

CAT SWARM OPTIMIZATION && CUCKOO SWARM OPTIMIZATION

Presented by: Adolfo Mora Cordova

OVERVIEW

- CSO
 - Origen Biológico
 - Estrategia
 - Algoritmo
-
- C_SO
 - Origen Biológico
 - Estrategia
 - Algoritmo

CSO: ALGORITMO

Cat swarm optimization

CSO ORIGEN BIOLÓGICO

El algoritmo se inspira directamente en el comportamiento observable de los gatos domésticos. A pesar de ser depredadores, los gatos pasan la gran mayoría de su tiempo (a menudo >70%) en un estado de aparente descanso. Sin embargo, este "descanso" es un estado de alerta.

CSO ESTRATEGIA

Modo de Búsqueda (Seeking Mode)

Un porcentaje de los "gatos" (soluciones) se dedica a explorar minuciosamente el área justo a su alrededor. Crean varias "copias" de sí mismos con pequeñas variaciones y ven si alguna es mejor. Esto permite al algoritmo refinar una buena solución y explorar colinas pequeñas en el paisaje de fitness.

Modo de Rastreo (Tracing Mode):

- El resto de los gatos ignora su entorno local y se lanza directamente hacia la mejor solución encontrada hasta ahora por todo el enjambre (G). Esto asegura que el conocimiento global se comparta y que las buenas soluciones se exploten rápidamente (convergencia).

MODO DE BÚSQUEDA (SEEKING MODE)

SMP se utiliza para definir el tamaño de la memoria de búsqueda de cada gato, lo que indica los puntos buscados por el gato. El gato elegirá un punto del grupo de memoria de acuerdo con las reglas descritas más adelante.

SRD declara la proporción de mutación para las dimensiones seleccionadas. En el modo de búsqueda, si se selecciona una dimensión para mutar, la diferencia entre el nuevo valor y el antiguo no superará el rango definido por SRD.

CDC revela cuántas dimensiones se modificarán. Todos estos factores juegan un papel importante en el modo de búsqueda.

SPC es una variable booleana que determina si el punto en el que el gato ya se encuentra será uno de los candidatos a moverse. No importa el valor de SPC

MODO DE BÚSQUEDA (SEEKING MODE)

Paso 1: Haz j copias de la posición actual de cat_k, donde j = SMP. Si el valor de SPC es verdadero, deja que j = (SMP-1), luego conserva la posición actual como uno de los candidatos.

Paso 2: Para cada copia, de acuerdo con CDC, suma o resta aleatoriamente SRD por ciento de los valores actuales y reemplaza los antiguos.

Paso 3: Calcula los valores de aptitud (FS) de todos los puntos candidatos.

Paso 4: Si todos los FS no son exactamente iguales, calcula la probabilidad de selección de cada punto candidato mediante la ecuación (1), de lo contrario, establece la probabilidad de selección de cada punto candidato en 1.

Paso 5: Elige aleatoriamente el punto al que moverse entre los puntos candidatos y reemplaza la posición de cat_k.

$$P_i = \frac{|FS_i - FS_b|}{FS_{\max} - FS_{\min}}, \text{ where } 0 < i < j \quad 1$$

MODO DE BÚSQUEDA (SEEKING MODE)

Paso 1: Actualizar las velocidades para cada dimensión ($v_{k,d}$) de acuerdo con la ecuación (2).

Paso 2: Verificar si las velocidades están dentro del rango de la velocidad máxima. En caso de que la nueva velocidad exceda el rango, establecerla igual al límite.

Paso 3: Actualizar la posición de catk de acuerdo con la ecuación (3)

$$v_{k,d} = v_{k,d} + r_1 \times c_1 \times (x_{best,d} - x_{k,d}) , \text{ where } d = 1,2,\dots,M$$

2

$$x_{k,d} = x_{k,d} + v_{k,d}$$

3

$x_{best,d}$ es la posición del gato que tiene el mejor valor de aptitud; $x_{k,d}$ es la posición de cat_k. c_1 es una constante y r_1 es un valor aleatorio en el rango de [0,1].

CSO: ALGORITMO

Paso 1: Crear N gatos en el proceso.

Paso 2: Espaciar aleatoriamente los gatos en el espacio de soluciones M-dimensional y seleccionar aleatoriamente valores, que estén dentro del rango de la velocidad máxima, para las velocidades de cada gato. Luego, elegir de manera casual un número de gatos y ponerlos en modo rastreo según MR, y los demás en modo búsqueda.

Paso 3: Evaluar el valor de aptitud de cada gato aplicando las posiciones de los gatos a la función de aptitud, que representa los criterios de nuestro objetivo, y mantener el mejor gato en la memoria. Nota que solo necesitamos recordar la posición del mejor gato (x_{best}) ya que representa la mejor solución hasta el momento.

CSO: ALGORITMO

Paso 4: Mover los gatos según sus banderas; si cat_k está en modo búsqueda, aplicar el gato al proceso de modo búsqueda, de lo contrario aplicarlo al proceso de modo rastreo. Los pasos del proceso se presentan arriba.

Paso 5: Volver a elegir un número de gatos y ponerlos en modo rastreo según MR, luego poner a los otros gatos en modo búsqueda.

Paso 6: Verificar la condición de terminación; si se cumple, terminar el programa, de lo contrario repetir los pasos 3 a 5.

C_SO: ALGORITMO

Cuckoo swarm optimization

C_SO

ORIGEN BIOLÓGICO

Este algoritmo se inspira en el "parasitismo de cría" de algunas especies de pájaros cuco (cucú).

- Parasitismo de Cría: El cuco hembra pone sus huevos en los nidos de otras especies de pájaros (los "anfitriones").
- Vuelos de Lévy (Lévy Flights):una serie de pasos cortos y locales (buscando en un área) interrumpidos por saltos largos y aleatorios (moviéndose a un área completamente nueva).
- Descubrimiento: El pájaro anfitrión tiene una cierta probabilidad (pa) de descubrir que el huevo no es suyo. Si lo hace, tiene dos opciones: tirar el huevo falso o (más drásticamente) abandonar el nido por completo y construir uno nuevo en otro lugar.

C_{SO} ESTRATEGIA

Exploración Global (Vuelos de Lévy):

La principal "fuerza" del algoritmo. Se genera un "cupo" (una nueva solución candidata) tomando un nido existente y haciéndolo dar un "Vuelo de Lévy". Gracias a los saltos largos, este mecanismo permite al algoritmo escapar de mínimos locales y explorar de forma muy eficiente regiones lejanas del espacio de búsqueda. Luego, esta nueva solución compite con el nido existente (selección "greedy").

Eliminación de Malas Soluciones (Abandono):

Una fracción (ρ) de los peores nidos (las peores soluciones) son "descubiertos" por el anfitrión. Estos nidos se descartan por completo y se reemplazan por nuevos nidos generados aleatoriamente en otras partes del mapa. Esto evita que el algoritmo pierda tiempo tratando de mejorar malas soluciones y promueve la diversidad.

LÉVY FLIGHT IS PERFORMED

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \alpha \oplus \text{Lévy}(\lambda),$$

$$\text{Lévy} \sim u = t^{-\lambda}, \quad (1 < \lambda \leq 3),$$

Donde $\alpha > 0$ es el tamaño del paso, que debe estar relacionado con las escalas del problema de interés.

El producto \oplus significa multiplicaciones elemento por elemento. Este producto elemento por elemento es similar a los utilizados en PSO

C_SO: ALGORITMO

1. Inicialización: Se crea una población X de pop_size nidos en posiciones aleatorias y se evalúa su fitness F . Se identifica el mejor nido global (G y G_f).
2. Bucle de Iteraciones:
3. Generar Nuevos Cucos (Vuelos de Lévy):
 - Para cada nido i en la población, se genera una nueva solución candidata X_{new} usando un Vuelo de Lévy: $X_{\text{new}} = X[i] + \text{step_size}$.
 - El step_size se calcula usando la fórmula de Lévy (que combina dos distribuciones normales) y a menudo se escala (en tu código, $\text{alpha} * (X[i] - G)$).
 - Se evalúa el fitness de la nueva candidata: $F_{\text{new}} = \text{objective}(X_{\text{new}})$.
 - Selección Greedy: Se compara el nido nuevo con el viejo. Si F_{new} es mejor que $F[i]$, el nido i se actualiza: $X[i] = X_{\text{new}}$, $F[i] = F_{\text{new}}$.

Cuckoo Search via Lévy Flights

```
begin
    Objective function  $f(\mathbf{x})$ ,  $\mathbf{x} = (x_1, \dots, x_d)^T$ 
    Generate initial population of
         $n$  host nests  $\mathbf{x}_i$  ( $i = 1, 2, \dots, n$ )
    while ( $t < \text{MaxGeneration}$ ) or (stop criterion)
        Get a cuckoo randomly by Lévy flights
        evaluate its quality/fitness  $F_i$ 
        Choose a nest among  $n$  (say,  $j$ ) randomly
        if ( $F_i > F_j$ ),
            replace  $j$  by the new solution;
        end
        A fraction ( $p_a$ ) of worse nests
            are abandoned and new ones are built;
        Keep the best solutions
            (or nests with quality solutions);
        Rank the solutions and find the current best
    end while
    Postprocess results and visualization
end
```

Figure 1: Pseudo code of the Cuckoo Search (CS).

C_SO: ALGORITMO

4. Abandono de Nidos (Descubrimiento):

- Se identifica un conjunto de nidos a abandonar (los que `rng.random() < pa`).
- Para cada nido abandonado, se genera una nueva solución $X_{\text{new_abandon}}$. Esto se hace comúnmente mediante una recombinación aleatoria de otros dos nidos (ej: $X_{\text{d1}} + \text{rand} * (X_{\text{d2}} - X_{\text{d3}})$).
- Se evalúa $F_{\text{new_abandon}}$.
- Selección Greedy (de nuevo): El nido abandonado solo se reemplaza si la nueva solución $X_{\text{new_abandon}}$ es mejor que la solución actual del nido que iba a ser abandonado ($F_{\text{new_abandon}} < F[\text{índice_abandonado}]$). Nota: Esta es una implementación; otra más simple es simplemente reemplazarlo sin importar qué.

Cuckoo Search via Lévy Flights

begin

 Objective function $f(\mathbf{x})$, $\mathbf{x} = (x_1, \dots, x_d)^T$

 Generate initial population of

n host nests \mathbf{x}_i ($i = 1, 2, \dots, n$)

while ($t < \text{MaxGeneration}$) or (stop criterion)

 Get a cuckoo randomly by Lévy flights

 evaluate its quality/fitness F_i

 Choose a nest among n (say, j) randomly

if ($F_i > F_j$),

 replace j by the new solution;

end

 A fraction (p_a) of worse nests

 are abandoned and new ones are built;

 Keep the best solutions

 (or nests with quality solutions);

 Rank the solutions and find the current best

end while

 Postprocess results and visualization

end

Figure 1: Pseudo code of the Cuckoo Search (CS).

C_SO: ALGORITMO

5. Actualización Global: Después de ambas fases, se vuelve a buscar en toda la población X para encontrar el mejor nido global G actualizado.
6. Repetir hasta max_iters.

Cuckoo Search via Lévy Flights

```
begin
    Objective function  $f(\mathbf{x})$ ,  $\mathbf{x} = (x_1, \dots, x_d)^T$ 
    Generate initial population of
         $n$  host nests  $\mathbf{x}_i$  ( $i = 1, 2, \dots, n$ )
    while ( $t < \text{MaxGeneration}$ ) or (stop criterion)
        Get a cuckoo randomly by Lévy flights
            evaluate its quality/fitness  $F_i$ 
            Choose a nest among  $n$  (say,  $j$ ) randomly
            if ( $F_i > F_j$ ),
                replace  $j$  by the new solution;
        end
        A fraction ( $p_a$ ) of worse nests
            are abandoned and new ones are built;
        Keep the best solutions
            (or nests with quality solutions);
        Rank the solutions and find the current best
    end while
    Postprocess results and visualization
end
```

Figure 1: Pseudo code of the Cuckoo Search (CS).

REFERENCE

Cat Swarm Optimization (CSO)

Chu, S. C., & Tsai, P. W. (2007). Cat Swarm Optimization. En Parallel and Distributed Computing, Applications and Technologies (PDCAT 2007) (pp. 854-858). Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-540-77090-8_102

Cuckoo Search (CS)

Yang, X. S., & Deb, S. (2009, Diciembre). Cuckoo search via Lévy flights. En 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC) (pp. 210-214). IEEE. DOI: 10.1109/NABIC.2009.5393690

THANK YOU