Evaluando el Rendimiento Multicore de una Aproximación Multiobjetivo de Inteligencia de Enjambre para la Inferencia Filogenética

Sergio Santander-Jiménez[1], Miguel A. Vega-Rodríguez[1], Juan A. Gómez-Pulido[1]y Juan M. Sánchez-Pérez[1].

Resumen— **Una amplia variedad de problemas de optimización precisa la combinación de computación paralela y bioinspirada para afrontar la búsqueda de soluciones en tiempos razonables. La era multicore ha permitido el desarrollo de estrategias para explotar las arquitecturas hardware con objeto de resolver estos problemas NP-Completos. Un ejemplo reconocido en el ámbito de la Bioinformática es la inferencia de árboles filogenéticos que describan la evolución de las especies. Debido a que el espacio de búsqueda aumenta exponencialmente con el número de especies, no se pueden aplicar búsquedas exhaustivas. Además, se dan dificultades adicionales cuando es preciso considerar simultáneamente múltiples criterios para resolver el problema. En este artículo estudiamos el rendimiento sobre arquitecturas multicores del algoritmo Multiobjective Artificial Bee Colony para la inferencia filogenética según los criterios de máxima parsimonia y máxima verosimilitud. Los resultados experimentales muestran que la propuesta mejora otras aproximaciones basadas en técnicas avanzadas de Computación de Alto Rendimiento sobre grandes conjuntos de datos.**

Palabras clave— **Inferencia Filogenética, Artificial Bee Colony, Arquitecturas Multicore, Optimización Multiobjetivo.**

I. Introducción

EN los últimos años, el desarrollo de las cada vez más populares arquitecturas multicore ha permitido a los investigadores afrontar complejos retos computacionales mediante el uso de técnicas de computación paralela. Aquellos problemas de optimización donde el espacio de búsqueda crece exponencialmente con el tamaño de los datos de entrada pueden ser resueltos en tiempos razonables gracias al empleo de Metaheurísticas y Paralelismo. En el terreno de la Bioinformática se pueden encontrar una amplia variedad de este tipo de problemas, los cuales se ven afectados por altos tiempos de procesamiento y consumo de memoria. La inferencia filogenética es considerada como uno de los más importantes ámbitos de estudio en esta disciplina [1]. A lo largo de los años, se han propuestos diversos métodos para resolver este problema de acuerdo a distintos criterios de optimalidad. Sin embargo, dado un conjunto de datos, los árboles filogenéticos obtenidos usando diferentes métodos pueden presentar relaciones evolutivas inconsistentes entre sí. Esta cuestión motivó diferentes propuestas basadas en optimización multiobjetivo aplicadas a la inferencia filogenética [2]. Considerando diversos objetivos de manera simultánea, la complejidad del proceso de búsqueda aumenta, por lo que el modo de afrontar estos problemas es la aplicación de metaheurísticas paralelas. En un contexto multicore, el diseño de algoritmos paralelos que exploten eficientemente el hardware resulta esencial para llevar a cabo análisis filogenéticos multiobjetivo en grandes conjuntos de datos.

En este artículo proponemos un estudio paralelo de una adaptación multiobjetivo del algoritmo Artificial Bee Colony (ABC) [3] para reconstruccion filogenética según los criterios de máxima parsimonia y máxima verosimilitud. El principal objetivo es mostrar que un aproximación diseñada para arquitecturas multicore simétricas puede mejorar los factores de aceleración y eficiencia con respecto a otras propuestas basadas en computación de alto rendimiento. Por ello, hemos hecho uso de un diseño algorítmico que explota las características de las máquinas multicore. Hemos elegido este algoritmo de inteligencia de enjambre debido a los prometedores resultados que ha obtenido para una amplia variedad de problemas [4], así como a que su diseño permite aplicar paralelismo de manera efectiva para mejorar los tiempos de ejecución [5]. Esta versión paralela del algoritmo Multiobjective Artificial Bee Colony (MOABC) ha sido probada usando cuatro bases de datos públicas de nucleótidos, tomándose los tiempos de ejecución necesarios en función del número de hilos, así como valores de aceleración y eficiencia. Además, presentamos los resultados biológicos obtenidos y una evaluación del rendimiento multiobjetivo mediante la métrica de hipervolumen.

Este artículo está organizado del siguiente modo. En la siguiente sección, se introducen los fundamentos del problema, así como modelos paralelos propuestos para enfocar la exploración del espacio de búsqueda de topologías filogenéticas. La Sección III describe los detalles algorítmicos de la propuesta. En la Sección IV mostramos, explicamos y comparamos los resultados experimentales obtenidos. Para finalizar, la Sección V incluye una serie de conclusiones y la definición de trabajo futuro.

II. Inferencia Filogenética

La inferencia filogenética engloba un amplio rango de técnicas de estimación que tratan de describir las relaciones evolutivas que se han dado a lo largo de la historia entre un conjunto de organismos [6].

[1]Universidad de Extremadura, Dept. Tecnología de los Computadores y de las Comunicaciones, Escuela Politécnica. Campus Universitario s/n, 10003, Cáceres, España. {sesaji, mavega, jangomez, sanperez}@unex.es

Estos métodos toman como entrada un conjunto de n secuencias de N caracteres que representan características moleculares de las especies en estudio, y generan como salida una estructura matemática que representa una hipótesis de la evolución de las especies: el *árbol filogenético*. A lo largo de los años se han propuesto diversos métodos para reconstrucción filogenética de acuerdo a múltiples criterios, siendo dos de los más populares los métodos de máxima parsimonia y máxima verosimilitud.

A. Máxima Parsimonia

Los métodos de máxima parsimonia tratan de encontrar aquellas hipótesis evolutivas que minimicen el número de cambios a nivel molecular necesarios para explicar los datos observados [7]. Es una aproximación inspirada en el razonamiento de Occam, que establece que la hipótesis más sencilla para explicar un determinado fenómeno será más plausible que otras posibles explicaciones más complejas.

La parsimonia de un árbol filogenético τ, inferido de un conjunto de n secuencias de N nucleótidos alineados, viene definida por la siguiente función [8]:

$$P(\tau) = \sum_{i=1}^{N} \sum_{(a,b)\in B(\tau)} C(a_i, b_i) \qquad (1)$$

donde (a, b) es la rama del conjunto B que relaciona los nodos a y b, a_i y b_i el valor del i-ésimo carácter de las secuencias asociadas a a y b, y $C(a_i, b_i)$ el coste de evolucionar del estado a_i al b_i. Se prefieren aquellas topologías que *minimicen* el valor de esta función, al representar una explicación más sencilla a la evolución de las especies de entrada.

B. Máxima Verosimilitud

La verosimilitud es una medida estadística cuya aplicación en filogenética busca dar lugar a aquel árbol filogenético que represente la historia evolutiva más probable de los organismos de entrada [7]. A la hora de calcular la verosimilitud, debemos tener en cuenta la topología del árbol filogenético, los valores de longitud de ramas (que representan tiempo evolutivo entre especies emparentadas) y las probabilidades de cambio a nivel de nucleótido dadas por el modelo de evolución molecular considerado [1].

Dados τ la topología a evaluar, D el conjunto de secuencias moleculares de N caracteres, D_i el estado del carácter i-ésimo de las secuencias y m el modelo evolutivo, se define la verosimilitud como la probabilidad condicional de los datos dados τ y m [1]:

$$L[D, \tau, m] = \Pr[D|\tau, m] = \prod_{i=1}^{N} \prod_{j=1}^{B} (r_i t_j)^{n_{ij}} \qquad (2)$$

donde B es el conjunto de ramas de τ, r_i la probabilidad de cambio del estado i, t_j la longitud de la rama j y n_{ij} el número de cambios de estado en j para el carácter i. Se prefieren aquellos árboles que *maximicen* la verosimilitud, al representar una hipótesis más probable de la historia evolutiva.

C. Filogenética y Paralelismo

Así como en el caso de otras propuestas basadas en criterios de optimalidad, los métodos de máxima parsimonia y máxima verosimilitud son considerados como problemas NP-Completos. Snell et al. identificaron dos factores claves que explican la complejidad del problema [7]:

1. Al aumentar el número de especies de entrada, el espacio de búsqueda de posibles topologías filogenéticas crece de manera exponencial. Dado n el número de especies a procesar, el número de posibles topologías viene dado por $\frac{(2n-5)!}{(n-3)!2^{n-3}}$.
2. Al aumentar el número de caracteres en las secuencias moleculares, los bucles de evaluación de topologías precisarán mayores tiempos de procesamiento y consumo de memoria. Además, la complejidad del proceso de evaluación se ve afectada al considerar múltiples criterios simultáneamente, como sucede al aplicar técnicas de optimización multiobjetivo.

Para poder resolver estos dos problemas es preciso combinar técnicas basadas en metaheurísticas y paralelismo. Esta metodología permitirá afrontar complejos análisis filogenéticos sobre grandes conjuntos de datos. Bader et al. propusieron uno de los modelos más populares para diseñar estrategias paralelas basadas en criterios de optimalidad, definiendo tres niveles de paralelismo [9]: paralelismo a nivel de tarea, de grano grueso y de grano fino.

El paralelismo a nivel de tarea consiste en llevar a cabo varios análisis filogenéticos completos sobre el mismo conjunto de datos (ejecuciones independientes del algoritmo), considerando diferentes topologías de partida. Por su parte, las estrategias de grano grueso permiten realizar en paralelo múltiples modificaciones topológicas y evaluaciones de árboles filogenéticos con objeto de profundizar en la exploración del espacio de búsqueda. El modo de implementar este nivel de paralelismo dependerá del diseño concreto del algoritmo utilizado. Dentro de esta categoría se incluyen los algoritmos basados en modelo maestro-esclavo [7]. Factores como la memoria consumida por los procesos o hilos, el porcentaje de tiempo serie y posibles dependencias y cuellos de botella en el diseño algorítmico resultan claves a la hora de explotar los recursos computacionales en este nivel. Finalmente, el nivel de grano fino trata de dividir los bucles de evaluación de árboles en tareas independientes que pueden ejecutarse en paralelo. Debido a que la evaluación de árboles no presenta dependencias entre caracteres diferentes en las secuencias, es posible paralelizar estos bucles mediante el cálculo de valores parciales que conformarán el valor final de parsimonia o verosimilitud.

En este trabajo resulta especialmente relevante la aplicación de paralelismo de grano grueso al adecuarse a los principios de diseño de los algoritmos basados en inteligencia de enjambre, permitiendo la paralelización de tareas independientes asignadas a los diferentes individuos que conforman la población.

III. Artificial Bee Colony

D. Karaboga propuso en 2005 un algoritmo de inteligencia de enjambre para resolver problemas de optimización inspirándose en el comportamiento colectivo que presentan las abejas de la miel, el Artificial Bee Colony [3]. En la naturaleza, las abejas realizan sus tareas e interactúan con otras para encontrar y explotar fuentes de alimento. Este comportamiento está gobernado por una serie de reglas bien definidas que permite al enjambre obtener una inteligencia colectiva. Este algoritmo considera tres tipos de abejas en la colmena, cada uno con un rol específico y un conjunto de tareas a realizar: abejas *obreras*, *observadoras* y *exploradoras*. El diseño del ABC puede ser paralelizado de una manera sencilla, al poder llevarse a cabo simultáneamente las tareas asignadas a un determinado grupo de abejas entre todas las que conforman dicho grupo. Esto facilita la posibilidad de reducir los tiempos computacionales haciendo uso de tecnologías como OpenMP.

A. Parallel Multiobjective Artificial Bee Colony

El objeto de este estudio es proponer una aproximación de grano grueso para paralelizar una adaptación multiobjetivo del ABC, el MOABC, para reconstrucción filogenética sobre máquinas multicore. Al aplicar técnicas de optimización multiobjetivo, la búsqueda de soluciones debe considerar la optimización conjunta de múltiples objetivos conflictivos, por lo que la evaluación de dichas soluciones debe realizarse según el concepto de *dominancia* [2]. Se dice que una solución x domina a una y si y solo si x presenta mejores o iguales valores que y en todos los objetivos y, al menos, es mejor en uno de ellos. Este tipo de metaheurísticas busca encontrar aquellas soluciones que se aproximen en mayor medida al conjunto de soluciones *Pareto* óptimas al problema.

Se han utilizado las librerias OpenMP para implementar la versión paralela del MOABC. Su elección se debe al hecho de que OpenMP permite explotar las capacidades de arquitecturas multicores sin modificar el diseño original del algoritmo, pudiéndose alcanzar mejores rendimientos que aproximaciones basadas en tecnologías como MPI sobre arquitecturas SMP [10]. El algoritmo MOABC toma los siguientes parámetros de entrada:

1. tamañoEnjambre. Tamaño de la población.
2. iteracionesMax. Número de iteraciones.
3. límite. Parámetro definido para determinar la presencia de óptimos locales.
4. mutación. Ratio de mutación a aplicar sobre las soluciones halladas para generar otras nuevas.
5. nHilos. Número de hilos OpenMP.

La salida del algoritmo consistirá en un conjunto de soluciones Pareto formado por árboles filogenéticos que representan un consenso entre los criterios de máxima parsimonia y máxima verosimilitud.

La inicialización de la población (compuesta por un 50 % de abejas obreras y un 50 % de observadoras) se lleva a cabo mediante la asignación de topologías filogenéticas iniciales seleccionadas de un repositorio de 1000 árboles generados mediante técnicas de bootstrap [1]. Tras ello, comienza el bucle principal compuesto por tres secciones básicas: fase de explotación por abejas obreras, fase de selección y explotación por observadoras y fase de búsqueda por exploradoras. Situamos este bucle dentro de una región paralela usando la directiva *#pragma omp parallel* para permitir la realización de tareas en paralelo usando *nHilos*. Las acciones asociadas a cada tipo de abejas pueden realizarse en paralelo aplicando directivas *#pragma omp for* ya que no existen dependencias entre las tareas llevadas a cabo por una abeja X y otras que pertenecen a su mismo grupo.

En primer lugar, las *abejas obreras* se encargan de buscar fuentes de alimento en la vecindad del panal. Esto equivale a la búsqueda de soluciones al problema para su explotación. Para ello, se muta la solución inicial asociada a cada obrera usando cambios topológicos aleatorios mediante el operador Nearest Neighbour Interchange (NNI) [1] y modificando longitudes de ramas usando una distribución gamma. La solución resultante es comparada con la original mediante un fitness multiobjetivo, que asigna un valor a una solución s de acuerdo al número de soluciones en la población dominadas por s y las soluciones que dominan a s [11]. Así, la abeja obrera almacenará aquella topología que minimice este valor de fitness. Durante este paso, los movimientos NNI, la actualización de ramas y la evaluación de parsimonia y verosimilitud de las topologías puede realizarse en paralelo:

```
#pragma omp for numthreads (nHilos) private (j, nuevaObrera)
for j = 1 to tamañoEnjambre/2 do
    nuevaObrera ← generarVecina(C[j],mutación)
    if MOFitness(nuevaObrera) < MOFitness(C[j]) then
        C[j] ← nuevaObrera
    end if
end for
```

Al completar sus tareas, las abejas obreras vuelvan al panal e interactúan con las abejas observadoras, realizando bailes para comunicar la calidad de las fuentes de alimento encontradas. De esta manera, las abejas observadoras seleccionarán las fuentes más prometedoras para su explotación. Este comportamiento se modela mediante el cálculo de un vector de probabilidades de selección de soluciones, para lo cual se ordenan las soluciones actuales usando los operadores *fast non dominated sort* y *distancia de crowding* [12]. El vector calculado a partir de esta ordenación será verificado por las observadoras para seleccionar las topologías más prometedoras. A partir de estos árboles, se generan soluciones vecinas empleando los operadores de mutación y se comparan con las originales usando el fitness multiobjetivo. Con objeto de promover la diversidad en la población, la solución vecina será preferida si obtiene un valor menor o igual de fitness con respecto a la seleccionada. Los procesos de selección, cambios a nivel topológico y la evaluación de árboles pueden llevarse a cabo en paralelo:

```
#pragma omp for numthreads (nHilos) private (j, nuevaObservadora)
```

```
for j = (tamañoEnjambre/2)+1 to tamañoEnjambre do
   obreraSeleccionada ← seleccionarObrera(probVector,C)
   nuevaObservadora ←
         generarVecina(obreraSeleccionada,mutación)
   if MOFitness(nuevaObservadora) ≤
         MOFitness(obreraSeleccionada) then
      C[j] ← nuevaObservadora
   else
      C[j] ← obreraSeleccionada
   end if
end for
```

En el tercer paso, las abejas exploradoras realizan la búsqueda aleatoria de nuevas fuentes para evitar la ausencia de alimento en el panal. Si el árbol filogenético asociado a una abeja no ha sido mejorado en *límite* iteraciones, esta abeja ha alcanzado un posible óptimo local y será reemplazada por una exploradora, que realizará búsquedas en regiones del espacio aún no tratadas. Para asignar las tareas de conversión a los hilos de una manera eficiente, previamente se calcula el número de soluciones no mejoradas para calcular el número de abejas exploradoras a generar. La conversión a abeja exploradora consiste en la selección de una nueva topología filogenética del repositorio y su mejora mediante optimización topológica y de ramas para generar nuevos árboles de calidad. La selección de árboles iniciales, la optimización topológica y el cómputo de los nuevos valores de parsimonia y verosimilitud se realiza dentro de un bucle paralelo como se indica:

```
numExploradoras ← BuscarSolucionesAgotadas(tamañoEnjambre, límite, C)
#pragma omp for numthreads (nHilos) private (j)
for j = 1 to numExploradoras do
   C[getBeeID(j)] ← generarExploradora()
end for
```

Completadas estas tres secciones, las *tamañoEnjambre*/2 mejores soluciones encontradas son asignadas a las abejas obreras como nuevos árboles iniciales, actualizándose el conjunto de Pareto de soluciones no dominadas y comenzando una nueva iteración. En este esquema paralelo, la carga de trabajo algorítmica es repartida entre los hilos de ejecución mediante el empleo de paralelismo de grano grueso. Es preciso considerar las dependencias entre pasos (como en los casos del cálculo del vector de probabilidades y número de exploradoras), así como posibles sobrecargas introducidas por OpenMP para comprender los factores de aceleración y eficiencia obtenidos por la propuesta.

IV. Resultados Experimentales

Con objeto de medir el rendimiento del MOABC en términos de aceleración y eficiencia, hemos llevado a cabo una serie de experimentos consistentes en cuatro ejecuciones completas del algoritmo para configuraciones de 2, 4, 6, 8, 12, 14 y 16 hilos OpenMP. Esta configuración de experimentos se ha realizado a fin de poder comparar con otras aproximaciones y mostrar la relevancia estadística de la propuesta. El hardware sobre el que se realizaron los experimentos es el siguiente: arquitectura de 4-procesadores AMD Opteron Magny-Cours 6174 de 12 cores a 2,2 GHz y 12MB Cache, ejecutando Scientific Linux 6.1. La aplicación fue compilada usando GCC 4.4.5.

Cada serie de experimentos se ha realizado sobre cuatro bases de datos públicas de nucleótidos [8]: $rbcL_55$, 55 secuencias de 1314 nucleótidos cada una del gen rbcL de diversas especies de plantas verdes. $mtDNA_186$, 186 secuencias de 16608 nucleótidos de ADN mitocondrial humano. $RDPII_218$, 218 secuencias de 4182 nucleótidos de ARN procariota. Y $ZILLA_500$, 500 secuencias de 759 nucleótidos del gen plástido rbcL. Una ejecución completa constará de 100 generaciones con un tamaño de enjambre de 100 abejas, usando un 5 % de ratio de mutación y el parámetro límite configurado a 15 iteraciones.

La Tabla I muestra los tiempos secuenciales de procesamiento necesarios para cada una de las instancias. Por su parte, la Tabla II indica los valores medios de aceleración y eficiencia hasta un máximo de 16 hilos. En la Figura 1 se presenta una representación gráfica de los factores de aceleración en comparación con el valor teórico lineal.

TABLA I: Tiempos secuenciales.

Instancia	Tiempo de ejecución
$rbcL_55$	7254,20 segundos
$mtDNA_186$	62404,82 segundos
$RDPII_218$	80278,85 segundos
$ZILLA_500$	85089,03 segundos

TABLA II: Valores de aceleración y eficiencia.

	$rbcL_55$		$mtDNA_186$	
Hilos	Aceleración	Eficiencia	Aceleración	Eficiencia
1	1	100 %	1	100 %
2	1,99460051	99,7300257 %	1,93608536	96,8042678 %
4	3,67392078	91,8480194 %	3,49034693	87,2586732 %
6	4,88096668	81,3494446 %	4,90193807	81,6989678 %
8	5,93483036	74,1853795 %	6,07722647	75,9653309 %
10	7,29422408	72,9422408 %	7,54790465	75,4790465 %
12	7,62128753	63,5107294 %	7,95185613	66,2654677 %
14	8,52982504	60,9273217 %	8,41027659	60,0734042 %
16	8,65108446	54,0692779 %	8,86530012	55,4081257 %
	$RDPII_218$		$ZILLA_500$	
Hilos	Aceleración	Eficiencia	Aceleración	Eficiencia
1	1	100 %	1	100 %
2	1,97128354	98,5641772 %	1,99091206	99,5456028 %
4	3,62240497	90,5601244 %	3,89831364	97,4578411 %
6	4,93587323	82,2645538 %	5,26050453	87,6750755 %
8	6,36198653	79,5248316 %	6,90515808	86,3144761 %
10	7,92441729	79,2441729 %	8,44623105	84,4623105 %
12	8,45198892	70,4332410 %	9,00639617	75,0533014 %
14	9,44069955	67,4335682 %	10,4663614	74,7597245 %
16	9,57614467	59,8509042 %	11,0960821	69,3505129 %

A la vista de estos resultados, podemos sugerir que este algoritmo alcanza mejores valores de aceleración a medida que incrementamos el número de especies en la entrada del procedimiento. Para justificar este resultado, usaremos el modelo de escalabilidad multicore propuesto por Hill y Marty [13]. Estos autores aumentaron la ley de Amdalh y propusieron una formulación actualizada para calcular el valor teórico de aceleración sobre arquitecturas multicore. Dada f la fracción paralelizable de la aplicación, n el total de recursos en chip y r los recursos base core equivalent (BCE), el factor de aceleración en arquitecturas SMP viene dado por [13]:

$$Speedup(f,n,r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f \cdot r}{perf(r) \cdot n}} \qquad (3)$$

De acuerdo a la ley de Amdalh extendida, obtener mayores factores de aceleración usando el mismo hardware requiere fracciones paralelizables de software cercanas a 1. Aplicando estas ideas a nuestra

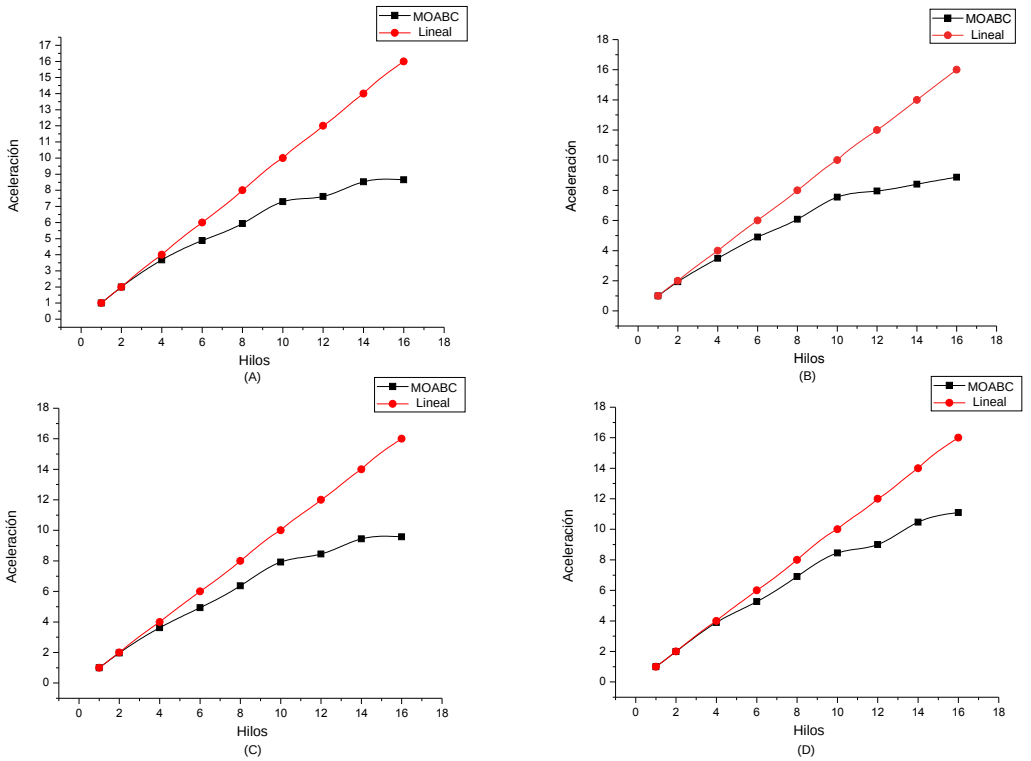

Fig. 1: Factores de aceleración para rbcL_55 (A), mtDNA_186 (B), RDPII_218 (C) y ZILLA_500 (D).

propuesta, podemos indicar que el número de tareas a realizar por una abeja se incrementa de manera proporcional al número de especies de entrada. El número y complejidad de las modificaciones a nivel topológico, junto con otras operaciones de procesado y evaluación de soluciones es mayor para grandes conjuntos de datos. Estas operaciones tienen lugar en regiones paralelas, de manera que la fracción paralela de código es mayor con respecto a la fracción de secciones críticas. Por ello, la escalabilidad del algoritmo mejora con instancias grandes, como puede observarse para el caso de *ZILLA_500* (consiguiéndose una eficiencia del 69.35 % para 16 hilos).

Debido a la gestión de bucles de OpenMP, se puede observar un menor incremento en el factor de aceleración al usar 12 hilos con respecto a otras configuraciones. Al usar un tamaño de enjambre de 100 abejas, el número de iteraciones para los pasos de explotación viene dado por tamañoEnjambre/2=50 iteraciones, que deben distribuirse entre 12 hilos. Estáticamente, cada hilo debe llevar a cabo 4 iteraciones, pero además deben repartirse dos adicionales (50 %12), incrementando la carga de trabajo un **25 %**.

A. Comparación con Otros Autores

En 2010, Cancino et al. estudiaron distintas estrategias paralelas para mejorar su software multiobjetivo para inferencia filogenética, PhyloMOEA, usando técnicas de computación con clusters [14]. Plantearon dos diferentes versiones, una basada en un esquema MPI maestro-esclavo y un sistema híbrido MPI-OpenMP que implementaba paralelismo de grano fino para reducir los tiempos de cómputo de la función de verosimilitud. La Tabla III presenta un comparación entre los factores de aceleración obtenidos por estas dos versiones y nuestro MOABC para el caso de 16 hilos de ejecución.

TABLA III: MOABC - PhyloMOEA. Aceleración.

Instancia	MOABC	PhyloMOEA MPI	PhyloMOEA Híbrido
rbcL_55	8,65	7,3	8,3
mtDNA_186	8,86	7,4	8,5
RDPII_218	9,58	9,8	10,2
ZILLA_500	11,09	6,7	6,3

Estos resultados muestran que los valores de aceleración alcanzados por el MOABC pueden competir con aproximaciones basadas en computación con clusters para las instancias *rbcL_55*, *mtDNA_186* y *RDPII_218*. Al considerar un mayor número de especies, como en *ZILLA_500*, se puede apreciar una mejora significativa con respecto a los valores obtenidos por Cancino et al. Esto sugiere que un diseño algorítmico con características paralelas intrínsecas puede dar lugar a mejores valores de aceleración para instancias complejas. Las arquitecturas multicores permiten llevar a cabo análisis filogenéticos a gran escala, mejorando otras propuestas que precisan de más recursos computacionales.

B. Resultados Filogenéticos

En esta subsección mostramos los resultados filogenéticos obtenidos por el MOABC. La Tabla IV presenta los valores de parsimonia y verosimilitud para los árboles más parsimoniosos y verosímiles obtenidos por el algoritmo en comparación con los resultados publicados por Cancino y Delbem en [8]. Esta tabla muestra que nuestras soluciones dominan los árboles generados usando PhyloMOEA. En la Tabla

V presentamos una comparativa adicional entre los árboles más verosímiles hallados por MOABC y PhyloMOEA usando el modelo evolutivo $HKY85 + \Gamma$, que permite mejorar los valores de verosimilitud sin afectar a la parsimonia [15]. Nuevamente, nuestra propuesta obtiene mejores resultados a nivel filogenético.

TABLA IV: Comparativa de resultados filogenéticos.

MOABC				
	Árbol más parsimonioso		Árbol más verosímil	
Instancia	Parsimonia	Verosimilitud	Parsimonia	Verosimilitud
rbcL_55	4874	**-23952,016**	**4883**	**-23945,068**
mtDNA_186	**2431**	**-40271,844**	**2449**	**-40212,226**
RDPII_218	**41488**	**-145742,297**	**42528**	**-144076,293**
ZILLA_500	**16218**	**-82682,267**	**16269**	**-82436,478**
PhyloMOEA				
	Árbol más parsimonioso		Árbol más verosímil	
Instancia	Parsimonia	Verosimilitud	Parsimonia	Verosimilitud
rbcL_55	4874	-24626,243	4884	-24583,330
mtDNA_186	2438	-41004,302	2450	-40894,343
RDPII_218	41534	-158724,280	42631	-156595,822
ZILLA_500	16219	-87275,281	16276	-86993,825

TABLA V: Comparativa usando $HKY85 + \Gamma$.

Instancia	MOABC	PhyloMOEA
rbcL_55	**-21821,480**	-21889,844
mtDNA_186	**-39890,140**	-39896,441
RDPII_218	**-134149,328**	-134696,535
ZILLA_500	**-80965,400**	-81018,06

Con objeto de evaluar el rendimiento multiobjetivo del algoritmo, hemos empleado la métrica de hipervolumen, que permite decidir el porcentaje del espacio de soluciones dominado por nuestras soluciones de Pareto. La Tabla VI muestra los valores medios de hipervolumen junto con los puntos de referencia necesarios para su cálculo. De acuerdo a la Columna 6, nuestras soluciones cubren más del 64.5 % del espacio definido por los puntos de referencia para *rbcL_55*, *mtDNA_186* y *ZILLA_500* y 70.37 % para *RDPII_218*, lo cual supone un porcentaje significativo del espacio de soluciones.

TABLA VI: Métrica de Hipervolumen.

	Mínimo		Máximo		Hipervolumen	
Instancia	Punto de Referencia		Punto de Referencia		Media (%)	Desv. estándar
	Pars.	Verosim.	Pars.	Verosim.		
rbcL_55	4774	-23495,5	5279	-25941,6	65,56	0,0005
mtDNA_186	2376	-39376,4	2656	-43567,2	64,57	0,0004
RDPII_218	40658	-140667,6	45841	-162933,1	70,37	0,0005
ZILLA_500	15893	-80850,9	17588	-89319,8	65,05	0,0009

V. Conclusiones

Hemos expuesto en este artículo un estudio del rendimiento obtenido al paralelizar una adaptación multiobjetivo del algoritmo ABC para inferir árboles filogenéticos en arquitecturas multicore. La propuesta ha sido probada mediante experimentación usando cuatro bases de datos reales de nucleótidos y verificándose el rendimiento al integrar técnicas de paralelismo de grano grueso. Los resultados obtenidos sugieren que nuestra propuesta puede obtener factores de aceleración muy prometedores al ser usada en conjunto de datos grandes en comparación a otras propuestas basadas en técnicas avanzadas de Computación de Alto Rendimiento. Así pues, se pueden llevar a cabo análisis filogenéticos multiobjetivo de manera eficiente gracias al diseño de estrategias algorítmicas que permitan explotar el paralelismo en máquinas multicore. Como líneas de trabajo futuro, plantearemos el estudio de otras técnicas paralelas para realizar análisis filogenético multiobjetivo en conjuntos de datos más grandes. Sistemas híbridos basados en GPU pueden ser analizados para introducir paralelismo de grano fino a nuestra propuesta de grano grueso, permitiendo reducir los tiempos requeridos para la evaluación de soluciones. Otra posible línea de trabajo es el diseño de equipos paralelos de algoritmos bioinspirados para mejorar los resultados filogenéticos presentados.

Agradecimientos

Agradecemos al Gobierno de Extremadura por la ayuda GR10025 proporcionada al grupo de investigación TIC015, así como a la Fundación Valhondo por el apoyo económico ofrecido a Sergio Santander-Jiménez para hacer posible este trabajo.

Referencias

[1] J. Felsenstein, *Inferring Phylogenies*. Sunderland, Massachusetts: Sinauer, ISBN: 0-87893-177-5, 2004.

[2] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. New York: John Wiley & Sons, ISBN: 978-0-471-87339-6, 2001.

[3] D. Karaboga, "An Idea Based on Honey Bee Swarm for Numerical Optimization", Erciyes University, Computer Engineering Department, Tech. Rep. TR06, 2005.

[4] D. Karaboga, y B. Basturk, "A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm", *Journal of Global Optimization*, vol. 39, no. 3, pp. 459-171, 2007.

[5] H. Narasimhan, "Parallel Artificial Bee Colony (PABC) Algorithm", en *World Congress on Nature Biologically Inspired Computing*, 2009, pp. 306-311.

[6] D. Swofford, G. Olsen, P. Waddell, P y D. Hillis, "Phylogenetic Inference", *Molecular Systematics*. Sinauer, 1996, vol. 2, pp. 407-514.

[7] Q. Snell, M. Whiting, M. Clement y D. McLaughlin, "Parallel phylogenetic inference", en *Proc. of the 2000 ACM/IEEE conference on Supercomputing*. IEEE Computer Society, 2000, Article 35.

[8] W. Cancino y A.C.D. Delbem, "A Multi-Objective Evolutionary Approach for Phylogenetic Inference", en *Evolutionary Multi-Criterion Optimization*. LNCS, Berlin / Heidelberg: Springer, 2007, vol. 4403, pp. 428-442.

[9] D.A. Bader, U. Roshan, A. Stamatakis y C.-W. Tseng, "Computational Grand Challenges in Assembling the Tree of Life: Problems and Solutions", *Advances in Computers*, vol. 68, pp. 127-176, 2006.

[10] G. Krawezik, G. Alleon y F. Cappello, "SPMD OpenMP versus MPI on a IBM SMP for 3 Kernels of the NAS Benchmarks", en *ISHPC 2002*. LNCS, Berlin / Heidelberg: Springer, 2002, vol. 2327, pp. 425-436.

[11] N. Weicker, G. Szabo, K. Weicker, y P. Widmayer, "Evolutionary Multiobjective Optimization for Base Station Transmitter Placement with Frequency Assignment", *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 189-203, 2003.

[12] K. Deb, A. Pratap, S. Agarwal y T. Meyarivan, "A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA II", *IEEE Trans. Evol. Comput*, vol. 6, pp. 182-197, 2002.

[13] M. D. Hill y M. R. Marty, "Amdahl's Law in the Multicore Era", *IEEE Computer*, vol. 41, no. 7, pp. 33-38, July 2008.

[14] W. Cancino, L. Jourdan, E.G. Talbi y A.C.D. Delbem, "Parallel Multi-Objective Approaches for Inferring Phylogenies", en *Proc. of EVOBIO'2010*. LNCS, Berlin / Heidelberg: Springer, 2010, vol. 6023, pp. 26-37.

[15] W. Cancino y A.C.D. Delbem, "A Multi-Criterion Evolutionary Approach Applied to Phylogenetic Reconstruction", en *New Achievements in Evolutionary Computation*. InTech, February 2010, pp. 135-156, ISBN: 978-953-307-053-7.