¿PUEDE UN ALGORITMO EVOLUTIVO PARALELO PROPORCIONAR GANANCIA SUPERLINEAL?

Enrique Alba

Dpto. Lenguajes y Ciencias de la Computación
Universidad de Málaga
Campus de Teatinos, 29071 *Málaga*
eat@lcc.uma.es

Palabras clave : Algoritmo Evolutivo Paralelo, Ganancia de Velocidad, Algoritmos Genéticos Descentralizados
Keywords : Parallel Evolutionary Algorithm, Speedup, Decentralized Genetic Algorithms

RESUMEN. *En este trabajo revisaremos inicialmente los resultados más importantes relativos a la obtención de ganancias (speedup) superlineales en algoritmos evolutivos paralelos. A la luz de estos resultados evaluaremos las posiciones críticas existentes actualmente. Comprobaremos cómo, a pesar de que a veces se han utilizado medidas no ortodoxas que han concluido en ganancias superlineales, los algoritmos evolutivos paralelos son capaces de proporcionar ganancias superlineales también cuando se utilizan medidas ortodoxas. Realizaremos una categorización de los tipos de medida de ganancia y demostraremos que es posible obtener ganancia superlineal. La explicación se encuentra en el cambio que se produce cuando un algoritmo evolutivo se ejecuta como un modelo descentralizado y también en su ejecución físicamente paralela. Estudiaremos distintos problemas atendiendo a distintas medidas de ganancia con la intención de proporcionar una metodología de trabajo en este campo.*

ABSTRACT. *In this work we revisit existing results on measuring speedup in parallel evolutionary algorithms. We will analyze some inconsistences in this measure and discuss conflictive claims. We will propose a categorization on the methods for measuring speedup and we will study the performance of parallel evolutionary algorithms on different problems in order to show the importance of the way in which the speedup is being measured. We conclude that superlinear speedup is possible in this field, although in a more reasonable way than other speedup criteria have suggested in the past. The reasons for a superlinear speedup can be found in the decentralization of the population when parallelized and in the physical parallelism provided by the underlying parallel hardware.*

1. INTRODUCCIÓN

Desde el nacimiento de la computación evolutiva las implementaciones de algoritmos sobre *hardware* paralelo han sido muy comunes (Alba & Troya 99a). Debido a que cualquiera de los paradigmas de algoritmos evolutivos (AAEE) manipula un conjunto de soluciones conocido como *población* de individuos, es lógico pensar que trabajar por separado y en paralelo en subconjuntos de esta población global puede reportar beneficios importantes (Hart et al. 97).

Sin embargo, conseguir un algoritmo más rápido (eficiencia), aunque importante, no es la única razón para utilizar un AE paralelo (AEP); de hecho, existen numerosas razones para ello:

- Disponer de más memoria para abordar problemas más complejos (potencia).
- Crear nichos a los que evolucionar en paralelo (versatilidad).
- Mejorar la diversidad presente en la población (problemas no-estacionarios).
- Reducir la probabilidad de caer en un óptimo local (eficacia).
- Cooperar con otra técnica de búsqueda en paralelo (aplicabilidad).

En la lista anterior hemos añadido al final de cada línea entre paréntesis la propiedad algorítmica que se mejora con cada una de las ventajas enumeradas. Aunque no es completa, basta por sí misma para justificar el interés en el trabajo con algoritmos evolutivos paralelos.

Cada uno de los puntos enunciados posee su propia problemática, pero en general el campo de los AAEEPP se enfrenta a varios problemas comunes. En primer lugar existe una gran diversidad de implementaciones paralelas, muchas de ellas con igual comportamiento que un AG secuencial y otras no, sin que hasta muy recientemente se estén desarrollando enfoques únicos comunes y metodologías de diseño y estudio (Alba & Troya 99b) (Sprave 99).

Un AE tradicional utiliza una única población sobre la que aplica operadores de variación y selección de los más aptos (Algoritmo 1) (Bäck 96). A este tipo de población se le denomina en *panmixia*. Los algoritmos en panmixia pueden ser fácilmente paralelizados por ejemplo con un esquema maestro/esclavo donde las evaluaciones de los individuos las realizan en paralelo los esclavos. Al esquema mencionado se le conoce como *paralelización global* y se comporta básicamente como un algoritmo en panmixia pero más rápido. Sin embargo, este modelo ha recibido poca atención históricamente (Cantú-Paz & Goldberg 97). La razón es que el modelo de paralelización global produce a duras penas una ganancia de velocidad (*speedup*) cercana a la lineal, y en general proporciona ganancias de velocidad realmente pequeñas.

ALGORITMO 1. ALGORITMO EVOLUTIVO SECUENCIAL

```
t := 0;
inicializar y evaluar [P(t)];
mientras no terminar hacer
        P'(t) := variación [P(t)];
        evaluar [P'(t)];
        P(t+1) := seleccionar [P'(t)];
        t := t +1;
fin mientras
```

Respecto al resto de AAEEPP, el hecho de que se utilicen poblaciones separadas con intercambios aislados de individuos (Tanese 89) o se utilice un esquema celular donde se disponen vecindarios solapados de individuos (Spiessens & Manderick 91) ha llevado tradicionalmente a distinguir entre algoritmos de *grano grueso* y algoritmos de *grano fino*, atendiendo a que la relación computación/comunicación sea elevada o no. Sin embargo, una apreciación importante es que realmente ambos son dos tipos de AAEE *descentralizados*.

La distinción grano grueso/fino es muy popular pero engañosa en el caso de los AAEEPP porque mezcla, ligándolos, el modelo y su implementación, quienes en realidad son dos cosas distintas. Podemos decir que se ha abusado de etiquetas tradicionales para nombrar conceptos nuevos. Algo similar ha ocurrido con la definición de ganancia en velocidad o *speedup*.

La definición de ganancia tradicional relaciona el tiempo de ejecución de un algoritmo secuencial con su implementación paralela utilizando n_{proc} procesadores. Esta definición tradicional tiene varias connotaciones que haremos en la sección siguiente, pero básicamente establece que no pueden obtenerse valores superlineales porque lo único que se hace es implementar el algoritmo secuencial en paralelo, con posible sobrecarga por las comunicaciones.

Sin embargo, los AAEE son estocásticos y su trabajo paralelo no es equivalente a su trabajo en secuencial. Al dar una estructura de cualquier tipo a la población descentralizamos el esquema básico de selección del algoritmo, perturbando así el tipo de búsqueda realizada. Esto lleva aparejado un cambio potencial en la solución devuelta por el algoritmo y también en el tiempo y esfuerzo necesario para ello (Gordon & Whitley 93).

Como consecuencia, muchos autores han informado de la aparición de ganancias superlineales en AAEEPP (Shonkwiler 93) (Belding 95). El problema está en determinar cómo se mide la ganancia para que claramente los resultados tengan sentido y para que no se utilice una definición lejana a la tradicional, pues de otra manera deberíamos asignarle otro nombre.

Nuestro objetivo en este artículo es revisar varias formas de medir ganancias en AAEEPP, proponer medidas que sean más justas, y estudiar un AEP atendiendo a ellas para comprobar la cuantía de la ganancia en múltiples problemas. Finalmente, plantearemos diversos problemas abiertos en relación a las situaciones en las que se obtiene ganancia superlineal.

El presente trabajo está estructurado de la siguiente forma. En la Sección 2 discutimos en detalle la definición de ganancia. La Sección 3 estudia la posibilidad teórica de que aparezcan ganancias superlineales en AAEEPP. En la Sección 4 describimos los problemas y algoritmos usados. Las secciones 5, 6 y 7 presentan nuestros resultados atendiendo a las consideraciones hechas, y para terminar, en la Sección 8 resumimos algunas conclusiones y líneas futuras.

2. TIPOS DE GANANCIAS MEDIDAS EN COMPUTACIÓN EVOLUTIVA

En esta sección nuestra intención es resumir y clasificar los trabajos existentes a través de la distinción de distintas maneras de medir la ganancia. La definición tradicional de ganancia (Akl 92) relaciona el (peor) tiempo de ejecución del algoritmo secuencial más veloz conocido con el (peor) tiempo de ejecución del algoritmo paralelo sobre n_{proc} procesadores:

$$\mathcal{S}(n_{proc}) = \frac{T_1}{T_{n_{proc}}} \tag{1}$$

Con esta definición se habla de:

$$\begin{array}{lcl} \text{ganancia sublineal} & \Leftrightarrow & \mathcal{S}(n_{proc}) < n_{proc} \\ \text{ganancia lineal} & \Leftrightarrow & \mathcal{S}(n_{proc}) = n_{proc} \\ \text{ganancia superlineal} & \Leftrightarrow & \mathcal{S}(n_{proc}) > n_{proc} \end{array} \tag{2}$$

Esta primera definición es bastante fuerte y restrictiva porque asume que se conoce el mejor algoritmo para resolver un problema, y por eso la denominaremos *ganancia fuerte* (Tabla 1). En general, este tipo de ganancia sería muy difícil de evaluar para un AEP porque en numerosas ocasiones dicho AEP es la única alternativa para solucionar el problema. Asimismo, esta definición exige de cualquier investigador conocer el algoritmo más rápido para resolver cualquiera de los problemas considerados, lo cual en muchas ocasiones no es realista. Al resto de las medidas las denominaremos *débiles*, porque asumimos que es posible que existan algoritmos más rápidos, pero nos interesa medir la ganancia para evaluar un modelo de AEP concreto.

La primera modificación que debemos introducir en la medida de la Ecuación 1 para aplicarla a un AE es utilizar tiempos medios y no absolutos en el peor caso, ya que una ejecución separada de un AE no es estadísticamente significativa. De esta manera la definición para la ganancia de un AEP relaciona los tiempos medios sobre uno y n_{proc} procesadores:

$$\mathcal{S}(n_{proc}) = \frac{\overline{T}_1}{\overline{T}_{n_{proc}}} \tag{3}$$

A continuación debemos discutir un punto importante: el criterio de parada que se está utilizando en el AE. En muchos trabajos se ha usado un número máximo de iteraciones globales predefinidas tanto en el caso secuencial como en el paralelo (Lin et al. 94) (Belding 95) (Calégari et al. 97) -tipo II.B en la Tabla 1-. Pero, ¿podemos comparar la ganancia de dos algoritmos que devuelven soluciones cuya calidad es distinta? De hecho, es esto lo que ocurre si a dos AAEE le imponemos un número predefinido de iteraciones. En nuestra opinión no es correcto medir una ganancia de esta forma porque la búsqueda es distinta. Los trabajos recientes de (Hart et al. 97) y (Cantú-Paz & Goldberg 97) así parecen también establecerlo.

Debemos entonces definir un criterio claro de terminación. El criterio obvio que parece más justo es detener a todos los algoritmos comparados cuando alcancen una solución de igual calidad (a ser posible un óptimo -tipo II.A-). Establecer un número predefinido de evaluaciones puede ser útil en algún tipo de estudio pero debe estar suficientemente justificado en su caso.

Una vez aclarado este punto debemos discutir ahora un tema clave. Puesto que los algoritmos evolutivos secuenciales han usado tradicionalmente una población única sin estructura, el algoritmo secuencial es distinto del algoritmo paralelo en el que se separa la población en trozos. Este tipo de ganancia débil en el que se compara la versión en panmixia de un AE con una versión descentralizada paralela (tipo II.A.1), por ejemplo usando 8 islas ligeramente acopladas, ha sido usado en numerosas ocasiones. Como veremos, la diferencia numérica entre ambos algoritmos induce una diferencia en tiempo que puede provocar ganancia superlineal. En nuestro ejemplo, el algoritmo en panmixia podría ser más de 8 veces más lento que el algoritmo con 8 islas secuenciales (Mühlenbein et al. 91) (Alba & Troya 99b).

Nuestra propuesta en este trabajo es clarificar las medidas para tener una comparación justa y significativa. Una comparativa así debería únicamente considerar el mismo algoritmo y cambiar el número de procesadores entre los que se ejecuta (versión *ortodoxa* II.A.2) ¿Significa esto que no pueden aparecer ganancias superlineales? Como veremos en breve, sí son posibles.

Tabla 1. *Gategorización de las medidas de ganancia.*

I.	GANANCIA FUERTE		
II.	GANANCIA DÉBIL		
	A.	GANANCIA CON PARADA SOLUCIÓN	
		1.	FRENTE A PANMIXIA
		2.	ORTODOXA
	B.	GANANCIA CON ESFUERZO PREDEFINIDO	

En resumen, hemos dividido la ganancia según su definición en varios tipos (Tabla 1). Aunque interesante, no trabajaremos en este artículo con el tipo I. Puesto que no nos parece correcto medir la ganancia atendiendo a un esfuerzo predefinido (II.B), nos centraremos sobre los dos tipos de ganancia débil con parada solución (tipos II.A.1 y II.A.2).

3. ...PERO, ¿ES POSIBLE EN TEORÍA UNA GANANCIA SUPERLINEAL EN UN AEP?

Numerosos autores, atendiendo a numerosos criterios y para numerosos algoritmos evolutivos paralelos, han informado de ganancias superlineales sobre *hardware* paralelo. Puede comprobarse interesantes revisiones en (Chipperfield & Fleming 96) y (Alba & Troya 99a).

Aparte de las consideraciones anteriores que nos han permitido ligar la medida usada en este campo y la medida tradicional, aún no hemos abordado un inquietante punto: ¿es posible teóricamente que un AE paralelo presente una ganancia superlineal? La respuesta afortunadamente es que sí, ya que la alternativa es la existencia de un error en el análisis.

Para comprender por qué y cómo es posible, debemos retomar el carácter estocástico de un algoritmo evolutivo. Es posible comprobar que, tanto en un algoritmo en panmixia como en otro descentralizado, paralelos o no, la probabilidad de encontrar un óptimo global que sea solución en la población inicial o tras un número cualquiera de pasos es, en general, no nula; quizás ínfima, pero no nula. Esto tiene como efecto que un esquema de paralelismo dado pueda localizar más rápido que otro la solución a un mismo problema, incluso demostrando cierta dependencia del número de procesadores que se esté usando.

Podemos resumir el resultado diciendo que "para cualquier unidad de tiempo T>0 la probabilidad de que un AE localice un óptimo global es no nula". Esta conclusión fue demostrada formalmente en (Shonkwiler 93) en donde se plantea incluso un modelo de ganancia para algoritmos evolutivos paralelos con la forma:

$$\mathcal{S}\left(n_{proc}\right) = n_{proc} \cdot s^{n_{proc}-1} \tag{4}$$

donde podemos comprobar la existencia de un factor de aceleración s que puede inducir ganancia superlineal siempre que s>1.

4. BANCO DE PRUEBAS Y ALGORITMOS ESTUDIADOS

En esta sección presentaremos los problemas que resolveremos para medir la ganancia de velocidad en un AEP, así como los detalles de dicho AEP. Con ello pretendemos localizar en una sección del trabajo los datos para entender y revisar en el futuro el artículo y sus resultados.

4.1. El Banco de Pruebas

El análisis que vamos a realizar contiene resultados sobre tres problemas: la esfera generalizada, la suma del subconjunto y el problema decepcionante masivamente multimodal.

La primera función es típica y se suele utilizar como caso base. Es una función continua simple de la que usaremos una instancia de n=16 variables codificadas en 32 bits cada una (SPH16-32). En nuestro caso el problema es de maximización (2^n óptimos) de la expresión:

$$f\left(x_i\big|_{i=1..n}\right)=\sum_{i=1}^{n} x_i^2 \qquad x_i \in [-5.12,5.12] \tag{5}$$

El segundo problema es la suma del subconjunto (SSS). Se trata de un problema NP-Completo consistente en encontrar, de entre un conjunto $W=\{\omega_1, \omega_2, ..., \omega_n\}$, el subconjunto $V \subseteq W$ de valores que sumados se acercan más, sin superar, a un cierto valor entero C. Usamos una instancia de tamaño n=128 generada según (Jelasity 97) para incrementar la dificultad.

La función objetivo calcula inicialmente el peso de una solución $\vec{x}=\{x_1, x_2, ..., x_n\}$ como $P(\vec{x})=\sum_{i=1}^{n} x_i \cdot \omega_i$ y el algoritmo usado debe maximizar la función

$$f(\vec{x})=a\cdot P(\vec{x})+(1-a)\cdot max[C-0.1\cdot P(\vec{x}),0] \tag{6}$$

donde a=1 cuando $\vec{x}$ es admisible, es decir, cuando $C-P(\vec{x})\geq 0$, y a=0 en otro caso. Las soluciones no admisibles se penalizan con el 10% de su valor (el resultado es siempre positivo).

DECEPCIÓN BIPOLAR (6 bits)	
#unos	**valor subfunción**
0	1.000000
1	0.000000
2	0.360384
3	0.640576
4	0.360384
5	0.000000
6	1.000000

Figura 1. *Contribución de cada subproblema* (*MMDP*).

El problema decepcionante masivamente multimodal (MMDP) está específicamente diseñado para resultar difícil a un AE (Goldberg et al. 92). El problema MMDP consta de k subproblemas de 6 bits, de forma que el óptimo vale k y se alcanza cuando cada subproblema consta de cero, o bien de seis unos. Cada subproblema contribuye a la adecuación de la cadena según el número de unos presentes (Figura 1). El número de óptimos locales es enorme y crece exponencialmente con el número de subproblemas: 22^k óptimos de los que sólo 2^k son globales. Aquí usaremos una instancia con k=40 subproblemas (MMDP40).

4.2. El Algoritmo Evolutivo Paralelo

El AEP que usamos es un algoritmo genético distribuido en el que existen múltiples islas ejecutando un AG de estado estacionario (Syswerda 91) -*distributed* ssGA o dssGA-. Las islas estacionarias se disponen en un anillo unidireccional e intercambian una copia de una cadena aleatoriamente elegida con cierta frecuencia (migración). La población receptora vecina en el anillo reemplaza a su peor cadena si la nueva recibida es mejor.

El algoritmo utilizado en este estudio es asíncrono porque numerosos estudios demuestran las grandes ventajas que pueden obtenerse con este tipo de algoritmos en cuanto a tiempo de ejecución (Munetomo et al. 93) (Hart et al. 97) (Alba & Troya 99b).

La ganancia débil frente a panmixia (tipo II.A.1) mide el tiempo medio que un algoritmo de estado estacionario secuencial necesita frente a un algoritmo distribuido paralelo que utiliza un número creciente de islas, una por procesador, desde 2 hasta 8, dividiendo la población total en tantas islas como procesadores. La ganancia débil ortodoxa (tipo II.A.2) la mediremos ejecutando un algoritmo distribuido con 8 islas sobre un número creciente de procesadores (desde 1 hasta 8). La Tabla 2 muestra la configuración utilizada en ambos casos (*weak1* y *weak2*). Cada columna representa una máquina y cada casilla contiene un identificador de isla.

El algoritmo paralelo se ejecuta en un *cluster* de estaciones de trabajo SUN ULTRASparc 1 con Solaris y 64 MB de memoria principal, conectadas entre sí por una red ATM. Todos los resultados son valores medios después de realizar 100 ejecuciones independientes.

Tabla 2. *Esquema para medir ganancia débil de tipo* II.A.1 (*arriba*) *frente a tipo* II.A.2 (*abajo*).

TIPO	Nº islas	proc#1	proc#2	proc#3	proc#4	proc#5	proc#6	proc#7	proc#8
weak1 **(II.A.1)**	1	1							
	2	1	2						
	3	1	2	3					
	4	1	2	3	4				
	5	1	2	3	4	5			
	6	1	2	3	4	5	6		
	7	1	2	3	4	5	6	7	
	8	1	2	3	4	5	6	7	8
weak2 **(II.A.2)**	8	1,2,3,4,5,6,7,8							
	8	1,2,3,4	5,6,7,8						
	8	1,2,3	4,5,6	7,8					
	8	1,2	3,4	5,6	7,8				
	8	1,2	3,4	5,6	7	8			
	8	1,2	3,4	5	6	7	8		
	8	1,2	3	4	5	6	7	8	
	8	1	2	3	4	5	6	7	8

En las pruebas usamos siempre un total de 512 individuos repartidos entre las islas usadas, selección proporcional a la adecuación, cruce de dos puntos generando un único hijo (probabilidad 1.0), y mutación binaria simple con probabilidad $1/\ell$, siendo ℓ la longitud de la cadena (ℓ=512, ℓ=128, ℓ=240 para SPH16-32, SSS128 y MMDP40, respectivamente). En cada isla sólo se reemplaza a la peor cadena si la nueva generada es mejor (en cada paso).

5. GANANCIA DÉBIL

En esta sección mediremos la ganancia débil frente a panmixia (*weak1*) y la compararemos con la ganancia débil ortodoxa (*weak2*). A priori, parece claro que un AE distribuido presenta ventajas sobre otro en panmixia, al menos en un rango de problemas muy interesante para la investigación aplicada (Gordon & Whitley 93). Esto hace esperar que en muchos casos, distribuir la población reduzca el número de evaluaciones y por tanto disminuya el tiempo de ejecución. Si, además, el algoritmo distribuido se ejecuta sobre múltiples procesadores, la el tiempo de espera por una solución se reduce aún más.

Los resultados de la Tabla 3, así como la Figura 2 izqda. demuestran que es posible obtener ganancias superlineales en la práctica (la línea discontinua indica ganancia lineal en la Figura 2 izqda.). La cuantía de esta ganancia para *weak1* está en rangos desproporcionados debido a que el punto de referencia es un algoritmo secuencial en panmixia. Véase por ejemplo que con 8 procesadores la ganancia de velocidad es de 113.3. La medida *weak2* comprueba lo que ocurre con 8 islas al ejecutarlas en un número de procesadores que varía entre 1 y 8. Vemos que los valores son bastantes más razonables (i.e. ganancia 21.6 para 8 procesadores) porque ahora el algoritmo ejecutado en paralelo es básicamente el mismo numéricamente.

Tabla 3. *Resultados de ganancia débil de tipo* 1 *y* 2 *para SPH16-32.*

Nº procesadores	1	2	3	4	5	6	7	8
weak1	1.000	6.360	17.20	36.910	58.840	79.950	93.910	113.310
weak2	1.000	6.262	8.365	11.865	13.881	15.718	18.650	21.621

La Figura 2 dcha. muestra que lo mismo ocurre cuando usamos un problema más complejo como SSS128. La medida débil ortodoxa (*weak2*) es más fiel a la realidad que la medida frente a panmixia (*weak1*) debido a que se compara el mismo algoritmo. De cualquier forma, en el caso de 2 procesadores la ganancia es superlineal tanto con *weak1* como con *weak2*.

A pesar de que la medida débil ortodoxa sea más ajustada a la realidad y al campo de la computación paralela, todavía queda mucho por explorar respecto a la ganancia en AAEEPP. Las siguientes dos secciones muestran resultados con II.A.2 y algunos de los retos futuros.

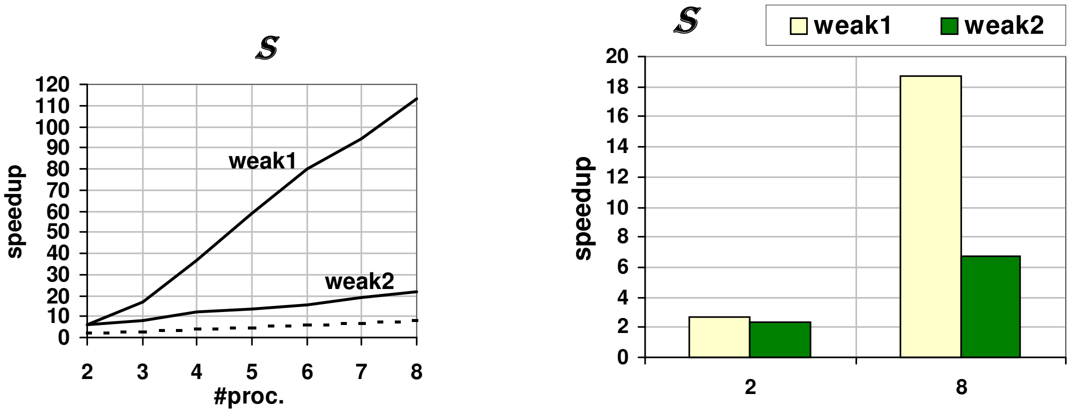

Figura 2. *Ganancia débil de tipo* 1 *frente a tipo* 2 *para SPH16-32* (izqda.) *y SSS128* (dcha.).

6. GANANCIA DÉBIL ORTODOXA

Como hemos observado es preferible usar una medida débil ortodoxa que permite obtener una idea más realista del algoritmo evaluado. Sin embargo, esto no significa que se pueda predecir la ganancia de un modelo paralelo, pues ésta depende por supuesto del problema, de su funcionamiento y de la implementación paralela. Para ilustrar el panorama de perfiles de ganancia posibles analizamos a continuación la ganancia débil ortodoxa (*weak2*) para los problemas SSS128 y MMDP40 (Tabla 4 y Figura 3).

Tabla 4. *Resultados de ganancia débil ortodoxa* (*weak2*) *para SSS128 y MMDP40.*

Nº procesadores	1	2	3	4	5	6	7	8
SSS128	1.000	2.367	1.899	7.222	3.353	3.142	3.575	6.734
MMDP40	1.000	1.431	2.163	2.848	3.373	3.740	6.354	8.058

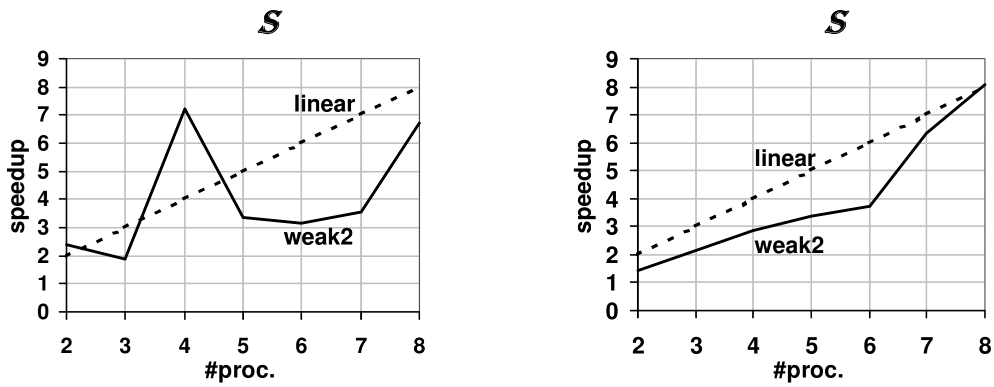

Figura 3. *Ganancia débil ortodoxa para los problemas SSS128* (izqda.) *y MMDP40* (dcha.).

El problema SSS128 presenta valores irregulares de ganancia, ya que se produce cierta vecindad entre las islas situadas en el mismo procesador (el efecto depende del problema). En este problema, debido a que el espacio de búsqueda es muy extenso y la diversidad de adecuaciones posibles muy intensa, el número de procesadores tiene un efecto especialmente visible en la ganancia obtenida.

En el caso de MMDP40 el resultado es bastante parecido al que puede obtenerse con algoritmos paralelos tradicionales. La ganancia es sublineal pero muy cercana a lineal. A partir de 6 procesadores la tendencia cambia porque con 7 y 8 procesadores situamos prácticamente una isla por procesador y el tiempo disminuye considerablemente frente a la asignación para menos de 6 procesadores, en la que muchos procesadores tienen dos islas.

7. CONSIDERACIONES ADICIONALES SOBRE LA GANANCIA DE VELOCIDAD

La ganancia es una medida útil que informa de la eficiencia de un algoritmo evolutivo paralelo. Además de las consideraciones previas sobre cómo medirla cabe hacer un estudio más profundo englobando aspectos adicionales. Algunas preguntas interesantes son:

- ¿Puede esperarse siempre una ganancia cercana a la lineal?
- ¿Es siempre un algoritmo distribuido mejor que otro en panmixia?
- ¿Cuáles son los ingredientes para obtener ganancia superlineal?

Aunque las respuestas no están totalmente claras, sí podemos avanzar algunas explicaciones lógicas. En general, no es posible esperar ganancia ni siquiera lineal (recuérdese el caso s<1 de la Ecuación 4). En primer lugar porque depende del modelo y del problema, y en segundo lugar porque existen parámetros algorítmicos que influyen en el tiempo de ejecución.

Respecto a la segunda pregunta, no siempre distribuir (o en general descentralizar) es mejor. El modelo paralelo puede modificar la velocidad de convergencia, y esto es bueno o malo dependiendo del problema. Por ejemplo, en el caso SSS128 usar islas en lugar de una población única proporciona mejores resultados únicamente cuando el intervalo de migración es elevado (16 o 32 generaciones). Para un acoplamiento fuerte entre las islas (migración en cada generación) el porcentaje de éxito es peor que para una población en panmixia (100%); de hecho, aumentar el número de islas es beneficioso (Figura 4 izqda., pasamos de 57% a 74%).

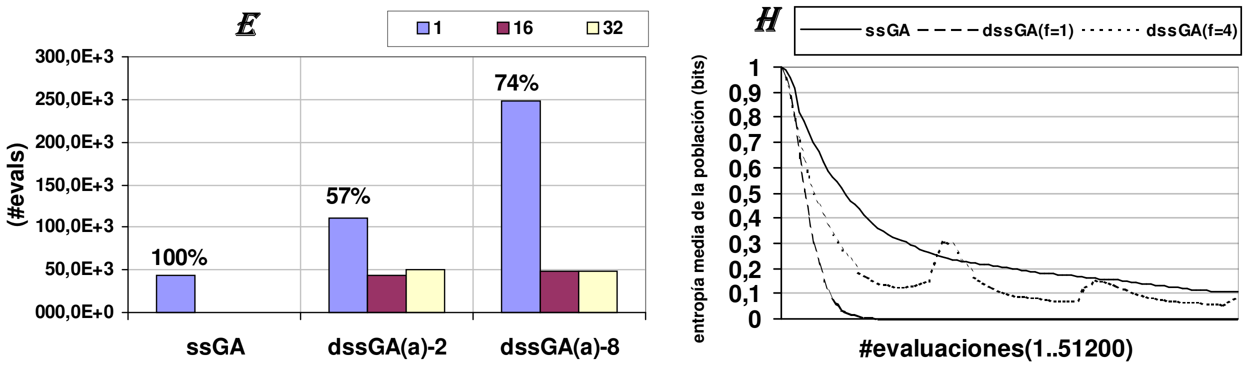

Figura 4. *Un algoritmo distribuido asíncrono con* 2 *u* 8 *islas e intervalo de migración* 1 *tiene peor porcentaje de éxitos que otro de estado estacionario secuencial en panmixia para SSS128* (izqda.). *Evolución de la diversidad en SPH16-32 con ssGA y dssGA* (dcha.).

Finalmente, los ingredientes para obtener ganancia superlineal podemos encontrarlos en dos frentes: numérico y físico. Desde el punto de vista numérico, proporcionar mayor diversidad, evolución casi separada, o un mayor número de islas es beneficioso en numerosas ocasiones. Compruebe por ejemplo en la Figura 4 dcha. cómo la diversidad (entropía media de la población) tiende a disminuir al distribuir en forma de islas la población. Como contrapartida, la diversidad puede fácilmente controlarse usando el intervalo de migración (pueden observarse los picos de diversidad en el caso de frecuencia de migración 4). Desde el punto de vista físico, la calidad de la implementación realizada, la topología, el grado de acoplamiento y el sincronismo son aspectos que deben cuidarse para no sobrecargar el tiempo de cálculo total.

8. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo hemos estudiado las consideraciones más importantes respecto a la ganancia de velocidad en algoritmos evolutivos paralelos. Hemos demostrado que es posible obtener ganancias superlineales en la práctica incluso imponiendo los requisitos más estrictos exigidos en el campo de la computación evolutiva, lo que nos ha llevado a proponer la ganancia débil ortodoxa (II.A.2) como un mecanismo más real y correcto que otras medidas existentes.

Parece necesario, a la luz de estos resultados, extender las conclusiones a problemas que incorporen distintas dificultades no consideradas en este trabajo, y también a otros modelos paralelos, si bien esto último ya se está estudiando como línea de trabajo en la actualidad.

La posibilidad de obtener ganancias superlineales es tan atractiva que merece un tratamiento aparte para profundizar en la caracterización de las situaciones que pueden conducir a conseguir este efecto.

REFERENCIAS

Akl S.G. (1992). *Diseño y Análisis de Algoritmos Paralelos*. RA-MA.

Alba E., Troya J.M. (1999a). "A Survey of Parallel Distributed Genetic Algorithms". *Complexity* 4 (4), pp. 31-52.

Alba E., Troya J.M. (1999b). "Analyzing Synchronous and Asynchronous Parallel Distributed Genetic Algorithms". *Future Generation Computer Systems*, (en prensa).

Bäck T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press.

Belding T.C. (1995). "The Distributed Genetic Algorithm Revisited". In Eshelman L.J. (ed.) *Proceedings of the Sixth ICGA*. Morgan Kaufmann, CA, pp. 114-121.

Calégari P., Guidec F., Kuonen P., Kobler D. (1997). "Parallel Island-Based Genetic Algorithm for Radio Network Design". *Journal of Parallel and Distributed Computing* 47, pp. 86-90.

Cantú-Paz E., Goldberg D.E. (1997). "Predicting Speedups of Idealized Bounding Cases of Parallel Genetic Algorithms". In Bäck T. (ed.), *Proceedings of the Seventh ICGA*. Morgan Kaufmann, CA, pp. 113-120.

Chipperfield A., Fleming P. (1996). "Parallel Genetic Algorithms". *Parallel and Distributed Computing Handbook*, Zomaya A.Y.H. (ed.), MacGraw-Hill, pp. 1118-1143.

Goldberg D.E., Deb K., Horn J. (1992). "Massively Multimodality, Deception and Genetic Algorithms". In Männer R., Manderick B. (eds.) Proceedings of the *PPSN II*. North-Holland, pp. 37-46.

Gordon V. S., Whitley D. (1993). "Serial and Parallel GAs as Function Optimizers". In Forrest S. (ed.) *Proceedings of the Fith ICGA*. Morgan Kaufmann, CA, pp. 177-183.

Hart W.E., Baden S., Belew R.K., Kohn S. (1997). "Analysis of the Numerical Effects of Parallelism on a Parallel Genetic Algorithm". In IEEE (ed.) *Procs. of the Workshop on Solving Comb. Opt. Problems in Parallel*. CD-ROM IPPS97.

Jelasity M. (1997). "A Wave Analysis of the Subset Sum Problem". In Bäck T. (ed.) *Proceedings of the Seventh ICGA*. Morgan Kaufmann, CA, pp. 89-96.

Lin S.C., Punch III W.F., Goodman E.D. (1994). "Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach". *Proceedings of the Sixth IEEE Parallel & Distributed Processing*, IEEE Press, pp. 28-37.

Mühlenbein H., Schomisch M., Born J. (1991). "The Parallel Genetic Algorithm as Function Optimizer". *Parallel Computing* 17, pp. 619-632.

Munetomo M., Takai Y., Sato Y. (1993). "An Efficient Migration Scheme for Subpopulation-Based Asynchronously Parallel GAs". Technical Report *HIER-IS-9301*, Hokkaido University.

Shonkwiler R. (1993). "Parallel Genetic Algorithms". In Forrest S. (ed.) *Proceedings of the Fith ICGA*. Morgan Kaufmann, CA, pp. 199-205.

Spiessens P., Manderick B. (1991). "A Massively Parallel Genetic Algorithm". In Belew R.K., Booker L.B. (eds.) *Proceedings of the Fourth ICGA*. Morgan Kaufmann, San Mateo, CA, pp. 279-286.

Sprave J. (1999). "A Unified Model of Non-Panmictic Population Structures in Evolutionary Algorithms". Technical Report CI-55/99, January.

Syswerda G. (1991). "A Study of Reproduction in Generational and Steady-State Genetic Algorithms". In Rawlins G. (ed.) *Foundations of GAs*. Morgan Kaufmann, pp. 94-101.

Tanese R. (1989). "Distributed Genetic Algorithms". In Schaffer J.D. (ed.) *Proceedings of the Third ICGA*. Morgan Kaufmann, CA, pp. 434-439.