UNIVERSIDAD
DE GRANADA

METAHEURÍSTICAS 2020/21

PRÁCTICA ALTERNATIVA

GREY WOLF OPTIMIZER

Tomás Ruiz Fernández
77385078-Z
tomasruiz@correo.ugr.es

Índice

1. Introducción

Este algoritmo está inspirado en el mecanismo de caza y la jerarquía de los lobos grises, consiste en una manada de lobos liderados por el *alfa*, *beta* y *delta*. Se puede observar en el comportamiento de los lobos una estricta jerarquía social, y en este caso imitaremos esa jerarquía dictando el movimiento del resto de lobos (*omegas*) según los tres lobos anteriormente mencionados.

Estos tres (las tres mejores soluciones) dictarán el movimiento pseudoaleatorio de los otros lobos. Para colocar a los lobos omega lo primero es tener en cuenta estas ecuaciones.

$$D = |C * Xp(t) - X(t)|$$
$$X(t+1) = Xp(t) - A * D$$

$$A = 2 * a * r1 - a$$
$$C = 2 * r2$$

[1] Ecuaciones GWO

Xp -> Posición de la presa (La solución *alfa*, *beta* o *delta*)
X -> Posición del lobo *omega*
t -> Paso
D -> Distancia
r1 -> Vector pseudoaleatorio de tamaño Dim con valores entre [0, 1)
r2 -> Vector pseudoaleatorio de tamaño Dim con valores entre [0, 1)
A -> Vector pseudoaleatorio
C -> Vector pseudoaleatorio
a -> Variable que va disminuyendo poco a poco, hace el radio de acción de los lobos *omega* más pequeño

Si nos detenemos a observar el comportamiento de estas ecuaciones vemos que el radio de movimiento dependerá de *a* y de la distancia entre los dos. *a* es una variable que irá disminuyendo (en mi caso lo he hecho linealmente) hasta llegar hasta 0 y empezando por 2. Los demás son factores aleatorios que determinarán en qué posición entre este área se ubican los lobos omega.

Al no saber dónde está la “presa” (la solución óptima), supondremos que los que mejor información tienen de la presa son las mejores soluciones.

Al tener tres lobos líderes lo que hacemos es hacer una media de las tres posiciones resultantes. Cada posición se calcula con unas *r* diferentes para que entre en juego el factor aleatorio.

Este algoritmo se divide en dos fases:

- **Acorralamiento:** El valor de *a* empieza por 2 y por tanto tenemos más área de acción, entonces es una fase que favorece mucho a la exploración. Esta fase se basa en la búsqueda de los lobos para cazar.

- **Ataque a la presa:** Cuando el valor de *a* es más pequeño realizamos una explotación de la solución mayor, básicamente ya hemos madurado nuestro conocimiento sobre la presa y toca atacar, por ende los “saltos” que pueden dar los lobos *omega* son muy pequeños.

Un esquema general del algortimo sería:

Initialize the grey wolf population Xi (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
Xα=the best search agent
Xβ=the second best search agent
Xδ=the third best search agent
while (t < Max number of iterations)
    for each search agent
        Update the position of the current search agent by equation (3.7)
    end for

    Update a, A, and C
    Calculate the fitness of all search agents
    Update Xα, Xβ, and Xδ t=t+1
end while

return Xα

[1] Pseudocódigo GWO

2. Pruebas y análisis

A la hora de probar este algoritmo la primera duda que tenía era sobre qué tamaño de población de lobos usar, he hecho varias pruebas y estos son los resultados:

Dimensión = 10
T = 10

Functions	GWO (20)	GWO (50)	GWO (100)	GWO (500)
F01	7,92E+11	1,89E+11	1,73E+11	1,59E+11
F02	2,08E+12	1,15E+10	1,60E+10	3,32E+09
F03	4,42E+06	2,17E+06	1,36E+06	2,60E+05
F04	3,61E+04	2,91E+04	2,79E+04	1,43E+04
F05	3,46E+04	3,68E+04	3,32E+04	3,68E+04
F06	1,50E+04	1,10E+04	1,11E+04	8,91E+03
F07	5,35E+04	5,55E+04	5,06E+04	4,99E+04
F08	2,86E+04	2,94E+04	2,68E+04	2,52E+04
F09	1,56E+05	6,38E+04	1,51E+04	1,60E+04
F10	1,20E+06	1,04E+06	1,02E+06	1,05E+06
F11	5,04E+04	7,50E+04	5,11E+04	4,70E+04
F12	3,70E+09	1,41E+09	1,80E+09	2,26E+09
F13	1,46E+07	1,31E+07	1,49E+07	1,46E+07
F14	1,57E+06	1,21E+06	8,48E+05	1,06E+05
F15	3,15E+06	2,00E+06	9,16E+05	9,21E+05
F16	2,47E+05	2,13E+05	1,28E+05	9,42E+04
F17	6,93E+04	6,95E+04	6,31E+04	6,23E+04
F18	2,72E+07	3,54E+07	4,44E+07	2,84E+07
F19	2,13E+06	2,52E+06	2,97E+06	1,43E+06
F20	1,15E+05	9,56E+04	1,08E+05	6,81E+04
F21	2,32E+05	2,18E+05	2,09E+05	1,59E+05
F22	1,24E+05	1,31E+05	1,29E+05	1,22E+05
F23	3,44E+05	3,42E+05	3,35E+05	3,35E+05
F24	3,43E+05	3,44E+05	3,46E+05	3,22E+05

F25	4,63E+05	4,33E+05	4,41E+05	4,48E+05
F26	9,11E+05	5,35E+05	3,97E+05	3,90E+05
F27	4,03E+05	4,02E+05	4,05E+05	3,97E+05
F28	5,69E+05	5,41E+05	5,38E+05	4,94E+05
F29	3,36E+05	3,39E+05	2,99E+05	2,89E+05
F30	5,16E+08	5,66E+08	7,12E+08	6,46E+08
Best	2	4	4	20

Por tiempo, solo he podido hacer estas pruebas, y observo que el tamaño tiene que ser grande, ya que el conocimiento al inicializar a los lobos es mayor ya que es aleatorio, y por ende el algoritmo funciona bien con tamaños de población grandes.

Ahora bien, tenemos un máximo de evaluaciones y un mayor tamaño de la población supone un menor número de iteraciones, es bueno tener más conocimiento del espacio de soluciones pero también hay que dejar que el algoritmo se desarrolle.

Para el máximo de iteraciones he considerado:

Iteraciones = Máx iteraciones / Nº Lobos

** Revisando me he dado cuenta de un fallo, debería restar 1 a esa divisón ya que evaluamos a los lobos antes de entrar al bucle.

He llegado a la conclusión de que se podría estimar según el tamaño del espacio de soluciones una cifra de población “idónea”.

Para inicializar utilizo numpy y genero vectores de la dimensión correspondiente con valores entre [-100, 100].

Todo lo demás lo hago siguiendo la base teórica, adjunto el código del algoritmo.

Pruebas comparando con otros algoritmos:

Dimensión = 10
T = 10
Población = 500

Evaluations: 100%

Functions	AEO	DE	GWO	PSO

F01	5.134e+06	0.000e+00	1.590e+08	5.255e+07
F02	1.000e+00	0.000e+00	3.315e+06	1.000e+00
F03	3.575e+02	0.000e+00	2.597e+02	1.989e+03
F04	1.414e+01	1.105e-04	1.429e+01	4.684e+01
F05	3.876e+01	1.151e+02	3.684e+01	3.212e+01
F06	2.617e+01	3.460e+01	8.911e+00	1.001e+01
F07	6.753e+01	3.848e+01	4.988e+01	4.275e+01
F08	2.808e+01	2.983e+01	2.518e+01	2.203e+01
F09	2.207e+02	1.938e+02	1.600e+01	5.686e+01
F10	1.172e+03	3.597e+02	1.047e+03	1.077e+03
F11	9.704e+01	1.942e-02	4.704e+01	3.843e+01
F12	2.525e+05	4.931e+00	2.262e+06	2.517e+06
F13	7.788e+02	5.988e+00	1.464e+04	8.409e+03
F14	7.452e+01	5.240e-02	1.062e+02	9.993e+01
F15	2.045e+02	6.060e-02	9.210e+02	2.066e+03

F16	1.804e+02	4.561e+02	9.415e+01	1.415e+02
F17	8.969e+01	2.350e+01	6.225e+01	6.497e+01
F18	1.926e+03	3.630e-02	2.843e+04	1.484e+04
F19	6.418e+01	5.192e-03	1.426e+03	3.220e+03
F20	1.284e+02	3.837e+02	6.807e+01	8.444e+01
F21	1.769e+02	1.889e+02	1.592e+02	1.320e+02
F22	1.148e+02	1.005e+02	1.224e+02	7.740e+01
F23	3.428e+02	8.098e+02	3.346e+02	3.304e+02
F24	3.460e+02	1.000e+02	3.216e+02	1.810e+02
F25	4.348e+02	4.040e+02	4.480e+02	4.478e+02
F26	5.047e+02	2.706e+02	3.895e+02	3.729e+02
F27	4.197e+02	3.897e+02	3.967e+02	4.134e+02
F28	5.469e+02	3.517e+02	4.943e+02	4.698e+02
F29	3.752e+02	2.375e+02	2.889e+02	3.193e+02
F30	2.471e+06	8.051e+04	6.457e+05	6.352e+05

Best	0	21	4	5

Dimensión = 30
T = 5 (Por tiempo, mi pc no da para más)
Población = 500

Evaluations: 100%

Functions	AEO	DE	GWO	PSO
F01	1.782e+08	4.909e+04	4.079e+09	4.175e+09
F02	1.000e+00	1.309e+19	8.439e+25	1.000e+00
F03	2.989e+04	3.481e+03	3.099e+04	5.453e+04
F04	2.018e+02	8.430e+01	3.147e+02	1.183e+03
F05	2.496e+02	2.015e+02	2.299e+02	2.170e+02
F06	6.894e+01	6.320e+00	2.862e+01	3.694e+01
F07	5.291e+02	2.334e+02	3.038e+02	3.596e+02
F08	1.894e+02	1.894e+02	2.200e+02	1.745e+02
F09	5.919e+03	6.530e+01	1.745e+03	2.842e+03
F10	6.042e+03	3.764e+03	6.625e+03	6.938e+03
F11	3.980e+02	7.958e+01	4.785e+02	1.207e+03

F12	8.585e+07	3.258e+05	3.850e+08	3.586e+08
F13	1.880e+06	1.536e+02	9.883e+07	4.508e+07
F14	1.797e+04	7.100e+01	4.269e+05	3.059e+05
F15	6.164e+04	6.256e+01	3.635e+06	2.737e+05
F16	2.008e+03	1.319e+03	1.620e+03	1.568e+03
F17	8.224e+02	4.809e+02	5.594e+02	4.725e+02
F18	3.110e+05	6.122e+01	1.324e+06	2.170e+06
F19	1.888e+06	3.572e+01	4.709e+06	1.260e+06
F20	7.208e+02	2.751e+02	6.357e+02	4.621e+02
F21	4.478e+02	3.255e+02	4.049e+02	4.113e+02
F22	1.987e+03	1.002e+02	1.973e+03	1.027e+03
F23	8.410e+02	5.346e+02	5.765e+02	6.404e+02
F24	8.788e+02	6.059e+02	6.622e+02	7.095e+02
F25	5.055e+02	3.870e+02	5.239e+02	6.864e+02
F26	4.529e+03	4.038e+02	3.377e+03	3.369e+03

F27	8.127e+02	4.925e+02	5.610e+02	8.068e+02
F28	5.547e+02	3.943e+02	6.980e+02	1.105e+03
F29	2.088e+03	1.025e+03	1.200e+03	1.409e+03
F30	9.839e+06	3.657e+03	2.092e+07	1.359e+07
Best	1	27	0	3

El algoritmo no rinde mal, está como en una media entres los otros tres, por supuesto creo que es un algoritmo que se le puede sacar mucho provecho, ya que explora muy bien el entorno de soluciones. Creo que encontrando un buen tamaño de la población daría mejores resultados. Obviamente depende mucho de la aleatoriedad, para todas las pruebas he cogido estas semillas:

seeds = [22, 82, 128, 333, 505, 234, 5345, 4574, 707, 4543]

Hablando de la eficiencia es un algoritmo que abusa mucho de la generación de números pseudoaleatorios y eso al final pasa factura.

En la ejecución con dimensión 30 utilizo el mismo número de lobos y creo que esto es un error, debería ajustarlo según la dimensión como se observa en los resultados. En general creo que utilizo una población demasiado pequeña.

3. Hibridación

Pensando cómo hibridar el sistema llegué a dos conclusiones, la primera era ir intercalando los dos algoritmos (GWO y Solis Wets) pero esto fue una mala idea ya que debido al tamaño que le viene bien al algoritmo GWO, las iteraciones que se ejecutan por algoritmo de una tirada son muy pocas, y no permite al algoritmo desarrollarse, hice una prueba pero fue desastrosa.

Entonces pensé en el comportamiento del algoritmo GWO y pensé en reemplazar esa fase, de ataque a la presa, por el algoritmo de búsqueda local con saltos Solis.

Por ende introduzco una variable escalado para que la variable *a* del algoritmo GWO se reduzca menos y por ende no acabe en 0, así evitamos parte del proceso final de ataque a la presa.

Reducción de *a*:

*a = a - 2(iteraciones*escalado)*

El algoritmo consiste en ejecutar un porcentaje de las iteraciones con el GWO y el resto con Solis.

Las pruebas las realizo con ejecutando un 70% con GWO y un 30% con Solis Wets.

Para la variable Delta decido un valor fijo pequeño, en este caso 10.

Pseudocódigo:

Inicio

*Ejecutar GWO con -> Iteraciones*0.7 (El propio algoritmo las reduce según la población como he comentado antes)*

Extraemos el lobo alfa

*Ejecutamos Solis -> Sol=Lobo alfa, Iteraciones*0.3*

Fin

Resultados:

Comparatica GWO vs GWO_Solis:

Functions	GWO, DIM=10	GWO_Solis, DIM=10	GWO, DIM=30	GWO_Solis, DIM=30
F01	1,59E+11	2,70E+10	4,08E+12	1,68E+12
F02	3,32E+09	8,57E+07	8,44E+28	1,65E+28
F03	2,60E+05	1,39E+05	3,10E+07	3,81E+07
F04	1,43E+04	8,89E+03	3,15E+05	2,17E+05
F05	3,68E+04	2,52E+04	2,30E+05	1,69E+05
F06	8,91E+03	4,68E+03	2,86E+04	2,49E+04
F07	4,99E+04	4,47E+04	3,04E+05	2,67E+05
F08	2,52E+04	1,61E+04	2,20E+05	1,49E+05
F09	1,60E+04	4,57E+03	1,75E+06	1,57E+06
F10	1,05E+06	8,45E+05	6,63E+06	5,39E+06
F11	4,70E+04	2,91E+04	4,79E+05	3,31E+05
F12	2,26E+09	1,20E+09	3,85E+11	1,13E+11
F13	1,46E+07	8,72E+06	9,88E+10	2,07E+10
F14	1,06E+05	9,58E+04	4,27E+08	8,91E+07
F15	9,21E+05	7,69E+05	3,64E+09	4,47E+08
F16	9,42E+04	1,13E+05	1,62E+06	1,00E+06
F17	6,23E+04	5,65E+04	5,59E+05	5,10E+05
F18	2,84E+07	2,38E+07	1,32E+09	7,87E+08
F19	1,43E+06	2,37E+06	4,71E+09	1,90E+09
F20	6,81E+04	5,64E+04	6,36E+05	4,68E+05
F21	1,59E+05	1,54E+05	4,05E+05	3,72E+05
F22	1,22E+05	1,15E+05	1,97E+06	1,57E+06
F23	3,35E+05	3,31E+05	5,77E+05	5,39E+05
F24	3,22E+05	3,11E+05	6,62E+05	6,54E+05
F25	4,48E+05	4,37E+05	5,24E+05	4,53E+05
F26	3,90E+05	3,52E+05	3,38E+06	2,99E+06
F27	3,97E+05	3,96E+05	5,61E+05	5,53E+05

F28	4,94E+05	4,85E+05	6,98E+05	5,56E+05
F29	2,89E+05	2,98E+05	1,20E+06	1,10E+06
F30	6,46E+08	7,24E+08	2,09E+10	1,20E+10
Best	4	26	1	29

Vemos que la propuesta de mezclar de esta forma los algoritmos funciona bien, mejora bastante al algoritmo y además en aplaca un poco mi poca experiencia decidiendo la población de lobos que usar.

Creo que es una buena combinación usar la primera etapa de este algoritmo para hacer una exploración y la última me parece sustituible por un algoritmo que explote la solución bien.

En conclusión me parece un algoritmo interesante con potencial para devolver buenas soluciones, y con un poco de trabajo no está mal en tema de eficiencia.

Como último dejo los resultados de la hibridación con el resto de algoritmos.

Dimensión = 10
T = 10
Población = 500
Delta = 10

Evaluations: 100%				
Functions	**AEO**	**DE**	**GWO_Solis**	**PSO**
F01	5,13E+09	0,00E+00	2,70E+10	5,26E+10
F02	1,00E+03	0,00E+00	8,57E+07	1,00E+03
F03	3,58E+05	0,00E+00	1,39E+05	1,99E+06
F04	1,41E+04	1,11E-01	8,89E+03	4,68E+04
F05	3,88E+04	1,15E+05	2,52E+04	3,21E+04
F06	2,62E+04	3,46E+04	4,68E+03	1,00E+04
F07	6,75E+04	3,85E+04	4,47E+04	4,28E+04
F08	2,81E+04	2,98E+04	1,61E+04	2,20E+04
F09	2,21E+05	1,94E+05	4,57E+03	5,69E+04
F10	1,17E+06	3,60E+05	8,45E+05	1,08E+06
F11	9,70E+04	1,94E+01	2,91E+04	3,84E+04
F12	2,53E+08	4,93E+03	1,20E+09	2,52E+09

F13	7,79E+05	5,99E+03	8,72E+06	8,41E+06
F14	7,45E+04	5,24E+01	9,58E+04	9,99E+04
F15	2,05E+05	6,06E+01	7,69E+05	2,07E+06
F16	1,80E+05	4,56E+05	1,13E+05	1,42E+05
F17	8,97E+04	2,35E+04	5,65E+04	6,50E+04
F18	1,93E+06	3,63E+01	2,38E+07	1,48E+07
F19	6,42E+04	5,19E+00	2,37E+06	3,22E+06
F20	1,28E+05	3,84E+05	5,64E+04	8,44E+04
F21	1,77E+05	1,89E+05	1,54E+05	1,32E+05
F22	1,15E+05	1,01E+05	1,15E+05	7,74E+04
F23	3,43E+05	8,10E+05	3,31E+05	3,30E+05
F24	3,46E+05	1,00E+05	3,11E+05	1,81E+05
F25	4,35E+05	4,04E+05	4,37E+05	4,48E+05
F26	5,05E+05	2,71E+05	3,52E+05	3,73E+05
F27	4,20E+05	3,90E+05	3,96E+05	4,13E+05
F28	5,47E+05	3,52E+05	4,85E+05	4,70E+05
F29	3,75E+05	2,38E+05	2,98E+05	3,19E+05
F30	2,47E+09	8,05E+07	7,24E+08	6,35E+08
Best	0	21	6	3

Dimensión = 30
T = 5
Población = 500
Delta = 10

Evaluations: 100%				
Functions	**AEO**	**DE**	**GWO_Solis**	**PSO**
F01	1,78E+11	4,91E+07	1,68E+12	4,18E+12
F02	1,00E+03	1,31E+22	1,65E+28	1,00E+03
F03	2,99E+07	3,48E+06	3,81E+07	5,45E+07
F04	2,02E+05	8,43E+04	2,17E+05	1,18E+06
F05	2,50E+05	2,02E+05	1,69E+05	2,17E+05

F06	6,89E+04	6,32E+03	2,49E+04	3,69E+04
F07	5,29E+05	2,33E+05	2,67E+05	3,60E+05
F08	1,89E+05	1,89E+05	1,49E+05	1,75E+05
F09	5,92E+06	6,53E+04	1,57E+06	2,84E+06
F10	6,04E+06	3,76E+06	5,39E+06	6,94E+06
F11	3,98E+05	7,96E+04	3,31E+05	1,21E+06
F12	8,59E+10	3,26E+08	1,13E+11	3,59E+11
F13	1,88E+09	1,54E+05	2,07E+10	4,51E+10
F14	1,80E+07	7,10E+04	8,91E+07	3,06E+08
F15	6,16E+07	6,26E+04	4,47E+08	2,74E+08
F16	2,01E+06	1,32E+06	1,00E+06	1,57E+06
F17	8,22E+05	4,81E+05	5,10E+05	4,73E+05
F18	3,11E+08	6,12E+04	7,87E+08	2,17E+09
F19	1,89E+09	3,57E+04	1,90E+09	1,26E+09
F20	7,21E+05	2,75E+05	4,68E+05	4,62E+05
F21	4,48E+05	3,26E+05	3,72E+05	4,11E+05
F22	1,99E+06	1,00E+05	1,57E+06	1,03E+06
F23	8,41E+05	5,35E+05	5,39E+05	6,40E+05
F24	8,79E+05	6,06E+05	6,54E+05	7,10E+05
F25	5,06E+05	3,87E+05	4,53E+05	6,86E+05
F26	4,53E+06	4,04E+05	2,99E+06	3,37E+06
F27	8,13E+05	4,93E+05	5,53E+05	8,07E+05
F28	5,55E+05	3,94E+05	5,56E+05	1,11E+06
F29	2,09E+06	1,03E+06	1,10E+06	1,41E+06
F30	9,84E+09	3,66E+06	1,20E+10	1,36E+10
Best	1	25	3	2

4. Bibliografía

[1] Mirjalili, Seyedali & Mirjalili, Seyed & Lewis, Andrew. (2014). Grey Wolf Optimizer. Advances in Engineering Software. 69. 46–61. 10.1016/j.advengsoft.2013.12.007.