Heurísticas para resolver el problema de la Supersecuencia Común más Corta

Marcela Rivera-Martínez[1] Luis-René Marcial-Castillo[2]
Lourdes Sandoval-Solís[3]

Benemérita Universidad Autónoma de Puebla, Puebla, Pue., México
[1]mrivmar@hotmail.com, [2]lmc@cs.buap.mx, [3]sandoval@cs.buap.mx
Paper received on 29/06/12, Accepted on 08/09/12.

Resumen. El problema de la Supersecuencia Común más Corta, es un problema NP y tiene aplicaciones en compresión de datos, ingeniería mecánica, biología molecular y planeación entre otros. Para resolver el problema de la Supersecuencia Común más Corta, en este trabajo, se describen algunas heurísticas, las cuales están basadas en la heurística clásica Majority Merge, sin embargo, difieren en el criterio que se sigue para elegir el símbolo que formará parte de la Supersecuencia. Se realizaron varios experimentos para comparar los resultados de las heurísticas con Majority Merge.

Palabras Clave: Supersecuencia Común más Corta, Heurísticas, Majority Merge**.**

1 Introducción.

El problema de la Supersecuencia Común más Corta (Shortest Common Supersequence, SCS) está dado de la siguiente forma: Dado un lenguaje L sobre un alfabeto Σ, se desea encontrar una de las cadenas w de menor longitud que sea una Supersecuencia de L. Se dice que S es una Supersecuencia de una cadena T, si S se puede obtener desde T mediante la inserción de cero o más símbolos de la misma cadena. Ejemplo: Dado Σ = {a, b} y L = {ab, ba}. Las siguientes cadenas son supersecuencias de L: "abba", "aabba", "aba", "bab". No son las únicas, existen muchas más, se dice que todas ellas son supersecuencias porque cada una de ellas contiene a todas las cadenas de L (aún cuando no las contengan de manera contigüa). Ahora, las supersecuencias comunes más cortas son dos y son: S = aba y S = bab. Ambas son supersecuencias de L y son las más cortas porque ambas son de longitud 3, se denota: |S| = 3, es decir, la supersecuencia puede no ser única.

El problema de la supersecuencia común más corta es un problema NP-duro bajo ciertas restricciones concernientes al tamaño del alfabeto y la longitud de las cadenas [6, 9]. El problema de la SCS tiene aplicaciones en compresión de datos, ingeniería mecánica, planeación y biología molecular entre otros [8].

Advances in Intelligent and Information Technologies.
Research in Computing Science 60, 2012, pp. 55 -64.

Existen diferentes formas de resolver el problema de la SCS, entre ellas tenemos:

Algoritmos secuencia por secuencia. Este tipo de algoritmos construyen la supersecuencia tomando pares de cadenas, la forma de elegir esos pares de cadenas varía dependiendo del algoritmo, entre ellos tenemos el algoritmo de programación dinámica [4], el algoritmo de torneo, etc.

Algoritmos de reducción y/o expansión. Inician con alguna supersecuencia común del lenguaje L y van eliminando (y/o expandiendo) dicha supersecuencia, entre ellos están el algoritmo de reducción-expansión [2,7], el algoritmo de Hubbell-Morris-Winkler [5], etc.

Algoritmos con metaheurísticas. La mayoría de ellos hibridan con heurísticas clásicas como Majority Merge (MM) [3], sin embargo, ésta heurística clásica no siempre funciona de manera adecuada, por lo cual se buscan nuevas heurísticas que ayuden a superar los problemas que presenta la heurística MM.

En éste trabajo se proponen heurísticas basadas en Majority Merge, pero que difieren de ésta en el criterio para elegir el nuevo elemento a adicionar a la Supersecuencia, lo cual permitirá encontrar mejores soluciones de la SCS y poder aplicarlas a problemas reales como la hibridación de oligonucleótidos [1, 2].

2 Heurísticas.

Muchos problemas de optimización no pueden ser resueltos utilizando métodos exactos, ya sea, porque tienen un alto grado combinatorio o por la dificultad de generar un modelo basado en programación matemática que represente exactamente una situación real. Para este tipo de problemas, se han venido generando desde la década de los sesenta métodos conocidos como heurísticas, capaces de encontrar soluciones de buena calidad, aproximadas a la solución óptima y en un periodo de tiempo razonable. De acuerdo con el estándar ANSI/IEEE 100-1984, "la heurística trata de métodos o algoritmos exploratorios, durante la solución de problemas, en los cuales las soluciones se descubren por la evaluación del progreso logrado en la búsqueda de un resultado final."

A continuación se describen 4 heurísticas que difieren en la forma de elegir el símbolo a adicionar a la Supersecuencia.

2.1 Heurística Majority Merge (MM).

La heurística MM es una de las heurísticas clásicas más utilizadas para resolver el problema de la SCS, está clasificada dentro de los algoritmos símbolo por símbolo. Dado un lenguaje L y un alfabeto Σ, el símbolo σ que se agrega es aquel que estando al frente de cada cadena L sea el de mayor frecuencia, y una vez agregado elimina ese símbolo del frente de cada cadena donde se encuentre, éste proceso se termina cuando el lenguaje L contiene únicamente la cadena vacía.

Algoritmo MM.

Entrada: Lenguaje (L) y Alfabeto (Σ).

$S \leftarrow$ cadena vacía.

REPETIR

Calcular $\Sigma' = \{a \in \Sigma \mid a = \max(\text{frec}(a))\}$, donde frec($a$) es la frecuencia con que aparece el símbolo a al inicio de cada cadena en L.

Elegir $\sigma \in \Sigma'$.

Actualizar L, eliminando σ del frente.

$S \leftarrow S\sigma$.

HASTA que L sea la cadena vacía.

Salida: S y su longitud.

Ejemplo 1: Sean $\Sigma = \{a, b, c\}$ y $L = \{acacb, cbaac, bbacc\}$.

La forma en que la heurística Majority Merge va construyendo la supersecuencia es la siguiente: Los caracteres al frente de cada cadena son: a, c y b. Se calcula la frecuencia de cada uno de ellos y se obtiene el máximo: máx{1, 1, 1} = 1. Por lo cual Σ' = {a, b, c}, de ahí se elige el símbolo a adicionar a la Supersecuencia, por ejemplo a, es decir, $S = a$, entonces, el símbolo a es eliminado del frente de las cadenas de L donde aparece, resultando el nuevo lenguaje $L=\{cacb, cbaac, bbac\}$. Este proceso se repite hasta que L sea la cadena vacía.

La heurística MM, tiene problemas en encontrar la Supersecuencia de menor longitud en cadenas de longitud aleatoria, debido a que el criterio que utiliza para adicionar un símbolo a la Supersecuencia hace que aquellas cadenas con mayor longitud sean las que vayan quedando en el lenguaje L aumentando de ésta forma la longitud de S.

2.2 Heurística H1.

Debido a la problemática que presenta la heurística MM, se asignan pesos a los caracteres de cada cadena, lo cual esta dado por la longitud de la cadena menos la posición del símbolo dentro de la cadena, es decir, para un caracter a_i de una cadena $A = a_1a_2...a_n$, el peso $w(a_i)$ es n-i.

Algoritmo H1.

Entrada: Lenguaje (L) y Alfabeto (Σ).

$S \leftarrow$ cadena vacía.

REPETIR

  PARA cada $a \in \Sigma$ HACER

    Calcular $w(a)$.

  FIN_PARA

Calcular $\Sigma' = \{ a \in \Sigma \mid wsum(a) = \max\{wsum(a) \text{ para cada } a \in \Sigma\}\}$, donde $wsum(a)$ es la suma de los pesos de las ocurrencias de a en el frente actual de cada cadena de L.

Elegir $\sigma \in \Sigma'$.

Actualizar L, eliminando σ del frente de cada cadena de L donde aparezca σ.

$S \leftarrow S\sigma$.

HASTA que L sea la cadena vacía.

Salida: S y su longitud.

Considerando el Ejemplo 1, veamos la forma de construir la supersecuencia aplicando la heurística H1: primero se calcula el peso para cada símbolo del alfabe-

to, obteniendo: *w(a) = 4, w(b) = 4* y *w(c) = 4*, ahora se calcula la suma de los pesos de cada símbolo del alfabeto y se obtiene: *wsum(a) = 4, wsum(b) = 4 y wsum(c) = 4*. Se calcula el máximo de la suma de los pesos y se forma Sigma Prima, obteniendo: $\Sigma' = \{a, b, c\}$ y de éste último conjunto, se elige el símbolo a adicionar a la Supersecuencia, digamos el símbolo *c*, ahora, el símbolo *c* se elimina de las cadenas de *L* donde aparezca al frente, obteniendo el nuevo lenguaje *L={acacb, baac, bbacc}.* Este proceso se repite hasta que sólo quede la cadena vacía en el lenguaje *L*.

2.3 Heurística H2.

Otra heurística diferente, llamada H2, considera no sólo el caracter que está al frente de cada cadena, sino considera además del primero, el segundo caracter y elige el símbolo *a* cuyo número de ocurrencias al frente de cada cadena más el número de ocurrencias del segundo símbolo más frecuente, sea máximo. Para ello se utiliza la siguiente notación:

L-a está formado por aquellas cadenas del lenguaje *L* que tienen al frente el caracter *a* y que se ha eliminado de las cadenas donde aparece como caracter inicial; *occ*(*a, L-a*) denota la cantidad de *a's* al frente de cada cadena del Lenguaje (*L-a*).

Algoritmo H2.
Entrada: Lenguaje (*L*) y Alfabeto (Σ).
S← cadena vacía.
REPETIR
  PARA cada $a \in \Sigma$ HACER
    Calcular *L-a.*
    Calcular *occ(a, L-a).*
  FIN_PARA
  PARA cada $a \in \Sigma$ HACER
Calcular r_a= max{*occ(a, L-a)* para cada $a \in \Sigma$}.
  FIN_PARA
Calcular $\Sigma' = \{a_i \in \Sigma \mid i = \text{máx}\{occ(a, L)+r_a\}\}$, donde *occ(a, L)* es la cantidad de *a's* en el frente actual de cada cadena de *L*.
  Elegir $\sigma \in \Sigma'$.
  Actualizar *L* eliminando σ del frente de cada cadena de *L* donde aparezca σ.
  $S \leftarrow S\sigma$.
HASTA que *L* sea la cadena vacía.
Salida: *S* y su longitud.

La forma que la heurística H2 forma la supersecuencia para el ejemplo 1 es la siguiente: Para cada símbolo del alfabeto se calcula un nuevo lenguaje (denotado *L-a*) y se obtiene: *L-a = {cacb, cbaac, bbacc}, L-b={acacb, cbaac, bacc} y L-c={acacb, baac, bbacc}*, ahora se calcula *occ(L-a)* de lo cual se obtiene: *occ(a,L-a) = 0, occ(b, L-a) =1, occ(c, L-a) = 2, occ(a, L-b) =1, occ(b, L-b)= 1, occ(c, L-b) =1, occ(a, L-c) = 1, occ(b, L-c) =2* y *occ(c, L-c) = 0,* se calcula r_a=*máx{0, 1, 2} = 2*, r_b=*máx{1} = 1,* r_c = *máx{0, 1, 2} = 2*. Ahora se calcula para cada caracter del alfabeto *occ(a, L)* y se obtiene: *occ(a, L) = 1, occ(b, L) = 1* y *occ(c, L) =1,* se procede a

calcular *máx{occ(a, L)+ r_a}* para cada símbolo *a* del alfabeto y se obtiene: *máx{(1+2), (1+1), (1+2)} = 3*, con ello se forma $\Sigma' = \{a, c\}$, de éste nuevo conjunto se elige el símbolo a adicionar a la Supersecuencia, digamos *a*, por lo cual se tiene *S = a* y se elimina el símbolo *a* de las cadenas de *L* donde aparezca al frente, obteniendo el nuevo lenguaje *L={cacb, cbaac, bbacc}*. Este proceso continúa hasta que el lenguaje *L* contenga únicamente la cadena vacía.

2.4 Heurística H3.

La heurística llamada H3, calcula para cada símbolo *a* al frente de cada cadena de *L*, la longitud de la supersecuencia mediante el algoritmo H1, al eliminar ese caracter *a* del frente (*L-a*), después de ésto H3 toma como símbolo para agregar a *S* aquel que tenga la mínima longitud, dado que generalmente hay empates H3 lo resuelve calculando para cada cadena en *L* que tenga al frente el caracter *a* (tal que este caracter haya sido elegido como mínimo en el paso anterior) la supersecuencia de (L_a) usando H1 y se elige el símbolo cuya longitud devuelta por H1 sea máxima. La notación que se usa es la siguiente:

L-a está formado por aquellas cadenas del lenguaje *L* que tienen al frente el caracter *a* y éste caracter se ha eliminado de las cadenas donde aparece como caracter inicial.

L_a está formada por aquellas cadenas de *L* que tienen al frente el caracter *a*.

Algoritmo H3.

Entrada: Lenguaje (*L*) y Alfabeto (Σ).
$S \leftarrow$ cadena vacía.
REPETIR
  PARA cada $a \in \Sigma$ HACER
    Calcular *L-a*.
  FIN_PARA
  Calcular $\Sigma' = \{a \in \Sigma$ tal que $a = \min\{|H1(L\text{-}a)|\}$ para cada $a \in \Sigma\}$.
  PARA cada $a \in \Sigma'$ HACER
    Calcular L_a.
  FIN_PARA
  Calcular $\Sigma^* = \{a \in \Sigma'$ tal que $a = \max\{|H1(L_a)|\}$ para cada $a \in \Sigma'\}$.
  Elegir $\sigma \in \Sigma^*$.
  Actualizar *L* eliminando σ del frente de cada cadena de *L* donde aparezca σ.
  $S \leftarrow S\sigma$.
HASTA que *L* sea la cadena vacía.
Salida: *S* y su longitud.

La heurística H3 forma la supersecuencia para el ejemplo 1 de la siguiente manera: Para cada símbolo del alfabeto se calcula *L-a* y se obtiene la longitud de la Supersecuencia Común más Corta de acuerdo a H1, así se obtiene: *L-a = {cacb, cbaac, bbacc}* y |*S*| = 8, *L-b = {acacb, cbaac, bacc}* y |*S*| = 8, y *L-c = {acacb, baac, bbacc}* y |*S*| = 9, después se obtiene el mínimo de las longitudes de las supersecuencias obtenidas por H1: *mín{8 ,8, 9}= 8* y con ello se forma $\Sigma' = \{a, b\}$. Ahora

para cada elemento de Σ' se calcula L_a y posteriormente se aplica H1 al lenguaje L_a obtenido, para el ejemplo esto quedaría como: L_a={*acacb*} con $|S| = 5$ y L_b={*bbacc*} con $|S| = 5$, ahora calculamos el valor máximo de las longitudes de las supersecuencias obtenidas $max\{5,5\} = 5$ y con ello formamos $\Sigma^* = \{a, b\}$, de éste último conjunto se elige el símbolo que formará parte de la Supersecuencia, digamos *b* ($S = b$) y se procede a eliminar *b* del frente de cada cadena de *L* donde aparezca, obteniendo el nuevo lenguaje *L={acacb, cbaac, bacc}*, el proceso se repite hasta que la cadena vacía sea la única cadena del lenguaje *L*.

2.5 Otras variantes.

Se proponen además algunas variantes para los algoritmos anteriores (MM, H1, H2 y H3). Una vez que se ha determinado la forma en la cual se elige el símbolo σ para que forme parte de la Supersecuencia *S*, puede ocurrir con mucha frecuencia (en el caso de cadenas de diferente longitud) que el símbolo σ no sea único, es decir, que existen empates, en cuyo caso las formas en que se determinará la forma de romper los empates serán las siguientes:

1. El símbolo que se concatenará para formar la supersecuencia *S*, se elige en orden lexicográfico.
2. La elección del símbolo se realiza de forma aleatoria(A).
3. El símbolo a elegir es aquel cuya longitud de la cadena donde se encuentre sea la máxima, en caso de que las cadenas coincidan en longitud, se toma en forma aleatoria(AS).

3 Experimentos.

Para el problema de la SCS no existen archivos benchmark, por lo cual los problemas se generaron de manera aleatoria, la longitud de la supersecuencia común más corta para cada problema se desconoce, sin embargo, se desea con estos experimentos observar y comparar los resultados de las heurísticas expuestas anteriormente para el caso promedio, y observar el comportamiento de la dispersión de los resultados en el rango del máximo y el mínimo valor encontrado para la longitud de la Supersecuencia, es decir, interesa observar cuál de todos los algoritmos encuentra la mejor solución con la menor dispersión y cuál algoritmo es el mejor en encontrar la mejor solución en el caso promedio.

Para cada experimento se reportan los siguientes resultados: el número 1 denota el promedio de la longitud de la SCS, el número 2 se refiere al valor máximo de la longitud de la SCS encontrado en todas las corridas y finalmente el número 3, reporta el valor mínimo de la longitud de la SCS hallado a lo largo de las corridas. Los algoritmos que se reportan son: Majority Merge (MM), Majority Merge aleatorio (MMA), Majority Merge de cadena máxima (MMAS), H1, H1 aleatorio (H1A), H1 de cadena máxima (H1AS), H2, H2 aleatorio (H2A), H2 de cadena máxima (H2AS), H3 usando H1 (H3-H1), H3 usando H1 aleatorio (H3-H1A), H3 usando H1 de cadena máxima (H3-H1AS), H3 aleatorio usando H1 (H3A-H1), H3 aleatorio usando H1 aleatorio (H3A-H1A), H3 aleatorio usando H1 de cadena máxima (H3-H1AS), H3

de cadena máxima usando H1 (H3AS-H1), H3 de cadena máxima usando H1 aleatorio (H3AS-H1A), y H3 de cadena máxima usando H1 de cadena máxima (H3AS-H1AS).

Para cada prueba generada aleatoriamente, se corrió 30 veces el mismo problema para reportar en la Tabla 1: en la columna 1 el valor promedio, en la columna 2 el máximo valor y en la columna 3 el mínimo valor, de la longitud de la Supersecuencia Común, el menor promedio encontrado por alguno de los algoritmos se remarca con un doble subrayado, y el valor más pequeño de la longitud durante las corridas encontrado durante la prueba se resalta con un estilo de "negritas".

El primer experimento consta de 6 problemas que tienen la característica de que para cada lenguaje la longitud de cada una de las cadenas que forman el lenguaje es constante, cada problema está especificado de la siguiente manera:

L1: |L| = 10, |Σ| = 3, cada cadena de longitud 7.
L2: |L| = 15, |Σ| = 3, cada cadena de longitud 10.
L3: |L| = 15, |Σ| = 4, cada cadena de longitud 10.
L4: |L| = 20, |Σ| = 5, cada cadena de longitud 10.
L5: |L| = 30, |Σ| = 5, cada cadena de longitud 15.
L6: |L| = 30, |Σ| = 5, cada cadena de longitud 20.

Tabla 1. Resultados de 30 corridas.

ALG	L1			L2			L3			L4			L5			L6		
	1	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**
MM	17.0	17	17	23.0	23	23	30.0	30	30	37.0	37	37	50.0	50	50	72.0	72	72
MMA	17.0	17	17	23.8	25	23	28.8	31	**27**	35.4	38	33	53.5	57	50	68.7	72	66
MMAS	17.0	17	17	23.0	23	23	28.4	29	28	35.7	36	35	52.0	52	52	66.5	67	66
H1	17.0	17	17	23.0	23	23	29.0	29	29	36.0	36	36	51.0	51	51	68.0	68	68
H1A	17.7	23	17	24.2	29	23	32.0	42	28	39.0	56	34	51.8	55	51	71.7	87	66
H1AS	17.0	17	17	23.0	23	23	28.7	29	28	36.2	37	35	51.0	51	51	66.7	68	65
H2	**16.0**	16	**16**	24.0	24	24	29.0	29	29	34.0	34	34	52.0	52	52	66.0	66	66
H2A	18.6	24	**16**	23.6	28	**22**	30.3	39	28	35.0	47	33	52.7	62	49	69.9	74	66
H2AS	18.2	23	**16**	23.4	26	**22**	30.1	36	28	34.8	40	33	52.3	62	50	68.5	72	66
H3-H1	**16.0**	16	**16**	**22.0**	22	**22**	28.0	28	28	32.0	32	32	**49.0**	49	49	**64.0**	64	64
H3-H1A	19.5	25	17	25.0	29	23	37.0	45	31	45.7	60	33	67.4	84	51	78.9	92	64
H3-H1AS	**16.0**	16	**16**	**22.0**	22	**22**	27.8	28	**27**	32.6	35	**31**	49.4	51	**48**	64.6	69	**63**
H3A-H1	**16.0**	16	**16**	**22.0**	22	**22**	28.0	28	28	31.4	32	**31**	**49.0**	49	49	**64.0**	64	64
H3A-H1A	19.9	28	17	25.1	29	23	37.2	44	32	46.4	64	35	68.0	90	56	80.4	101	66
H3A-H1AS	**16.0**	16	**16**	**22.0**	22	**22**	**27.6**	28	**27**	32.5	35	**31**	49.5	51	**48**	64.6	68	**63**
H3AS-H1	**16.0**	16	**16**	**22.0**	22	**22**	28.0	28	28	**31.0**	31	**31**	**49.0**	49	49	**64.0**	64	64
H3AS-H1A	19.3	22	**16**	25.5	32	**22**	36.4	42	32	45.2	54	37	65.6	82	52	81.1	92	64
H3AS-H1AS	**16.0**	16	**16**	**22.0**	22	**22**	27.7	28	**27**	32.5	37	**31**	49.6	51	49	65.0	67	**63**

Se puede observar en la tabla anterior que el algoritmo *H3AS-H1* encuentra la mayoría de veces (5 de 6), en promedio las mejores Supersecuencias (excepto para **L3**), en segundo lugar los algoritmos que encuentran supersecuencias más cortas son *H3-H1 y H3A-H1* (4 de 6 veces). Se puede observar también que para **L2** y **L4**, se llega a encontrar una Supersecuencia de longitud igual a la encontrada en el promedio por *H3AS-H1*, por algoritmos diferentes a los que encontraron la menor supersecuencia en el promedio de 30 corridas; para **L5** y **L6** se encuentran, incluso, longitudes de la supersecuencia menores que los encontrados por *H3AS-H1* en el promedio

de 30 corridas. Considerando la Supersecuencia de menor longitud como aquella que se reporta como mínima al calcular el promedio de las 30 corridas, el algoritmo que encuentra con mayor frecuencia las de menor longitud es *H3-H1AS*, es decir, *H3* usando *H1* donde la forma de romper empates es de cadena máxima. Finalmente también se observa en la Tabla 1 que en general, los algoritmos que rompen los empates de manera aleatoria (A) tienen un rango amplio entre la longitud mínima y la longitud máxima encontrada para una Supersecuencia, mientras que los algoritmos que rompen el empate de acuerdo a la máxima longitud de la cadena (AS) se dispersan menos, es decir, la diferencia encontrada entre el valor máximo y el valor mínimo de la longitud de la Supersecuencia es poca (entre cero y once). También se realizó 100 veces el mismo experimento ejecutando cada algoritmo para los seis problemas descritos y se observó que los resultados no varían cuando el número de veces que se ejecutan los algoritmos aumenta.

El segundo experimento consta de lenguajes en los cuales la heurística MM tiene un mal comportamiento y la Supersecuencia común más corta es conocida [1]. Son tres lenguajes descritos de la siguiente manera:

L1: 9 cadenas a^{40}, 4 cadenas ba^{39}, 2 cadenas bba^{38} y 1 cadena $bbba^{37}$.
L2: 9 cadenas a^{40}, 4 cadenas $b^{13}a^{27}$, 2 cadenas $b^{26}a^{14}$ y 1 cadena $b^{39}a$.
L3: 8 cadenas $a^{20}b^{20}$, 8 cadenas $b^{20}c^{20}$.

Para **L1**, la solución óptima conocida es una cadena de longitud 43; la longitud de la cadena óptima para **L2** es de 79 y para **L3** la longitud de la Supersecuencia óptima es de 60.

Se observa en los resultados de la Tabla 2, que para **L1**, el algoritmo *Majority Merge* en sus tres versiones obtiene las peores longitudes de la Supersecuencia, para el caso promedio y para el valor mínimo de todas las corridas; los algoritmos *H3* en sus diferentes versiones, salvo aquellos combinados con *H1A*, son los que obtienen los valores óptimos en el caso promedio y en el valor mínimo de todas las corridas.

Para **L2**, nuevamente los algoritmos *H3* en todas sus versiones, excepto los que se combinan con *H1A*, son los que obtienen los valores óptimos conocidos tanto para el caso promedio como para el valor mínimo de todas las corridas.

Se observa que para el lenguaje **L3**, tanto *Majority Merge* como *H2* obtienen el valor óptimo de la Supersecuencia.

Por lo que resumiendo, la familia de *H3*, excepto la que se combina con *H1A*, obtiene el óptimo conocido en dos ocasiones de tres problemas propuestos. Además, se observa que la dispersión de la familia *H3*, excepto la que se combina con *H1A*, es cero para los problemas que encuentra el óptimo conocido.

También se realizó el mismo experimento 100 veces y se observó que los resultados no varían cuando el número de veces que se ejecutan los algoritmos para resolver el mismo problema aumenta.

4 Conclusiones.

De acuerdo a los resultados obtenidos al realizar los experimentos, con problemas generados aleatoriamente, se puede concluir lo siguiente:

En general, la mejor heurística es H3AS-H1, ya que encontró la menor supersecuencia común en la mayoría de los problemas presentados. Cuando existen empates en el caracter a adicionar a la Supersecuencia, la variante que toma en orden lexicográfico dicho caracter, encuentra mejores resultados que las de forma aleatoria o las guiadas por la longitud de la cadena, en caso de que se reporte el promedio de las longitudes.

Si la longitud de la Supersecuencia a reportar es aquella que tenga el mínimo valor a lo largo de todas las corridas efectuadas, en caso de empate, los algoritmos que utilizan el criterio de cadena máxima para tomar la decisión sobre el caracter a adicionar a la Supersecuencia son los mejores.

Se observa menor dispersión en los algoritmos, en caso de empate, lo deciden tomando en consideración la longitud de la cadena.

Para los lenguajes especiales con mal comportamiento, los mejores algoritmos son los que corresponden a la familia H3, excepto los que se combinan con H3-H1A.

No es significativo el aumento en la cantidad de corridas para cada problema.

Tabla 2. Resultados de lenguajes especiales: 30 corridas.

ALG	L1			L2			L3		
	1	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**
MM	157.00	157	157	121.00	121	121	60.00	60	**60**
MMA	157.00	157	157	121.00	121	121	77.53	80	68
MMAS	157.00	157	157	121.00	121	121	79.30	80	79
H1	92.00	92	92	91.00	91	91	79.00	79	79
H1A	92.93	97	92	92.23	97	91	80.06	83	79
H1AS	92.00	92	92	91.00	91	91	79.56	80	79
H2	116.00	116	116	121.00	121	121	60.00	60	**60**
H2A	81.66	86	80	121.00	121	121	79.36	86	72
H2AS	81.36	90	80	121.00	121	121	79.53	88	67
H3-H1	43.00	43	**43**	79.00	79	**79**	78.00	78	78
H3-H1A	50.90	55	46	80.16	85	**79**	80.73	90	76
H3-H1AS	43.00	43	**43**	79.00	79	**79**	77.93	79	76
H3A-H1	43.00	43	**43**	79.00	79	**79**	78.66	79	78
H3A-H1A	50.50	56	45	80.46	83	**79**	79.96	86	73
H3A-H1AS	43.00	43	**43**	79.00	79	**79**	78.30	79	77
H3AS-H1	43.00	43	**43**	79.00	79	**79**	78.20	79	78
H3AS-H1A	50.13	60	44	80.06	85	**79**	79.96	90	73
H3AS-H1AS	43.00	43	**43**	79.00	79	**79**	78.36	79	76

Agradecimientos.

Agradecemos el apoyo financiero de la Vicerrectoría de Investigación y Estudios de Posgrado de la Benemérita Universidad Autónoma de Puebla a través del proyecto 0333 Aplicación de las Meta-heurísticas Inspiradas en la Naturaleza.

Referencias

1. Branke J., Middendorf M., Schneider F.: Improved heuristic and a genetic algorithm for finding short supersequences. OR Spektrum. Vol. 20, 39–45. (1998).
2. Kang N. and Hon Wai L.: Towards a better solution to the shortest common supersequence problem: the deposition and reduction algorithm. BMC Bioinformatics. 7(Suppl 4):S12. (2006).
3. Foulser, D. E., Li M., Yang Q.: Theory and algorithms for plan merging, Artificial Intelligence. vol. 5, 143-181. (1992).
4. Fraser C. B., Irving R. W.: Approximation algorithms for the shortest common supersequence. Nordic Journal of Computing. Vol. 2, 303–325. (1995).
5. Hubbell E. A., Morris M. S., and Winkler J. L.: Computeraided engineering system for design of sequence arrays and lithographic masks. US Patent 5571639. (1996).
6. Middendorf M.: More on the complexity of common superstring and supersequence problems. Theoretical Computer Science. Vol. 125, 205–228. (1994).
7. Ning K., Leong H. W.: Towards a better solution to the shortest common supersequence problem: the deposition and reduction algorithm. Symposium of Computations in Bioinformatics and Bioscience with the International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06). Vol. 1, 84-90 (2006).
8. Rahmann S.: The shortest common supersequence problem in a microarray production setting Approximation algorithms for the shortest common supersequence. Bioinformatics, Vol. 19, 156–161. (2003).
9. Timkovsky V. G.: Complexity of common subsequence and supersequence problems and related problems. Cybernetics and Systems Analysis. Vol. 25, 565–580. (1990).