JÓVENES EN LA CIENCIA
REVISTA DE DIVULGACIÓN CIENTÍFICA

COMPARATIVA DE ALGORITMOS BIOINSPIRADOS PARA LA OPTIMIZACIÓN DE REDES NEURONALES

Aguayo González Jaime Francisco (1), Felipe De Jesús Trujillo Romero (2)

1 [Licenciatura en Computación, Universidad de Guanajuato] | [jaime.aguayo@cimat.mx]

2 [División de Ingenierías, Campus Irapuato Salamanca, Universidad de Guanajuato] | [fdj.trujillo@ugto.mx]

Resumen

En este artículo se presenta una comparativa de diferentes algoritmos bioinspirados con la finalidad de ser usados para encontrar la estructura óptima de una red neuronal artificial. En la primera parte se presentan los algoritmos revisados y se explica el porqué de la selección de los algoritmos que se comparan. Posteriormente se realiza una comparativa de su rendimiento en la optimización de funciones difíciles de optimizar. Finalmente se usan los mejores algoritmos para encontrar la estructura óptima de una red neuronal artificial de tipo backpropagation, esto es, encontrar tanto el número de neuronas como el número de capas de la red. Los resultados y la discusión de estos se presenta al final del artículo, así como sugerencias para trabajo futuro.

Abstract

In this article we show a comparative between some bioinspired algorithms. Our goal is to show the capability of the algorithms to find the optimal ANN structure for a problem in order to implement the network in a embedded system. Firstly, the chosen of the ABC, CSA, CSO, WSO, GWO and MBO algorithms is explained. In the second part, the results of the optimization of several functions using the algorithms are discussed. With those results, the four most competitive algorithm are chosen to be compared in two ANN problems. The third part is the result of testing the algorithms in the optimization of two neural networks, one working as a XOR gate and the other classifying digits written in a grid. The results clearly exhibit the capability of the methods toward finding the optimal structure in the ANN's.

Palabras Clave

ANN; biología; computación; inteligencia artificial; machine learning; metaheuristica; optimización.

INTRODUCCIÓN

Los problemas de optimización en la vida real usualmente son difíciles de resolver, incluso algunas veces es necesario lidiar con problemas de tipo NP-hard. Para muchos de estos problemas se han desarrollado estrategias metaheurísticas que ayudan a agilizar la tarea de optimización, de hecho, muchas de las soluciones a los problemas que nos enfrentamos en la vida real solo son posibles obtenerlas a prueba y error. Como resultado de esta necesidad, no es de extrañarse que exista una gran cantidad de estos métodos. En este artículos pondremos atención a una clase especial de estrategias metaheuristicas, llamadas algoritmos bioinspirados de optimización.

Como su nombre lo indica, los algoritmos bioinspirados son métodos de optimización que han sido creados a partir de inspiraciones de sucesos biológicos naturales. Estos eventos biológicos se han desarrollado y perfeccionado de manera natural a través de procesos de evolución, por lo que la idea de *imitar* a la naturaleza parece una buena idea. Tal vez esto último ha motivado a los investigadores a crear muchos algoritmos en los últimos años, muestra de ello son los más de 40 algoritmos publicados para 2013 [1], pero dentro del presente trabajo podremos ver que muchos más se han desarrollado desde entonces.

Uno de los primeros algoritmos bioinspirados que surgieron fue el *Particle Swarm Optimization* (PSO), que tuvo éxito por su resultados aceptables y su simplicidad en la implementación. De hecho, uno de los primeros usos del PSO fue la optimización de los pesos de una red neuronal artificial XOR [2]. Una red neuronal artificial (ANN por su nombre en inglés) es un ejemplo de métodos inspirados en la naturaleza, sin embargo en este caso su función es muy distinta, pues esta sirve para realizar aproximaciones, como bien lo explica el teorema de aproximación universal [3]. Las redes neuronales artificiales han sido de sumo interés en las últimas dos décadas, tanto para investigadores como para practicantes de diferentes disciplinas, en particular se ha tenido interés en optimizarlas. La optimización de una ANN puede darse desde una de al menos tres propiedades: la optimización de los pesos de las conexiones de la red, la optimización de las funciones de activación y la optimización de la estructura de la red neuronal, es decir, el número de capas y neuronas. El uso de estrategias metaheuristicas en el diseño de redes neuronales no es un tema nuevo, de hecho en [3] se puede encontrar un buen resumen del trabajo realizado en el área en las últimas dos décadas. Nuestro interés es la comparativa de diversos algoritmos bioinspirados de optimización para la optimización de la estructura de una red.

Con la estructura de la red neuronal, nos referimos al número de capas de la red y el número de neuronas de cada una. Encontrar la estructura óptima de una red neuronal puede ser estudiada desde al menos dos puntos de vista, que el error que comete la red en un conjunto de datos de prueba sea el menor posible, o que fijando un error chico, el tiempo que le tome a la red en procesar la información sea bajo. En el presente artículo veremos el primer enfoque sin dejar de lado el segundo.

Tabla 1: Algoritmos bioinspirados a comparar

Algoritmo	Abreviación	Año de publicación	Autor	Referencia
Artificial Bee Colony	ABC	2007	Karaboga y Basturk	[4]
Chicken Swarm Optimization	CSO	2014	Xianbing Meng, Yu Liu et al.	[5]
Cuckoo Search Algorithm	CSA	2009	Yang	[6]
Gray Wolfe Optimizer	GWO	2014	Mirjalili S., Mirjalili S.M., Lewis A.	[7]

Monarch Butterfly Optimization	MBO	2015	Wang et al	[8]
Whale Swarm Optimization	WSO	2017	Zeng et al.	[9], [10]

Para la comparativa que se presenta a continuación, se seleccionaron los algoritmos presentados en la tabla 1. Los algoritmos en la tabla fueron seleccionados de entre los más de 15 algoritmos consultados por lo siguiente:

CSA: Al ser el algoritmo que menos parámetros solicita al usuario y estar basado en los vuelos de Lévy. Además de tener las mejores propiedades de entre *Harmony Search* y los métodos tipo enjambre según el artículo que lo presenta.

ABC: Al ser un algoritmo de características similares a ACO (ampliamente encontrado en la bibliografía) pero con mejor desempeño, además de estar basado en el orden jerárquico de los individuos.

GWO: Al ser un algoritmo robusto y complicado de implementar, uno espera que tenga buenos resultados, además de ser basado en orden jerárquico y tener una búsqueda local y una global que ayuda a tener una convergencia rápida.

CSO: Al igual que GWO, basado en el orden jerárquico. Es una versión más simple que el algoritmo anterior. Además de ser un algoritmo de reciente creación.

MBO: Al ser un algoritmo novedoso en la división de los agentes en dos grupos y por tener muy buenos resultados en problemas de alta dimensionalidad, lo que podría ser útil al optimizar la red neuronal.

WSO: Al ser implementado en el artículo [10], en una red neuronal tipo *feedforward* y por ser un algoritmo publicado apenas el año pasado, lo que lo hace el más reciente de los algoritmos analizados.

En la siguiente sección se describe la metodología para probar los algoritmos, tanto en funciones de difícil optimización, como en un ejemplo de reconocimiento de dígitos usando redes neuronales de tipo *backpropagation*. En la tercera sección se presentan y discuten los resultados obtenidos. Al final se propone trabajo a realizar en un futuro.

MATERIALES Y MÉTODOS

Los algoritmos a comparar fueron implementados en C++ usando la librería GSL de la GNU que se puede encontrar en [11] y de la librería estándar de C++. Los algoritmos se implementaron según los artículos consultados que se citan en la tabla presentada con anterioridad.

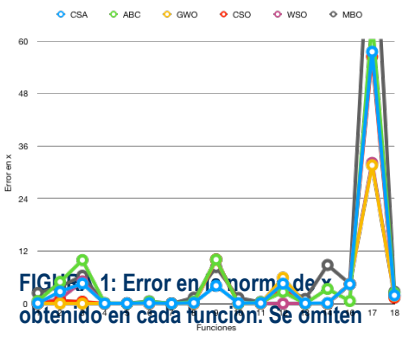

FIGURA 1: Error en la norma de x obtenido en cada función. Se omiten

La primera prueba consistió en optimizar 18 funciones, que fueron seleccionadas de [12] por tener múltiples mínimos locales, o por tener un mínimo global de difícil acceso.

La segunda prueba consistió en encontrar los pesos y la estructura de una red neuronal XOR, mientras que la tercera prueba, consistió en entrenar una red neuronal de tipo *backpropagation* para el reconocimiento de dígitos escritos en una rejilla de 5x3.

Para las pruebas con la red neuronal, se utilizó la librería *Fast Neural Network* que puede ser encontrada en [13], con funciones sigmoidales en las neuronas ocultas y funciones lineales en las neuronas de salida.

RESULTADOS Y DISCUSIÓN

Antes de continuar con los resultados, cabe mencionar que los parámetros usados en los algoritmos del experimento 1 fueron ajustados haciendo algunas pruebas con la primeras tres funciones, mientras que para los siguientes experimentos se usaron los parámetros sugeridos en los artículos que los presentan.

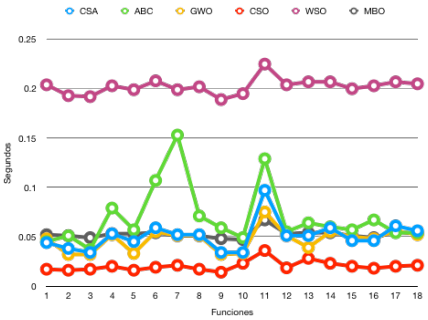

FIGURA 2: Comparativa en el tiempo de ejecución de los algoritmos.

Para cada función se ejecutó el algoritmo 100 veces. En la figura 1 se ilustra el promedio de la distancia euclideana entre el óptimo teórico y el resultado que arroja el método en cada prueba, es decir, si x^* es el óptimo teórico y x el óptimo calculado por el método, entonces se reporta $\| x - x^* \|$. En la gráfica en cuestión se observa que el resultado de los métodos CSA, GWO y WSO son de los más pequeños, mientras que ABC y MBO fueron los que obtuvieron los peores resultados. En la figura 2 se observan los tiempos de ejecución promedios de las 100 pruebas. El más rápido fue el CSO y el más lento fue el WSO, estando todos los demás entre éstos. Para el CSA y GWO obtenemos tiempos aceptables, mientras que para CSO, que tuvo el tiempo más bajo, tampoco se obtiene un resultado bajo en el error. Por lo tanto, podemos decir que los algoritmos con mejores resultados en tiempo de ejecución fueron CSA, CSO y GWO, mientras que en menor error fueron CSA, GWO y WSO. Tomaremos los cuatro para las siguientes pruebas.

Tabla 2: Resultados de la optimización de la red neuronal XOR.

Algoritmo	Número de capas ocultas	Neuronas capa 1	Tiempo de ejecución (s)	ECM
CSO	1	2	135.86	5.85e-06
GWO	1	5	124.27	2.86e-05
WSO	1	4	128.77	5.07e-05
CSA	1	22	383.747	1.35e-05

Muestro en la tabla 2 los resultados de la segunda prueba, donde se determinó la estructura óptima de una red tipo XOR. El número máximo permitido de capas ocultas fue 4 y el máximo de neuronas por cada de 32. El error cuadrático medio de tolerancia fue de 1E-5 y un máximo de épocas de 600. Para los algoritmos bioinspirados se usaron 50 agentes y 500 iteraciones. Desde el punto de vista del primer enfoque de optimización de la estructura de una red neuronal, el mejor algoritmo fue CSO pues obtiene un error menor, desde el otro enfoque, dado que la literatura marca que la estructura óptima de este problema es una sola capa oculta de dos neuronas, podemos decir que la optimización fue exitosa con CSO al llegar a una red lo más chica posible, mientras que en los demás solo pudieron determinar correctamente el uso de una capa oculta. El método que tuvo el peor desempeño fue CSA al ser el más lento y el que determinó un número muy grande de neuronas.

Tabla 3: Resultados de la optimización de la red neuronal de reconocimiento de dígitos.

Algoritmo	Número de capas ocultas	Neuronas capa 1	Tiempo de ejecución (s)	ECM
CSO	1	32	200.33	1.35e-05
GWO	1	31	198.20	7.85e-06
WSO	1	26	202.32	6.83e-06
CSA	1	31	668.50	8.25e-06

Para la última prueba, se usaron las mismas restricciones que en el anterior para reconocimiento de dígitos en una rejilla de 5x3. Los resultados se muestran en la tabla 3. En este caso vemos que los algoritmos seleccionan una sola capa oculta y que el número de neuronas

ronda de 26 a 32, siendo WSO el que menos neuronas dictamina y CSO es que menos. El error obtenido con cada uno es pequeño y muy parecido, por lo que los cuatro hicieron un buen trabajo, sin embargo se prefiere el resultado de WSO al usar menos número de neuronas.

Cabe mencionar que los resultados obtenidos no son los mejores desde el punto de vista de hacer la red con el menor número de neuronas posibles, ya que con el uso de 1 capa oculta con 8 neuronas y un número máximo de épocas de 1000, se llega a tener un ECM de **9.32e-05. Este último ejemplo indica que tomando como función objetivo el valor del error cuadrático medio de la red, no nos garantiza que la arquitectura de la red que se genere, tenga un número pequeño de neuronas, pues después de muchas pruebas, se puede observar como depende de factores como el número máximo de épocas y el ECM deseado. De hecho, en otra prueba realizada con un número máximo de épocas mayor (1000), se obtuvieron mejores resultados en la prueba de reconocimiento de dígitos, obteniendo la tabla 4. Sin embargo, es de gran ayuda porque todos los métodos coincidieron en el número de capas a usar, que por cierto es el óptimo al ser el menor número de capas que resuelven el problema. Además se obtienen resultados muy buenos como CSO en la tabla 2 o WSO en la tabla 3.**

Tabla 4: Resultados de la optimización de la red neuronal de reconocimiento de dígitos con un mayor número de épocas.

Algoritmo	Número de capas ocultas	Neuronas capa 1	ECM
CSO	1	8	9.32e-05
GWO	1	14	9.35e-05
WSO	1	9	8.80e-05
CSA	1	12	9.25e-05

CONCLUSIONES

Como conclusión vemos que los algoritmos bioinspirados son muy útiles en problemas donde la función objetivo es muy difícil de describir, como en el caso de la arquitectura de una red neuronal artificial. Se obtuvieron resultados favorables, que en algunos casos no dieron el resultado más óptimo en el sentido “menor número de capas y neuronas es mejor”, pero sí que obtuvieron buenos resultados que pueden ser de mucha ayuda en problemas donde de no se tenga la más mínima idea de qué tipo de arquitectura se tiene que implementar en una red neuronal para resolverlo. Con estos resultados se podría buscar una estructura que bien podría ser implementada en un sistema embebido.

Como trabajo futuro se podría modificar la función objetivo para lograr un resultado deseado, es decir, optimizar según otros criterios. De esta forma podriamos usar los algoritmos bioinspirados para podar una red neuronal dada. Esto ultimo es muy útil dado el aumento del uso de sistemas embebidos, donde al implementar una red neuronal, se tiene que lidear con los pocos recursos de éstos dispsitivos.

AGRADECIMIENTOS

Agradezco a todos los involucrados en el proyecto, en particular a mi asesor. Un agradecimiento especial a mis tías Aurora y María de Jesús Aguayo quienes estuvieron apoyándome durante el verano.

REFERENCIAS

[1] Fister Jr., I., Yang, X.S., Fister, I., Brest, J. & Fister, D. (2013). A Brief Review of Nature-Inspired Algorithms for Optimization. Elektrotehniski Vestnik/ Electrotechnical Review, 80(3), 1-7.
[2] Kennedy, J. & Eberhart, R. (1995). Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks. IV. 1942–1948.
[3] Ojha, V.K., Abraham, A., & Snášel, V. (2017). Metaheuristic Design of Feedforward Neural Networks: A Review of Two Decades of Research. *Eng. Appl. of AI, 60*, 97-116.
[4] Karaboga, D. & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of global optimization, 39(3), 459–471.
[5] 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).
[6] Yang, X. S., & Deb, S.(2009). Cuckoo search via lévy flights. In Nature & Biologically Inspired Computing. NaBIC 2009. 210–214.
[7] Mirjalili, S., Mirjalili, S.M., & Lewis, A. Gray wolf optimizer. Advances in Engineering Software, 69 (2014), 46-61.
[8] Wang, G. G., Deb, S., & Cui, Z. (2015). Monarch butterfly optimization. Neural Computing and Applications, 10,
[9] Zeng, B., Gao, L., & Li, X. (2017). Whale Swarm Algorithm for Function Optimization. ICIC.
[10] Aljarah, Ibrahim & Faris, Hossam & Mirjalili, Seyedali. (2016). Optimizing connection weights in neural networks using the whale optimization algorithm. Soft Computing. 22
[11] https://www.gnu.org/software/gsl/
[12] Wikipedia contributors. (2018, June 26). Test functions for optimization. In *Wikipedia, The Free Encyclopedia*. Retrieved 23:00, July 25, 2018, from https://en.wikipedia.org/w/index.php?title=Test_functions_for_optimization&oldid=847632071 [13] http://leenissen.dk/fann/wp/