# ALGORITHME GREYLAG GOOSE OPTIMISATION

## Définition

L'algorithme Greylag Goose Optimization (GGO), proposé par El-Kenawy et al. en 2024, est basé sur le concept d'intelligence en essaim et est conçu pour résoudre des problèmes d'optimisation complexes. Cet algorithme s'inspire du comportement social et de l'activité dynamique des oies migratrices, qui doivent parcourir de longues distances en optimisant leurs déplacements pour économiser de l'énergie, notamment en adoptant des formations en V. Ce comportement coordonné des oies a inspiré la conception d'un algorithme d'optimisation capable de gérer des problèmes complexes de manière efficace.

## Modélisation

L'algorithme Greylag Goose Optimization (GGO),commence par la génération aléatoire de plusieurs agents, chaque agent représentant une solution candidate $S_i (i = 1,2,...,n)$, où $n$ désigne la taille du troupeau ou le nombre d'agents.

L'algorithme GGO divise le troupeau en deux groupes : le premier groupe d'oies avec $n_1$ oies pour exploration et un deuxieme groupe avec $n_2$ oies pour l'exploitation. Le nombre d'agents ou de solutions dans chaque groupe est géré dynamiquement à chaque itération en fonction de la meilleure solution. Ainsi, le nombre d'agents dans le groupe d'exploration $n_1$ diminue tandis que celui du groupe d'exploitation $n_2$ augmente. Cependant, si la valeur de la fonction objective de la meilleure solution ne change pas pendant trois itérations consécutives, l'algorithme augmente le nombre d'agents dans le groupe d'exploration $n_1$ afin de trouver une nouvelle meilleure solution et d'éviter les optima locaux.

Dans la phase d'exploration, l'algorithme GGO suit trois stratégies, La première stratégie est appliquée si le nombre d'itérations $t \ mod \ 2 = 0$ Dans ce cas, le GGO met à jour la position des agents selon deux paramètres. Le premier paramètre est $r$ si $r < 0.5$, on vérifie le second paramètre $A$ si $|A| < 1$, les agents explorent de nouveaux endroits près de leur emplacement actuel afin de trouver une meilleure solution en utilisent l'équation1 :

$$S_i^{t+1} = S^* - A \cdot |C \cdot (S^* - S_i^t)| \tag{1}$$

Où :

$S_i^t$, désigne la $ième$ solution à l'itération $t$.
$S^*$, désigne la solution la meilleure obtenue jusqu'à présent.

Les valeurs $A$ et $C$ sont mises à jour pendant les itérations : $A = 2\alpha r_1 - \alpha$ et $C = 2r_2$ .
Où :

$\alpha$, est un paramètre qui varie de manière linéairement de 2 à 0.
$r_1$ et $r_2$, sont des valeurs qui changent dans l'intervalle [0, 1] de manière aléatoire.

Lorsque $|A| \geq 1$, on peut forcer une exploration plus large en choisissant trois agents ou solutions aléatoires, nommés $S_1$, $S_2$ et $S_3$. Cela permet de forcer les agents à ne pas être influencés par une position de leader et ainsi obtenir une plus grande exploration. Dans ce cas, la solution actuelle sera mise à jour à l'aide de l'équation (2) :

$$S_i^{t+1} = w_1 \cdot S_1 + z \cdot w_2 \cdot (S_2 - S_3) + (1 - z) \cdot w_3 \cdot (S_i^t - S_1) \tag{2}$$

Où :
$w_1$, $w_2$ et $w_3$, sont des valeurs qui varient aléatoirement entre [0, 2].
$z$, est une valeur calculée à l'aide de l'équation (3) :

$$z = 1 - (\frac{t}{T})^2 \tag{3}$$

Où :
$t$, désigne l'itérations actuel.
$T$, désigne le nombre maximum d'itérations.

Pendant la phase d'exploration décrite précédemment, une deuxième stratégie est mis en place lorsque la valeur du paramètre $\alpha$ et la valeur du second paramètre $A$ sont diminuées et que la valeur du nombre aléatoire du paramètre $r_3$ est supérieur ou égal à 0.5. Dans ce cas, l'emplacement des agents sera mis à jour à l'aide de l'équation (4) :

$$S_i^{t+1} = w_4 |S^* - S_i^t| \exp^{bl} \cos(2\pi l) + [2w_1(r_4 + r_5)]S^* \tag{4}$$

Où :
$b$, est une constante.
$l$, est une valeur aléatoire dans l'intervalle $[-1, 1]$.
$w_4$, désigne une valeur aléatoire entre [0, 2].
$r_4$ et $r_5$, ce sont des valeurs aléatoire entre $[0, 1]$.

Cependant, si $t \mod 2 \neq 0$ alors la troisième stratégie sera adoptée. Dans ce cas, l'emplacement des agents sera mis à jour à l'aide de l'équation (5) :

$$S_i^{t+1} = S_i^t + D(1 + z) \cdot w \cdot (S_i^t - S^*) \tag{5}$$

L'équipe d'exploitation est chargée d'améliorer les solutions déjà en place en utilisant deux stratégies différentes pour atteindre son objectif d'exploitation. La première stratégie est utilisée pour progresser dans la direction de la meilleure solution, en utilisant trois agents $Sl_1$, $Sl_2$ et $Sl_3$ qui guident d'autres agents pour changer leurs positions vers la meilleure position estimée. L'équations suivantes (6) montre le processus de mise à jour de la position :

$$\begin{aligned} S_1^t &= Sl_1 - A_1 \cdot |C_1 \cdot Sl_1 - S| \\ S_2^t &= Sl_2 - A_2 \cdot |C_1 \cdot Sl_2 - S| \\ S_3^t &= Sl_3 - A_3 \cdot |C_1 \cdot Sl_3 - S| \end{aligned} \tag{6}$$

où :
$A_1$, $A_2$ et $A_3$, sont calculés à l'aide de $A = 2\alpha_1 - \alpha$ et $C_1$, $C_2$, $C_3$ sont calculés à l'aide de $C = 2r_2$.

Chaque solution de la population sera mise à jour en calculant la moyenne des trois solutions $S_1$, $S_2$ et $S_3$ comme suit (7 :

$$S_i^{t+1} = \frac{1}{3} \sum_{i=1}^{3} S_i^t \tag{7}$$

La deuxième stratégie consiste à chercher à proximité de la meilleure solution, en utilisant l'équation précédentes(5

À la fin de chaque itération, l'algorithme GGO met à jour les agents dans les groupes d'exploration et d'exploitation. À la dernière étape, GGO retourne la meilleure solution.

## Algorithme

Les principales étapes de l'algorithme GGO sont présentées dans l'Algorithme 1 :

**Algorithme 1** Pseudo code de l'algorithme Greylag Goose Optimisation (GGO)

Générer aléatoirement une solution initiale de $n$ solutions ;
Initialiser les paramètres $a$, $A$, $C$, $b$, $l$, $c$, $r_{1...5}$, $w$, $w_{1...4}$, $A_{1...3}$, $C_{1...3}$ ;
**Répéter**
  **pour** $i$ de 1 à $n_1$ **faire**
    **si** $t \mod 2 = 0$ **alors**
      **si** $r3 < 0.5$ **alors**
        **si** $|A| < 1$ ;
          Mettre à jour la position $S_i^{t+1}$ en utilisant l'Equation(1) ;
        **sinon**
          Mettre à jour $z$ en utilisant l'Equation(3) ;
          Mettre à jour la position $S_i^{t+1}$ en utilisant l'Equation(2) ;
        **fsi** ;
      **sinon**
        Mettre à jour la position $S_i^{t+1}$ en utilisant l'Equation(4) ;
      **fsi** ;
    **sinon**
      Mettre à jour la position $S_i^{t+1}$ en utilisant l'Equation(5) ;
    **fsi** ;
  **fait** ;
  **pour** $i$ de 1 à $n_2$ **faire**
    **si** $t \mod 2 = 0$ **alors**
      Calculer $S_1$, $S_2$, $S_3$ en utilisant l'Equation(6) ;
      Mettre à jour la position $S_i^{t+1}$ en utilisant l'Equation(7) ;
    **sinon**
      Mettre à jour la position $S_i^{t+1}$ en utilisant l'Equation(5 ;
    **fsi** ;
  **fait** ;
  **si** ($S^*$ est identique aux deux itérations précédentes) ;
    $n_1 = n_1 + 1$ et $n_2 = n_2 - 1$ ;
  **fsi** ;
  Mettre à jour les paramètres ;
**Jusqu'à** nombre maximum d'itérations $t$ atteint ou solution optimale trouvée
retourner la solution optimale $S^*$ ;