PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Reporte Técnico RT 09-21

MOE: un entorno de trabajo para optimización multiobjetivo con algoritmos evolutivos

Alexis Rodríguez y Sergio Nesmachnow

2009

MOE: un entorno de trabajo para optimización multiobjetivo con algoritmos evolutivos
Rodríguez, Alexis; Nesmachnow , Sergio
ISSN 0797-6410
Reporte Técnico RT 09-21
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay, 2009

MOE: un entorno de trabajo para optimización multiobjetivo con algoritmos evolutivos

Alexis Rodríguez y Sergio Nesmachnow

Centro de Cálculo, Instituto de Computación
Facultad de Ingeniería, Universidad de la República, Uruguay
{arodrig,sergion}@fing.edu.uy

Resumen Este trabajo presenta los detalles de diseño e implementación de un entorno de trabajo para el uso de algoritmos evolutivos paralelos en la resolución de problemas de optimización multiobjetivo. El documento detalla la arquitectura del sistema y la implementación de una versión paralelo-distribuida del algoritmo NSGA-II [6], así como el análisis de la calidad de resultados y la eficiencia computacional en la resolución del conjunto de problemas sintéticos ZDT [18].

1. Introducción

Los Algoritmos Evolutivos (AEs) se han popularizado como métodos robustos y efectivos para resolver problemas de optimización. Aunque tradicionalmente se han considerado problemas que optimizan una única función objetivo, en la última década se han desarrollado varios AE para optimización multiobjetivo (MOEAs), que tienen características que los diferencian de los AEs tradicionales.

Las técnicas de procesamiento paralelo se aplican a los AEs para mejorar la eficiencia computacional y la calidad de resultados [4]. Desde la perspectiva de la eficiencia, paralelizar un AE permite acelerar la resolución de problemas que requieren utilizar poblaciones numerosas o evaluar complejas funciones objetivo. Desde el punto de vista algorítmico, los modelos paralelos pueden explotar el paralelismo intrínseco del mecanismo evolutivo, trabajando simultáneamente sobre varias poblaciones independientes para resolver un problema.

Pocos trabajos han abordado el estudio del paralelismo aplicado a los MOEAs y su influencia en la eficiencia computacional y calidad de soluciones. Este trabajo propone el diseño y análisis del *framework* MOE, un entorno de desarrollo que implementa MOEAs en un ambiente de ejecución multiproceso.

El resto del documento se organiza del modo que se describe a continuación. La sección 2 introduce los problemas de optimización multiobjetivo (MOPs). La sección 3 presenta a los MOEAs y una breve reseña de propuestas de implementación de bibliotecas de MOEAs paralelos. La sección 4 presenta el *framework* MOE, describiendo el diseño y la implementación del entorno de programación y de la version de NSGA-II incluida. El análisis de eficiencia computacional y calidad de resultados obtenidos al resolver los problemas ZDT se presenta en la sección 5. Por último, la sección 6 presenta las conclusiones y posibles líneas de trabajo futuro.

2. Problemas de optimización multiobjetivo

Un MOP plantea optimizar (minimizar o maximizar) un conjunto de funciones, habitualmente en conflicto entre sí. Como existen múltiples funciones objetivo, no existe una única solución al problema, sino un conjunto de soluciones que plantean diferentes compromisos entre los valores de las funciones a optimizar. A continuación se presenta la formulación general de un MOP, que sigue una gran parte de los problemas de optimización del mundo real, aunque en la práctica se aborden mediante un enfoque monoobjetivo [5].

$$\begin{aligned} \text{Maximizar o minimizar: } & F(x) = (f_1(x), f_2(x), \ldots, f_n(x)) \\ \text{Sujeto a: } & G(x) = (g_1(x), g_2(x), \ldots, g_s(x)) = 0 \\ & H(x) = (h_1(x), h_2(x), \ldots, h_r(x)) \leq 0 \\ & x_i^L \leq x_i \leq x_i^U \ 1 \leq i \leq N \end{aligned}$$

Sea Ω la región factible del MOP, determinada por las restricciones G y H, la solución es un vector de variables de decisión $x = (x_1, x_2, \ldots, x_N) \in \Omega$, que representa un compromiso adecuado para las funciones $f_1, f_2, \ldots, f_M$. Para el caso de minimización, un punto x^* es llamado óptimo de Pareto si $\forall x \in \Omega$ se cumple que $f_i(x) = f_i(x^*) \forall i \in 1, \ldots, M$ o para al menos un valor de i $f_i(x) > f_i(x^*)$. Asociada a esta definición se introduce la relación de dominancia, un orden parcial entre soluciones del MOP; un vector $w = (w_1, w_2, \ldots, w_N)$ domina a otro $v = (v_1, v_2, \ldots, v_N)$ (se nota $w \preceq v$) si $w_i \leq v_i \ \forall i \in 1, \ldots, M \ \wedge \ \forall i \in 1, \ldots, M / w_i < v_i$. La resolución de un MOP implica hallar un conjunto de soluciones no dominadas, de acuerdo a la definición presentada anteriormente.

El conjunto de soluciones óptimas del MOP se denomina conjunto óptimo de Pareto, definido por $P^* = x \in \Omega / \neg x' \in \Omega f(x') \preceq f(x)$. La región de puntos definida por el conjunto óptimo de Pareto en el espacio de las funciones objetivo se conoce como frente de Pareto, y está formalmente definido por $FP^* = \{u = f_1(x), f_2(x), \ldots, f_M(x)\} / x \in P^*$.

3. AE para optimización multiobjetivo

Los AEs basan su funcionamiento en la simulación del proceso de evolución natural de los seres vivos [9]. Consisten en una técnica que aplica operadores estocásticos sobre un conjunto de individuos -la población- con el propósito de mejorar su *fitness*, una medida relacionada con la función objetivo del problema. Cada individuo representa una solución potencial del problema, codificada de acuerdo a un esquema de representación. La población se genera aleatoriamente, y evoluciona aplicando iterativamente operadores de reproducción: recombinaciones de individuos -*cruzamientos*- y modificaciones aleatorias -*mutaciones*-. La evolución es guiada por una estrategia de selección de los individuos más adaptados a la resolución del problema, de acuerdo a sus valores de *fitness*.

Los AEs trabajan sobre un conjunto de soluciones y se adaptan de manera sencilla para resolver MOPs. A diferencia de otros métodos, los MOEAs no requieren de múltiples ejecuciones para hallar una aproximación al frente de Pareto. Además, los MOEAs son menos sensibles a la forma y continuidad del frente de Pareto y permiten abordar problemas de grandes dimensiones.

Un MOEA debe cumplir dos propósitos simultáneamente: aproximarse al frente de Pareto y mantener la diversidad de las soluciones, para no converger a una sección del frente. El mecanismo evolutivo permite lograr el primer propósito, mientras que para preservar la diversidad los MOEAs aplican técnicas específicas como *nichos*, *sharing* o *crowding*.

El algoritmo 1 presenta el esquema de un MOEA. Se destacan dos operadores característicos de un MOEA: el *operador de diversidad* que aplica la técnica para evitar la convergencia puntual a un sector del frente de Pareto y un procedimiento para *asignar fitness* que considera los valores de las funciones objetivo del problema y (en general) el concepto de dominancia de Pareto.

Algoritmo 1 Esquema algorítmico de un MOEA

```
generacion = 0
Inicializar la población P(0)
repetir
  evaluar P(t)
  operador de diversidad P(generacion)
  asignar fitness P(generacion)
  mating pool = seleccion(P(generacion))
  hijos = recombinación(mating pool)
  mutación(hijos)
  generacion++
  P(generacion) = reemplazar(hijos)
hasta que se cumpla condición de parada
retornar mejor solucion encontrada
```

3.1. El algoritmo NSGA-II

NSGA-II (*Non-dominated Sorting Genetic Algorithm*, versión II) [6] es un MOEA que incluye tres características distintivas para resolver aspectos criticados de la implementación original de NSGA [16]: el *ordenamiento no-dominado* elitista que utiliza una población auxiliar y disminuye el ordende complejidad de los chequeos de dominancia, la *preservación de diversidad* mediante una técnica de *crowding* que no necesita especificar parámetros adicionales, y la *asignación de fitness* en base a rangos de no dominancia que considera los valores de distancia de crowding que evalúan la diversidad de las soluciones.

El algoritmo 2 presenta un esquema de NSGA-II, según Deb [6]. Se destacan los operadores utilizados para la asignación de fitness, el ordenamiento no dominado y la evaluación de la diversidad mediante la técnica de crowding.

Algoritmo 2 Esquema del algoritmo NSGA-II

$generacion = 0$
Inicializar la población $P(0)$
repetir
  evaluar $P(generacion)$
  $Frentes =$ **ordenamiento no dominado**$(Padres \cup Hijos)$
  $i = 1; NuevaPop = \emptyset$
  mientras $|NuevaPop| + |Frentes(i)| \leq \#P$ **hacer**
    calcular distancia de crowding$(Frentes(i))$
    $NuevaPop = NuevaPop \cup Frentes(i)$
    $i++$
  fin
  ordenar por distancia$(Frentes(i))$
  $NuevaPop = NuevaPop \cup Frentes(i)[1 : (sizepop - |NuevaPop|)]$
  $generacion++$
  $Poblacion(generacion) =$ **seleccion y reproduccion**$(NuevaPop)$
hasta que se cumpla condicion de parada
retornar mejor solucion encontrada

3.2. Algoritmos evolutivos paralelos (pEAs)

Las técnicas de paralelismo se aplican a los AEs para mejorar su eficiencia y calidad de resultados [4]. Usando un modelo de evolución diferente al secuencial, que divide la población en varios elementos de cómputo para explotar la búsqueda simultánea, los pEAs aceleran la resolución de problemas que motivan usar poblaciones numerosas y/o complejas funciones objetivo. Los pEAs se clasifican según la organización de la población en tres grandes familias [3]:

- *pEAs maestro-esclavo*, que distribuyen la evaluación de la función de *fitness*, y mantienen la evolución panmíctica de los modelos secuenciales.
- *pEAs de población distribuida*, que trabajan con subpoblaciones (islas) que restringen las interacciones y un operador de migración que intercambia individuos entre islas, introduciendo diversidad.
- *pEAs celulares*, que poseen una estructura espacial subyacente a la población y un modelo de propagación de características de individuos (difusión) de acuerdo a la topología de conexión de elementos de procesamiento.

3.3. Trabajos relacionados

Diversas reseñas han resumido la investigación sobre MOEAs paralelos [5,17], aunque no ha sido frecuente desarrollar bibliotecas o *frameworks*. Las primeras versiones paralelas de NSGA fueron modelos maestro-esclavo aplicados en fluidodinámica [10,11]. Deb et al. propusieron una descomposición de dominio en NSGA-II, modificando el concepto de dominancia [7]. Duarte et al. [8] presentaron un NSGA distribuido para diseño de redes utilizando simulaciones.

En trabajos previos de uno de los autores [14,15] se propuso PNSGA-II, un NSGA-II de poblaciones distribuidas asincrónicas para entornos heterogéneos,

que logró buenos resultados de calidad y eficiencia sobre un amplio conjunto de problemas de prueba y un caso realista de diseño de redes de comunicaciones confiables.

La Universidad de Málaga ha trabajado en el diseño de algoritmos de optimización en entornos distribuidos. El proyecto Mallba [2] no incorporó MOEAs, pero inició una línea de trabajo basada en métodos instanciables que incorporan paralelismo. DEME [13] estudió la resolución de MOPs con métodos integrados en un *framework* con una *API* concisa que fomenta reusar elementos comunes, pero no incluye paralelismo. Su continuación, JMetal [12], incorporó paralelismo utilizando una biblioteca que implementa el modelo maestro-esclavo.

El trabajo previo sobre PNSGA-II y las bibliotecas diseñadas en la Universidad de Málaga han sido referencias directas para el diseño del *framework* MOE.

4. El *framework* MOE

El *framework* MOE surgió como un esfuerzo por consolidar el conocimiento adquirido en investigaciones previas y para ser utilizado como plataforma para futuros proyectos en el área. MOE se diseñó para incorporar de manera amigable técnicas de paralelismo a los MOEAs, para mejorar su eficiencia computacional y el mecanismo de búsqueda. Intentando abarcar a una amplia gama de usuarios, el diseño de MOE planteó alcanzar varios objetivos:

SIMPLICIDAD: MOE minimiza el impacto de la complejidad. Sigue un diseño sencillo, y puede ser usado por un amplio espectro del público objetivo.

FLEXIBILIDAD: MOE se diseñó preparado para el cambio, siguiendo una arquitectura modular que permite agregar, retirar y/o modificar funcionalidades.

ORTOGONALIDAD: los componentes del sistema siguen la filosofía *"haz solo una cosa y hazlo realmente bien"*. Una acción sobre un objeto no propaga ni provoca efectos colaterales, ni altera al sistema de modo inesperado.

DESEMPEÑO: el diseño, las herramientas utilizadas y la implementación intentan aprovechar al máximo los recursos computacionales, para permitir que MOE sea eficientemente aplicable a problemas reales.

MOE se implementó en *C++*, usa MPI para la ejecución distribuida, y es portable a plataformas *UNIX-Linux*. MOE está compuesto por tres paquetes (véase figura 1) que pueden ser usados como bibliotecas en otras aplicaciones.

Paquete engine Las clases de *engine* (véase figura 2) permiten instanciar un MOEA, representar los individuos, los operadores y los problemas. La clase `MOEA` implementa el esqueleto de un MOEA (véase figura 3) y utiliza las otras clases del paquete para implementar los operadores evolutivos. MOE implementa el modelo paralelo de poblaciones distribuidas usando la clase `Migration` que selecciona (usando `Selection`), transporta (usando la clase `NetStream`, tomada de Mallba) e intercambia individuos entre islas. `Population` mantiene una colección de objetos `Solution`, liberando a `MOEA` de administrar la población.

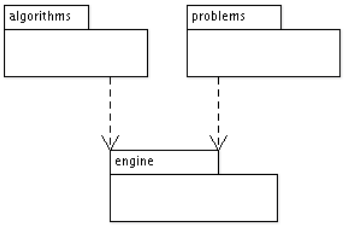

Figura 1. Paquetes de MOE.

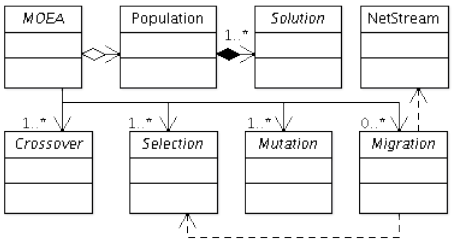

Figura 2. Clases del paquete *engine*.

En `Solution` (figura 4) reside la lógica para evaluar las funciones objetivo y las restricciones del MOP, modeladas como instancias de las clases `Objective` y `Restriction`. Las especializaciones de `Solution` deben proveer mecanismos a través de la herencia para transportar una instancia usando la clase `NetStream`.

```
virtual void run() {
initialize();
    while (!stopCondition()) {
        generation();
        generations++;
        if (migrationCondition())
            migrate();
        notifyObservers(this);
    }
    finalize();
}
```

Figura 3. MOEA: ciclo principal.

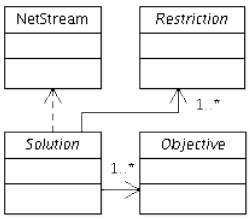

Figura 4. Vista de la clase `Solution`.

Las clases `Statistics` y `Historic` utilizan el patrón de diseño *observer* para interactuar con `MOEA` a través de notificaciones (véase figura 5), y son fácilmente activadas y desactivadas, inclusive en tiempo de ejecución. La recepción, consolidación y presentación de estadísticas se realiza por `StatisticsSink`, un objeto activo independiente que usa una instancia de MOEA para realizar la interacción panmítica usada en PNSGA-II para mejorar la calidad de resultados[14]. `Statistics` implementa estadísticas dinámicas (extensibles), que incluyen un contador de soluciones diferentes y el tamaño del frente encontrado, y estadísticas estáticas para el análisis a posteriori, abiertas para que el usuario las construya.

Paquetes algorithms y problems *Algorithms* contiene las clases que implementan MOEAs (la versión actual incluye una implementación de NSGA-II). El paquete *problems* contiene las clases que implementan codificaciones binarias y reales; operadores de mutacion real, *flip-bit* y polinomial, cruzamiento de un punto (SPX), de dos puntos (2PX), y binario simulado (SBX); operadores de selección (proporcional y por torneo); las restricciones y objetivos de los problemas ZDT y el operador de migración con topología de anillo unidireccional.

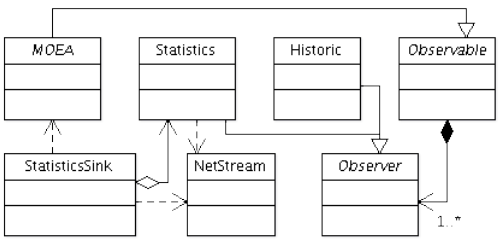

Figura 5. Vista de las clases `Statistics`, `Historic` y `StatisticsSink`.

5. Evaluación experimental

Los experimentos realizados para comprobar la correctitud del *framework* MOE y de la implementación de NSGA-II provista, analizaron la calidad de resultados y la eficiencia computacional al resolver los problemas ZDT. Se consideraron las métricas: cantidad de puntos no dominados (ND), distancia generacional (DG), *spread* y *spacing* [5] y *speedup* de la versión paralela para diferentes números de islas. El análisis se realizó sobre 10 computadores homogéneos con Intel Pentium D a 2.8GHz (doble núcleo), 1GB de RAM, Linux Fedora Core 6 de 64 bits, red FastEthernet 100 Mb. y MPICH versión 1.2.7.

En los experimentos se usó *SBX* y mutación polinomial para codificación real, y *2PX* y mutación *flip bit* para codificación binaria. Se tomaron los valores de parámetros para el NSGA-II utilizados en trabajos previos [14,15]: población de 400 individuos, criterio de parada de 500 generaciones, probabilidad de recombinación 0.9, probabilidad de mutación $1/\#variables$, migración de 4 individuos con una frecuencia e 25 generaciones, y 20 interacciones panmícticas. Se utilizó una población de gran tamaño para demandar un poder de cómputo que permitiera apreciar las virtudes del paralelismo, dada la relativa sencillez de los problemas abordados.

Calidad de resultados El cuadro 1 reporta el promedio y desviación estándar (σ) de las métricas obtenidos en 50 ejecuciones independientes del NSGA-II serial y paralelo (utilizando cuatro islas), para cada problema ZDT. Se omiten los valores de *spread* y *spacing* para *ZDT5*, porque su frente de Pareto es discreto.

Los resultados exitosos para los problemas ZDT determinan la correctitud funcional de la implementación de MOE y de la implementación de NGSA-II provista. Al comparar los resultados con los obtenidos por PNSGA-II (utilizando los mismos operadores y parámetros) se observa que MOE obtiene mejores valores de ND, *spread* y *spacing*, pero tiene peores valores de distancia generacional para la mayoría de los problemas estudiados. Este comportamiento indica la necesidad de un análisis más profundo de la implementación de NSGA-II, tarea que se propone como trabajo futuro. El algoritmo paralelo alcanza resultados de alta calidad, aunque similares a los obtenidos por la versión secuencial.

problema	modelo		ND	DG	spacing	spread
ZDT1	serial	prom.	396,96	0,000070	0,0753	0,0026
		σ	1,76	0,000006	0,0007	0,0009
	paralelo	prom.	396,56	0,000072	0,0753	0,0027
		σ	2,18	0,000010	0,0009	0,0010
ZDT2	serial	prom.	396,74	0,000026	0,0752	0,0023
		σ	1,70	0,000001	0,0008	0,0013
	paralelo	prom.	396,76	0,000025	0,0751	0,0028
		σ	1,78	0,000001	0,0009	0,0016
ZDT3	serial	prom.	398,22	0,000162	0,3172	0,1002
		σ	1,22	0,000003	0,0039	0,0020
	paralelo	prom.	398,50	0,000162	0,3176	0,1003
		σ	1,16	0,000010	0,0043	0,0017
ZDT4	serial	prom.	393,72	0,000070	0,0748	0,0030
		σ	2,34	0,000004	0,0008	0,0011
	paralelo	prom.	394,54	0,000070	0,0750	0,0030
		σ	2,07	0,000005	0,0007	0,0011
ZDT5	serial	prom.	53,76	0,022580	N/A	N/A
		σ	1,78	0,021282		
	paralelo	prom.	55,60	0,007117		
		σ	10,56	0,012770		
ZDT6	serial	prom.	330,74	0,000160	0,0778	0,0036
		σ	6,38	0,000005	0,0009	0,0008
	paralelo	prom.	338,64	0,000161	0,0773	0,0035
		σ	5,92	0,000004	0,0012	0,0009

Cuadro 1. Resultados numéricos para los problemas ZDT.

Eficiencia Computacional El cuadro 2 presenta los tiempos de ejecución (en ms.) y speedup al resolver los problemas ZDT usando 1, 2, 4 y 8 subpoblaciones en el NSGA-II paralelo. En el *escenario A* se usaron los valores paramétricos presentados anteriormente, mientras que en el *escenario B* se duplicó el número de generaciones para evaluar la escalabilidad del modelo paralelo al incrementar el tiempo de cómputo.

La Figura 6 presenta los valores de tiempos de ejecución y speedup promedio (sobre los seis problemas ZDT) al utilizar diferente número de subpoblaciones. Se observa una disminución significativa del tiempo de ejecución al utilizar 2 y 4 islas. Teóricamente, la mejora el desempeño mejora al incorpor más islas, hasta que las comunicaciones y sincronizaciones influyan en el tiempo de ejecución total. En el escenario A el umbral se alcanza al utilizar 4 islas, mientras que en el escenario B se alcanza para 8 islas. Ambos escenarios reportan speedup superlineal, un comportamiento observado con frecuencia al paralelizar AEs [1,15].

6. Conclusiones

Este trabajo ha presentado MOE, un *framework* para la resolución de MOPs utilizando MOEAs, que incorpora técnicas de procesamiento paralelo de una manera amigable y transparente para el usuario. La implementación de MOE está basada en un diseño modular con paquetes y clases poco acopladas, haciendo a MOE preparado para el cambio. MOE brinda una interfaz sencilla, que lo hace utilizable por una amplia gama de usuarios.

escenario	# islas		ZDT1	ZDT2	ZDT3	ZDT4	ZDT5	ZDT6
escenario A	1 (serial)	prom.	20205,7	20371,5	20343,2	22043,8	20963,2	21337,0
		σ	117,9	62,5	83,7	52,7	329,7	74,7
	2	prom.	7923,3	7910,9	8060,3	8073,0	7381,6	7794,0
		σ	325,974	290,726	324,051	266,434	365,979	344,714
		speedup	2,55	2,58	2,52	2,73	2,84	2,74
	4	prom.	5023,0	5199,0	5192,5	4778,2	4422,0	4687,5
		σ	347,234	518,063249	564,873	161,507	173,767	145,374
		speedup	4,02	4,02	3,92	4,61	4,74	4,55
	8	prom.	6109,0	6149,3	6157,5	6484,5	6235,7	6129,6
		σ	870,159	491,080	446,753	1218,659	1113,704	674,786
		speedup	3,31	3,31	3,30	3,40	3,36	3,48
escenario B	1 (serial)	prom.	60417,3	60801,9	61044,6	65757,0	64033,0	63088,1
		σ	168,6	243,0	191,7	144,8	309,2	532,8
	2	prom.	19830,8	19934,5	19959,5	19844,9	18327,0	19166,7
		σ	202,1	160,2	115,9	145,5	172,2	71,3
		speedup	3,05	3,05	3,06	3,31	3,49	3,29
	4	prom.	8878,9	8884,9	8874,1	8422,9	7460,0	8254,1
		σ	156,3	180,7	104,3	168,8	232,1	407,5
		speedup	6,80	6,84	6,88	7,81	8,58	7,64
	8	prom.	7460,2	7360,6	7408,0	7181,4	6621,2	6743,9
		σ	709,8	159,6	251,7	719,2	483,0	165,1
		speedup	8,10	8,26	8,24	9,16	9,67	9,35

Cuadro 2. Tiempos de ejecución y speedup.

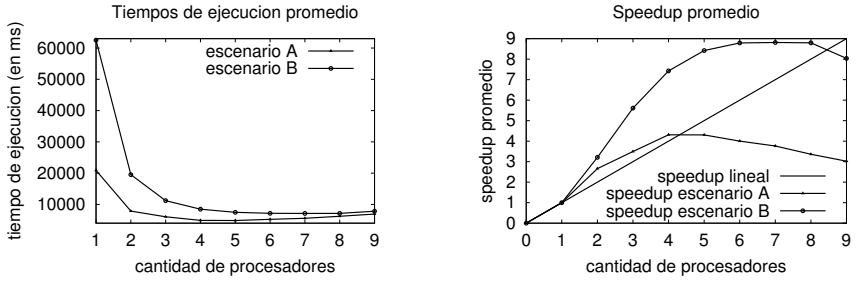

Figura 6. Tiempos de ejecución promedio y evolución del *speedup*.

Los resultados de la evaluación sobre el conjunto de problemas estándar ZDT mostraron la correctitud del diseño de MOE y de la implementación de NSGA-II provista. MOE alcanzó mejores resultados que PNSGA-II para tres de las cuatro métricas de calidad de soluciones analizadas, y obtuvo valores de *speedup* superlineal al trabajar con poblaciones numerosas.

Las líneas de trabajo futuro se orientan a consolidar MOE como herramienta, incorporando nuevos algoritmos, problemas y operadores. En particular, se planea incorporar otros MOEAs y analizar las implementaciones con respecto a las versiones de referencia. El NSGA-II incluido en MOE se está usando en varios proyectos para resolver MOPs realistas y verificar la capacidad de resolución exitosa y eficiente que fue reportada para los problemas ZDT en este artículo.

Referencias

1. E. Alba. Parallel evolutionary algorithms can achieve super-linear performance. *Inf. Process. Lett.*, 82(1):7–13, 2002.
2. E. Alba, F. Almeida, M. Blesa, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, C. León, G. Luque, and J. Petit. Efficient parallel lan/wan algorithms for optimization. the mallba project. *Parallel Computing*, 32(5-6):415–440, 2006.
3. E. Alba and M Tomassini. Parallelism and genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, 2002.
4. E. Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.
5. C. Coello, D. Van Veldhuizen, and G. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.
6. K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Proc. of the PPSN VI Conference*, pages 849–858, Paris, France, 2000. Springer.
7. K. Deb, P. Zope, and A. Jain. Distributed Computing of Pareto-Optimal Solutions with Evolutionary Algorithms. In *2nd International Conference on Evolutionary Multi-Criterion Optimization*, pages 534–549, Faro, Portugal, 2003. Springer.
8. S. Duarte, B. Baran, and B. Benitez. Telecommunication network design with parallel multiobjective evolutionary algorithms. In *Proc. of IFIP/ACM Latin America Networking Conference*, pages 1–11, 2003.
9. D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
10. R. Mäkinen, P. Neittaanmäki, J. Periaux, M. Sefrioui, and J. Toivanen. Parallel genetic solution for multiobjective MDO. In *Proc. of Parallel CFD'96 Conference*, pages 352–359, 1996.
11. N. Marco, S. Lanteri, J. Desideri, and J. Pariaux. A parallel genetic algorithm for multi-objective optimization in computational fluid dynamics. In *Evolutionary Algorithms in Engineering and Computer Science*, chapter 2, pages 445–456. Wiley, Chichester, UK, 1999.
12. A. Nebro, J. Durillo, and E. Alba. jMetal: a Java Framework for Developing Multi-Objective Optimization Metaheuristics. Technical Report ITI-2006-10, Universidad de Málaga, 2006.
13. A. Nebro, F. Luna, and E. Alba. DEME: DistributEd MEtaheuristics, 2006. Web site: http://neo.lcc.uma.es/software/deme. Consultado en julio de 2007.
14. S. Nesmachnow. Una Versión Paralela del Algoritmo Evolutivo para Optimización Multiobjetivo NSGA-II. In *Actas del X Congreso Argentino de Ciencias de Computación*, pages 1933–1944, La Matanza, Argentina, 2004.
15. S. Nesmachnow. Un algoritmo evolutivo multiobjetivo paralelo aplicado al diseño de redes de comunicaciones confiables. In *Actas del IV Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, pages 145–152, Granada, España, 2005.
16. N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.
17. D. Van Veldhuizen, J. Zydallis, and G. Lamont. Considerations in engineering parallel multiobjective evolutionary algorithms. *Evolutionary Computation*, 7(2):144–173, 2003.
18. E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.