Algoritmos genéticos paralelos

Antonio la Torre de la Fuente

16 de septiembre de 2005

Resumen

Este trabajo pretende presentar de una manera general el estado actual de los algoritmos genéticos paralelos, su evolución a partir de los algoritmos genéticos tradicionales, establecer una clasificación no exhaustiva de los mismos, así como esbozar los principales problemas y retos que plantea el uso de estas técnicas. A lo largo del trabajo veremos que estos algoritmos son capaces de encontrar soluciones de una calidad similar a la obtenida por los algoritmos en su versión secuencial y, algo fundamental, en un tiempo considerablemente menor, llegando en ocasiones a obtener un aumento de rendimiento superlineal.

Índice general

Capítulo 1

Introducción

Los algoritmos genéticos son un subconjunto de unas técnicas heurísticas conocidas como técnicas evolutivas. Estos algoritmos están basados en la teoría de la evolución de Darwin, imitando el comportamiento de los mecanismos de reproducción y selección dentro de una especie. Su gran versatilidad para resolver problemas de muy diferentes campos han hecho que hayan adquirido una gran popularidad en los últimos años.

1.1. Orígenes

Como ya se ha dicho, los algoritmos genéticos se basan en los mecanismos de selección que utiliza la naturaleza, según los cuales los individuos más aptos de una población son los que sobreviven y los que, por tanto, servirán de base para generaciones posteriores.

Un investigador de la Universidad de Michigan, John Holland, se interesó por estos mecanismos, asombrado por la capacidad de la naturaleza de perfeccionar a sus organismos. A principio de los años 60 comenzó a desarrollar estas ideas y a adaptarlas para la resolución de problemas computacionales. Dos fueron los objetivos sobre los que centró su investigación:

- imitar los procesos adaptativos de los sistemas naturales
- diseñar sistemas informáticos capaces de resolver problemas usando los mecanismos más importantes presentes en la naturaleza

15 años más tarde, un ingeniero industrial llamado David Golberg conoció a John Holland y se convirtió en alumno suyo. Éste intentó aplicar los algoritmos genéticos a problemas industriales, aunque Holland intentó disuadirle por pensar que eran demasiado complicados de resolver. Finalmente, consiguió aplicarlos

de manera satisfactoria y esto, unido a las aplicaciones posteriores en diversos campos que de ellos se hicieron, propició que los algoritmos genéticos adquirieran la popularidad de la que aún hoy gozan.

1.2. Fundamentos de los algoritmos genéticos

Los algoritmos genéticos son métodos de resolución de problemas de búsqueda y optimización que hacen uso de los mismos métodos de los que se sirve la naturaleza, según la teoría de Darwin, para la evolución biológica: la reproducción, la mutación y la selección de los mejores individuos de una población.

En un algoritmo genético, se parametrizará el problema con una serie de variables ($x_i ... x_n$) que serán codificadas en un cromosoma. El objetivo será maximizar (o minimizar) este conjnto de variables. Para ello, se crea una población inicial de individuos que son potenciales soluciones del problema. Normalmente, esta población inicial se obtendrá de manera aleatoria, aunque bien podría utilizarse otros métodos basados en el conocimiento que tengamos del problema para facilitar la convergencia del algoritmo. Sobre la población de posibles soluciones, el algoritmo genético aplicará los operadores de cruce (reproducción), mutación y selección y, tras cada iteración, evaluará los individuos de la nueva población generada a partir de la anterior. Los mejores individuos serán la base de la nueva generación, que se obtendrá con la aplicación de los operadores antes citados.

La gran ventaja de los algoritmos genéticos es que no tienen un campo de aplicación específico. Basta con definir una representación del problema y una función de evaluación de los individuos para poder resolver casi cualquier tipo de problemas. El problema que se deducirá rápidamente es que, al tratarse de una técnica heurística, el tiempo de ejecución puede convertirse en un obstáculo insalvable si el espacio de búsqueda de soluciones es muy amplio. Además, si no se tiene cuidado al elegir el factor de mutación, podría darse el caso de que el algoritmo convergiera hacia un óptimo local, sin llegar nunca al óptimo global del problema. Por eso, garantizar la diversidad de los individuos es fundamental al usar este tipo de técnicas. Es por ello, tanto por la velocidad de ejecución como por la diversidad de los individuos, por lo que surgen aproximaciones paralelas de los algoritmos genéticos.

1.3. Algoritmos genéticos paralelos

Los algoritmos genéticos paralelos (o, para abreviar a partir de ahora, AGP) surgen ante la necesidad de cómputo requerida por problemas de extrema complejidad, cuyo tiempo de ejecución utilizando los tradicionales algoritmos genéticos secuenciales es prohibitivo. Es por eso que se buscó la manera de poder adaptar este tipo de heurísticas a distintas configuraciones de cómputo paralelo, lo que dio lugar a tres grandes modelos de algoritmos genéticos paralelos [9]: (1) algoritmos maestro-esclavo, (2) algoritmos de grano fino y (3) algoritmos de grano grueso. Es en estos últimos, también llamados algoritmos genéticos distribuidos, en los que se centrará este trabajo y sobre los que se han realizado un mayor número de estudios.

Algunos de estos estudios presentan, además, conclusiones bastante interesantes. La primera de ellas es la influencia de tener varias subpoblaciones en lugar de una sola a la hora de garantizar la diversidad de los individuos. Una buena política de migración de estos individuos puede ayudar significativamente a que el algoritmo no se quede atrapado en máximos (o mínimos) locales, y evitar así una convergencia prematura. La segunda conclusión tiene que ver con el *speed-up* conseguido mediante el uso de estas técnicas. Algunos estudios proponen que se puede alcanzar incluso la superlinealidad en determinados casos, aunque no se ha conseguido aún comprobar exactamente a qué es debido.

En los capítulos siguientes se profundizará en la clasificación de los algoritmos genéticos paralelos, así como en las cuestiones más importantes a tener en cuenta a la hora de su implementación y sus aportanciones principales con respecto a los algoritmos genéticos secuenciales.

Capítulo 2

Clasificación de los AGP

Antes de iniciar la paralelización de un algoritmo, de cualquier tipo, es imprescindible realizar una primera fase de estudio de qué elementos de los que forman el algoritmo son susceptibles de paralelizarse [4]. En el caso de los algoritmos genéticos es claro que la evaluación de la adecuación de los individuos es una tarea cuya paralelización no afecta al comportamiento del algoritmo. Sin embargo, el operador de selección sí debe ser aplicado de foma global a toda la población si queremos que el comportamiento del algoritmo siga siendo el mismo que el de la versión secuencial. Esta primera aproximación para la paralelización de los algoritmos genéticos dará lugar a un conjunto de algoritmos que recibirán el nombre de algoritmos **maestro-esclavo**.

La otra gran aproximación a la paralelización de algoritmos genéticos consiste en dividir la población inicial en subpoblaciones de mayor o menor tamaño que se comuniquen de alguna manera. Este sistema de paralelización es el que más éxito ha obtenido en comparación con su equivalente secuencial. Además de las mejoras de rendimiento debidas a la subdivisión del problema se ha demostrado que el hecho de considerar subpoblaciones que evolucionan independientemente suele, por lo general, ayudar en el proceso de búsqueda. Este tipo de algoritmos se dice que da lugar a *especies* o *nichos* de individuos separados. Dentro de esta segunda aproximación podemos distinguir entre algoritmos de **grano grueso** y algoritmos de **grano frino**. La diferencia entre ambos es el tamaño de las poblaciones (mayor en los primeros) y la forma en que los individuos interactúan los unos con los otros.

Veamos con más detalle cada una de estas aproximaciones a la paralelización de los algoritmos genéticos.

2.1. Algoritmos maestro-esclavo

Estos algoritmos trabajan con una única población de individuos que será gestionada por el nodo maestro. La evaluación de la adecuación de los individuos y/o la aplicación de los operadores genéticos puede ser realizada por los nodos esclavos. A cada nodo esclavo le corresponderá una parte de la población total, sobre la cual realizará las operaciones antes citadas. Una vez terminado este proceso, devolverán el resultado al nodo maestro, que realizará la selección de individuos.En la figura 2.1 podemos encontrar un pequeño esquema de la arquitectura de este tipo de algoritmos.

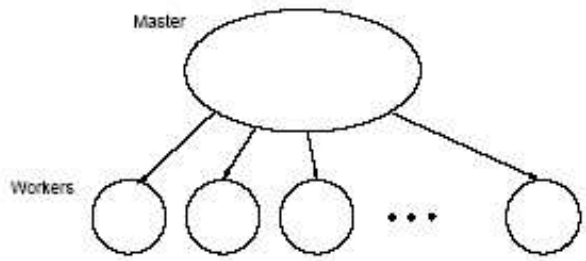

Figura 2.1: Representación esquemática de un AGP con arquitectura maestro-esclavo [9]

Normalmente, la operación que se suele implementar en paralelo es la evaluación de la adecuación de los individuos, porque suele ser la más compleja. Además, este valor es independiente del resto de la población, por lo que su implementación es muy sencilla.

El intercambio de información entre los nodos es sencillo: el nodo maestro envía el subconjunto de inviduos que corresponde a cada nodo, y éstos le devuelven los valores de adecuación para cada uno de ellos. La comunicación puede ser implementada de dos maneras: síncrona o asíncrona. En la primera de ellas, el nodo maestro espera a recibir los valores de adecuación de todos los individuos para generar la siguiente generación. En la segunda, el algoritmo no espera a que los nodos más lentos envíen sus valores de *fitness*. De este modo conseguimos agilizar el proceso, pero el comportamiento no es exactamente el mismo que el de un algoritmo genético secuencial. La implementación síncrona, en cambio, sí mantiene este comportamiento.

Este tipo de algoritmos no especifican nada acerca de la arquitectura subya-

cente [9], por lo que pueden ser implementados sin problemas en computadores tanto de memoria compartida como de memoria distribuida. En el caso de los computadores de memoria compartida, la población podría estar almacenada en memoria, y cada procesador esclavo podría leer directamente los individuos que le hubieran sido asignados. En caso de usar un computador de memoria distribuida, sería el nodo maestro el encargado de asignar y distribuir individuos entre los nodos esclavos y de recoger, una vez evaluada, la adecuación de cada uno de ellos.

Varios estudios han sido realizados para evaluar el rendimiento de este tipo de algoritmos. Algunos resultados interesantes pueden verse, por ejemplo, en el que realizaron Abramson, Mills y Perkins [1], que usando dos computadores de memoria compartida observaron un incremento de prestaciones razonablemente alto en comparación con los algoritmos genéticos tradicionales usando hasta 16 procesadores. A partir de ese número, el comportamiento se degradaba, dato éste que justificaban por el aumento del coste de las comunicaciones.

2.2. Algoritmos de grano fino

Este tipo de algoritmos han sido diseñados para ser implementados usando computadores masivamente paralelos. Ahora, la población se encuentra dividida espacialmente entre los distintos procesadores e, idealmente, cada procesador debería albergar un único individuo. El cruce y la selección de individuos se hará entre individuos que pertenezcan a un mismo *vecindario*, formado por un conjunto de individuos adyacentes según la representación espacial antes citada. Sin embargo, se permite el solapamiento entre *vecindarios* para propiciar la interacción, aunque leve, entre todos los individuos de la población. La figura 2.2 recoge un esquema de esta aproximación.

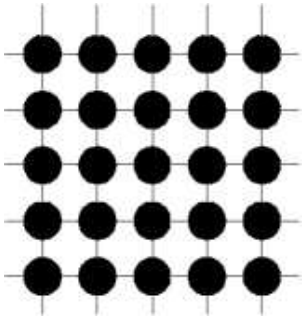

Figura 2.2: Representación esquemática de un AGP de grano fino [9]

Se han llevado a cabo algunos estudios para determinar si el tamaño de los

vecindarios influía o no en la presión de selección de los individuos. Sarma y De Jong [18] encontraron que la relación entre el radio de los *vecindarios* y el radio de la población completa influía en este aspecto, y cuantificaron el tiempo que tardaba en propagarse una solución buena al resto de la población con diferentes tamaños para los *vecindarios*.

Otros estudios han intentado determinar qué tipo de representaciones espaciales daban mejores resultados. Schwehm [19] experimentó con varias representaciones: un anillo, un toro, un cubo 16x8x8 y un hipercubo 4x4x4x4x4. El problema con el que trabajó fue el particionamiento de grafos, y encontró que el algoritmo que usó la estructura de toro convergió más rápidamente que el resto, aunque no da detalles sobre la calidad de las soluciones encontradas.

2.3. Algoritmos de grano grueso

Las características más importantes de estos algoritmos son el uso de múltiples poblaciones y la migración de individuos entre ellas. Dado que cada una de las poblaciones evolucionan independientemente, el ratio de migración será muy importante de cara a obtener resultados satisfactorios. Se puede observar una posible solución usando este tipo de algoritmos en la figura 2.3.

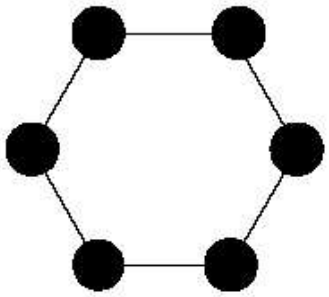

Figura 2.3: Representación esquemática de un AGP con arquitectura maestro-esclavo [9]

Grosso, en 1985 [11], realizó el que se puede considerar como el primer estudio de los algoritmos genéticos paralelos con múltiples poblaciones. Algunos de sus resultados, lejos de aclarar si el uso de este tipo de algoritmos es recomendable en sustitución de los tradicionales algoritmos genéticos secuenciales, arrojan nuevas incógnitas. Por ejemplo, observó que los individuos de las poblaciones,

cuando éstas están aisladas, convergen más rápidamente hacia valores óptimos, pero que estos valores no son tan buenos como los obtenidos con los algoritmos secuenciales. Con valores bajos en el ratio de migraciones los cambios apenas son apreciables. Una vez alcanzado un determinado ratio de migración, que dependerá del problema y de la topología empleada, las soluciones encontradas por estos algoritmos igualan en calidad a las encontradas por el algoritmo secuencial.

Como éste y otros estudios demuestran, el buen funcionamiento de estos algoritmos depende en gran medida de varios factores, de entre los que destacan la frecuencia con la que se realicen las migraciones y la topología de comunicación entre las subpoblaciones que se use. Estos dos factores los veremos con mayor detalle a continuación.

2.3.1. Influencia de las migraciones

Hay que tener en cuenta varios factores en la migración de individuos entre poblaciones. El primero de ellos es la frecuencia con la que éstas se llevarán a cabo. La mayor parte de las implementaciones existentes programan las migraciones a intervalos fijos o, dicho de otra forma, de una forma síncrona. Por otro lado, se puede introducir asincronía en la política de migraciones haciendo que éstas sean efectuadas sólo cuando se produzca un evento. En el estudio descrito en la tesis de Grosso [11], éste programó las migraciones para que fueran efectuadas únicamente cuando la población estuviera cerca de converger. Braun [7] hizo algo parecido, aunque en este caso las migraciones se realizaban una vez la población había convergido completamente con el propósito de restaurar la diversidad. Éste es un aspecto muy importante, ya que una migración prematura de individuos puede entrañar serios problemas, ya que la calidad de los individuos enviados podría no aportar ninguna mejora en el resto de poblaciones, con lo que estaríamos desperdiciando valiosos y costosos recursos de red. Sin embargo, una migración muy tardía puede afectar también negativamente y aumentar considerablemente el tiempo de convergencia al óptimo global o, incluso, propiciar que ésta no se produzca. Por tanto, encontrar el momento adecuado para la migración de los individuos es un aspecto muy a tener en cuenta a la hora de diseñar un algoritmo de este tipo.

Otro aspecto muy a tener en cuenta es qué individuo se envía en las migraciones. Pettey, Leuze y Grefenstette [17] proponen enviar el mejor individuo de cada población a todas sus adyacentes. Los resultados experimentales que obtuvieron desvelaron que los resultados así obtenidos eran de una calidad similar a los obtenidos por algoritmos genéticos con una única población. Una aproximación diferente es la que desarrollaron Marin, Trelles-Salazar y Sandoval [15]. Propusie-

ron que cada subpoblación, transcurridas las generaciones necesarias para realizar la migración, enviaran su mejor individuo a un nodo maestro, que se encargaría de elegir a los mejores individuos entre todos los recibidos y de reenviarlos de nuevo a todas las poblaciones. Con un número pequeño de nodos (alrededor de seis) obtuvieron resultados que propiciaban *speed-ups* casi lineales en los tiempos de ejecución.

2.3.2. Influencia de la topología de comunicación

Aunque muchas veces no se tenga demasiado en cuenta, éste es un factor muy importante a la hora de implementar un algoritmo genético paralelo, ya que determina la velocidad con la que las buenas soluciones encontradas se propagan hacia el resto de poblaciones. Si el grado de conectividad es alto o e diámetro de la red es pequeño (o ambas cosas) las soluciones buenas se expandirán rápidamente a través de todas las poblaciones, favoreciendo así la evolución hacia el óptimo. En caso contrario, las poblaciones estarán más aisladas las unas de las otras, por lo que evolucionarán mucho más despacio, y llevará más tiempo la incorporación de los genes potencialmente buenos a las nueva soluciones.

Por otro lado, es necesario tener en cuenta que la densidad de conexiones puede influir negativamente en el algoritmo, ya que supone una sobrecarga importante en el tráfico de la red. Por eso, es muy importante elegir una buena topología para obtener el máximo rendimiento del algoritmo.

Por lo general, se suelen utilizar topologías estáticas, es decir, topologías predefinidas que no cambian según avanza la ejecución del algoritmo. Las más utilizadas suelen ser las topologías de anillo y de hipercubo de cuatro dimensiones. Sin embargo, algunas implementaciones proponen que las conexiones entre poblaciones no estén definidas desde el principio, sino que se establezcan sobre la marcha cuando alguna población cumpla algún criterio, como la diversidad de la población [16] o la distancia entre los genotipos de las dos poblaciones [14]. Esta medidad de distancia se puede simplificar calculando únicamente la distancia entre los dos mejores individuos de ambas poblaciones.

2.4. Algoritmos Híbridos

Este último tipo de algoritmos son, quizá, los más complejos de los vistos hasta el momento, dado que combinan dos de los algoritmos anteriores una estructura de dos niveles. En el nivel superior suele haber siempre un algoritmos de grano grueso (multipoblación). En el nivel inferior se han probado los tres tipos

de algoritmos vistos: maestro-esclavo, de grano fino y de grano grueso. A esta estructuración en dos niveles se le denomina jerarquía y, por eso, muchas veces a estos algoritmos se les denomina algoritmos genéticos jerárquicos (del inglés *Hierarchical Genetic Algorithms*).

La primera posibilidad es combinar un algoritmo multipoblación en el nivel superior con un algoritmo de grano fino en el nivel inferior (ver figura 2.4). Existen varias implementaciones de este tipo de algoritmos. Gruau [12] usó un algoritmo genético multipoblación cuyas poblaciones estaban conectadas en forma toroidal de dos dimensiones. Cada una de las poblaciones fue organizada siguiendo una estructura de grano fino, en este caso una cuadrícula de dos dimensiones. Otros autores, como Lin, Goodman y Punch [13], conectaron las poblaciones del nivel superior usando una topología de anillo, y conservaron la distribución en cuadrícula de dos dimensiones para el nivel inferior. Compararon esta implementación con varios tipos de algoritmos genéticos, para lo cual usaron un problema de planificación. Entre los algoritmos usados para la comparativa se encuentran algunas implementaciones paralelas de las vistas anteriormente, así como algoritmos genéticos secuenciales. Los resultados que obtuvieron con este algoritmo híbrido fueron mejores que los obtenidos por el resto de aproximaciones.

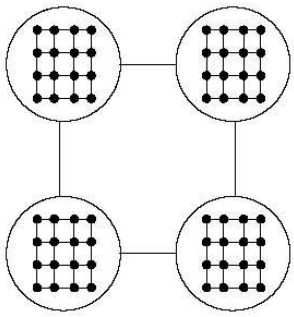

Figura 2.4: Representación esquemática de un algoritmo híbrido de tipo multipoblación en el nivel superior y de grano fino en el nivel inferior [9]

Como segunda alternativa para implementar algoritmos híbridos se puede usar un algoritmo multipoblación en el nivel superior y un algoritmo maestro-esclavo en cada una de las poblaciones (ver figura 2.5). Bianchini y Brown [6] experimentaron con estos algoritmos, consiguiendo soluciones de la misma calidad que las obtenidas con algoritmos paralelos tradicionales (multipoblación y maestro-esclavo) en menos tiempo.

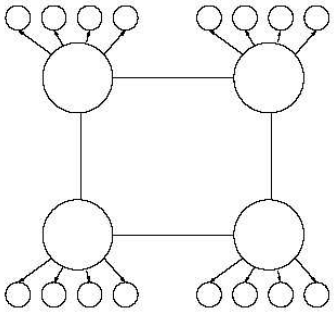

Figura 2.5: Representación esquemática de un algoritmo híbrido de tipo multipoblación en el nivel superior y maestro-esclavo en el nivel inferior [9]

La tercera opción de hibridación es usar algoritmos multipoblación en ambos niveles (superior e inferior). Usando una topología altamente conectada en el nivel inferior y una frecuencia de migración elevada se consigue una mejor distribución de las soluciones óptimas a través de las poblaciones (ver figura 2.6). En el nivel superior se suele optar por frecuencias de migraciones bajas, lo cual hace que la complejidad no crezca demasiado, siendo aproximadamente igual a la de un algoritmo paralelo con multipoblaciones tradicional [10].

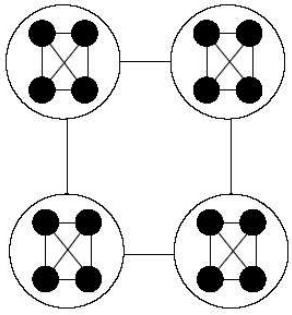

Figura 2.6: Representación esquemática de un algoritmo híbrido de tipo multipoblación tanto en el nivel superior como en el nivel inferior [9]

Capítulo 3

Sincronismo y asincronismo en los AGP

Cuando se pretende implementar un algoritmo genético paralelo, del tipo que sea, una duda surge inmediatamente: ¿deben las migraciones efectuarse de forma síncrona o asíncrona? En el caso de las implementaciones síncronas, cada nodo del sistema distribuido espera, llegado el momento de las migraciones, a recibir todos los individuos de sus nodos adyacentes. Esto no es así en el caso de las implementaciones asíncronas, en las que todo individuo recibido será insertado en la población nada más llegar, independientemente del resto. Esto hace que las poblaciones evolucionen de distinta manera, según la velocidad de cómputo de cada nodo, y que se difumine el concepto de generación.

Un experimento interesante a este respecto es el efectuado por Alba y Troya [3], para el que usaron varias implementaciones paralelas de algoritmos genéticos, tanto síncronas como asíncronas, distintas frecuencias de migraciones y número de nodos. Probaron los algoritmos con varios tests de diferente complejidad. Los resultados obtenidos pueden verse con más detalle en [3], aunque podemos avanzar que, por regla general, las implementaciones asíncronas obtenían soluciones de igual calidad que las síncronas, evaluando un número similar de individuos, pero en un tiempo inferior. Además, observaron que esta mejora dependía también de la complejidad del problema a resolver y del número de nodos utilizados, ya que si el problema resultaba demasiado simple, o el número de nodos era excesivo, la disminución del tiempo para encontrar una solución óptima era despreciable.

Un hallazgo curioso relacionado con el uso de una política de comunicaciones asíncrona es el hecho de que estos algoritmos presentan una mayor resistencia al uso de frecuencias de migración erróneas. Es decir, si para un problema se eligen unas frecuencias inapropiadas, tanto si éstas son excesivas como si son insuficien-

tes, la versión síncrona del algoritmo tendrá problemas para encontrar soluciones en un tiempo razonable, llegando en algunos casos a no ser capaces de hacerlo. Por el contrario, parece que esto afecta en menor medida a los algoritmos asíncronos.

En resumen, la calidad de las soluciones encontradas no se ve afectada por el uso de algoritmos síncronos o asíncronos, aunque, en general, los tiempos de búsqueda serán inferiores en el caso de estos últimos.

Capítulo 4

Superlinealidad en los AGP

Desde un principio quedó claro que el motivo principal por el que se inició la investigación en el campo de los algoritmos genéticos paralelos es la necesidad de cómputo requerida por algunos problemas de gran complejidad que difícilmente podrían ser resueltos con heurísticas tradicionales.

Llegados a este punto se hace necesario algún método para determinar el beneficio obtenido al usar estas técnicas frente a las aproximaciones tradicionales. Podría darse el caso de que el beneficio obtenido no compensase la complejidad añadida al distribuir el algoritmo genético, además del tiempo necesario para estudiar el problema y establecer los parámetros adecuados para el algoritmo (tamaño de poblaciones o frecuencia de las migraciones, por ejemplo). Es por eso que en los últimos años se haya empezado a estudiar este aspecto.

La mayoría de los resultados actuales se centran en dos tipos de medidas para discernir si el comportamiento del algoritmo genético paralelo es o no superlineal con respecto al algoritmo secuencial equivalente. La primera es el número de evaluaciones necesarias para encontrar el valor óptimo. La segunda es el tiempo necesario para encontrar este valor. Ambas medidas, junto con otras más complejas, son en principio válidas, aunque una puede ser más apropiada que la otra según el entorno de experimentación con el que se cuente.

La medida del número de evaluaciones puede ser más apropiada si se cuenta con un conjunto de máquinas heterogéneas, con distintas configuraciones y capacidades de cálculo, ya que las conclusiones así obtenidas pueden ser más fácilmente extrapolables a configuraciones distintas (un aumento de la potencia de cómputo de los equipos utilizados afectará sensiblemente al tiempo de ejecución pero no al número de evaluaciones necesarias para encontrar el valor óptimo). Por otro lado, la medida de tiempo de ejecución es muy útil si el entorno de experi-

mentación es lo suficientemente homogéneo, ya que proporciona un valor que nos permite observar muy directamente la mejora obtenida con la paralelización del algoritmo.

Teóricamente, los algoritmos genéticos paralelos, y otros métodos de búsqueda estocásticos, pueden presentar un *speedup* (cociente entre el tiempo de cómputo de un algoritmo genético secuencial y su versión paralela) superlineal. De hecho, Shonkwiler en [20] afirma que existe una relación exponencial entre el *speedup* y el número de procesadores. Sin embargo, no siempre es posible conseguir la superlinealidad al implementar un algoritmo genético paralelo, como veremos a continuación.

Algunos estudios, como el presentado por Cantú-Paz en [8], sostienen que el motivo por el que se obtiene un *speedup* superlineal en estos algoritmos es porque la versión paralela realiza menos trabajo que la versión secuencial. La justificación que ofrece es que la presión ejercida por las migraciones hace que el algoritmo converja más rápidamente hacia el valor óptimo. También Alba en [2] da una explicación similar, en cuanto a que las versiones paralelas de los algoritmos genéticos realizan menor trabajo que sus versiones secuenciales, aunque en este caso la explicación ofrecida tiene que ver con el espacio de soluciones que exploran ambas versiones.

Otros autores, como Belding en [5], afirman que el motivo de alcanzar *speedups* superlineales tiene más que ver con las configuraciones hardware utilizadas. Belding justificó los resultados obtenidos en el estudio antes citado diciendo que al pasar de un algoritmo secuencial a uno paralelo, las subpoblaciones utilizadas podían ser almacenadas completamente en las cachés de los equipos, lo cual incidía notablemente en el aumento de rendimiento observado. También Alba en [2] afirma algo parecido, aunque sin entrar en tanto detalle. Entre las diferentes razones que propone para los aumentos de rendimiento observados en sus experimentos se encuentra la configuración hardware de los equipos utilizados.

Un último factor a tener en cuenta que varios autores citan como importante es la implementación de los algoritmos, ya que distintas implementaciones de un algoritmo pueden ser más o menos eficientes. En algunos casos, dependiendo, por ejemplo, de las estructuras de datos utilizadas, las versiones paralelas de un algoritmo pueden resultar mucho más eficientes.

En resumen, por el momento no es posible afirmar si un algoritmo va a presentar un *speedup* superlineal o no, dado que, aunque teóricamente esto es posible, son muchos los factores a tener en cuenta, por lo que no es posible, por lo menos

apriori, determinar cómo se comportará el algoritmo aunque, también es cierto, por regla general el aumento de prestaciones obtenido suele ser notable.

Capítulo 5

Conclusiones

Los algoritmos genéticos paralelos surgen ante las necesidades de cómputo cada vez más exigentes de los problemas que pueden ser resueltos con ellos. Ante esta situación, los algoritmos tradicionales (secuenciales) se muestran insuficientes, y su tiempo de ejecución se vuelve prohibitivo.

A lo largo de este trabajo hemos tratado de ofrecer una visión general del estado actual de este tipo de técnicas e intentado justificar por qué su uso es válido para resolver los problemas a los que antes se hizo referencia. Hemos visto los distintos tipos de algoritmos genéticos paralelos existentes, la utilidad de cada una de ellas y sus características más importantes. Tres son los más importantes: los algoritmos genéticos con arquitectura maestro-esclavo, los algoritmos genéticos de grano fino y los algoritmos genéticos con multipoblaciones.

Los primeros vimos que eran la adaptación directa de los algoritmos genéticos secuenciales equivalentes, y que su funcionamiento era exactamente el mismo que el de éstos, por lo que los resultados obtenidos era de esperar que fueran idénticos.

Los segundos son un caso especial, que normalmente requieren de arquitecturas hardware muy específicas para su implantación y que, de momento, tienen un interés más teórico que práctico.

Por último, los terceros, los algoritmos genéticos con multipoblaciones, son los que centran la atención de los investigadores hoy día, y los que más cuestiones plantean, ya que en su funcionamiento influyen numerosos factores y no son una conversión tan directa de los algoritmos secuenciales como los primeros. De hecho, el trabajar cada uno con una subpoblación de tamaño más reducido y el intercambio de información sobre los individuos mediante las migraciones hacen de estos algoritmos interesantes casos de estudio y, por lo que se ha visto en los

experimentos realizados, técnicas muy potentes para la resolución de un amplio abanico de problemas.

Por último, dado que el objetivo de estos algoritmos era facilitar la resolución de problemas con grandes necesidades de cómputo, parecía lógico indagar sobre el aumento de rendimiento observado al utilizar estas técnicas. Según los estudios consultados, parece ser que teóricamente es posible obtener un *speedup* superlineal, aunque esto depende de muchos factores, por lo que no es posible determinar apriori si en cada caso concreto se obtendrá un aumento de rendimiento de este calibre.

Finalmente, y aunque no siempre se alcance un *speedup* superlineal, los algoritmos genéticos paralelos han demostrado que son capaces de obtener soluciones de una calidad similar a la de las obtenidas por los algoritmos genéticos tradicionales en un tiempo sensiblemente inferior, por lo que su aplicación a problemas reales de gran complejidad parece apropiada.

Bibliografía

[1] D. Abramson, G. Mills, and S. Perkins. Parallelisation of a genetic algorithm for the computation of efficient train schedules. In *Proceedings of 1993 Parallel Computing and Transputers Conference*, pages 139–149, 1993.

[2] E. Alba. Parallel evolutionary algorithms can achieve super-linear performance. *Information Processing Letters*, (82):7–13, 2002.

[3] E. Alba and José M. Troya. Analyzing synchronous and asynchronous parallel distributed genetic algorithms. *Future Generation Computer Systems*, (17):451–465, 2001.

[4] Enrique Alba. *Análisis y Diseño de Algoritmos Genéticos Paralelos Distribuidos*. PhD thesis, Universidad de Málaga, 1999.

[5] T.C. Belding. The distributed genetic algorithm revisited. *Eschelman*, pages 114–121, 1995.

[6] R. Bianchini and C.M. Brown. Parallel genetic algorithms on distributed-memory architectures. *Transputer Research and Applications*, (6):67–82, 1993.

[7] H. C. Braun. On solving travelling salesman problems by genetic algorithms. In *Proceedings of the First Parallel Problem Solving from Nature Conference*, pages 129–133, Springer-Verlag (Berlin), 1990.

[8] E. Cantú-Paz. Migration policies, selection pressure, and parallel evolutionary algorithms. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, San Francisco, CA, 1999.

[9] Erick Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles, Réseaux et Systems Répartis*, 10(2):141–171, 1998.

[10] D.E. Goldberg. *Personal Communication*, 1996.

[11] P. B. Grosso. *Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model*. PhD thesis, Unpublished Doctoral Dissertation, The University of Michigan, 1985.

[12] F. Gruau. *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm*. PhD thesis, L'Université Claude Bernard-Lyon I, 1994.

[13] S. Lin, E. Goodman, and W. Punch. Investigating parallel genetic algorithms on job shop scheduling problem. In *Proceedings of the Sixth International Conference on Evolutionary Programming*, pages 383–393, Springer Verlag (Berlin), 1997.

[14] S. C. Lin, W. Punch, and E. Goodman. Coarse-grain parallel genetic algorithms: Categorization and new approach. In *Proceedings of the Sixth IEEE Symposium on Parallel and Distributed Processing*, IEEE Computer Society Press (Los Alamitos, CA), 1994.

[15] F. J. Marin, O. Trelles-Salazar, and F. Sandoval. Genetic algorithms on lan-message passing architectures using pvm: Application to the routing problem. In *Proceedings of the Thirth Parallel Problem Solving from Nature Conference*, pages 534–543, Springer-Verlag (Berlin), 1994.

[16] M. Munetomo, Y. Takai, and Y. Sato. An efficient migration scheme for subpopulation-based asynchronously parallel genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, page 649, Morgan Kaufmann (San Mateo, CA), 1993.

[17] C. B. Pettey, M. R. Leuze, and J. J. Grefenstette. A parallel genetic algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 155–161, Lawrence Erlbaum Associates (Hillsdale, NJ), 1987.

[18] J. Sarma and K. D. Jong. An analysis of the effects of neighborhood size and shape on local selection algorithms. In *Proceedings of the Fourth Parallel Problem Solving from Nature Conference*, pages 236–244, Springer-Verlag (Berlin), 1996.

[19] M. Schwehm. Implementation of genetic algorithms on various interconnection networks. In *Proceedings of Parallel Computing and Transputers Application Conference*, pages 195–203, IOS Press (Amsterdam), 1992.

[20] R. Shonkwiler. Parallel genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 199–205, Morgan Kaufmann (San Mateo, CA), 1993.