DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI

ISSN: 2007-6363

Publicación Semestral Pädi Vol. 11 No. Especial 3 (2023) 57-65

# Algoritmo de optimización metaheurístico inspirado en el autómata celular LIFE
# Metaheuristic optimization algorithm inspired by the LIFE cellular automata

O. López-Arias[a], J. C. Seck-Tuoh-Mora[b], N. Hernández-Romero[b], J. Medina-Marín[b], G. Juárez-Martínez[c]

[a] *Doctorado en Ciencias en Ingeniería, con Énfasis en Análisis y Modelación de Sistemas, Universidad Autónoma del Estado de Hidalgo, 42184, Pachuca, Hidalgo, México.*
[b] *Área Académica de Ingeniería y Arquitectura, Universidad Autónoma del Estado de Hidalgo, 42184, Pachuca, Hidalgo, México.*
[c] *Laboratorio de Robótica de Vida Artificial, Escuela Superior de Computo, Instituto Politécnico Nacional, México.*

**Resumen**

Debido a que existe gran variedad de problemas de optimización, surgió la idea de diseñar una nueva metaheurística inspirada en el comportamiento dinámico de los autómatas celulares. Para ello, se adaptan las reglas de evolución de un conocido autómata celular llamado “El juego de la vida o LIFE”. Estas reglas se implementaron con vectores de valores reales, de esta manera se realizan las acciones de exploración y explotación en el proceso de optimización global. El algoritmo se probó mediante un estudio comparativo utilizando tres metaheurística reconocida por su rendimiento, una de ellas inspirada en un autómata celular distinto. Esta metaheurística se denomina: Algoritmo de autómatas celulares de estado continuo (CCAA, por sus siglas en inglés). Para el estudio se utilizaron librerías de funciones de prueba presentadas en la última competición realizada del congreso de computación evolutiva 2022 (CEC 2022, por sus siglas en inglés). Se comprobó mediante pruebas estadísticas que los resultados obtenidos entre los distintos algoritmos, que la propuesta llamada Algoritmo Optimizador Life (AOL) fue capaz de competir y, en algunos casos, mejorar la solución buscada. Se concluye que las reglas propuestas tienen un grado de eficiencia muy aceptable con los algoritmos comparados.

*Palabras Clave:* Optimización global, juego de la vida, metaheurística, aplicaciones de ingeniería.

**Abstract**

Since there is a great variety of optimization problems, the idea of designing a new metaheuristic inspired by the dynamic behavior of cellular automata arose. For this purpose, the evolution rules of a well-known cellular automaton called "The game of life or LIFE" were adapted. These rules were implemented with vectors of real values, thus performing the exploration and exploitation actions in the global optimization process. The algorithm was tested by a comparative study using three metaheuristics recognized for their performance, one of them inspired by a different cellular automaton. This metaheuristic is called: Continuous State Cellular Automata Algorithm (CCAA). For the study, libraries of test functions presented at the last competition of the Evolutionary Computing Congress 2022 (CEC 2022) were used. It was verified by statistical tests that the results obtained among the different algorithms, that the proposal called Algorithm Optimizer Life (AOL) was able to compete and, in some cases, improve the solution sought. It is concluded that the proposed rules have a very acceptable degree of efficiency with the compared algorithms.

*Keywords:* Global optimization, game of life, metaheuristics, engineering applications.

## 1. Introducción

La resolución de problemas de optimización consiste en mejorar alguna solución relativamente buena conseguida previamente o, incluso, llegar a conseguir una solución óptima de acuerdo con un problema formulado en lenguaje matemático. En este contexto, el criterio que evalúa la calidad de una solución es cuantitativo, generalmente asociado a un costo y denominado función objetivo. Estos problemas implican cada vez un mayor número de variables, así como una relación altamente no lineal entre las mismas, que dificultan su optimización (Sarker, R. Kamruzzaman, J. y Newton C., 2003). Esto se ve con claridad en varios campos de aplicación como problemas de diseño mecánico, ingeniería

*Autor para la correspondencia: Omar López-Arias: lo387695@uaeh.edu.mx
**Correo electrónico**: lo387695@uaeh.edu.mx (Omar López-Arias), jseck@uaeh.edu.mx (Juan Carlos Seck-Tuoh-Mora), nhromero@uaeh.edu.mx (Norberto Hernández-Romero), jmedina@uaeh.edu.mx (Joselito Medina-Marín), gjuarezm@ipn.mx (Genaro Juárez Martínez).

**Historial del manuscrito:** recibido el 06/07/2023, última versión-revisada recibida el 11/10/2023, aceptado el 11/10/2023, publicado el 20/11/2023. **DOI:** https://doi.org/10.29057/icbi.v11iEspecial3.11425

química, electrónica de potencia o de sistemas de generación de energía (Kumar, K. y Davim, J. P. 2020). La mayoría de las veces será posible conseguir una solución óptima de un problema, sin embargo, para llegar a ella será necesario utilizar métodos con una complejidad algorítmica muy alta (Ramos, Sánchez, Ferrer, Barquín y Linares, 2010). En otros casos, para obtener soluciones necesitan información adicional que no está disponible, por lo que no conviene seguir este camino, ya que por su naturaleza son altamente no lineales, y en muchas ocasiones no diferenciables, no separables y multimodales (Brú, Fleites y Barroso, 2004). Los algoritmos clásicos de optimización basados en métodos de gradientes no pueden aplicarse de manera efectiva al quedar atrapados en mínimos locales, o carecer de las condiciones necesarias para poder converger de manera adecuada (Kumar y Davim, 2020). Debido a esto han surgido gran variedad de algoritmos metaheurísticos, los cuales han tomado como inspiración el comportamiento de sistemas físicos, biológicos, sociales o artificiales, para poder realizar las tareas de optimización sin tener que depender de la naturaleza del problema a optimizar (Gogna y Tayal, 2013). Algunos ejemplos son los algoritmos de inteligencia de enjambre en los cuales la capacidad para encontrar valores óptimos surge de forma emergente por la interacción sencilla de agentes en el espacio de búsqueda mediante un control descentralizado (Hassanien, y Emary 2018; Bansal, Singh y Pal, 2019).

El teorema de No Free Lunch (Wolpert y Macready, 1997), dice que no existe un algoritmo metaheurístico que pueda optimizar todo tipo de problemas mejor que los demás; es por esto, que existe la necesidad de seguir ideando nuevas metaheurísticas que sean capaces de optimizar problemas de mayor complejidad y en un número mayor de dimensiones. Una parte fundamental para diseñar un algoritmo metaheurístico se basa en el correcto equilibrio de las 2 fases principales de búsqueda de soluciones, conocidas como la etapa de exploración y explotación. Mientras la etapa de explotación guía la búsqueda teniendo en cuenta la información que se obtiene de las mejores soluciones encontradas hasta el momento, la exploración propicia descubrir regiones sin explorar y evitar una convergencia prematura (Kumar, K. y Davim, J. P. 2020). Lograr un balance entre estos dos objetivos es un problema de vital importancia que enfrenta la mayoría de las técnicas de búsqueda y optimización actuales.

Los autómatas celulares han sido un modelo muy recurrente de estudio para entender cómo la complejidad surge de una población formada de elementos sencillos que interaccionan de manera local. El modelo clásico de autómatas celulares es discreto en tiempo, espacio y en los posibles estados que cada individuo o célula puede tener. Sin embargo, a pesar de su sencillez, estas condiciones las hacen fáciles de implementar en una computadora y son capaces de producir comportamientos globales emergentes complejos (McIntosh, 2009). Por tal motivo, los autómatas celulares han sido ampliamente estudiados para entender y simular fenómenos físicos, químicos y biológicos (Salcido, 2011; Kotyrba, Volna y Bujok, 2015; Bilan, Bilan y Motornyuk, 2020).
La hipótesis que proponemos es que los autómatas celulares, con su capacidad inherente para modelar interacciones locales y la propagación de información, pueden proporcionar un marco sólido para el diseño de algoritmos metaheurísticos eficaces. Estos algoritmos podrían aprovechar la dinámica de los autómatas celulares para explorar y explotar el espacio de búsqueda de manera más eficiente y efectiva en comparación con enfoques tradicionales.
Para poner a prueba esta hipótesis, es necesario investigar cómo los principios de los autómatas celulares, como la adaptación local y la emergencia de patrones globales, pueden traducirse en estrategias de búsqueda inteligente en el contexto de la optimización. Además, se requerirá la evaluación y comparación de estos algoritmos metaheurísticos inspirados en autómatas celulares con los enfoques convencionales en una variedad de problemas de optimización.
Ejemplos de estos algoritmos de optimización inspirados en autómatas celulares para problemas discretos incluyen la optimización por enjambre de partículas celulares con búsqueda local adaptativa simple (CAPSO-SALS) (Seck-Tuoh-Mora, J.C. Medina-Marin, J. Martinez- Gomez, E.S. Hernandez-Gress, E.S. Hernandez-Romero, N. y Volpi-Leon, V, 2019), el algoritmo de búsqueda de vecindario local (LNSA) (Hernández-Gress, E. S. Seck-Tuoh-Mora, J. C. Hernández-Romero, N. Medina-Marín, J. Lagos-Eulogio, P. y Ortíz-Perea, J, 2020) y el algoritmo de búsqueda de vecindario global-local (GLNSA) ,(Escamilla Serna, Seck-Tuoh-Mora, Medina-Marin, Hernandez-Romero, Barragan-Vite, y Corona Armenta, 2021)
Para problemas continuos y búsqueda de óptimos globales, existen también la optimización por enjambre de partículas celulares (CPSO) ,(Shi, Y., Liu, H. Gao Liang, and Zhang, G. 2011), (Lagos-Eulogio, Seck-Tuoh-Mora, Hernandez-Romero, y Medina-Marin, 2017), el algoritmo de autómatas celulares de estado continuo (CCAA), (Seck-Tuoh-Mora, Hernández-Romero, Lagos-Eulogio, Medina-Marin y Zúñiga-Peña, 2021). Y el algoritmo de autómatas celulares mayoritario-minoritario (MmCAA),(Seck-Tuoh-Mora, Hernandez-Romero, Santander-Baños, Volpi-Leon, Medina-Marin y Lagos-Eulogio, 2022).

La parte original de este trabajo continúa adaptando un autómata celular distinto, llamado el juego de la vida o LIFE para proponer una nueva metaheurísticas para la optimización global de funciones denominada Algoritmo Optimizador Life (AOL). Esta adaptación se materializa en un algoritmo basado en tres nuevas reglas de exploración y explotación de fácil implementación tomando como inspiración las reglas que conforman el autómata celular LIFE.
AOL se compara con otras 3 metaheurísticas ampliamente reconocidas por su flexibilidad y rendimiento en la resolución de problemas de optimización de pruebas y aplicaciones de ingeniería.
Para las pruebas experimentales, se toman las 12 funciones de prueba que fueron presentadas en la competición del congreso de computación evolutiva del 2022 (CEC 2022, por sus siglas en ingles), que organiza la IEEE cada año, (Biedrzycki, Arabas y Warchulski, 2022).

Los experimentos realizados en este trabajo consideraron 30 dimensiones fijas, utilizando pruebas estadísticas de Wilcoxon, el cálculo de las medias obtenidas por los algoritmos y sus desviaciones estándar. Se pudieron

comparan y clasificar a los algoritmos seleccionados para el estudio según (Momin Jamil and Xin-She Yang, 2013).

Las metas de este trabajo es continuar con una generación de algoritmos metaheurísticos inspirados en autómatas celulares que no solo sean eficientes, sino también altamente adaptables y capaces de abordar problemas de optimización complejos en diversas disciplinas. Esto podría tener un impacto significativo en la resolución de problemas prácticos y en la investigación científica.

## 2. Descripción del juego de la vida de Conway o LIFE

Este tipo especial de autómatas fue diseñado por el matemático británico John Horton Conway en 1970; se caracteriza por tener reglas muy sencillas que pueden dar lugar a comportamientos muy complejos (Conway, 1970). Décadas después se sigue examinando hasta dónde llega la complejidad que genera y cuáles son las aplicaciones prácticas de esta construcción matemática y de otras similares (Adamatzky, 2010; Li, Wu y Li, 2018).

Para explicar en qué consiste este peculiar juego surgido del mundo de la computación teórica, primero hay que describir con más detalle lo que es un autómata celular. En palabras sencillas, es un modelo matemático que cambia paso a paso. Suele tener el aspecto de un tablero infinito, normalmente, de celdas cuadradas denominadas "células". El transcurrir del tiempo lo marca una especie de reloj universal Con cada tic de este reloj se aplican unas reglas predefinidas a las células y se toma una decisión sobre lo que sucede individualmente con cada una de ellas. Las celdas más simples solo tienen dos estados: vivas o muertas. Cuando se ha completado el cálculo de todo lo que sucede en el espacio cuadriculado la operación comienza de nuevo (McIntosh, 2009).

Elementos de un autómata celular:

**Espacio regular.** Ya sea una línea, un plano de 2 dimensiones o un espacio n-dimensional. Cada división homogénea del espacio es llamada célula.
**Conjunto de Estados**. Es finito y cada elemento o célula del espacio puede tomar un valor en a partir de un conjunto finito de estados . También se denomina alfabeto. Puede ser expresado en valores o colores.
**Configuración Inicial**. Es la asignación inicial de un estado a cada una de las células del espacio.
**Vecindades.** conjunto de células finito que se consideran adyacentes a una célula dada, así como la posición relativa respecto a ella.
**Función de Transición Local**. Es la regla de evolución que determina el comportamiento del autómata celular a partir del estado de la célula y su vecindad. Define cómo debe cambiar de estado cada célula dependiendo su estado anterior y de los estados anteriores de su vecindad. Suele darse como una expresión algebraica o un grupo de ecuaciones.
*Características del juego de la vida (LIFE)*

Se lleva a cabo un tablero comúnmente finito de dos dimensiones, subdividido en casillas llamadas celdas, la cuales pueden tener dos estados distintos: célula viva o célula muerta. Cada ocho casillas determinan su "vecindad", la cual cambia o evoluciona con cada iteración mediante la intervención de 3 reglas específicas (Eppstein, 2010; Garcia, Gomes, Jyh, Ren, y Sales, 1993). A continuación, se muestran las reglas de LIFE.

**Regla 1-** Supervivencia: si la célula central se encuentra viva y cumple con el requisito de tener 2 o 3 vecinos vivos sobrevivirá en el siguiente estado.
**Regla 2-** Nacimiento: Si la célula central se encuentra muerta o vacía y contienen en su vecindad exactamente 3 células vivas su estado futuro será el de una célula viva.
**Regla 3-** Fallecimiento: si la célula central se encuentra viva y con menos de 2 vecinos, fallece por aislamiento o soledad en el siguiente estado. Una célula viva que tenga más de 3 vecinos vivos muere por superpoblación en el siguiente estado.

Es fácil hacer pruebas con lápiz y papel dibujando algunas células y calculando lo que sucederá a cada paso, como se muestra en la Figura 1. Para la representación, se define la célula actual en la posición central del conjunto de células, las células con estado vivo se representan en color azul y con el estado de muerto a las de color blanco. En el primer paso o iteración se observa que la célula central comienza en un estado y, dependiendo del número de vecinos, esta cambia de estado en la siguiente iteración según la regla correspondiente; así continúa sucesivamente hasta alcanzar el número de iteraciones requeridas. A veces todas las células mueren, a veces entran en bucle, a veces dibujan curiosos patrones y otras parecen expandirse sin fin.

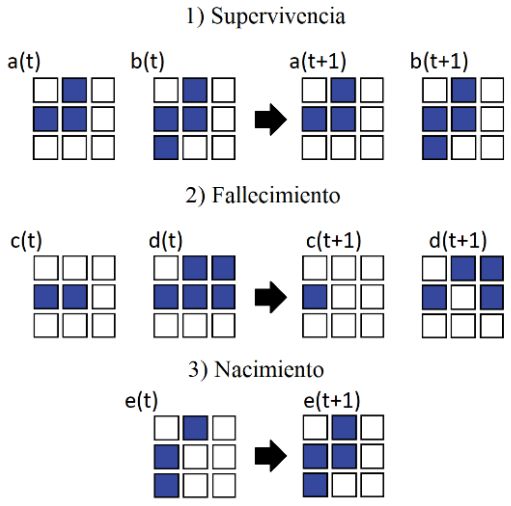


Figura 1: Reglas de evolución de LIFE.

Además del estudio comparativo entre los métodos seleccionados destaca uno en particular y de publicación reciente, inspirado de igual manera en otro autómata celular distinto a LIFE, el cual se denomina Algoritmo de Autómatas Celulares de Estado Continuo (CCAA), (Seck-Tuoh-Mora, Hernández-Romero, Lagos-Eulogio, Medina-Marin y Zúñiga-Peña, 2021), esto para demostrar que las capacidades de los autómatas celulares resultan útiles en el diseño de métodos metaheurísticos de optimización. A continuación, se describe un poco del método CCAA para tenerlo en consideración al leer la propuesta Algoritmo de Optimización Life (AOL).

### *2.1. Características del algoritmo CCAA*

La esencia del CCAA es emular el comportamiento dinámico de un autómata celular. Cada solución (o celular) es una celda del autómata, cuyo estado se define mediante diferentes reglas de evolución seleccionadas al azar de un conjunto de reglas disponibles, donde cada regla produce una nueva solución. Las distintas reglas de evolución generan soluciones nuevas y distintas, de las cuales se selecciona la de menor coste o la mejor del momento para sustituir a la célula actual. Así, toda la población mejora sus posiciones a medida que el sistema evoluciona en cada iteración del proceso de optimización.

Hay reglas de evolución que necesitan otro vecino para generar una nueva solución; este vecino es otra célula que también se elige aleatoriamente. Otras reglas sólo necesitan el mejor coste obtenido hasta el momento para modificar los valores de las soluciones obtenidas, mientras que otras reglas requieren únicamente la información que contiene ella misma. En conjunto, las reglas permiten que cada célula tenga acceso a la información de todas las demás células para generar un vecindario de nuevas soluciones. De esta manera se genera un cambio grande o pequeño en la posición de la nueva célula en cada vecino.

Este algoritmo propone 15 reglas de evolución que se aplican sobre las células en la población inicial y sus costos en la función de prueba (o función objetivo). La estrategia general consiste en generar un vecindario por cada célula con sus vecinos correspondientes y, de ese vecindario, tomar la mejor solución como nuevas células de una forma probabilística o si mejora el costo de la célula original. Los diferentes parámetros de las reglas se enfocan en que una célula pueda alejarse o hacer pequeños cambios de su posición actual o acercarse o alejarse de otra célula. Así, el criterio de selección de los parámetros CCAA más adecuados es equilibrar los cambios generados por las reglas en las células para obtener un proceso de optimización más eficiente (Seck-Tuoh-Mora, Hernández-Romero, Lagos-Eulogio, Medina-Marin y Zúñiga-Peña, 2021).

## 3. Desarrollo del algoritmo AOL

Este trabajo propone la adaptación de las reglas de evolución del autómata celular para resolver problemas continuos con valores de vectores reales en lugar de forma binaria. Con esto se reduce drásticamente el cálculo computacional del algoritmo. Para esto se tomaron las reglas del autómata celular original (LIFE) como inspiración para crear un algoritmo que sea capaz de generar un comportamiento adecuado en el proceso de búsqueda de soluciones, tanto en la fase de exploración como de explotación.

### *3.1. Reglas propuestas de AOL*

A continuación, se describe la analogía que se ocupó para el desarrollo de las reglas propuestas en el diseño del algoritmo de optimización LIFE.
Principalmente se toma las reglas del autómata celular LIFE para desarrollar las reglas o ecuaciones de AOL con una cierta similitud. Para esto hay que definir los elementos que tendrá AOL. Es decir, el espacio, el conjunto de estados, la configuración inicial, los vecinos y las reglas de transición.

Para el espacio se conforma del número total de células o posibles soluciones del algoritmo.

Para definir el estado del algoritmo metaheurístico se utiliza la equivalencia mostrada en la Figura 2, del lado izquierdo se observa el autómata celular original, se representa mediante estados binarios es decir ceros y unos que corresponden al estado muerto y vivo respectivamente. Por el otro lado, del lado derecho se observa la representación de AOL, para poder utilizar valores de numero reales en esta implementación, se toma como una célula todo el renglón $\bar{X}_1$ que contiene el conjunto de valores reales 0.5, 0.1, 0.8 etc. como se observa en la Figura 2.

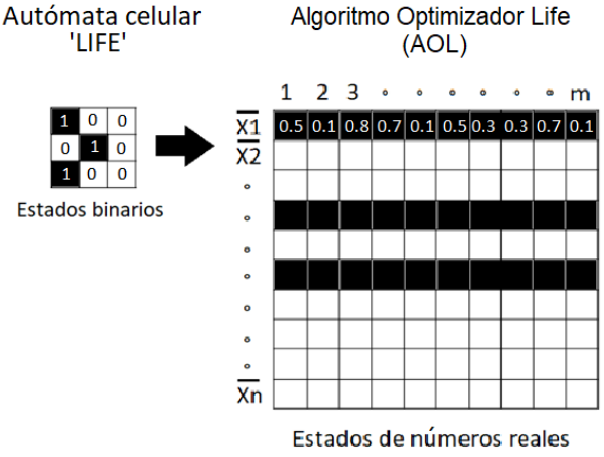


Figura 2: Analogía del estado propuesto para el algoritmo.

Así, el espacio de soluciones estará formado de $\bar{X}_n$ células, mientras que los mejores resultados (o células) hasta el momento serán llamados “valores menores principales”, son representadas en la Figura 2, por las casillas marcadas en negro y corresponden al estado vivo o el valor de 1 en el estado binario original. A los peores resultados o casillas en blanco les corresponde el estado de muerto o de 0 en binario. Así quedan definidos los 2 estados del nuevo algoritmo metaheurístico LIFE.

Para las reglas de transición se tomaron las condiciones necesarias de las reglas originales del autómata celular con unos pequeños cambios, es decir en la Regla 1 (de supervivencia), en lugar de tener 2 o 3 células vivas para que el estado o la célula sobreviva solo se necesitan 2 y en la Regla 2 (de nacimiento) no es necesario que la célula o estado en turno sea necesariamente uno de los peores resultados o estar ‘muerto’ para aplicar la regla. La Regla 3 (de fallecimiento) no tuvo cambios significativos en las condiciones. Así del siguiente modo quedan las reglas correspondientes:

*Regla 1: Si existen 2 valores menores principales (o 2 células vivas) esta célula sobrevivirá al siguiente estado* y mutará de la siguiente manera (ecuación 1). Donde $C_i$ es la célula actual elegida al azar, $C_{best}$ es la mejor célula hasta el momento,

$P_1$ es un factor de ponderación, este se encontró mediante la experimentación del algoritmo al observar en que proporción mejoraba el desempeño del algoritmo. Así esta regla calcula la diferencia entre las células actual y la mejor del momento se multiplica por una ponderación y se le suma a la misma célula actual permitiendo explorar el espacio de búsqueda de una forma más precisa, es decir la fase de explotación (ecuación 1).

$$C_i = [(C_i - C_{best}) * P_1] + C_i \qquad (1)$$

*Regla 2: Si existen exactamente 3 valores menores. esta célula sobrevivirá al siguiente estado* y mutará de la siguiente manera (ecuación 2). De igual forma al ser un promedio, esta regla se enfoca en explorar la fase de explotación del algoritmo propuesto en cambio con la regla anterior se toma en cuenta el promedio de los tres mejores valores hasta el momento indicados por $C_{best1}, C_{best2}, C_{best3}$ y el valor $P_2$ es un factor de ponderación obtenido de igual forma que $P_1$, experimentalmente.

$$C_i = \left[\left(\frac{C_{best1}+C_{best2}+C_{best3}}{3}\right) * P_2\right] + C_i \qquad (2)$$

*Regla 3: Si existen menos de 2 o más de 3 valores menores principales*. Esta célula muere y mutara al tomar la media de los vecinos restantes excluyendo las mejores células del momento y multiplicándolo por un factor de ponderación $P_3$ como se muestra en la (ecuación 3). Esta regla aborda más la fase de exploración del algoritmo. Cabe mencionar que todas las reglas se ven afectadas por los valores $P_1$, $P_2$ y $P_3$, permitiendo que las reglas anteriores, aunque se basan más en la fase de explotación, también sean capaces de abordar la fase de exploración (ecuación 3). En los experimentos realizados se encontró que manteniendo un valor de $P_1$ bajo respecto a $P_2$ y $P_3$, ayudaba al desempeño del algoritmo y decidió tomarse los siguientes valores $P_1 = 0.2$, $P_2 = 0.6$ y $P_3 = 0.9$.

$$C_i = \left[\left(\frac{C_{rest}}{N}\right) * P_3\right] \qquad (3)$$

De esta manera quedan definidas las nuevas reglas del algoritmo diseñado.

Adicionalmente, se utilizó un factor de peso inercial, el cual surgió como una mejora en algoritmos de ooptimización por enjambres de partículas (PSO, por sus siglas en inglés), (Shi y Eberhart, 1999). Al introducir este factor se mejora el desempeño del algoritmo modificando en cada iteración su comportamiento de acuerdo con la velocidad de convergencia de este. Es decir, se evita el estancamiento en soluciones locales y se garantiza la búsqueda global de soluciones. Este peso inercial "$w$"queda definido de la siguiente manera (ecuación 4).

$$w = \frac{(ciclo_{max} - ciclo) * (w_{inicial} - w_{final})}{ciclo_{max}} + w_{final} \qquad (4)$$

La ecuación 4 utiliza valores limite $w_{inicial}$ y $w_{final}$ los cuales se utilizan para reducir linealmente "$w$". Se tomaron los valores de 0.9 y 0.4 respectivamente, y para el $ciclo_{max}$ que corresponde al número total de iteraciones, se consideraron 500.

### *3.2. Estructura del algoritmo*

La lógica del algoritmo queda representada en el diagrama de flujo de la Figura 3.

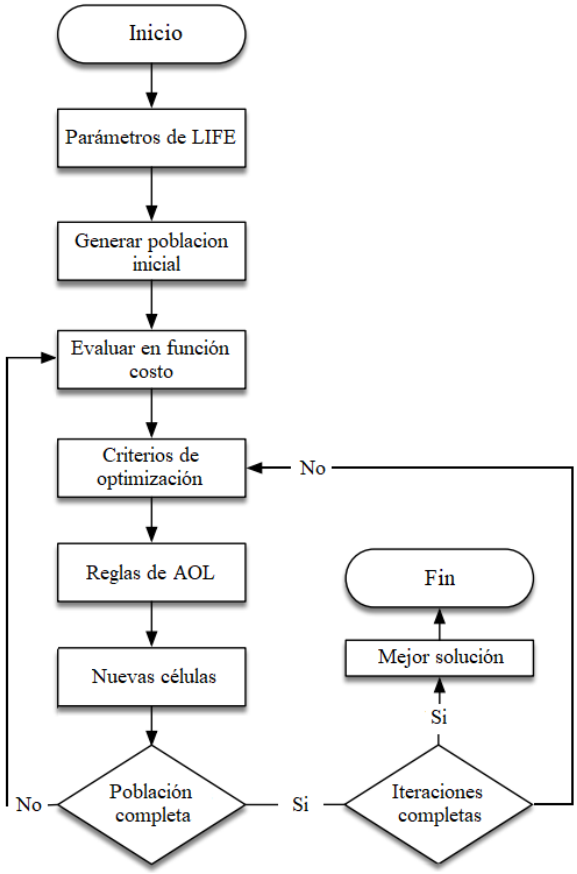


Figura 3: Diagrama de flujo del algoritmo.

Como primer paso se requiere fijar los parámetros iniciales, es decir, el tamaño del espacio de soluciones, número de células, número de vecinos, cantidad de iteraciones, ganancias ocupadas por las reglas y el factor de convergencia. Así, es posible generar una población inicial aleatoria y de dimensiones acordes.

La población inicial serán las soluciones propuestas iniciales $\bar{X}_n$, las cuales serán evaluadas en las funciones de prueba con el fin de ponderarlas y clasificarlas bajo nuestro criterio de estado vivo o muerto. Posteriormente, se seleccionan y mutan para obtener una nueva población mediante las reglas propuestas durante un numero prefijado de iteraciones. Con lo anterior, se obtiene un valor adecuado para la función de prueba utilizada. Es importante mencionar que en algunos casos este valor podría ser el mejor resultado posible; es decir, el óptimo de la función.

A continuación, se describen el procedimiento, las funciones de prueba, los parámetros utilizados y posteriormente los principales resultados de las pruebas realizadas.

## 4. Simulación experimental del algoritmo

La plataforma MATLAB® es una herramienta eficaz y de uso muy común en distintos centros de educación e investigación para enseñar una gran variedad de conceptos. Debido a esto se decidió realizar en esta plataforma la implementación del algoritmo diseñado en este trabajo. A continuación, se describen las especificaciones técnicas del software y equipo utilizado: MATLAB® (versión 2022b); procesador Intel Xeon CPU a 3.1 GHz, 64 GB en RAM con sistema operativo macOS Big Sur.

### *4.1 Funciones de prueba CEC2022*

Dentro de las funciones de prueba seleccionadas se encuentran: funciones básicas, funciones unimodales, funciones híbridas y funciones de composición. La función unimodal se ocupa principalmente para probar la capacidad de explotación del algoritmo. Las funciones básicas son una mezcla enfocada en probar ambas etapas de búsqueda. Las funciones híbridas y de composición se centran en la capacidad de escapar de soluciones locales, evitar el estancamiento, además de poder contemplar varios mínimos locales y probar el buen equilibrio de las reglas. Todas las funciones mencionadas se definen a continuación (Biedrzycki, Arabas y Warchulski, 2022).

Función 1. Función de Zakharov

$$f_1(x) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^4$$

Función 2. Función de Rosenbrock

$$f_2(x) = \sum_{i=1}^{D-1} \left(100\left(x_i^2 - x_{i+1}\right)^2 + (X_{I+1}+\} - 1)^2\right)$$

Función 3. Función de Schaffer ampliada

$$f_3(x) = g(x_1, x_2) + g(x_2, x_3) + \cdots + g(x_{D-1}, x_D) + g(x_D, x_1)$$

Función 4. Función de Rastrigin

$$f_4(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi X_I) + 10\right)$$

Función 5. Función de Levy

$$f_5(x) = \sin^2(\pi w_1) + \sum_{i=1}^{D-1} (w_1 - 1)^2\,[1 + 10\sin^2(\pi w_1 - 1)] + (w_D - 1)^2[1 + \sin^2(2\pi w_D)]$$

Función 6. Función de Bent Cigar

$$f_6(x) = x_i^2 + 10^{6\sum_{i=2}^{D} x_i^2}$$

Función 7. Función HGBat

$$f_7(x) = \left|\left(\sum_{i=1}^{D} x_i^2\right)^2 - \left(\sum_{i=1}^{D} x_i\right)^2\right|^{0.5} + \left(0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i\right)\Big/ D + 0.5$$

Función 8. Función elíptica altamente condicionada

$$f_8(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$$

A continuación, se presentan y analizan los resultados pertinentes a las funciones anteriores.

### *4.2 Simulación y experimentación*

Las primeras simulaciones realizadas consistieron en probar el algoritmo de optimización LIFE en cada una de las 12 funciones de prueba tomadas de (Biedrzycki, Arabas y Warchulski 2022), AOL se contempló bajo siguientes criterios.

Número de funciones = 12;
Número de células = 20;
Número de vecinos = 6;
Numero de iteraciones =500.
Ganancias $P_1 = 0.2$, $P_2 = 0.6$, $P_3 = 0.9$.

Las segundas simulaciones realizadas se centraron en el estudio comparativo entre el desempeño del algoritmo de optimización LIFE y tres algoritmos publicados recientemente y son reconocidos por su eficacia en la optimización de problemas unimodales y multimodales en múltiples dimensiones. Por estas razones, representan un conjunto adecuado de metaheurísticas para analizar y ponderar AOL frente a los algoritmos más avanzados.

Los algoritmos seleccionados son la optimización del halcón de Harris (HHO), por su amplia utilización en problemas de optimizaciónen los últimos años (Ali Asghar, Seyedali Mirjalili, Hossam Faris, Ibrahim Aljarah, Majdi Mafarja, and Huiling Chen, 2019), la atracción por superposición ponderada (WSA), por su análisis de superposición para resolver problemas con varios criterios (Adil Baykasoglu and Sener Akpinar, 2017) y el Algoritmo de autómatas celulares de estado continuo (CCAA) descrito en la sección 2.1. principalmente por su similitud en la inspiración del algoritmo y el número de reglas que propone. Los ajustes de los parámetros de estos algoritmos se han extraído de sus artículos originales y se muestran en la Tabla 1.

Las implementaciones originales en Matlab de estos algoritmos se tomaron directamente de las direcciones web indicadas en los artículos de referencia.

Tabla 1: Configuración de los parámetros de los algoritmos empleados para la comparación con AOL

| Algoritmo | Parámetros |
|---|---|
| HHO | $E_0 = 2 * rand - 1, J = 2 * (1 - rand)$ |
| WSA | $\tau = 0.8\ sl_0 = 0.035, \lambda = 0.75, \varphi = 0.001$ |
| CCAA | SmartCells_no=20; Neighbors_no=6; Max_iteration=500; |

Al ser parecidos en implementación debido a su inspiración en autómatas celulares los parámetros del AOL y CCAA fueron idénticos en cuestión de las 'poblaciones' de células es decir el número de células y vecino. El número de iteraciones y de corridas por algoritmo para obtener los datos correspondientes fueron 500 iteraciones y 30 corridas para cada algoritmo.

## 5. Resultados y discusiones

Antes de iniciar con el análisis de los resultados es importante mencionar que, para que un algoritmo metaheurístico tenga un comportamiento adecuado, cada evolución en las soluciones debe tener cambios abruptos de posición en las etapas iniciales del proceso de optimización, indicando un adecuado proceso de exploración. Estos cambios deben ir reduciéndose conforme el proceso avanza y llegar a su fase final para enfocarse en la explotación de las regiones ya detectadas como cercanas a los valores óptimos (Van den Bergh y EngelBrecht, 2006). Esta dinámica puede garantizar que la metaheurística converja en una posición del espacio de búsqueda.

Se comenzó probando el Algoritmo Optimizador Life AOL en las funciones de prueba para observar que tan bien se acercaba al valor deseado de estas, e ir calibrado las ganancias $P_1$, $P_2$ y $P_3$, Además se analizó qué reglas tenían un mayor efecto en las iteraciones del algoritmo, para confirmar un correcto equilibrio entre las fases de exploración y explotación. Se encontró que la regla 2 se ve aplicada un mayor número de veces; es decir, la explotación es mayor. Sin embargo, aunque las demás reglas se ven menos afectadas, son capaces de cambiar los lugares de explotación de la regla 2, por lo que aportan una gran ayuda a la etapa de exploración, ya que además del peso inercial que regula de una mejor manera la velocidad de convergencia, le permite una evolución más suave a la solución obtenida en la etapa final de búsqueda.

Una vez sintonizado el algoritmo se comparó en rendimiento con cada uno de los tres algoritmos mencionados para cada función de prueba. En la Tabla 2 se presentan los resultados de la prueba estadística de suma de rangos de Wilcoxon, entre la comparación de AOL y HHO, WSA y CCAA. Donde el símbolo + representa un mejor resultado que es estadísticamente significativo, el símbolo aprox ≈ indica que no hay diferencia significativa, y - indica un peor resultado estadísticamente significativo. La tercera columna muestra el lugar obtenido por el número de funciones es decir LIFE obtuvo 5 funciones con el primeros puesto 3 funciones en el segundo puesto y 4 en el tercer puesto. La columna Avg muestra la clasificación media de cada algoritmo obtenida mediante la optimización de las funciones de prueba, y la columna Rank muestra el orden de clasificación de cada algoritmo según su media.

Tabla 2: Prueba de suma de rangos de Wilcoxon y clasificación de los algoritmos comparados.

| Algoritmo | +/ - / ≈ | No:1/ 2 / 3 /4 | Avg | Rank |
|---|---|---|---|---|
| AOL | - / - / - | 5 / 5 / 2 / 0 | 1.75 | 1 |
| CCAA | 7 / 3/ 2 | 3 / 4 / 5 / 0 | 2.16 | 3 |
| HHO | 6 / 5 / 1 | 4 / 3 / 5 / 0 | 2.08 | 2 |
| WSA | 12 / 0 / 0 | 0 / 0 / 0 /12 | 4 | 4 |

AOL obtuvo la primera clasificación, resaltando que superó a CCAA en 7 funciones de prueba a HHO en 6 funciones y a WSA en las 12 funciones, obteniendo este último el cuarto lugar en la Tabla 3. Por el otro lado HHO supero a AOL en 4 funciones, CCAA supero a AOL en 3 funciones y resultando en igualdad de condiciones HHO y AOL con una función y CCAA y AOL con 2 funciones.

En la Tabla 3, se calcula el valor medio y la desviación típica, los mejores resultados se muestran en negrita con fondo naranja. AOL obtuvo 5 de los mejores valores medios para las funciones, por debajo de este se encuentra HHO con 4 valores medios y CCAA con 3, WSA obtuvo el último lugar con ninguno. Por el lado de valores mínimos para la desviación estándar, AOL obtuvo 6 de los mejores valores, HHO y CCAA obtuvieron 3, cabe señalar que estos dos últimos algoritmos resultaron muy parejos en su rendimiento.

De acuerdo con las pruebas estadísticas realizadas es evidente que AOL obtuvo los mejores resultados lo que demuestra su mejor desempeño y una mejor obtención de resultados adecuados, mostrando un equilibrio aceptable entre las acciones de exploración y explotación en este tipo de problemas. En las funciones donde no obtuvo un buen resultado (Función 1 y 6) el algoritmo AOL se propone mejorar modificando algunas ganancias para que sean capaces de evolucionar conforme el algoritmo trabaje, es decir que no sean fijas y tengan un comportamiento más dinámico de acuerdo al tipo de función utilizada.

En comparación con el algoritmo CCAA el cual comparte la misma inspiración que AOL es importante hacer notar que las reglas de LIFE son menos de una tercera parte en comparación con las 15 reglas del algoritmo CCCA. Así se puede concluir que se obtuvo un algoritmo similar en rendimiento al ya publicado (CCCA); principalmente esto valida y refuerza la idea de utilizar la gran variedad de reglas que existen en los autómatas celulares para el diseño, desarrollo e implementación en nuevos métodos metaheurísticos que sean cada vez más eficientes, exactos y fáciles de implementar en la gran mayoría de problemas que existen. En este caso particular se logró reducir las reglas de implementación, lo que significa un resultado adecuado con un menor gasto computacional y una forma más sencilla de solucionar problemas de ingeniería. De esta forma, así como existe una amplia investigación en técnicas metaheurísticas inspiradas en otras ramas de investigación (Yang, 2010), se ha dado a conocer la importancia de la investigación inspirado en autómatas celulares.

Tabla 3: Metaheurísticas comparadas con AOL en las 12 funciones de prueba CEC2022.

| Función | Estadísticas | AOL | HHO | WSA | CCAA |
|---|---|---|---|---|---|
| 1 | AVG | 8161.71 | **2334.97** | 39578.36 | 10462.43 |
| | STD | 2260.62 | **1150.88** | 7996.65 | 4711.17 |
| 2 | AVG | 505.18 | **494.43** | 2969.21 | 563.45 |
| | STD | **32.1** | 50.65 | 462.72 | 64.83 |
| 3 | AVG | **612.03** | 654.73 | 681.5 | 641.03 |
| | STD | **2.98** | 7.76 | 5.42 | 13.16 |
| 4 | AVG | 922.23 | **875.6** | 975.86 | 884.7 |
| | STD | **22.13** | **11.14** | 10.49 | 11.41 |
| 5 | AVG | **1639.8** | 2680.31 | 3365.17 | 2345.46 |
| | STD | 546.31 | 264.53 | 259.9 | **239.83** |
| 6 | AVG | 353053.68 | 67484.05 | 100000000 | **4587.91** |
| | STD | 207275.18 | 32040.44 | 0 | **2640.26** |
| 7 | AVG | 2094.38 | 2162.28 | 2212.09 | **2092.86** |
| | STD | **15.89** | 53.62 | 37.79 | 35.28 |
| 8 | AVG | 2258.49 | 2271.88 | 2584.14 | **2229.15** |
| | STD | 32.73 | 68.88 | 196.09 | **2.15** |
| 9 | AVG | **2489.05** | 2496.97 | 3375.66 | 2532.36 |
| | STD | **4.27** | 9.35 | 255.19 | 21.3 |
| 10 | AVG | **3213** | 3643.24 | 5690.29 | 3442.7 |
| | STD | 1371.19 | **744.76** | 1362.55 | 758.31 |
| 11 | AVG | 3234.37 | **2985.52** | 8737.89 | 3915.7 |
| | STD | **143.09** | 175.52 | 607.02 | 685.2 |
| 12 | AVG | **3051.77** | 3141.61 | 3686.07 | 3055.07 |
| | STD | **32.07** | 82.42 | 194.28 | 72.31 |

## 6. Conclusiones

En este trabajo se presenta el nuevo Algoritmo de Optimización Life. Este fue inspirado en las reglas de evolución del juego de la vida, debido a su capacidad para producir comportamientos complejos mediante las 3 reglas desarrolladas e implementadas sin la necesidad de un control central.

Los experimentos consistieron en poner a prueba la nueva metaheurística con 12 funciones de prueba (CEC 2022), siendo estas unimodales, básicas, hibridas y de composición lo que nos permite tener una idea de que tan bien puede funcionar la propuesta AOL en distintas circunstancias, para esto, lograr la sintonización del algoritmo fue la pieza relevante ya que se obtuvo mediante la variación en las ganancias $P_1$, $P_2$ y $P_3$. Así observando el desempeño del algoritmo, se logró saber que la que la Regla 1 necesitaba atenuar su comportamiento con una reducción en su ganancia, en comparación a las reglas restantes. Y en conjunto con el factor de convergencia (peso inercial), la evolución de las células mejoro en función de la calidad de las soluciones.

Las pruebas estadísticas nos permitieron realizar una comparación entre el desempeño de cada algoritmo respecto a la propuesta AOL en cada función de prueba, destacando entre los resultados que supero en 7 funciones a CCAA, en 6 funciones a HHO y en las 12 funciones a WSA. Por el lado de la medición de la desviación estándar obtuvo los mejores resultados en función de la minimización del error deseado, superando a él algoritmo HHO uno de los más reconocidos y bastamente empleado en problemas de optimización por la comunidad científica, y en comparación con CCAA logro superarlo con una menor cantidad de reglas, lo que demuestra que las 3 reglas de AOL presentan una buena aleatoriedad y un buen intercambio de información en cada iteración lo que produce un equilibrio apropiado entre las acciones de exploración y explotación en el proceso de búsqueda de soluciones en las funciones utilizadas.
Se concluye que la propuesta AOL fue capaz de competir y superar a las metaheurísticas seleccionadas y reconocidas en la literatura con una implementación mucho más rápida y simple.

Como trabajos futuros se propone utilizar este algoritmo propuesto en problemas de ingeniería y compararlos con los ya existentes en la literatura. Además, es deseable continuar tomando inspiración de otros comportamientos dinámicos de autómatas celulares para establecer nuevos algoritmos metaheurísticos que sean capaces de establecer estrategias multiobjetivo que aborden un mayor número de problemáticas.

## Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACYT), por la beca otorgada para la realización del Doctorado en Ciencias en Ingeniería, con Énfasis en Análisis y Modelación de Sistemas y a los catedráticos del Área Académica de Ingeniería y Arquitectura de la Universidad Autónoma del Estado de Hidalgo.

## Referencias

Adamatzky, A. (Ed.). (2010). Game of life cellular automata (Vol. 1). London: Springer.

Adil Baykasoglu and ¸Sener Akpinar. (2017). Weighted superposition attraction ˇ (wsa): A swarm intelligence algorithm for optimization problems–part 1: Unconstrained optimization. Applied Soft Computing, 56:520–540.

Ali Asghar Heidari, Seyedali Mirjalili, Hossam Faris, Ibrahim Aljarah, Majdi Mafarja, and Huiling Chen. (2019). Harris hawks optimization: Algorithm and applications. Future generation computer systems, 97:849–872.

Bansal, J. C., Singh, P. K., Pal, N. R. (Eds.). (2019). Evolutionary and swarm intelligence algorithms (Vol. 779). Cham: Springer.

Biedrzycki, R., Arabas, J., Warchulski, E. (2022, July). A version of nl-shade-rsp algorithm with midpoint for cec 2022 single objective bound constrained problems. In 2022 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-8). IEEE.

Bilan, S. M., Bilan, M. M., & Motornyuk, R. L. (Eds.). (2020). New methods and paradigms for modeling dynamic processes based on cellular automata. United States America byIGI Global.

Brú, M. V. M., Fleites, G. L., & Barroso, E. M. (2004). Una comparación de dos métodos de gradiente en el Escalamiento Multidimensional. Ciencias Matemáticas, 22(1): 44-53.

Conway, J. (1970). The game of life. Scientific American, 223(4): 4.

Eppstein, D. (2010). Growth and decay in life-like cellular automata.Game of Life cellular automata, pp. 71-97.

Escamilla Serna, N. J. Seck-Tuoh-Mora, J. C. Medina-Marin, J. Hernandez-Romero, N. Barragan-Vite, I. y Corona Armenta, J. R. (2021). A global-local neighborhood search algo- rithm and tabu search for flexible job shop scheduling problem. PeerJ Computer Science, 7:e574.

Garcia, J. B. C., Gomes, M. A. F., Jyh, T. I., Ren, T. I., Sales, T. R. M. (1993). Nonlinear dynamics of the cellular-automaton ‘‘game of Life’’.Physical Review E, 48(5): 3345.

Gogna, A., Tayal, A. (2013). Metaheuristics: review and application. Journal of Experimental & Theoretical Artificial Intelligence, 25(4): 503-526.

Hassanien, A. E., Emary, E. (2018). Swarm intelligence: principles, advances, and applications. CRC Press.

Hernández-Gress, E. S. Seck-Tuoh-Mora, J. C. Hernández-Romero, N. Medina-Marín, J. Lagos-Eulogio, P. y Ortíz-Perea, J. (2020). The solution of the concurrent layout scheduling problem in the job-shop environment through a local neighborhood search algorithm. Expert Systems with Applications, 144:113096.

Kotyrba, M., Volna, E., Bujok, P. (2015). Unconventional modelling of complex system via cellular automata and differential evolution. Swarm and Evolutionary Computation, 25: 52-62.

Kumar, K., Davim, P. J. (Eds.). (2020). Modern Manufacturing Processes. Woodhead Publishing.

Kumar, K. y Davim, J. P. (2020). Optimization Using Evolutionary Algorithms and Metaheuristics: Applications in Engineering. CRC Press, Boca Raton, FL, USA, 1 edition.

Lagos-Eulogio, P. Seck-Tuoh-Mora, J. C. Hernandez- Romero, N. y Medina-Marin,J. (2017).A new design method for adaptive iir system identification using hybrid cpso and de. Nonlinear Dynamics, 88(4):2371–2389.

Li, X., Wu, J., Li, X. (2018). Theory of practical cellular automaton. Berlin/Heidelberg, Germany: Springer.

Lozano, D., Velázquez, F., Zepeda, A. (2010). Optimización estructural de forma en el diseño de cavidades en elementos planos mediante algoritmos evolutivos. Mecánica Computacional, 29(12): 1143-1159.

McIntosh, H. V. (2009). One dimensional cellular automata. Luniver Press.

Momin Jamil and Xin-She Yang, (2013). A literature survey of benchmark functions for global optimisation problems. International Journal of Math- ematical Modelling and Numerical Optimisation, 4(2):150–194.

Ramos, A., Sánchez, P., Ferrer, J. M., Barquín, J., Linares, P. (2010). Modelos matemáticos de optimización. Publicación Técnica, 1.

Sarker, R. Kamruzzaman, J. y Newton Charles. (2003).Evolutionary optimization (evopt): a brief review and analysis. International Journal of Computational Intelligence and Applications, 3(4):311–330.

Salcido, A. (Ed.). (2011). Cellular automata: Innovative modelling for science and engineering. John Wiley & Sons. pp. 55-60.

Seck-Tuoh-Mora, J.C. Medina-Marin, J. Martinez- Gomez, E.S. Hernandez-Gress, E.S. Hernandez-Romero, N. y Volpi-Leon, V. (2019). Cellular particle swarm optimization with a simple adaptive local search strategy for the permutation flow shop scheduling problem. Archives of Control Sciences, 29(2):205–226.

Seck-Tuoh-Mora, J. C., Hernandez-Romero, N., Lagos-Eulogio, P., Medina-Marin, J., Zuñiga-Peña, N. S. (2021). A continuous-state cellular automata algorithm for global optimization. Expert Systems with Applications, 177: 114930.

Seck-Tuoh-Mora, J. C. Hernandez-Romero, N. Santander-Baños, F. Volpi-Leon, V. Medina-Marin, J. and Lagos-Eulogio, P. (2022). A majority–minority cellular automata algorithm for global optimization. Expert Systems with Applications, 203:117379.

Shi, Y., Eberhart, R. C. (1999, July). Empirical study of particle swarm optimization. In Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406) (Vol. 3, pp. 1945-1950). IEEE.

Shi, Y., Liu, H. Gao Liang, and Zhang, G. (2011). Cellular particle swarm optimization. Information Sciences, 181(20):4460–4493..

Valencia, P. E. (1997, August). Optimización mediante algoritmos genéticos. In Anales del Instituto de Ingenieros de Chile, 109 (2): 83-92.

Van den Bergh, F., Engelbrecht, A. P. (2006). A study of particle swarm optimization particle trajectories. Information sciences, 176(8): 937-971.

Van den Bergh, F., & Engelbrecht, A. P. (2010). A convergence proof for the particle swarm optimiser. Fundamenta Informaticae, Vol. 105 (4), 341-374.

Wolpert, D. H., Macready, W. G. (1997). No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1): 67-82.

Yang, X. S. (2010). Engineering optimization: an introduction with metaheuristic applications. John Wiley & Sons.