Heterogeneidad, WAN y Nuevas Aplicaciones de los Algoritmos Evolutivos Paralelos

Enrique Alba, Antonio J. Nebro y Francisco Chicano

Resumen— **Este trabajo pretende cubrir un vacío en el campo de las nuevas tendencias de optimización con algoritmos evolutivos paralelos. En primer lugar presentamos una descripción unificada de estos algoritmos, y después pasamos a discutir los principales desafíos a los que se enfrentan dichos algoritmos en relación a las nuevas tendencias, es decir, los mecanismos básicos para desarrollar aplicaciones sobre Internet. Para trabajar en el nuevo ámbito de computación mundial necesitamos proponer tareas que hagan uso de máquinas heterogéneas, que están separadas por redes de área extensa (WAN), con la finalidad de resolver problemas en dominios de amplio interés multidisciplinar. En este artículo identificaremos las dificultades técnicas para llevar a cabo algoritmos paralelos en este entorno, propondremos soluciones concretas, y discutiremos los resultados que estas soluciones están arrojando sobre problemas de muy distinto corte. Nuestras conclusiones muestran que el trabajo en sistemas heterogéneos abre un campo de nuevas aplicaciones más eficientes y que el desarrollo de herramientas WAN es una línea de interés real, presentando, para refrendar nuestros resultados, estudios numéricos sobre problemas de telecomunicaciones y bio-informática como punta del iceberg de las tendencias actuales.**

Palabras clave— **Algoritmos Evolutivos Paralelos, Heterogeneidad, Ejecución WAN, Telecomunicaciones, Bio-Informática.**

I. Introducción

EL dominio de trabajo con algoritmos evolutivos paralelos (PEA's) ha experimentado en la última década numerosos avances, tal como ha ocurrido con el resto de ramas de la computación evolutiva. Además de permitir resolver problemas que antes eran irresolubles en un tiempo razonable, el uso de algoritmos paralelos de este tipo ha permitido alcanzar nuevas cotas de eficiencia que eran necesarias para su aplicación a tareas del mundo real.

Los PEA's representan un tipo especial de algoritmo, no una simple versión más rápida de los algoritmos evolutivos de ejecución secuencial existentes [15][4]. El modelo de búsqueda usado en paralelo permite disminuir la probabilidad de caer en óptimos locales, obtener soluciones simultáneamente al mismo problema, trabajar en dominios multiobjetivo y abordar con mayor velocidad problemas de elevada epistasis o multimodalidad.

Los objetivos perseguidos en este trabajo son múltiples. En primer lugar se pretenden abordar algunas cuestiones importantes en relación al uso de PEA's en las redes de comunicación actuales. Además, ofreceremos alternativas de solución para varios tipos de problemas que se presentan en el diseño y uso de PEA's modernos, tales como su ejecución en sistemas heterogéneos y el uso de redes LAN de alta velocidad y su extensión a red WAN. Finalmente, identificaremos dos dominios de aplicación de interés europeo e internacional, tales como las aplicaciones en telecomunicaciones y en bioinformática.

Nuestra intención final no es únicamente identificar y dar propuestas para estas cuestiones clave, sino que llegaremos hasta el final de la cadena proponiendo algoritmos y resultados numéricos que nos permitan evaluar la viabilidad de algunas de las soluciones propuestas. Por supuesto, existen nuevos retos en el campo de los PEA's que quedan fuera del ámbito de este artículo, principlamente debido a problemas de espacio; nos referimos, por ejemplo, a la relación de los PEA's con aplicaciones tradicionales de inteligencia artificial, diseño de agentes web, aplicaciones industriales de problemas de optimización tradicionales, etc.

Nuestras conclusiones muestran que la ejecución heterogénea puede resultar muy efectiva, incluso en relación a la ejecución en plataformas homogéneas modernas, así como el hecho de que el uso de redes WAN puede representar un salto cualitativo en cuanto a la capacidad de resolver problemas muy duros actualmente. Adicionalmente, concluimos que la aplicación de PEA's a los nuevos dominios de interés mundial es bastante efectiva; para demostrarlo presentamos resultados de algoritmos paralelos que resuelven dos problemas típicos de las comunicaciones, así como discutimos algunas líneas de actuación en bio-informática, tanto de algoritmos secuenciales híbridos como de sus posibles contrapartidas paralelas.

Este trabajo está organizado de la siguiente forma. En la Sección II discutimos las bases de una descripción única para los PEA's existentes que permitan guiar el diseño de nuevos algoritmos. La Sección III discute primero algunos problemas comunes al trabajo con PEA's, y después presenta resultados de un algoritmo heterogéneo hecho en Java sobre varios problemas. En la Sección IV identificamos las características que debe cumplir un algoritmo paralelo en este campo, así como hacemos propuestas de solución a los problemas relacionados. Finalmente, la Sección V discute varias aplicaciones de interés de los PEA's en dos dominios de relevancia actual. Al final del trabajo aportamos un resumen con nuestras conclusiones, así como delineamos las líneas de trabajo futuro a la luz de los resultados obtenidos (Sección VI).

Departamento de Lenguajes y Ciencias de la Computación, E.T.S.I. Informática, Campus Teatinos, 29071 Málaga (España) eat@lcc.uma.es antonio@lcc.uma.es jfrancis@terra.es

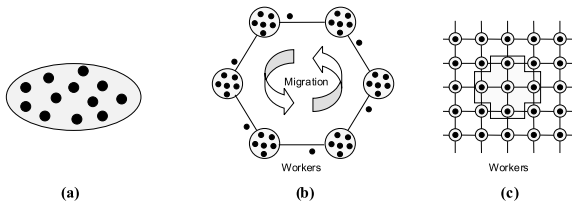

Fig. 1. Un EA en panmixia (a), y dos EAs estructurados: un modelo distribuido (b) y uno celular (c).

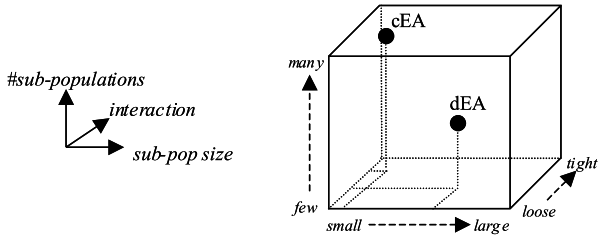

Fig. 2. Cubo de Algoritmos Evolutivos Estructurados.

II. Algoritmos Evolutivos Paralelos

Los últimos años han visto la aparición de algunas técnicas y modelos para unificar el campo de trabajo en algoritmos evolutivos paralelos (PEA's) [6][27]. Esto ha sido necesario debido a un excesivo enfoque en la aplicación de los algoritmos en lugar de en su diseño. Intentaremos dar ahora una descripción unificada que permita trasvasar conocimientos entre distintos modelos y que ayude al lector a comprenderlos.

Un EA simple utiliza una única población sobre la que aplica los operadores de selección y de variación con una filosofía de mejora iterativa [8]. A este tipo de algoritmo de población única se le denomina *en panmixia* (Figura 1a). En estos modelos, cualquier individuo (punto negro en la figura) es una pareja potencial de cualquier otro en el proceso de selección.

Un algoritmo en panmixia puede paralelizarse fácilmente usando un modelo maestro/esclavo, en el cual el maestro ejecuta el algoritmo secuencial tradicional, pero envia a los procesadores esclavos los individuos para su evaluación en paralelo. A este algoritmo se le denomina de *paralelización global* [4], y permite ahorrar tiempo cuando la función de evaluación es costosa y el número de equipos moderado. El algoritmo subyacente continúa siendo el mismo que en el caso secuencial.

El hecho de estructurar la población puede conducir a uno de los dos modelos clásicos que han inspirado la mayoría de implementaciones parelas existentes: un modelo *distribuido* (Figura 1b) o bien un modelo *celular* (Figura 1c). Estos dos modelos suelen encontrarse paralelizados en máquinas MIMD o SIMD, respectivamente, pero nada impide que reciban otro tipo de paralelización o incluso simulación concurrente en un mono-procesador [16].

La Figura 2 recoge gráficamente esta clasificación, mostrando que en muchos casos un algoritmo estructurado puede no ser exactamente de ninguno de los dos tipos. Se puede observar cómo un EA distribuido suele tener una población mucho mayor que un individuo (uno celular normalmente usa un sólo indidivudo por población); además, suele mostrar una evolución desacoplada (en el caso del celular existe una elevada interacción entre vecinos). Finalmente, en el caso distribuido se utilizan pocos subalgoritmos (mientras que en el celular se usan muchos sub-algoritmos).

En el Algoritmo 2 presentamos el pseudo-código de un tipo de PEA muy extendido: un algoritmo genético (GA) usando operadores típicos. Inicialmente se crea una población $P(t=0)$ conteniendo μ individuos aleatoriamente generados. Cada individuo codifica (genotipo) las p variables del problema en forma de un vector binario $B = \{\mathbf{0},\mathbf{1}\}$ o real R, o incluso como árboles sintácticos o cadenas de longitud variable en otro tipo de EA's. La función de evaluación Φ se utiliza para calcular el valor de adecuación de cada individuo. El criterio de parada típico de un EA consiste en encontrar la solución o bien alcanzar un tope máximo de iteraciones del bucle principal. La solución del algoritmo es el mejor individuo encontrado durante todo el proceso de búsqueda.

Algoritmo Genético Paralelo

SUBALGORITMO $\#i$

$t := 0$
inicializar: $P(0) = \{\vec{a}_1(0), \ldots, \vec{a}_\mu(0)\} \in I^\mu$;
evaluar: $P(0) : \{\Phi(\vec{a}_1(0)), \ldots, \Phi(\vec{a}_\mu(0))\}$;
mientras no $\iota(P(t))$ **do** // *Ciclo Reproductor*
  seleccionar: $P'(t) := s_{\Theta_s} P(t)$;
  recombinar: $P''(t) := \oplus_{\Theta_{rec}} P'(t)$;
  mutar: $P'''(t) := m_{\Theta_m} P''(t)$;
  evaluar: $P'''(t) : \{\Phi(\vec{a}'''_1(t)), \ldots, \Phi(\vec{a}'''_\lambda(t))\}$;
  reemplazar: $P(t+1) := r_{\Theta_r}(P'''(t) \cup Q)$;
  <**com. con los vecinos (sínc/asínc)**>
  $t := t + 1$;
fin mientras

Cada operación de selección o de variación (recombinación, mutación, etc.) utiliza un conjunto de parámetros que guían su aplicación (Θ_i en el algoritmo anterior). En particular, ya que se trata de meta-heurísticos de comportamiento estocástico, estos conjuntos suelen contener al menos una probabilidad de aplicación que regula el funcionamiento del operador asociado.

En cada iteración se generan λ individuos, de entre los que se eligen μ para la nueva población que reemplazará a la población actual, completando así un paso evolutivo. La mayoría de algoritmos actuales son elitistas, ya que conservan al menos a la mejor solución entre dos generaciones consecutivas para evitar que se degrade o pierda la mejor solución encontrada hasta el momento. Asimismo, para generar la nueva población se puede considerar (estado estacionario) o no (generacional) al conjunto de individuos de la población anterior [28].

En el caso de un PEA, existen múltiples

sub-algoritmos ejecutando en paralelo un EA de cualquier tipo sobre sub-poblaciones distintas. Los algoritmos paralelos colaborantes pueden o no ser del mismo tipo (lo normal es que lo sean, pero no es obligatorio), y quizás se ejecuten en máquinas de igual arquitectura y sistema operativo o tal vez no (sistema heterogéneo).

Cada sub-algoritmo ejecuta la política de interacción a través de intercambios de información (estadísticas, individuos, etc.) con un conjunto de vecinos (de acuerdo a topologías en anillo, toro, estrella o árbol). Este intercambio puede ser síncrono o no, dependiendo de la aplicación y el sistema de ejecución utilizado. Esta política de interacción puede ser tan elaborada como sea necesaria, por ejemplo teniendo en cuenta la marcha de la evolución o el tipo de enlace o vecinos con el que se desea interaccionar. Véase un estudio de interacción en algoritmos distribuidos en [5] y un estudio sobre algoritmos celulares en [26].

Para terminar esta sección, vamos a identificar los puntos de investigación abiertos en este campo más destacados. Entre ellos podemos mencionar el uso de sistemas heterogéneos de ejecución de PEA's, la aplicación en dominios de interés real, el uso de técnicas de Ingeniería del Software en el diseño del algoritmo, la definición de conceptos teóricos y el estudio de medidas de rendimiento justas y con significado (tales como la ganancia) para PEA's.

Con respecto al uso de heterogeneidad y aplicaciones, diferimos la discusión a las siguientes dos secciones porque las hemos elegido para ser el núcleo del presente trabajo. En relación al uso de técnicas formales de diseño, debemos mencionar que es muy importate el trasvase de conocimiento desde el dominio del modelado OMT y UML a este campo; puede consultarse un estudio de este tipo en [7], así como algunas bibliotecas existentes que incorporan una cierta preocupación por un diseño serio, tales como *Evolving Objects* (`http://geneura.ugr.es/~jmerelo/EO.html`) o numerosos otros sistemas en C++ y Java (visite `http://neo.lcc.uma.es`). Respecto al uso de medidas justas puede consultarse un breve estudio reciente sobre *speedup* [2]. Por último, el lector puede conocer el aspecto teórico de los PEA's a partir de estudios extensos como [10].

III. Heterogeneidad en las Plataformas de Ejecución para PEA's

En los últimos años el diseño y desarrollo de aplicaciones paralelas se centra básicamente en dos tipos de sistemas paralelos: multiprocesadores y sistemas distribuidos. Dentro de estos últimos, las redes locales compuestas por ordenadores homogéneos han sido los sistemas más utilizados debido a su facilidad de construcción y a que ofrecen una relación coste/prestaciones que es imposible de igualar por otro tipo de sistema paralelo equivalente.

El hecho de usar máquinas homogéneas, entendiendo como tales a aquellas que tienen una misma arquitectura y ejecutan el mismo sistema operativo, tiene la ventaja de que permite obviar algunos problemas que se presentan al comunicar procesos que se ejecutan en nodos diferentes de la red. El ejemplo más claro es el de la representación interna de los datos. En una red homogénea los procesos pueden intercambiar datos sin preocuparse por cómo éstos se representan internamente, ya que dicha representación es la misma en todas las máquinas.

Sin embargo, la realidad actual en los laboratorios de investigación muestra que en las redes coexisten máquinas de distinto tipo, es decir, heterogéneas. Es más, con la difusión que ha alcanzado Internet, es factible la utilización conjunta de cientos o miles de ordenadores, y en este contexto la heterogeneidad es la norma. Es por ello que mecanismos de comunicación de bajo nivel comúnmente usados, como es el caso de los sockets, han de verse complementados con el fin de abordar los problemas que la heterogeneidad conlleva. Así, aparecen formatos que permiten representar de forma canónica los datos, como XDR o CDR, que permiten transmitir datos entre máquinas de distinto tipo. Bibliotecas de paso de mensajes como PVM o MPI tienen como uno de sus objetivos principales el abstraer la heterogeneidad al programador. Pero es la aparición de lenguajes multiplataforma, como es el caso de Java, lo que está permitiendo abordar el desarrollo de aplicaciones distribuidas sin que la heterogeneidad sea un problema.

La programación paralela sobre sistemas heterogéneos plantea ciertos interrogantes. Por ejemplo, determinar si el rendimiento del programa paralelo es satisfactorio, o la influencia de la heterogeneidad en el comportamiento de un algoritmo paralelo. Este último punto es particularmente importante en algoritmos estocásticos, como es el caso de los algoritmos evolutivos.

Consideremos un algoritmo evolutivo paralelo en el que existe una subpoblación en cada nodo de una red, de manera que cada subpoblación evoluciona de forma independiente, pero existe un intercambio periódico de individuos entre subpoblaciones. Es evidente que la potencia del procesador de cada nodo (asumiendo que cada nodo es un computador monoprocesador) va a influir en la velocidad relativa en la que evoluciona cada subpoblación y, por tanto, en el resultado numérico [3]. Esto es especialmente cierto en un algoritmo asíncrono. Pero hay otros factores más sutiles en sistemas heterogéneos, como por ejemplo la implementación de la función de generación de números aleatorios, que puede dar lugar a que una subpoblación que evolucione más lentamente que otra se dirija más rápidamente hacia la solución al problema.

Para determinar la influencia de la heterogeneidad en algoritmos evolutivos paralelos hemos llevado a cabo una serie de experimentos sobre varios tipos de computadores, que abarcan desde estaciones de trabajo hasta multiprocesadores, y sobre redes homogéneas y heterogéneas. Concretamente, las pruebas se han realizado sobre los siguientes sistemas:

- Un cluster de cuatro multiprocesadores Digi-

tal AlphaServer (cada uno con cuatro procesadores), interconectados por una red Memory Channel y por Fast Ethernet. Los procesadores son Alpha a 300MHz. El sistema operativo es Digital UNIX 4.0D.
- Un cluster de ocho computadores personales que ejecutan Windows NT 4.0 Workstation. El procesador en todas ellas es un Pentium III a 550 MHz.
- Una máquina SGI Octane, con dos procesadores MIPS R10000 a 250 MHz. El sistema operativo es IRIX 6.5.
- Un computador personal con procesador Pentium III a 300 MHz, que ejecuta Windows NT 4.0 y RedHat Linux 6.2.

Para medir el rendimiento hemos usado la ganancia de velocidad y la fracción serie [19]. La ganancia de velocidad se ha tomado respecto al sistema más rápido [14]. De esta forma se puede garantizar que si la ganancia obtenida es superior a la unidad, el rendimiento del sistema paralelo es superior al de cualquier sistema monoprocesador. La fracción serie es una medida que permite determinar si la pérdida de eficiencia ante un aumento del número de nodos se debe a las partes del programa que no se ejecutan en paralelo. Si la ganancia de velocidad es baja pero la fracción serie permanece constante, se puede afirmar que los resultados son buenos [19].

El algoritmo genético paralelo que hemos elegido para evaluar aquí consta de ocho subpoblaciones, sobre cada una de las cuales se aplica un algoritmo genético de estado estacionario, con selección por torneo binario, recombinación de dos puntos y mutación por inversión binaria. Se han usado ocho subpoblaciones interconectadas de acuerdo a una topología de anillo unidireccional, de forma que cada subpoblación envía una copia de un individuo elegido aleatoriamente, que es incorporado en la siguiente subpoblación si es mejor que su peor individuo. La recepción de individuos es asíncrona. Para analizar el algoritmo se han usado dos problemas, ONEMAX y P-PEAKS. Mediante ONEMAX se trata de maximizar el número de unos en una cadena de 512 bits, mientras que P-PEAKS es un generador de problemas multimodales [17].

TABLA I
GANANCIAS DE VELOCIDAD OBTENIDAS POR ONEMAX EN SISTEMAS HOMOGÉNEOS Y HETEROGÉNEOS

Sistema	*#Procs.*	*Tiempo*	*Ganancia*	*Fr. Serie*
Cluster NT	1	309.9		
Cluster NT	8	38.5	8.051	-0.001
LINUX	1	116.2		
Heterogéneo	8	21.3	5.435	0.067

En la Tabla I se muestran los resultados obtenidos al resolver el problema ONEMAX sobre una red homogénea de ocho computadores personales que ejecutan Windows NT 4.0 Workstation, y los obtenidos con una red heterogénea compuesta, en este orden, por cuatro computadores personales con Windows NT 4.0 Workstation, el sistema SGI Octane, y dos máquinas AlphaServer. Aunque las tres últimas máquinas son multiprocesadores, se ha incluido una única sub-población por máquina. Para medir el rendimiento en la configuración heterogénea se ha tomado como tiempo de referencia el obtenido en el ordenador personal que ejecuta Linux, que ha sido el sistema más rápido. Una discusión detallada de estos resultados, así como los obtenidos con otras configuraciones, se puede consultar en [3].

Se puede observar que con la configuración homogénea se obtiene una ganancia de velocidad ligeramente superlineal (8.051 con ocho procesadores). En cambio, con la configuración heterogénea es de 5.435, que es un buen resultado teniendo en cuenta que la fracción serie es muy pequeña (0.067). No obstante, el tiempo medio absoluto de la configuración heterogénea es notablemente menor que el de la configuración homogénea con igual número de procesadores (21.3 segundos respecto a 38.5).

TABLA II
GANANCIAS DE VELOCIDAD OBTENIDAS POR P-PEAKS EN SISTEMAS HOMOGÉNEOS Y HETEROGÉNEOS

Sistema	*#Procs.*	*Tiempo*	*Ganancia*	*Fr. Serie*
Cluster NT	1	2425.4		
Cluster NT	8	300.0	8.083	-0.00
PC NT	1	2425.4		
Heterogéneo	8	316.8	7.532	-0.008

En la Tabla II se incluyen los resultados obtenidos con el problema P-PEAKS. En este caso, el tiempo de referencia para obtener la ganancia de velocidad en el caso heterogéneo es el mismo que en la configuración homogénea, ya que el ordenador personal Pentium III 550 con Windows NT 4.0 ha sido el más rápido (antes lo fue el PC con Linux). La configuración homogénea ha vuelto a dar una ganancia de velocidad ligeramente superlineal (8.093), mientras que con la configuración heterogénea se ha obtenido una ganancia casi lineal de 7.532. El tiempo medio absoluto es ligeramente mejor con la configuración homogénea (300 segundos) que con la heterogénea (316.8).

Para poder analizar estos resultados hay que tener en cuenta las siguientes consideraciones (ver [3]). En primer lugar, los ordenadores personales ejecutando Windows NT 4.0 han sido los más rápidos con diferencia, lo que indica que, al margen de la frecuencia de sus procesadores sea la mayor, la implementación de la máquina virtual de Java en estos sistemas es muy eficiente. Sin embargo, los resultados globales de estos sistemas en el caso del problema ONEMAX no han sido los mejores, ya que ha sido necesario computar aproximadamente diez veces más iteraciones que en el resto de los sistemas para llegar a la solución. Esto indica que la implementación de la función de números aleatorios en estos sistemas es más deficiente que en el resto, ya que esta es la única diferencia entre los sistemas elegidos.

Por último, los sistemas Digital AlphaServer han sido los más lentos, hasta el punto de que las pruebas con ocho procesadores para resolver el problema

P-PEAKS han requerido un tiempo medio absoluto mayor que el obtenido por un único computador personal con Windows NT. Los motivos están relacionados con el hecho de que los AlphaServer poseen los procesadores más lentos (y más antiguos), y que la implementación de la máquina virtual de Java es muy poco eficiente.

Teniendo en cuenta estos factores, a pesar de incluir dos máquinas Digital Alphaserver los resultados obtenidos con la configuración heterogénea han sido mejores en el caso de ONEMAX en cuanto a tiempo absoluto, aunque no en ganacia de velocidad. Con P-PEAKS la ganancia de velocidad ha sido casi lineal pero el tiempo absoluto ha sido peor, aunque la diferencia en tiempo con la versión homogénea se puede considerar escasa.

Por tanto, se puede concluir que la reutilización de diferentes máquinas en una red heterogénea puede ser más beneficiosa en muchos casos que adquirir una red completa de computadores nuevos e iguales.

IV. Detalles Técnicos para la Ejecución WAN de PEA's

Los problemas actuales se caracterizan por incorporar un alto número de restricciones, ser multiobjetivo, presentar un espacio multimodal, alta epistasis entre sus variables y requerir por tanto grandes cantidades de cómputo. Parece, pues, apropiado el uso de algoritmos paralelos. Tradicionalmente, esto ha supuesto utilizar algún tipo de multiprocesador de memoria compartida o cluster de máquinas interconectados en una LAN para ejecutar en paralelo la técnica de resolución, tal y como se ha comentado en la sección anterior. Sin embargo, los avances en computación con Internet han abierto la posibilidad de utilizar una WAN como sistema de cálculo para algoritmos de optimización.

Si bien existe una experiencia considerable en aplicaciones paralelas sobre entornos LAN, ésta es escasa en entornos WAN. La novedad de este entorno para ser usado en computación, su constante crecimiento y su naturaleza heterogénea representan una considerable complejidad para su estudio y utilización. El funcionamiento de algoritmos en este entorno depende mucho de parámetros dinámicos de la red como, por ejemplo, la latencia, el ancho de banda o el tráfico de comunicación.

Existen varios intentos actualmente de dar soluciones a sistemas de optimización WAN, naturalmente con distintos tipos de enfoque debido a su procedencia. Un ejemplo distinguido podemos encontrarlo en `DREAM`, *The Distributed Resource Evolutionary Algorithm Machine*, un entorno WAN muy ambicioso para optimización, simulación, comunicación y negociación entre agentes usando sistemas evolutivos y bio-inspirados (`http://www.dcs.napier.ac.uk/~benp/dream/dream.htm`).

Un proyecto de cierta similitud de ámbito nacional español está siendo desarrollado en la actualidad. El proyecto `MALLBA` utiliza RedIRIS para conectar entre sí las redes de máquinas de Málaga, La Laguna y Barcelona. Conocer mejor el comportamiento de Internet es uno de los subobjetivos del proyecto `MALLBA`. Para ello, los trabajos en algoritmos tanto exactos como heurísticos se están realizando en tres fases anuales: secuencial, LAN y WAN. Tanto en este proyecto como en cualquier otro que pretenda utilizar una red WAN se ponen de manifiesto numerosos problemas técnicos que deben ser resueltos. A continuación discutimos algunos de ellos.

1. Debe ser posible acceder al estado de la red WAN para después tomar decisiones en los algoritmos paralelos. En nuestro caso estamos desarrollando un protocolo que permita a un proceso actuar como cliente de un servicio de información sobre retrasos en los enlaces WAN, que dependen del tipo de los paquetes intercambiados y del tráfico existente.
2. Debe incorporarse en los algoritmos paralelos un mecanismo que permita conocer el coste de enviar información para ser procesada remotamente en relación con el coste de esperar hasta poder procesarla localmente. Esto hace posible tomar decisiones en el algoritmo sobre la política de migración que debe usarse.
3. Deben proveerse métodos de acceso a la carga de los procesadores que colaboran en la WAN, para detectar estados ociosos y asignar carga de trabajo.
4. Deben establecerse políticas de equilibrado de carga para una colaboración eficiente entre los procesos (por ejemplo en el caso de las colas paralelas de sub-problemas abiertos de un algoritmo de ramificación y acotación).
5. Deben proporcionarse mecanismos de acceso y modificación del estado de los sub-algoritmos cooperantes, con la intención de adecuar dinámicamente la búsqueda al estado global del sistema.
6. Deben programarse algoritmos paralelos débilmente acoplados, para evitar que el alto retraso asociado a los envíos WAN afecten negativamente a la eficiencia en tiempo.
7. Deben programarse algoritmos asíncronos, lo que permitirá evolucionar a los distintos algoritmos componentes de manera lo más independiente posible.
8. Dependiendo del protocolo utilizado, puede que sea necesario incorporar mecanismos de tolerancia a fallos, para permitir una degradación gradual del sistema cuando se pierden mensajes o cuando algún sub-algoritmo falla. Además, también hay que tener en cuenta que, en comparación con lo que sucede en LAN, en una WAN no se tiene el mismo grado de garantía respecto al correcto funcionamiento de todos los nodos de la red. En este contexto es normal que algunos computadores dejen de estar disponibles por varias causas (cortes en la red, caída de sistema operativo, fallos hardware, etc).

Aunque las soluciones apuntadas puedan parecer

muy genéricas, representan las decisiones reales que deben tomarse, y permitirán al lector interesado conocer cómo empezar a diseñar un algoritmo para la WAN y/o Internet en particular. Su identificación no es evidente, sobre todo para investigadores no experimentados en paralelismo o en el uso de redes de área extensa. El uso de un algoritmo WAN queda por supuesto supeditado a que la tarea represente un volumen de trabajo tal que admita los retrasos de comunicación de mensajes en WAN, típicamente de varias decenas de milisegundos o superiores.

Para implementar el sistema el lenguaje Java se presenta como uno de los más idóneos por su estrecha cercanía al trabajo en Internet y al paralelismo. Alternativamente, el uso de bibliotecas de paso de mensajes como MPI puede resultar interesante sobre todo en sistemas homogéneos, ya que la inter-operatividad de este estándar descansa aún en el dominio de la teoría. PVM permite comunicar máquinas heterogéneas usando un formato interno para representar los datos, aunque no provee los niveles de eficiencia ni uso en WAN de otros estándares. Por otro lado, sistemas como CORBA podrían utilizarse, pero quizás algunas de sus funciones más atractivas no serían necesarias en aplicación para PEA's. Finalmente, las nuevas tendencias en relación a intercambio de información en XML, el uso de nuevos lenguajes como C#, o de nuevos protocolos como SOAP (ambos integrados en la plataforma `Microsoft.NET`) pueden acelerar la implementación y permitir una mayor abstracción de los detalles sobre comunicaciones.

V. Nuevos Dominios de Aplicación para los PEA's

Con un número creciente de servicios para el usuario, la telecomunicación es un campo donde se exploran numerosas líneas de investigación. Muchos problemas de este área pueden formularse como tareas de optimización [11][12][18][20]. En la Sección V-A analizamos soluciones basadas en algoritmos genéticos para dos problemas de telecomunicaciones: el posicionamiento óptimo de antenas repetidoras y el diseño de códigos correctores de errores.

Por otro lado, hoy en día las Redes Neuronales Artificiales abarcan un gran espectro de aplicaciones [1]. En la Sección V-B comparamos distintos métodos de aprendizaje sobre problemas de bio-informática. Nuestro objetivo a largo plazo es estudiar este campo usando paradigmas paralelos, aunque en este artículo analizamos únicamente el caso secuencial por tratarse de una investigación en curso.

Nuestra intención es identificar brevemente puntos de interés en el uso de PEA's, aunque, por una evidente falta de espacio, quedarán necesariamente fuera aplicaciones de relevancia actual.

A. Telecomunicaciones

En esta Sección analizamos soluciones basadas en algoritmos genéticos para resolver el problema del diseño de códigos correctores de errores y de localización óptima de antenas en una red de radio.

A.1 Diseño de Códigos de Corrección de Errores en Telecomunicaciones

Para transmitir los mensajes de forma fiable y rápida a través de un canal con ruido se necesitan *códigos binarios* (p.ej. lineales de bloque). Cuanto mayor sea la mínima distancia de Hamming entre cualquier par de palabras (d), mayor será el número de bits erróneos que el código es capaz de corregir. En nuestro caso, dado el número de bits por palabra (n) y el número de palabras (M), queremos encontrar un código con valor d máximo. La función de adecuación que usamos fue inicialmente propuesta en [11]:

$$f(\overrightarrow{x}) = \frac{1}{\sum_{i=1}^{M} \sum_{j=1, i \neq j}^{M} \frac{1}{d_{ij}^2}} \qquad (1)$$

donde d_{ij} representa la distancia de Hamming entre las palabras i-ésima y j-ésima del código.

Para resolver el problema se han empleado seis tipos de algoritmos genéticos [4], tres secuenciales y tres versiones paralelas: ssGA, genGA, cGA, dssGA, dgenGA y dcGA. Para los algoritmos secuenciales se ha usado una población de 64 individuos, selección por ruleta, recombinación uniforme con probabilidad 1.0 y tendencia de 0.6 al mejor padre, mutación con probabilidad $\frac{1}{M \times n}$ y reemplazo si es mejor. En los algoritmos distribuidos se emplean 8 islas de 16 individuos, migración asíncrona de un individuo cada 256 iteraciones, con política de selección aleatoria y reemplazo si es mejor. Las máquinas usadas tienen una CPU Ultra Sparc a 143 MHz con 64 Mb de RAM.

TABLA III
Número de evaluaciones y tiempo empleado por cada algoritmo para resolver el problema con $n = 12$ y $M = 24$.

	Secuencial (ssGA)		*Distribuido (dssGA)*	
	#evals.	*tiempo (s)*	*#evals.*	*tiempo (s)*
ssGA	15476	85.37	36803	22.3
genGA	32756	217.47	42317	28.9
cGA	52757	280	89598	62

Como se aprecia en la Tabla III, los algoritmos ssGA (fila 1, columnas 1 y 2) y dssGA (fila 1, columnas 3 y 4) son los mejores desde el punto de vista del tiempo y el esfuerzo numérico. Sin embargo, de entre los algoritmos secuenciales, sólo cGA consigue una solución óptima en el 100% de los casos, mostrando la potencia intrínseca de los algoritmos descentralizados. Como conclusión global, dssGA es el mejor algoritmo evaluado, puesto que también encuentra el óptimo en el 100% de los casos y es el más rápido.

A.2 Localización Óptima de Antenas en el Diseño de una Red por Radio

Un objetivo importante en el diseño de una red de radio es cubrir el máximo área posible con el mínimo número de transmisores. Hemos elegido este

problema por la relevancia económica derivada del ahorro de antenas repetidoras. La parte del área que cubre cada transmisor se denomina *celda*. En nuestro caso supondremos que tanto las celdas como el área están discretizadas. La función de adecuación que usaremos fue propuesta en [9] y consiste en evaluar el siguiente cociente:

$$f(\overrightarrow{x}) = \frac{Cobertura(\overrightarrow{x})^{\alpha}}{Número\ de\ transmisores(\overrightarrow{x})} \qquad (2)$$

donde $Cobertura(\overrightarrow{x})$ es el porcentaje de área cubierta por la solución $\overrightarrow{x}$ y α permite dar mayor o menor importancia a la cobertura. En nuestros estudios hemos usado un área de 287×287 con 149 transmisores que cubren un área de 41×41, y con $\alpha = 2$.

Los algoritmos genéticos empleados para resolver el problema han sido ssGA y su versión distribuida dssGA. En el primero se ha empleado una población de 512 individuos, selección mediante ruleta, recombinación de 2 puntos con probabilidad 1.0, mutación con probabilidad $1/longcadena$ y reemplazo si es mejor. En el segundo se han usado 8 islas con 64 individuos y migración asíncrona de un individuo cada 2048 iteraciones con política de selección aleatoria y reemplazo si es mejor. Las máquinas empleadas tienen una CPU Ultra Sparc a 143 MHz con 64 Mb de RAM.

TABLA IV
NÚMERO DE EVALUACIONES Y TIEMPO CON SSGA Y DSSGA

	#evals.	*tiempo (horas)*
dssGA (8 CPUs)	505624	1.38
dssGA (1 CPU)	491496	10.87
ssGA	173013	3.86

De la Tabla IV se deduce que desde el punto de vista numérico el mejor algoritmo es ssGA, pero teniendo en cuenta el tiempo de búsqueda, el mejor es dssGA en 8 procesadores. La aceleración que presenta dssGA cuando lo trasladamos de 1 a 8 procesadores es de 7.87, casi lineal.

B. Bio-Informática

Una *Red Neuronal Artificial* (ANN) puede definirse como un conjunto de *unidades de proceso* (*neuronas*) que se comunican entre ellas por medio de señales digitales (analógicas en Biología tradicional). Estas señales viajan entre las unidades a través de *conexiones* ponderadas mediante *pesos sinápticos*. Para una descripción más amplia consúltese [22]. Nosotros consideraremos únicamente el *perceptrón multicapa* [24]. Nuestro objetivo es entrenar la red y para ello presentamos un resumen del uso de Retropropagación (BP) [25], algoritmos evolutivos y una combinación de ambos (para más información consulte [13]). Los problemas analizados han sido elegidos por su relación con la bio-informática:

- ECOLI: Pretende la predicción de la localización de proteinas en células eucarióticas [23]. Hay 336 patrones con 8 atributos descriptivos y su pertenencia a 8 clases diferentes. La arquitectura de la red usada es 8-4-2-8.
- BC: Corresponde al diagnóstico del cáncer de mama [21]. Hay 683 patrones con 9 atributos descriptivos y un atributo predictivo booleano (maligno o benigno). La arquitectura de la red usada es 9-4-3-1.

Los algoritmos evolutivos empleados han sido: ssGA, (1,10)-ES sin recombinación ni mutaciones correlacionadas, $UMDA_c$ (un Algoritmo de Estimación de Distribución o EDA) y $MIMIC_c$ (otra variante EDA). La diferencia entre $UMDA_c$ y $MIMIC_c$ reside en la forma en que se realiza la factorización de la función de densidad conjunta de los individuos seleccionados. En ssGA se ha empleado una población de 100 individuos, selección mediante ruleta, recombinación uniforme con probabilidad 1.0 y tendencia de 0.8 al mejor padre y mutación con probabilidad $1/longcadena$. En ES se ha tomado $\tau = 1/\sqrt{2n}$ y $\tau' = 1/\sqrt{\sqrt{2n}}$ siguiendo [8]. Los EDAs simulan 250 individuos en cada generación y seleccionan la mitad mejor para el aprendizaje de la función de densidad de probabilidad conjunta. Por último, en BP se ha usado una tasa de aprendizaje $\gamma = 0.1$ y un momento $\alpha = 0.5$.

TABLA V
RESULTADOS OBTENIDOS EN EL PROBLEMA ECOLI

	ECOLI	
Algoritmo	*error*	*% acierto*
BP	0.1289 ± 0.0017	8.3333 ± 6.3909
GA	0.1001 ± 0.0038	47.8308 ± 7.8949
ES	0.0891 ± 0.0027	65.8929 ± 2.5301
$UMDA_c$	0.0808 ± 0.0022	58.5970 ± 5.9286
$MIMIC_c$	0.0802 ± 0.0018	58.5075 ± 4.8153
GA + BP	0.1522 ± 0.0040	8.0398 ± 5.9927
ES + BP	0.0939 ± 0.0019	53.5417 ± 4.2519
$UMDA_c$ + BP	0.1593 ± 0.0011	9.8209 ± 6.9430
$MIMIC_c$ + BP	0.1585 ± 0.0010	10.4179 ± 7.3287

TABLA VI
RESULTADOS OBTENIDOS EN EL PROBLEMA BREAST CANCER

	Breast Cancer	
Algoritmo	*error*	*% acierto*
BP	0.2244 ± 0.0074	63.2650 ± 2.9311
GA	0.1125 ± 0.0062	90.8676 ± 1.1248
ES	0.0776 ± 0.0039	95.8565 ± 0.4529
$UMDA_c$	0.0746 ± 0.0035	95.2353 ± 0.4609
$MIMIC_c$	0.0753 ± 0.0042	95.0735 ± 0.5892
GA + BP	0.1817 ± 0.0059	71.3824 ± 3.0779
ES + BP	0.0952 ± 0.0098	93.7189 ± 1.2528
$UMDA_c$ + BP	0.2747 ± 0.0100	51.3529 ± 3.4916
$MIMIC_c$ + BP	0.2659 ± 0.0206	54.2206 ± 7.2556

Se ha realizado una validación cruzada dividiendo el conjunto de patrones en 5 trozos. Los resultados se encuentran en las Tablas V y VI. Observando la columna de *error* vemos que las ESs y los EDAs ofrecen mejores resultados que los demás algoritmos. Los algoritmos híbridos son peores que sus versiones sin hibridar. Esto puede ser debido a que BP realiza demasiado cálculo o a que los parámetros γ y α no son los más adecuados.

Considerando los resultados que hemos presentado con algoritmos secuenciales, parece claro también

que en este dominio es necesario el uso de algoritmos paralelos que permitan tanto mejorar estos resultados numéricos como disminuir los elevados tiempos de espera experimentados al abordar estas aplicaciones. En esta línea, tanto un algoritmo distribuido como uno celular pueden aliviar el problema; el primero por una previsible disminución del tiempo de ejecución, y el segundo por una previsible disminución del esfuerzo algorítmico que redundaría en un menor tiempo de espera.

VI. Conclusiones y Trabajo Futuro

En este trabajo hemos intentado capturar algunos de los puntos de interés actual en el dominio de los algoritmos evolutivos paralelos. Hemos identificado los problemas relativos a los algoritmos en sí mismos, tales como el uso de sistemas heterogéneos y la ejecución *eficiente*. También hemos analizado brevemente aplicaciones en dos dominios de gran interés actual como las telecomunicaciones y la bioinformática.

Nuestro espíritu no ha sido únicamente idenfiticar problemas, sino proponer soluciones concretas y analizar el comportamiento de algoritmos reales dentro de las limitaciones de un artículo de reducida extensión.

Como trabajo futuro parece claro que aún restan muchos puntos que estudiar en cuanto a utilidad de los sistemas propuestos aquí y también por otros autores, ya que en muchos casos las propuestas teóricas de algoritmos se separan de la realidad de su ejecución y sobre todo no aclaran la utilidad directa de los logros. Es por esto que deben proponerse aplicaciones del mundo real y algoritmos eficientes en cualquiera de los puntos abiertos en la actualidad.

Reconocimientos

Trabajo parcialmente financiado por la Comisión Interministerial de Ciencia y Tecnología (CICYT) mediante el contrato TIC99-0754-C03-03.

Referencias

[1] J. T. Alander. Indexed bibliography of genetic algorithms and neural networks. Technical Report 94-1-NN, University of Vaasa, Department of Information Technology and Production Economics, 1994.

[2] E. Alba. Parallel evolutionary algorithms can achieve superlinear speedup. *Information Processing Letters*, 2001 (to appear).

[3] E. Alba, A.J. Nebro, and J.M. Troya. Heterogeneous computing and parallel genetic algorithms. *Journal of Parallel and Distributed Computing*, 2001 (to appear).

[4] E. Alba and J. M. Troya. A survey of parallel distributed genetic algorithms. *Complexity*, 4(4):31–52, 1999.

[5] E. Alba and J. M. Troya. Influence of the migration policy in parallel distributed GAs with structured and panmictic populations. *Applied Intelligence*, 12(3):163–181, 2000.

[6] E. Alba and J. M. Troya. Analyzing synchronous and asynchronous parallel distributed genetic algorithms. *Future Generation Computer Systems*, 17:451–465, January 2001.

[7] E. Alba and J. M. Troya. Gaining new fields of application for OOP: the parallel evolutionary algorithm case. *Journal of Object Oriented Programming*, 2001 (to appear).

[8] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, New York, 1996.

[9] P. Calégari, F. Guidec, P. Kuonen, and D. Kobler. Parallel island-based genetic algorithm for radio network design. *Journal of Parallel and Distributed Computing*, 47:86–90, 1997.

[10] E. Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Press, 2000.

[11] H. Chen, N.S. Flann, and D.W. Watson. Parallel genetic simulated annealing: A massively parallel SIMD algorithm. *IEEE Transactions on Parallel and Distributed Systems*, 9(2):126–136, 1998.

[12] C.H. Chu, G. Premkumar, and H. Chou. Digital data networks design using genetic algorithms. *European Journal of Operational Research*, 127:140–158, 2000.

[13] C. Cotta, E. Alba, R. Sagarna, and P. Larrañaga. "adjusting weights in artificial neural networks using evolutionary algorithms". In P. Larrañaga and J.A. Lozano, editors, *"Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation"*, pages 357–373. Kluwer Academic Publishers, 2001.

[14] V. Donaldson, F. Berman, and R. Paturi. Program speedup in a heterogeneous computing network. *Journal of Parallel and Distributed Computing*, 21:316–322, 1994.

[15] V. S. Gordon and D. Whitley. Serial and parallel genetic algorithms as function optimizers. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 177–183. Morgan Kaufmann, 1993.

[16] F. Herrera and M. Lozano. Gradual distributed real-coded genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):43–63, 2000.

[17] K. A. De Jong, M. A. Potter, and W. M. Spears. Using problem generators to explore the effects of epistasis. In T. Bäck, editor, *Proceedings of the 7th International Conference of Genetic Algorithms*, pages 338–345. Morgan Kaufman, 1997.

[18] A. Kapsalis, V.J. Rayward-Smith, and G.D. Smith. Using genetic algorithms to solve the radio link frequency assigment problem. In D.W. Pearson, N.C. Steele, and R.F. Albretch, editors, *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, pages 37–40. Springer-Verlag, 1995.

[19] A. H. Karp and H. P. Flatt. Measuring parallel processor performance. *Communications of the ACM*, 33(5):539–543, 1990.

[20] S. Khuri and T. Chiu. Heuristic algorithms for the terminal assignment problem. In *Proceedings of the 1997 ACM Symposium on Applied Computing*, pages 247–251. ACM Press, 1997.

[21] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1–18, 1990.

[22] J. L. McClelland and D. E. Rumelhart. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. The MIT Press, 1986.

[23] K. Nakai and M. Kanehisa. A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics*, 14:897–911, 1992.

[24] F. Rosenblatt. *Principles of Neurodynamics*. Spartan Books, New York, 1959.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by backpropagating errors. *Nature*, 323:533–536, 1986.

[26] J. Sarma and K. A. De Jong. An analysis of the effect of the neighborhood size and shape on local selection algorithms. In H. M. Voigt, W. Ebeling, I. Rechenberg, and H. P. Schwefel, editors, *Parallel Problem Solving from Nature (PPSN IV)*, volume 1141 of *Lecture Notes in Computer Science*, pages 236–244. Springer-Verlag, Heidelberg, 1996.

[27] J. Sprave. A unified model of non-panmictic population structures in evolutionary algorithms. In *Congress on Evolutionary Computation (CEC'99)*, pages 1384–1391. IEEE Press, Piscataway, NJ, 1999.

[28] G. Syswerda. A study of reproduction in generational and steady-state genetic algorithms. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann, 1991.