Analizando el Desempeño de Distintas Topologías en Algoritmos Evolutivos Distribuidos

Bermúdez, Carlos; Alfonso, Hugo; Salto, Carolina

LISI - Laboratorio de Investigación en Sistemas Inteligentes
Facultad de Ingeniería, Universidad Nacional de La Pampa
Calle 110 esq. 9, (6360) General Pico, La Pampa, Argentina
bermudezc@yahoo.com, {alfonsoh | saltoc}@ing.unlpam.edu.ar

Resumen: En este trabajo se presenta un estudio comparativo de la eficiencia de algoritmos evolutivos trabajando sobre un entorno distribuido. La distribución utilizada consistió en dividir la población global en subpoblaciones (islas) interconectadas a través de diferentes topologías. Se evaluó el desempeño de estos algoritmos a partir de la elección entre las diversas topologías implementadas, las diferentes cantidades de individuos a migrar entre las islas y la cantidad de máquinas en las que se distribuirá la ejecución de cada una de las islas. Para medir la eficiencia del desempeño se usó, entre otras métricas, el speedup que nos permite evaluar el impacto del agregado de elementos de procesamiento al cluster de computadoras a usar. Los resultados obtenidos muestran que los algoritmos distribuidos superan a su contraparte secuencial tanto en tiempo (alto speedup) como en esfuerzo computacional (menor número de puntos visitados durante la búsqueda de la solución).

Palabras Clave: Paralelismo, Algoritmo Evolutivo, modelo isla, speedup, serial fraction.

1. Introducción

Las metaheurísticas fueron inicialmente desarrolladas para resolver problemas de optimización con variables discretas, los que reciben el nombre de problemas de optimización combinatoria (POC). Las metaheurísticas combinan los métodos heurísticos básicos dentro de estructuras (*frameworks)* de alto nivel, para lograr una mayor eficiencia y efectividad al explorar dentro de un espacio de búsqueda.

Existen diversas maneras de clasificar y describir los algoritmos metaheurísticos. Una clasificación importante, y que será utilizada en este trabajo, es la de métodos basados en población (MBP). Estos métodos trabajan en cada iteración del algoritmo, con un conjunto (población) de soluciones en vez de una única solución. de esta manera los MBP proveen una forma natural para explorar el espacio de búsqueda. Por lo tanto la manera en que esta población es manipulada afectará de forma directa la performance. Una clase de MBPs son los algoritmos evolutivos [10] (EA), los cuales se utilizarán en este trabajo, en particular los algoritmos genéticos [8] (GA).

Como una alternativa para mejorar la eficiencia de las metaheurísticas, han surgido varias propuestas que utilizan un enfoque de procesamiento distribuido de éstas. Con el uso de esta técnica se tratan de evitar los problemas típicos: uso intensivo de CPU, falta de memoria, como así también lograr una reducción en los tiempos de procesamiento, entre otros. Un método común para distribuir un EA, es utilizar múltiples poblaciones (islas) interconectadas de forma tal, que se pueda intercambiar individuos a través de un proceso llamado migración. En un algoritmo evolutivo distribuido (dEA) con múltiples islas se debe definir: tanto el tamaño de la población como el número de las islas, la topología de conexión o migración entre ellas, la tasa de migración, la

frecuencia de migración y la política para seleccionar los emigrantes y los individuos existentes que serán reemplazados por estos individuos que emigran.

El objetivo principal de este trabajo es realizar un análisis comparativo de la medida de eficiencia en distintos dEAs. En particular se adoptó un modelo isla y se medirá el desempeño que presenten, al modificar la topología de comunicación, la tasa de migración y la cantidad de máquinas. Dentro de las medidas de eficiencia que se van a analizar se encuentra el *speedup*, ya que éste nos permite determinar si el agregado de un elemento al cluster mejora el desempeño del algoritmo que se está evaluando.

En la segunda parte de este trabajo se hace una descripción de los dEAs, para poder luego enfocarnos en los distintos aspectos del modelo isla. En la cuarta sección se presentan las distintas métricas que utilizamos para medir la performance del modelo propuesto. Luego en las secciones 5 y 6 se describen los diferentes experimentos que se realizaron y los resultados obtenidos, respectivamente. Por último se hace un análisis más general de los resultados, presentando ventajas y desventajas al utilizar métodos distribuidos y/o paralelos.

2. Algoritmos Evolutivos Paralelos

Los EAs imitan el proceso de evolución natural que experimentan los individuos de las distintas especies. Esta evolución funciona básicamente con la *creación* de nueva información genética, utilizando procesos de *evaluación* y *selección*. Los individuos que estén mejor adaptados, tanto con relación al resto de los individuos de su población como al entorno al que están expuestos, tienen mayor posibilidad de vivir más tiempo y de reproducirse, generando así una progenie con un alto contenido de su información genética. En el transcurso de la evolución se van generando sucesivas poblaciones con la información genética de los individuos cuya adaptación fue superior a la media.

Los EA que utilizan una única población, a la que se les aplican los operadores de selección, recombinación y mutación, se denominan panmícticos o secuenciales. Cada individuo de esta única población puede potencialmente cruzarse con cualquier otro.

Por lo general los EAs secuenciales hacen uso intensivo de los recursos de computación, especialmente de la memoria física y del procesador, como consecuencia de trabajar con una única población de soluciones. Una alternativa a este enfoque sería disminuir la cantidad de evaluaciones que son necesarias para alcanzar una solución. Otra alternativa más viable es ejecutar el EA de forma paralela, esta técnica permite manejar poblaciones de mayor dimensión, con lo cual se podría resolver problemas de mayor tamaño y complejidad.

Un punto crítico a tener en cuenta al efectuar la paralelización, es que hay operaciones que necesitan disponer de toda la población para su cálculo, como es el caso del operador de selección de parejas que presenta un cuello de botella ya que utiliza panmixia. Otra alternativa para distribuir un EA sería, contrariamente al caso anterior en que se utilizaba una única población, dividirla en subpoblaciones de cierto tamaño que se comunican entre sí. Estos algoritmo así paralelizados se comportan de forma distinta a los panmícticos, basándose en el hecho de la evolución natural en donde no es posible que cualquier individuo se aparee con cualquier otro. Esta variante logra en la mayoría de los casos mejorar el proceso de búsqueda.

La evolución en grupos separados es más natural, dando lugar a especies o nichos. Estos grupos pueden verse como islas de individuos que están espacialmente estructurados y van evolucionando en forma paralela.

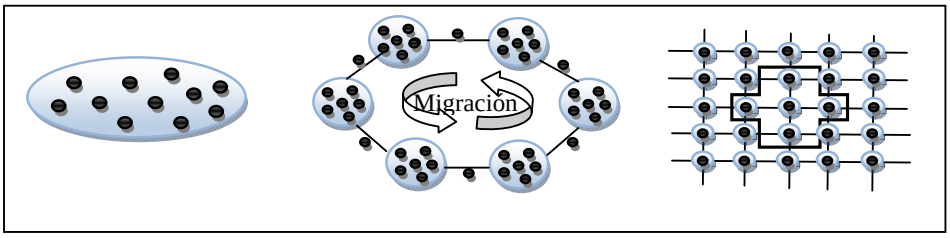

Figura 1. Distribución espacial de la población, (a) Panmíctico, (b) distribuido y (c) celular.

En la Figura 1 se pueden observar los tipos más comunes de EAs. En la Figura 1(a) se muestra una población única denominada panmíctica. El modelo distribuido [11], Figura 1(b), también llamado modelo isla, en donde simplemente se divide una población panmíctica en varias subpoblaciones más pequeñas, cada una de éstas corre un EA secuencial estándar y a los individuos se les permite migrar entre poblaciones con una frecuencia determinada. En este gráfico los círculos representan las subpoblaciones y las flechas muestran el patrón de intercambio poblacional que se utilizará en el modelo. Existe una variada gama de patrones que se pueden utilizar para el intercambio de individuos, siendo los más comunes: anillo, grilla de dos y tres dimensiones, estrella e hipercubos. En la Figura 1(c) se puede observar el modelo celular o modelo de difusión [9], en donde los individuos están dispuestos en una topología en forma de grilla. En este modelo cada individuo interactúa sólo con unos pocos individuos, siempre dentro de su vecindario.

En particular, en este trabajo usamos el modelo isla, centrando el análisis en la topología de comunicación, la tasa de migración y la escalabilidad de máquinas.

3. Modelo Isla

Una población estructurada o distribuida, es simplemente una población en la que un individuo dado tiene su propio vecindario, que generalmente es mucho más chico que el tamaño de la población total. Es decir que, de los restantes individuos de la población que podrían ser considerados para cruzar, como es el caso de una población panmíctica, sólo los individuos que están en el mismo vecindario pueden interactuar.

El modelo isla o modelo multipoblacional, es muy similar al modelo panmíctico original, tanto que el correspondiente EA no necesita una reestructuración radical, y son sólo variantes del estándar. El fenómeno de formación de nichos es común en la biología. Los nichos imponen restricciones de cruza entre los individuos y así la posibilidad de diferenciación de las especies. Las islas y los valles aislados, son el tipo de ambiente natural que puede favorecer tal fenómeno evolutivo. Cuando otros individuos tienen la posibilidad de interactuar con estas islas, ocurren procesos evolutivos interesantes, tales como: colonización, extinción y punto de equilibrio. De la observación de estas poblaciones biológicas reales, los investigadores pueden obtener ideas y aplicarlas para mejorar sus algoritmos.

La principal ventaja del modelo isla es que se puede explorar el espacio de búsqueda del problema, de forma más intensiva y que se puede disminuir el estancamiento poblacional, gracias a una mejor capacidad para mantener, por sobre todo, la diversidad.

La idea principal del modelo isla [12, 7], según Tomassini [13], es subdividir una simple población panmíctica de tamaño N en s islas, de tamaño más pequeño n, tal que $\sum_{i=1}^{s} n_i = N$. El tamaño de una población distribuida no tiene que estar necesariamente relacionada con su versión panmíctica, simplemente se utiliza de esta forma para poder hacer comparaciones del desempeño que presenta la versión distribuida contra la panmíctica. Cada población corre un EA estándar como si estuviera aislado del resto. Sin embargo, de tanto en tanto, las islas reciben y envían individuos desde y hacia otras islas. Estos grupos de individuos de tamaño m, con $m << n_i$, son llamados

emigrantes. El intercambio de estos individuos puede ser hecho en forma asíncrona o síncrona en el tiempo. Esto significa que los intercambios pueden suceder en un tiempo fijo, preestablecido, o el intercambio puede suceder en tiempos independientes.

Todas las islas pueden correr en una única máquina, lo que llamaremos una ejecución serie. También se puede repartir la cantidad de islas entre las distintas máquinas que componen el cluster, en este caso se denomina una ejecución en paralelo.

Además de los parámetros normales de un EA en un modelo isla, se necesitan otros adicionales como: número de subpoblaciones, frecuencia de migración de los individuos, el número de individuos que van a migrar y la topología de comunicación. La existencia de estos grados de libertad adicionales permite que el algoritmo sea más flexible, pero también más difícil de controlar debido a que los parámetros pueden interactuar de muchas maneras. Desafortunadamente no existe una teoría general que nos oriente a la hora de seleccionar los parámetros que mejor se adaptan a nuestro algoritmo y en la mayoría de los casos debe hacerse en forma empírica.

Otro factor a tener en cuenta, a la hora de la migración, es determinar qué individuos se deben seleccionar de la población de origen y qué individuos se deben reemplazar en la población destino. Esto es, seleccionar los mejores y reemplazar los peores, seleccionar los peores y reemplazar los mejores, etc. entre otras varias alternativas posibles.

Se pueden utilizar varias topologías para la migración de individuos entre las islas. La más común es disponer la población en forma de anillo. En ésta, la población está dispuesta en círculo y el intercambio toma lugar entre las subpoblaciones vecinas, como muestra la Figura 2(a).

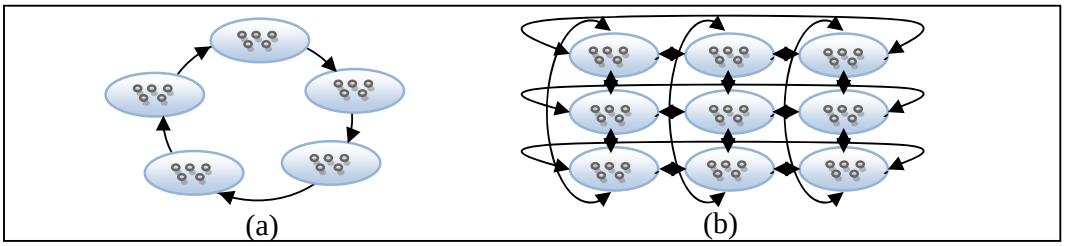

Figura 2. Topologías de comunicación, (a) anillo (b) grilla toroidal.

Otra posibilidad es como muestra la Figura 2(b), en donde las islas están dispuestas de manera tal que forman una malla toroidal o grilla. En este caso la migración ocurre entre los vecinos más cercanos, es decir que un individuo cualquiera de la población se comunicará con sus vecinos, que se encuentran al norte, sur, este y oeste.

Las dos topologías anteriores son estáticas, es decir que siempre se realiza el intercambio de individuos entre los mismos vecinos. En el otro extremo tenemos topologías dinámicas, en las cuales, las subpoblaciones de destino van cambiando con el tiempo, esto permite mantener una mayor diversidad en las subpoblaciones. Una forma de lograr esto es seleccionando la isla destino de forma aleatoria. En este trabajo se utilizará una modalidad total de migración, en la que una isla envía y recibe individuos de cada una de las islas restantes, este método tendría la misma finalidad que la migración aleatoria, manteniendo la mayor diversidad posible.

4. Medidas de Performance en EAs

A diferencia de los métodos exactos, en donde la eficiencia con respecto al tiempo es la principal métrica de éxito, en las metaheurísticas paralelas se necesita evaluar dos aspectos principales para determinar su eficiencia: con qué rapidez se obtiene el resultado y cuán lejos del óptimo se encuentra.

Una de las métricas más utilizadas para medir metaheurísticas paralelas es el *speedup*. Su función es comparar el tiempo que toma el algoritmo en su versión serie contra el tiempo que se necesita para resolverlo en su versión paralela. Con T_m denotamos el tiempo de ejecución para un algoritmo en m procesadores, entonces el speedup S_m como se observa en la Ecuación 1, es la relación entre la ejecución más rápida en un único procesador T_1 y el tiempo de ejecución en m procesadores T_m:

$$S_m = \frac{T_1}{T_m} \tag{1}$$

El problema es que no se puede utilizar directamente esta métrica en algoritmos estocásticos. Para este tipo de métodos, el speedup debe comparar el tiempo medio de todas las ejecuciones en serie contra el tiempo medio de todas las ejecuciones paralelas como se ve en la Ecuación 2:

$$S_m = \frac{E[T_1]}{E[T_m]} \tag{2}$$

Con esta definición podemos distinguir varios tipos de speedup: sublineal cuando $S_m<m$, lineal cuando $S_m=m$, y superlineal en el caso que sea $S_m>m$. Existe una dificultad con esta métrica ya que los investigadores no están de acuerdo con el significado de T_1 y T_m. Alba [2] hace una clasificación del speedup en función de estos valores, como se puede observar en la Figura 3.

- **I.** Speedup fuerte
- **II.** Speedup débil
  - **a.** Speedup con parada por solución
    - **i.** Versus panmíctico
    - **ii.** Ortodoxo
  - **b.** Speedup con esfuerzo predefinido

Figura 3. Taxonomía del speedup.

El speedup fuerte (I) compara el tiempo que tarda en correr un algoritmo paralelo contra el mejor de todos los algoritmos secuenciales. Ésta es la definición más exacta del speedup, pero debido a la dificultad que presenta encontrar el mejor de todos los algoritmos secuenciales, no es muy utilizado por los diseñadores de algoritmos paralelos. El speedup débil (II) compara el algoritmo paralelo desarrollado por el investigador contra su propia mejor versión serial. En este caso existen dos criterios para que se detenga el algoritmo: por la calidad de la solución o por esfuerzo máximo. Este último es descartado por Alba y Troya [1], ya que los algoritmos paralelos deben ser comparados corriéndolos hasta encontrar una solución de la misma calidad para todos y no hasta que se complete el mismo número de pasos para todos los algoritmos. Por tal motivo se proponen dos variantes del speedup débil con parada por solución: La primera sería comparar el algoritmo paralelo contra la versión secuencial canoníca (II.a.i), que serían dos algoritmos diferentes. La segunda sería comparar el tiempo de corrida del algoritmo paralelo en un procesador contra el tiempo de corrida del mismo algoritmo en m procesadores (II.a.ii).

Existen otras clasificaciones del speedup, como la que utilizan los autores Barr y Hickman [6] que son similares a las definidas anteriormente pero con términos diferentes, ya que clasifican el speedup como absoluto o relativo que tendría una correspondencia con el ortodoxo y el fuerte respectivamente que propone Alba.

Existen otras medidas de performance para evaluar una metaheurística paralela. La *eficiencia* e_m , Ecuación 3, es una normalización del speedup y permite comparar diferentes algoritmos, cuando ésta es igual a la unidad, e_m=1, obtenemos un speedup lineal:

$$e_m = \frac{S_m}{m} \tag{3}$$

Otra métrica muy utilizada es el *Serial Fraction* que nos permite medir la performance de cualquier algoritmo paralelo y además, puede ayudar a identificar algunos efectos de forma más sutil que si estuviéramos solamente utilizando el speedup. Esta medida se denomina serial fraction del algoritmo f_m, Ecuación 4.

$$f_m = \frac{\frac{1}{sm} - \frac{1}{m}}{1 - \frac{1}{m}} \quad (4)$$

Idealmente, el serial fraction debe permanecer constante para un algoritmo. Si el valor del speedup es pequeño, podemos decir que el resultado es bueno si f_m permanece constante para diferentes valores de m. Por otra parte, un incremento de f_m es una advertencia que indica que la granularidad de las tareas paralelas es demasiado fina. Es posible un tercer escenario, en donde ocurre una significativa reducción de f_m, esto indicaría que se está en presencia de un speedup superlineal y f_m tomaría valores negativos.

5. Experimentación

El dEA considerado en este trabajo utiliza en cada isla un GA básico. El dEA será analizado con distintas topologías de comunicación y variaciones en la tasa de migración. Los experimentos que realizamos fueron, primeramente con una población panmíctica sobre una máquina y luego con cuatro islas, que fueron localizadas en una, dos y cuatro máquinas.

Dentro de las topologías de comunicación entre las islas se adoptó la disposición en forma de anillo circular, la grilla de dos dimensiones y por último se realizaron pruebas con una comunicación total entre las islas. Estas cuatro islas se corrieron en una única máquina para obtener la versión serie, que fue utilizado para contrastar con los demás resultados y obtener el speedup. Luego se ejecutaron estas cuatro islas en forma paralela en dos máquinas, ubicando dos islas por máquina. Por último se utilizaron cuatro máquinas, poniendo a ejecutar cada isla en una máquina distinta.

Utilizamos como regla nemotécnica para poder referenciar cada uno de los experimentos, la siguiente notación. La primera letra indica el tipo de topología, correspondiendo la A para anillo, la G para grilla y la T para total. Los siguientes dos caracteres indican la cantidad de emigrantes. Por último los restantes dos caracteres indicarían la cantidad de máquinas en que se ejecuta. Para ejemplificar mejor esto, supongamos que para un experimento de cuatro islas con una topología en anillo, con cinco emigrantes y ejecutado sobre cuatro máquinas, tendríamos “A5E4M” como nomenclatura.

Para probar los algoritmos se usaron dos problemas clásicos, en primer lugar las pruebas fueron con el problema One-Max [6], luego se efectuó el mismo procedimiento en forma separada con un problema más complejo, como es el P-Peaks [5].

Estos algoritmos fueron implementados dentro de la biblioteca MALLBA [3]. Para más información, se puede visitar la URL http://neo.lcc.uma.es /mallba/easy-mallba/index.hml. Todo el código de esta biblioteca está desarrollado en C++ y brinda un esqueleto algorítmico para optimización combinatoria, es decir que debe ser debidamente instanciada con las características específicas del problema para que pueda trabajar. Los experimentos se realizaron sobre un cluster homogéneo conformado por cuatro máquinas, equipadas con procesadores Pentium IV, 256Mb RAM, Disco 20Gb y sistema operativo Linux, interconectadas con un enlace fast Ethernet de 100Mbps.

Los parámetros que se utilizaron son iguales para todos los experimentos, lo cual nos permite hacer una comparación más equitativa. En nuestro caso utilizamos cuatro islas, una población total de μ=512 individuos, cada isla trabaja con una subpoblación de μ/4. La cantidad de hijos fue de λ=256 y cada isla genera λ/4 hijos.

El algoritmo itera hasta alcanzar el valor óptimo o el número máximo de evaluaciones, en nuestro caso treinta mil. La representación adoptada es binaria. Para generar los hijos se selecciona de cada subpoblación dos ternas de individuos. De estas ternas se selecciona un padre de cada una, aplicando selección por torneo. Luego a los dos padres seleccionados se les aplica crossover de dos puntos con una probabilidad de 0,85, para generar dos hijos. En el paso siguiente se les aplica mutación *swap* a estos hijos, con una probabilidad de 0,001. Por último, se une la población μ con los λ hijos generados y de estos se selecciona la nueva población utilizando el método de la ruleta.

También se efectuaron pruebas variando la cantidad de emigrantes entre las islas, para cada una de las topologías utilizadas, en este caso migramos: uno, dos y cinco individuos. Siempre utilizando la misma frecuencia de migración, cada diez iteraciones. La política de migración empleada consiste en seleccionar los mejores individuos de cada isla y enviarlos a la isla destino para que reemplacen a los peores.

6. Análisis de Resultados

En esta sección analizaremos los resultados obtenidos por las diferentes variantes de dGAs propuestos sobre los problemas seleccionados. Por cada variante algorítmica hemos realizado treinta ejecuciones independientes usando los parámetros descriptos en la sección previa. La cantidad de ejecuciones se debe a los requisitos planteados anteriormente, en donde se necesitaba un promedio de varias corridas, debido a la naturaleza estocástica del algoritmo.

Con respecto al speedup, se utilizara la versión ortodoxa del mismo, por lo tanto se contrastarán los tiempos obtenidos en dos y cuatro máquinas contra los tiempos obtenidos en una máquina, pero siempre con el mismo algoritmo paralelo. Mientras que el experimento realizado con una población panmíctica se utilizará como referencia para determinar a partir de qué número de máquinas es más productivo un algoritmo paralelo.

En las Tablas 1 y 2 están plasmados los valores que se obtuvieron para el problema One-Max y P-Peaks respectivamente. Para este último se utilizó una representación de 512 bits para cada individuo y 512 picos. Estas tablas contienen las siguientes columnas: número de evaluaciones (#Eval.), número de iteraciones (#Iter.) y el tiempo expresado en segundos. Estas columnas se repiten para las ejecuciones en una, dos y cuatro máquinas. En la última fila de estas tablas se encuentran los valores obtenidos con una población panmíctica, que servirán como referencia al analizar los restantes experimentos. Cabe resaltar que no se incluye una columna con la mejor solución encontrada ya que todos los algoritmos pudieron resolver el problema al nivel requerido de exactitud, obteniendo el óptimo en todas las ejecuciones, para todas las variantes algorítmicas.

	1M			2M			4M		
	#Eval.	#Iter.	Tiempo	#Eval.	#Iter.	Tiempo	#Eval.	#Iter.	Tiempo
A1E	48981,333	763,333	160,997	50429,867	785,967	63,579	53719,467	837,367	35,130
G1E	49518,933	771,733	168,407	54760,533	853,633	63,091	53418,667	832,667	36,815
T1E	48885,333	761,833	154,657	50922,667	793,667	61,438	52595,200	819,800	32,358
A2E	49785,600	775,900	163,527	51313,067	799,767	64,605	53273,600	830,400	35,585
G2E	49164,800	766,200	167,058	53568,000	835,000	62,709	53324,800	831,200	36,496
T2E	49367,467	769,367	157,747	51419,733	801,433	62,315	54950,400	856,600	34,981
A5E	48674,133	758,533	160,452	51509,333	802,833	64,956	52106,667	812,167	34,666
G5E	48868,267	761,567	166,594	56352,000	878,500	67,080	52759,467	822,367	36,465
T5E	50007,467	779,367	162,931	51121,067	796,767	64,733	54150,400	844,100	35,068
Pan.	100428,800	390,300	75,218						

Tabla 1. Datos obtenidos para One-Max.

De los valores que presenta la Tabla 1 se procede en primer lugar a hacer un análisis sobre el número de evaluaciones y tiempo consumido por cada una de las topologías, contrastando las evaluaciones y el tiempo contra la cantidad de emigrantes, para una, dos y cuatro máquinas.

El patrón de comportamiento que se produce en una y cuatro máquinas es similar con respecto al tiempo empleado para encontrar la solución óptima independiente de la cantidad de individuos a migrar. Es importante resaltar que la cantidad de evaluaciones necesarias para hallar el valor óptimo es mucho mayor para los algoritmos que se ejecutan en cuatro máquinas que aquellos que se ejecutan en una sola máquina. Esto a pesar de tener un tiempo de ejecución mucho mayor con una máquina, se debe al hecho de que al trabajar sobre una máquina, todas las islas se encuentran en una situación semejante, en cuanto al número de iteraciones. Por lo tanto, la difusión de la mejor solución, o de los mejores individuos se realiza de forma más temprana que utilizando cuatro máquinas, en donde las islas se pueden encontrar en números de iteración muy dispares y una isla, en estado avanzado, deba esperar para recibir una buena solución de sus vecinas, ya que se trabajó de forma asíncrona.

Otro aspecto importante a resaltar, es que en la versión propuesta con migración total se observa un tiempo bastante inferior a las restantes topologías. Esto último se debe a que las mejores soluciones que una isla encuentra se propagan mucho más rápido al resto, lo cual hace notar la importancia de mantener la mayor diversidad posible, cuando necesitamos disminuir los tiempos de ejecución.

El número de evaluaciones para encontrar el valor óptimo en la versión panmíctica es mucho mayor que las arrojadas por los distintos algoritmos que utilizan una población distribuida espacialmente. Esta ventaja se debe principalmente que al tener un mayor número de poblaciones, se brinda la posibilidad de que se seleccione una mayor cantidad de buenos individuos, en contraposición a los que serían seleccionados si estuviéramos utilizando una única población.

Como se puede ver en la tabla anterior, el resultado presenta un comportamiento muy dispar, utilizando uno, dos y cinco emigrantes, lo que no nos permite efectuar un análisis para determinar si la cantidad de individuos afecta en forma directa el resultado.

A continuación presentamos los resultados de valores de speedup alcanzados por los algoritmos distribuidos para el problema en cuestión.

	Tiempo	Speedup	Eficiencia	S.Fraction
A1E1M	160,997	1,000		
A1E2M	63,579	2,532	1,266	-0,210
A1E4M	35,130	4,583	1,146	-0,042

	Tiempo	Speedup	Eficiencia	S.Fraction
A2E1M	163,527	1,000		
A2E2M	64,605	2,531	1,266	-0,210
A2E4M	35,585	4,595	1,149	-0,043

	Tiempo	Speedup	Eficiencia	S.Fraction
A5E1M	160,452	1,000		
A5E2M	64,956	2,470	1,235	-0,190
A5E4M	34,666	4,628	1,157	-0,045

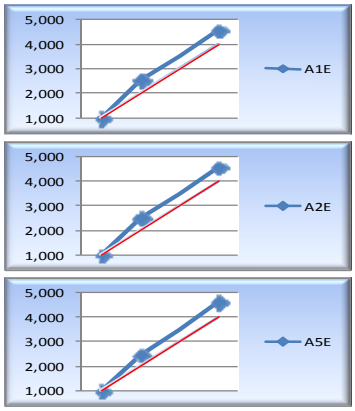

Figura 4. Performance de la topología anillo para el problema One-Max.

Para el caso de la topología en anillo, se obtuvo un speedup superlineal como lo muestra la Figura 4. Por otra parte, con respecto al serial fraction, podemos observar que es negativo lo que nos estaría confirmando la presencia de un speedup superlineal. Además en el caso de dos máquinas es mucho menor que en cuatro máquinas en el que prácticamente se aproxima a cero. Esto estaría indicando que paralelizar el algoritmo sobre cuatro máquinas seria lo ideal ya que si agregamos más máquinas al cluster no tendríamos un beneficio muy significativo.

Para la topología en forma de grilla se puede observar en la Figura 5 un comportamiento similar a la topología en anillo, ya que los tiempos de ejecución son muy parecidos, con un leve incremento que se debe al hecho de emigrar dos individuos en lugar de uno, como es el caso del

	Tiempo	Speedup	Eficiencia	S.Fraction
G1E1M	168,407	1,000		
G1E2M	63,091	2,669	1,335	-0,251
G1E4M	36,815	4,574	1,144	-0,042

	Tiempo	Speedup	Eficiencia	S.Fraction
G2E1M	167,058	1,000		
G2E2M	62,709	2,664	1,332	-0,249
G2E4M	36,496	4,577	1,144	-0,042

	Tiempo	Speedup	Eficiencia	S.Fraction
G5E1M	166,594	1,000		
G5E2M	67,080	2,483	1,242	-0,195
G5E4M	36,465	4,569	1,142	-0,041

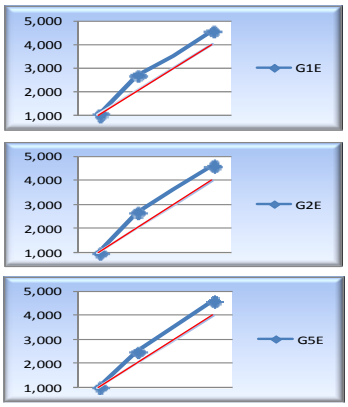

Figura 5. Performance de la topología grilla para el problema One-Max.

	Tiempo	Speedup	Eficiencia	S.Fraction
T1E1M	154,657	1,000		
T1E2M	61,438	2,517	1,259	-0,205
T1E4M	32,358	4,779	1,195	-0,054

	Tiempo	Speedup	Eficiencia	S.Fraction
T2E1M	157,747	1,000		
T2E2M	62,315	2,531	1,266	-0,210
T2E4M	34,981	4,509	1,127	-0,038

	Tiempo	Speedup	Eficiencia	S.Fraction
T5E1M	162,931	1,000		
T5E2M	64,733	2,517	1,258	-0,205
T5E4M	35,068	4,646	1,162	-0,046

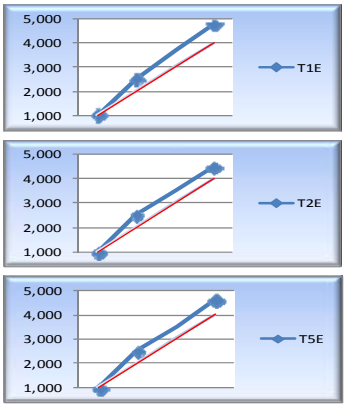

Figura 6. Performance de la topología total para el problema One-Max.

anillo. Nuevamente el serial fraction próximo a cero en las ejecuciones sobre cuatro máquinas nos indica que no se conseguirán grandes mejoras al incorporar más máquinas al cluster.

En la migración total, como muestra la Figura 6, también se logra obtener un speedup superlineal al igual que en las dos versiones de migración anteriores. En este caso, aunque la cantidad de islas vecinas a las que se está migrando es mayor que en la versión anillo y malla, con lo que tendríamos un incremento lógico de tiempo, el algoritmo arroja tiempos bastante bajos, sobre todo con un emigrante. Esto sería un indicador de que el desempeño que se obtiene migrando a la mayor cantidad de vecinos posible es mucho más beneficioso, en cuanto a reducir los tiempos de ejecución.

A continuación se muestran, en la Tabla 2, los valores obtenidos para el problema P-Peaks, utilizando los mismos parámetros que en el problema One-Max. En ésta se puede observar que tanto el incremento del tiempo como el número de evaluaciones es mucho mayor con respecto al problema One-Max. Este aumento se debe a la complejidad que presenta el problema P-Peaks, ya que tiene que comparar cada uno de los individuos con los 512 picos que se generaron, para poder determinar cual se encuentra más próximo al óptimo.

La opción total de migración va superando a las restantes topologías, tanto en cantidad de evaluaciones como en tiempo, al incrementar la cantidad de máquinas para su ejecución. Con una máquina la topología total es bastante peor que las restantes, pero al utilizar dos máquinas se mejora la situación equiparándose a las demás topologías. Por último con la ejecución sobre cuatro máquinas, la ventaja que se obtiene con este método de comunicación sobre las restantes topologías es bastante superior. En este problema también se puede observar que la versión panmíctica

	1M			2M			4M		
	#Eval.	#Iter.	Tiempo	#Eval.	#Iter.	Tiempo	#Eval.	#Iter.	Tiempo
A1E	50664,533	789,633	2860,216	50286,933	783,733	1360,760	50261,333	783,333	679,524
G1E	50698,667	790,167	2889,597	49787,733	775,933	1357,595	50909,867	793,467	689,589
T1E	49841,067	776,767	2789,462	50406,400	785,600	1359,399	50764,800	791,200	680,640
A2E	49591,467	772,867	2811,188	50291,200	783,800	1379,680	50553,600	787,900	681,349
G2E	50041,600	779,900	2849,339	49683,200	774,300	1358,713	52883,200	824,300	707,140
T2E	51598,933	804,233	2864,196	50890,667	793,167	1361,895	50856,533	792,633	683,106
A5E	49472,000	771,000	2795,982	49685,333	774,333	1340,044	50811,733	791,933	685,936
G5E	48814,933	760,733	2784,654	51210,667	798,167	1392,813	52501,333	818,333	703,691
T5E	50289,067	783,767	2817,890	49557,333	772,333	1347,245	50389,333	785,333	678,382
Pan.	104371,200	405,700	1492,475						

Tabla 2. Datos obtenidos para P-Peaks.

	Tiempo	Speedup	Eficiencia	S.Fraction
A1E1M	2860,216	1,000		
A1E2M	1360,760	2,102	1,051	-0,048
A1E4M	679,524	4,209	1,052	-0,017

	Tiempo	Speedup	Eficiencia	S.Fraction
A2E1M	2811,188	1,000		
A2E2M	1379,680	2,038	1,019	-0,018
A2E4M	681,349	4,126	1,031	-0,010

	Tiempo	Speedup	Eficiencia	S.Fraction
A5E1M	2795,982	1,000		
A5E2M	1340,044	2,086	1,043	-0,041
A5E4M	685,936	4,076	1,019	-0,006

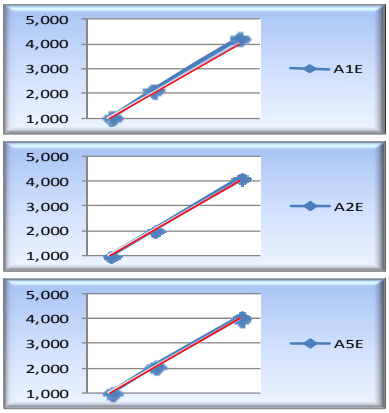

Figura 7. Performance de la topología anillo para el problema P-Peaks.

presenta un número elevado de evaluaciones con respecto a las restantes topologías, por el mismo motivo que en el problema One-Max.

Con respecto al tiempo empleado por la topología en anillo, probado en distintas máquinas, podemos observar en la Figura 7 que el speedup logrado, si bien no es tan superlineal como se da en el problema One-Max, es muy satisfactorio ya que supera levemente la linealidad.

En este experimento tenemos nuevamente un serial fraction negativo que aumenta a medida que incrementamos la cantidad de máquinas.

	Tiempo	Speedup	Eficiencia	S.Fraction
G1E1M	2889,597	1,000		
G1E2M	1357,595	2,128	1,064	-0,060
G1E4M	689,589	4,190	1,048	-0,015

	Tiempo	Speedup	Eficiencia	S.Fraction
G2E1M	2849,339	1,000		
G2E2M	1358,713	2,097	1,049	-0,046
G2E4M	707,140	4,029	1,007	-0,002

	Tiempo	Speedup	Eficiencia	S.Fraction
G5E1M	2784,654	1,000		
G5E2M	1392,813	1,999	1,000	0,000
G5E4M	703,691	3,957	0,989	0,004

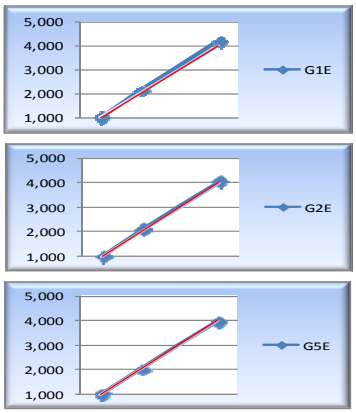

Figura 8. Performance de la topología grilla para el problema P-Peaks.

Los tiempos logrados con la topología de comunicación en grilla, Figura 8, arrojaron valores de speedup muy similares a la comunicación en anillo, pero levemente inferiores. También se puede observar para este caso, que a medida que aumenta la cantidad de emigrantes se va perdiendo

	Tiempo	Speedup	Eficiencia	S.Fraction
T1E1M	2789,462	1,000		
T1E2M	1359,399	2,052	1,026	-0,025
T1E4M	680,640	4,098	1,025	-0,008

	Tiempo	Speedup	Eficiencia	S.Fraction
T2E1M	2864,196	1,000		
T2E2M	1361,895	2,103	1,052	-0,049
T2E4M	683,106	4,193	1,048	-0,015

	Tiempo	Speedup	Eficiencia	S.Fraction
T5E1M	2817,890	1,000		
T5E2M	1347,245	2,092	1,046	-0,044
T5E4M	678,382	4,154	1,038	-0,012

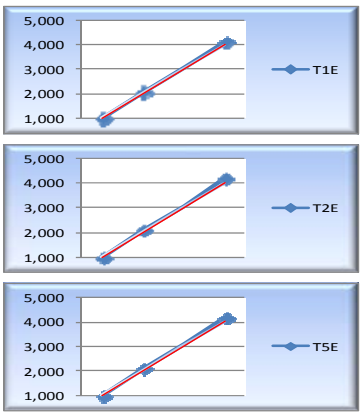

Figura 9. Performance de la topología total para el problema P-Peaks.

rendimiento como lo muestra el serial fraction que específicamente para cinco emigrantes ya se obtienen valores positivos.

En el caso del método de comunicación en forma total, se puede confirmar a través de la Figura 9 que el speedup alcanzado con esta topología es mayor que en las otras dos variantes probadas. Esto nuevamente nos da un indicio sobre cómo mejora el desempeño del algoritmo al enviar individuos al mayor número posible de islas.

7. Conclusiones

En este trabajo hemos analizado el comportamiento de algoritmos evolutivos distribuidos al variar algunas características del mismo. En primer lugar se probaron distintas topologías de comunicación entre las islas, para determinar si a mayor diversidad se logra alguna reducción en los tiempos de ejecución. Por otro lado se realizaron pruebas con distintas tasas de migración entre las islas, a fin de obtener algún indicio de cómo afecta al resultado el volumen de emigrantes. Por último se realizaron pruebas incrementando la cantidad de máquinas que componen el cluster y con esto determinar si tanto un problema simple como uno de complejidad elevada son aptos para ser ejecutados en forma paralela.

Con respecto a la topología de comunicación nuestros resultados muestran que usar una topología de comunicación total logra mejorar la eficiencia del algoritmo, superando las dos topologías restantes cuando se ejecutaron los experimentos utilizando cuatro máquinas. Esto se debe a que los mejores individuos que cada isla encuentra se difunden en forma más rápida al resto.

Los resultados que se lograron al variar la tasa de migración presentaron un comportamiento muy dispar en cada una de las topologías. Este hecho no nos permite obtener una conclusión que sea relevante. Este suceso tendría su explicación en el hecho que siempre se está emigrando el mejor individuo de cada subpoblación, por lo tanto enviar un individuo solo tendría el mismo efecto que enviar varios.

Por último se obtuvieron muy buenos resultados paralelizando los algoritmos en dos y cuatro máquinas, este hecho lo demuestra el speedup que se obtuvo en cada caso. Logrando un speedup superlineal para el problema One-Max y un speedup que supera ligeramente la linealidad para el problema P-Peaks.

La medida de eficiencia que se obtuvo en cada caso nos permite hacer comparaciones entre los dos problemas evaluados y se puede decir que en el caso de One-Max se obtiene un mayor beneficio al ejecutarlo en forma paralela. No siendo por esto despreciable la reducción en cuanto a tiempo y cantidad de evaluaciones que se obtienen para el problema P-Peaks.

Los resultados obtenidos de los experimentos confirman que es propicio ejecutar un algoritmo en forma paralela con lo que se gana una cantidad de tiempo importante. Por este motivo sería importante probar este efecto sobre un cluster con más máquinas, por ejemplo en ocho y dieciséis máquinas, sobre todo para comprobar si el aumento que experimenta el serial fraction nos da un indicio sobre el poco beneficio que se lograría incorporar más máquinas. Con respecto a la cantidad de islas vecinas a las que se envían individuos, sería importante probar con una mayor cantidad de islas y usar la topología total de comunicación para determinar si este hecho se cumple para cualquier cantidad de islas o tiene un límite máximo.

Reconocimientos

Este trabajo ha sido realizado con el apoyo de la Universidad Nacional de La Pampa y de la ANPCYT de Argentina. Se reconoce especialmente la cooperación de los integrantes del grupo del proyecto por aportar nuevas ideas y criticas constructivas.

Bibliografía

1. **Alba, E. and Troya, J.** An Analysis of Synchronous and Asynchronous Parallel Distributed Genetic Algorithms with Structured and Panmictic Islands. *Springer Berlin. Vol 1586, Pgs 248-256. ISBN 978-3-540-65831-3.* 1999.
2. **Alba, E.** Parallel evolutionary algorithms can achieve super-linear performance. *Information Processing Letters, Elsevier. Vol 82, Pgs 7-13.* 2002.
3. **Alba, E..** Parallel Metaheuristics. *Wiley-Interscience, ISBN: 0471678066.* 2005.
4. **Alba, E., et al.** MALLBA: A Library of Skeletons for Combinatorial Optimization. *Volume 2400 of LNCS, Pages 927-932, Springer.* 2002.
5. **Alba, E., Nebro, A. and Troya, J.** Heterogeneous Computing and Parallel Genetic Algorithms. *Journal of Parallel and Distributed Computing. Vol 62, Issue 9.* 2002.
6. **Barr, R. and Hickman, B.** Reporting Computational Experiments with Parallel Algorithms: Issues,Measures, and Experts Opinions. *Invited paper, ORSA Journal on Computing, vol. 5, Pg. 2-18.* 1993.
7. **Grefenstette, J.** *Parallel Adaptative Algorithms for function optimization.* Technical Report CS-81-19, 1981.
8. **Holland, J.** Adaptation in Natural and Artificial Systems. *University of Michigan Press, Ann Arbor.* 1975.
9. **Manderick, B. and Spiessens, P.** Fine-grained parallel genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithm. Pgs 428-433.* 1989.
10. **Michalewicz, M.** *Genetic Algorithms Data Structures Evolution Programs.* (3rd revised edn), Springer, Berlin 1996.
11. **Tanese, R.** Distributed Genetic Algorithms. *Proceedings of the Third International Conference on Genetic Algorithms. Pgs 434-439.* 1989.
12. **Tanese, R.** Parallel Genetic Algorithms for a Hypercube. *In J.J. Grefenstette, editor, Proceedings of the Second International Conference on Genetic Algorithms.* 1987.
13. **Tomassini, M.** Spatially Structured Evolutionary Algorithms. *Springer-Verlag New York, Inc., ISBN:3540241930 .* 2005.