parallel multi-objective evolutionary algorithms by means of cooperative coevolution. Computers & OR **40**(6) (2013) 1552–1563
10. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Guervós, J.J.M., Schwefel, H., eds.: Parallel Problem Solving from Nature - PPSN VI, 6th International Conference, Paris, France, September 18-20, 2000, Proceedings. Volume 1917 of Lecture Notes in Computer Science., Springer (2000) 849–858

11. Talbi, E., Mostaghim, S., Okabe, T., Ishibuchi, H., Rudolph, G., Coello, C.A.C.: Parallel approaches for multiobjective optimization. In Branke, J., Deb, K., Miettinen, K., Slowinski, R., eds.: Multiobjective Optimization, Interactive and Evolutionary Approaches [outcome of Dagstuhl seminars]. Volume 5252 of Lecture Notes in Computer Science., Springer (2008) 349–372
12. Nebro, A.J., Durillo, J.J.: A study of the parallelization of the multi-objective metaheuristic MOEA/D. In Blum, C., Battiti, R., eds.: Learning and Intelligent Optimization, 4th International Conference, LION 4, Venice, Italy, January 18-22, 2010. Selected Papers. Volume 6073 of Lecture Notes in Computer Science., Springer (2010) 303–317
13. Hiroyasu, T., Yoshii, K., Miki, M.: Discussion of parallel model of multi-objective genetic algorithms on heterogeneous computational resources. In Lipson, H., ed.: Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings, London, England, UK, July 7-11, 2007, ACM (2007) 904
14. Deb, K., Zope, P., Jain, A.: Distributed computing of pareto-optimal solutions with evolutionary algorithms. In Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L., eds.: Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003, Faro, Portugal, April 8-11, 2003, Proceedings. Volume 2632 of Lecture Notes in Computer Science., Springer (2003) 534–549
15. Xiao, N., Armstrong, M.P.: A specialized island model and its application in multiobjective optimization. In Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U., Beyer, H., Standish, R.K., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A.C., Dowsland, K.A., Jonoska, N., Miller, J.F., eds.: Genetic and Evolutionary Computation - GECCO 2003, Genetic and Evolutionary Computation Conference, Chicago, IL, USA, July 12-16, 2003. Proceedings, Part II. Volume 2724 of Lecture Notes in Computer Science., Springer (2003) 1530–1540
16. Zhi-xin, W., Ju, G.: A parallel genetic algorithm in multi-objective optimization. In: Control and Decision Conference, 2009. CCDC '09. Chinese. (2009) 3497–3501
17. Märtens, M., Izzo, D.: The asynchronous island model and NSGA-II: study of a new migration operator and its performance. In Blum, C., Alba, E., eds.: Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013, ACM (2013) 1173–1180
18. Cheng, R., Jin, Y., Narukawa, K.: Adaptive reference vector generation for inverse model based evolutionary multiobjective optimization with degenerate and disconnected pareto fronts. In: Evolutionary Multi-Criterion Optimization. Springer International Publishing (2015) 127–140
19. García-Sánchez, P., Ortega, J., González, J., Castillo, P.A., Merelo, J.J.: Addressing high dimensional multi-objective optimization problems by coevolutionary islands with overlapping search spaces. In Squillero, G., Burelli, P., eds.: Applications of Evolutionary Computation - 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 - April 1, 2016, Proceedings, Part II. Volume 9598 of Lecture Notes in Computer Science., Springer (2016) 107–117
20. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation **8**(2) (2000) 173–195
21. Luke, S., 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`.