Práctica final: Implementación de algoritmo YSGA

Instrucciones:

Implementa el algoritmo metaheurístico YSGA en Matlab. Se evaluará al comparar los resultados de su implementación con los resultados de la versión original, una diferencia demasiado grande causará la pérdida de puntos en esta práctica.

Entregable:

Código .m del algoritmo.

Descripción básica del algoritmo:

Algoritmo metaheurístico inspirado en la estrategia de cacería del pez Yellowsaddle Goatfish. Cuenta con cinco operadores de movimiento principales:

- Inicialización
- Persecución por el pez cazador líder
- Acorralamiento por los peces bloqueadores
- Cambio de roles
- Cambio de zonas

Formulación de los operadores de movimiento

Inicialización:

Una población de **N** individuos es aleatoriamente generada en el espacio de búsqueda. Una vez generada, la población total **N** se divide en **K** subgrupos utilizando el algoritmo de agrupamiento *K-means*.

Estos subgrupos representan los diferentes grupos de cacería del pez. Cada individuo cuenta con los siguientes parámetros:

- Calidad de su solución (Fitness)
- Posición actual (Position)

Los subgrupos pueden ser representados por una celda de matrices que contenga todos los grupos con sus parámetros, o por matrices individuales para cada grupo que contengan los parámetros previamente listados.

Como parte de la inicialización, se identifican los mejores individuos de cada grupo y el mejor de ellos se destaca como el mejor global (GlobalBest). Este GlobalBest deberá ser actualizado

cada iteración y se compone de su posición (GlobalBest Position), su calidad (GlobalBest Fitness) y el número del grupo que le corresponde.

Persecución por el pez cazador líder:

Cada subgrupo tendrá un solo cazador líder, el cuál deberá ser quien tenga la mejor calidad de solución (fitness) de todo el subgrupo. Su comportamiento de persecución es modelado utilizando la función del *vuelo de Lévy*.

La función del *vuelo de Lévy* requiere tres parámetros: cantidad de pasos a generar, cantidad de dimensiones del problema y un parámetro *Beta* que controla la distancia de los pasos.

El parámetro *Beta* es determinado con la siguiente formulación:

```
beta = 1.99 + (.001*iteracionActual/(iteracionMaxima/10));
if beta > 2
    beta = 2;
end
```

El *vuelo de Levy* es utilizado de la siguiente manera:

```
levySteps = levyStep(10,dimensiones,beta);
```

La línea anterior nos generaría un vector de 10 pasos aleatorios.

Finalmente, los movimientos del pez cazador líder son influenciados por la posición del mejor global, siendo generados con la siguiente formulación:

```
if (PosicionLider-posicionGlobalBest) ~= 0
    for i = 1:10
        levySteps(i,:) = levySteps(i,:).*(PosicionLider-posicionGlobalBest);
        posicionesNuevas(:,i) = posicionLider + levySteps(i,:);
    end
else
    for i = 1:dimensiones
        posicionesNuevas (i,:) = posicionLider + levySteps(i,:);
    end
end
```

Las diez posiciones generadas se evalúan para conocer su calidad (Fitness), si alguna de ellas es mejor a la calidad actual del líder, se reemplaza al líder por esa nueva posición y calidad.

Acorralamiento por los peces bloqueadores:

El movimiento de los peces bloqueadores utiliza la ecuación de la espiral logarítmica para lograr el comportamiento de acorralamiento. Este movimiento está formulado de la siguiente manera:

```
alpha = -1+iteracion*((-1)/iteracionMaxima);
```

```
b = 1;

for i = 2:cantidadPecesGrupo
    p = (alpha-1)*rand+1;
    D = zeros(1,dim);
    for k = 1:dimensiones
       r = (rand*2-1);
       D(k) = abs(posicionLiderGrupo(k) * r – posicionPez(k));
       PosicionNueva = D(k) * exp(b.*p) .* cos(p.*2*pi) + posicionLiderGrupo;
    end
end
```

Este comportamiento debe ser realizado por todos aquellos peces de un grupo que no sean el cazador lider. Los parámetros *alpha, b* y *p* son parte de la ecuación de la espiral logarítmica y vienen de su matemática. La variable *D* representa la distancia entre la posición actual del pez que se moverá hacia el lider, modificada ligeramente por un valor aleatorio *r*.

Cambio de roles:

El cambio de roles sucede de manera natural al actualizar en cada iteración al pez que tiene la mejor calidad de solución (Fitness). Siempre el pez líder será quien tenga la mejor calidad del grupo.

Cambio de zona:

El cambio de zona sucede cuando un grupo ha explorado por completo su zona y ya no está encontrando ninguna mejor solución. En cada iteración, después del movimiento de persecución del pez cazador líder, si ninguno de los 10 movimientos generados resultó en una mejor solución, se incrementa un contador de estancamiento ya que no hubo mejora. Cuando este contador llega a 10, se considera que la zona ha sido explorada y se realiza el cambio de zona en dirección a la mejor solución global con la siguiente formulación:

```
for i=1:dimensiones
     posicionActual(i,:) =  (GlobalBestPosition(i) + posicionActual (i,:))/2;
end
```

Este movimiento se realiza para cada uno de los individuos del grupo estancado y al finalizar, se reinicia el contador de estancamiento del grupo a 0.

Pseudocódigo:

Pseudo-code

1. Input: m, k, t_{max}, **S**
2. Initialize the goatfish population $\boldsymbol{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ \ldots, \boldsymbol{p}_m\}$
3. Calculate the fitness value of each particle
4. Identify the global best $\boldsymbol{\Phi}_{best}$
5. Partition the population $\boldsymbol{P}$ into k clusters $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k\}$

6. Identify the chaser $\boldsymbol{\Phi}_l$ and the blocker $\boldsymbol{\varphi}_g$ fish for every cluster
7. While ($t < t_{max}$)
8. For every cluster $\boldsymbol{c}_l$
9. Execute hunting routine for chaser fish
10. Execute blocking routine for blocker fish
11. Calculate the fitness value of each goatfish
12. If $\boldsymbol{\varphi}_g$ has better fitness value than $\boldsymbol{\Phi}_l$
13. exchange roles by updating $\boldsymbol{\Phi}_l$
14. End If
15. If $\boldsymbol{\Phi}_l$ has better fitness value than $\boldsymbol{\Phi}_{best}$
16. update $\boldsymbol{\Phi}_{best}$
17. End If
18. If the fitness value of $\boldsymbol{\Phi}_l$ has not improved
19. $q \leftarrow q + 1$
20. End If
21. If $q > \lambda$
22. execute a routine for changing zone
23. $q \leftarrow 0$
24. End If
25. End For
26. $t \leftarrow t + 1$
27. End While
28. Output $\boldsymbol{\Phi}_{best}$

Table 1. Pseudo-code of the YSGA algorithm