Metaheurística Multiobjetivo Híbrida para el Alineamiento Múltiple de Secuencias

Álvaro Rubio-Largo[1], Miguel A. Vega-Rodríguez[2], Sergio Santander-Jiménez[2]

[1] NOVA Information Management School (Universidade NOVA de Lisboa, Portugal)
[2] Dpto. de Tecnología de los Computadores y de las Comunicaciones
(Universidad de Extremadura, España)
{arl, mavega, sesaji}@unex.es

Resumen En los últimos 25 años el Alineamiento Múltiple de Secuencias ha sido un problema clave en diversas áreas de la biología computacional. Este problema consiste en el alineamiento de al menos tres secuencias biológicas, considerado como un problema NP-completo. En este trabajo proponemos el uso de Computación Evolutiva y Optimización Multiobjetivo para resolver este problema, más concretamente, se ha desarrollado una versión multiobjetivo de una metaheurística memética. Con el fin de demostrar la efectividad y precisión de nuestra propuesta, se presenta un estudio comparativo entre nuestra metaheurística y los alineadores de mayor relevancia publicados en la literatura. A la vista de los resultados, podemos afirmar que el nuestra propuesta es capaz de proporcionar alineamientos más precisos y conservativos que otros alineadores publicados en la literatura.

1. Introducción

El Alineamiento Múltiple de Secuencias (Multiple Sequence Alignment, MSA) [1] consiste en alinear simultáneamente tres o más secuencias biológicas (nucleótidos o aminoácidos). Este alineamiento será capaz de determinar residuos en secuencias homólogas, o lo que es lo mismo, encontrar ancestros comunes entre las secuencias a alinear. Uno de los principales propósitos del alineamiento múltiple es detectar relaciones biológicas entre diversas secuencias, lo cuál es crítico a la hora de inferir relaciones filogenéticas entre grupos de organismos [5]. El alineamiento múltiple de secuencias fue definido como un problema de optimización NP-completo [20], cuya complejidad es $O(k2^k L^k)$, siendo k el número de secuencias de entrada y L la longitud de la secuencia más larga.

Diferentes heurísticas han sido publicadas en la literatura con el fin de resolver este problema en un tiempo razonable. Estas heurísticas pueden dividirse en 3 grupos: *progresivas*, *basadas en consistencia* e *iterativas*. Los métodos *progresivos* más importantes son: Clustal W [18], PRANK [12], Fast Statistical Alignment (FSA) [2], Kalign [10], DIALIGN-TX [17] y Clustal Ω [16]. Estos métodos comienzan calculando una matriz de distancia por cada par de secuencias. A partir de la matriz de distancia se utiliza un algoritmo de agrupamiento

Actas de la XVII Conferencia de la Asociación Española para la Inteligencia Artificial, pp. 315-324

para construir el árbol guía, uno de los algoritmos más conocidos para crear el árbol guía es UPGMA (Unweighted Pair Group Method with Arithmetic Mean). Tras eso, en el alineamiento final se alinean *progresivamente* cada una de las secuencias siguiendo las indicaciones del árbol guía. La principal desventaja de los métodos *progresivos* se encuentra en la posibilidad de introducir algún gap erróneo al principio, ya que este será propagado al alineamiento final. El segundo grupo incluye los métodos *basados en consistencia*. Estos métodos comienzan construyendo una base de datos local y global de alineamientos por cada par de secuencias, la cuál será de gran ayuda para construir un alineamiento múltiple preciso. Entre los métodos *basados en consistencia* más importantes podemos encontrar: Tree-based Consistency Objective Function For alignment Evaluation (T-Coffee) [13], PROBabilistic CONSistency-based multiple sequence alignment (ProbCons) [4], ProbAlign [15] y MSAProbs [11]. Por último, encontramos los métodos *iterativos*. La metodología seguida por estos métodos comienza realizando un alineamiento *progresivo*, a continuación realizan varias iteraciones con el objetivo de corregir cualquier gap erróneo introducido en la fase *progresiva*. Este proceso *iterativo* se repite hasta que no se puede mejorar más la calidad del alineamiento o hasta que se realizan un número predefinido de iteraciones. Entre los métodos *iterativos* más conocidos están: MUltiple Sequence Comparison by Log-Expectation (MUSCLE) [6], Multiple Alignment using Fast Fourier Transform (MAFFT) [9], ProbCons [4] (permite una fase final de refinamiento iterativo) y MUMMALS [14].

En este artículo utilizamos Optimización Multiobjetivo y Computación Evolutiva con el fin de optimizar simultáneamente la calidad y la consistencia del alineamiento final. Para ello, hemos elegido una metaheurística memética: Shuffled Frog-Leaping optimization Algorithm (SFLA) [8], la cual se basa en la interacción de diferentes individuos y el intercambio global de información entre ellos. El algoritmo SFLA original ha sido modificado para poder optimizar varias funciones objetivo simultáneamente e hibridizado con una búsqueda local, lo denominamos Hybrid Multiobjective Memetic Metaheuristic for the Multiple Sequence Alignment (H4MSA).

El resto del artículo se organiza como sigue: la formulación del problema se encuentra en la Sección 2. Una descripción de la metaheurística memética propuesta (H4MSA) se presenta en la Sección 3. En la Sección 4 se presenta una comparativa entre nuestra propuesta y otros alineadores publicados en la literatura. Por último, las conclusiones y líneas de trabajo futuro aparecen en la Sección 5.

2. Alineamiento Múltiple de Secuencias

Dado un conjunto de secuencias S: $\{s_1, s_2, \ldots s_k\}$ con longitud $|s_1|$, $|s_2|$, ..., $|s_k|$ definido sobre el alfabeto Σ. El alineamiento múltiple de secuencias del conjunto S se define como S': $\{s'_1, s'_2, \ldots, s'_k\}$, donde la longitud de las k secuencias es exactamente la misma. S' está definido sobre el mismo alfabeto que S (Σ) con un símbolo adicional ($-$, gap), por lo tanto, S' se define sobre

el alfabeto $\Sigma \cup \{-\}$. De esta forma, un alineamiento múltiple de secuencias se obtiene añadiendo gaps a las secuencias del conjunto S hasta que la longitud de todas sea el mismo. Se puede representar de forma matricial, donde las filas son las secuencias y las columnas representan los símbolos alineados. Cada columna de un alineamiento debe contener al menos un símbolo del alfabeto Σ, o lo que es lo mismo, no pueden existir columnas completas con gaps. A continuación se muestra un ejemplo:

s_1: `THELASTCAT` ($|s_1|$=10)
s_2: `THEFATCAT` ($|s_2|$=9)
s_3: `AVERYFASTCAT` ($|s_3|$=13)

$\rightarrow$

s'_1: `THE----LASTCAT` ($|s'_1|$=14)
s'_2: `THE----FA-TCAT` ($|s'_2|$=14)
s'_3: `A--VERYFASTCAT` ($|s'_3|$=14)

Para encontrar un alineamiento de calidad, se propone el uso de optimización multiobjetivo. Por lo tanto, se busca la solución (alineamiento) que simultáneamente maximice dos funciones objetivo: WSP (Weighted Sum-of-pairs function with affine gap Penalties, f_1) y TC (número Total de columnas Conservadas, f_2) [6].

La primera función objetivo (WSP, f_1) necesita maximizar la siguiente ecuación:

$$WSP(S') = \sum_{l=1}^{AL} \sum_{i=1}^{k-1} \sum_{j=i}^{k} W_{i,j} \times \delta(s'_{i,l}, s'_{j,l}) - \sum_{i=1}^{k} AGP(s'_i) \quad (1)$$

En la ecuación 1, AL indica la longitud del alineamiento, δ es la matriz de substitución utilizada, ya sea PAM (Pointed Accepted Mutation) o BLOSUM (BLOck SUbstitution Matrix). Por otro lado, $W_{i,j}$ hace referencia al peso de la secuencia entre s_i y s_j. Para calcular el peso entre dos secuencias, se utiliza la siguiente ecuación:

$$W_{i,j} = 1 - \frac{LD(s_i, s_j)}{max(|s_i|, |s_j|)} \quad (2)$$

La distancia de Levenshtein (LD) entre dos secuencias no alineadas indica el número mínimo de *inserciones*, *borrados* o *substituciones* necesarios para transformar una secuencia en la otra. En la ecuación 1, $AGP(s'_i)$ corresponde con la penalización por gap de la secuencia s'_i:

$$AGP(s'_i) = (g_o \times \#gaps) + (g_e \times \#spaces) \quad (3)$$

donde g_o es la penalización de empezar un gap y g_e es la penalización de extender un gap. En este trabajo δ=BLOSUM62, g_o=6 y g_e=0,85.

La segunda función objetivo (TC, f_2) hace referencia al número total de columnas que contienen exactamente el mismo compuesto. Esta función objetivo necesita ser máxima para asegurar la conservación del alineamiento.

En este trabajo, no se permiten alineamientos con una longitud superior al 50 % de la secuencia no alineada más larga. La elección de este factor está basada en la observación, ya que la mayoría de alineamientos no suelen contener más del 50 % de gaps.

3. H4MSA

La metaheurística SFLA (Shuffled Frog-Leaping optimization Algorithm) fue propuesta por Eusuff y Lansey [8]. El procedimiento de búsqueda comienza con una generación aleatoria de soluciones (ranas) en un estanque. A continuación, la población se divide en comunidades aisladas, las cuales evolucionarán de forma independiente, permitiendo así explorar regiones distintas del espacio de búsqueda. En cada comunidad, las ranas evolucionan compartiendo sus ideas con sus ranas vecinas, es decir, aquellas ranas con mejores ideas compartirán sus ideas con las demás. Después de un número de iteraciones, las comunidades se mezclan y se crean nuevas comunidades a través de un proceso de barajado.

La representación de una solución en H4MSA es diferente de la representación binaria tradicional, donde un 1 indicaba la presencia de un gap y un 0 indicaba un compuesto. En H4MSA, una solución únicamente almacena el *número de grupos de gaps*, seguido por la información de cada grupo: *posición inicial del primer gap* y *número de gaps sucesivos* (valor representado en negativo). Por ejemplo, la representación para el alineamiento mostrado en la Sección 2 sería: s'_1: (3, -3), s'_2: (3, -3), (9) y s'_3: (1,-1).

Dado un conjunto de k secuencias no-alineadas (S), un número de comunidades (m) que contendrán n ranas, un número fijo N de iteraciones y un criterio de parada, el procedimiento de H4MSA sería:

1. Generar aleatoriamente y evaluar $m \times n$ alineamientos.
2. Realizar una *Ordenación No-dominada* [3] de las $m \times n$ ranas.
3. Dividir las $m \times n$ ranas entre las m comunidades. De forma que la primera mejor rana vaya a la primera comunidad (Y_1), la segunda mejor a la segunda (Y_2), la m mejor a la comunidad m (Y_m), la $m+1$ mejor a Y_1, etc.
4. En cada comunidad (Y_i), seleccionar la mejor rana global (X_{gb}) y la peor (X_{lw}) y mejor (X_{lb}) rana local.
   4.1. X_{lw} reemplaza una porción de su alineamiento con información obtenida de X_{lb}, generándose una nueva rana (X_{new}). Por ejemplo:

```
THE----LAST CAT      TH--ELAS--T CAT    TH--ELAS--T  CAT
THE----FA-T CAT  +   --THEFAT-- CAT- =  --THEFAT-- -  CAT
A--VERYFAST CAT      -AVERYFAST- CAT    -AVERYFAST-  CAT
```

   4.2. Aplicar el siguiente proceso de mutación a X_{new}:
      1) *Mover un bloque*: seleccionar aleatoriamente un bloque de gaps/componentes y moverlo una posición a la izquierda/derecha.

```
TH -- ELAS--TCAT      THE -- LAS--TCAT
 -- THEFAT---CAT  →   T -- HEFAT---CAT
 - AVERYFAST-CAT      A - VERYFAST-CAT
```

      2) *Unir dos grupos*: seleccionar aleatoriamente una secuencia, elegir un grupo de gaps/componentes y unir con el grupo más cercano.

```
THE -- LAS -- TCAT      THE--LAS -- TCAT
T -- HEFAT --- CAT  →   T--HEFAT --- CAT
A - VERYFAST - CAT      AVERYFAST -- CAT
```

3) *Dividir un grupo*: seleccionar aleatoriamente una secuencia, elegir un grupo de gaps/componentes y dividir el grupo en dos grupos de aproximadamente el mismo tamaño.

```
T HE --LAS--TCAT      T HE- -LAS--TCAT
T -- HEFAT---CAT  →   T -H- EFAT---CAT
A VE RYFAST--CAT      A VER YFAST--CAT
```

4) *Compactar Alineamiento*: borrar aquellas columnas con todo gaps.

```
THE--LAS- - TCAT      THE--LAS-TCAT
T-H-EFAT- - -CAT  →   T-H-EFAT--CAT
AVERYFAST - -CAT      AVERYFAST-CAT
```

4.3. Evaluar X_{new}. Si X_{new} *domina* a X_{lw}, entonces ir a 4.8.
4.4. X_{lw} reemplaza una porción de su alineamiento con información obtenida de X_{gb}, generándose una nueva rana (X_{new}).
4.5. Aplicar el proceso de mutación a X_{new}.
4.6. Evaluar X_{new}. Si X_{new} *domina* a X_{lw}, entonces ir a 4.8.
4.7. Aplicar *Búsqueda Local* a X_{lw} y evaluar el nuevo alineamiento producido. En el proceso de búsqueda local, se utiliza la heurística Kalign2 [10] con el fin de re-alinear una pequeña porción (5-25 % del alineamiento).

```
T HE-- LAS--TCAT                T HE-- LAS-TCAT
T -H-E FAT---CAT  → Kalign2 →   T HE-- FAT--CAT
A VERY FAST--CAT                A VERY FAST-CAT
```

4.8. Reemplazar X_{lw} por X_{new} y actualizar el *conjunto de soluciones no-dominadas* con X_{new}.
4.9. Si el número máximo de iteraciones (N) no ha sido alcanzado, ir a 4.1, de lo contrario, continuar con la siguiente comunidad.
5. Mezclar las ranas de las m comunidades, obtenemos $m \times n$ ranas.
6. Si el criterio de parada se cumple, devolver el *conjunto de soluciones no-dominadas*. De lo contrario, ir a 2.

H4MSA devuelve un *conjunto de soluciones no-dominadas*, es decir, un conjunto de alineamientos que balancean la calidad (f_1) y la conservación (f_2).

4. Resultados

En esta sección estudiaremos el rendimiento de la metaheurística propuesta (H4MSA) utilizando una batería de pruebas ampliamente utilizada en la literatura: BAliBASE [19]. En este trabajo, hemos utilizado la última versión de BAliBASE (versión 3.0), la cuál consta de 218 conjuntos de secuencias no alineadas. Estos 218 conjuntos están divididos en seis subfamilias, según sus características biológicas: RV11 (38 conjuntos), RV12 (44 conjuntos), RV20 (41 conjuntos), RV30 (30 conjuntos), RV40 (49 conjuntos) y RV50 (16 conjuntos).

En el estudio comparativo, incluimos los alineadores más relevantes publicados en la literatura: Clustal Ω [16] (v1.2.1), Clustal W [18] (v2.1), DIALIGN-TX

Tabla 1. Valor medio de Q-score y TC-score en cada subfamilia de BAliBASE.

	RV11		RV12		RV20		RV30		RV40		RV50	
	Q(%)	TC(%)	Q(%)	TC(%)	Q(%)	TC(%)	Q(%)	TC(%)	Q(%)	TC(%)	Q(%)	TC(%)
H4MSA	**76,18**	**60,07**	**95,24**	**89,05**	**94,10**	**55,71**	**88,03**	**63,49**	**93,54**	**66,82**	89,38	**63,64**
MSAProbs	68,18	44,40	94,63	*87,03*	92,83	46,94	86,46	61,15	92,32	61,04	**90,76**	*61,21*
MUSCLE	68,26	44,09	94,47	86,19	*92,84*	*47,94*	*87,55*	*62,27*	*92,54*	60,39	89,44	59,31
MAFFT-EINSi	66,00	44,02	93,61	83,90	92,64	45,15	86,12	59,20	91,43	57,51	89,91	59,85
T-Coffee	65,72	41,41	94,47	85,93	91,57	40,60	83,69	47,76	89,64	55,38	89,49	59,11
ProbCons	66,97	41,96	94,12	86,05	91,68	41,16	84,53	54,73	90,03	53,61	89,41	57,89
MAFFT-LINSi	67,12	45,00	93,63	84,23	92,62	45,74	85,55	57,33	91,91	60,09	*89,99*	56,61
ProbAlign	*69,50*	*45,69*	*94,64*	86,69	92,59	44,44	85,30	56,95	92,21	*61,23*	88,86	55,52
FSA-maxsn	61,88	36,58	93,65	84,79	90,10	34,34	81,37	48,26	91,61	58,46	87,19	56,57
MAFFT-GINSi	60,71	34,66	92,70	82,52	90,50	39,13	85,31	53,22	88,64	51,56	88,39	54,99
MUMMALS	66,94	41,97	94,30	84,47	90,62	43,13	84,79	49,81	87,14	48,84	87,91	53,29
Clustal Ω	59,01	36,22	90,60	79,38	91,16	45,29	86,24	57,91	90,10	58,26	86,20	53,74
Kalign2	60,53	36,87	91,21	79,34	90,08	36,25	81,26	47,99	88,33	50,78	82,01	43,96
DIALIGN-TX	50,47	26,81	88,21	75,69	87,81	30,78	76,14	38,90	83,40	45,17	82,15	47,05
H4MSA**	57,20	35,32	89,13	73,02	86,70	33,47	73,93	39,63	83,17	45,35	77,70	46,60
FSA	50,27	27,23	92,38	82,22	86,50	18,99	68,96	26,29	86,09	47,80	78,95	42,06
Clustal W	50,06	22,99	86,49	71,70	85,20	22,16	72,50	27,59	78,94	39,82	74,24	31,16
PRANK	46,18	21,62	83,77	62,08	80,14	12,42	57,84	6,38	74,77	34,22	67,35	20,99

[17] (v1.0.2), FSA y FSA con la opción -maxn [2] (v1.15.9), Kalign2 [10] (v2.03), MAFFT [9] (v7.215, tres versiones: L-INS-i, E-INS-i y G-INS-i), MSAProbs [11] (v0.9.7), MUMMALS [14] (versión 08/02/2008), MUSCLE [6] (v4.0), ProbCons [4] (v1.12), PRANK [12] (con bandera +F y utilizando árbol Clustal W), ProbAlign [15] (v1.4) y T-Coffee [13] (v11.0).

Las métricas utilizadas para evaluar los distintos alineadores son: Q-score y TC-score [6]. Por un lado, Q-score (calidad del alineamiento) indica el número de residuos (componentes) correctamente alineados con respecto a un alineamiento de referencia (*true alignment*). Por otro lado, TC-score es el número de columnas correctamente alineadas dividido entre el número total de columnas en el alineamiento de referencia. El programa para calcular las métricas y los alineamientos de referencia se pueden descargar en [7].

La configuración de la metaheurística H4MSA es: m=5, n=20, N=10. Además el criterio de parada se estableció en 50000 evaluaciones de las funciones objetivo. En cada conjunto de datos de BAliBASE, se realizaron un total de 30 ejecuciones independientes de H4MSA con el fin de extraer conclusiones estadísticamente relevantes. H4MSA fue compilado con g++ (GCC) 4.4.5. Finalmente, H4MSA y todos los alineadores citados anteriormente fueron ejecutados en una máquina Intel (2.3GHz) con 1GB RAM.

En la Tabla 1 presentamos el valor medio de Q-score y TC-score obtenido por cada alineador en cada una de las seis familias de BAliBASE. Como podemos observar, H4MSA obtiene los mejores resultados en casi la totalidad de las subfamilias. RV11 es una subfamilia con un porcentaje de identidad muy bajo (0-20 %), en este caso vemos unas diferencias notables entre H4MSA y el segundo mejor alineador (ProbAlign): 6,68 % para Q-score y 14,38 % para TC-score. En la Figura 1 se muestra para cada subfamilia de BAliBASE, una comparativa visual de los resultados.

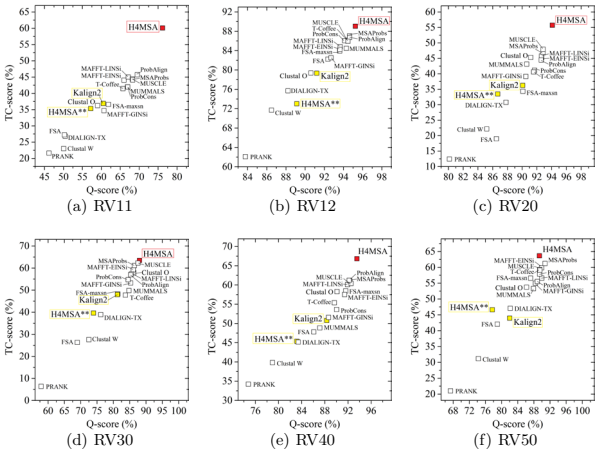

Figura 1. Comparativa, en términos de Q-score y TC-score, entre los distintos alineadores en las distintas subfamilias de BAliBASE.

Con el fin de demostrar que las diferencias entre los distintos alineadores son estadísticamente significativas, se ha realizado un test no paramétrico (test de los rangos con signo de Wilcoxon) entre cada par de alineadores utilizando un nivel de confianza del 1 % (p-value< 0,01). En la Tabla 2 se presentan los resultados obtenidos tras realizar el estudio estadístico. Como podemos ver, en RV11, las diferencias de Q-score y TC-score entre H4MSA y cualquier otro alineador son siempre estadísticamente significativas. En RV12, RV20, RV30 y RV40, H4MSA obtiene valores más altos de Q-score y TC-score que los otros alineadores, sin embargo, en algún caso las diferencias no resultaron estadísticamente significativas. En la Tabla 1 podíamos apreciar que H4MSA obtenía ligeramente peor media de Q-score que MAFFT-EINSi, MAFFT-LINSi, MSAProbs, MUSCLE, ProbCons y T-Coffee; sin embargo, como se puede ver en la Tabla 2 las diferencias entre dichos alineadores y H4MSA no son estadísticamente significativas.

En la Figura 2 se muestra una comparativa entre los distintos alineadores en los 218 conjuntos de BAliBASE. Atendiendo a la calidad y conservación obtenidas por los diferentes alineadores (Figura 2(a) y 2(b)), podemos observar que los dos mejores alineadores son: H4MSA y MSAProbs. Si comparamos nuestra metaheurística (H4MSA) con MSAProbs, podemos ver una diferencia cercana a 1,9 % en Q-score y superior al 6 % en TC-score. En la Figura 2(c) se presenta

Tabla 2. Estudio estadístico entre H4MSA y otros alineadores (BAliBASE). En la tabla, el símbolo ○ indica que existen diferencias estadísticamente significativas y el símbolo • indica que existen diferencias estadísticamente no-significativas.

	RV11		RV12		RV20		RV30		RV40		RV50	
	Q	TC	Q	TC	Q	TC	Q	TC	Q	TC	Q	TC
MSAProbs	○	○	•	•	•	○	•	•	•	•	•	•
MUSCLE	○	○	•	○	•	○	•	•	•	•	•	•
MAFFT-EINSi	○	○	○	○	•	○	•	•	•	○	•	•
T-Coffee	○	○	•	○	○	○	○	○	○	○	•	•
ProbCons	○	○	•	○	○	○	•	○	○	○	•	•
MAFFT-LINSi	○	○	○	○	•	○	•	•	•	○	•	•
ProbAlign	○	○	•	•	•	○	•	○	•	•	•	○
FSA-maxsn	○	○	•	○	○	○	○	○	○	○	•	○
MAFFT-GINSi	○	○	○	○	○	○	•	•	○	○	•	•
MUMMALS	○	○	•	○	○	•	○	○	○	○	•	•
Clustal Ω	○	○	○	○	○	○	○	•	○	○	○	○
Kalign2	○	○	○	○	○	○	○	○	○	○	○	○
DIALIGN-TX	○	○	○	○	○	○	○	○	○	○	○	○
H4MSA**	○	○	○	○	○	○	○	○	○	○	○	○
FSA	○	○	○	○	○	○	○	○	○	○	○	○
Clustal W	○	○	○	○	○	○	○	○	○	○	○	○
PRANK	○	○	○	○	○	○	○	○	○	○	○	○

el tiempo total (en segundos) necesario por cada alineador para resolver los 218 conjuntos contenidos en BAliBASE. A la vista de los resultados, podemos decir que el tiempo requerido por H4MSA (11467 segundos) está claramente por debajo del tiempo que necesita el segundo mejor alineador: MSAProbs (21427 segundos).

Por último, dado que H4MSA utiliza el alineador Kalign2 como búsqueda local, es necesario una comparativa directa entre H4MSA, H4MSA sin búsqueda local (H4MSA**) y Kalign2. En la Figura 2 podemos ver que H4MSA obtiene mejores resultados que Kalign2 y H4MSA** en todos los casos. Por un lado, observamos que H4MSA es capaz de mejorar un 7 % el valor de Q-score y un 17 % el valor de TC-score obtenido por Kalign2. Por otro lado, vemos una mejora cercana al 11 % (Q-score) y 21 % (TC-score) entre H4MSA y H4MSA**.

Como conclusión de este estudio comparativo, podemos decir que H4MSA es un alineador muy prometedor, destacando su particular buen desempeño en los cojuntos de datos con un bajo porcentaje de similaridad.

5. Conclusiones y Trabajo Futuro

En este trabajo se ha propuesto una metaheurística multiobjetivo memética: Hybrid Multiobjective Memetic Metaheuristic for Multiple Sequence Alignment (H4MSA). Esta propuesta está basada en el algoritmo SFLA (Shuffled Frog-Leaping Algorithm), aprovechando de esta forma los beneficios de la inteligencia de ejambre para alinear múltiples secuencias. Además, H4MSA está diseñada para optimizar de manera simultánea la calidad y la conservación del alineamiento final. Además, H4MSA utiliza el alineador Kalign2 como búsqueda local.

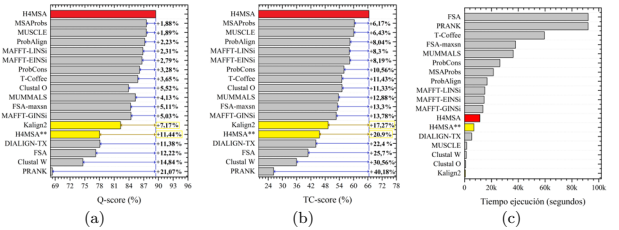

Figura 2. Comparativa entre los diferentes alineadores. En (a) y (b) se presentan los valores medios de Q-score y TC-score obtenidos por cada método en los 218 conjuntos de datos de BAliBASE. En (c), se muestra, para cada alineador, el tiempo de ejecución total (en segundos) invertido en alinear los 218 conjuntos de datos de BAliBASE.

Se ha llevado a cabo un estudio comparativo entre H4MSA y otros dieciseis alineadores propuestos en la literatura: Clustal Ω, Clustal W, DIALIGN-TX, FSA, Kalign2, MAFFT (L-INS-i, E-INS-i, y G-INS-i), MSAProbs, MUMMALS, MUSCLE, ProbCons, PRANK, ProbAlign y T-Coffee. Tras la comparativa, podemos concluir diciendo que nuestra metaheurística obtiene resultados muy prometedores, destacando su particular eficiencia a la hora de resolver conjuntos de datos con un porcentaje de similaridad muy bajo (lo cuales son los más frecuentes en escenarios reales).

Como línea de trabajo futuro, se estudiará el uso de otros algoritmos de inteligencia colectiva con el fin de realizar un estudio comparativo. Por otro lado no se descarta incorporar conocimiento específico del problema aborado con el fin de incrementar la efectividad de la metaheurística propuesta.

Agradecimientos Álvaro Rubio-Largo agradece a la Fundação para a Ciência e a Tecnologia (Portugal) su contrato postdoctoral (SFRH/BPD/100872/2014). Sergio Santander-Jiménez agradece a la Universidad de Extremadura el apoyo económico ofrecido dentro del Plan de Iniciación a la Investigación, Desarrollo Tecnológico e Innovación 2015 (ACCION-III-04). Gracias a la Junta de Extremadura por la ayuda GR15011 concedida al grupo de investigación TIC015.

Referencias

1. D. J. Bacon and W. F. Anderson. Multiple sequence alignment. *J. Mol. Biol.*, 191:153–161, 1986.
2. R. K. Bradley, A. Roberts, M. Smoot, S. Juvekar, J. Do, C. Dewey, I. Holmes, and L. Pachter. Fast statistical alignment. *PLoS Computational Biology*, 5(5):e1000392, 2009.

3. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2000.
4. C. B. Do, M. S. Mahabhashyam, M. Brudno, and S. Batzoglou. ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Research*, 15(2):330–340, 2005.
5. R. Doolittle. Similar amino acid sequences: chance or common ancestry? *Science*, 214:149–159, 1981.
6. R. C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32:1792–1797, 2004.
7. R. C. Edgar. BENCH: A collection of protein sequence alignment benchmarks including BALIBASE v3, PREFAB v4, OXBENCH, and SABRE. *http* : *//www.drive5.com/bench*, 2015.
8. M. Eusuff, K. Lansey, and F. Pasha. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering Optimization*, 38(2):129–154, 2006.
9. K. Katoh, K. Misawa, K. Kuma, and T. Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic Acids Research*, 30(14):3059–3066, 2002.
10. T. Lassmann, O. Frings, and E. L. L. Sonnhammer. Kalign2: high-performance multiple alignment of protein and nucleotide sequences allowing external features. *Nucleic Acids Research*, 37(3):858–865, 2009.
11. Yongchao Liu, Bertil Schmidt, and Douglas L. Maskell. MSAProbs: multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities. *Bioinformatics*, 26(16):1958–1964, 2010.
12. A. Loytynoja and N. Goldman. An algorithm for progressive multiple alignment of sequences with insertions. *Proceedings of the National Academy of Sciences of the United States of America*, 102(30):10557–10562, 2005.
13. C. Notredame, D. G. Higgins, and J. Heringa. T-Coffee: a novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205 – 217, 2000.
14. J. Pei and N. V. Grishin. MUMMALS: multiple sequence alignment improved by using hidden markov models with local structural information. *Nucleic Acids Research*, 34(16):4364–4374, 2006.
15. U. Roshan and D. R. Livesay. Probalign: multiple sequence alignment using partition function posterior probabilities. *Bioinformatics*, 22(22):2715–2721, 2006.
16. Fabian Sievers, Andreas Wilm, David Dineen, Toby Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Soding, Julie Thompson, and Desmond Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega, 2011.
17. A. R. Subramanian, M. Kaufmann, and B. Morgenstern. DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment. *Algorithms for Molecular Biology*, 3:6, 2008.
18. J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.
19. J. D. Thompson, P. Koehl, and O. Poch. BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins*, 61:127–136, 2005.
20. L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.