

Cómputo Paralelo: Metaheurísticas en paralelo



Miguel Ángel López Bernal

May 15, 2025

1 Introducción

Los problemas de optimización son uno de los temas más importantes en la actualidad, por lo que su resolución es de suma importancia. Existen muchos métodos de optimización, las metaheurísticas son una forma muy intuitiva de resolverlos. Estos algoritmos basan su idea en conceptos de la naturaleza. En este trabajo se hizo una exploración del algoritmo de las luciérnagas que, como su nombre lo indica, está basado en el comportamiento de estas, que a través del brillo de cada una de ellas es capaz de atraer a sus compañeros o alertar de los peligros inminentes.

Por la naturaleza de este proceso. Sin embargo, aplicado a problemas de optimización de muchas variables, este requiere una mayor cantidad de individuos para poder alcanzar resultados óptimos, lo que podría implicar un crecimiento exponencial de este. Es por ello que se hará una implementación en paralelo, buscando una optimización del tiempo de ejecución de este. Durante este trabajo se hará una breve explicación de cómo se busca la paralelización e implementación, además de los resultados de optimización de funciones y mejoras en tiempo usando este algoritmo en paralelo.

2 Problemas de optimización

Buscamos resolver el siguiente problema,

$$\min_x f(x)$$

es decir, buscamos un x de tal forma que la función alcance un punto crítico, para este caso, buscamos puntos donde la función alcance un valor mínimo. Por ejemplo si buscamos optimizar la siguiente función,

$$\min_{\vec{x}} f(\vec{x}) = x_1^2 + x_2^2$$

Podemos resolver este problema de manera analítica, y con la facilidad de tener una función convexa, obtenemos que su mínimo global esta en el punto $\vec{x}^* = (0, 0)$. Sin embargo, existirán muchas funciones que no puedan ser resueltas de manera analítica, por ello haremos uso de algoritmos genéticos.

En aplicaciones, se busca optimizar funciones de costo, es decir, si buscamos un modelo que ajuste nuestros datos experimentales de la mejor forma posible para poder explicar un fenómeno de interés. Estos problemas tienen generalmente la siguiente forma,

$$\min_{\vec{c}} \sum_{i=1}^n (y_i - \hat{f}(\vec{x}, \vec{c}))^2$$

En este caso en especial tenemos una función del tipo mínimo cuadrado, sin embargo, dependiendo de los grados de libertad de nuestro problema, será mas o menos complejo hallar $\vec{c}$ de tal forma que la suma se minimice.

3 Algoritmo de las luciérnagas

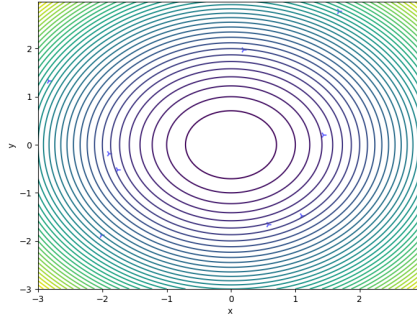
Este algoritmo obtiene su nombre del comportamiento de las luciérnagas, y como menciona [?] ??, se asume lo siguiente,

- Una luciérnaga será atraída a otra, independientemente de su sexo ya que estas son unisexuales.
- La atracción es proporcional al brillo donde la luciérnaga menos brillante será atraída a la mas brillante. También, la atracción decrementa cuando la distancia de dos luciérnagas incrementa.
- Si el brillo de ambas luciérnagas es el mismo, las luciérnagas se mueven de manera aleatoria.

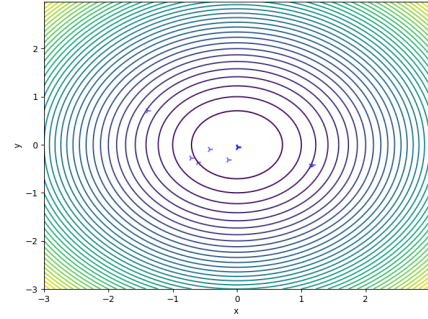
3.1 La atracción de la luciérnaga

La atracción(el brillo) I de una luciérnaga j está basado en el grado del brillo de la luciérnaga i y la distancia r_{ij} entre la luciérnaga i y la luciérnaga j .

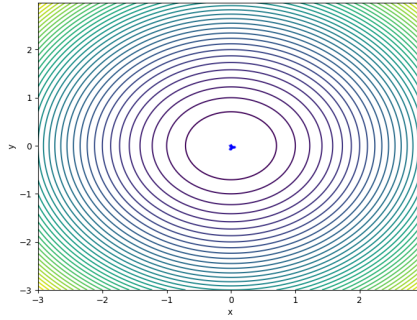
Si tenemos n luciérnagas, y x_i corresponde a la solución de la luciérnaga i . El brillo de la luciérnaga i esta asociado con la función objetivo $f(x_i)$. El brillo I de una luciérnaga es escogido para revelar su reciente posición de de la función objetivo. En este caso para un problema de minimización se definirá como sigue,



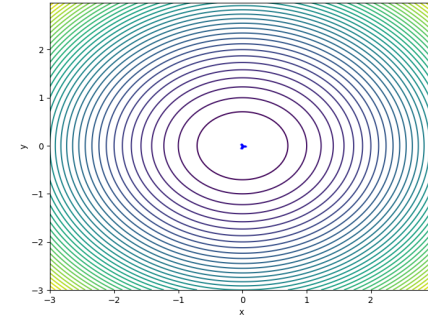
(a) Estado inicial



(b) Primer iteración



(c) Segunda iteración



(d) Tercer iteración

Figure 1: Representación de los conjuntos de nivel de $f(x,y) = x^2 + y^2$, y la ubicación de las libélulas en las iteraciones.

$$I_i = \frac{1}{f(x_i)}$$

La luciérnaga menos brillante es atraída y movida a la mas brillante, y cada luciérnaga tiene un cierto grado de atracción β . Cuyo valor esta basado en la distancia entre luciérnagas. La función de atracción esta dada por,

$$\beta = \beta_0 e^{-\gamma r^2}$$

donde β_0 es el valor de la atracción de la luciérnaga en $r = 0$ y γ es el coeficiente de absorción de luz del medio.

3.2 El movimiento hacia una luciérnaga atractora

Como menciona [1], el movimiento de una luciérnaga i en la posición x_i moviéndose hacia una luciérnaga mas brillante j en la posición x_j esta dado por,

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r^2} (x_j - x_i) + \alpha \epsilon_i$$

El segundo término se debe a la atracción de la luciérnaga x_j y $\alpha \epsilon_i$ es un parámetro de aleatoriedad. Entonces, si $\beta_0 = 0$ se convierte en un simplemente movimiento aleatorio.

El algoritmo compara la atractividad de la nueva posición de la luciérnaga con la anterior. Si la nueva posición posee un valor de atracción mayor, la luciérnaga es movida a la nueva posición, de otra forma la luciérnaga mantiene su posición actual. El criterio de paro esta basado en un numero arbitrario predefinido de iteraciones o un a tolerancia.

El movimiento de la luciérnaga mas brillante esta basado en,

$$x_i(t+1) = x_i(t) + \alpha \epsilon_i$$

En la figura 1 se puede observar el comportamiento del algoritmo de las luciérnagas tras algunas iteraciones con una población inicial de 10 individuos. En 1a podemos observar la posición inicial de las libélulas. Posteriormente 1b muestra las nuevas posiciones de las luciérnagas después de la primera iteración. En este caso el algoritmo se acerca muy rápido a el punto donde se encuentra el valor mínimo de la función como lo muestran 1c y 1d.

A continuación se muestra el algoritmo.

Input: n : El número de individuos en la generación; d : La dimensión del dominio de la función; $\vec{r}$: Las restricciones del problema ; α : El coeficiente de aleatoriedad; β : El factor de atracción; $maxIters$: El número máximo de iteraciones; TOL : La tolerancia deseada.

Output: x^* : El valor aceptable de $f(x^*)$

```

1 Define  $t = 0, \alpha = 0$ 
2 while  $t < maxIters$  do
3   for  $k = 1; n$  do
4     if  $f(x_k) < TOL$  then
5       Mostrar Procedimiento exitoso
6       return  $x^*$ 
7     end
8   end
9   for  $i = 1; n$  do
10    for  $j = 1; n$  do
11      if  $i \neq j$  then
12        if  $bright_j > bright_i$  then
13          movefirefly  $i$  to firefly  $j$ 
14        end
15      end
16    end
17  end
18 end
19 Mostrar Procedimiento no exitoso
20 return
```

Algorithm 1: Algoritmo de las luciérnagas

4 Implementación en paralelo

La naturaleza de este algoritmo permite hacer algunas implementaciones en paralelo, pero esto incurre en algunos ajustes adicionales. Ya que, en secuencial vamos actualizando cada luciérnaga en orden, es decir, conforme el ciclo avanza la luciérnaga en turno actualizara su posición con respecto al resto que es mayor que esta. Por el contrario, en paralelo se hacen varias actualizaciones a la ves, por lo que para evitar las condiciones de carrera, se creo una matriz adicional para guardar las nuevas posiciones. Sin embargo, esta actualización también podía ser paralelizada. El nuevo algoritmo paralelizado se veria de la siguiente manera

Input: n : El número de individuos en la generación; d : La dimensión del dominio de la función; $\vec{r}$: Las restricciones del problema ; α : El coeficiente de aleatoriedad; β : El factor de atracción; $maxIters$: El número máximo de iteraciones; TOL : La tolerancia deseada.

Output: x^* : El valor aceptable de $f(x^*)$

```

1 Define  $t = 0, \alpha = 0$ 
2 while  $t < maxIters$  do
3   Ciclo para actualizar las posiciones en paralelo
4   Paralelizar el ciclo for  $k = 1; n$  do
5     if  $f(x_k) < TOL$  then
6       Mostrar Procedimiento exitoso
7       return  $x^*$ 
8     end
9   end
10  Paralelizar el ciclo for  $i = 1; n$  do
11    for  $j = 1; n$  do
12      if  $i \neq j$  then
13        if  $bright_j > bright_i$  then
14          movefirefly  $i$  to firefly  $j$ 
15        end
16      end
17    end
18  end
19 end
20 Mostrar Procedimiento no exitoso
21 return

```

Algorithm 2: Algoritmo de las luciérnagas

En la línea 3 es la actualización necesaria para evitar condiciones de carrera, pero se pudo paralelizar el ciclo. En la línea 10, se paralelizó el primer ciclo, y en este caso, se dividirá en partes y cada una de ellas tomara una parte de la población, y hará la comparación con el resto. Y finalmente en la línea 4 se hizo un ciclo para la verificación de la tolerancia, para que el programa finalice una vez alcanzada la tolerancia.

5 Resultados

5.1 Optimización de funciones

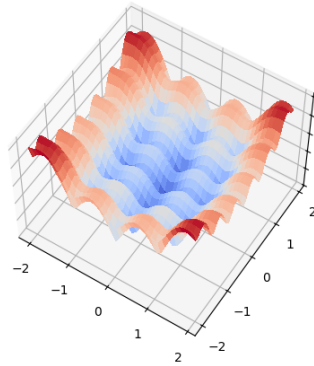
Para probar el algoritmo se seleccionaron las siguientes funciones:

5.1.1 Función sinoidal

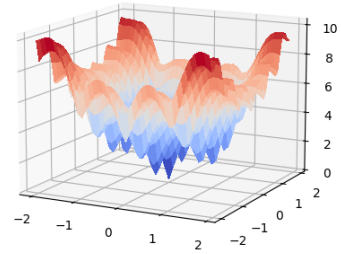
$$f(x, y) = x^2 + y^2 + 3\sqrt{\sin^2(5x) + \sin^2(5y)} + 0.1$$

función restringida al intervalo $\Omega = [-2, 2] \times [-2, 2]$

La función sinoidal tiene la siguiente forma visualizada en 3 dimensiones,

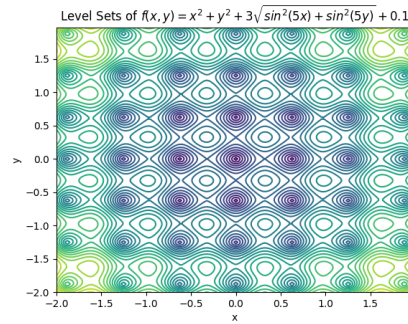


(a) Función sinoidal



(b) Función sinoidal

Si visualizamos sus curvas de nivel,

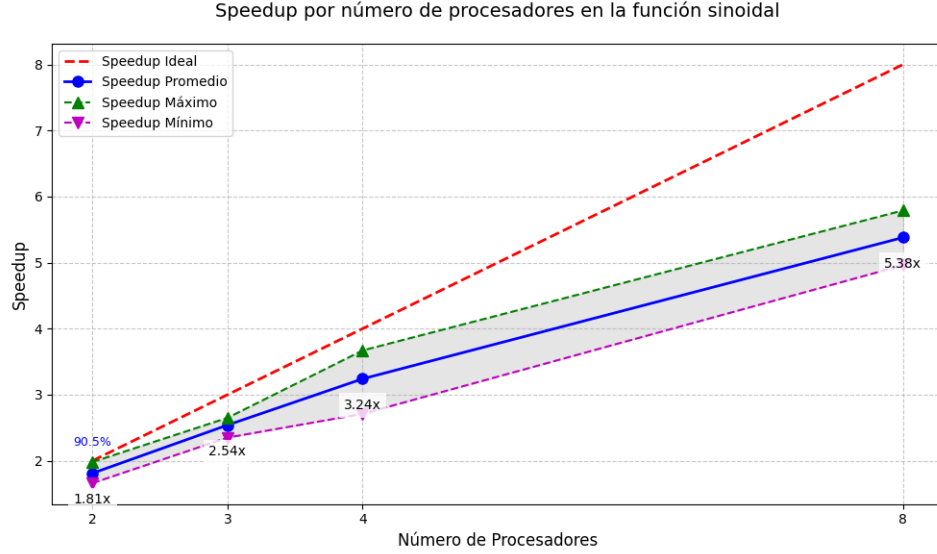


Observamos de la tabla 1 que el punto con el valor mas pequeño es $(-0.00000, 0.00004)$

Table 1: Results for sinusoidal function

x	y	$f(x, y)$	Time (s)	Num. of processes
0.00001	-0.00032	0.370763	43.640000	2
-0.00003	0.00029	0.203192	44.660000	2
-0.00013	0.00034	0.105385	45.025000	2
-0.00000	0.00004	0.100576	43.809000	2
-0.00031	0.00018	0.442786	30.176000	3
-0.00004	0.00005	0.100947	30.446000	3
-0.00033	-0.00041	0.336370	29.544000	3
0.00005	-0.00029	0.549921	30.519000	3
-0.00020	-0.00006	0.103052	23.570000	4
-0.00004	-0.00003	0.100835	22.944000	4
-0.00006	-0.00004	0.241745	22.269000	4
-0.00001	-0.00008	0.326942	23.195000	4
0.00009	0.00025	0.103939	14.818000	8
0.00007	0.00008	0.101570	14.415000	8
-0.00060	-0.00100	0.117476	17.474000	8
-0.00010	0.00006	0.101811	15.797000	8

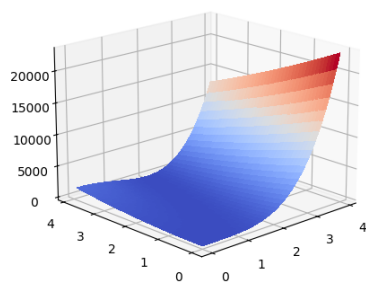
También se realizó la gráfica del speed up, se muestra en la figura ??.



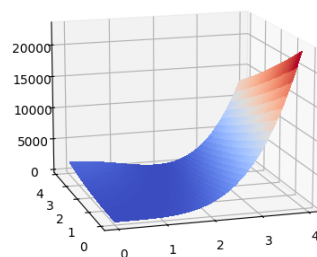
5.1.2 Función de Rosenbrock

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

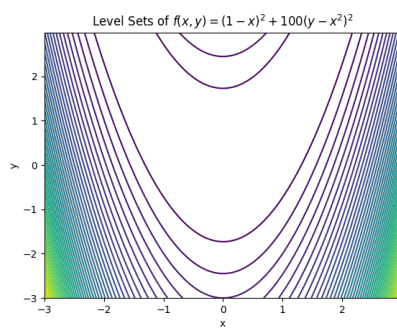
función restringida al intervalo $\Omega = [0, 4] \times [0, 4]$,



(a) Función sinoidal



(b) Función sinoidal



El valor mas pequeño, obtenido esta en el punto $(0.999, 0.998)$, muy cercano al punto mínimo real.

Table 2: Results for Rosenbrock function

x	y	$f(x, y)$	Time (s)	Num. of process
0.86978	0.75673	0.302152	43.875000	2
0.99921	0.99843	0.000002	44.551000	2
0.98083	0.96220	0.000704	43.653000	2
1.06747	1.13957	0.004621	39.992000	2
0.87613	0.76711	0.017672	30.967000	3
1.00992	1.01981	0.000265	30.341000	3
0.78916	0.62282	0.044476	30.663000	3
1.18412	1.40104	0.045826	30.756000	3
1.01623	1.03345	0.005409	23.068000	4
0.91812	0.84307	0.006842	23.307000	4
1.04727	1.09691	0.098390	22.788000	4
1.27633	1.62845	0.079607	22.238000	4
1.20499	1.45073	0.058402	14.361000	8
1.01578	1.03092	0.008167	14.361000	8
1.20059	1.44135	0.040272	14.271000	8
0.96590	0.93227	0.006078	15.395000	8

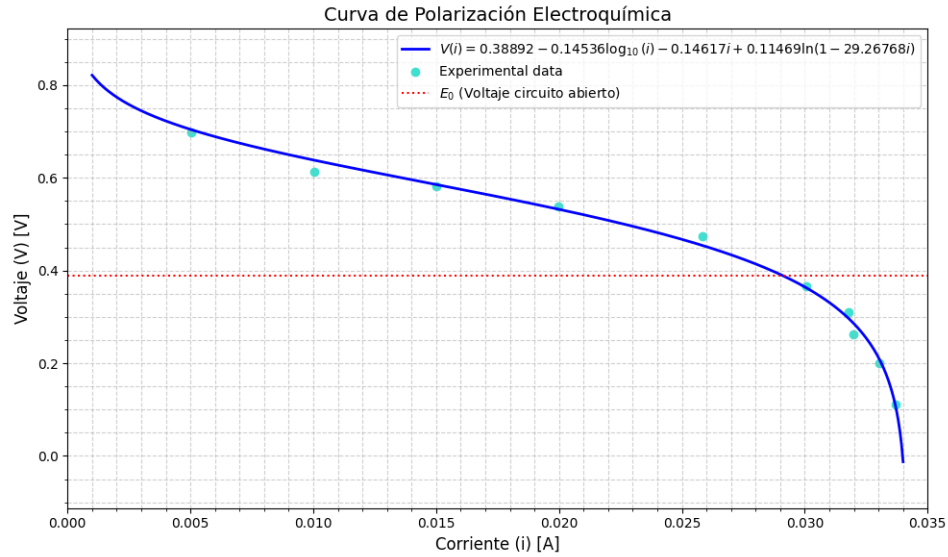
5.1.3 Función de celda de combustible

Para hallar los valores que optimizan la función se seleccionó la función MSE, y los resultados se muestran en la tabla 3

Table 3: Resultados de optimización para la función de costo

Num. of process	E_0^*	b	Re	C_1	C_2	$f(\mathbf{x})$	Evals.	Tiempo (s)
2	0.50000	0.09597	0.78897	0.12632	29.09704	0.001898	15	43.852
2	0.50000	0.08574	0.00000	0.12715	29.14902	0.006826	15	43.430
2	0.50000	0.09600	0.10404	0.15108	28.56697	0.003643	15	44.649
3	0.46451	0.10846	1.00000	0.11287	29.20636	0.002032	15	32.379
3	0.26834	0.20000	0.98010	0.08302	29.42460	0.005840	15	32.548
3	0.39769	0.14137	0.50129	0.10988	29.29446	0.002054	15	33.662
4	0.44596	0.11837	1.00000	0.11501	29.17188	0.002122	15	27.107
4	0.29783	0.18561	0.37491	0.09770	29.38153	0.038016	15	26.186
4	0.47053	0.10259	0.74394	0.11004	29.35596	0.002108	15	28.346
8	0.38554	0.14620	0.78500	0.10554	29.28708	0.002266	15	18.065
8	0.40481	0.13420	0.41240	0.10349	29.44888	0.009234	15	17.270
8	0.38892	0.14536	0.14617	0.11469	29.26768	0.001958	15	17.114

Se muestran el ajuste obtenido en la siguiente gráfica.



6 Conclusiones

Los resultados obtenidos durante la realización de este trabajo nos permite ver que el algoritmo de las luciérnagas es un muy buen algoritmo para la resolución de problemas de optimización con restricción. No se exploró de manera mas profunda en la variación de los parametros α , β y γ y se optó por utilizar valores ya estudiados, obteniendo buenos resultados, pero podría explorarse a futuro estos parámetros con la implementación en paralelo y ver sis se puede mejorar el rendimiento.

7 Referencias

- [1]Johari, Nur & Zain, Azlan & Mustaffa, Noorfa & Udin, Amirmudin. (2013). Firefly Algorithm for Optimization Problem. Applied Mechanics and Materials. 421. 512-517. 10.4028/www.scientific.net/AMM.421.512.
- [2]Yang, X.-S. (2010). Firefly algorithms for multimodal optimization. In Stochastic algorithms: Foundations and applications (pp. 169-178). Springer, Berlin, Heidelberg. arXiv version: <https://arxiv.org/abs/1003.1466>