

# ALGORITMOS ASÍNCRONOS COMBINADOS EN UN AMBIENTE HETEROGÉNEO DE RED

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

## Resumen

*Se han estudiado diversas implementaciones de los algoritmos genéticos paralelos (AGP), generalmente para computadores paralelos que no son de fácil acceso en países de escasos recursos. En su lugar, se propone el aprovechamiento de las redes de computadoras existentes, trabajando en un ambiente típicamente asíncrono. Para esto, se propone combinar las tradicionales implementaciones de los AGP con diferentes algoritmos de optimización, en cada una de las máquinas de una red heterogénea formando un "Equipo de Algoritmos" (Team Algorithm). Así, se propone utilizar optimizadores diferenciados (como el algoritmo del gradiente, del ascenso suficiente, de métrica variable y otros), según sea el desempeño relativo de cada procesador. Resultados experimentales demuestran la validez de la propuesta por su mejor adaptación a las características de heterogeneidad de los procesadores y aleatoriedad de las comunicaciones de la red.*

## Palabras Claves

Algoritmos Genéticos Paralelos (AGP), Team Algorithm, Asincronismo, Red Heterogénea de Computadoras.

## 1. Introducción

Los Algoritmos Genéticos (AG) han demostrado ser una buena herramienta en la búsqueda de óptimos globales, debido a su relativa sencillez de implementación y a la robustez de su aplicación [1,2]; 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

obteniendo resultados parciales que serán transmitidos a los otros procesadores, colaborando todos para llegar a la solución global del problema [3,4].

Desde que se presentaron las primeras ideas acerca de la paralelización de los Algoritmos Genéticos hasta nuestros días, se han realizado numerosas implementaciones diferentes [4], destacándose las implementaciones asíncronas [5, 6] donde cada subpoblación es procesada 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, extremadamente perjudiciales cuando se trabaja con una red de computadoras cuyo tráfico no se puede controlar totalmente y con máquinas heterogéneas 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 (*speedups*) superlineales [5].

No es de extrañar entonces la gran cantidad de trabajos dedicados a la implementación de Algoritmos Genéticos Paralelos en un contexto asíncrono [9]. En este sentido, se destacan los trabajos que se vienen realizando en la Universidad de Carnegie Mellon [6, 10] combinando diversos algoritmos en un ambiente asíncrono, técnica conocida como *A-Teams (Asynchronous Teams)*.

El presente trabajo se inspira en la técnica de los A-Teams para proponer la utilización de diversos métodos de optimización, de forma similar a la utilización del método de optimización local propuesto en [5]. Con este fin, 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 [11] 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 heterogéneos, 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 [1] así como un operador de optimización, de forma similar a lo propuesto en [5], con la variante de que cada procesador podrá utilizar un método diferente de optimización, conforme a la técnica de los A-

Teams [10]. 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 la arquitectura computacional utilizada, sea ésta un hipercubo [12], toroide [4], anillo o malla totalmente interligada [13].

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

- m1:** Método del Gradiente [14].
- m2:** Algoritmo de Ascenso Suficiente [14].
- m3:** Algoritmo Modificado de Hookes y Jeeves [14].
- m4:** Algoritmo de Métrica Variable [15].

De esta forma, el AG utilizará un operador de *optimización local*, que 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, optimizando al individuo escogido, posiblemente hasta un máximo local;
3. reemplaza al individuo escogido por su versión optimizada.

A continuación se presenta el pseudocódigo de la versión secuencial utilizando el operador de optimización local. 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;  
 $t \leftarrow t + 1;$   
DO WHILE NOT (Criterio_de_Parada)  
   $t \leftarrow t + 1;$   
  Reproducción(Selección, Cruzamiento y Mutación);  
  IF ( $t \geq numMax$ ) THEN Optimizador_Local;  
  Estadística_de_la_Población;  
END DO
```

### Pseudocódigo 1: Algoritmo Genético Combinado

La utilización del *Optimizador\_Local* suele ser perjudicial cuando es aplicado secuencialmente, 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 éstos que son los que realizan los cálculos propiamente dichos, conforme se ilustra en el Pseudocódigo 3.

```
Leer_Datos;  
Levantar_Procesos_Esclavos;  
Enviar_Parámetros_a_cada_Esclavo;  
 $fin \leftarrow FALSE;$   
DO WHILE NOT ( $fin$ )  
  Recibir_Mensaje_Terminación_de_Esclavos;  
  IF (Todas las máquinas terminaron) THEN  $fin \leftarrow TRUE;$   
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;
 $t \leftarrow 0;$ 
DO WHILE (TRUE)
   $t \leftarrow t + 1;$ 
  Reproducción (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 ( $t \geq numMax$ ) THEN Optimizador_Local;
  IF (Criterio de fin) THEN Mensajes_al_Master;
END DO

```

**Pseudocódigo 3: Estructura de cada Proceso Esclavo.**

Se hace notar 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 *Optimizador\_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 optimización 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 [11]:

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

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

Para la obtención de estos datos experimentales se utilizaron las siguientes *workstations* interconectadas a una red *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 descritas 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, descritos en la sección anterior, los que a su vez fueron utilizados como operadores del Algoritmo Genético.

**Tabla 1 : Algoritmos Genéticos implementados**

Implementación	Comunicación	Algoritmo	Símbolo
Secuencial	-	-	SS
Secuencial	-	Combinado	SC <sub>m<sub>i</sub></sub> *
Paralela	Síncrona	-	PS
Paralela	Síncrona	Combinado	PS <sub>m<sub>i</sub>m<sub>k</sub></sub> *
Paralela	Asíncrona	-	PA
Paralela	Asíncrona	Combinado	PAm <sub>i</sub> m <sub>k</sub> *

\* donde m<sub>i</sub> representa el optimizador local utilizado en la *workstation* SUN y m<sub>k</sub> 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  $\epsilon$  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% fueron 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<sub>1</sub>m<sub>3</sub> con la función F2, PAm<sub>1</sub>m<sub>1</sub> y PAm<sub>1</sub>m<sub>3</sub> con la función F5 de De Jong).

**Tabla 2 : Resultados Experimentales**

Función	Implementación	Desempeño	Tiempo(seg)
F2	SS	3838.72119	3.664911
	SCm <sub>1</sub>	3853.59815	2.980207
	PA	3887.17944	1.416100
	PAm <sub>1</sub> m <sub>1</sub>	3871.78833	3.370761
	PAm <sub>1</sub> m <sub>3</sub>	3888.93872	3.986055
	PS	3820.40942	3.145111
	PSm <sub>1</sub> m <sub>1</sub>	3777.49829	3.634333
	PSm <sub>1</sub> m <sub>4</sub>	3672.52002	3.509075
	PSm <sub>1</sub> m <sub>3</sub>	3786.31470	3.756697
F5	SS	3.790546	3.94999
	SCm <sub>1</sub>	3.817955	1.800819
	PA	3.815276	3.276627
	PAm <sub>1</sub> m <sub>1</sub>	3.817958	1.448785
	PAm <sub>1</sub> m <sub>4</sub>	3.817958	2.792711
	PAm <sub>1</sub> m <sub>3</sub>	3.817958	2.326697
	PS	3.791299	3.834485
	PSm <sub>1</sub> m <sub>1</sub>	3.817954	3.480729
	PSm <sub>1</sub> m <sub>4</sub>	3.817899	3.988115
	PSm <sub>1</sub> m <sub>3</sub>	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 ( $m_1$ ) en un procesador y el de Métrica Variable en el otro ( $m_4$ ).

**Tabla 3: Resultados para diferentes particiones.**

<b>Función De Jong</b>	<i>Desempeño Óptimo</i>	<i>Desempeño Experimental</i>	<b>Tiempo (seg)</b>	<i>Subpoblación SUN</i>	<i>Método SUN</i>	<i>Método DEC</i>
<b>F5</b>	3.817958	3.563560	3.517408	<b>20</b>	$m_1$	$m_1$
	3.817958	3.563581	2.068437	<b>30</b>	$m_1$	$m_1$
	3.817958	3.817958	1.448785	<b>40</b>	$m_1$	$m_1$
	3.817958	3.817958	2.236543	<b>50</b>	$m_1$	$m_1$
	3.817958	3.817958	1.268846	<b>20</b>	$m_4$	$m_4$
	3.817958	3.817958	3.086533	<b>30</b>	$m_4$	$m_4$
	3.817958	3.817958	2.556160	<b>40</b>	$m_4$	$m_4$
	3.817958	3.817958	2.981364	<b>50</b>	$m_4$	$m_4$
	3.817958	3.817958	1.574217	<b>50</b>	$m_1$	$m_4$
	3.817958	3.817958	2.054251	<b>50</b>	$m_4$	$m_1$
	3.817958	3.817958	1.040199	<b>20</b>	$m_1$	$m_4$
	3.817958	3.817958	1.085705	<b>20</b>	$m_4$	$m_1$
	3.817958	3.817958	2.965580	<b>40</b>	$m_1$	$m_4$
	3.817958	3.817958	3.391128	<b>40</b>	$m_4$	$m_1$

#### 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 [4, 5]. Como una forma de aprovechar estas características en un sistema computacional heterogéneo y mejorar los tiempos de ejecución, se postuló combinar estos Algoritmos Genéticos 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 heterogéneo, aprovechando esta heterogeneidad para privilegiar la diversidad de la población y la aleatoriedad;
- d. la posibilidad de mejorar los tiempos de ejecución y desempeño general del algoritmo combinando los efectos de paralelizar los AG, implementarlo en forma asíncrona y combinarlo con otros algoritmos, en una implementación totalmente heterogénea.

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 descrita permiten el uso eficiente de los recursos computacionales disponibles.

## REFERENCIAS

- [1] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company Inc, 1989.
- [2] H. Mühlenbein, "Major Publications - Technical Reports and Publications". Disponible en: [http://set.gmd.de/AS/ga/publi\\_new.html](http://set.gmd.de/AS/ga/publi_new.html).
- [3] H. Mühlenbein, M. Gorges-Schleuter, O. Krämer, "Evolution Algorithm in Combinatorial Optimization", *Parallel Computing*, Vol. 7, 1988, pp. 65 - 85.
- [4] Erick Cantú-Paz, "A Summary of Research on Parallel Genetic Algorithms", *Technical Report N° 95007*, Julio 1995.
- [5] H. Mühlenbein, M. Schomisch, J. Born, "The Parallel Genetic Algorithm as Function Optimizer", *Proceedings of the Fourth International Conference on Genetic Algorithm*, 1991, pp. 271 - 278.

- [6] S. N. Talukdar, S. S. Pyo, T. C. Giras, "Asynchronous 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 Asynchronous 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] S. Sachdev, "A-Teams Project Home Page." Disponible en: [http://www.cs.cmu.edu/afs/cs/project/edrc-22/project/ateams/www/home\\_page.html](http://www.cs.cmu.edu/afs/cs/project/edrc-22/project/ateams/www/home_page.html).
- [11] A. De Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems", Doctoral Thesis, University of Michigan, 1975.
- [12] 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.
- [13] E. Cantú - Paz, M. Mejía - Olvera, "Experimental results in Distributed Genetic Algorithms", *International Symposium on Applied Corporate Computing*, pp. 99 - 108, Monterrey - México, 1994.
- [14] O. Armitano, J. Edelman, U. García Palomares, *Programación No Lineal*, Editorial Limusa. 1985.
- [15] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press. 1991.