METAHEURÍSTICAS

Práctica alternativa: Grey Wolf Optimizer.

Mario Muñoz Mesa
mario43514@correo.ugr.es

28 de junio de 2022

Índice

1. Descripción GWO.

Grey Wolf Optimizer (GWO) se trata de una técnica tipo Swarm Intelligence (SI); esto es, una población de agentes navegan usando una inteligencia social y colectiva simulada.

Además los algoritmos SI presenta algunas ventajas:

- Preservan información del espacio de búsqueda durante las iteraciones mientras que algoritmos evolutivos olvidan la información del generaciones previas.
- Normalmente guardan la mejor solución hasta el momento.
- Normalmente tienen pocos parámetros para ajustar.
- Tienen menos operadores comparado con algoritmos evolutivos (cruce, mutación, elitismo...)
- Son fáciles de implementar.

En concreto GWO está inspirado en la jerarquía social y de comportamiento de caza de manadas de lobo gris.

1.1. Inspiración.

Los lobos grises viven en manada, en grupos de unos 5 a 12 en promedio, y destacan por ser grandes depredadores. Conviven en una jerarquía social dominante, de más a menos dominante: α, β, δ, ω. Resumidamente:

- Alphas.

  Los alphas machos y hembras lideran la manada. Son responsables de tomar decisiones sobre la caza, lugar para dormir, momento de levantarse,... etc. Las decisiones de los alphas son acatadas por la manada. Sin embargo también existe cierto comportamiento democrático donde los alphas siguen a los otros lobos de la manada.

- Betas.

  Los betas son los de segundo nivel en la jerarquía. Son lobos subordinados que ayudan a los alphas en la toma de decisiones u otras actividades de la manada. Los betas respetan al alpha pero ordenan a lobos de jerarquía inferior. También refuerzan las órdenes del alpha en la manada y retroalimentan al alpha.

- Omegas.

  Los más bajos en la jerarquía son los omegas. Los omegas tienen que obedecer las órdenes del resto de lobos dominantes. Son los últimos en comer. Parece que no juegan un rol importante pero sin ellos se han observado problemas y peleas en la manada en caso de perder a omegas. Los omegas satisfacen la estructura dominante de la manada.

- Deltas.

  Los lobos que no son alphas, betas u omegas son llamados subordinados o deltas. Los lobos delta tienen que obedecer a los laphas y betas pero dominan a los omega. Algunos ejemplos pueden ser: antiguos alphas, lobos que protegen a la manada... .

Otra característica interesante es la forma de cazar. Las fases son:

- Rastrear, seguir y acercarse a la presa.
- Perseguir, rodear y hostigar a la presa hasta que deje de moverse.
- Atacar a la presa.

1.2. Modelo matemático y algoritmo.

Se presentan los modelos matemáticos de la jerarquía social, seguimiento, rodeamiento y ataque a la presa.

1.2.1. Jerarquía social.

Para modelar matemáticamente la jerarquía de lobos grises, se considera la primera solución como la alpha. La segunda y tercera solución serán beta y delta respectivamente. El resto de soluciones candidatas se considerarán omega. La caza estará guiada por α, β y δ, los lobos ω siguen a los 3 anteriores.

1.2.2. Rodear a la presa.

Como se mencionó, los lobos grises rodean a la presa durante la caza. Las ecuaciones matemáticas propuestas para modelar esto son:

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)|$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D}$$

donde t indica la iteración actual, $\vec{A}$ y $\vec{C}$ son vectores de coeficientes reales, $\vec{X}_p$ es el vector de posición de la presa, y $\vec{X}$ denota el vector de posición de un lobo gris.

Los vectores $\vec{A}$ y $\vec{C}$ se calculan como:

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a}$$

$$\vec{C} = 2 \cdot \vec{r}_2$$

donde $\vec{a}$ decrementa linealmente de 2 a 0 en el transcurso de iteraciones y $\vec{r}_1$, $\vec{r}_2$ son vectores aleatorios con valores en $[0, 1]$.

Con estas ecuaciones, un lobo gris en la posición (X, Y) puede actualizar su posición en base a la posición (X^*, Y^*) de la presa. Se pueden alcanzar diferentes posiciones alrededor del mejor agente respecto a la posición actual ajustando los valores de $\vec{A}$ y $\vec{D}$. Por ejemplo $(X^* - X, Y^*)$ puede alcanzarse con $\vec{A} = (1, 0)$ y $\vec{C} = (1, 1)$. Se muestra figura que ilustra esto mismo, junto a extensión 3D, igualmente se puede extender a espacios de búsqueda de n dimensiones.

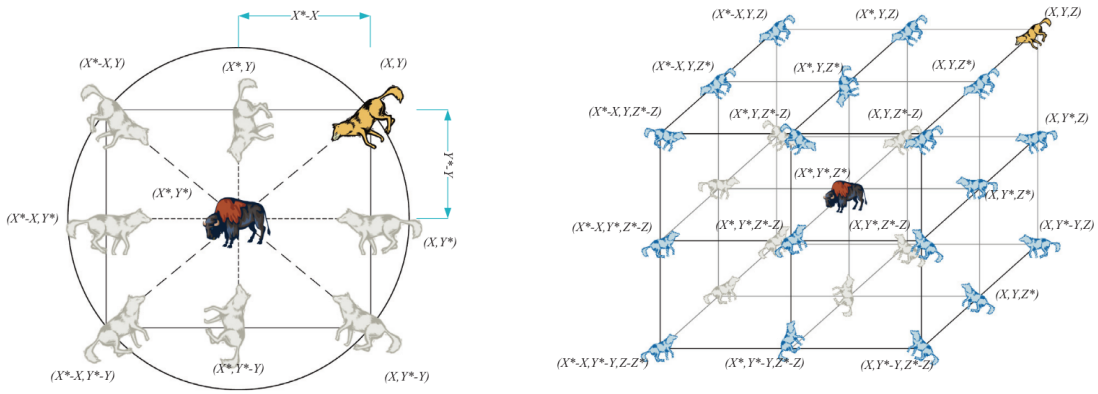

Figura 1: Posibles posiciones alrededor de presa.

Los vectores $\vec{r}_1$ y $\vec{r}_2$ permiten cualquier posición intermedia a las ilustradas. Así un lobo gris podría moverse alrededor de la mejor posición hasta el momento a cualquier posición en el espacio de búsqueda usando las ecuaciones anteriores.

1.2.3. Cazar.

Los lobos grises suelen reconocer y rodear a la presa. La caza usualmente es guiada por el alpha. El beta y delta ocasionalmente también participan. Sin embargo en nuestro espacio de búsqueda abstracto no se conoce la localización del óptimo (presa). Para manejar esto, se supone que el alpha, beta y delta tienen mejor conocimiento de la localización de la presa. Por tanto se guardan las 3 mejores soluciones hasta el momento y se obliga al resto de agentes de búsqueda a actualizar sus posiciones acorde a la posición de lo mejores agentes de búsqueda. Las fórmulas propuestas son:

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|, \quad \vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|, \quad \vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}|$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot (\vec{D}_\alpha), \quad \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot (\vec{D}_\beta), \quad \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot (\vec{D}_\delta)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \qquad (3.7)$$

La siguiente figura ilustra el posible movimiento de omegas en base a las posiciones de alpha, beta y delta.

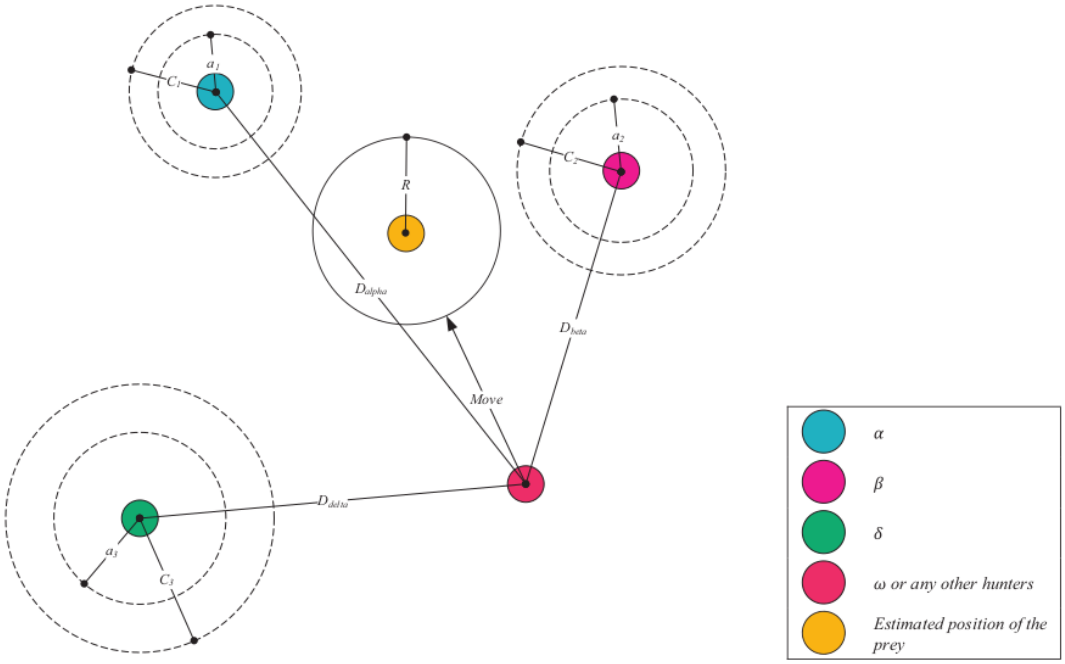

Figura 2: Actualización de posición.

1.2.4. Atacar a la presa (explotación).

Los lobos grises finalizan la caza atacando a la presa cuando deja de moverse. Para modelar el acercamiento a la presa se decrementa el valor de $\vec{a}$. El rango de fluctuación de $\vec{A}$ también decrementa por $\vec{a}$, esto es, $\vec{A}$ toma valor aleatorio en el intervalo $[-2a, 2a]$, donde a decrementa de 2 a 0 durante las iteraciones. Cuando los valores de $\vec{A}$ están en $[-1, 1]$, la siguiente posición del agente de búsqueda puede ser cualquier posición entre la suya y la de la presa. Cuando $|\vec{A}| < 1$ el lobo ataca a la presa.

Con los operadores ya propuestos, el algoritmos GWO permite a los agentes actualizar su posición en base a la localización de alpha, beta y delta; y atacar a la presa.

1.2.5. Buscar a la presa (exploración).

Los lobos grises mayormente buscan en base a la posición de alpha, beta y delta. Divergen entre sí para buscar la presa y convergen para atacarla. Para modelar esto matemáticamente cuando

$|\vec{A}| > 1$ el agente buscador diverge de la presa. Esto enfatiza la exploración de GWO.

Otra componente que favorece la exploración es $\vec{C}$. El vector $\vec{C}$ toma valores en $[0, 2]$. Esta componente provee pesos aleatorios para acentuar ($|\vec{C}| > 1$) o minorar ($|\vec{C}| < 1$) el efecto de la presa. Esto ayuda a que GWO tenga un comportamiento más aleatorio, favoreciendo la exploración y evitando quedar atrapado en óptimos locales.

1.2.6. Pseudocódigo.

El proceso de búsqueda comienza creando una población de lobos grises (soluciones candidatas). Conforme se itera, los lobos alpha, beta y delta estiman la probable posición de la presa. Cada solución candidata actualiza su posición a la presa. El parámetro a decrementa de 2 a 0 para enfatizar la exploración y explotación respectivamente. Las soluciones candidatas tienden a diverger de la presa cuando $|\vec{A}| > 1$ y converger cuando $|\vec{A}| < 1$. Finalmente el algoritmo termina con una condición de parada como puede ser un máximo de iteraciones.

Initialize the grey wolf population X_i (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
X_α=the best search agent
X_β=the second best search agent
X_δ=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 X_α, X_β, and X_δ
  t=t+1
end while
return X_α

Figura 3: Pseudocódigo.

2. Adaptación de GWO a APC.

2.1. Descripción del problema APC.

Nos ocupamos del problema del Aprendizaje de Pesos en Características (APC). Plantea, en el contexto de un problema de clasificación, elegir o ajustar un vector de pesos $\mathbf{w} = (w_1, \ldots, w_d) \in [0, 1]^d$ asociado a las características en base a un criterio de mejora del clasificador. Hemos denotado con d al número de características.

En nuestro caso trabajaremos con clasificador tipo 1-NN, y el criterio será maximizar:

$$F(\mathbf{w}) := \alpha \cdot \text{tasa\_clas}(\mathbf{w}) + (1 - \alpha) \cdot \text{tasa\_red}(\mathbf{w})$$

donde

$$\text{tasa-clas} := 100\frac{\text{nº instancias bien clasificadas en training}}{\text{nº instancias en training}}, \qquad \text{tasa-red} := 100\frac{\text{nº valores } w_i < 0.2}{\text{nº características}}$$

y $alpha = 0.5$ que pondera la importancia entre el acierto y la reducción de características para el nuevo mejor clasificador que se pretende encontrar.

De lo que nos ocupamos por tanto es de obtener $\arg\max_{\mathbf{w}\in[0,1]^d} F(\mathbf{w})$ para un clasificador 1-NN que utilizará la distancia ponderada

$$d_{\mathbf{w}}(u,v) = \sqrt{\sum_{i=1}^{d} w_i(u_i - v_i)^2}, \quad u, v \in \mathbb{R}^d$$

para clasificar. Queremos que aumente el acierto y reduzca el número de características, cualquier característica con peso menor estricto a 0.2 se descarta.

2.2. Funciones y estructuras.

Se ha decidido crear una estructura, `Wolf`, que contiene: un `vector<double>` con el vector de posiciones, y su fitness asociado.

Además se han definido los comparadores mayor, $>$, y menor, $<$. Un lobo es mayor a otro si su función objetivo es mayor, y menor si la función objetivo es menor.

Para evaluar la función objetivo asociada a la posición de cada lobo y asignarlo al lobo se ha creado la función `obj_to_wolf`:

Algorithm 1: obj_to_wolf

Input: Wolf w, vector de elementos muestrales *training*
begin
  for $i = 0$ ***to*** $training.size()$ **do**
    $class\_labels.push\_back(\ one\_NN\_lo(training[i], training, w.pos, i)\ )$
  end
  $w.obj \leftarrow obj\_function(class\_rate(class\_labels, training),\ red\_rate(w.pos))$
end

Dado que nuestro problema está restringido a pesos entre 0 y 1, se ha creado una función que restringe las posiciones de los lobos a $[0, 1]$

Algorithm 2: restrict_01

Input: Wolf w
begin
  for $i = 0$ ***to*** $num\_feats - 1$ **do**
    if $w.pos[i] < 0$ **then**
      $w.pos[i] \leftarrow 0$
    end
    else
      if $w.pos[i] > 1$ **then**
        $w.pos[i] \leftarrow 1$
      end
    end
  end
end

También se ha creado una función que inicializa las posiciones de los lobos:

Algorithm 3: init_wolf_pos

Input: Wolf w, distribución uniforme $rand\_real\_01$
begin
  $w.pos.resize(num\_feats)$
  for $i = 0$ ***to*** $num\_feats - 1$ **do**
    $w.pos[i] \leftarrow rand\_real\_01(generator)$
  end
end

Y una función para actualizar las posiciones y valor de función objetivo del alpha, beta y delta:

Algorithm 4: update_alpha_beta_delta

Input: Agentes (lobos) $wolfs$, lobo alpha $alpha$, lobo beta $beta$, lobo delta $delta$
begin
  for w ***in*** $wolfs$ **do**
    if $w.obj > alpha.obj$ **then**
      $alpha.obj \leftarrow w.obj$
      $alpha.pos \leftarrow w.pos$
    end
    else
      if $w.obj > beta.obj$ ***and*** $w.obj < alpha.obj$ **then**
        $beta.obj \leftarrow w.obj$
        $beta.pos \leftarrow w.pos$
      end
      else
        if $w.obj > delta.obj$ ***and*** $w.obj < beta.obj$ ***and*** $w.obj < alpha.obj$ **then**
          $delta.obj \leftarrow w.obj$
          $delta.pos \leftarrow w.pos$
        end
      end
    end
  end
end

Mostramos finalmente el pseudocódigo de GWO adaptado a nuestro problema, problema de maximización, donde la condición de parada será por llegar a un máximo número de evaluaciones de la función objetivo (se ha procurado que se realicen exactamente las evaluaciones indicadas para tener resultados comparables al resto):

Algorithm 5: gwo

Input: vector de elementos muestrales *training*, número de agentes (lobos) a usar *num_agents*, número máximo de evaluaciones de la función objetivo *max_evals*

Output: posición del alpha tras ejecutar todas las iteraciones *alpha.pos*

begin

$n \leftarrow num\_feats$ `// número componentes posición igual al número de características del problema`

`// inicialización de los agentes`

for $i = 0$ ***to*** $num\_agents - 1$ **do**

$init\_wolf\_pos(w, \mathcal{U}(0,1))$

$obj\_to\_wolf(w, training)$

$wolfs.push\_back(w)$

end

$sort(wolfs)$ `// ordenar por función objetivo`

`// se guardan alpha, beta y delta`

$alpha \leftarrow wolfs[num\_agents - 1]$

$beta \leftarrow wolfs[num\_agents - 2]$

$delta \leftarrow wolfs[num\_agents - 3]$

$evals \leftarrow num\_agents; \quad it \leftarrow 1$

while $evals < max\_evals$ **do**

$a \leftarrow 2 - 2 \cdot \frac{it}{max\_iters}$

`// Actualiza las posiciones de cada agente incluyendo omegas`

for w ***in*** $wolfs$ ***and*** $evals < max\_evals$ **do**

for $j = 0$ ***to*** $n - 1$ **do**

$r1 \leftarrow u \in \mathcal{U}(0,1); \quad r2 \leftarrow u \in \mathcal{U}(0,1)$

$A1 \leftarrow 2 \cdot a \cdot r1 - a$

$C1 \leftarrow 2 \cdot r2$

$D_\alpha \leftarrow |C1 \cdot alpha\_pos[j] - w.pos[j]|$

$X1 \leftarrow alpha\_pos[j] - A1 \cdot D_\alpha$

$r1 \leftarrow u \in \mathcal{U}(0,1); \quad r2 \leftarrow u \in \mathcal{U}(0,1)$

$A2 \leftarrow 2 \cdot a \cdot r1 - a$

$C2 \leftarrow 2 \cdot r2$

$D_\beta \leftarrow |C2 \cdot beta\_pos[j] - w.pos[j]|$

$X2 \leftarrow alpha\_pos[j] - A2 \cdot D_\beta$

$r1 \leftarrow u \in \mathcal{U}(0,1); \quad r2 \leftarrow u \in \mathcal{U}(0,1)$

$A3 \leftarrow 2 \cdot a \cdot r1 - a$

$C3 \leftarrow 2 \cdot r2$

$D_\delta \leftarrow |C3 \cdot delta\_pos[j] - w.pos[j]|$

$X3 \leftarrow delta\_pos[j] - A3 \cdot D_\delta$

$w.pos[j] \leftarrow \frac{X1+X2+X3}{3}$

end

$restrict\_01(w)$

$obj\_to\_wolf(w, training)$

$evals \leftarrow evals + 1$

end

`// Actualiza alpha, beta y delta`

$update\_alpha\_beta\_delta(wolfs, alpha, beta, delta)$

$it \leftarrow it + 1$

end

return *alpha.pos*

end

2.3. Procedimiento considerado en el desarrollo.

Para el desarrollo de la práctica no se ha utilizado ningún framework, se han implementado las funciones necesarias en C++. Todo el código se encuentra en `pwo.cpp` .

Para la ejecución basta indicar un argumento con el archivo a ejecutar, en este caso se utiliza la semilla por defecto 1234, que es la misma que se utilizó para los resultados de esta memoria.

También se incluye un fichero `pwo_all_datasets.sh` que ejecuta, con la semilla por defecto, `pwo` sobre los 3 datasets

2.4. Análisis de resultados.

Los 3 datasets con los que se ha trabajado son:

- **Ionosphere**: Conjunto de datos de radar que fueron recogidos por un sistema *Goose Bay*, Labrador. Consta de 352 instancias con 34 características y 2 clases.
- **Parkinsons**: Conjunto de datos orientado a distinguir entre la presencia y ausencia de la enfermedad de Parkinson. Consta de 195 ejemplos con 22 características y 2 clases.
- **Spectf-Heart**: Conjunto de datos de detección de enfermedades cardiacas a partir de imágenes médicas de tomografía computerizada del corazón de pacientes. Consta de 267 ejemplos con 44 características y 2 clases.

Se muestra la tabla resumen:

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
1-NN	68.89334	0	34.44672	0.3458	38.60188	0	19.30094	0.0934	75.563	0	37.78152	0.4114
Relief	87.74648	3.529414	45.63794	1.7594	95.89542	4.545452	50.22044	0.3688	83.87396	0	41.93698	2.2224
BL	89.2032	85.2941	87.24868	3255.044	88.278	79.99998	84.139	396.8584	85.31934	85.9091	85.61422	5350.368
AGG-BLX	86.04024	86.47058	86.25542	22970.16	87.76384	88.1818	87.97282	4628.332	86.5126	83.18182	84.84722	25376.7
AGG-CA	87.46882	81.17646	84.32264	20903.28	91.80432	85.45454	88.62944	4654.054	86.4622	77.27276	81.86746	24268.34
AGE-BLX	86.61974	88.2353	87.4275	23650.94	88.2517	88.18182	88.21672	4683.564	86.2101	85.45456	85.8323	24587.5
AGE-CA	84.6197	83.5294	84.07456	21042.54	89.72538	89.0909	89.40814	5315.516	84.47058	80.45454	82.46258	24840.7
AM-(10,1.0)	89.17104	88.2353	88.70318	23181.3	86.63292	88.18182	87.40736	4859.698	84.75628	91.36364	88.05998	27185.14
AM-(10,0.1)	84.61168	90.58826	87.59996	24288.08	88.18556	87.27272	87.72914	5184.384	84.8067	86.36364	85.5852	26415.84
AM-(10,0.1mej)	84.6197	90.00002	87.30984	24726.3	87.21256	88.1818	87.69718	5253.436	85.59666	86.36362	85.98012	27331
ES	86.0443	87.0588	86.5515	24078.8	87.1862	87.2727	87.2295	4531.11	84.7731	85.4545	85.1138	27283.2
BMB	88.6036	87.6471	88.1253	20831.4	92.7901	86.3636	89.5769	4260.19	86.7647	85	85.8824	24984.8
ILS	86.8934	90.5882	88.7408	20233.8	90.7375	90.9091	90.8233	2603.94	84.7563	91.8182	88.2872	27572.8
ILS-ES	84.9014	92.3529	88.6272	22766.7	90.3165	90	90.1582	4651.03	83.916	90.4545	87.1853	26372.9
GWO	81.7827	92.9412	**87.3619**	20679.1	84.5547	89.0909	**86.8228**	4370.57	83.6471	94.0909	**88.869**	27055.2

Nota: en azul el mejor resultado de algoritmos poblacionales, en morado el mejor resultado de los basados en trayectorias, y en negrita el de GWO.

Nota: basados en la propia inspiración del algoritmo se han tomado 10 lobos o agentes.

Nota: para que sea comparable con el resto de algoritmos también se ha impuesto 15000 como número máximo de evaluaciones de la función objetivo.

Vemos que se han obtenido unos resultados decentes en los 3 datasets, de hecho en Spect-Heart se obtiene un nuevo máximo. En Ionosphere se obtiene resultado cercano a los mejores, en Parkinsons sin embargo queda por debajo de los resultados del resto de algoritmos. Se observa dependencia del dataset para los resultados.

Excepto en el caso de Spectf-Heart no se observan mejores resultados que los algoritmos basados en trayectorias. Sí cabe mencionar que supera a ES, algoritmo que también disminuye la exploración cuando se acerca a solución prometedora, en Ionosphere y Spectf-Heart.

Como nuestro algoritmo es poblacional puede tener más interés compararlo con los poblacionales. En este caso se observan, en general, resultados similares a los mejores de los genéticos a pesar de

usar en GWO una población menor. No queda atrás como algoritmo poblacional a pesar de tener métodos de exploración y explotación bastante diferentes al resto. Parece interesante ver cómo afectaría una búsqueda local pues en los meméticos esa búsqueda local supuso mejoras en 2 de los 3 datasets.

En todos los casos se obtienen resultados mejores que algoritmos como BL que considerábamos de comparación para el resto de algoritmos.

En cuanto a los tiempos de ejecución se obtienen tiempos similares al resto de algoritmos poblacionales, nada destacable.

3. Hibridación con Búsqueda Local.

Se ha decidido introducir Búsqueda Local sobre los lobos alpha, beta y delta que son los que dirigen al resto hacia la solución prometedora. Estas búsquedas serán realizadas cada 30 iteraciones.

La búsqueda local realizada será de baja intensidad, se realizarán $2n$ evaluaciones de la función objetivo siendo n el número de características que presente el dataset. La idea es intentar mejorar las soluciones más prometedoras, y que conducen al a resto, cada cierto número de iteraciones, intentando llegar así a una mejor solución.

Se muestra el pseudocódigo de la búsqueda local de baja intensidad:

Algorithm 6: li_ls

Input: conjunto de elementos muestrales *training*, cromosoma c, evaluaciones de f.obj ya realizadas *obj_eval_count*

begin

$n \leftarrow c.genes.size()$ `// equivale a number_of_features`

`// inicialización de índices de genes`

for $i = 0$ ***to*** n **do**

$comp\_indexes.push\_back()$

end

$best\_obj \leftarrow c.obj$

$gen\_neighbours \leftarrow 0$

$mod\_pos \leftarrow 0$

while $gen\_neighbours < 2 * n$ ***and*** $obj\_eval\_count < 15000$ **do**

`// aleatorizamos componentes a mutar si se han recorrido todas o se mejoró f.obj`

if new_best_obj ***or*** $mod\_pos \,\%number\_of\_features == 0$ **then**

$new\_best\_obj \leftarrow false$

$shuffle(comp\_indexes)$

$mod\_pos \leftarrow 0$

end

`// Se toma componente a mutar y se muta`

$comp\_to\_mut \leftarrow comp\_indexes[mod\_pos \,\%n]$

$muted\_c \leftarrow c$

$mutation(muted\_c.genes,\ comp\_to\_mut,\ \mathcal{N}(0, 0.3))$

$gen\_neighbours \leftarrow gen\_neighbours + 1$

`// se asigna f. obj a cromosoma con los pesos mutados (con leave one out)`

$obj\_to\_chromosome(muted\_c,\ training)$

`// si se ha mejorado, actualizamos mejor objetivo, cromosoma y vecinos generados`

if $muted\_c.obj > best\_obj$ **then**

$c \leftarrow muted\_c$

$best\_obj \leftarrow obj$

$new\_best\_obj \leftarrow true$

end

$obj\_eval\_count \leftarrow obj\_eval\_count + 1$

$mod\_pos \leftarrow mod\_pos + 1$

end

end

Nota: también se controla la condición de 15000 evaluaciones de la función objetivo.

De esta forma el pseudocódigo de `gwo_ls` es igual a `gwo` añadiendo la comprobación del número de iteraciones para realizar la búsqueda local. Queda tal que así:

Algorithm 7: gwo_ls

Input: vector de elementos muestrales *training*, número de agentes (lobos) a usar *num_agents*, número máximo de evaluaciones de la función objetivo *max_evals*

Output: posición del alpha tras ejecutar todas las iteraciones *alpha.pos*

begin

...

`// inicialización de los agentes`

for $i = 0$ ***to*** $num\_agents - 1$ **do**

...

end

...

while $evals < max\_evals$ **do**

...

`// Actualiza las posiciones de cada agente incluyendo omegas`

for w ***in*** $wolfs$ ***and*** $evals < max\_evals$ **do**

for $j = 0$ ***to*** $n - 1$ **do**

...

end

...

end

`// Actualiza alpha, beta y delta`

$update\_alpha\_beta\_delta(wolfs, alpha, beta, delta)$

if $it\,\%30 == 0$ **then**

$li\_ls(training, alpha, evals)$

$li\_ls(training, beta, evals)$

$li\_ls(training, delta, evals)$

end

$it \leftarrow it + 1$

end

return *alpha.pos*

end

3.1. Análisis de resultados.

Se muestra la tabla resumen:

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
1-NN	68.89334	0	34.44672	0.3458	38.60188	0	19.30094	0.0934	75.563	0	37.78152	0.4114
Relief	87.74648	3.529414	45.63794	1.7594	95.89542	4.545452	50.22044	0.3688	83.87396	0	41.93698	2.2224
BL	89.2032	85.2941	87.24868	3255.044	88.278	79.99998	84.139	396.8584	85.31934	85.9091	85.61422	5350.368
AGG-BLX	86.04024	86.47058	86.25542	22970.16	87.76384	88.1818	87.97282	4628.332	86.5126	83.18182	84.84722	25376.7
AGG-CA	87.46882	81.17646	84.32264	20903.28	91.80432	85.45454	88.62944	4654.054	86.4622	77.27276	81.86746	24268.34
AGE-BLX	86.61974	88.2353	87.4275	23650.94	88.2517	88.18182	88.21672	4683.564	86.2101	85.45456	85.8323	24587.5
AGE-CA	84.6197	83.5294	84.07456	21042.54	89.72538	89.0909	89.40814	5315.516	84.47058	80.45454	82.46258	24840.7
AM-(10,1.0)	89.17104	88.2353	88.70318	23181.3	86.63292	88.18182	87.40736	4859.698	84.75628	91.36364	88.05998	27185.14
AM-(10,0.1)	84.61168	90.58826	87.59996	24288.08	88.18556	87.27272	87.72914	5184.384	84.8067	86.36364	85.5852	26415.84
AM-(10,0.1mej)	84.6197	90.00002	87.30984	24726.3	87.21256	88.1818	87.69718	5253.436	85.59666	86.36362	85.98012	27331
ES	86.0443	87.0588	86.5515	24078.8	87.1862	87.2727	87.2295	4531.11	84.7731	85.4545	85.1138	27283.2
BMB	88.6036	87.6471	88.1253	20831.4	92.7901	86.3636	89.5769	4260.19	86.7647	85	85.8824	24984.8
ILS	86.8934	90.5882	88.7408	20233.8	90.7375	90.9091	90.8233	2603.94	84.7563	91.8182	88.2872	27572.8
ILS-ES	84.9014	92.3529	88.6272	22766.7	90.3165	90	90.1582	4651.03	83.916	90.4545	87.1853	26372.9
GWO	81.7827	92.9412	**87.3619**	20679.1	84.5547	89.0909	**86.8228**	4370.57	83.6471	94.0909	**88.869**	27055.2
GWO-LS	85.493	91.1765	**88.3347**	20170.2	89.2375	90.9091	**90.0733**	4917.15	86.7815	93.1818	**89.9817**	26389.8

Nota: en azul el mejor resultado de algoritmos poblacionales, en morado el mejor resultado de los basados en trayectorias, y en negrita el de GWO.

Nota: basados en la propia inspiración del algoritmo se han tomado 10 lobos o agentes.

Nota: para que sea comparable con el resto de algoritmos también se ha impuesto 15000 como número máximo de evaluaciones de la función objetivo.

Tal y como ocurrió con los algoritmos genéticos, la hibridación con búsqueda local supone una mejora de resultados. De hecho ahora se consigue superar a los algoritmos genéticos en todos los datasets.

Observamos que el uso de búsqueda local de baja intensidad en GWO, GWO-LS, ha supuesto mejora en los 3 datasets respecto a GWO; dedicar más evaluaciones de función objetivo a explotar soluciones con BL ha resultado efectivo, puede que GWO peque de explotar poco con sus dos componentes de exploración. Se han conseguido resultados muy cercanos a los mejores tanto en Ionosphere como Parkinsons, y se ha obtenido nuevo mejor resultado en Spectf-Heart.

Como era de esperar no se han incrementado los tiempos de ejecución; ahora se dedican evaluaciones de función objetivo a BL que sabemos que tiene tiempos de ejecución bastante bajos.

4. Experimentación.

Se ha experimentado con mayores tamaños de población para GWO y GWO-LS, en concreto con 20 y 30 agentes. El código para esto se encuentra en `gwo_exp.cpp`. Se muestran los resultados:

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
1-NN	68.89334	0	34.44672	0.3458	38.60188	0	19.30094	0.0934	75.563	0	37.78152	0.4114
Relief	87.74648	3.529414	45.63794	1.7594	95.89542	4.545452	50.22044	0.3688	83.87396	0	41.93698	2.2224
BL	89.2032	85.2941	87.24868	3255.044	88.278	79.99998	84.139	396.8584	85.31934	85.9091	85.61422	5350.368
AGG-BLX	86.04024	86.47058	86.25542	22970.16	87.76384	88.1818	87.97282	4628.332	86.5126	83.18182	84.84722	25376.7
AGG-CA	87.46882	81.17646	84.32264	20903.28	91.80432	85.45454	88.62944	4654.054	86.4622	77.27276	81.86746	24268.34
AGE-BLX	86.61974	88.2353	87.4275	23650.94	88.2517	88.18182	88.21672	4683.564	86.2101	85.45456	85.8323	24587.5
AGE-CA	84.6197	83.5294	84.07456	21042.54	89.72538	89.0909	89.40814	5315.516	84.47058	80.45454	82.46258	24840.7
AM-(10,1.0)	89.17104	88.2353	88.70318	23181.3	86.63292	88.18182	87.40736	4859.698	84.75628	91.36364	88.05998	27185.14
AM-(10,0.1)	84.61168	90.58826	87.59996	24288.08	88.18556	87.27272	87.72914	5184.384	84.8067	86.36364	85.5852	26415.84
AM-(10,0.1mej)	84.6197	90.00002	87.30984	24726.3	87.21256	88.1818	87.69718	5253.436	85.59666	86.36362	85.98012	27331
ES	86.0443	87.0588	86.5515	24078.8	87.1862	87.2727	87.2295	4531.11	84.7731	85.4545	85.1138	27283.2
BMB	88.6036	87.6471	88.1253	20831.4	92.7901	86.3636	89.5769	4260.19	86.7647	85	85.8824	24984.8
ILS	86.8934	90.5882	88.7408	20233.8	90.7375	90.9091	90.8233	2603.94	84.7563	91.8182	88.2872	27572.8
ILS-ES	84.9014	92.3529	88.6272	22766.7	90.3165	90	90.1582	4651.03	83.916	90.4545	87.1853	26372.9
GWO	81.7827	92.9412	87.3619	20679.1	84.5547	89.0909	86.8228	4370.57	83.6471	94.0909	88.869	27055.2
GWO-LS	85.493	91.1765	88.3347	20170.2	89.2375	90.9091	**90.0733**	4917.15	86.7815	93.1818	**89.9817**	26389.8
GWO20	86.6036	92.3529	**89.4783**	101867	87.7112	90.9091	89.3101	23075.8	85.3613	93.1818	89.2716	125041
GWO20-LS	85.7425	92.9412	89.3418	99135.5	83.1059	90.9091	87.0075	23455.8	85.6387	92.7273	89.183	127089
GWO30	84.9054	92.3529	89.4783	101867	89.83	90.9091	89.3101	23075.8	85.0588	93.1818	89.2716	125041
GWO30-LS	88.6157	90.5882	89.3418	99135.5	89.2638	90.9091	87.0075	23455.8	84.4622	93.6364	89.183	127089

El tamaño de la población no ha supuesto grandes mejoras, aunque sí se consiguió mejorar los resultados de Ionosphere. Sí se observa que con mayor tamaño de población no se obtuvieron mejores resultados con el algoritmo híbrido; la mejora que se obtenía con BL no es consistente con mayor población.

Aplicar la búsqueda local a menos de los 3 agentes que dirigen al resto (alpha, beta y delta); es decir, solo alpha, solo beta, alpha y beta... etc, tampoco supuso mejoras.

5. Referencias.

- https://www.sciencedirect.com/science/article/abs/pii/S0965997813001853