Chicken Swarm Optimization

Un algoritmo bio-inspirado

Jaime Francisco Aguayo González
Dpto. de Matemáticas UG

June 8, 2018

CSO algorithm

Introducción

Sobre biología

Modelación matemática

Construcción del algoritmo

Implementación y resultados

Introducción

Dentro del área de optimización existe una familia de algoritmos denominados algoritmos bio-inspirados de optimización. Estos algoritmos se distinguen por ser algoritmos iterativos cuyo funcionamiento se inspira en comportamientos de la naturaleza. Algunos ejemplos son:

- Ant colony optimization
- Bacterial foraging
- Particle swarm
- Cuckoo search
- Bat algorithm


Pollos

Inspiración

Los pollos son una de las especies más abundantes en el planeta.

Algunas investigaciones sugieren que los gallos y gallinas pueden apreciar cómo perciben el mundo sus similares y pueden utilizar esa información en su propio beneficio.

Si un pollo macho está buscando comida y localiza un bocado suculento, usualmente tratará de impresionar a las hembras mientras hace un característico llamado para anunciar su hallazgo.

Éste simpático hecho se replica con las gallinas cuando cuidan a sus hijos.

Inspiración

Éste compañerismo no se comparte entre pollos pertenecientes a grupos distintos.

Los gallos producen un cacareo distintivo cuando pollos de un grupo distinto invade su territorio.

El gallo alfa tiene todo el acceso para buscar comida y pelear con otros gallos que invadan el territorio donde habita su grupo social.

Inspiración

En general, el comportamiento social de los pollos varía con el género.

El gallo alfa tiene todo el acceso para buscar comida.

Los pollos varones no tan dominantes, estarán cerca del gallo alfa para urgar y encontrar comida.

Los pollos sumisos, en cambio, son relegados a la periferia.

Por último los pollitos bebés buscan su comida alrededor de su mamá.

Inspiración

Esta inteligencia colectiva con la cual los pollos buscan comida, inspiró a los autores a crear el CSO ya que la búsqueda de comida puede ser fácilmente asociado al problema de optimizar una función.

Formulación

La sociedad de los pollos se divide en varios grupos. Cada grupo es compuesto por un gallo dominante, varias gallinas, y varios pollitos.

Formulación

La división de los pollos en grupos y el rol que juega cada uno dependerá en los valores de los valores de salud física de los pollos en ellos mismos. El pollo con el mejor valor en salud física será clasificado como gallo. Los pollos con un peor desempeño, será asignado como un pollito. Todas las demás serán gallinas. El grupo en el cual las gallinas van a vivir, es escogido de manera aleatoria. La relación de madre e hijo igualmente es asociada de manera aleatoria.

Formulación

El orden gerárquico, relación de dominación-sumisión y madre-hijo dentro del grupo permanecerá inalterable. El estatus que cada ave tiene dentro del grupo sólo se alterará después de ***G*** pasos, un número que se espera sea grande.

Formulación

Todos los pollos siguen al gallo líder de su grupo en la búsqueda de comida, al mismo tiempo pueden prevenir que los demás no se coman su comída. Asumimos además que de manera aleatoria, los pollos robarán la comida que otros hayan encontrado. Los pollitos buscan comida cerca de donde está su madre gallina. Los individuos dominantes tienen ventaja en la búsqueda de comida.

Formulación matemática

Denotamos con RN, HN, CN y MN al número de gallos, gallinas, pollitos y las gallinas madre respectivamente. $RN + HN + CN = N$

La posición de cada pollo al tiempo $t \geq 0$ lo denotamos por $x_{i,j}^t$ con $i \in [1, ..., N]$, $j \in [1, ..., D]$.

Los mejores pollos de los RN corresponden a los de menor valor de salud física, es decir, menor valor en f.

Construcción del algoritmo: Gallos

$$x_{i,j}^{t+1} = x_{i,j}^{t} * (1 + Randn(0, \sigma^2)) \tag{1}$$

$$\sigma^2 = \begin{cases} 1, \text{ si } f_i \leq f_k \\ \exp \frac{(f_k - f_i)}{|f_i| + \varepsilon}, \text{ en otro caso} \end{cases} \qquad k \in [1, N], k \neq i \tag{2}$$

Construcción del algoritmo: Gallos

Randn$(0, \sigma^2)$ es una muestra Gaussiana con media 0 y varianza σ^2.

ε es un valor pequeño para evitar errores por división entre cero.

El índice del gallo, k, es seleccionado aleatoriamente del grupo de gallos y f_k es la puntuación del gallo k, es decir, $f(x_k^t)$, igualmente $f_i = f(x_i^t)$.

Construcción del algoritmo: Gallinas

$$x_{i,j}^{t+1} = x_{i,j}^{t} + S1 * Rand(x_{r1,j}^{t} - x_{i,j}^{t}) + S2 * Rand(x_{r2,j}^{t} - x_{i,j}^{t}) \quad (3)$$

$$S1 = \exp\left(\frac{(f_i - f_{r1})}{abs(f_i) + \varepsilon}\right) \quad (4)$$

$$S2 = \exp(f_{r2} - f_i) \quad (5)$$

Construcción del algoritmo: Gallinas

Rand es un número tomado de una muestra uniforme en $[0, 1]$.

$r1 \in [1, .., N]$ es el índice del gallo líder del grupo donde la gallina ***i*** se encuentra, mientras que $r2 \in [1, .., N]$ es el índice de un pollo elegido aleatoriamente del total de pollos pero distinto del gallo líder (pues hay que respetar el orden jerárquico).

Construcción del algoritmo: Pollitos

$$x_{i,j}^{t+1} = x_{i,j}^{t} + FL * (x_{m,j}^{t} - x_{i,j}^{t}) \tag{6}$$

FL (FL $\in (02)$) es un parámetro que determina que tanto siguen los pollitos a su madre. LF es seleccionado uniformemente en $(0, 2)$.

Chicken Swarm Optimization. Framework of the CSO

Initialize a population of N chickens and define the related parameters;
Evaluate the N chickens' fitness values, t=0;
While (t < Max_Generation)
  If (t % G == 0)
    Rank the chickens' fitness values and establish a hierarchal order in the swarm;
    Divide the swarm into different groups, and determine the relationship between the chicks and mother hens in a group; End if
  For i = 1 : N
    If i == rooster Update its solution/location using equation (1); End if
    If i == hen Update its solution/location using equation (3); End if
      If i == chick Update its solution/location using equation (6); End if
    Evaluate the new solution;
    If the new solution is better than its previous one, update it;
  End for
End while

Construcción del algoritmo: Parámetros

En el artículo se propone usar los valores $G \in [2, 20]$ y $FL \in [0.4, 1]$.

Después de jugar un rato con varias funciones, los valores que mejor me funcionaron fueron $G \in [2, 16]$ y $FL \in [0.3, 1)$, y en lo particular para la función de Rosenbrock y la función de Wood, funcionaron mejor valores pequeños de FL y valores de G entre 5 y 12.

RN = ceil(0.15 * N);
HN = ceil(0.7 * N);
MN = ceil(0.2 * N);

Implementación

Para realizar las pruebas se implementó el algoritmo en c++ usando las funciones de métodos numéricos del curso anterior, y las funciones de la GSL.

También se escribió una versión en python para hacer algunas animaciones.

Implementación: Función de Rosenbrock

$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$$

Implementación: Función de Rosenbrock n=2

Para este problema se usaron 18 pollos, $FL = 0.5$ y $G = 5$. Los límites de busqueda fueron $-1 \leq x_i \leq 2$. Para la búsqueda en linea se usó el punto inicial $(1.2, -1.2)$.

Método	N	$\| \nabla f(x*) \|$
CSO	6,000	1.47e-05
Backtracking	3,000	0.169
Interpolación	3,000	0.015

DEMO

Implementación: Función de Rosenbrock n=100

Para este problema se usaron 18 pollos, $FL = 0.5$ y $G = 5$. Los límites de busqueda fueron $-1 \leq x_i \leq 2$. Para la búsqueda en linea se usó el punto inicial $(1.2, -1.2, \ldots, 1.2, -1.2)$.

Método	N	$\| \nabla f(x*) \|$
CSO	10,000	18.1641
Backtracking	3,000	0.045
Interpolación	3,000	0.125

Implementación: Función de Rosenbrock n=100

En este caso, al aumentar el número de iteraciones al doble, apenas y se ve un progreso. Además, aumentar el número de pollos a más de 24 empeora el resultado, al igual que usando menos de 10 agentes.

Al ampliar los límites de búsqueda, la precisión no cambia en ambos casos, sin embargo para límites mayores a -10 y 10, es necesario aumentar el número de pollos para obtener un resultado similar.

Implementación: Función de Wood

Para este problema se usaron 18 pollos, $FL = 0.4$ y $G = 5$. Los límites de busqueda fueron $-2 \leq x_i \leq 2$. Para la búsqueda en linea se usó el punto inicial $(-3, -1, -3, -1)$.

Método	N	$\| \nabla f(x*) \|$
CSO	10,000	1.33
Backtracking	3,000	7.78e−03
Interpolación	3,000	1.67e−03

Implementación: Función de Wood

Una observación: En este caso, al aumentar el número de iteraciones al doble, se logra dar con el mínimo con $\| \nabla f(x*) \| = 0.134$

Ventajas

Es muy rápido con pocos pollos y se nota sobre todo si la dimensión es grande.

El artículo menciona que es mejor que otros métodos bio-inspirados como PSO, DE y BA

No es necesario calcular el gradiente, ni dar un punto inicial.

Es difícil que se quede en un óptimo local.

Desventajas

No es muy preciso en el resultado.

Cuando el número de pollos incrementa, el algoritmo se vuelve lento.

Tal vez este algoritmo podría usarse bien para encontrar un punto inicial y usar algún otro método para encontrar con precisión el óptimo.

Artículo

Meng X., Liu Y., Gao X., Zhang H. (2014)
A New Bio-inspired Algorithm: Chicken Swarm Optimization.
In: Tan Y., Shi Y., Coello C.A.C. (eds) Advances in Swarm Intelligence. ICSI 2014. Lecture Notes in Computer Science, vol 8794. Springer, Cham