ALGORITMOS GENÉTICOS ASÍNCRONOS COMBINADOS PARA UNA RED HETEREOGÉNEA DE COMPUTADORAS

Benjamín Barán y *Enrique Chaparro*
Centro Nacional de Computación
Universidad Nacional de Asunción
Campus Universitario
San Lorenzo – Paraguay
Casilla de Correos 1439
Tel : (595) (21) 585.550 y 585.619
E-mail : bbaran@una.py

Resumen

Los algoritmos genéticos paralelos (AGP) han demostrado ser una excelente herramienta en la optimización de complejas funciones multimodales de diversa naturaleza debido a su robustez. Mucho trabajo ha sido dedicado al estudio de diversas implementaciones en computadoras paralelas constituidas por procesadores generalmente similares. Lastimosamente, estas computadoras paralelas no están fácilmente disponibles en instituciones o países de escasos recursos. En su lugar, el presente trabajo propone el aprovechamiento de las redes de computadoras hetereogéneas existentes, trabajando en un ambiente típicamente asíncrono, adaptando los algoritmos genéticos paralelos a esta realidad. Para esto, se propone combinar las tradicionales implementaciones de los AGP con diferentes algoritmos de optimización local en cada una de las máquinas del sistema computacional hetereogéneo disponible.

Inspirado en una propuesta de Mühlenbein et al. [1,2] que propone la utilización de optimizadores locales en cada uno de los procesadores de una computadora paralela (hipercubo de 16 nodos), reportando aceleraciones superlineales, la presente propuesta utiliza optimizadores locales diferenciados (como el algoritmo del gradiente, del ascenso suficiente, de métrica variable y otros), según sea el desempeño relativo del procesador que lo ejecute y el tamaño de la subpoblación correspondiente. Resultados experimentales optimizando las tradicionales funciones de De Jong [3] en una red de *workstations* y computadoras personales, demuestran la validez de la presente propuesta al obtener mejores resultados que los algoritmos tradicionalmente utilizados en un contexto paralelo, cuando implementados en una red hetereogénea de computadoras. Esto porque la implementación propuesta se adapta mejor a las características de hetereogeneidad de los procesadores y aleatoriedad de las comunicaciones en una red de computadoras.

Palabras Claves

Algoritmos Genéticos Paralelos (AGP), Combinación de Algoritmos, Asincronismo, Red Hetereogénea de Computadoras.

1. Introducción

Los Algoritmos Genéticos (AG) han demostrado ser una muy buena herramienta en la búsqueda de óptimos globales, debido a su relativa sencillez de implementación y a la robustez de su aplicación [4]; sin embargo, los AG necesitan generalmente de mucho tiempo de procesamiento para llegar a buenos resultados. Una solución a este problema es la paralelización del Algoritmo Genético, dividiendo el problema global en subproblemas que son asignados a varios procesadores de un sistema computacional. De esta forma, estos procesadores realizarán la computación del problema asignado obteniendo resultados parciales que serán transmitidos a los otros procesadores, colaborando todos para llegar a la solución global del problema[5].

Desde que se presentaron las primeras ideas acerca de la paralelización de los Algoritmos Genéticos hasta nuestros días, se han realizado numerosas implementaciones en este campo [5], destacándose las implementaciones asíncronas [1, 6] donde cada subpoblación es procesada en paralelo por otro procesador de un sistema computacional que intercambia *migrantes* con sus procesadores vecinos, siguiendo una determinada política migratoria [7], pero sin interrumpir el procesamiento en espera de comunicación con algún otro procesador. Este tipo de implementación tiene una doble ventaja para los sistemas computacionales heterogéneos:

a) por un lado elimina los tiempos muertos producidos por las barreras de sincronización, que pueden ser extremadamente perjudiciales cuando se trabaja con una red de computadoras cuyo tráfico no se puede controlar totalmente y con máquinas hetereogéneas cuyo balanceamiento de carga es sumamente difícil de optimizar [8],

b) por el otro, el procesamiento independiente de las subpoblaciones ha demostrado tener importantes ventajas sobre la simple paralelización del procesamiento de una población global, reportándose inclusive aceleraciones (*speedup)* superlineales [1].

No es de extrañar entonces la gran cantidad de trabajos dedicados a la implementación de Algoritmos Genéticos Paralelos en un contexto asíncrono [9].

El presente trabajo se organiza de la siguiente forma: la siguiente sección describe el algoritmo implementado. La sección 3 presenta la plataforma computacional utilizada y algunos resultados significativos utilizando funciones de prueba de De Jong [3] y finalmente, la sección 4 enfatiza las conclusiones del trabajo.

2. Algoritmos Genéticos Asíncronos Combinados

La presente sección postula la utilización de diversos procesadores, posiblemente hetereogéneos, procesando subpoblaciones de tamaños proporcionales al desempeño relativo de los procesadores (y por consiguiente de tamaños diferentes). Cada procesador aplicará a su población los tradicionales operadores genéticos de Selección, Cruzamiento y Mutación [4] así como un operador de optimización local similar al propuesto en [1]. Los individuos de cada subpoblación podrán migrar de un procesador a otro conforme a una política migratoria definida por los siguientes parámetros:

- El *intervalo de migración* : que establece cada cuantas generaciones se realizará la migración de una cierta cantidad de individuos desde una subpoblación a la otra.

- La *tasa de migración* : que indica cuantos individuos han de comunicarse a la otra subpoblación cuando se cumple el intervalo de migración.

- El *criterio de selección de los migrantes* : que determina la política que se aplicará para la selección de los individuos que han de migrar. Por ejemplo, se puede establecer que los migrantes sean elegidos al azar. Sin embargo, es posible ayudar a las otras subpoblaciones seleccionando los individuos mejor adaptados (es decir, de mejor *fitness*). Esta última opción parece más natural desde el punto de vista biológico, pues en la naturaleza, las migraciones involucran grandes esfuerzos a los migrantes, por lo que serán los mejores los que terminen la travesía.

- La *topología de comunicación* : que define el sentido de las migraciones, es decir, el o los procesadores destino de la migración. Generalmente, la topología de comunicación mapea directamente a la arquitectura computacional utilizada, sea esta un hipercubo [10], toroide [1], anillo o malla totalmente interligada [11].

Una variante de la presente propuesta respecto a otras anteriores, es la utilización de optimizadores locales diferentes según sean las características del procesador que lo ejecuta. Para los resultados experimentales de la sección 4 se utilizaron algoritmos numéricos de optimización local, disponibles en una biblioteca, de entre los que se destacan:

m1 : Método del Gradiente [4].
m2 : Algoritmo de Ascenso Suficiente [5].
m3 : Algoritmo Modificado de Hookes y Jeeves [5].
m4 : Algoritmo de Métrica Variable [4].

De esta forma, un operador de optimización local trabaja de la siguiente forma:

1. escoge un individuo en forma aleatoria o aplicando el operador de selección, haciendo que un individuo más adaptado tenga mayor probabilidad de ser elegido;
2. aplica el algoritmo numérico asignado a ese procesador, al individuo escogido, optimizándolo hasta un posible máximo (generalmente local);

3. se reemplaza al individuo escogido por su versión optimizada.

La versión secuencial del pseudocódigo implementado utilizando el operador de optimización local es presentado a continuación. Puede notarse que el mismo solo se aplica después de la iteración *numMax* para evitar la optimización local de individuos que aun no tienen información respecto a posibles óptimos globales.

```
Inicializa_la_Población;
Estadística_de_la_Población;
t ← 0;
DO WHILE NOT ( Criterio_de_Parada )
        t ← t + 1;
        Reproducción( Selección, Cruzamiento y Mutación );
        IF ( t ≥ numMax ) THEN Optimizador_Local;
        Estadística_de_la_Población;
END DO
```

Pseudocódigo 1: Algoritmo Genético Combinado

La utilización del optimizador local puede ser perjudicial cuando aplicado secuenciálmente, pues puede disminuir rápidamente la variedad genética, acelerando la convergencia del algoritmo a valores sub-óptimos. Sin embargo, cuando es aplicado a subpoblaciones en un contexto paralelo, la existencia de migrantes minimiza el peligro de pérdida genética, sin perder la capacidad de acelerar convenientemente todo el proceso. En consecuencia, los optimizadores locales muestran todo su potencial en un contexto paralelo, que puede ser implementado utilizando un proceso *Master* que se encarga de administrar todo el sistema (ver Pseudocódigo 2) incluyendo el lanzamiento de diversos procesos *Esclavos* en cada uno de los procesadores disponibles en el sistema computacional, procesos estos que son los que realizan los cálculos propiamente dichos, conforme se ilustra en el Pseudocódigo 3.

```
Leer_Datos;
Levantar_Procesos_Escalvos;
Enviar_Parámetros_a_cada_Esclavo;
fin ← FALSE;
DO WHILE NOT ( fin )
        Recibir_Mensaje_Terminación_de_Esclavos;
        IF ( Todas las máquinas terminaron ) THEN fin ← TRUE;
END DO
Enviar_Mensaje_de_Fin_a_Esclavos;
Eliminar_Procesos_Esclavos;
```

Pseudocódigo 2: Estructura del Proceso Master

```
Recibir_Parámetros;
Iniciar_Población;
Estadística_de_la_Población;
Selección_de_Individuos;
t ← 0;
DO WHILE ( TRUE )
        t ← t + 1;
        Reproducción ( Selección, Cruzamiento y Mutación );
        Escoger_Migrantes;
        Enviar_Migrantes ( a otros procesos esclavos );
        Recibir_Migrantes ( de otros procesos esclavos );
        Seleccionar_Individuos( Manteniendo tamaño de la Población );
        Estadística_ de_la_Población;
        IF ( t ≥ numMax ) THEN  Optimizador_Local;
        IF ( Criterio de fin ) THEN  Mensajes_al_Master;
END DO
```

Pseudocódigo 3: Estructura de cada Proceso Esclavo

Note que en una implementación asíncrona el número de migrantes recibidos puede variar de una generación a otra, por lo que es necesario aplicar el operador de selección para mantener constante el tamaño de la subpoblación en cada procesador.

Conforme fuera mencionado, la implementación del operador de optimización local puede variar de un procesador a otro, haciendo cálculos más precisos en las computadoras de mayor desempeño y hasta eliminándolo completamente de las computadoras más lentas, sirviendo este mecanismo como una forma de mejorar el balanceamiento de carga sin tener que utilizar subpoblaciones de tamaños muy dispares. De esta forma, es posible aprovechar toda la capacidad computacional de las redes de computadoras existentes, aun en presencia de computadoras con substanciales diferencias de desempeño.

Cabe mencionar que en sistemas distribuidos con suficientes recursos de máquinas, es posible crear un proceso especializado en optimizar suficientemente a los buenos candidatos, sin implementar un algoritmo genético. Así, cuando un procesador encuentra un buen individuo, por ejemplo al uniformar su subpoblación en una dada generación, este puede ser enviado al referido proceso para su opmización posterior y eventual difusión a los demás procesadores del sistema.

Finalmente, se enfatiza el hecho de que cada procesador trabaja sin coordinación con los demás procesadores, posiblemente con tamaños diferentes de la subpoblación e implementaciones y criterios diferentes para los operadores genéticos, lo que contribuye a la diversidad genética y por consiguiente, a la obtención de mejores soluciones que las obtenidas con implementaciones totalmente uniformes. Sin embargo, estas implementaciones asíncronas pierden el concepto de generación pues cada procesador

puede realizar un número diferente de iteraciones que a su vez no están sincronizadas entre sí.

3. Resultados Experimentales

Los resultados experimentales presentados a continuación se basan en la optimización de las siguientes funciones de De Jong [5]:

$$\textbf{F2 :}\ f(x_i) = 100 * (x_1^2 - x_2)^2 + (1 - x_1)^2\ ,\ \forall\ -2.048 \le x_i \le 2.048$$

$$\textbf{F5 :}\ f(x_i) = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}\ ,\ \forall\ -65.536 \le x_i \le 65.536$$

Para la obtención de estos datos experimentales se utilizaron las siguientes *workstations* interconectadas a una red tipo *Ethernet* (10Base T) con un centenar de computadoras personales:

1. Workstation DEC 3000 modelo 300, con procesador alpha de 150 MHz, 32 MB en RAM y sistema operativo OSF/1 v. 2.0.

2. Workstation SUN SPARC Station 5, con procesador Sparc de 66 MHz, 32 MB en RAM y sistema operativo Solaris 5.3.

Las diversas implementaciones realizadas y descriptas en la Tabla 1 fueron codificadas en PVM (Parallel Virtual Machine) en la versión extendida de ANSI C . Estas implementaciones fueron luego combinadas con los distintos métodos numéricos de optimización, descriptos en la sección anterior, los que fueron utilizados como operadores del Algoritmo Genético.

Tabla 1 : Algoritmos Genéticos implementados

Implementación	Comunicación	Algoritmo	Símbolo
Secuencial	-	-	SS
Secuencial	-	Combinado	SCm_i
Paralela	Síncrona	-	PS
Paralela	Síncrona	Combinado	PSm_im_k
Paralela	Asíncrona	-	PA
Paralela	Asíncrona	Combinado	PAm_im_k

donde m_i representa el optimizador local utilizado en la *workstation* SUN y m_k el utilizado en la DEC en caso de las implementaciones combinadas, conforme a la descripción de índices de la sección anterior.

En las mediciones de tiempo y desempeño, se consideró que una dada ejecución estaba terminada si se cumplía uno de los siguientes criterios de parada:

1. Varianza de la población menor que un ε dado;
2. Numero máximo de iteraciones;
3. Tiempo máximo de ejecución.

El desempeño de un algoritmo se midió corriendo el mismo algoritmo N veces (típicamente, N = 50) y calculando el valor medio de los tiempos empleados y los valores obtenidos para las funciones objetivos que se estaban maximizando.

La Tabla 2 muestra algunos de los resultados experimentales obtenidos con una población de 100 individuos de los cuales 40% son asignados a la SUN y el resto a la DEC 3000. Como puede observarse, las implementaciones paralelas asíncronas y combinadas (con optimizadores locales) son claramente superiores a las implementaciones paralelas más tradicionales, tanto en la calidad de la solución encontrada (desempeño) como en el tiempo de procesamiento (ver por ejemplo **PAm_1m_3** con la función F2, **PAm_1m_1** y **PAm_1m_3** con la función F5 de De Jong).

Tabla 2 : Resultados Experimentales

Función	Implementación	*Desempeño*	Tiempo(seg.)
F2	SS	3838.72119	3.664911
	SCm_1	3853.59815	2.980207
	PA	3887.17944	1.416100
	PAm_1m_1	3871.78833	3.370761
	PAm_1m_3	3888.93872	3.986055
	PS	3820.40942	3.145111
	PSm_1m_1	3777.49829	3.634333
	PSm_1m_4	3672.52002	3.509075
	PSm_1m_3	3786.31470	3.756697
F5	SS	3.790546	3.94999
	SCm_1	3.817955	1.800819
	PA	3.815276	3.276627
	PAm_1m_1	3.817958	1.448785
	PAm_1m_4	3.817958	2.792711
	PAm_1m_3	3.817958	2.326697
	PS	3.791299	3.834485
	PSm_1m_1	3.817954	3.480729
	PSm_1m_4	3.817899	3.988115
	PSm_1m_3	3.054766	3.655223

La Tabla 3 muestra el efecto de desbalancear las cargas, variando la cantidad de individuos a ser asignados a cada procesador. Como es de esperar, los tiempos de procesamiento dependen del balanceamiento de carga pero en general, la utilización de algoritmos combinados diferentes en cada procesador resulta casi siempre beneficioso, sobre todo cuando las cargas están bien balanceadas. Como ejemplo, se puede observar que los mejores desempeños del algoritmo se obtuvieron al combinar el algoritmo genético paralelo con el método del Gradiente ($\mathbf{m_1}$) en un procesador y el de Métrica Variable en el otro ($\mathbf{m_4}$).

Tabla 3: Resultados para diferentes particiones.

Función De Jong	***Desempeño Óptimo***	***Desempeño Experimental***	**Tiempo (segundos)**	***Subpoblación SUN***	***método SUN***	***método DEC***
F5	3.817958	3.563560	3.517408	**20**	$\mathbf{m_1}$	$\mathbf{m_1}$
	3.817958	3.563581	2.068437	**30**	$\mathbf{m_1}$	$\mathbf{m_1}$
	3.817958	3.817958	1.448785	**40**	$\mathbf{m_1}$	$\mathbf{m_1}$
	3.817958	3.817958	2.236543	**50**	$\mathbf{m_1}$	$\mathbf{m_1}$
	3.817958	3.817958	1.268846	**20**	$\mathbf{m_4}$	$\mathbf{m_4}$
	3.817958	3.817958	3.086533	**30**	$\mathbf{m_4}$	$\mathbf{m_4}$
	3.817958	3.817958	2.556160	**40**	$\mathbf{m_4}$	$\mathbf{m_4}$
	3.817958	3.817958	2.981364	**50**	$\mathbf{m_4}$	$\mathbf{m_4}$
	3.817958	3.817958	1.574217	**50**	$\mathbf{m_1}$	$\mathbf{m_4}$
	3.817958	3.817958	2.054251	**50**	$\mathbf{m_4}$	$\mathbf{m_1}$
	3.817958	3.817958	1.040199	**20**	$\mathbf{m_1}$	$\mathbf{m_4}$
	3.817958	3.817958	1.085705	**20**	$\mathbf{m_4}$	$\mathbf{m_1}$
	3.817958	3.817958	2.965580	**40**	$\mathbf{m_1}$	$\mathbf{m_4}$
	3.817958	3.817958	3.391128	**40**	$\mathbf{m_4}$	$\mathbf{m_1}$

4. Conclusiones

Las implementaciones paralelas mejoran los tiempos de computación del Algoritmo Genético en una relación que puede llegar a ser superlineal, llegando inclusive a mejores soluciones que los algoritmos secuenciales al trabajar sobre subpoblaciones independientes [5]. Como una forma de aprovechar estas características en un sistema computacional hetereogéneo y mejorar los tiempos de ejecución, se postuló combinar estos Algoritmo Genético Paralelos con diversos métodos numéricos de optimización en un ambiente computacional asíncrono, lográndose mejores resultados sin una pérdida importante de la información genética y la aleatoriedad requerida.

Los resultados experimentales de la sección anterior demuestran:

a. la superioridad de las implementaciones asíncronas sobre las más tradicionales implementaciones secuenciales y síncronas;

b. la utilidad de combinar otros métodos de optimización con los algoritmos genéticos, inclusive en una implementación totalmente secuencial;

c. la ventaja de utilizar diferentes métodos en cada uno de los procesadores de un sistema computacional hetereogéneo, aprovechando esta hetereogeneidad para privilegiar la diversidad de la población y la aleatoriedad;

d. la posibilidad de mejorar los tiempos de ejecución y desempeño general del algoritmo combinado los efectos de paralelizar los AG, implementarlo en forma asíncrona y combinarlo con otros algoritmos, en una implementación totalmente hetereogénea.

En conclusión, la reconocida potencialidad de los algoritmos genéticos paralelos puede ser hoy aprovechada no solo en los centros que cuentan con importantes computadoras paralelas, sino por cualquier institución con acceso a una red de computadoras, por más heterogéneas que puedan ser sus máquinas, pues implementaciones como la descripta permiten el uso eficiente de los recursos computacionales disponibles.

REFERENCIAS

[1] H. Mühlenbein, M. Schomisch, J. Born, "The Parallel Genetic Algorithm as Function Optimizer", proceeding of the Fourth International Conference on Genetic Algorithm, 1991, pp. 271 - 278.

[2] H. Mühlenbein, M. Gorges-Schleuter, O. Krämer, "Evolution Algorithm in Combinatorial Optimization", *Parallel Computing*, Vol. 7, 1988, pp. 65 - 85.

[3] A. De Jong, " An Analysis of the Behavior of a Class of Genetic Adaptive Systems ", Doctoral Thesis, University of Michigan, 1975.

[4] D.E. Goldberg, " Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Publishing Company, Inc. 1989.

[5] Erick Cantú-Paz, "A Summary of Research on Parallel Genetic Algorithms", Technical Report Nº 95007, julio 1995.

[6] S. N. Talukdar, S. S. Pyo, T. C. Giras, " Asínchronous Procedures for Parallel Processing ", Electric Power Research Institute Final Report. Carnegie - Mellon University. Pittsburgh - Pensilvania, 1983.

[7] M. Mejía, E. Cantú, “DGENESIS, Software para la Ejecución de Algoritmos Genéticos Distribuidos”, XX Conferencia Latinoamericana de Informática - CLEI PANEL. México 1994.

[8] B. Barán, E. Kaszkurewicz, A. Bhaya, “Parallel Asínchronous Team Algorithms: Convergence and Performance Analysis ”, *IEEE Transactions on Parallel and Distributed Systems*, Vol.7, No. 7, 1983.

[9] B. Barán, N. Cáceres, E. Chaparro, “Reducción del Tiempo de Búsqueda utilizando una Combinación de Algoritmos Genéticos y Métodos Numéricos”, III Encuentro Chileno de Computación, Arica - Chile, 1995.

[10] R.Tanese, “Parallel Genetic Algorithm for a hypercube”, In John Grefenstette, editor. Proceeding of the Second International Conference on Genetic Algorithms. Lawrence Erlbaum Associates, Publishers, 1987.

[11] E. Cantú - Paz, M. Mejía - Olvera, “Experimental results in Distribuited Genetic Algorithms”, International Symposium on Applied Corporate Computing, pp. 99 - 108, Monterrey - México, 1994.