Investigarea unui Algoritm Metaeuristic - Gray Wolf Optimizer

Gabriel Luican

Universitatea Transilvania Brașov, Facultatea de Matematica si Informatica

Automate, calculabilitate și complexitate

Lect. dr. Luciana Majercsik

Mai 2024

Autor: Gabriel Luican

Contact: gabriel.luican@gmail.com

Capitolul I - Descrierea algoritmului

Gray Wolf Optimizer (GWO) este un algoritm metaeuristic bazat pe populație, inspirat de comportamentul social al lupilor cenușii din natură. Acesta a fost propus de Seyedali Mirjalili în 2014.

Inspirația de la lupii cenușii

Lupii cenușii sunt prădători apex, poziționați în partea de sus a lanțului trofic. Ei preferă să trăiască în grupuri (haite), fiecare grup conținând de obicei 5 până la 12 indivizi. În cadrul haitei, există o ierarhie strictă a dominanței sociale:

- Lupul Alfa (α) este individul dominant, iar alți membri ai haitei îi urmează ordinele.
- Lupii Beta (β) sunt subordonați lupului Alfa și ajută la luarea deciziilor. Ei sunt, de asemenea, potențiali candidați pentru a deveni următorul Alpha.
- Lupii Delta (δ) se supun lupilor Alpha și Beta, dar domină lupii Omega
- Lupii Omega (ω) sunt considerați țapul ispășitor în haită, și sunt cei mai puțin importanți.

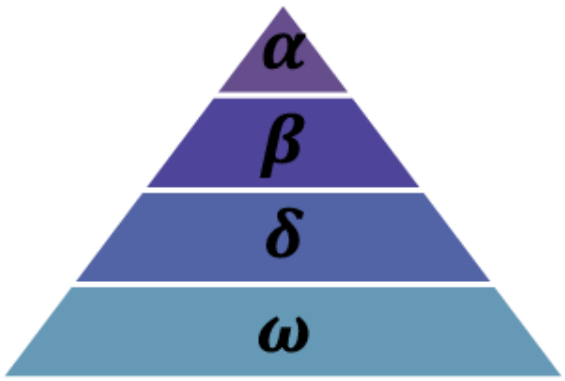

Fig. 1. Ierarhia lupilor cenușii (dominanta de sus in jos)

Prezentarea generala a algoritmului

GWO simulează comportamentul de vânătoare al lupilor cenușii pentru a găsi soluții optime pentru problemele de optimizare.

Algoritmul funcționează în trei etape principale:

- În căutarea prăzii: Lupii explorează spațiul de căutare pentru a găsi soluții potențiale.
- Încercuirea prăzii: Odată găsită o soluție promițătoare, lupii colaborează pentru a o încercui.
- Atacarea prăzii: Cea mai bună soluție este rafinată în continuare prin ajustarea poziției lupilor.

Fig. 2. (A) Explorarea, căutarea, si pândirea (B-D) Urmarirea, apropierea, și încercuirea (E) Atacarea prazii

Avantajele algoritmului

- Puțini parametri: GWO necesită foarte puțini parametri de reglare, ceea ce îl face ușor de utilizat.
- Fără derivare: Spre deosebire de unele metode de optimizare, GWO nu se bazează pe informații derivate în timpul căutării inițiale.
- Simplu, ușor de utilizat, flexibil, scalabil, și are o capacitate specială de a găsi echilibrul corect între explorare și exploatare în timpul căutării, ceea ce duce la o convergență favorabilă
- Aplicabilitate: A fost aplicat cu succes la o gamă largă de probleme de optimizare din diverse domenii precum:
    - optimizarea protocoalelor de rutare din punct de vedere energetic pentru rețele eterogene de senzori wireless
    - identificarea parametrilor efectivi pentru modelele de celule de combustibil cu membrană electrolitică polimerică
    - găsirea parametrilor optimi pentru proiectarea eficientă a palelor turbinelor eoliene sau minimizarea utilizării materialelor în construcții

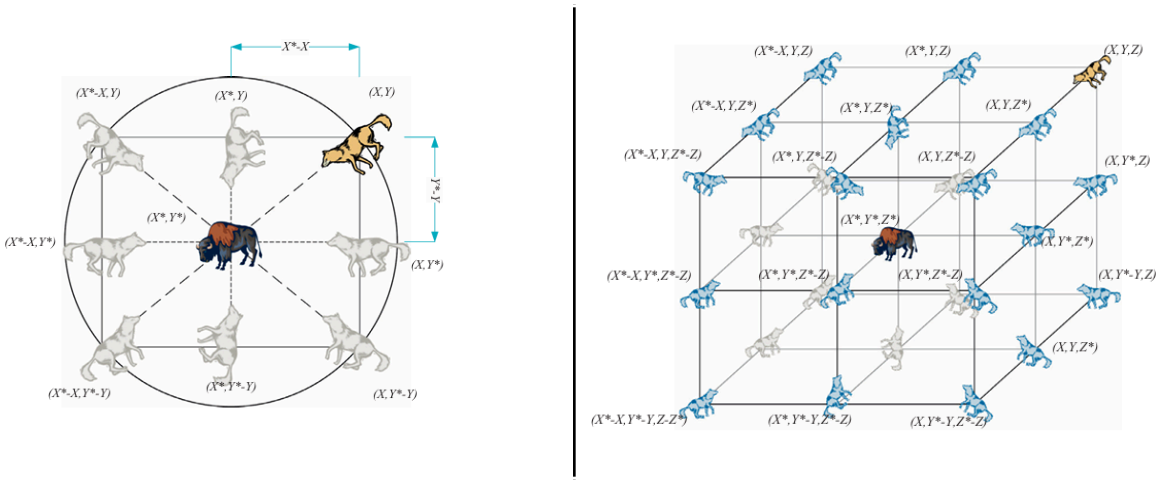

Fig. 3. Reprezentarea în spațiile 2D si 3D pentru vectorii poziție a următoarei posibile locații

Dezavantajele algoritmului

- Viteza de convergență: GWO poate converge lent, mai ales atunci când se ocupă de probleme complexe sau de optimizare dimensională înaltă. Are performanțe slabe pentru funcțiile deplasate, a căror soluție optimă este departe de zero. Procesul de căutare poate necesita un număr mare de iterații pentru a ajunge la o soluție optimă.
- Lipsa diversității: Comportamentul social al lupilor cenușii, poate duce uneori la o lipsă de diversitate în populație. Dacă populația inițială nu este suficient de diversă, algoritmul se poate bloca la o soluție optimă locală.
- Gestionarea limitată a constrângerilor: GWO nu gestionează în mod inerent bine constrângerile. Dacă problema implică constrângeri (de exemplu, constrângeri de inegalitate), sunt necesare tehnici suplimentare pentru a le pune în aplicare.

Capitolul II- Aria de aplicabilitate a algoritmului

Optimizare cu un singur obiectiv (Single-objective optimization problem):

- **Planificare și programare:** GWO a fost utilizat pentru optimizarea programelor de producție, alocarea resurselor și planificarea rutelor vehiculelor.
- **Probleme de găsire a căii Optime:** Cum ar fi problema comis-voiajorului (TSP) și problema acoperirii seturilor.
- **Alocarea sarcinilor:** În cloud computing, pentru alocarea eficientă a sarcinilor către resursele disponibile.

Optimizare Multi-obiectiv (MOP - Multi-objective Optimization Problems):

- **Optimizarea Ingineriei:** Proiectarea structurilor mecanice și optimizarea proceselor industriale.

- **Optimizare în energii regenerabile:** Optimizarea panourilor solare și a turbinele eoliene pentru a maximiza eficiența și a minimiza costurile.
- **Optimizare în bioinformatică:** Proiectarea de vaccinuri și modelarea plierii proteinelor.

Capitolul III - Exemplificarea algoritmului

Pașii algoritmului

1. Initializarea parametrilor: nr_{pop}, $iter_{max}$
2. Găsește pozițiile X_{α}, X_{β}, X_{δ}
3. Pentru $iter = 1$, $iter_{max}$
    a. Calculeaza $a = 2 \times (1 - \frac{iter}{iter_{max}})$
    b. Calculeaza X_1, X_2, X_3
    c. Calculeaza $X_{nou} = \frac{X_1 + X_2 + X_3}{3}$
    d. Verifica noua soluție
    e. Aplica selectia greedy. Daca solutia $f(X_{nou})$ este mai buna, actualizeaza solutia
    f. Seteaza $iter = iter + 1$

Exemplu pe Modified Rosenbrock Function

Initializare

$f(x, y) = -(x^2 + y^2 - 3)^2, 6 < x, y < 6, nr_{pop} = 5, iter_{max} = 5$

Gaseste X_{α}, X_{β}, X_{δ}

X_t	x = -5+RAND()*(5-(-5))	y = -5+RAND()*(5-(-5))	f(x,y) = -(x²+y²-3)²
X_{t1}	3.94	1.59	226.55
X_{t2}	3.58	-3.19	399.70
X_{t3}	0.72	-4.93	476.25
X_{t4}	-1.42	-4.44	350.81
X_{t5}	-2.80	-1.70	59.75

X_{α} = [-2.80,-1.70]

X_{β} = [3.94,1.59]

X_δ = [-1.42,-4.44]

Pentru iter = 1

$$a = 2 \times (1 - \frac{iter}{iter_{max}}) = 2 \times 1 - \frac{1}{5} = 1.6$$

X_t = [3.94,1.59]

$A_1 = 2a \times rand - a = 2 \times 1.6 \times 0.63 - 1.6 = 0.41$

$C_1 = 2\,rand = 2 \times 0.82 = 1.64$

$D_\alpha = |C_1 X_\alpha - X(t)| = |1.64 \times [-2.80, -1.70] - [3.94, 1.59]| = [-8.42, -4.31]$

$X_1 = X_\alpha - A_1 D_\alpha = [-2.80, -1.70] - 0.41 \times [-8.42, -4.31] = [0.65, 0.06]$

$A_2 = 2a \times rand - a = 2 \times 1.6 \times 0.15 - 1.6 = 1.12$

$C_2 = 2\,rand = 2 \times 0.93 = 1.86$

$D_\beta = |C_2 X_\beta - X(t)| = |1.84 \times [3.94, 1.59] - [3.94, 1.59]| = [3.30, 1.33]$

$X_2 = X_\beta - A_2 D_\beta = [3.94, 1.59] - 1.12 \times [3.30, 1.33] = [0.24, 0.10]$

$A_3 = 2a \times rand - a = 2 \times 1.6 \times 0.23 - 1.6 = -0.86$

$C_3 = 2\,rand = 2 \times 0.68 = 1.36$

$D_\delta = |C_3 X_\delta - X(t)| = |1.36 \times [-1.42, -4.44] - [3.94, 1.59]| = [-5.87, -7.62]$

$X_3 = X_\delta - A_3 D_\delta = [-1.42, -4.44] - (-0.86) \times [-5.87, -7.62] = [-6.49, -11.03]$

Calculeaza X_{nou}

$X_{nou} = \frac{X_1 + X_2 + X_3}{3} = [-1.86, -3.62]$,

$x_{nou} = \frac{0.65 + 0.24 + (-6.49)}{3} = -1.86$

$$y_{nou} = \frac{0.06 + 0.10 + (-11.03)}{3} = -3.62$$

Verificăm noua soluție

Aplicam selectia greedy. Daca solutia $f(X_{nou})$ este mai buna, actualizeaza solutia

$$f(x_{nou}, y_{nou}) = 183.98 < f(x_{t1}, y_{t1}) = 226.55$$

Actualizam solutia

X_t	x = -5+RAND()*(5-(-5))	y = -5+RAND()*(5-(-5))	f(x,y) = -(x²+y²-3)²
~~X_{t1}~~	~~3.94~~	~~1.59~~	~~226.55~~
X_{nou}	-1.86	-3.62	183.98
X_{t2}	3.58	-3.19	399.70
X_{t3}	0.72	-4.93	476.25
X_{t4}	-1.42	-4.44	350.81
X_{t5}	-2.80	-1.70	59.75

Bibliografie

[1] Bonabeau E, Dorigo M, Theraulaz G. Swarm intelligence: from natural to artificial systems: OUP USA; 1999.

[2] Dorigo M, Birattari M, Stutzle T. Ant colony optimization. Comput Intell Magaz, IEEE 2006;1:28–39.

[3] Kennedy J, Eberhart R. Particle swarm optimization, in Neural Networks, 1995. In: Proceedings, IEEE international conference on; 1995. p. 1942–1948.

[4] Wolpert DH, Macready WG. No free lunch theorems for optimization. Evolut Comput, IEEE Trans 1997;1:67–82.

[5] Kirkpatrick S, Jr. DG, Vecchi MP. Optimization by simulated annealing. Science, vol. 220; 1983. p. 671–80.

[6] Beni G, Wang J. Swarm intelligence in cellular robotic systems. In: Robots and biological systems: towards a new bionics?, ed. Springer; 1993. p. 703–12.

[7] Basturk B, Karaboga D. An artificial bee colony (ABC) algorithm for numeric function optimization. In: IEEE swarm intelligence symposium; 2006. p. 12–4.

[8] Olorunda O, Engelbrecht AP. Measuring exploration/exploitation in particle swarms using swarm diversity. In: Evolutionary computation, 2008. CEC 2008 (IEEE World Congress on Computational Intelligence). IEEE Congress on; 2008. p. 1128–34.

[9] Alba E, Dorronsoro B. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. Evolut Comput, IEEE Trans 2005;9:126–42.

[10] Lin L, Gen M. Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation. Soft Comput 2009;13:157–68.

[11] Mirjalili S, Hashim SZM. A new hybrid PSOGSA algorithm for function optimization. In: Computer and information application (ICCIA), 2010 international conference on; 2010. p. 374–77.

tnt3.ir/wp-content/uploads/j.advengsoft.2013.12.007_578192a5ab0b6fce941a37cf0c5955c6.pdf

Grey Wolf Optimizer - ScienceDirect