ISSN 1870-4069

Variante eficiente de red neuronal pulsante con auto-onda dinámica para resolver el problema del camino más corto

Oliver Espinosa, Manuel Mejía-Lavalle, José Ruiz, Gerardo Reyes, Alicia Martínez

Tecnológico Nacional de México / CENIDET,
Cuernavaca, Morelos, México
{oliver.espinosa18ce, mlavalle,
josera,greyes,amartinez}@cenidet.edu.mx

Resumen. Desde hace aproximadamente 10 años se ha estudiado el comportamiento de las Redes Neuronales Artificiales Pulsantes en el campo de la Optimización de Trayectorias, específicamente en el problema de la ruta más corta. Sin embargo, existen modelos de Redes Neuronales Pulsantes que necesitan un gran número de iteraciones antes de encontrar la ruta más corta entre un punto y otro. En este artículo se presenta una variante de Red Neuronal Pulsante para resolver el problema de la ruta más corta de manera eficiente. Esta variante cuenta con una velocidad dinámica de propagación de auto-onda, la cual se ajusta de manera heurística para evitar iteraciones donde no hay cambios en la Red. Para mostrar la eficiencia del modelo, se realizan experimentos y se comparan los resultados contra los obtenidos por otros modelos de Redes Neuronales Pulsantes. Para la comparativa se utilizan paradigmas que emplean una velocidad de auto-onda estática y modelos con una velocidad de auto-onda dinámica.

Palabras clave: red neuronal pulsante, problema del camino más corto, optimización, auto-onda dinámica.

Efficient Variation of Pulsed Neural Network with Dynamic Auto-Wave to Solve the Shortest Path Problem

Abstract. For about 10 years the behavior of Pulsed Neural Network has been studied in the optimization field of Shortest Path Problem. However, there are models of Pulse Coupled Neural Network that need a large number of iterations before to find the shortest path between two points. This paper presents a variation of Pulse Coupled Neural Network to solve the shortest path problem in an efficient way. This variant has a dynamic auto-wave propagation speed, which adjusts in a heuristic way to avoid iterations where there are no changes in the network. To show the efficiency of the model, experiments are performed and the results are compared against two other models of Pulsating Neural Networks. In the comparison, paradigms that use a static auto-wave speed and models with a dynamic auto-wave speed are used.

Keywords: pulsed neural network, shortest path problem, optimization, dynamic auto-wave.

1. Introducción

El campo de optimización de trayectorias tiene como propósito encontrar la mejor trayectoria que le permita resolver una tarea específica. Uno de los problemas más estudiados de la optimización de trayectorias es el problema del camino más corto o *Shortest Path* (SP, por sus siglas en inglés). El objetivo de resolver el problema del camino más corto es encontrar el camino de menor costo entre una fuente dada y un destino en una red determinada, es decir, encontrar la ruta más corta en un grafo de un vértice a otro. Aquí, el "más corto" significa que el costo total que se correlaciona con la ruta debe ser el menor [1].

La teoría de grafos siempre se utiliza como la base matemática para la solución de este tipo de problemas. En un grafo $G = (V, E)$, donde la distancia entre los vértices v_i y v_j es w_{ij}, se requiere encontrar el camino más corto entre un vértice de inicio s y un vértice meta g. En grafos con pocos nodos la solución podría resultar sencilla, pero con grafos de mayor magnitud, la complejidad computacional se vuelve alta. Los algoritmos tradicionales generalmente tienen complejidad exponencial, siendo un problema NP.

Existen diversas soluciones algorítmicas para resolver el problema del camino más corto. Desde las soluciones exhaustivas como la Búsqueda en anchura o la Búsqueda en Profundidad, hasta las soluciones con mayor eficiencia como el *Branch & Bound* [2], el algoritmo A* [3], Dijkstra [4] y el algoritmo Bellman-Ford [5]. El problema del camino más corto tiene diversas aplicaciones en la robótica, las telecomunicaciones, el transporte, la teoría de juegos, las redes de computadoras, búsquedas *web*, entre otros.

Desde el trabajo original de Hopfield [6] en 1985, se han realizado diversas investigaciones utilizando las Redes Neuronales Artificiales (RNA) para resolver problemas de optimización combinatoria. En el presente artículo se propone utilizar el enfoque de las RNA para resolver el problema SP, en particular de las *Pulse-Coupled Neural Network* (PCNN, por sus siglas en inglés). Las PCNN son redes no supervisadas, forman una red auto organizada que no requiere entrenamiento. Las PCNN son capaces de emular el comportamiento de las neuronas corticales como se observa en las cortezas visuales de los mamíferos [7].

Ha aumentado el interés en el uso de PCNN para diversas aplicaciones, como el reconocimiento de objetivos [8], el procesamiento de imágenes [9], el reconocimiento de patrones [10] y reducir el efecto de ruido gaussiano [11]. Algunas investigaciones recientes muestran que la dinámica espacio temporal de las PCNN proporciona una buena capacidad computacional para resolver algunos problemas de optimización. En 1999, Caulfield y Kinser [12] presentaron la idea de utilizar la onda automática en PCNNs para encontrar la solución al problema del laberinto. Su modelo puede encontrar la ruta más corta rápidamente y con el mínimo esfuerzo, donde la solución está relacionada con la longitud de la ruta más corta e independiente de la complejidad del gráfico de la ruta. Sin embargo, se necesitan muchas neuronas para encontrar la ruta más corta en grandes laberintos o gráficos, ya que una neurona corresponde a una unidad de longitud de ruta. Posteriormente Qu [13] presenta el modelo *Modified Pulse-*

Coupled Neural Network (MPCCN), el cual tiene un costo computacional menor. El número de iteraciones de la MPCNN es proporcional a la longitud del camino más corto. Ma [1] publicaron el modelo *Auto-Wave Neural Network* (AWNN), el cual aprovecha la característica de auto-onda de las PCNN para resolver el problema de SP. Paredes-Cano [14] presentaron una mejora del modelo AWNN además de una técnica para explicitar el conocimiento generado por la red neuronal. Los modelos mencionados anteriormente sugieren que la velocidad de la onda debe mantener una velocidad constante. Posteriormente se presentaría un modelo llamado *Self-adaptive autowave pulse-coupled neural network* (SAPCNN) [15]. Este modelo puede ajustar la velocidad de propagación de la onda automáticamente de acuerdo con el estado actual de la red. Adicionalmente los resultados indican que el modelo logra calcular el camino más corto con menor cantidad de iteraciones.

En este artículo se propone el modelo *Self-adaptive autowave modified pulse-coupled neural network* (SAM-PCNN), el cual es un modelo modificado de PCNN que es capaz de adaptar la velocidad de onda de manera más eficiente que la SAPCNN. Los resultados de la experimentación muestran que el modelo SAM-PCNN tiene mayor eficiencia que los modelos que cuentan con una velocidad constante de onda, además de que también supera al modelo SAPCNN, el cual cuenta con una velocidad dinámica auto-onda.

El articulo se organiza de la siguiente manera. En la Sección 2 se hace una breve revisión de los modelos de PCNN para optimización de trayectorias. En la Sección 3 se hará una descripción del modelo propuesto SAM-PCNN. En la Sección 4 se describen los experimentos y los resultados obtenidos. En la Sección 5 se discuten los resultados y en la sección 6 se concretan las conclusiones y el trabajo futuro.

2. Modelos de PCNN para optimización de trayectorias

2.1. Auto-Wave Neural Network

Se analizaron diferentes modelos de PCNN para su implementación, el primero de ellos es el modelo AWNN (*Auto-Wave Neural Network*, por sus siglas en inglés) presentado en [1]. Dicho modelo está diseñado para resolver el problema de la ruta más corta. Utiliza como entrada la matriz de adyacencias de un grafo ponderado. En la Figura 1 se muestra una neurona de este modelo, el cual está compuesto por la capa de recepción, de enlace y de pulsos. La capa de enlace está formada por la combinación de W y F. La matriz U representa la capa de enlace mientras que la capa de pulsos está compuesta por el umbral E y la salida Y.

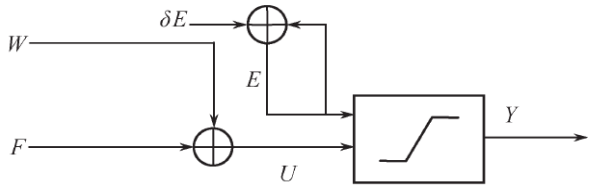

Fig. 1. Modelo de la neurona *Auto-Wave Neural Network* [1].

Cada neurona recibe como entrada información de la matriz de adyacencia W y además recibe un valor F del costo adicional en caso de continuar por cierta ruta: sumando W y F se obtiene el valor U que es el costo total incurrido por algún camino hasta un momento dado determinado por el umbral dinámico T. En cada época el parámetro E se incrementa por un valor constante δE: si el costo acumulado en U es igual o supera al umbral T, entonces la neurona se activa tomando la salida Y el valor de "1", de otra manera la salida de la neurona será cero.

Dado un problema con n nodos, la red neuronal tendrá entonces tantas neuronas como $n \times n$. El efecto de auto-onda se obtiene precisamente al ir realizando incrementos en el tiempo del umbral T. Las siguientes ecuaciones resumen el comportamiento de la neurona AWNN.

$$F_i[n] = F_h[n] + W_{hi}, \tag{1}$$

$$U_{ij}[n] = F_i[n] + W_{ij}, \tag{2}$$

$$Y_{ij} = \begin{cases} 1\ T[n] \geq U_{ij}[n] \\ 0\ de\ otra\ manera \end{cases}, \tag{3}$$

$$T[n] = T[n-1] + \delta E. \tag{4}$$

Se seleccionó el modelo AWNN por que cuenta con un mecanismo sencillo y fácil de comprender. Dicho modelo es un buen candidato para comenzar a entender la manera en que las Redes Neuronales Pulso Acopladas son empleadas en problemas de optimización de trayectorias.

2.2. Auto-Wave Neural Network explicitada

Este modelo de PCNN surge a partir de un artículo realizado en CENIDET [14]. Es una extensión del modelo AWNN presentado en [1]. Para hacer más eficiente el proceso los autores proponen un valor inicial de E como el costo menor del nodo de inicio a sus siguientes nodos. Está modificación permite que desde la primera interacción pulse por lo menos una neurona.

Adicionalmente se propone un método para recuperar la ruta calculada por la red neuronal. La explicitación del conocimiento es un sub-tema dentro de las RNA y se debe a que, en el surgimiento de la tecnología, existía la crítica de que aunque los paradigmas neuronales daban buenos resultados, no era posible acceder al conocimiento que de manera automática el algoritmo había generado. Se decía entonces que las RNA eran algoritmos de caja negra. Sin embargo, con el tiempo, han surgido propuestas exitosas para precisamente transparentar el conocimiento oculto entre las neuronas, de tal manera que el usuario tenga acceso a, por ejemplo, las reglas de producción que se generan de manera automática como resultado de la actividad interna de la RNA. En ese sentido, en [14] se logra rescatar el camino óptimo o de menor costo, extrayendo el conocimiento que queda oculto en la capa de salida Y de la RNP propuesta.

La capa de salida Y es básicamente una matriz de tamaño $n \times n$ con valores de ceros y unos. La ruta se recupera de manera inversa, saltando del nodo final al nodo inicial. En la matriz de salida Y, el nodo final se encontrará en la columna i con el mismo número que el nodo meta. En esa columna se buscará el valor "1" y ahí donde aparezca

se tomará el valor del renglón j. Con ese valor de renglón se entrará nuevamente a la matriz Y pero ahora buscando en la columna con valor j, el valor "1" y repitiendo el proceso hasta llegar al nodo inicial. Al final los valores j así recuperados definirán la ruta de costo mínimo.

Para ilustrar este proceso de explicitación de conocimiento se presenta un ejemplo. En un grafo de cinco nodos se busca la trayectoria más corta del nodo A al nodo D. Una vez que la red neuronal ha calculado la trayectoria que se muestra la Figura 2, la matriz de salida Y es tal cómo se muestra en la Tabla 1.

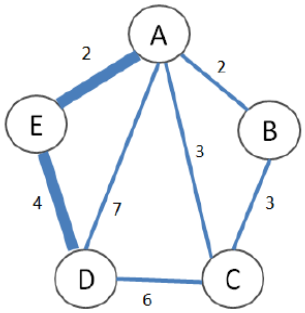

Fig. 2. Grafo del ejemplo que muestra la solución de la RNA [14].

Tabla 1. Matriz de salida del ejemplo [14].

Y	A (i1)	B (i2)	C (i3)	D (i4)	E (i5)
A (j1)	1	1	1	0	1
B (j2)	0	0	0	0	0
C (j3)	0	0	0	0	0
D (j4)	0	0	0	0	0
E (j5)	0	0	0	1	0

El nodo final D se encuentra en $i = 4$ y $j = 5$. Usando el valor de j como referencia, vamos a la columna $i = 5$, correspondiente al nodo E, y en donde el elemento marcado con "1" se encuentra en el renglón $j = 1$. Usando el valor de j una vez más como referencia, vamos a la columna $i = 1$, correspondiente al nodo A y en donde el elemento marcado con "1" se encuentra en el renglón $j = 1$. En este momento se termina la explicitación pues se alcanzó el nodo inicial. De esta manera se recupera el camino de menor costo que en este caso es $A - E - D$.

2.3. Self-Adaptive Autowave Pulse-Coupled Neural Network

El modelo SAPCNN [15] propaga la onda de forma dinámica según el estado de la red. Esto representa una mejora, pues la mayoría de los métodos sugieren que la onda

automática de los modelos PCNN debe mantener una velocidad constante para encontrar los caminos más cortos. En la Figura 3 se muestra el diagrama del modelo.

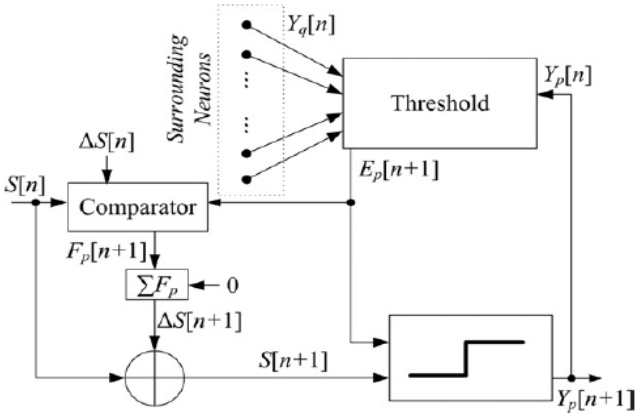

Fig. 2. Diagrama del modelo SAPCNN [15].

En el diagrama se puede ver que las salidas de la matriz de pulsos Y son las entradas de la neurona, las cuales se reciben el umbral $E[n]$. La actividad interna $S[n]$, la cual se incrementa por $\Delta S[n]$ en cada iteración.

$\Delta S[n]$ puede adquirir diferentes valores dependiendo del *comparator* como aparece en el diagrama de F_P que a su vez depende de la relación que existe entre el umbral $E[n+1]$ y $S[n]+\Delta S[n]$. En las ecuaciones se puede encontrar una descripción más detallada del comportamiento de este modelo.

$$Ep[n+1]=\begin{cases} V_E \; if \; Y_P[n]=1 \\ \min\left(S[n]+W_{qp}[n], Ep[n]\right) if \; Y_P[n]=0\,, Y_q[n]=1 \text{ y } p, \in R_q \\ E_p \; de \; otra \; manera \end{cases} \tag{5}$$

$$F_p[n+1]=\begin{cases} N \; if \; E_p[n+1] < S[n]+\Delta S[n] \\ -1 \; if \; E_p[n+1]=S[n]+\Delta S[n] \\ 0 \; de \; otra \; manera \end{cases}, \tag{6}$$

$$\Delta S[n]\begin{cases} \gamma \; if \; F_m > 0 \\ \Delta S[n]+\gamma \; if \quad F_m = 0, \\ \Delta S[n] \; if \; de \; otra \; manera \end{cases} \tag{7}$$

$$S[n+1] < S[n]+\Delta S[n], \tag{8}$$

$$Y_p[n+1]=\begin{cases} 1 \; if \; S[n+1] \geq Ep[n+1] \\ 0 \; de \; otra \; manera \end{cases}. \tag{9}$$

Este modelo está diseñado para resolver el problema del camino más corto, sin embargo, los autores reportan que también puede ser usado para la planeación de trayectorias para robots móviles.

3. Variante propuesta Self-Adaptive Autowave Modified Pulse-Coupled Neural Network (SAM-PCNN)

El objetivo del modelo SAM-PCNN que aquí se propone es resolver el problema del camino más corto o *Shortest Path* (SP, por sus siglas en inglés) en un grafo ponderado no dirigido en la menor cantidad de iteraciones. Cuenta con una velocidad de onda dinámica que puede adaptarse dependiendo del estado de la red. Sin embargo, a diferencia de la SAPCNN [15], el modelo SAM-PCNN logra adaptar la velocidad de onda de manera exacta en cada iteración, logrando eliminar las iteraciones donde no hay cambios en la Red.

El diagrama del modelo propuesto se presenta en la Figura 4. Se puede apreciar que la neurona se alimenta de las salidas del resto de las neuronas y se retroalimenta de su propia salida para calcular el umbral. Posteriormente se calcula la velocidad dinámica de la auto onda, la cual ayuda a la actividad interna de la Red a alcanzar el umbral de por lo menos una neurona en cada iteración.

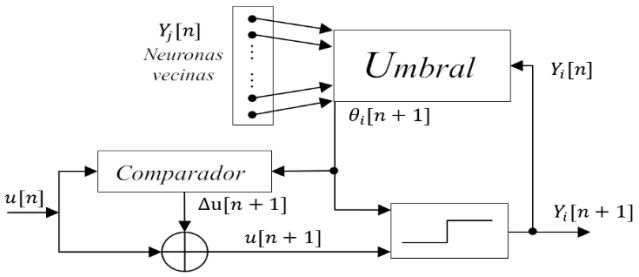

Fig. 3. Diagrama de la neurona del modelo propuesto SAM-PCNN.

Para explicar con mayor claridad el modelo propuesto, es necesario definir las notaciones más relevantes del tema. Los índices i y j se refieren a diferentes neuronas. Para hacer referencia a los vecinos de la neurona i se utiliza R_i. W_{ij} es la distancia positiva diferente de cero que existe entre la neurona i y la neurona j sólo sí $j \in R_i$, de lo contrario, W_{ij} es infinito. N es el número total de neuronas.

La descripción matemática del modelo SAM-PCNN es la siguiente: cada una de las neuronas tiene una salida binaria Y_i la cual se calcula de acuerdo con la ecuación (10).

$$Y_i[n+1] = \begin{cases} 1 \; if \; u[n+1] > \theta_i[n+1] \\ 0 \;\; de \; otra \; manera \end{cases}. \tag{10}$$

Si $Y_i[n] = 1$, se dice que la neurona i ha pulsado en la iteración n, u es la actividad interna de la red y θ_i es el umbral de la neurona, el cual se calcula con la ecuación (11).

$$\theta_i[n+1] = \begin{cases} V_\theta & if \; Y_i[n] = 1 \\ \min\left(u[n] + W_{ji}[n], \theta_i[n]\right) & \text{if } Y_i[n] = 0, Y_j[n] = i \; y \; i \; \in R_j, \\ \theta_i[n] & de \; otra \; manera \end{cases} \tag{11}$$

$$V_\theta = NW_{max}. \tag{12}$$

El umbral θ_i puede tomar tres valores diferentes dependiendo de las salidas de la Red. V_θ es un valor alto que el umbral toma cuando la neurona tiene una salida positiva. Se calcula con la Ecuación 12 multiplicando el número total de neuronas N por la máxima distancia de $W.V_\theta$ evita que la neurona pulse nuevamente. Si la neurona vecina j pulsa, se busca el valor mínimo entre $u[n] + W_{ji}[n]$ y el valor actual del umbral $\theta_i[n]$.

La parte novedosa de este modelo es la manera en que se comporta la actividad interna u. El mecanismo que permite que u incremente de manera dinámica de tal forma que pueda alcanzar el umbral de por lo menos una neurona en cada iteración, es el que se muestra en la ecuación (13) y se muestra en el diagrama como Comparador.

$$\Delta u[n+1] = \min(x) - u[n] \qquad (\forall x \in \theta[n+1] | x > u[n]). \tag{13}$$

En la expresión anterior se muestra que el incremento de u es Δu. Se calcula con la diferencia entre el valor mínimo de las x y u. Siendo x un elemento de los umbrales θ mayores que u. El cálculo de $u[n+1]$ se muestra en la ecuación (14).

$$u[n+1] = u[n] + \Delta u[n+1]. \tag{14}$$

En la ecuación (15) se muestra el cálculo de W la representa la matriz de distancias. Cuando una neurona ha pulsado se modifican las distancias que llevan a esa neurona por el valor de V_θ.

$$W_{ji}[n+1] = \begin{cases} V_\theta \; if \; Y_i[n+1] = 1 \text{ y } i \in R_j \\ W_{ji}[n] \; de \; otra \; manera \end{cases}. \tag{15}$$

Si una neurona i es alcanzada por la auto-onda proveniente de una neurona j, se guarda la neurona j como el padre de la neurona i utilizando la notación R_i^P. Cuando la onda de disparo alcanza la posición de la neurona meta, la neurona meta se disparará y se determinará una ruta desde la meta a la neurona inicio moviéndose sucesivamente hacia el padre de cada neurona. Este será el camino mas corto desde la neurona de inicio a la neurona meta.

3.1. Pseudocódigo SAM-PCNN

Para tener mayor claridad del funcionamiento del modelo propuesto, se presenta el siguiente pseudocódigo para mostrar los pasos de la red propuesta cuando se utiliza para resolver el problema del camino más corto. Aquí la neurona de inicio se representa con una s, la neurona meta se representa con una g. Uno de los aspectos importantes de este algoritmo y de las PCNN en general, es que se puede hacer en paralelo, lo que resulta en un cálculo más eficiente, especialmente con búsquedas de gran magnitud.

---

1) Inicializar la red: V_θ de acuerdo con la Ecuación 12, $n = 0$, $Y_i[0] = 0$, $u[0] = 0$, $\Delta u[0] = 0$, $Y_s[0] = 1$, $\theta_s[0] = V_\theta$,
2) Para cada neurona i en la SAM-PCNN:
    a. Calcular $\theta_i[n+1]$ de acuerdo con la Ecuación 11.
    b. Calcular la velocidad de la auto-onda $\Delta u[n+1]$ de acuerdo con la Ecuación 13.
    c. Calcular la actividad interna $u[n+1]$ de acuerdo con la Ecuación 14.

d. Calcular $Y_i[n+1]$ de acuerdo con la Ecuación 10.
e. Si $Y_i[n+1]=1$.
   i. Actualizar $W_{ji}[n+1]$ con forme la Ecuación 15.
   ii. $\theta_i[n+1]=V_\theta$.
   iii. Guardar los padres de las neuronas que han sido estimuladas
f. Si la neurona g ha pulsado ir a paso 3, de lo contrario $n=n+1$ y regresar al paso 2.

3) Explicitar la ruta generada buscando en el vector de neuronas padre desde la neurona g hasta que la neurona padre $R_i^P=s$.

3.2. Caso de ejemplo ilustrativo

Tomando en cuenta el grafo que se presenta en la Figura 5, se busca el camino más corto del nodo de inicio A al nodo meta C. El algoritmo SAPCNN [15] tiene el siguiente comportamiento:
Al inicializar la red en la iteración $n=0$, se hace pulsar el nodo A

1. Cuando $n=1$: El valor de la actividad interna calculado por las Ecuaciones 6, 7 y 8 es $S[1]=2$, y no le permite alcanzar el umbral de ninguna de las neuronas vecinas.
2. Cuando $n=2$: La actividad interna toma el valor $S[2]=3$, permitiéndole a la neurona B generar un pulso en su salida $Y_B[2]=1$.
3. Cuando $n=3$: $S[3]=5$ y no alcanza al umbral de la neurona C.
4. Cuando $n=2$: La actividad interna es $S[4]=8$ y $Y_C[4]=1$, dando por finalizada la búsqueda.

En comparación con el modelo SAPCNN [15], el modelo SAM-PCNN es capaz de encontrar esta trayectoria en menos iteraciones. En la iteración $n=0$ los valores de la red son: $\theta_s[0]=V_\theta, Y_s[1]=1, u[0]=0$. A partir de ahí, la Red SAM-PCNN se comporta de la siguiente manera:

1. Cuando $n=1$:
   Ecuación 11 cálculo de umbrales: $\theta_B[1]=3, \theta_C[1]=10$
   Ecuación 13 cálculo de velocidad de auto-onda: $\Delta u[1]=3$
   Ecuación 14 cálculo de actividad interna: $u[1]=3$
   Ecuación 10 se obtiene salida: $Y_B[1]=1$
   Actualizar umbrales $\theta_B[2]=V_\theta$
   Guardar padres: $R_B^P=s$
2. Cuando $n=2$:
   Ecuación 11 cálculo de umbrales: $\theta_C[2]=8$
   Ecuación 13 cálculo de velocidad de auto-onda: $\Delta u[2]=5$
   Ecuación 14 cálculo de actividad interna: $u[2]=8$
   Ecuación 10 se obtiene salida: $Y_C[2]=1$
   Actualizar umbrales $\theta_C[2]=V_\theta$
   Guardar padres: $R_C^P=B$

Se terminan las iteraciones de la Red ya que $Y_C = 1$. Se obtiene la ruta calculada recorriendo el vector de padres empezando por la neurona meta: $C \rightarrow R_C^P = B \rightarrow R_B^P = A$. Se hace notar que, para este caso, AWNN [14] requeriría ocho iteraciones.

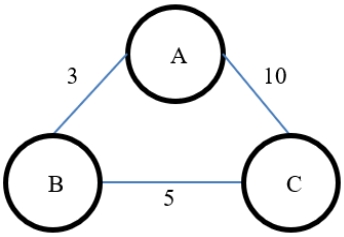

Fig. 4. Grafo del ejemplo.

4. Experimentación y resultados

En esta Sección se presentan experimentos que permiten ilustrar de mejor manera la eficiencia del modelo SAM-PCNN. Para realizar los experimentos se estudiaron cuidadosamente y se implementaron los modelos de PCNN propuestos por Ma [1], Paredes-Cano [14] y Li [15]. La implementación de los algoritmos se hizo en C++ de manera secuencial, en un equipo de cómputo con un microprocesador i5-7200U de séptima generación @ 2.50 GHz 2.70 GHz con 8 GB de memoria RAM. Para el registro del tiempo de procesamiento se presenta un tiempo promediado de 100 repeticiones.

4.1. Experimento 1

Para el primer experimento se utilizó un grafo asimétrico con diez nodos y veinte aristas presentado en Li [15] el cual se muestra en la Figura 6. Se buscaron 10 caminos diferentes dentro del grafo. Las búsquedas se originaron desde todos los nodos al nodo 10 y del nodo 10 al nodo 1. En la Tabla 2 se presentan los resultados de cada uno de los algoritmos al realizar estas búsquedas.

En la tabla 2 se puede apreciar que los algoritmos AWNN [1] y AWNN Explicitada [14], necesitan una mayor cantidad de iteraciones para el cálculo de la ruta. Al modelo AWNN [1] le toma una iteración por cada unidad de distancia. Es decir, si la distancia del camino más corto es 15 como en el ejercicio 1, requerirá de 15 iteraciones para resolver el problema. Al modelo AWNN Explicitada [14] le toma menos iteraciones debido a que la velocidad de la auto onda es dinámica en la primera iteración y posteriormente se vuelve constante.

Al analizar los resultados del modelo SAPCNN [15], es notorio que puede encontrar el camino más corto en menos iteraciones, ya que cuenta con una auto-onda que se propaga de manera dinámica dependiendo del estado de la Red, lo que le permite a su actividad interna cumplir fácilmente con las condiciones necesarias para generar pulsos en las neuronas de la Red.

Sin embargo, a pesar del mecanismo dinámico de este modelo, aún existen iteraciones donde no hay cambios en la Red, es decir, iteraciones donde ninguna de sus neuronas pulsa.

Tabla 1. Resultado de las iteraciones por algoritmo del Experimento 1.

			Iteraciones			
Experimento	Ruta	Costo	AWNN [1]	AWNNE [14]	SAPCNN [15]	SAM-PCNN
1	1-6-10	15	15	11	9	**6**
2	2-7-10	16	16	16	10	**6**
3	3-8-10	20	20	17	13	**8**
4	4-9-10	14	14	8	8	**4**
5	5-9-10	10	10	8	5	**3**
6	6-10	10	10	6	6	**4**
7	7-10	14	14	13	9	**6**
8	8-10	16	16	13	11	**8**
9	9-10	7	7	5	4	**2**
10	10-6-1	15	15	9	9	**4**
		Total	137	106	84	**51**
Diferencia con respecto a SAM-PCNN			+86	+55	+33	**0**

Por otro lado, el modelo propuesto SAM-PCNN presenta una menor cantidad de iteraciones para encontrar el camino más corto. El modelo cuenta con un mecanismo de auto onda que adapta su velocidad de manera heurística, lo cual le permite optimizar el número de iteraciones necesarias para el cálculo de la ruta.

En total el modelo propuesto SAM-PCNN hizo 86 iteraciones menos que el modelo AWNN [1], 55 iteraciones menos que el modelo AWNN Explicitada [14] y 33 iteraciones menos que SAPCNN [15].

En la Figura 7 se muestra de manera gráfica la comparativa de las iteraciones realizadas por cada modelo en cada uno de los experimentos. Las figuras en negro representan los resultados del modelo AWNN [1] con respecto de los demás modelos.

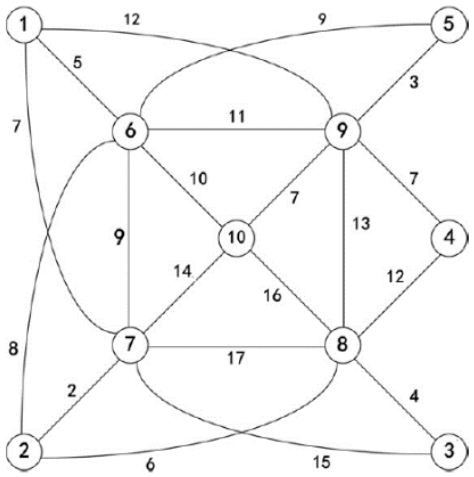

Fig. 6. Grafo asimétrico con 10 nodos y 20 aristas [15].

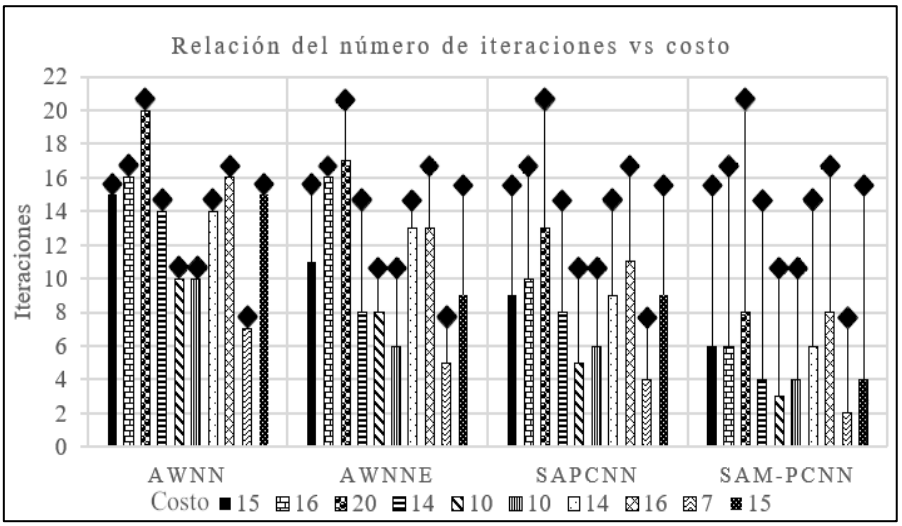

Fig. 7. Gráfica comparativa del número de iteraciones contra el costo o la distancia óptima en el Experimento 1.

4.2. Experimento 2

Para el segundo experimento se generaron aleatoriamente 4 grafos de 30, 50, 200 y 500 nodos respectivamente. Dichos grafos fueron generados tal como se especifica en [16]. El objetivo del experimento fue buscar el camino más corto entre el nodo 1 y el último nodo de cada grafo. El resultado de las iteraciones de cada algoritmo en cada grafo se muestra en la figura 8.

En la tabla 3 se presentan los tiempos de procesamiento en segundos obtenidos en cada una de las búsquedas.

Los resultados de la Tabla 2, muestran que, con grafos con pocos nodos, la diferencia en tiempo es apenas perceptible, pero cuando se trata de grafos de mayores dimensiones, la diferencia es considerable. Por otro lado, se puede apreciar que el tiempo de procesamiento del modelo SAM-PCNN tiene un comportamiento lineal con respecto al número de nodos.

Tabla 2. Comparativa de tiempos de procesamiento en segundos del Experimento 2.

	Nodos			
Algoritmo	30	50	200	500
AWNN	0.005914	0.002860	0.272663	13.8336
AWNN E	0.006121	0.002719	0.268951	13.7898
SAPCNN	0.002203	0.001512	0.111357	4.20460
SAM-PCNN	**0.002172**	**0.001152**	**0.111235**	**1.69298**

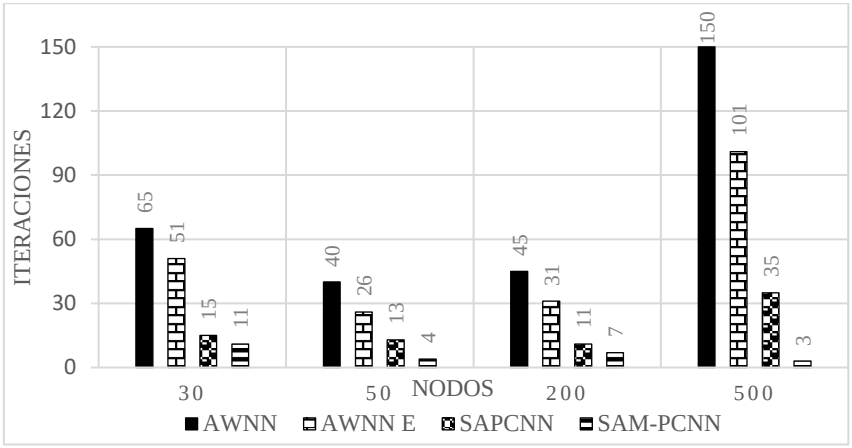

Fig. 8. Comparativa de iteraciones del Experimento 2.

5. Discusión

Encontrar la solución de la ruta óptima es un problema NP completo, incluso el muy citado algoritmo Dijkstra tiene complejidad algorítmica de $O(N^2)$, donde N es el número de nodos de un grafo G. Para los modelos de PCNN como el modelo AWNN [1], el cual cuenta con una velocidad constante de propagación, la complejidad algorítmica es $O(LN)$, donde L es equivalente a la distancia entre un nodo de inicio y el nodo más lejano. En el caso de los modelos SAPCNN [15] y SAM-PCNN, los cuales cuentan con una velocidad dinámica de propagación, la complejidad algorítmica es $O(N)$, debido a que para cada iteración es necesario calcular los umbrales, la velocidad de la auto onda y la actividad interna. Tomando en cuenta que M es el número de iteraciones necesarias para encontrar la solución, la complejidad de ambos modelos es $O(MN)$. Sin embargo, para el modelo propuesto SAM-PCNN, el número de iteraciones es proporcional al número de neuronas cuya distancia al nodo de inicio sea menor que la distancia del nodo objetivo. Es decir, si se desea calcular el camino más corto del nodo A al nodo C, pero el nodo B es más cercano al nodo A, el número de neuronas más cercanas que C es 1 que se puede representar como P. En este caso M es igual a $P + 1$, pues en la primera iteración pulsará la neurona B y en la siguiente iteración pulsará la neurona C. Por lo tanto, la complejidad algorítmica para el modelo propuesto SAM-PCNN puede re-escribirse como $O((P + 1)N)$.

6. Conclusiones y trabajo futuro

En este artículo se presentó el modelo SAM-PCNN para resolver de manera eficiente el problema del camino más corto. Para poder realizar la experimentación, fue necesario estudiar e implementar los modelos AWNN [1], AWNN Explicitada [14] y SAPCNN [14]. Se mostró que el modelo propuesto no sólo es una buena alternativa para resolver

este tipo de problemas, sino que además resulta ser más eficiente que otros modelos de PCNN que persiguen el mismo objetivo.

De acuerdo con los resultados del experimento 1, el modelo SAM-PCNN es hasta 39.28% más eficiente con respecto del modelo SAPCNN [15] y hasta un 62.77% más eficiente que el modelo AWNN [1].

En esta investigación los algoritmos fueron implementados de manera secuencial, por lo que los resultados presentados pueden mejorar considerablemente con una implementación paralela. Esto último se tiene contemplado como parte del trabajo futuro relacionado con esta investigación, además de probar el modelo SAM-PCNN en otros ámbitos como la Planeación de Trayectorias Discretas para Robots Móviles.

Referencias

1. Ma, Y., Zhan, K., Wang, Z.: Pulse-Coupled Neural Networks. Applications of Pulse-Coupled Neural Networks. Springer Heidelberg Dordrecht London New York, pp. 447–465 (2011)
2. Morrison, D. R., Jacobson, S.H., Sauppe, J.J., Sewell, E.C.: Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. Discrete Optimization, 19, pp. 79–102 (2016)
3. Thulasiraman, K., Arumugam, S., et.al.: Handbook of graph theory combinatorial optimization, and algorithms. CRC Press (2016)
4. Dijkstra, E.W.: A note on two problems in connection with graphs, Numer. Math. 1(1), 269–271 (1959)
5. Cherkassky, B.V., Goldberg, A.V., Radzik, T.: Shortest paths algorithms: theory and experimental evaluation, Math. Programming 73 (2), 129–174 (1996)
6. Hopfield, J.J., Tank, D.W.: Neural computation of decisions in optimization problems. Biol. Cybern, vol. 52, pp. 141–152 (1985)
7. Eckhorn, R., Reitboeck, H.J., Arndt, M., Dicke, P.W.: Feature linking via synchronous among distributed assemblies: Simulations of results from cat visual cortex. Neural Comput, vol. 2, pp. 293–307 (1990)
8. Ranganath, S.H., Kuntimad, G.: Object detection using pulse coupled neural networks. IEEE Trans. Neural Netw 10(3), 615–620 (1999)
9. Gu, X., Yu, D., Zhang, L.: Image shadow removal using pulse coupled neural network. IEEE Trans. Neural Netw 16(3), pp. 692–698 (1999)
10. Muresan, R.C.: Pattern recognition using pulse-coupled neural networks and discrete Fourier transforms. Neurocomputing, vol. 51, pp. 487–493 (2003)
11. Ortiz, E., Mejía-Lavalle, M., Sossa, H.: Uso de redes neuronales pulsantes para mejorar el filtrado de imágenes contaminadas con ruido Gaussiano. Research in Computing Science, Vol 114, pp. 45–58 (2016)
12. Caulfield, H.J., Kinser: Finding M.: The shortest path in the shortest time using PCNNs. IEEE Trans. Neural Netw 10(3), 604–606 (1999)
13. Qu, H., Yi, Z.: A new algorithm for finding the shortest paths using PCNNs Chaos. Solutions Fractals 33(4), 1220–1229 (2007)
14. Paredes-Cano, J.J., Mejía-Lavalle, M., Mújica-Vargas, D., Reyes-Salgado, G.: Red Neuronal Pulsante Explicitada para el Problema del Camino más Corto, IEEE (2018)
15. Li, X., Ma, Y., Feng, X.: Self-adaptive autowave pulse-coupled neural network for shortest-path problema. Neurocomputing, vol. 115, pp. 63–71 (2013)
16. Qu, H., Yi, Z., Yang, S.X.: Efficient Shortest-Path-Tree Computation in Network Routing Based on Pulse-Coupled Neural Networks. IEEE Transactions on Cybernetics, 43(3), 995–1010 (2013)