Análisis de Diseños Paralelos Multiobjetivo y Políticas de Planificación en Biología Evolutiva

Sergio Santander-Jiménez
INESC-ID, IST, Universidade de Lisboa,
Lisboa 1000-029, Portugal
sergio.jimenez@tecnico.ulisboa.pt

Miguel A. Vega-Rodríguez
INTIA, Universidad de Extremadura,
Cáceres 10003, España
mavega@unex.es

Leonel Sousa
INESC-ID, IST, Universidade de Lisboa,
Lisboa 1000-029, Portugal
leonel.sousa@ist.utl.pt

***Resumen*—Las metaheurísticas paralelas representan una de las técnicas de mayor popularidad para abordar la resolución de problemas de optimización complejos. Sin embargo, una de las cuestiones principales que surgen al emplear estos métodos viene dada por la aparición potencial de problemas de desequilibrio de carga. De hecho, la complejidad de los problemas de optimización actuales obliga a la adopción de múltiples estrategias de búsqueda condicionales dentro de la metaheurística, lo cual incide en el impacto que implica el desequilibrio de carga en el rendimiento paralelo. Este trabajo analiza diferentes políticas de planificación y diseños algorítmicos para abordar el desequilibrio de carga en la metaheurística *Multiobjective Shuffled Frog-Leaping Algorithm*. Tomando como caso de estudio un problema de biología evolutiva, la reconstrucción filogenética, abordamos la evaluación de tres políticas de planificación en el algoritmo original, así como una alternativa de diseño basada en el uso de contadores. Los resultados obtenidos en cuatro bases de datos reales dan cuenta de la influencia de las aproximaciones estudiadas en los tiempos de sobrecarga, destacando los beneficios de integrar políticas dinámicas y diseños que respeten las mecánicas del algoritmo.**

***Index Terms*—Biología evolutiva, desequilibrio de carga, computación paralela, metaheurísticas multiobjetivo**

I. Introducción

La resolución eficiente de problemas de optimización se ha convertido en una de las líneas de investigación predominantes en multitud de dominios científicos. En contextos reales, es obligatorio el diseño de aproximaciones algorítmicas innovadoras para cumplir los requisitos de calidad de solución y, al mismo tiempo, lidiar con la naturaleza NP-completa de estos problemas. Sin embargo, la dificultad de estos procesos de optimización continúa creciendo con la adopción de formulaciones más realistas, escenarios *Big Data*, múltiples funciones objetivo, etc. Así, es habitual la presencia de tiempos de ejecución elevados que resultan inasumibles incluso al emplear técnicas estocásticas de resolución. Las metaheurísticas paralelas han demostrado aplicabilidad potencial para lidiar con estas cuestiones, obteniendo resultados significativos [1].

A pesar de la naturaleza intrínsecamente paralela de las metaheurísticas basadas en población, estos métodos pueden verse afectados por dos fuentes principales de desequilibrio de carga. Al nivel del diseño metaheurístico, los operadores de búsqueda pueden mostrar distintos requisitos temporales, hecho que puede volverse más remarcable en situaciones de cambio de contexto de explotación a exploración (o viceversa). Al nivel de implementación del problema, las soluciones generadas pueden mostrar características divergentes que influyen, por ejemplo, en los cálculos efectuados en las funciones objetivo, dando lugar a tiempos de evaluación no homogéneos. Estas cuestiones implican un aumento de la sobrecarga y, consecuentemente, un empeoramiento del rendimiento paralelo.

Así, las metaheurísticas paralelas deben considerar este tipo de factores para conseguir una explotación precisa de los recursos hardware. Este trabajo se focaliza en el análisis de distintas políticas de planificación y diseños para abordar problemas de desequilibrio de carga. Adoptamos como caso de estudio un problema de biología evolutiva: la inferencia de relaciones ancestrales entre especies [2]. En escenario monoobjetivo, existen propuestas como IQ-TREE [3] y GARLI [4] que han dado cuenta de los beneficios de emplear computación de altas prestaciones y métodos bioinspirados de búsqueda en este problema. Las aproximaciones multiobjetivo también representan una tendencia significativa en el caso de datos de ADN [5], al contribuir potencialmente a la resolución de problemas de incongruencia en contextos reales.

Abordaremos este problema biológico desde la perspectiva multiobjetivo, usando una metaheurística basada en población denominada *Multiobjective Shuffled Frog-Leaping Algorithm* (MO-SFLA) [6]. Bajo implementaciones MPI+OpenMP, examinaremos el impacto de distintas políticas de planificación (estática, guiada y dinámica) en el rendimiento paralelo del diseño algorítmico original, así como diseños alternativos de la aproximación. El análisis planteado será efectuado sobre cuatro instancias reales del problema basadas en secuencias de proteínas, las cuales presentan un nivel superior de complejidad con respecto a otros tipos de datos biológicos [7]. La bondad de los resultados obtenidos será estudiada mediante comparativas con otros métodos del estado del arte.

Este artículo se organiza del siguiente modo. La Sección II detalla la formulación del problema estudiado. La Sección III describe diseños y políticas de planificación para MO-SFLA, examinándose resultados en la Sección IV. Finalmente, la Sección V destaca conclusiones y líneas futuras de trabajo.

Este trabajo ha sido parcialmente financiado por la AEI (Agencia Estatal de Investigación, España) y el FEDER (Fondo Europeo de Desarrollo Regional, UE), bajo el proyecto TIN2016-76259-P (proyecto PROTEIN), así como por la FCT (Fundação para a Ciência e a Tecnologia, Portugal) bajo los proyectos UID/CEC/50021/2013 y LISBOA-01-0145-FEDER-031901 (PTDC/CCI-COM/31901/2017, HiPErBio). Sergio Santander-Jiménez agradece a la FCT el soporte recibido a través del contrato postdoctoral SFRH/BPD/119220/2016.

II. Formulación del Problema

La inferencia de relaciones ancestrales entre organismos representa uno de los problemas más importantes en biología evolutiva [2]. Dicho problema involucra el estudio de un alineamiento múltiple de secuencias de tamaño $N \times M$ (donde N es el número de secuencias y M la longitud de secuencia), el cual es procesado para identificar patrones de divergencia y similitud. Esta información es utilizada para inferir organismos ancestrales hipotéticos y definir el curso de la evolución a través de una estructura arborescente, conocida como árbol filogenético $T = (V, E)$. En particular, los nodos internos del conjunto de nodos V describen los ancestros potenciales cuya evolución dio lugar a los organismos caracterizados en el alineamiento de entrada, organismos que se localizan en los nodos hoja. La definición de relaciones evolutivas se efectúa a través del conjunto de ramas E, donde se definen los enlaces ancestro-descendiente entre nodos emparentados.

La aplicación de procedimientos de optimización en este problema tienen por objeto identificar el árbol filogenético que maximice o minimice un determinado criterio de optimalidad biológico, lo cual implica la exploración de un espacio de filogenias S. La principal cuestión que surge en este contexto viene dada por el crecimiento exponencial de S con el número de secuencias N, definiéndose el número de posibles soluciones candidatas en S de acuerdo al doble factorial $(2N-5)!!$ [2]. Otros factores, tales como la longitud de secuencia M, el tipo de datos contenido en el alineamiento (por ejemplo, aminoácidos) y la consideración conjunta de múltiples criterios de optimalidad, contribuyen al elevado coste computacional del problema. Todo ello impide la aplicación de aproximaciones serie tradicionales.

En este artículo, afrontamos estas cuestiones mediante aproximaciones metaheurísticas paralelas. La formulación adoptada del problema involucra la optimización de dos funciones objetivo: parsimonia y verosimilitud. Por un lado, la parsimonia cuantifica el número de cambios observados entre las secuencias de nodos emparentados. El objetivo radica en inferir la hipótesis evolutiva más simple, dando prioridad a la solución $T = (V, E)$ que minimice el número de cambios medidos a través del valor de parsimonia $P(T)$ [2]:

$$P(T) = \sum_{i=1}^{M} \sum_{(u,v)\in E} C(u_i, v_i), \tag{1}$$

donde $(u, v) \in E$ representa la rama de la topología que conecta dos nodos $u, v \in V$, u_i y v_i son los valores de estado del i-ésimo carácter de las secuencias correspondientes a u y v, y $C(u_i, v_i)$ indica las mutaciones observadas entre u y v ($C(u_i, v_i)$=1 si $u_i \neq v_i$ y $C(u_i, v_i)$=0 en caso contrario).

Por su parte, la función de verosimilitud establece una medida probabilística de la plausibilidad de que el árbol filogenético evaluado haya dado lugar efectivo a la diversidad observada en el alineamiento de entrada. Los cálculos de verosimilitud se realizan conforme a modelos de evolución que definen las probabilidades de observar mutaciones desde cada posible valor de estado de carácter a cualquier otro. Bajo esta función, se da prioridad a la solución $T = (V, E)$ que maximice el valor de verosimilitud $L(T)$ [2]:

$$L(T) = \prod_{i=1}^{M} \sum_{x,y\in\Lambda} \pi_x \left[P_{xy}(t_{ru}) L_p\left(u_i = y\right)\right] \times \left[P_{xy}\left(t_{rv}\right) L_p\left(v_i = y\right)\right], \tag{2}$$

donde Λ es el alfabeto de estados de carácter (aminoácidos en el caso de secuencias de proteínas), π_x la probabilidad estacionaria de observar el valor de estado x, $r \in V$ la raíz de T con hijos $u, v \in V$, $P_{xy}(t)$ la probabilidad de mutación de x a otro estado y en un tiempo t (siendo t_{ru} y t_{rv} la longitud de las ramas $(r, u), (r, v) \in E$), y $L_p(u_i = y)$, $L_p(v_i = y)$ las verosimilitudes parciales de observar y en u_i y v_i.

III. Descripción de Métodos

MO-SFLA es una metaheurística que integra mecanismos multiobjetivo en el método *frog leaping optimization* [8]. En cada iteración, los *popSize* individuos que conforman la población son ordenados (según sus valores de fitness) y distribuidos uniformemente en *m* particiones denominadas memeplexes. Cada memeplex contiene *n=popSize/m* individuos, cuya evolución se produce separadamente durante n_l pasos de aprendizaje. Una vez todos los memeplexes han sido procesados, son combinados para permitir una compartición de información entre todos los memeplexes, actualizando la población para la siguiente generación.

Para la adaptación al problema abordado, MO-SFLA incluye una representación del individuo basada en matrices de distancia filogenéticas. Una solución candidata vendrá codificada por una matriz simétrica δ que contiene en cada entrada $\delta[x, y]$ la distancia evolutiva entre los organismos x e y. El mapeo de soluciones desde este espacio de decisión matricial al espacio de filogenias se lleva a cabo mediante el método de reconstrucción de árboles filogenéticos BIONJ [2]. La etapa de inicialización de la población considerará matrices de distancia calculadas a partir de topologías filogenéticas iniciales generadas mediante técnicas de *bootstraping* [2].

El Pseudocódigo 1 detalla el diseño original de MO-SFLA, el cual realiza el proceso de optimización hasta que un determinado criterio de parada (en nuestro caso, un máximo de evaluaciones) es satisfecho. Al comienzo de cada generación, los individuos de la población son ordenados según su calidad multiobjetivo (línea 7 en el Pseudocódigo 1), manteniendo las *popSize* soluciones más prometedoras conforme a sus valores de ranking Pareto (*fast non-dominated sort*) y densidad (distancia de *crowding*) [9]. Tras la ordenación, se definen los memeplexes mediante la distribución de individuos de manera iterativa, asignando el individuo i-ésimo al memeplex $i\,\%m$.

La generación de nuevas soluciones candidatas dentro de cada memeplex se realiza mediante la definición de distintos operadores de búsqueda (líneas 17-26), que son aplicados según el estado actual del proceso de optimización. Dado un memeplex Mem_i, generamos una nueva solución P'_{new} conforme a la siguiente formulación:

Pseudocódigo 1 MO-SFLA Paralelo

```
1:  MPI_Init /* inicializando proceso MPI #i */
2:  #pragma omp parallel num_threads (num_hilos)
3:  mientras ! criterio de parada (maxEval) hacer
4:      /* Definición y asignación de memeplexes */
5:      #pragma omp single
6:      si proceso maestro entonces
7:          {Mem_1 ... Mem_m} ← Ordenación y Distribución (P, popSize)
8:          BestGlobal ← Identificar Mejor Global ({Mem_1 ... Mem_m})
9:          /* ∀i : i = 2 a m */
10:         MPI_Send (Mem_i, n, i), MPI_Send (BestGlobal, 1, i)
11:     si no
12:         MPI_Recv (Mem_i, n, master_id), MPI_Recv (BestGlobal, 1, master_id)
13:     fin si
14:     BestLocal ← Identificar Mejor Local (Mem_i)
15:     /* Tareas paralelas: evolución del memeplex asignado */
16:     #pragma omp for
17:     para j = 1 to n_l hacer
18:         P'_new ← Aprendizaje Mejor Local (BestLocal, Mem_i(n−j))
19:         si ! P'_new ≻ Mem_i(n−j) entonces
20:             P'_new ← Aprendizaje Mejor Global (BestGlobal, Mem_i(n−j))
21:             si ! P'_new ≻ Mem_i(n−j) entonces
22:                 P'_new ← Búsqueda Local (Mem_i(n−j))
23:             fin si
24:         fin si
25:         Mem_i(n−j) ← P'_new
26:     fin para
27:     /* Entrega de resultados */
28:     #pragma omp single
29:     si proceso maestro entonces
30:         /* ∀i : i = 2 a m */
31:         MPI_Recv (Mem_i, n, i)
32:         P ← Mezclar Memeplexes (P, {Mem_1 ... Mem_m})
33:         ParetoFront ← Actualizar Frente de Pareto (P)
34:     si no
35:         MPI_Send (Mem_i, n, master_id)
36:     fin si
37: fin mientras
38: MPI_Finalize /* finalizando proceso MPI #i */
```

$$D_{xy} = rand() \cdot (Ref.\delta[x,y] - Mem_{ij}.\delta[x,y]), \quad (3)$$

$$P'_{new}.\delta[x,y] = Mem_{ij}.\delta[x,y] + D_{xy}, \quad (4)$$

donde *rand()* es un número aleatorio de una distribución uniforme en el rango $[0, 1]$ y Mem_{ij} el individuo en Mem_i procesado en el paso de aprendizaje j. *Ref* representa una solución que es tomada como referencia durante este proceso. Inicialmente, se toma como *Ref* el mejor individuo local de Mem_i (línea 18). En caso de que la solución resultante P'_{new} no mejore a Mem_{ij}, se repite el proceso anterior tomando como *Ref* el mejor individuo global de toda la población (línea 20). Si sigue sin observarse mejora, se aplica una búsqueda local basada en los operadores topológicos *nearest neighbour interchange* y *subtree pruning and regrafting* [2] (línea 22).

Al final de la generación (líneas 32, 33), los memeplexes son reunificados para actualizar la estructura de la población, la cual es usada a su vez para identificar soluciones no dominadas y así almacenarlas en la estructura *ParetoFront*. Al satisfacerse el criterio de parada, dicha estructura contendrá las soluciones no dominadas encontradas a lo largo de la ejecución, representando la salida del algoritmo.

III-A. Diseños Paralelos y Políticas de Planificación

MO-SFLA presenta dos características clave que hacen a esta metaheurística adecuada para su ejecución en clusters multicore: 1) el procesamiento de memeplexes puede efectuarse de manera independiente de una memeplex a otro; y 2) la generación de soluciones candidatas dentro del mismo memeplex no presenta dependencias de un paso de aprendizaje (iteración del bucle) a otro. Estos dos niveles son apropiados para su explotación mediante MPI+OpenMP [10], [11].

En cada generación, un proceso maestro se encargará inicialmente de la gestión de la población y la definición de memeplexes, distribuyendo adecuadamente los individuos. Los memeplexes resultantes serán asignados a distintos procesos trabajadores mediante las funciones MPI `MPI_Send` y `MPI_Recv` (línea 10 en el Pseudocódigo 1 para el lado emisor - maestro, y línea 12 para el lado receptor - trabajador). Concluida la comunicación, cada proceso (incluyendo el maestro) realizará el procesamiento de los memeplexes asignados (líneas 15-26), usando la directiva `#pragma omp for` para distribuir las iteraciones del bucle entre los hilos de ejecución. Terminado el procesamiento de memeplexes, los resultados obtenidos son comunicados mediante funciones MPI al proceso maestro (líneas 31 -maestro, y 35 - trabajador), efectuándose entonces las tareas finales de la generación.

En este diseño, los procesos se ven afectados por distintas fuentes de desequilibrio de carga. En primer lugar, la metaheurística plantea hasta tres estrategias de búsqueda diferentes que pueden ser ejecutadas en una misma iteración del bucle paralelo definido en la línea 17. Así, cada iteración puede implicar un número variable de operaciones en conformidad con el éxito de las estrategias aplicadas. Además, la generación de una nueva solución candidata involucra dos mecanismos específicos del problema: la reconstrucción del árbol y el cómputo de las funciones objetivo. Estas operaciones se caracterizan no solo por el hecho de incluir cálculos computacionalmente costosos, sino también por presentar divergencias en tiempo de ejecución según las características de la solución candidata.

Para afrontar dichos problemas de desequilibrio de carga, consideraremos primero el uso de las políticas de planificación de bucles paralelos definidas en el estándar OpenMP. Concretamente, examinamos tres aproximaciones diferentes:

1. *Estática*. Las iteraciones son divididas en bloques que se asignan de manera estática a cada hilo en modo *round-robin* según el identificador de hilo. Dado que esto no permite la resolución de desequilibrios de carga potenciales (al tener cada hilo sus iteraciones asignadas a priori), usaremos esta política como punto de referencia.
2. *Dinámica*. En esta aproximación, los hilos solicitan gradualmente nuevas iteraciones al terminar las que han ido manejado previamente. Así, los hilos más rápidos pueden comenzar el procesamiento de nuevos bloques inmediatamente tras acabar su carga de trabajo inicial.
3. *Guiada*. De manera similar a la aproximación dinámica, la planificación guiada permite la asignación gradual de iteraciones a los hilos. Se asignan nuevas iteraciones mediante bloques de tamaño proporcional al número de iteraciones aún no asignadas dividido entre el número de hilos, reduciendo estos bloques de manera exponencial hasta un mínimo de 1 (configuración estándar).

Pseudocódigo 2 MO-SFLA Paralelo Basado en Contadores

```
1:  MPI_Init /* inicializando proceso MPI #i */
2:  #pragma omp parallel num_threads (num_hilos)
3:  mientras ! criterio de parada (maxEval) hacer
4:      #pragma omp single
5:          BestGlobal, {Mem_1 ... Mem_m} ← Gestionar Población (P, popSize)
6:          Asignación Memeplex-Proceso (BestGlobal, Mem_i, i) /*∀i : i = 2 a m*/
7:          Best_local ← Identificar Mejor Local (Mem_i)
8:      #pragma omp for
9:      para j = 1 to n_l hacer
10:         switch (Mem_i(n−j).counter)
11:             caso 0: P'_new ← Aprendizaje Mejor Local (Best_local, Mem_i(n−j))
12:             caso 1: P'_new ← Aprendizaje Mejor Global (Best_global, Mem_i(n−j))
13:             caso 2: P'_new ← Búsqueda Local (Mem_i(n−j))
14:         si P'_new ≻ Mem_i(n−j) || Mem_i(n−j).counter == 2 entonces
15:             Mem_i(n−j) ← P'_new, Mem_i(n−j).counter ← 0
16:         si no
17:             Mem_i(n−j).counter ← Mem_i(n−j).counter + 1
18:         fin si
19:     fin para
20:     #pragma omp single
21:         Comunicación de Resultados (Mem_i, master_id) /*∀i : i = 2 a m*/
22:         P, ParetoFront ← Tareas Finales de la Generación (P, {Mem_1 ... Mem_m})
23: fin mientras
24: MPI_Finalize /* finalizando proceso MPI #i */
```

Una alternativa a esta aproximación radica en la adaptación del diseño para que cada iteración del bucle de procesamiento de memeplexes ejecute solo una de las estrategias de búsqueda definidas. Esta idea viene plasmada en el Pseudocódigo 2, la cual se basa en el empleo de contadores para medir el número de intentos en que un individuo de la población no ha sido mejorado. En función del valor de este contador, se decide la estrategia de búsqueda a ser aplicada sobre dicho individuo (líneas 10-13 del Pseudocódigo 2). Dado un individuo Mem_{ij}, un valor de 0 en su contador implicará la generación de la nueva solución empleando el mejor individuo local de Mem_i, mientras que un valor de 1 conllevará el empleo del mejor individuo global y un valor de 2 la búsqueda local. En caso de observarse mejora, el contador asociado será reinicializado y la solución almacenada en Mem_{ij} (línea 15). En caso contrario, la solución generada es descartada y el contador de Mem_{ij} incrementado (línea 17). De esta manera, en la siguiente generación el proceso trabajador efectuará un nuevo intento sobre Mem_{ij} considerando una estrategia de búsqueda diferente. Como en la versión original, se pueden emplear las políticas de planificación de OpenMP para complementar el mecanismo de balanceo basado en contadores.

IV. Resultados Experimentales

En esta sección abordamos la evaluación experimental de los diseños paralelos y políticas de planificación considerados para MO-SFLA. Para ello, examinaremos el rendimiento paralelo obtenido en cuatro bases de datos reales de proteínas, las cuales son descritas en la Tabla I. La experimentación fue efectuada sobre 4 nodos de cómputo de una infraestructura clúster con interconexión Gigabit Ethernet. Estos nodos están compuestos por procesadores AMD Opteron 6174 de 12 cores a 2,2GHZ (un total de 48 cores disponibles) con 12MB de memoria caché L3 y 16GB de RAM DDR3, Ubuntu 14.04LTS como sistema operativo y el compilador GCC 5.2.1.

De acuerdo a las características del clúster empleado y los estudios paramétricos efectuados, los parámetros de entrada de MO-SFLA fueron establecidos a los siguientes valores: *popSize* = 128, m = 4, n = 32, n_l = 32, *maxEval* = 10000.

Tabla I
Conjuntos de Datos de Proteínas Usados en la Experimentación

Instancia	N	M	Organismo	Modelo Evolutivo [2]
M88x3329	88	3329	Hongos termófilos [12]	LG+Γ
M187x814	187	814	Hongos micorrícicos [13]	LG+Γ
M260x1781	260	1781	Beta vulgaris [14]	JTT+Γ
M355x1263	355	1263	Hemiascomicetos [15]	LG+Γ

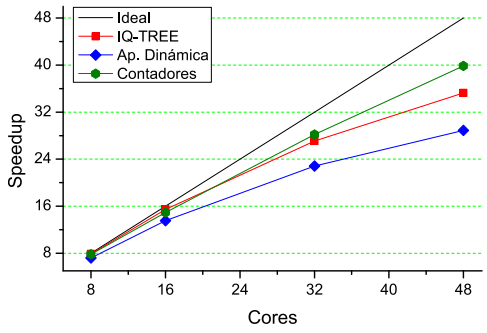
Tabla II
Comparativa Paralela (Speedups *SU* y Eficiencia *EF*)

	Estática		Dinámica		Guiada		Contadores	
Cores	SU	EF(%)	SU	EF(%)	SU	EF(%)	SU	EF(%)
M88x3329 (T_{serie} = 57111,63 segundos)								
8	6,93	86,59	**7,21**	**90,10**	7,09	88,65	**7,81**	**97,67**
16	11,58	72,38	**13,53**	**84,57**	13,31	83,18	**14,94**	**93,38**
32	19,10	59,69	**22,83**	**71,36**	22,34	69,82	**28,18**	**88,06**
48	26,18	54,54	**28,88**	**60,17**	27,84	57,99	**39,86**	**83,04**
M187x814 (T_{serie} = 44171,92 segundos)								
8	6,16	77,06	**6,54**	**81,78**	6,46	80,74	**7,70**	**96,27**
16	10,70	66,88	**12,18**	**76,12**	11,96	74,78	**14,75**	**92,19**
32	18,58	58,06	**21,85**	**68,28**	20,99	65,60	**27,71**	**86,59**
48	23,96	49,92	**28,33**	**59,03**	26,04	54,25	**39,66**	**82,63**
M260x1781 (T_{serie} = 66748,33 segundos)								
8	7,37	92,16	**7,69**	**96,18**	7,55	94,33	**7,86**	**98,28**
16	12,75	79,69	**14,18**	**88,61**	13,70	85,61	**15,65**	**97,81**
32	20,45	63,91	**23,29**	**72,78**	22,51	70,36	**29,20**	**91,25**
48	26,36	54,92	**30,21**	**62,94**	27,90	58,11	**40,56**	**84,49**
M355x1263 (T_{serie} = 96219,62 segundos)								
8	6,79	84,90	**7,47**	**93,40**	7,22	90,21	**7,92**	**98,95**
16	11,92	74,50	**13,76**	**86,03**	13,69	85,56	**15,68**	**98,00**
32	21,96	68,63	**25,59**	**79,97**	23,77	74,28	**29,87**	**93,34**
48	29,30	61,04	**32,97**	**68,69**	30,80	64,17	**42,63**	**88,82**

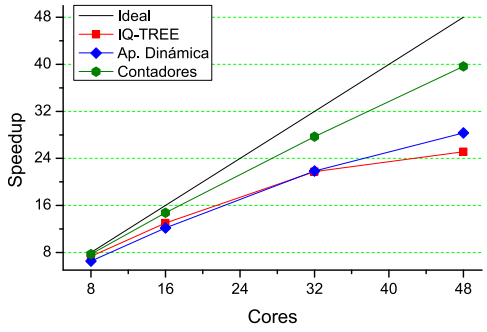
IV-A. Evaluación del Rendimiento Paralelo

Con objeto de examinar el rendimiento paralelo, emplearemos las métricas paralelas de aceleración o *speedup* y eficiencia [11]. Consideraremos los resultados temporales medianos (procedentes de 11 ejecuciones independientes por experimento) obtenidos sobre distintos tamaños del sistema (8, 16, 32 y 48 cores). La Tabla II presenta los resultados observados, así como los tiempos de la versión serie de MO-SFLA T_{serie} usados como referencia en el cálculo de las métricas paralelas.

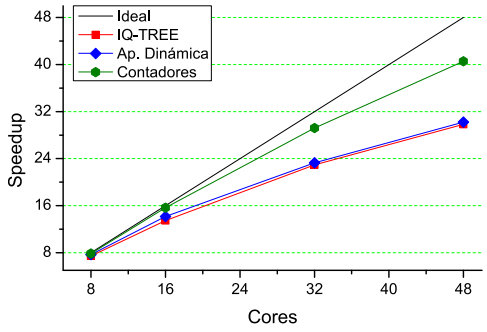
Centrándonos en primer lugar en los resultados del diseño original (columnas 2, 3 - versión estática, 4, 5 - dinámica, y 6, 7 - guiada), es posible observar el efecto del desequilibrio de carga en el rendimiento de la versión inicial estática. De hecho, la política estática no resulta adecuada para superar el umbral de 50 % de eficiencia en el caso de instancias como M187x814 con 48 cores, lo cual sugiere una baja utilización de los recursos hardware disponibles. Por su parte, el empleo de políticas de planificación dinámica da lugar a una reducción considerable en la sobrecarga de la versión estática, reduciendo los tiempos de espera a nivel de hilos hasta un 33,6 % (para la instancia M355x1263) y los de sincronización inter-proceso hasta un 45,4 % (M187x814). Esto implica que, para el diseño original de MO-SFLA, la planificación dinámica consigue el



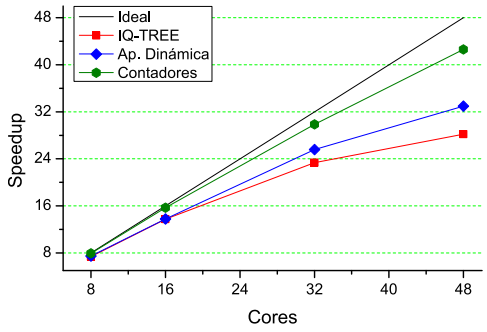
(a) M88x3329



(b) M187x814



(c) M260x1781



(d) M355x1263

Figura 1. Comparativas de Rendimiento Paralelo (MO-SFLA Original Dinámico, MO-SFLA Basado en Contadores e IQ-TREE)

rendimiento paralelo más satisfactorio en la comparativa, con eficiencias medias de 90,4 % (8 cores), 83,8 % (16 cores), 73,1 % (32 cores) y 62,7 % (48 cores). En esta comparativa, la política guiada representa un escenario intermedio, con eficiencias medias de 88,5 % (8 cores), 82,3 % (16 cores), 70,0 % (32 cores) y 58,6 % (48 cores). Con respecto a la versión estática, la planificación guiada reduce hasta un 13,7 % (M355x1263) los tiempos de espera entre hilos, así como hasta un 33,3 % (M187x814) la sincronización entre procesos. Sin embargo, estos resultados no igualan la calidad de la política dinámica, la cual representa la aproximación más adecuada en el contexto del diseño original de MO-SFLA.

Considerando estos resultados, hemos aplicado la política dinámica a su vez en el diseño basado en contadores de MO-SFLA. Los *speedups* y eficiencias obtenidos son mostrados en las columnas 8, 9 de la Tabla II. Conforme a estos resultados, el mecanismo alternativo estudiado consigue mejorar de manera significativa la escalabilidad paralela del diseño original, consiguiendo factores de aceleración medios de 7,8x (8 cores), 15,1x (16 cores), 28,7x (32 cores) y 40,7x (48 cores). A través de un mayor equilibrado de tareas en el bucle paralelo de procesamiento de memeplexes, esta alternativa permite reducir los tiempos de espera entre hilos hasta un 72,1 % (M88x3329), dando lugar a una reducción de hasta 92,0 % (M187x814) en los tiempos de sincronización entre procesos. Como resultado, el diseño basado en contadores obtiene una eficiencia media de 84,7 % para 48 cores, representando una mejora notable con respecto al 62,7 % de la versión original dinámica.

Para validar la bondad de estos resultados, la Figura 1 presenta una comparativa de rendimiento paralelo entre el diseño original con planificación dinámica de MO-SFLA, el diseño basado en contadores, y la herramienta filogenética paralela IQ-TREE (la cual integra técnicas MPI+OpenMP) [3]. Estos resultados dan cuenta de que el diseño original dinámico obtiene una mejoría con respecto a IQ-TREE a partir de escenarios de 32 / 48 cores para M187x814, así como en M260x1781 y, especialmente, M355x1263. Por su parte, el diseño basado en contadores presenta las mejores propiedades de escalabilidad de la comparativa, mejorando de manera significativa los resultados de IQ-TREE en todos los escenarios de evaluación considerados. Todo ello redunda en la relevancia de la estrategia adoptada para minimizar problemas de balanceo de carga en el diseño algorítmico de MO-SFLA.

IV-B. Resultados Biológicos

A continuación abordamos la validación de calidad de solución del método, usando resultados medianos de 31 ejecuciones independientes por instancia. En primer lugar, examinaremos el rendimiento multiobjetivo mediante el empleo de la métrica de hipervolumen (usando los puntos de referencia definidos en [6]). En este contexto, es preciso señalar que, para este problema, no se observaron diferencias estadísticamente significativas entre las muestras de hipervolumen de la versión original de MO-SFLA y la versión basada en contadores, de acuerdo a los tests de ANOVA / Wilcoxon-Mann-Whitney [16]. La Tabla III introduce los valores medianos de hipervolumen observados, en comparación a los obtenidos por el algoritmo estándar NSGA-II [9]. MO-SFLA es capaz de obtener valores de hipervolumen en el rango 75,39 % - 81,99 %, en comparación a los hipervolúmenes de 74,95 % -

Tabla III
Comparativas Multiobjetivo: Hipervolumen

Instancia	MO-SFLA	NSGA-II
M88x3329	**75,39±0,04**	74,95±0,12
M187x814	**77,58±0,58**	76,20±0,58
M260x1781	**80,71±0,23**	77,46±0,91
M355x1263	**81,99±0,19**	80,52±0,22

Tabla IV
Comparativas Filogenéticas: Parsimonia

Instancia	MO-SFLA	TNT
M88x3329	**33490**	**33490**
M187x814	**29847**	**29847**
M260x1781	**43529**	**43529**
M355x1263	**54823**	**54823**

Tabla V
Comparativas Filogenéticas: Verosimilitud

Instancia	MO-SFLA	IQ-TREE	GARLI
M88x3329	**-149094,84**	**-149094,84**	-149111,00
M187x814	**-133871,90**	-133871,96	-133876,72
M260x1781	**-163895,79**	-163899,10	-163982,64
M355x1263	**-231186,34**	-231301,11	-231859,24

80,52 % mostrados por NSGA-II. Estos resultados sugieren que MO-SFLA da lugar a la obtención de frentes de Pareto de calidad significativa en todas las instancias analizadas.

Las Tablas IV y V introducen comparativas de calidad biológica con otros métodos filogenéticos del estado del arte. En la Tabla IV, comparamos dicha calidad desde la perspectiva de parsimonia con respecto a la herramienta heurística TNT [17]. Según este criterio, el método multiobjetivo MO-SFLA es capaz de mantener, en todas las instancias analizadas, la calidad filogenética obtenida por TNT. Por su parte, la Tabla V presenta una comparativa de resultados de verosimilitud con IQ-TREE [3] y el algoritmo evolutivo GARLI [4]. Conforme a los resultados obtenidos, MO-SFLA da lugar a una mejora en verosimilitud en las instancias con mayor número de secuencias, confirmando la bondad de las soluciones generadas.

V. Conclusiones

Este trabajo ha versado sobre la evaluación de diseños paralelos y políticas de planificación para solucionar problemas de balanceo de carga en métodos metaheurísticos. Usando como caso de estudio una metaheurística multiobjetivo para reconstrucción filogenética, MO-SFLA, hemos identificado las distintas fuentes de desequilibrio de carga presentes en su diseño algorítmico. Para afrontarlos, hemos considerado el uso de distintas políticas de planificación del estándar OpenMP (estática, dinámica y guiada), así como una aproximación algorítmica alternativa basada en el empleo de contadores para equilibrar la carga de trabajo de los hilos de ejecución.

Hemos evaluado experimentalmente las técnicas planteadas en un clúster multicore compuesto por 48 núcleos de procesamiento, empleando para ello cuatro instancias del problema basadas en datos de secuencias de proteínas. Para el diseño original, la aproximación dinámica dio lugar a un rendimiento más satisfactorio que el resto de políticas, mejorando la eficiencia media del algoritmo de 55,2 % (versión estática) a 62,7 % para 48 cores. Además, la introducción del mecanismo alternativo de balanceo basado en contadores permitió ir un paso más allá en la explotación de los recursos hardware, demostrando eficiencias hasta de 88,8 % sin afectar significativamente a la calidad de los frentes de Pareto generados.

Como trabajo futuro, pretendemos afrontar la explotación de recursos de computación heterogénea para acelerar el método propuesto, explorando alternativas dependientes e independientes el problema. Se estudiarán a su vez estrategias adaptativas para mejorar las capacidades de búsqueda en espacios de decisión difíciles de abordar, así como la evaluación del diseño en otros problemas bioinformáticos.

Referencias

[1] E. Alba, G. Luque, and S. Nesmachnow, "Parallel metaheuristics: recent advances and new trends," *International Transactions in Operational Research*, vol. 20, no. 1, pp. 1–48, 2013.

[2] P. Lemey, M. Salemi, and A.-M. Vandamme, *The Phylogenetic Handbook: a Practical Approach to Phylogenetic Analysis and Hypothesis Testing*. Cambridge: Cambridge Univ. Press, 2009.

[3] L. Nguyen, H. Schmidt, A. Haeseler, and B. Minh, "IQ-TREE: A fast and effective stochastic algorithm for estimating maximum likelihood phylogenies," *Mol. Biol. Evol.*, vol. 32, no. 1, pp. 268–274, 2015.

[4] A. L. Bazinet, D. J. Zwickl, and M. P. Cummings, "A Gateway for Phylogenetic Analysis Powered by Grid Computing Featuring GARLI 2.0," *Systematic Biology*, vol. 63, no. 5, pp. 812–818, 2014.

[5] W. Cancino, L. Jourdan, E.-G. Talbi, and A. C. B. Delbem, "Parallel Multi-Objective Approaches for Inferring Phylogenies," in *Proc. of EVOBIO'2010*, ser. LNCS, vol. 6023. Springer, 2010, pp. 26–37.

[6] S. Santander-Jiménez, M. A. Vega-Rodríguez, and L. Sousa, "Multi-objective Frog-Leaping Optimization for the Study of Ancestral Relationships in Protein Data," *IEEE Trans. Evol. Comput.*, pp. 1–14 (DOI: 10.1109/TEVC.2017.2 774 599), 2018.

[7] H. Matsuda, H. Yamashita, and Y. Kaneda, "Molecular Phylogenetic Analysis using both DNA and Amino Acid Sequence Data and Its Parallelization," *Genome Informatics*, vol. 5, pp. 120–129, 1994.

[8] A. Sarkheyli, A. M. Zain, and S. Sharif, "The role of basic, modified and hybrid shuffled frog leaping algorithm on optimization problems: a review," *Soft Computing*, vol. 19, no. 7, pp. 2011–2038, 2015.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multi–Objective Genetic Algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[10] W. Gropp, W. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message Passing Interface. 3rd edition*. Cambridge, MA, USA: The MIT Press, 2014.

[11] R. van der Pas, E. Stotzer, and C. Terboven, *Using OpenMP - The Next Step*. Cambridge, MA, USA: The MIT Press, 2017.

[12] I. Morgenstern *et al.*, "A molecular phylogeny of thermophilic fungi," *Fungal Biology*, vol. 116, no. 4, pp. 489–502, 2012.

[13] A. Kovalchuk, A. Kohler, F. Martin, and F. O. Asiegbu, "Diversity and evolution of ABC proteins in mycorrhiza-forming fungi," *BMC Evolutionary Biology*, vol. 15, no. 249, pp. 1–19, 2015.

[14] R. Stracke *et al.*, "Genome-wide identification and characterisation of R2R3-MYB genes in sugar beet (Beta vulgaris)," *BMC Plant Biology*, vol. 14, no. 249, pp. 1–17, 2014.

[15] P. J. Dias and I. Sá-Correia, "The drug:H+ antiporters of family 2 (DHA2), siderophore transporters (ARN) and glutathione:h+antiporters (GEX) have a common evolutionary origin in hemiascomycete yeasts," *BMC Genomics*, vol. 14, no. 901, pp. 1–22, 2013.

[16] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures. 5th edition*. NY, USA: Chapman & Hall/CRC, 2011.

[17] P. A. Goloboff and S. A. Catalano, "TNT version 1.5, including a full implementation of phylogenetic morphometrics," *Cladistics*, vol. 32, no. 3, pp. 221–238, 2016.