Chicken Swarm Optimization

Jaime Francisco Aguayo González[1]

June 8, 2018

[1]Dpto. de Matemáticas, Universidad de Guanajuato.

Resumen: Este proyecto consiste en una breve presentación del método bio-inspirado de optimización denominado Chicken Swarm Optimization presentado en el artículo [1]. Se presentarán algunos posibles usos y se verán algunos ejemplos ad hoc con los vistos en clase para evaluar el algoritmo.

1. 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

Estos algoritmos suelen ser ligeros al ejecutarse y dar una solución aceptable en muchas funciones cuya derivada no existe o es muy difícil de calcular. En el artículo [1], describen el algoritmo *Chicken Swarm Optimization* (CSO) que es un algoritmo inspirado en el comportamiento social de los pollos al buscar su comida.

En el presente trabajo se describirá la biología existente subyacente, después se hará una formulación matemática que modela dicho comportamiento bajo ciertas suposiciones, al final daremos una implementación del algoritmo en varios ejemplos, comparando el funcionamiento del método.

2. Biología del algoritmo

En el mundo hay más de 19.000 millones de pollos, lo que los convierte en una de las especies más abundantes en el planeta.

Algunas investigaciones sugieren que los pollos 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 realizando una danza particular mientras hace un característico llamado para

anunciar su hallazgo.

No obstante, cuando los machos subordinados realizan esta danza y canto, corren el riesgo de ser detectados y atacados por el macho dominante. De esta forma, si el macho dominante se encuentra cerca, el subordinado suele realizar su danza en silencio, en un intento por impresionar a las hembras sin que el dominante lo note. También hay machos que tratan de engañar a las hembras haciendo este canto, aunque en realidad no hayan encontrado nada. De más está decir: frecuentemente las hembras aprenden rápidamente quiénes son estos machos.

Es justo el orden jeréarquico social el que juega un papel fundamental en la vida de los pollos. Los individuos dominantes de la sociedad tienen prioridad en acceder a comida, por ejemplo los gallos dominantes tendrán más facilidad para encontrar comida y en ellos recae el poder de ceder su lugar de comer primero a sus compañeros, compartiendo la comida que encontró. Este simpático hecho se replica con las gallinas cuando cuidan a sus hijos.

Éste compañerismo no se comparte entre pollos pertenecientes a grupos distintos. De hecho los gallos producen un cacareo distintivo cuando pollos de un grupo distinto invaden su territorio.

En general, el comportamiento social de los pollos varía con el género. El gallo alfa tiene todo el acceso para buscar comida y pelear con otros gallos que invaden el territorio donde habita su grupo social. Los pollos varones no tan dominantes, estarán cerca del gallo alfa para urgar y encontrar comida. Los pollos débiles, en cambio, son relegados a la periferia donde buscan su comida, existiendo en todo momento una competencia en la búsqueda de comida. Por último los pollitos bebés buscan su comida alrededor de su mamá.

Esta inteligencia colectiva con la cual los pollos buscan comida, inspiró a los autores a crear el CSO ya que la busqueda de comida puede ser fácilmente asociado a encontrar la solución de un problema de optimización.

2. Construcción del algoritmo

Es momento ahora de describir matemáticamente el algoritmo. En el documento se manejan 4 idealizaciones del comportamiento de los pollos.

1. La sociedad de los pollos se divide en varios grupos. Cada grupo es compuesto por un gallo dominante, una pareja de gallinas, y varios pollitos.
2. 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.
3. 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.
4. 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.

A lo largo del presente trabajo denotamos con RN, HN, CN y MN al número de gallos, gallinas, pollitos y las gallinas madre respectivamente, y N el número total de pollos y a la posición de cada pollo al tiempo $t \geq 0$ lo denotamos por $x_{i,j}^t$ con $i \in [1, ..., N]$, $j \in [1, ..., D]$ donde D es la dimensión del espacio donde se busca comida. Además, considerando los problemas de optimización como problemas de minimización, los mejores pollos de los RN corresponden a los de menor valor de salud física.

3.1 Movimiento de los Pollos

Como se mencionó, los gallos con mejores puntuaciones tendrán prioridad en acceder a la comida que los demás, esto puede ser modelado premitiendoles buscar en un rango más amplio de lugares que los gallos con peores putuajes. Matematicamente esto puede ser formulado como:

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

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

Aquí $\mathit{Randn}(0, \sigma^2)$ es una muestra Gaussiana con media 0 y varianza σ^2, mientras que 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, $f(x_k^t)$, lo mismo f_i con el gallo i.

El movimiento de las gallinas, como bien se dijo, es seguir a los gallos de su grupo en la búsqueda de comida y además, tendrán la labor de robar comida de encontrada por otros pollos si es que éstos no las rechazan. Nuevamente recordamos que las gallinas dominantes tendrán ventaja en la búsqueda de comida. Dado lo anterior, se formula lo siguiente.

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

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

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

Donde *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), $r1 = r2$.

Notemos que como $f_i > f_{r1}$ y $f_i > f_{r2}$ por lo que $S2 < 1 < S1$. Si $S1 = 0$, entonces la gallina estará robando comida de el pollo $r2$. Entre mayor sea la diferencia en los valores de los pollos en 5, menor será $S2$ y por lo tanto la brecha de jerarquía entre los dos será grande lo que impide que sea fácil robar la comida de otros pollos. Si S2 es chico, digamos 0, entonces la gallina estará buscando comida en su propio espacio. Si el valor de la i-ésima gallina es chico, entonces $S1$ se aproxima a 1 y la brecha social entre la gallina el gallo líder de su grupo es corta, lo que muestra la jerarquía de las gallinas al acceder a la comida.

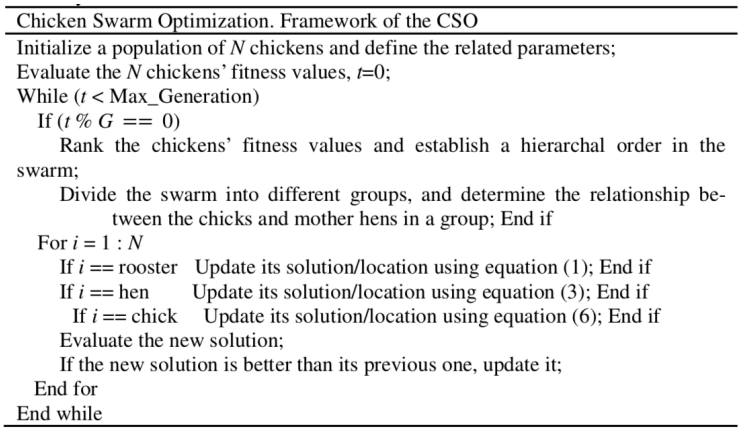
Finalmente para el caso de los pollitos, éstos se mueven alrededor de su madre para buscar comida. Denotemos por $x_{m,j}^t$ a la posición de la madre del pollo i. Luego:

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

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

3.2 Algoritmo

Dada la fomulación anterior, el algoritmo del Chicken Swarm Optimization (CSO) es el siguiente:


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

Figure 1: Algoritmo CSO completo.

4. 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. Éstas últimas fueron de mucha ayuda, sobre todo las funciones de generación de números aleatorios y su librería de estadística.

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 Rosembrock y la función de Wood, funcionaron mejor valores pequeños de *FL* y valores de *G* entre 5 y 12.

También el artículo sugiere que el número de gallos sea pequeño y que las gallinas sean mayoritarias, ya que es así como se presenta en la naturaleza. En la implementación que se realizó, se usaron los valores:
RN = ceil(0.15 * N)
HN = ceil(0.7 * N)
CN = N - RN - HN
MN = ceil(0.2 * N)

En la carpeta del presente proyecto se adjunta el código correspondiente. A continuación haré una breve comparativa entre el método CSO y los métodos de búsqueda en linea de backtracking e interpolación cuadrática.

4.1 Optimización de la función de Rosenbrock

Vamos a optimizar la función de Rosembrock de orden n definida como

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

El óptimo es $x^* = (1, 1, ..., 1)$.

Para optimizar, se usaron 18 pollos y los parámetros $FL = 0.5$ y $G = 5$. Los límites de busqueda fueron $-1 \leq x_i \leq 2$. Además, para comparar el método se solucionó usando descenso del gradiente con busqueda en linea backtracking e interpolación cuadrática. Para la búsqueda en linea se usó el punto inicial $(1.2, -1.2, ..., 1.2, -1.2)$.

n = 2

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

n = 100

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

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.

También es de observarse que al algoritmo le cuesta trabajo ubicar el mínimo para problemas de alta dimensionalidad.

4.2 Optimización de la 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 el descenso del gradiente se usó el punto inicial $(-3, -1, -3, -1)$. Los resultados se presentan a continuación:

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

En este caso, al aumentar el número de iteraciones al doble, se logra dar con el mínimo con $\| \nabla f(x*) \| = 0.134$. A pesar de que el número de iteraciones es elevado, el algoritmo es bastante rápido que no se nota una diferencia comparado con el método de búsqueda en línea.

4.3 Optimización de una función con múltiples mínimos locales

Una posible aplicación de estos métodos bio-inspirados es obtener una solución en problemas cuya función objetivo tiene muchos mínimos locales, o funciones no diferenciables. En este caso se usará la función Drop-Wave de dimensión 2 definida como

$$f(x) = \frac{1 + \cos(12\sqrt{x_1^2 + x_2^2})}{0.5(x_1^2 + x_2^2) + 2}$$

La función tiene un mínimo local $x^* = (0, 0)$ y. varios mínimos locales que se pueden observar en la figura 2, donde además se grafican los puntos para la iteración 32.

Es de notarse que el algoritmo no solo dio con la solución exacta, sino que lo hizo en 31 iteraciones, tomando solo 18 pollos y un intervalo $-5 \leq x_1, x_2 \leq 5$.

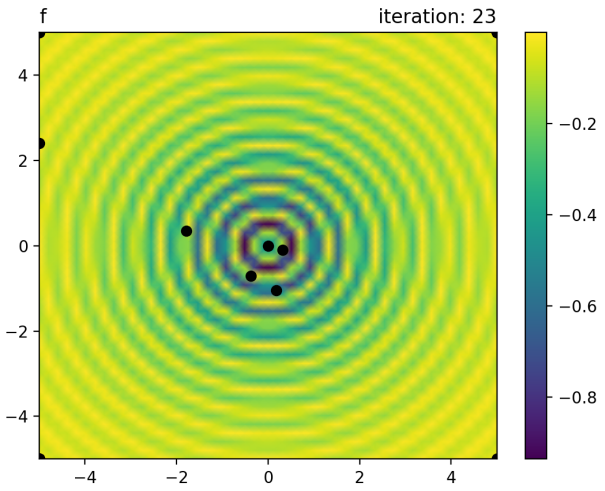


Figure 2: Curvas de nivel y posiciones de los pollos.

5. Conclusiones

A manera de conclusión me gustaría mencionar los aspectos positivos y negativos del algoritmo.

Ventajas del algoritmo:

- 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 por lo que es útil para funciones con multiples óptimos locales.

Desventajas del algoritmo:

- No es muy preciso en el resultado.
- Cuando el número de pollos incrementa, el algoritmo se vuelve lento.

Me gustaría resaltar que este algoritmo podría usarse bien para encontrar un punto inicial y usar algún otro método para encontrar con precisión el óptimo, de esta forma podemos optimizar funciones de las cuales, a priori, no se tiene información sobre dónde localizar el punto óptimo.

Dada su facilidad de implementación y del poco consumo de recursos necesarios, este algoritmo puede ser implementado en sistemas embebidos que requieran optimizar funciones, por ejemplo redes neuronales artificiales.

Referencias

1. 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 (*Book)*

2. *http://www.bbc.com/mundo/vert-earth-38659051 (website)*