Una versión paralela del NSGA II utilizando multi-threads

Pablo Ezzatti
CeCal, Facultad de Ingeniería.
Universidad de la República, Uruguay
pezzatti@fing.edu.uy

Resumen

El trabajo presenta una versión paralela mediante estrategias de multi-threads, del algoritmo evolutivo para optimización multiobjetivo NSGA-II. Se muestran los detalles de diseño e implementación de la versión paralela, en la que se define una estructura de vecindad la cual estipula la interacción entre las distintas sub-poblaciones. Se analiza la calidad de resultados y la eficiencia computacional, comparando con los resultados y tiempos de ejecución de la versión secuencial del algoritmo NSGA-II sobre un conjunto de problemas de prueba estándar.

Palabras clave: Algoritmos Evolutivos, Optimización Multiobjetivo, Paralelismo, NSGA-II.

Destinado al Workshop de Agentes y Sistemas Inteligentes.

I. Introducción

La mayor parte de los problemas de optimización presentes en la realidad constan de varias funciones objetivo que deben satisfacerse simultáneamente y que normalmente se encuentran en conflicto entre sí. Históricamente, con el fin de simplificar la resolución de dichos problemas, el enfoque ha sido transformarlos a problemas de una función objetivo (mono-objetivo).

Los algoritmos evolutivos (AE), se muestran como una alternativa interesante para la resolución de problemas multi-objetivo (PMO), debido a su cualidad de trabajar sobre una población de soluciones (factibles). Básicamente, el mecanismo de funcionamiento de los EA consiste en emular el proceso evolutivo de las especies sobre una población de soluciones del PMO, seleccionando un número predefinido de ellos en base a su aptitud, aplicándoles operadores de recombinación y mutación para producir una nueva generación que sustituye a la anterior. En los últimos años se generaron variadas propuestas de algoritmos evolutivos multiobjetivo (AEMO), cuyos detalles se presentan en los relevamientos realizados en [2,3,7,18].

Uno de los AEMO más difundido y que ha mostrado obtener resultados de buena calidad sobre una amplia gama de PMO, es el Non-dominated Sorting Genetic Algorithm versión II (NSGA II) [6] desarrollado por Deb y sus colegas en el año 2000.

Los grandes esfuerzos computacionales requeridos al abordados problemas de gran complejidad con técnicas evolutivas, hace razonable la aplicación de técnicas de procesamiento paralelo y distribuido para su abordaje. En este contexto, múltiples versiones de AE paralelos se han desarrollado que han mostrado su eficiencia y eficacia para la resolución de complejos problemas de optimización combinatoria. En el área de los AEMO paralelos paradójicamente no se ha alcanzado tal nivel de desarrollo pese a que generalmente la resolución de los problemas multiobjetivo requiere mayor capacidad de cómputo que los monoobjetivo.

Este trabajo presenta una propuesta de paralelización del algoritmo NSGA II. Mediante una estrategia híbrida que combina características de los modelos celulares y de población distribuida [1], desarrollada sobre una arquitectura multiprocesador múltiple instrucción - múltiple dato.

La sección 2 presenta una introducción a los problemas multiobjetivo. La sección 3 describe el algoritmo NSGA II. En la sección 4 se ofrece una breve introducción a los algoritmos evolutivos paralelos. En la sección 5 se explica la propuesta de paralelismo aplicado al algoritmo NSGA II. Los problemas estudiados y los resultados experimentales se presentan y comparan en la sección 6. Por último, la sección 7 ofrece las conclusiones y propuestas de trabajo futuro.

II. Planteo del Problema de optimización multiobjetivo

Un PMO [7] plantea un conjunto de N variables de decisión, M funciones a maximizar (minimizar), un conjunto de J restricciones de desigualdad y K restricciones de igualdad. En la Figura 1 se resume la formulación general de un PMO.

$$
\begin{aligned}
Min./Max. \quad & f_m(x), && m=1,2,\ldots,M;\\
sujeto\ a: \quad & g_j(x) \geq 0, && j=1,2,\ldots,J;\\
& h_k(x) = 0, && k=1,2,\ldots,K;\\
& x_i^{(L)} \leq x_i \leq x_i^{(U)}, && i=1,2,\ldots,N.
\end{aligned}
$$

Figura 1. Problema de optimización multiobjetivo.

Las restricciones determinan el conjunto de soluciones factibles del problema. Cada vector solución X, está compuesto por N variables $X = (x_1,x_2,\ldots,x_N)$.

Cuando se trabaja en problemas de optimización mono-objetivo, el criterio de optimalidad es inmediato y está dado por la función a optimizar. Sin embargo, en el caso de los PMO se necesitan otros criterios para comparar soluciones, debido a que una buena solución para uno de los objetivos puede no ser necesariamente el óptimo o ni siquiera un buen valor para otro de los objetivos. Por este motivo se trabaja con la noción de dominancia de Pareto.

Una solución factible x1 para un PMO se dice no dominada u óptima en el sentido de Pareto, si no existe otra solución factible x2 que cumpla que:

1. x2 tiene los mismos o mejores valores que x1 para cada función objetivo en el modelo.
2. x2 tiene estrictamente mejor valor que x1 para al menos una función objetivo.

El conjunto de todas las soluciones no dominadas define, en el espacio de las funciones objetivo, el frente óptimo ó de Pareto. En la Figura 2 se puede observar, destacado por una línea negra gruesa, un ejemplo de frente de Pareto de un problema genérico de minimización de dos funciones objetivo.

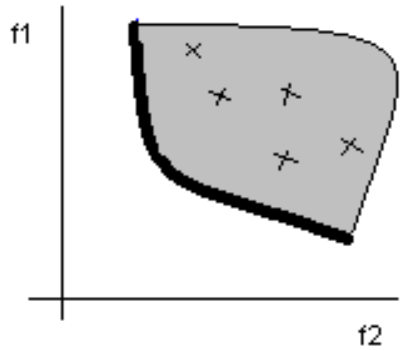

Figura 2. Frente de Pareto.

III. NSGA II

En esta sección se presenta una breve introducción a los AE y su aplicación a los PMO, en particular al NSGA y su sucesor el NSGA II, algoritmo base del trabajo realizado.

Algoritmos evolutivos

Los AE son métodos eficientes de búsqueda, pertenecientes a la familia de las heurísticas. Inspirados en el proceso de evolución de los seres vivos, utilizan el principio de selección natural para la resolución de problemas. Los AE son capaces de realizar búsquedas robustas y multidireccionales en espacios complejos trabajando con una *población* de soluciones potenciales, donde cada *individuo* es la codificación de una posible solución del problema. Una virtud de este tipo de algoritmos es que requieren poca información del problema y permiten generalmente la obtención de buenas soluciones aproximadas en tiempos razonables.

```
t=0;
Inicializar la población P(t);
Evaluar P(t),
Mientras no criterio de parada
        t=t+1;
        P(t) = seleccionar P(t-1)
        Recombinar P(t)
        Mutar P(t)
        Evaluar P(t)
Fin mientras
```

Cuadro 1. Pseudocódigo de un AG simple.

La evolución se basa en la selección de los mejores individuos de acuerdo a criterios de evaluación de la función de sus capacidades de resolución del problema, dadas por una función de *fitness*, sobre los cuales se aplican operadores de recombinación y mutación para explorar el espacio de búsqueda. Existen diversas maneras de realizar la selección, la recombinación y la mutación, así como diversos operadores desarrollados para problemas particulares. Goldberg [10] presento el algoritmo genético simple cuyo pseudocódigo se puede observar en el Cuadro 1.

Algoritmos evolutivos multiobjetivos

Los AEMO surgieron como una extensión natural de los algoritmos evolutivos mono-objetivos. Los dos requisitos deseables de un AEMO son: que conduzca la búsqueda hacia el frente de Pareto y que mantenga la diversidad de la población en esta frontera.

La característica de los algoritmos evolutivos de mantener una población de soluciones es una condición sumamente aprovechable para la resolución de PMO, debido a que la solución de este tipo de problemas no está dada por un único valor, sino por un conjunto de valores.

Una primera aproximación simple es utilizar la función de *fitness* como función de acumulación de los distintos objetivos. Extensiones más completas utilizan el operador de selección en base al criterio de no dominancia, así como estrategias para preservar la diversidad de soluciones en el frente de Pareto (operadores de nichos, corwding, sharing, etc) [7].

Los AEMO se pueden dividir en dos categorías: los de primera generación, que no incorporan elitismo y los de segunda generación que poseen mecanismos explícitos de elitismo.

NSGA

Srinivas y Deb desarrollaron en el año 1994 el AEMO Non-dominated Sorting Genetic Algorithm (NSGA) [13] implementado en base al concepto de orden de no dominancia propuesto originalmente por Goldberg [10].

El algoritmo se compone de operadores de cruzamiento y de mutación simple y de un operador de selección basado en el criterio de la ordenación de los individuos en los distintos frentes de dominancia y distancia de Sharing para la distribución de los individuos a lo largo de cada frente.

El procedimiento de asignación de fitness ordena los individuos en los distintos frentes. Luego para cada solución, se le asigna un valor de fitness dependiendo del frente en que se encuentra y se le degrada dividiéndolo por un valor de "conteo de nicho". Definido como la sumatoria de las distancias de cada solución a las restantes soluciones que pertenezcan al mismo frente.

El costo computacional del algoritmo esta determinado, por el peor de los costos entre el procedimiento de ordenar por no dominancia y del cálculo de la función de Sharing. La primera tarea posee un orden de complejidad $O(MP^2)$ (siendo M la cantidad de objetivos y P el tamaño de la población), mientras que el cálculo de la función de Sharing necesita que se compare cada solución contra todas las otras soluciones del mismo "nivel" de frente, lo que implica $O(LP^2)$ operaciones siendo L la cantidad de variables del problema.

NSGA- II

El AEMO de segunda generación Non-dominated Sorting Genetic Algorithm versión II (NSGA-II) fue propuesto por Deb en 2000 [6] a partir del NSGA, modificando el mecanismo de preservación de la diversidad e incorporando entre otros, un mecanismo explícito de elitismo.

Se abandona la utilización de distancia de Sharing manejada por el NSGA, para emplear un operador de selección por torneo de Crowding como método de preservación de diversidad. El mismo trabaja de la siguiente forma:

- Si un individuo *i* tiene mejor ranking (pertenece a un frente mejor) que un individuo *j*, gana *i*.

- Sino, se evalúa la distancia de crowding, considerando el tamaño del mayor paralelogramo que circunda cada individuo (en el espacio de búsqueda) sin incluir cualquier otro de la población; gana el individuo que posea mayor distancia.

La forma de trabajo es simple, en cada generación el algoritmo identifica los distintos frentes, selecciona los individuos de los primeros *F* frentes hasta obtener *H* inividuos, si con el frente *F* se excede de *P* individuos, se elige por distancia de crowding la cantidad necesaria. Una vez obtenido el "maiting pool" se generan *H* pares y se elige por torneo de crowding, luego se recombina y muta. En el Cuadro 2 se puede observar el pseudocódigo del algoritmo NSGA-II.

```
t=0;
Inicializar P(t);
Mientras no termine
        t=t+1;
        Identificar los distintos frentes en P(t-1)
        P(t) = seleccionar por torneo de Crowding P(t-1)
        Recombinar P(t)
        Mutar P(t)
Fin mientras
```

Cuadro 2. Pseudocódigo del algoritmo NSGA II.

El algoritmo posee un costo computacional de orden $O(MP^2)$, siendo *M* el número de objetivos considerado y *P* el tamaño de la población.

IV. Algoritmos Evolutivos Paralelos

Los grandes esfuerzos computacionales requeridos al abordar problemas de gran complejidad con técnicas evolutivas, hacen razonable la aplicación de técnicas de procesamiento paralelo y distribuido a los algoritmos evolutivos.

Existe una estrecha vinculación entre las distintas estrategias de paralelismo aplicadas a los algoritmos evolutivos y las arquitecturas de hardware que se utilizan en la práctica. Debido a esta relación a continuación se presentamos algunos conceptos de arquitecturas de computadores.

Una de las taxonomías más difundidas para clasificar las arquitecturas de las computadoras es la propuesta por Flynn [9] que clasifica los computadores paralelos considerando la estructura de procesamiento y el esquema de datos, las cuatro categorías que define son las siguientes:

- mono instrucción - mono dato (single instruction - single data SISD).
- mono instrucción - múltiple dato (single instruction - multiple data SIMD).
- múltiple instrucción - mono dato (multiple instruction - single data MISD).
- múltiple instrucción - múltiple dato (multiple instruction - multiple data MIMD).

En la paralelización de algoritmos evolutivos existe un largo camino recorrido. Una clasificación posible en base a la cantidad de poblaciones utilizadas, la interacción entre los individuos de la/las población/es y el sistemas de comunicación, es la que propone Cantú-Paz [1]:

- **paralelización global** también llamado modelo maestro-esclavo, en donde se mantiene una única población en un proceso y se efectúan evaluaciones de la función de *fitness* en paralelo. Con esta metodología no se afecta la operativa del mecanismo evolutivo y se consiguen los mismos resultados que con el algoritmo secuencial.
- **grano grueso** también se lo describe como modelo distribuido, modelo de islas o multi-poblacional. En este caso se divide la población en múltiples subpoblaciones que evolucionan aisladas, intercambiando ocasionalmente algunos individuos mediante un operador de migración.

- **grano fino** o modelo celular, consiste en distribuir la población entre las unidades de proceso (el ideal considera un individuo por elemento de proceso) y definir una estructura de vecindad, la cual estipula la interacción entre los distintos individuos. Este modelo está muy vinculado a la arquitectura de la máquina en la que se procesa, y está estrechamente asociado a las máquinas masivamente paralelas.
- **híbridos jerárquicos**, que combinan esquemas de multi-población con estrategias de paralelización global o de grano fino.

En los AEMO las posibilidades de aplicar técnicas de cálculo paralelo y distribuido son aún mayores que en el caso de la optimización mono objetivo. Además de las posibilidades descriptas anteriormente, se dispone de diversas opciones de "descomposición de dominio", por ejemplo que cada procesador se encargue de una sub-zona del frente, que priorice un objetivo, etc, en [4,8,17,19] se pueden observar distintos ejemplos de paralelismo aplicado a MOEAS.

Cantú-Paz [1] y Coello et al. [3] han presentado revisiones de técnicas de paralelismo aplicadas a los AE y los AEMO respectivamente.

V. Propuesta

La propuesta consiste en una versión paralela del algoritmo NSGA II utilizando librerías de multi-threads. Está basada en los códigos del NSGA II desarrollados por Deb et al en el Kanpur Genetic Algorithms Laboratory (KanGAL) [11].

El trabajo se diseñó para ejecutar sobre la máquina Sun SPARCcenter 2000, disponible en la Facultad de Ingeniería de la Universidad de la Republica, Uruguay. La computadora posee 16 Microprocesadores Super Scalable Processor ARChitecture (SuperSPARC II) versión 8 de 85 MHz (arquitectura RISC), y se clasifica como MIMD en la taxonomía de Flynn.

Si bien el poder de cómputo de la máquina es hoy en día inferior al de cualquier PC de escritorio, el poseer 16 procesadores con acceso independiente a memoria la transforma en una importante herramienta para la investigación en procesamiento paralelo.

Para la implementación se manejaron las primitivas de comunicación CrearProceso, ReunirProcesos y Semáforos, presentes en todos los sistemas de programación paralelos multithreading. En nuestro caso se utilizó la librería pthread de C [15], la cual implementa las primitivas mencionadas mediante las funciones pthread_create, pthread_join, pthread_mutex_lock y pthread_mutex_unlock.

Basados en el computador de 16 procesadores y su memoria compartida se diseñó un algoritmo evolutivo paralelo, clasificable dentro de la clase de los híbridos según la taxonomía propuesta por Cantú-Paz [1]. Fundamentalmente se dispone de un proceso central que crea *T* threads, tantos como sean especificados en la configuración, cada uno de estos *T* threads ejecuta el algoritmo NSGA-II con una población propia de tamaño igual a *p/T*, siendo *p* la población total del algoritmo (grano grueso), mientras que se especifica una estructura de vecindad para la interacción entre las distintas poblaciones (celular).

El Cuadro 3 resume la operativa del algoritmo. En la etapa de inicialización, el proceso central realiza un ciclo que itera tantas veces como procesos se desee tener, en el cual se genera información específica para cada thread.. Luego de inicializados todos los procesos, el sistema se puede ver como conectado por una grilla toroidal, donde cada procesador es vecino de otros cuatro tal como se presenta en la Figura 3. Cada thread trabaja en forma semi-independiente, comunicando solamente un individuo cada cierta cantidad de generaciones a sus cuatro vecinos, utilizando primitivas no bloqueantes.

```
Comienzo
Para i:=1 hasta T
        Generar info para el Thread i
        P(s)
        Crear el Thread i
Fin Para

Para i:=1 hasta T
        Recibir el Thread i
        % recepción bloqueante
Fin Para
NSGA II con la unión de las poblaciones.
```

Cuadro 3-a. Pseudocódigo del proceso central.

```
Obtener info central
V(s)
NSGA II con intercambio de individuos y
población p/T
Devolver resultado
```

Cuadro 3-b. Pseudocódigo de los procesos en c/thread,

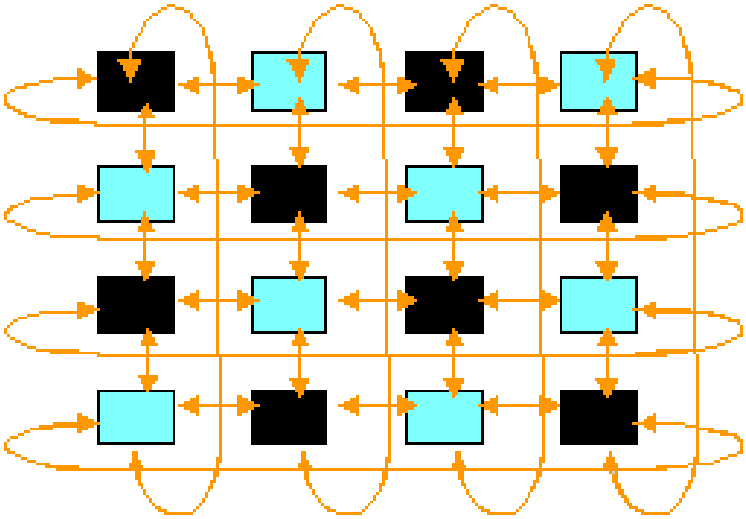

Figura 3. Esquema de comunicación entre los procesos.

Se pueden adoptar diversos criterios de selección del individuo a comunicar. Entre otros, puede elegirse un individuo en forma aleatoria, tomar el individuo con menor distancia de crowding en el primer frente, seleccionar basándose en alguno de los objetivos, etc. En la propuesta implementada se adoptó la política de seleccionar un individuo que pertenezca al frente no dominado y tenga valor mínimo en la primer o segunda función objetivo. La función a ponderar se escoge dependiendo de la suma de las coordenadas del procesador en la grilla. Si el resultado es par se escoge el individuo que minimiza la función 1, si es impar se escoge el que minimiza la función 2. Este criterio no es adecuado para abordar problemas con más de dos objetivos.
En cada generación, también se obtienen los cuatro "peores" individuos utilizando los criterios antes descriptos, se revisa si algún procesador vecino envió algún individuo y de haberlo hecho, se le intercambia por uno de los cuatro individuos antes mencionados.
La forma de interacción entre los individuos utiliza el paradigma de memoria compartida. Manejando el algoritmo un espacio de memoria de tamaño 4 * tamaño de individuo * cantidad de procesos, disponiendo cada proceso de un espacio de 4 individuos en el cual interactúa accediendo en forma mutuo-excluda con sus 4 vecinos.

Una vez culminados los *T* threads, el proceso central obtiene la población de cada uno de ellos y a esta nueva población le aplica el algoritmo NSGA II con un número pequeño de generaciones. Es necesario aplicar el algoritmo a la unión de las poblaciones para mejorar la cantidad y la distribución en el frente de individuos no dominados, ya que mientras se ejecuta en paralelo el

control de no dominancia y el mecanismo de diversidad se aplican en cada sub-población en forma independiente.

VI. Resultados

En esta sección se describen los problemas utilizados para evaluar la versión paralela del algoritmo desarrollado y los resultados obtenidos al resolverlos.

El objetivo de los experimentos consistió en estudiar la calidad de las soluciones obtenidas por las versiones paralelas y la relación de desempeño entre las distintas configuraciones (cantidad de procesadores) paralelas y la secuencial.

Problemas

Se utilizaron tres problemas estándar en el área de optimización multiobjetivo, el problema de Kursawe (KUR) que data 1990, el problema propuesto en el año 2000 por Poloni (POL) y el problema 5 de Zitzler, Deb y Thiele (ZDT5) [5] presentado en el año 2000.

El problema KUR se formula en la Figura 4, posee tres variables, dos funciones objetivo y su frente de Pareto es desconexo en 4 partes.

$$\text{KUR:}\begin{cases} \textit{Minimizar} & f_1(x) = \sum_{i=1}^{2}\left[-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2})\right] \\ \textit{Minimizar} & f_2(x) = \sum_{i=1}^{3}\left[|x_i|^{0.8} + 5\sin(x_i^3)\right] \\ & -5 \le x_i \le 5, i = 1,2,3 \end{cases}$$

Figura 4. Problema de Kursawe.

El problema POL se presenta en la Figura 5, posee dos variables, dos funciones objetivo y posee frente de Pareto desconexo en dos tramos

$$\text{POL:}\begin{cases} \textit{Minimizar} & f_1(x) = \left[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2\right] \\ \textit{Minimizar} & f_2(x) = \left[(x_1 + 3)^2 + (x_2 + 1)^2\right] \\ \textit{donde} & A_1 = 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2, \\ & A_2 = 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2 \\ & B_1 = 0.5\sin x_1 - 2\cos x_1 + \sin x_2 - 1.5\cos x_2 \\ & B_1 = 1.5\sin x_1 - \cos x_1 + 2\sin x_2 - 0.5\cos x_2 \\ & -\pi \le (x_1, x_2) \le \pi \end{cases}$$

Figura 5. Problema de Poloni.

El problema ZDT5 se ofrece en la Figura 6, posee once variables discretas, dos funciones objetivo discretas y su frente de Pareto es discreto se compone de 31 valores.

$$\text{ZDT5:}\begin{cases} \textit{Minimizar} & f_1(x), \\ \textit{Minimizar} & f_2(x) = g(x)h(f_1(x), g(x)) \\ \textit{donde} & \\ & f_1(x) = 1 + u(x_1) \\ & g(x) = \sum_{i=2}^{11} v(u(x_i)) \\ & v(u(x_i)) = \begin{cases} 2 + u(x_i) & si\ \ u(x_i) < 5 \\ 1 & si\ \ u(x_i) = 5 \end{cases} \\ & h(f_1, g) = 1/f_1(x) \end{cases}$$

Figura 6. Problema de Poloni.

Resultados

Se evaluaron las versiones serial y paralela con 4, 8 y 16 procesadores del NSGA II aplicadas a los problemas planteados, sobre la máquina Sun SPARCcenter 2000. En todos los casos se efectuaron 20 ejecuciones independientes.

Se utilizaron diversas métricas para evaluar la calidad de los resultados y eficiencia de los algoritmos. Para evaluar la cercania al frente de Pareto se utilizó la métrica error ratio – ξ (una variante de error ratio) debido a que no se disponía de los frentes verdaderos sino de frentes calculados mediante algoritmos enumerativos [15] con determinada precisión, por lo tanto para ésta métrica se considera que un punto pertenece al frente de Pareto real si se encuentra a una distancia menor a un valor ξ de alguno de los puntos del frente calculado. Otras métricas usuales como distancia generacional, spacing y spread fueron utilizadas para evaluar la calidad de resultados. Para evaluar la performance de los algoritmos se utilizaron las tradicionales medidas de speed up y eficiencia.

En la Tabla 1 se pueden observar los valores de los parámetros con los que se ejecutó el algoritmo para los problemas KUR y POL, mientras que en la Tabla 2 se presentan los valores para el problema ZDT5. Como criterio de parada se utilizaron 500 generaciones para la versión serial, mientras que en las versiones paralelas se utilizaron 495 generaciones trabajando en forma distribuida, más 5 en conjunto.

Codificación	Binaria
Bits utilizados para codificar c/variable.	20
Cantidad total de generaciones.	500
Población Total.	512
Probabilidad de cruzamiento.	0.9
Tipo de cruzamiento.	1 punto
Probabilidad de mutación.	0.025
Frecuencia de migración (generaciones).	10

Tabla 1. Parámetros para los problemas KUR y POL.

Codificación	Binaria
Bits utilizados para codificar c/variable.	1v-30,10-5
Cantidad total de generaciones.	500
Población Total.	512
Probabilidad de cruzamiento.	0.9
Tipo de cruzamiento.	1 punto
Probabilidad de mutación.	0.025

Tabla 2. Parámetros para el problema ZDT5.

Los resultados de las distintas configuraciones del algoritmo para los problemas KUR y POL se presentan en la Tabla 3, mientras que los resultados del problema ZDT5 se muestran en la Tabla 4.

Prolemab	***Proc.***		***Individuos no dominados***	***Spacing***	***Spread***	***Distancia Generacional***	***Error ratio - 0.02***
KUR	1	Promedio	512	0,139	0,547	0,014	0,148
		Desv. Est.	0	0,003	0,019	0,001	0,011
	4	Promedio	405,5	0,141	0,749	0,011	0,156
		Desv. Est.	8,83	0,005	0,025	0,001	0,014
	8	Promedio	370,25	0,145	0,858	0,013	0,216
		Desv. Est.	15,57	0,007	0,033	0,001	0,027
	16	Promedio	294	0,160	0,932	0,018	0,207
		Desv. Est.	20,06	0,012	0,038	0,008	0,026
POL	1	Promedio	511	0,171	0,526	0,010	0,119
		Desv. Est.	0,37	0,003	0,28	0,001	0,012
	4	Promedio	511,3	0,170	0,488	0,011	0,108
		Desv. Est.	0,44	0,004	0,040	0,001	0,011
	8	Promedio	511,2	0,165	0,518	0,011	0,118
		Desv. Est.	0,62	0,002	0,025	0,001	0,016
	16	Promedio	511,35	0,160	0,585	0,011	0,124
		Desv. Est.	0,59	0,006	0,039	0,001	0,013

Tabla 3. Resultados para los problemas KUR y POL.

En la Tabla 3 se puede observar que el problema POL presenta resultados de calidad similar para todas las versiones. Por otra parte, las soluciones obtenidas para el problema KUR en las versiones paralelas son de calidad inferior a las obtenidas por la versión secuencial. Dicho problema posee el frente de Pareto desconexo en cuatro partes, siendo más difícil lograr una buena distribución.

Problema	***Proc.***		***Individuos no dominados***	***Distancia generacional***
ZDT5	1	Promedio	512	0,014
		Desv. Est.	0	0
	4	Promedio	405,5	0,011
		Desv. Est.	8,83	0,001
	8	Promedio	370,25	0,013
		Desv. Est.	15,57	0,001
	16	Promedio	294	0,018
		Desv. Est.	20,06	0,008

Tabla 4. Resultados para el problema ZDT5.

Analizando los resultados presentados en la Tabla 4 es posible concluir que el problema ZDT5 tuvo un comportamiento similar al problema KUR, obteniendo las versiones paralelas resultados de calidad sensiblemente inferior a las obtenidas por la versión serial, disminuyendo la calidad de las soluciones a medida que se utiliza una cantidad mayor de procesadores.

Problemas		***POL***			***KUR***			***ZDT5***		
Modelo serial	*Tiempo (s)*	1442			1374,95			1305,1		
	Desv. Est.	81,14			12,25			35,96		
Modelo Paralelo	*Procesadores*	**4**	**8**	**16**	**4**	**8**	**16**	**4**	**8**	**16**
	Tiempo (s)	118,4	52,3	37,1	130,2	58,05	40,25	152,2	82,75	53,25
	Desv. Est.	4,06	0,47	0,45	5,34	1,57	0,44	1,52	3,21	1,55
Medida	*Speedup*	12,18	27,57	38,87	10,56	23,69	34,16	8,57	15,77	24,51
	Eficiencia	3,05	3,45	2,43	2,64	2,96	2,14	2,14	1,97	1,53

Tabla 5. Efectos del paralelismo.

En cuanto a la eficiencia computacional la versión paralela, la Tabla 5 muestra el excelente desempeño de las tres configuraciones consideradas, alcanzando, todas ellas niveles de speedup super-lineal y eficiencia mayor a uno. Si bien la eficiencia fue muy buena al trabajar con 16 procesadores, la misma decayó con respecto a la conseguida trabajando con 4 u 8 procesadores. Para los casos estudiados el tope de escalabilidad se alcanza entre 8 y 16 procesadores, aunque debe mencionarse que al utilizar la totalidad de los procesadores disponibles existe una virtual degradación del desempeño computacional debido a los procesos de administración y del sistema operativo, que no fue evaluada.

VII. Conclusiones y Trabajo Futuro

El trabajo constituye un aporte en un área sumamente promisoria como es la aplicación de técnicas de paralelismo a algoritmos evolutivos multiobjetivos, en donde las propuestas han sido bastante limitadas.

Se presentó una versión paralela del algoritmo NSGA II mediante la utilización de threads sobre un equipo multiprocesador de memoria compartida (con 16 procesadores).

Se presentaron resultados preliminares de evaluación de la versión paralela desarrollada resolviendo tres problemas estándares de la literatura con distintas configuraciones del algoritmo (4, 8 y 16 procesadores), también se compararon los resultados y el desempeño con los obtenidos por la versión secuencial.

Mientras que en el caso del problema de POL todas las versiones obtuvieron resultados similares, la versión paralela obtuvo resultados de peor calidad que los de la versión secuencial al resolver los problemas KUR y ZDT5. La condición del problema KUR de poseer un frente desconexo en cuatro partes y la cualidad del frente del problema ZDT5 que es discreto y se compone de 31 valores, los presenta como problemas en los cuales es más difícil lograr una buena distribución de las soluciones. En este sentido, experimentos aún no formalizados sugieren que aumentar la interacción panmítica puede generar importantes mejoras en los resultados.

La aplicación del paralelismo mostró una mejora en cuanto a la eficiencia computacional del algoritmo, presentando en todos los casos speedup super-lineal y niveles de eficiencia superiores a uno. De confirmarse la escalabilidad de estos resultados, esta capacidad será de gran utilidad al momento de abordar problemas de optimización de gran porte, relacionados con aplicaciones de la vida real.

Una serie de aspectos sobre el diseño, la implementación y la aplicabilidad de la versión paralela del algoritmo NSGA II desarrollada admiten ser explorados con mayor detalle. En la actualidad, dentro del grupo de trabajo se están investigando algunos de estos aspectos, mientras que se prevé abordar otros en el futuro cercano.

Si bien el trabajo se centró en la aplicación de técnicas de alto desempeño a un algoritmo evolutivo multiobjetivo y las pruebas efectuadas arrojaron valores alentadores en lo referente a eficiencia computacional, haber efectuado las pruebas en base a tres problemas no permite obtener conclusiones determinantes. Por este motivo, es necesario que el conjunto de problemas de prueba sea extendido para estudiar la aplicabilidad del trabajo a problemas de dimensión mayor, con más objetivos o mayor cantidad de variables.

Como tarea pendiente debe estudiarse la evaluación del efecto de la frecuencia de migración en los resultados. En forma independiente, puede ser interesante estudiar la estipulación dinámica de la frecuencia, dependiendo de la etapa en la que se encuentra el algoritmo, tomando en cuenta las generaciones transcurridas u otros criterios a determinar.

También parece importante evaluar otros criterios para la elección del individuo a comunicar, principalmente en los casos en que los problemas a resolver son de más de dos objetivos.

Los promisorios resultados de eficiencia computacional obtenidos, motivan a realizar pruebas en máquinas que dispongan de mayor potencia y de mayor cantidad de procesadores. Este aumento en la cantidad de unidades de proceso trae aparejado el estudio del efecto de disminuir el tamaño de la población, en la calidad de los resultados.

Bibliografia

[1] Cantú-Paz E., *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publisher, 2000.

[2] Coello, C., *Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques*. Knowledge and Information Systems. International Journal, 1(3):269-308, 1999.

[3] Coello C., Van Veldhuizen D., Lamont G., *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer Academic Publisher, 2002.

[4] de Toro F., Ortega J., Fernández J. ,Díaz A., *PSFGA: A parallel Genetic Algorithm for Multiobjective Optimization*. 10th Euromicro Workshop on Parallel and Distributed Processing. Gran Canaria, 2002.

[5] Deb K., *Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems*. Evolutionary Computation 7(3): 205-230, 1999.

[6] Deb K., Agrawal S., Pratab A., Meyarivan T., *A Fast Elitist Non-Dominated Sorting GeneticAlgorithm for Multi-Objective Optimization: NSGA-II*, Proceedings of the Parallel Problem Solving from Nature VI Conference, 849-858. 2000.

[7] Deb K., *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Inc., New York, 2001.

[8] Deb K., Zope P., Jain A., *Distributed Computing of Pareto-Optimal Solutions Using Multi-ObjectiveEvolutionary Algorithms*. Report No. 2002008, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur, Setiembre 2002.

[9] Flynn M., *Some Computer Organizations and Their Effectiveness*, IEEE Trans. Comput., Vol. C-21, 94, 1972.

[10] Goldberg D., *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[11] Kanpur Genetic Algorithms Laboratory (KanGAL). Disponible en línea: *http://www.iitk.ac.in/kangal/soft.htm* Consultada marzo 2004.

[13] Kernighan B., Ritchie D, *El lenguaje de programación C*. Prentice-Hall, 1991.

[14] Srinivas N., Deb K., *Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms*.Evolutionary Computation 2(3): 221-248, 1994.

[15] The ESaM Project. Disponible en línea: *http://neo.lcc.uma.es/software/esam/problems/* Consultada marzo 2004.

[16] Nichols B., Buttlar D., Proulx J., *Pthreads Programming: A POSIX Standard for Better Multiprocessing.* O'Reilly Nutshell, 1996.

[17] Van Veldhuizen D., Lamont G., *Multiobjective Evolutionary algorithms Test Suites*, Proceedings of the 1999 ACM Symposium on Applied Computing, Janice Carrol et al., editor 351-357. 1999.

[18] Van Veldhuizen D., Zydallis J., Lamont G., *Considerations in engineering parallel multiobjective. evolutionary algorithms*. IEEE Trans. Evolutionary Computation 7(2): 144-173, 2003.

[19] Van Veldhuizen D., Lamont G., *Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art.* IEEE Trans. Evolutionary Computation, 8(2):125-147, 2000.

[20] Watanabe S., Hiroyasu T., Miki M., *Parallel Evolutionary Multi-Criterion Optimization for Block Layout Problems*. PDPTA 2000.

[21] Zitzler E., Deb K., Thiele L., *Comparison of of Multiobjective Evolutionary Algorithms: Empirical Results*. IEEE Trans. Evolutionary Computation 8(2): 173-195, 2000.