UNIVERSIDAD DE GRANADA

MH

Metaheurísticas

Curso: 3º **Grupo:** 1

Práctica Final
The Ant Lion Optimizer

Autor: Mario Carmona Segovia
DNI: 45922466E **E-mail:** mcs2000carmona@correo.ugr.es

Profesor: Francisco Herrera

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Curso 2020 - 2021

Índice

Índice de figuras

Índice de tablas

1. Resumen de la metaheurística

Esta información ha sido extraida del paper del algoritmo [1] y de una wiki sobre el algoritmo [2].

1.1. Inspiración

La metaheurística Ant Lion Optimizer está inspirada en la interacción entre las hormigas y las hormigas león. Las hormigas león son una especie de animales similares a las libélulas en su etapa adulta. Su nombre proviene de su forma cuando se encuentra en la etapa de larva. En esta etapa tienen una forma simular a una hormiga.

La metaheurística simula la caza de las hormigas por parte de las hormigas león, que son sus depredadoras. La forma de cazar de las hormigas león en su etapa de larva consiste en excavar un hoyo en el suelo, que suelen ser suelos arenosos; formando en el suelo una especie de embudo. Una vez termina de excavar, se coloca en el fondo del embudo enterrando todo su cuerpo menos las mandíbulas que quedan sobresaliendo un poco de la tierra.

En resumen la metaheurística simula esta caza que está compuesta de los siguientes pasos:

- Movimiento aleatorio de la hormigas
- Construcción de las trampas
- Caída de las hormigas en la trampas
- Captura de las hormigas
- Reconstrucción de la trampa

Todos estos pasos están implementados como parte de la metaheurística.

Para poder realizar la metaheurística es necesario guardar la posición de todas la hormigas y las hormigas león. Esta información se guarda en dos matrices, una para las hormigas y otra para las hormigas león.

La posición de una hormiga o de una hormiga león se representa con un vector de n variables reales.

1.2. Movimiento aleatorio de la hormigas

Al igual que pasa en la naturaleza, las hormigas en la metaheurística se mueven de forma aleatoria. Este movimiento aleatorio introduce una componente de diversidad en la metaheurística.

Este movimiento aleatoria está acotado para que las variables que componen la posición de una hormiga sólo tomen valores posibles para esas variables. Además de esta cota en el máximo y mínimo valor que se puede tomar, existe el rango del movimiento aleatorio, es decir, como de

lejos se puede mover de forma aleatoria la hormiga. Este rango disminuye conforme avanzan las iteraciones para comenzar con mucha diversidad al principio y terminar el algoritmo con poca diversidad.

La nueva posición de la hormiga se calcula de la siguiente forma:

$$X_i^t = \frac{(X_i^t - a_i) \ * \ (d_i - c_i^t)}{(b_i - a_i)} + c_i$$

La variable a es la cota inferior y la variable b es la cota superior del valor que puede tomar la variable. La variable c y d representan el rango del movimiento, c representa la cota inferior y d representa la cota superior. Y X_i^t es la suma acumulada aleatoria de la variables i en la iteración t. El resultado de esta fórmula es R_x^t, siendo x la hormiga león usada para guiar el movimiento y t la iteración actual.

1.3. Construcción de las trampas

Durante la ejecución del algoritmo no todas la hormigas león construyen la trampa al mismo tiempo. En cada iteración del algoritmo elige una hormiga león para cada una de las hormigas usando una ruleta, donde se calcula el peso de cada hormiga león teniendo en cuenta su fitness y el fitness acumulado por todas las hormigas león, y se selecciona de forma aleatoria generando un valor en el rango [0,1].

Cuando se haya seleccionado la hormiga león, esta influye en el movimiento de la hormiga a la que ha sido asociada, provocando que se hunda en la trampa y se acerca cada vez más al fondo de ella, donde se encuentra la hormiga león.

Esta selección con ruleta provoca que aquellas hormigas león con mayor peso tengan una mayor probabilidad de crear la trampa que atrape a la hormiga.

1.4. Caída de las hormigas en las trampas

El movimiento de la hormiga es aleatorio, pero ese movimiento se ve influenciado por la trampa de la hormiga león a la que ha sido asociada, y además influye la trampa creada por la mejor hormiga león. Se hace de esta forma para mantener el elitismo, por lo que en el movimiento de todas la hormigas influirá la mejor hormiga león, por lo tanto esta hormiga león siempre crea su trampa, mientras siga siendo la mejor hormiga león.

Este elitismo en el movimiento se representa con la media de los movimientos aleatorios generados por ambas trampas:

$$Ant_i^t = \frac{R_A^t + R_E^t}{2}$$

Donde A es la hormiga león seleccionada con la ruleta y E la mejor hormiga león.

1.5. Captura de las hormigas

Una vez la hormiga a llegado al fondo de la trampa, cuando esta tiene un fitness menor que el de la hormiga león a la que pertenece la trampa, este menor fitness representa que la hormiga esta enterrada en la tierra al fondo de la trampa; la hormiga león captura a la hormiga.

$$Antlion_j^t = Ant_i^t \qquad iff(Ant_i^t) \ < \ f(Antlion_i^t)$$

1.6. Reconstrucción de las trampas

Una vez la hormiga león ha captura a la hormiga, se simula el movimiento de la hormiga león hacia otra posición donde crear la nueva trampa. Este cambio de posición se realiza en el algoritmo asignando a la hormiga león la posición de la hormiga que ha capturado.

2. Análisis del algoritmo evolutivo

En primer lugar vamos a ver como se comporta el algoritmo Ant Lion en relación a los algoritmos de la competición CEC2017.

2.1. Posición en el ranking de la competición

Para obtener la posición del algoritmo en la competición, se ha comparado con todos los algoritmos presentados en la competición CEC2017, obteniendo el ranking a partir de la media de los resultados obtenidos en cada función.

2.1.1. Dimensión 10

Algoritmo	**Ranking**
EBOwithCMAR	3.450
MM_OED	3.617
jSO	4.017
ELSHADE_SPACMA	4.683
RB-IPOP-CMA-ES	6.083
DE	7.117
PPSO	7.283
GSKA	7.433
DYYPO	7.700
MOS	8.750
TLBO-FL	8.833
PSO	11.500
SSA	12.067
AEO	12.467
Ant Lion	15.000

Tabla 1: Ranking Ant Lion - Dimensión 10

2.1.2. Dimensión 30

Algoritmo	Ranking
EBOwithCMAR	2.633
ELSHADE_SPACMA	2.883
jSO	2.933
MM_OED	3.667
RB-IPOP-CMA-ES	4.667
PPSO	8.067
MOS	8.317
GSKA	8.367
DYYPO	8.600
TLBO-FL	8.600
DE	8.700
SSA	12.033
PSO	12.600
AEO	12.933
Ant Lion	15.000

Tabla 2: Ranking Ant Lion - Dimensión 30

En ambas dimensiones el algoritmos ha queda en última lugar y con el peor valor de ranking posible. Este nos indica que sus resultados van a estar enormemente separados del algoritmo que está justo por encima de él.

2.2. Comparación de los resultados

Una vez vista la posición en la que se haya el algoritmo Ant Lion, voy a comparar los resultados en las distintas funciones con algoritmos de referencia clásicos, como DE y PSO; y algunos modernos pero más asequibles, AEO y SSA.

Viendo el resultado en el ranking no se espera que el algoritmo gane a alguno de los algoritmos, ni si quiera a PSO.

2.2.1. Dimensión 10

	AEO	Ant_LION	DE	PSO	SSA
Best	0	0	21	8	1
F01	5133778,78	27028194360	0	52551101,2843137	3765,829481794
F02	1	8,549867E+017	0	1	1
F03	357,5283114706	1342917	0	1988,8789721569	1E-10
F04	14,1446846675	5501,656	0,000110504	46,8426931529	4,1722461794
F05	38,7550811373	226,7146	115,0819266667	32,1200361373	48,4044775882
F06	26,1708063235	141,7755	34,5970701961	10,0100629392	26,321271602
F07	67,5254553529	239,7163	38,4805584314	42,7520633529	61,9804303725
F08	28,0813021569	146,6455	29,8291998039	22,0318125882	38,4871733431
F09	220,6905729608	3406,132	193,7879727451	56,8646680471	243,9286432745
F10	1171,6547498039	5138,309	359,6807647059	1076,8948962745	1085,7101735294
F11	97,0394383333	65026030	0,0194194193	38,427373402	95,2865849804
F12	252464,276678431	5721202000	4,9310863431	2516940,72382353	23134,8688411765
F13	778,7868645098	2841536000	5,9881427451	8408,5995323529	8101,3705870588
F14	74,5243931373	2215434000	0,0523983308	99,9258436863	118,1036978627
F15	204,4903833922	769546800	0,0606027963	2065,840625	524,3520335294
F16	180,410618451	1837,763	456,0592137255	141,4558021706	188,392458302
F17	89,6863405686	1583,008	23,5045311765	64,9718998627	97,6907592745
F18	1926,033136549	14468750000	0,0362999907	14840,0973715686	13604,2452390196
F19	64,182199598	12289130000	0,0051923244	3219,6057411765	124,601438902
F20	128,3756979804	1152,342	383,6904686275	84,4391033725	105,7736046078
F21	176,942994549	728,6146	188,8937176471	131,9519646078	104,3911663529
F22	114,7547752745	3102,498	100,481	77,4041808039	109,088398451
F23	342,7784139216	2035,93	809,7517235294	330,404375098	348,7045044889
F24	345,9974841373	992,2088	100	181,0388360392	259,6582712745
F25	434,801705098	2320,812	404,0113156863	447,8069123529	437,9551664706
F26	504,7361672353	3133,919	270,5882352941	372,9315798039	390,409494202
F27	419,7187245098	2355,893	389,728327451	413,4232378431	411,6924003922
F28	546,8650154902	1717,335	351,7333098039	469,7623388235	431,8347390196
F29	375,1733578431	46058,53	237,5455352941	319,2936092157	367,6340145098
F30	2470924,30004804	506074300	80512,174672549	635248,199333333	1469003,63445098

Tabla 3: Comparación resultados Ant Lion - Dimensión 10

El algoritmo Ant Lion no gana en ninguna de las funciones, pero es que además ha quedado última en todas ellas, llegando a enormes diferencias entre los algoritmos, como pasa con la función 19. Viendo los resultado en dimensión 10 no es muy buena opción su utilización dado que no mejora a los otros algoritmos que son más sencillos de implementar y de comprender.

2.2.2. Dimensión 30

Una vez vistos los resultados en dimensión 10, vamos a ver como se comporta el algoritmo si se aumenta la dimensión a 30.

	AEO	**Ant_LION**	**DE**	**PSO**	**SSA**
Best	0	0	24	2	3
F01	178164635,176471	76905917500	49094,7221568628	4175042517,64706	3765,1477069412
F02	1	2,134458E+061	1,308599303E+019	1	1
F03	29892,330254902	1088370000	3481,0788823529	54534,2773137255	2,41565E-05
F04	201,821947451	34919,15	84,3038629412	1183,1913260784	83,8822402093
F05	249,6142507843	626,0394	201,4980607843	217,0438413725	264,9099132157
F06	68,9419696471	147,8837	6,3202614647	36,9387168235	63,8378042745
F07	529,0749282353	960,5016	233,4221882353	359,582077451	460,8903094118
F08	189,3701482353	521,0267	189,3719607843	174,5205048039	220,5350423529
F09	5918,5534666667	33585,55	65,2973029608	2841,8463756863	6211,9975156863
F10	6042,3757784314	10296,47	3763,561372549	6938,0890215686	4843,5091568627
F11	398,0148978431	618581300	79,5827970588	1207,1347803922	153,7264097059
F12	85847397,8039216	29488190000	325765,150980392	358570597,215686	2528490,94411765
F13	1880356,84519608	44187810000	153,6136039216	45075853,7686275	192595,729509804
F14	17969,2468445098	1251168000	71,0022382353	305870,526392157	3286,4881323529
F15	61639,2648294118	6515670000	62,5567343137	273710,076019608	88523,2954313725
F16	2007,7714196078	25734,34	1319,2334901961	1568,2914596078	1657,9891337255
F17	822,3996780392	283873,3	480,8806464706	472,5475807843	847,1460447059
F18	310964,701019608	4736259000	61,2224523529	2170364,16843137	88546,4832941176
F19	1888271,53627451	6647938000	35,7226817647	1260193,66290196	150965,713019608
F20	720,7965280392	3496,869	275,1128247059	462,1450868627	645,0489992157
F21	447,7568772549	1136,054	325,4940980392	411,2915998039	450,236195098
F22	1987,1465217647	11053,25	100,2234960784	1026,5725205882	4196,221111
F23	840,9894833333	5760,65	534,6091764706	640,4297605882	836,5145307843
F24	878,78126	2796,969	605,9018411765	709,5055852941	926,3820403922
F25	505,5116776471	6745,541	387,027972549	686,3826468627	417,5792415686
F26	4529,0261901961	13633,49	403,7672980392	3369,3574784314	5521,6873580392
F27	812,6841731373	7947,232	492,5197803922	806,8056270588	655,0159421569
F28	554,6830056863	7448,291	394,2695647059	1105,0358203922	489,6094933333
F29	2088,1168627451	236014,7	1025,0945705882	1409,1257576471	1934,2986921569
F30	9838704,15098039	10274980000	3656,5519411765	13585754,5913726	2308203,21313726

Tabla 4: Comparación resultados Ant Lion - Dimensión 30

Viendo los resultados podemos comprobar como se repite la misma situación, no se gana a nadie pero es que encima queda último en todas las funciones, por lo que tampoco al aumentar de dimensión puede llegar a ser una opción como algoritmo a usar en problemas reales.

3. Análisis del algoritmo memético

Una vez que hemos comprobado como se comporta el algoritmo Ant Lion vamos a intentar convertirlo en un algoritmo memético añadiendo una búsqueda local, en este caso Solid-Wets. Esta búsqueda local se realizará cada cierto número de generaciones ó iteraciones y sobre ciertas soluciones. En general los algoritmos meméticos pueden ser una solución para auquellos algoritmos genéticos que tienen mucha exploración pero poca explotación, y además estos algoritmos añaden cierta velocidad a la ejecución del algoritmo.

Los parámetros en la realización de la BL son los siguientes:

- Número de generaciones entre BL: 20
- Número máximo de evaluaciones por BL: 500
- Porcentaje de individuos que realizan la BL: 10 %

Los individuos sobre los que se realiza la BL se eligen de forma aleatoria, dando posibilidad de mejora a soluciones que no tienen porque ser la mejores.

Esta nueva componente de BL hace que el algoritmo tenga una mayor explotación de la que tenía el algoritmo original.

3.1. Posición en el ranking de la competición

Para obtener la posición del algoritmo memético en la competición, se ha comparado con todos los algoritmos presentados en la competición CEC2017, obteniendo el ranking a partir de la media de los resultados obtenidos en cada función.

3.1.1. Dimensión 10

Algoritmo	**Ranking**
EBOwithCMAR	3.450
MM_OED	3.617
jSO	4.017
ELSHADE_SPACMA	4.683
RB-IPOP-CMA-ES	6.083
DE	7.117
PPSO	7.283
GSKA	7.433
DYYPO	7.700
MOS	8.750
TLBO-FL	8.833
PSO	11.500
SSA	12.067
AEO	12.467
Ant Lion Memético	15.000

Tabla 5: Ranking Ant Lion Memético - Dimensión 10

3.1.2. Dimensión 30

Algoritmo	**Ranking**
EBOwithCMAR	2.633
ELSHADE_SPACMA	2.883
jSO	2.933
MM_OED	3.667
RB-IPOP-CMA-ES	4.667
PPSO	8.067
MOS	8.317
GSKA	8.367
DYYPO	8.600
TLBO-FL	8.600
DE	8.700
SSA	12.033
PSO	12.600
AEO	12.933
Ant Lion Memético	15.000

Tabla 6: Ranking Ant Lion Memético - Dimensión 30

En ambas dimensiones sigue siendo el peor algoritmo, por lo que no se aprecia ninguna mejora respecto del algoritmo original.

3.2. Comparación de los resultados

Una vez vista la posición en la que se haya el algoritmo Ant Lion Memético, que es la misma que la del algoritmos Ant Lion original; voy a comparar los resultados en las distintas funciones con el algoritmo Ant Lion original, para ver si hay una mejora con el memético respecto del original, ya que la posición en el ranking es la misma.

3.2.1. Dimensión 10

	Ant_LION	**Ant_LION_MEMETICO**
Best	29	30
F01	27028194360	26977887032,438
F02	8,549867E+017	8,549867E+017
F03	1342917	1342917
F04	5501,656	5501,656
F05	226,7146	226,7146
F06	141,7755	141,7755
F07	239,7163	239,7163
F08	146,6455	146,6455
F09	3406,132	3406,132
F10	5138,309	5138,309
F11	65026030	65026030
F12	5721202000	5721202000
F13	2841536000	2841536000
F14	2215434000	2215434000
F15	769546800	769546800
F16	1837,763	1837,763
F17	1583,008	1583,008
F18	14468750000	14468750000
F19	12289130000	12289130000
F20	1152,342	1152,342
F21	728,6146	728,6146
F22	3102,498	3102,498
F23	2035,93	2035,93
F24	992,2088	992,2088
F25	2320,812	2320,812
F26	3133,919	3133,919
F27	2355,893	2355,893
F28	1717,335	1717,335
F29	46058,53	46058,53
F30	506074300	506074300

Tabla 7: Comparación resultados Ant Lion Memético - Dimensión 10

Los resultados obtenidos con el algoritmo Ant Lion Memético son en todos las funciones iguales, menos en la primera que varía ligeramente mejorando a la original. Teniendo en cuenta estos resultados no se puede decidir cuál de los dos algoritmos es mejor, para ello ser debería realizar una análisis de los tiempos de ambos algoritmos en las distintas funciones. En general los algoritmos meméticos suelen ser más rápidos que los algoritmos genéticos, dado que muchas de las evaluaciones las realiza la búsqueda local que suele ser más rápida que el algoritmo genético.

3.2.2. Dimensión 30

	Ant_LION	**Ant_LION_MEMETICO**
Best	29	30
F01	76905917500	76308287496,788
F02	2,134458E+061	2,134458E+061
F03	1088370000	1088370000
F04	34919,15	34919,15
F05	626,0394	626,0394
F06	147,8837	147,8837
F07	960,5016	960,5016
F08	521,0267	521,0267
F09	33585,55	33585,55
F10	10296,47	10296,47
F11	618581300	618581300
F12	29488190000	29488190000
F13	44187810000	44187810000
F14	1251168000	1251168000
F15	6515670000	6515670000
F16	25734,34	25734,34
F17	283873,3	283873,3
F18	4736259000	4736259000
F19	6647938000	6647938000
F20	3496,869	3496,869
F21	1136,054	1136,054
F22	11053,25	11053,25
F23	5760,65	5760,65
F24	2796,969	2796,969
F25	6745,541	6745,541
F26	13633,49	13633,49
F27	7947,232	7947,232
F28	7448,291	7448,291
F29	236014,7	236014,7
F30	10274980000	10274980000

Tabla 8: Comparación resultados Ant Lion Memético - Dimensión 30

Al aumentar la dimensión a 30, la situación sigue siendo la misma por lo que tampoco podemos decantarnos por uno u otro algoritmo sin realizar una análisis más intenso.

4. Análisis del algoritmo modificado

Viendo que tanto el algoritmo Ant Lion como el algoritmo Ant Lion Memético no destacan dentro de la competición CEC2017, he decido realizar las siguientes modificaciones a ambos algoritmos para ver si su comportamiento mejora:

- Modificación del descenso del rango de movimiento de las hormigas.
- Actualización de la lista de ant lions.

4.1. Modificaciones

4.1.1. Rango del movimiento

El rango de movimiento está compuesto por dos cotas, la inferior, c; y la superior, d.

En el algoritmo original estas cotas se van modificando con la siguiente fórmula:

$$c\_i = \frac{lb\_i}{I} \qquad d\_i = \frac{ub\_i}{I}$$

Siendo lb_i la cota inferior actual de la variable i, ub_i la cota superior de la variable i, e I un valor que se calcula.

La variable I se calcula con la siguiente fórmula:

$$I = 10^w \frac{t}{T}$$

Siendo t el número actual de iteraciones, T el número máximo de iteraciones, y w una variable que cambia de valor dependiendo del porcentaje de iteraciones respecto del máximo se hayan realizado.

Esta forma de calcular el rango nos da como resultado la siguiente gráfica:

Figura 1: Rango de movimiento en el algoritmo original

El eje X de la gráfica representa las distintas iteraciones del algoritmo. Como se puede apreciar en pocas iteraciones hemos pasado del máximo rango posible al mínimo rango posible, esto provoca una deficiente exploración en el algoritmo, y lo que explicaría los malos resultados, tanto del algoritmo original, como del algoritmo memético, que no soluciona este problema porque lo que hace es introducir más explotación pero no exploración.

Para solucionar este problema y tener una distribución entre exploración y explotación adecuada durante toda la ejecución del algoritmo, es decir mucha exploración al inicio y mucha explotación al final; he creado la siguiente fórmula con la que calcular el rango durante la ejecución.

La fórmula es la siguiente:

$$c\_i = lb\_i + iteraciones * incremento$$

$$d\_i = ub\_i - iteraciones * incremento$$

Siendo iteraciones el número actual de iteraciones e incremento una constante del algoritmo.

Como se puede comprobar la fórmula consiste en un decremento constante de la cota superior y en un incremento constante de la cota inferior.

Esta forma de calcular el rango nos da como resultado la siguiente gráfica:

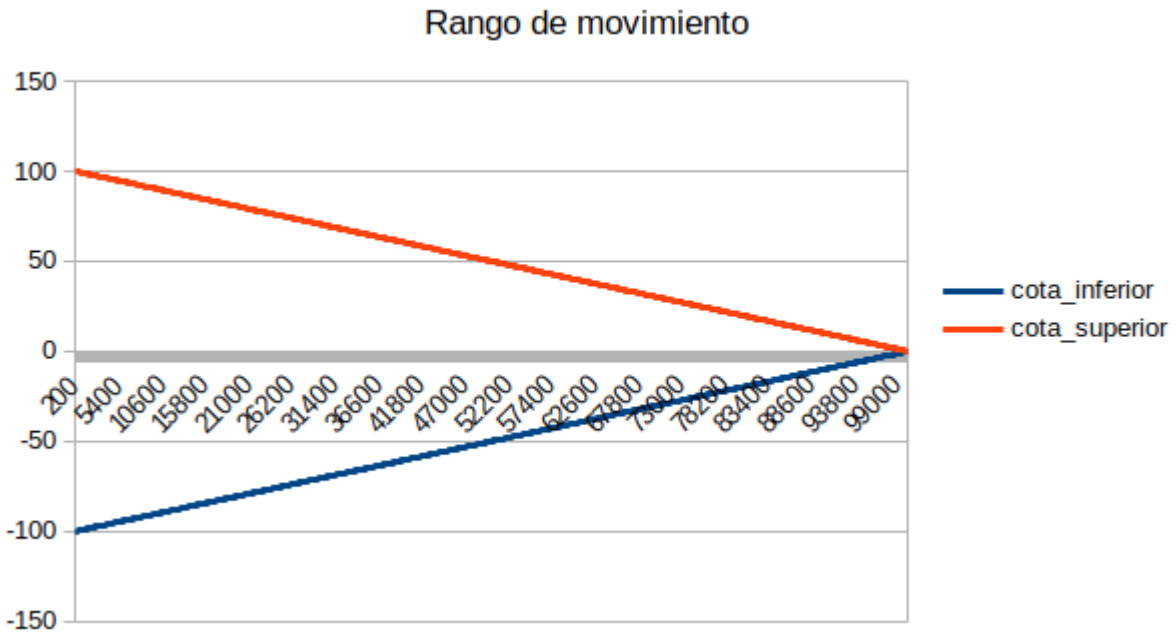

Figura 2: Rango de movimiento en el algoritmo modificado

4.1.2. Actualización de la lista de ant lions

En el problema original una ant lion se actualizaba si su fitness era mayor que el fitness de la hormiga a la que había afectado en su movimiento.

He decidido modificar esta forma de actualizar los ant lions, utilizando un método más elitista, que consiste en quedarme en cada iteración con los mejores en la lista de ant lions y el resto en la lista de hormigas. Por lo que si hay 20 individuos juntando ambas listas, los 10 mejores estarán en la lista de ant lions.

4.2. Análisis del algoritmo modificado

Una vez vistas las modificaciones realizadas hay que comprobar si dan buenos resultados o no.

4.2.1. Posición en el ranking de la competición

- Dimensión 10

Algoritmo	**Ranking**
EBOwithCMAR	3.483
MM_OED	3.617
jSO	4.017
ELSHADE_SPACMA	4.750
RB-IPOP-CMA-ES	6.250
DE	7.417
PPSO	7.517
GSKA	7.700
DYYPO	7.967
MOS	9.150
TLBO-FL	9.167
Ant_LION_MODI	10.700
PSO	12.300
SSA	12.733
AEO	13.233

Tabla 9: Ranking Ant Lion Modificado - Dimensión 10

La posición obtenida es buena respecto de la obtenida con el algoritmo original. Este nuevo algoritmo ha ganado a algoritmos como PSO ó SSA. Pero estos resultados nos indican que todavía no sigue siendo una opción viable, ya que todavía sigue habiendo algoritmos más sencillos, eficientes y modernos por encima de él.

- Dimensión 30

Algoritmo	**Ranking**
EBOwithCMAR	2.633
ELSHADE_SPACMA	2.883
jSO	2.933
MM_OED	3.667
RB-IPOP-CMA-ES	4.733
MOS	8.850
PPSO	8.567
GSKA	8.867
TLBO-FL	9.133
DYYPO	9.167
DE	9.200
Ant_LION_MODI	9.333
SSA	12.700
PSO	13.533
AEO	13.800

Tabla 10: Ranking Ant Lion Modificado - Dimensión 30

Al aumentar la dimensión a 30 sigue ganando a los mismos 3 algoritmos, pero en este caso con una mejor posición, por lo tanto viendo los resultados puede que el nuevo algoritmo mejore conforme aumenta la dimensión, para asegurarnos deberíamos probar dimensiones mayores y ver como se comporta.

4.3. Análisis del algoritmo memético modificado

Al ver que hay mejoras con el algoritmo modificado con la versión genética del algoritmo original, he decidido probar ha realizar la modificación en el algoritmo memético para ver si en este caso si hay mejoras respecto del genético modificado.

4.3.1. Posición en el ranking de la competición

- Dimensión 10

Algoritmo	**Ranking**
EBOwithCMAR	3.450
MM_OED	3.650
jSO	4.017
ELSHADE_SPACMA	4.750
RB-IPOP-CMA-ES	6.150
DE	7.383
PPSO	7.483
GSKA	7.567
DYYPO	7.867
MOS	9.050
TLBO-FL	9.167
Ant_LION_MODI	11.467
PSO	12.133
SSA	12.733
AEO	13.133

Tabla 11: Ranking Ant Lion Memético Modificado - Dimensión 10

En dimensión 10 ha seguido ganando a mismos tres algoritmos, pero ha conseguido una peor posición. Por lo tanto para dimensión 10 no es buena opción usar el algoritmo memético.

- Dimensión 30

Algoritmo	**Ranking**
EBOwithCMAR	2.633
ELSHADE_SPACMA	2.883
jSO	2.933
MM_OED	3.667
RB-IPOP-CMA-ES	4.667
PPSO	8.233
MOS	8.483
DYYPO	8.667
TLBO-FL	8.700
GSKA	8.700
DE	8.900
Ant_LION_MODI	11.733
SSA	12.733
PSO	13.367
AEO	13.700

Tabla 12: Ranking Ant Lion Memético Modificado - Dimensión 30

Al aumentar a dimensión 30 sigue sin ser buena opción usar el algoritmo memético.

Viendo los resultados obtenidos tanto en dimensión 10 como 30 podemos afirmar que el algoritmo memético con la modificación en ningún caso es mejor que el algoritmo genético modificado. Esto puede ser debido a que el algoritmo genético modificado tenga una distribución explotación-explotación adecuada.

Referencias

[1] Seyedali Mirjalili. "The Ant Lion Optimizer". En: *Advances in Engineering Software* 83 (2015), págs. 80-98. ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2015.01.010. URL: https://www.sciencedirect.com/science/article/pii/S0965997815000113.

[2] *Wikiversity: Ant Lion Optimizer*. URL: https://en.wikiversity.org/wiki/Ant_lion_optimizer.