SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Redes Neuronales Pulsantes como Alternativa para la Optimización de Trayectorias

presentada por

Ing. José de Jesús Paredes Cano

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Director de tesis

Dr. Manuel Mejía Lavalle

Codirector de tesis

Dr. Juan Humberto Sossa Azuela

Cuernavaca, Morelos, México. Diciembre de 2018.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Morelos a 30 de noviembre del 2018
OFICIO No. DCC/238/2018

Asunto: Aceptación de documento de tesis

DR. GERARDO V. GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del **Ing. José de Jesús Paredes Cano**, con número de control M16CE084, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis profesional titulado "**Redes Neuronales Pulsantes como Alternativa para la Optimización de Trayectorias**" y hemos encontrado que se han realizado todas las correcciones y observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

DIRECTOR DE TESIS

Dr. Manuel Mejía Lavalle
Doctor en Ciencias Computacionales
8342472

CO-DIRECTOR DE TESIS

Dr. Juan Humberto Sossa Azuela
Doctor en Informática

REVISOR 1

M.C. Gerardo Reyes Salgado
Maestro en Ciencias de la Computación
2493370

REVISOR 2

Dra. Andrea Magadán Salazar
Doctorado en Ciencias Computacionales
10654097

C.p. M.T.I. María Elena Gómez Torres - Jefa del Departamento de Servicios Escolares.
Estudiante
Expediente

NACS/lmz

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Mor., 4 de diciembre de 2018
OFICIO No. SAC/561/2018

Asunto: Autorización de impresión de tesis

ING. JOSÉ DE JESÚS PAREDES CANO
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"Redes Neuronales Pulsantes como Alternativa para la Optimización de Trayectorias"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
EXCELENCIA EN EDUCACIÓN TECNOLÓGICA®
"CONOCIMIENTO Y TECNOLOGÍA AL SERVICIO DE MÉXICO"

SEP TECNM CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO SUBDIRECCIÓN ACADÉMICA

DR. GERARDO VICENTE GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO

C.p. M.T.I. María Elena Gómez Torres .- Jefa del Departamento de Servicios Escolares.
Expediente

GVGR/mcr

Dedicatoria

Dedico este nuevo logro **a mis padres y hermano** quienes a través de valores, esfuerzo y amor me han acompañado a lo largo de mi camino.

A mi familia por su cariño, apoyo incondicional y vivencias a su lado, siempre están en mi mente y corazón.

Dedico esta tesis **a mis amigos y a mi novia** quienes han contribuido en mi formación como persona.

De igual forma dedico este trabajo **a mis profesores y compañeros** que a lo largo de mi educación me han formado como estudiante y profesionista, y por los momentos compartidos.

Agradecimientos

José de Jesús Paredes Cano agradece al **Consejo Nacional de Ciencia y Tecnología (CONACyT)** por la beca otorgada para realzar los estudios de posgrado.

Se agradece también al **CONACyT** y a la **SIP-IPN** por el apoyo brindado en el marco de proyectos 65 (Fronteras de la Ciencia) y 20180730, respectivamente.

Al **Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET)** por haber hecho posible este trabajo de investigación.

Se agradece a los **profesores investigadores de CENIDET** que contribuyeron a mi formación durante la maestría, en especial al Dr. Manuel Mejía Lavalle y al Dr. Juan Humberto Sossa Azuela por guiar acertadamente el desarrollo de esta tesis, así mismo agradezco a mis revisores el Dr. Gerardo Reyes Salgado y a la Dra. Andrea Magadán Salazar por sus observaciones, comentarios y sugerencias sobre este proyecto.

A mi familia por estar siempre en cada momento iluminando mi camino con cuantiosas muestras de amor.

A todas las personas y amigos que han compartido su tiempo en tantas experiencias vividas.

Contenido

Resumen

En esta tesis se trabaja con redes neuronales artificiales pulsantes o de tercera generación para optimización de trayectorias con una aplicación en el problema del camino más corto entre dos nodos. Se realiza un análisis de trabajos previos en CENIDET con redes neuronales con especial atención a la tercera generación, un análisis del estado del arte de este tipo de redes neuronales para problemas de optimización de trayectorias y de las bases de datos mencionadas para evaluar dicho modelo.

Se analiza principalmente el modelo neuronal AWNN adaptado para el problema del camino más corto y el modelo ICM para comprender las características de los modelos derivados de la red neuronal pulso acoplada (PCNN). Se realiza una aportación al modelo AWNN de un algoritmo de "Reconstrucción de ruta" a través de la explicitación del conocimiento de la red.

Las pruebas se realizaron con bases de datos de tres tipos: propuestas en la literatura, un "Generador de casos de prueba" aleatorios que van desde los 6 hasta los 9000 nodos y un caso real de una red de carreteras, como parte de la comparación se implementa el algoritmo Dijkstra tomado de la literatura.

Los resultados finales muestran que los algoritmos pulsantes son una propuesta altamente competitiva si se realiza una implementación completamente paralela en hardware especializado, ya que permitieron resolver grafos de hasta 9000 nodos de manera óptima en un tiempo aproximado de 10 segundos en una implementación secuencial mostrado en este trabajo.

Finalmente se realiza un análisis de la implementación de este paradigma para problemas de optimización de trayectorias, en especial para el problema del camino más corto como alternativa a los métodos tradicionales.

Abstrat

In this thesis we work with artificial spiking neural networks or third generation for trajectory optimization with an application in the shortest path between two nodes problem. An analysis of previous works in CENIDET with neural networks is carried out with special focus on the third generation. A state-of-the-art analysis of this type of neural networks is carried out for problems of optimization of trajectories and of the aforementioned databases to evaluate said model.

We analyze the neural model of AWNN adapted for the shortest path problem. We also analyze other models, such as the ICM, to understand the characteristics of the models derived from the neural network coupled by pulses (PCNN). A contribution to the AWNN model of a "Reconstruction of route" algorithm is made through the explicitation of knowledge of the network.

The tests were conducted with databases of three types: proposals in the literature, a random "Generator of test cases" ranging from 6 to 9,000 nodes and a real case of a road network. As part of the comparison, the Dijkstra algorithm taken from the literature is implemented.

The final results show that the pulsing algorithms are a highly competitive proposal if a completely parallel implementation is made in specialized hardware, since they allowed to solve graphs of up to 9000 nodes optimally in an approximate time of 10 seconds in a sequential implementation shown in this work.

An analysis is made of the use of this paradigm for trajectory optimization problems, especially for the shortest path problem as an alternative to traditional methods.

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

En este primer capítulo se realiza la descripción de la motivación de esta investigación, el problema de estudio que se resolvió, el objetivo general y los objetivos específicos que se alcanzaron, los alcances y limitaciones que guiaron el desarrollo del proyecto y finalmente la metodología empleada para completar la investigación.

1.1. Motivación

La obtención de la trayectoria óptima o sub-óptima (T.Ó.S-Ó) dada una red conectada representada por un grafo es un tema de actual y gran interés en el campo de la computación, en la distribución de rutas para robots, en el enrutamiento de redes de comunicación y de rutas de transporte (Figura 1.1).

Figura 1.1 Aplicaciones de Optimización de trayectorias

Entre otras aplicaciones como: Robótica móvil, Inteligencia artificial, Aprendizaje automático, Investigación de operaciones, Teoría de juegos, Redes de computadoras,

Internet, Diseño industrial, Telemática, Fenómenos de transporte, Diseño de circuitos electrónicos altamente integrados (VLSI), Situaciones de riesgo y toma de decisiones [Moustapha, 2016].

Este problema se encuentra clasificado como un problema de optimización combinatoria, en el cual, dadas las condiciones restringidas, se considera todo el espacio de posibles soluciones para encontrar la variable que hace que la función objetivo sea mayor o menor [Ma, 2010]. La teoría de grafos, dada por Euler en 1736 se utiliza para representar el grafo a través de una matriz de adyacencia de tamaño $n\ x\ n$ donde n es la dimensión del grafo, y las filas y columnas hacen referencia a las aristas para almacenar en cada casilla la longitud entre cada par de nodos del grafo. En dicho problema si su dimensión aumenta, los algoritmos tradicionales son métodos ineficaces y pueden consumir una cantidad considerable de tiempo de la Unidad Central de Procesamiento (CPU).

Actualmente, existen una gran variedad de técnicas para la T.Ó.S-Ó, entre los algoritmos eficientes más conocidos está el de Dijsktra, Bellman-Ford, Floyd-Warshall, Johnson, Gabow, Thorup, Pettie-Ramachandran o Viterbientre [Moustapha, 2016], sin embargo, la optimización de trayectorias es una tarea la cual se sigue explorando y perfeccionando.

Desde el trabajo de Hopfield y Tank en 1985, muchas investigaciones se han dirigido a la aplicación de las redes neuronales de hopfield a problemas de optimización combinatoria [Hopfield, 1985]. Los principales inconvenientes de las redes de Hopfield se enumeran como: (1) Invalidez de las soluciones obtenidas. (2) Proceso de valor de ajuste de prueba y error de los parámetros de redes. (3) Bajo rendimiento de computación [Qu, 2007]. En la década de los 90 Eckhorn estudio el fenómeno de las ráfagas de impulsos sincrónicas observando en las cortezas visuales de los gatos [Eckhorn, 1990], estos modelos matemáticos se conocen como Redes Neuronales Pulso-Acopladas o PCNN por sus siglas en inglés, principalmente aplicados al procesamiento de imágenes por Kuntimad [Lindblad, 2005]. Caulfield y Kinser presentaron la idea de utilizar la onda

automática de las PCNN para encontrar la solución al problema del laberinto [Caulfield, 1999].

En el CENIDET se ha comenzado a implementar este tipo de Redes Neuronales Artificiales (RNA) principalmente en el área de Visión Artificial para completar la extracción de características, la segmentación y la compresión de imágenes, así como a problemas de reconocimiento de patrones y clasificación.

Mediante esta tesis se implementó un algoritmo de optimización de trayectorias basados en Redes Neuronales Pulso-Acopladas (PCNN), así como la implementación del mismo con la Interfaz de programación de aplicaciones (API) de Open Multi-Processing (OpenMP) como parte de la optimización del algoritmo y se realizó una comparación con los métodos tradicionales Dijkstra y Bellman-Ford experimentado con tres tipos de bases de datos, conformando un estudio comparativo entre este tipo de RNA evaluado su desempeño en tiempo de procesamiento, calidad de la obtención de la T.Ó.S-Ó y el espacio de búsqueda explorado durante el proceso de búsqueda de la T.Ó.S-Ó con respecto a las técnicas tradicionales de optimización de trayectorias.

1.2. Descripción del problema

Muchas áreas de aplicación en la Inteligencia Artificial se enfrentan a un problema de búsqueda de trayectorias, dentro del dominio de optimización se encuentra la la optimización combinatoria en la cual la obtención de la T.Ó.S-Ó es un problema clásico (Figura 1.2) que tiene características difíciles de resolver como su complejidad computacional, por lo que se le denomina problema NP-completo.

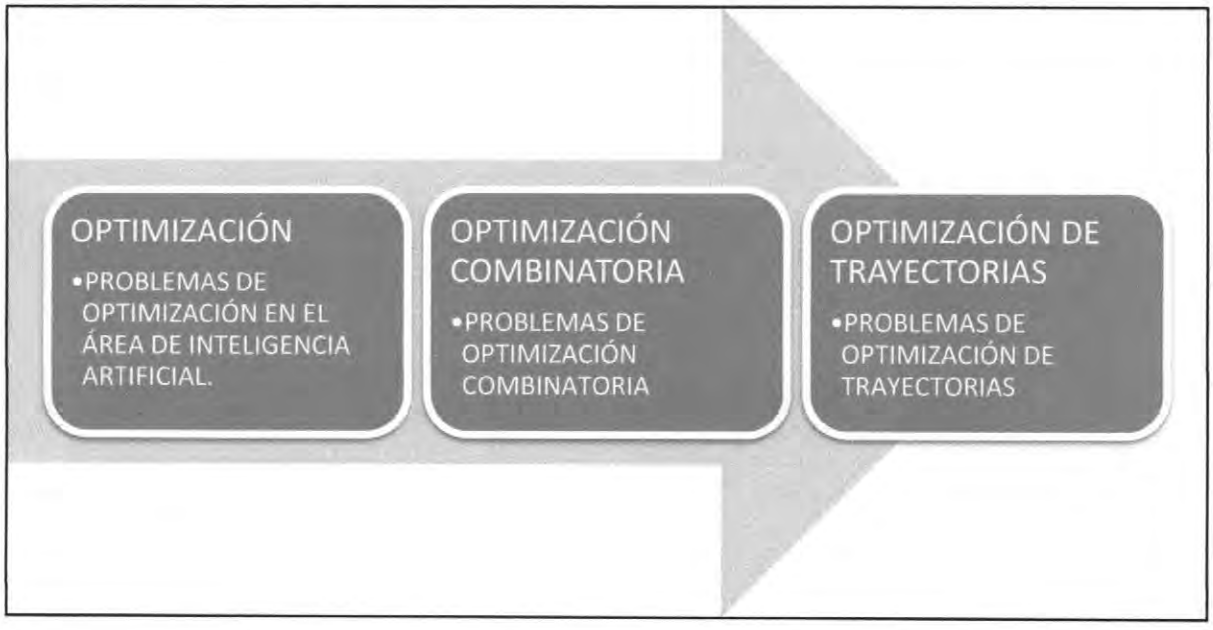


Figura 1.2 Ubicación del problema

Los problemas NP-completos de tener una solución polinómica tendrían también una solución en tiempo polinómico, los cuales al aumentar su dimensionalidad aumenta proporcionalmente su tiempo de procesamiento para obtener una solución óptima.

Los trabajos presentados sobre problemas de optimización de trayectorias basados en Redes Neuronales Artificiales Pulsantes (RNAP) son variantes de la PCNN que, en su naturaleza de algoritmos paralelos, estos están basados en inspiraciones biológicas del procesamiento de la corteza visual de los mamíferos, los cuales dejan ver que presentan

niveles de efectividad superiores a las técnicas tradicionales de optimización de trayectorias [Ma, 2010].

Una forma propuesta para resolver este problema consiste en implementar técnicas de Inteligencia Artificial, como las RNA, especialmente empleando las propiedades de las PCNN (Figura 1.3). Las PCNN han mostrado ventaja con respecto a las técnicas tradicionales en la obtención de la T.Ó.S-Ó [Ma, 2010].

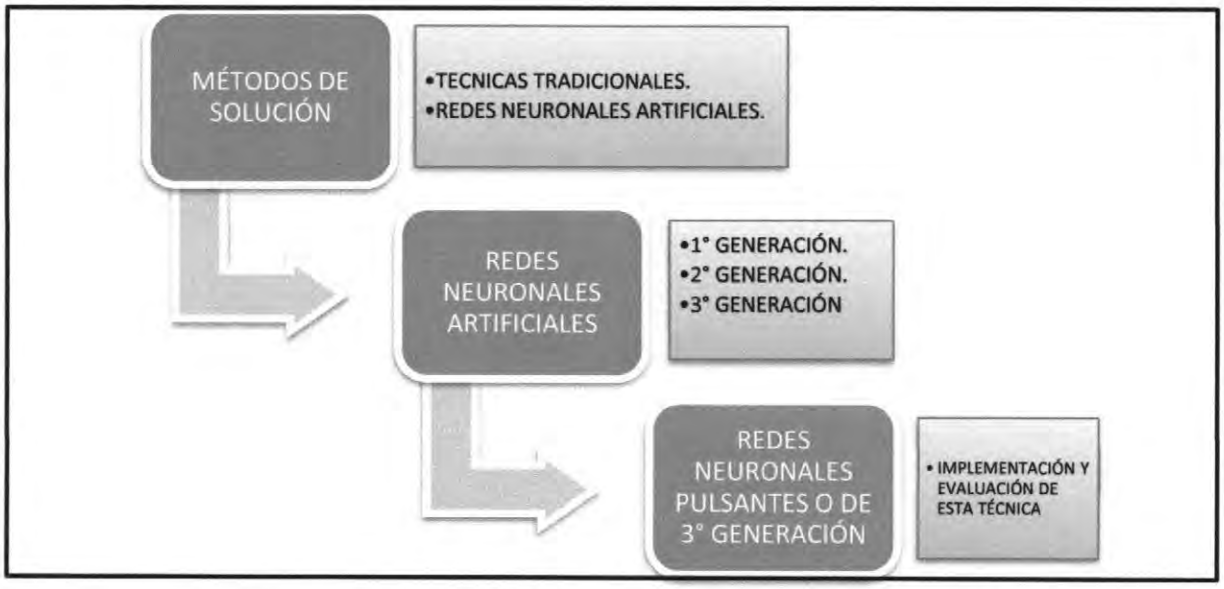


Figura 1.3 Propuesta de solución

1.2.1. Delimitación del problema específico

El problema específico de la presente investigación se puede enunciar como: encontrar los parámetros de la RNAP basada en PCNN que obtengan la T.Ó.S-Ó a partir de un grafo no dirigido representado por una matriz de costos, por lo que se pretende resolver el problema del camino más corto por medio de Redes Neuronales Pulsantes (Figura 1.4).

Figura 1.4 Delimitación del problema.

1.2.2. Complejidad del problema

En el CENIDET se ha implementado la red PCNN para realizar tareas de Visión Artificial en el procesamiento de imágenes, no obstante, esta red no ha sido evaluada para problemas de optimización de trayectorias los cuales tienen características difíciles de resolver como su complejidad computacional ya que se denominan problemas NP-completos.

Hasta el momento no existe un estudio previo que permita comparar, de manera ágil y confiable, a las técnicas de optimización de trayectorias tradicionales para el problema de la ruta óptima o sub-óptima y la técnica Auto-Wave Neural Network (AWNN) o Red Neuronal de la onda automática basada en PCNN para establecer cuales técnicas son superiores en calidad de la obtención de la T.Ó.S-Ó, tiempo de procesamiento y porcentaje de búsqueda explorado durante el proceso de obtención de la T.Ó.S-Ó, ni con un estudio precedente que permita determinar cuáles son los parámetros iniciales del método propuesto para obtener la trayectoria óptima.

Por medio de esta tesis se llevó a cabo el estudio, la implementación y la experimentación de un algoritmo de obtención de la T.Ó.S.Ó basado en PCNN, así como la comparación de su desempeño contra la técnica tradicional Dijsktra, de modo que se puedan establecer claramente sus ventajas y desventajas para obtener la trayectoria óptima en una red dada representada por la matriz de costos de un grafo no dirigido.

1.2.3. Hipótesis

Probar que, a través de las Redes Neuronales de tercera generación, especialmente las PCNN son útiles para optimizar trayectorias y así resolver el problema del camino más corto. De igual forma, los patrones de conexión derivados del modelo PCNN son apropiados para calcular soluciones a problemas de optimización de trayectorias con ventaja en sencillez de cálculo, tiempo de procesamiento, y calidad de resultados con respecto a los métodos clásicos.

1.3. Objetivo general

Implementar, experimentar y evaluar la Red Neuronal Artificial Pulsante (RNAP) en el dominio de la optimización de trayectorias para la obtención de la trayectoria óptima o sub-óptima.

1.4. Objetivos específicos

1. Conocer los conceptos básicos y el funcionamiento de RNAP en el dominio de optimización de trayectorias.
2. Estudiar y comprender como las RNAP permiten resolver problemas de optimización de trayectorias.

3. Implementar un modelo de las RNAP seleccionadas para resolver problemas de optimización de trayectorias.
4. Experimentar con el modelo de RNAP seleccionado en base de datos de trayectorias conocidas y con casos de problemas planteados en la literatura.
5. Evaluar el desempeño de la RNAP en el dominio de optimización de trayectorias con respecto a las técnicas tradicionales.
6. Redactar un artículo de resultados obtenidos.

1.5. Alcances y limitaciones del proyecto

1.5.1. Alcances:

1. Estudiar los modelos propuestos de RNAP para optimización de trayectorias.
2. Seleccionar por lo menos tres e implementar una variante de PCNN para problemas de optimización de trayectorias.
3. Experimentar con al menos cinco bases de datos de trayectorias conocidas.
4. Experimentar con casos de problemas planteados en la literatura.
5. Realizar el estudio comparativo entre la RNAP y las técnicas tradicionales basado en tiempo de procesamiento y resultados en la literatura de optimización de trayectorias.

1.5.2. Limitaciones:

1. Las comparaciones se realizarán con respecto a lo reportado en la literatura.
2. El desarrollo de los algoritmos será utilizando software de desarrollo de uso libre.
3. No se busca mejorar resultados si no detectar claramente las ventajas y desventajas de la implementación de algoritmos de optimización de trayectorias con variantes de la PCNN.

1.6. Justificación y beneficios

Las PCNN son una prometedora herramienta para la optimización ya que sus patrones de conexión son apropiados para calcular soluciones a problemas de optimización, al ser una emulación de un sistema biológico, representan un nuevo enfoque para resolver estos problemas típicos de optimización de trayectorias de gran utilidad en el área de la Inteligencia Artificial.

Aunque las RNAs constituyen una técnica ampliamente explotada en los ámbitos de la clasificación, el seguimiento de señales, la aproximación de funciones, la predicción y el reconocimiento, las PCNN representan aún un nuevo objeto de estudio en el CENIDET en el área de optimización de trayectorias, y sus parámetros de funcionamiento se encuentran en constante modificación ya que han atraído la atención de los investigadores alrededor del mundo.

Este trabajo aportará conocimiento y una implementación en el área de optimización de trayectorias, útiles para el CENIDET y para investigadores del ramo, primero con respecto a las técnicas de optimización de trayectorias en las PCNN y finalmente con un estudio comparativo con las técnicas tradicionales reportados en la literatura.

1.7. Metodología de solución

El proceso completo de la investigación consta de cuatro etapas que se incluyen en el diagrama de la Figura 1.5 y se describen a continuación:

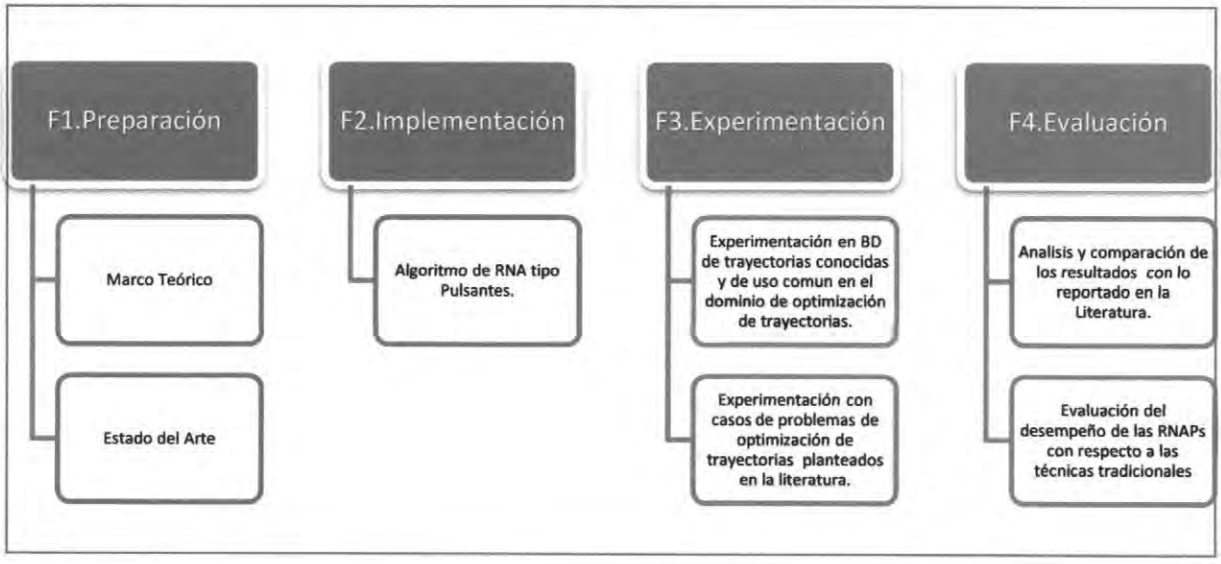


Figura 1.5 Diagrama de Metodología de solución

En la primera etapa se construyó el marco teórico y el estado del arte sobre el problema de la ruta más corta en optimización de trayectorias y sobre tres modelos de PCNN para la obtención de la T.Ó.S-Ó.

En la segunda etapa se implementó un algoritmo tradicional (Dijkstra) y uno basado en PCNN (AWNN) para el problema de la ruta más corta, cabe resaltar que se realizó una aportación al algoritmo AWNN para la "Reconstrucción de la ruta" explicitando el conocimiento de esta RNA. También se implementó dicho algoritmo con la API OpenMP aprovechando la naturaleza paralela de las RNAs de tercera generación.

En la tercera fase de experimentación se realizaron tres tipos de pruebas; el primer tipo con matrices de adyacencia de distintos tamaños generadas por un "Generador de casos de prueba", el segundo corresponde a casos de prueba planteados en la literatura reportados en el estado del arte y el tercer tipo corresponde a un caso real de red de carreteras de la ciudad de Oldenburg [Brinkhoff, 2002].

Los tres tipos de pruebas se implementaron en los algoritmos desarrollados en C++ y en el software "Grafos" para Windows [arodrigu, 2003-2012] con el algoritmo Dijkstra con ciertas restricciones del software.

En la cuarta fase de evaluación se llevó a cabo el análisis y discusión de los resultados obtenidos, los detalles de la implementación, la capacidad para resolver problemas de optimización de trayectorias y la comparación con las técnicas tradicionales por medio de la calidad de la ruta, tiempo de procesamiento y espacio de búsqueda explorado en los tres tipos de bases de datos obtenidas durante el desarrollo del proyecto.

1.8. Organización de la tesis

Con el fin de orientar al lector acerca del contenido del presente trabajo se enumeran y describen brevemente sus partes constitutivas:

El Capítulo 2 contiene el estudio del estado del arte. Se establecen los antecedentes realizados en CENIDET y los trabajos relacionados externos a CENIDET, así como un análisis del estado del arte.

El Capítulo 3 contiene el marco teórico, el cual contiene los conceptos básicos ocupados en el desarrollo de este proyecto.

El Capítulo 4 abarca un análisis y los detalles de la implementación del algoritmo basado en PCNN para el problema del camino más corto.

El Capítulo 5 muestra la experimentación realizada con los algoritmos implementados en las bases de datos obtenidas y los resultados.

El Capítulo 6 contiene las conclusiones y trabajo a futuro de esta investigación.

Capítulo 2

Estado del arte

En este capítulo se detallan los antecedentes más próximos e importantes desarrollados en CENIDET, así como los trabajos relacionados externos al CENIDET sobre optimización de trayectorias y el uso de PCNN, del mismo modo se resume dicha información y se cierra el capítulo con una discusión sobre lo escrito en estos materiales.

2.1. Antecedentes institucionales

En esta sección se mencionan algunos proyectos desarrollados en CENIDET que experimentan con PCNN aplicadas a dominios de Visión Artificial.

Implementación y Evaluación de Redes Neuronales Artificiales tipo "Pulse-Coupled Neural Networks" (PCNN) Aplicadas a Visión Artificial [Cardenas, 2015]. Consiste en un estudio comparativo sobre la segmentación de imágenes por medio de técnicas como PCNN y técnicas clásicas, utilizando para ello metodologías como entropía cruzada y Pixel Correspondance Metric (PCM), se desarrolló una herramienta llamada "Implementation and Evaluation Pulse Coupled" (Figura 2.1) para evaluar el desempeño de las PCNN en la segmentación de imágenes.

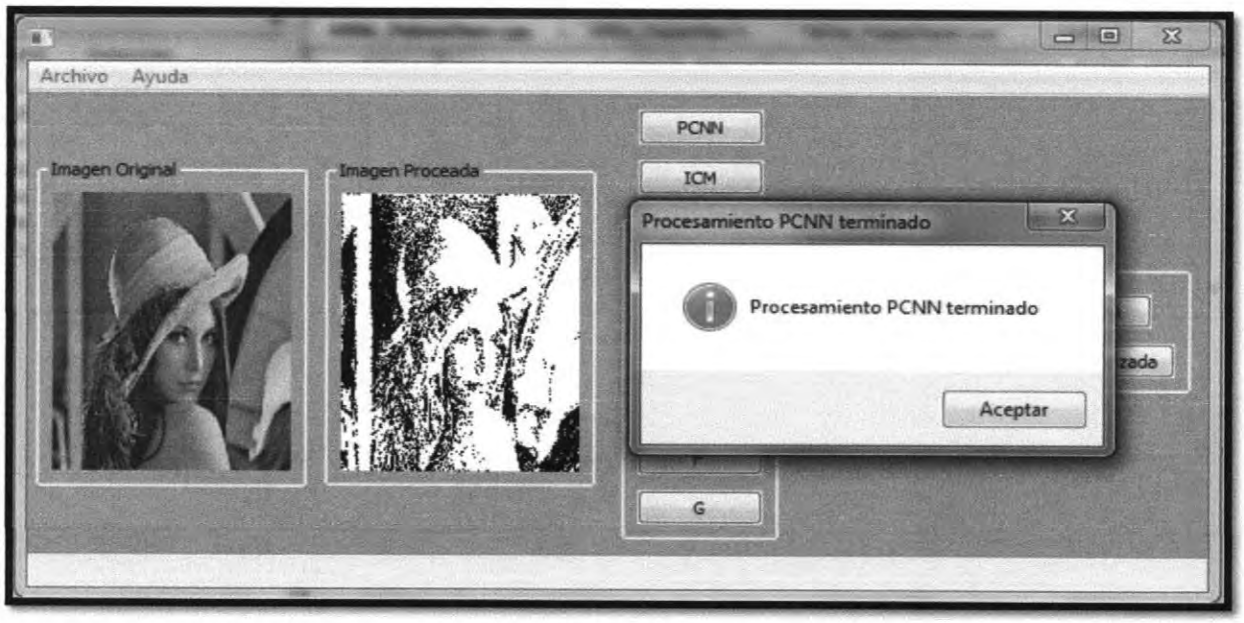


Figura 2.1 Interfaz gráfica de evaluación de PCNN e ICM para segmentación de imágenes [Cárdenas, 2015].

Extracción de Características de Imágenes Digitales mediante una Red Neuronal Artificial Pulsante [Zarate, 2015]. Trabaja con los modelos de PCNN, Intersecting Cortical Model (ICM) y Spiking Cortical Model (SCM) mostrados a continuación (Figura 2.2, 2.3 y 2.4 respectivamente) para la extracción de descriptores de las imágenes.

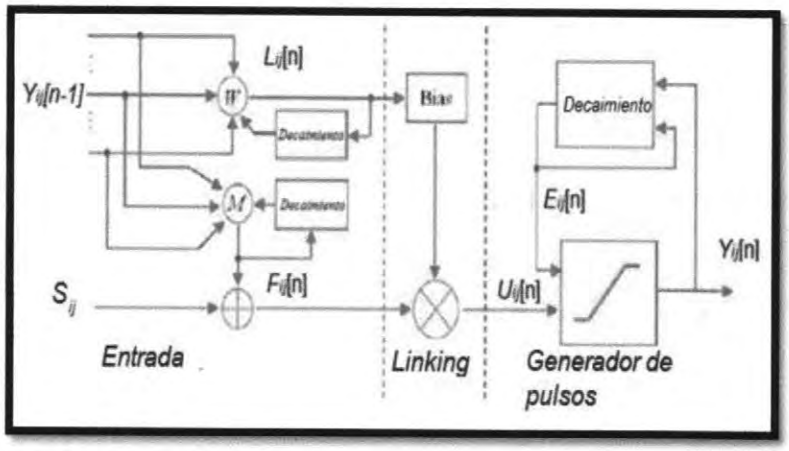


Figura 2.2 Modelo PCNN implementado [Zarate, 2015].

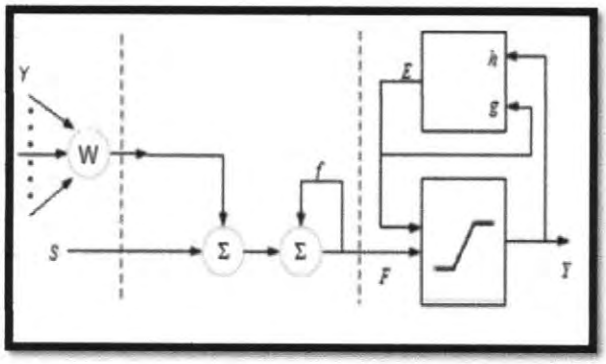


Figura 2.3 Modelo ICM implementado [Zarate, 2015].

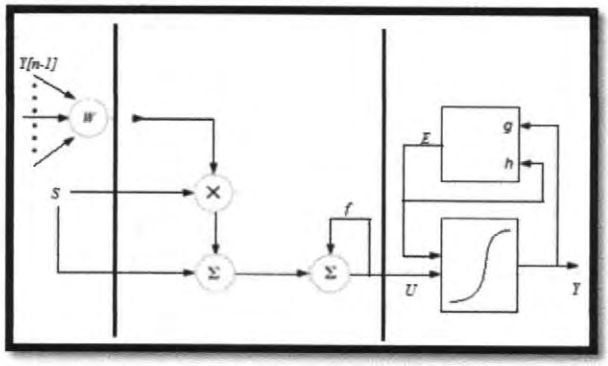


Figura 2.4 Modelo SCM implementado [Zarate, 2015].

Detección de Ruido Impulsivo o Gaussiano en imágenes Monocromáticas Mediante Redes Neuronales Artificiales Pulso-Acopladas [Ortiz, 2017]. Se presenta un esquema de comparación entre técnicas tradicionales y Redes Neuronales Pulso Acopladas para llevar a cabo la detección y filtrado de Ruido impulsivo y Gaussiano en imágenes digitales (Figura 2.5).

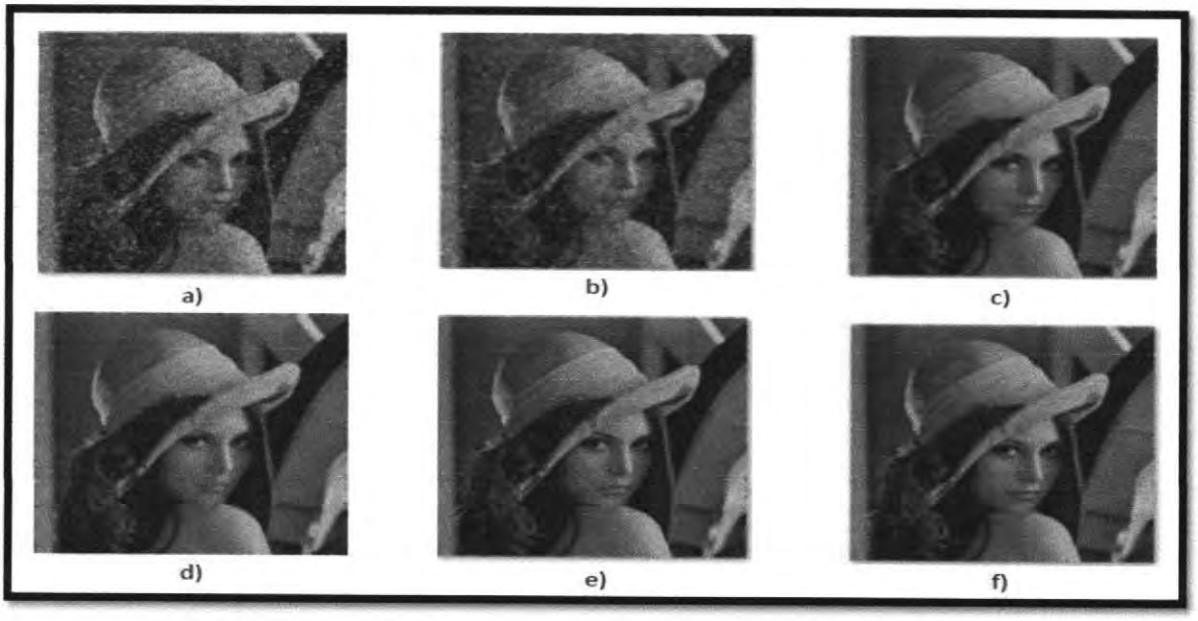


Figura 2.5 a) Imagen con ruido 10 %, filtrada con b) filtro promedio, c) filtro mediana, d) filtro morfológico, e) filtro con SCM, f) filtro con ICM [Ortiz, 2017].

Aplicación del descenso de gradiente para el aprendizaje de neuronas pulsantes de Izhikevich [Hernández, 2017]. Se trabajó con el modelo Izhikevich adaptado para la clasificación con aplicación en Visión Computacional (Figura 2.6), se propuso un algoritmo de aprendizaje supervisado basado en el descenso del gradiente para neuronas de dicho modelo y se experimentó con distintas bases de datos académicas, de cerebros y de imágenes.

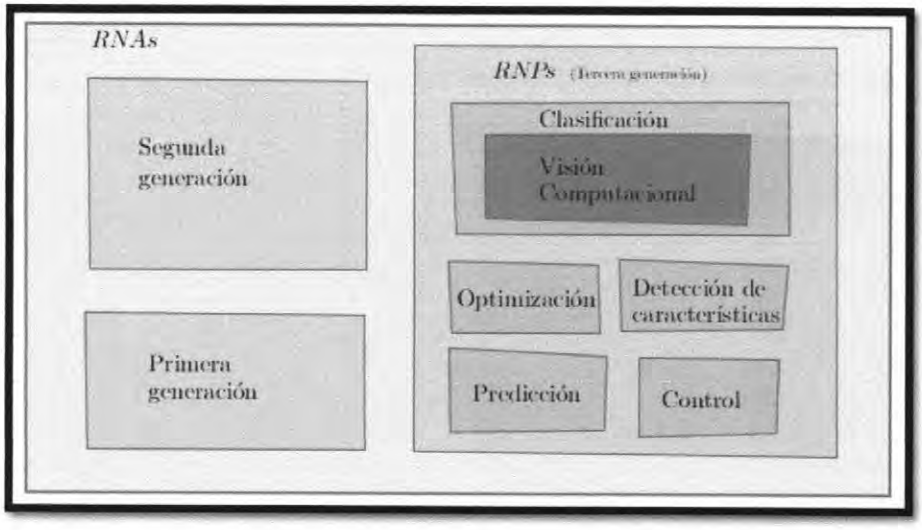


Figura 2.6 Delimitación del Problema para clasificación con aplicación en Visión Computacional [Hernández, 2017].

2.2. Trabajos relacionados

Se revisaron artículos recientes que fueron considerados inicialmente para analizar los modelos propuestos para el problema del camino más corto (SP) y otros problemas de optimización de trayectorias, así como para tomar medidas de comparación entre el modelo implementado y las técnicas tradicionales para esta investigación.

Finding the Shortest Path in the Shortest Time Using PCNN's [John, 1999]. Se evalúa la PCNN para determinar la trayectoria más corta en un laberinto de una forma

no determinística (tomando todos los caminos posibles) con tiempo constante por paso. Se discute la simulación secuencial de computadora digital con dicha complejidad de cada paso depende únicamente del número de neuronas, de lo contrario en paralelo la cantidad de tiempo de computación podría depender únicamente del tiempo para realizar una sola computación neuronal (dado un procesador por neurona).

A new algorithm for finding the shortest paths using PCNNs [Qu, 2007]. Estudian el rendimiento de la onda automática como característica de la PCNN para encontrar la solución al problema del laberinto, y a problemas de optimización combinatoria como el problema del trayecto más corto (SP). Se estudia multi-output model of pulse coupled neural networks / modelo de Múltiples-Salidas de Redes Neuronales Pulso-Acopladas (MPCNNs). Se presenta un nuevo algoritmo para encontrar el problema SP usando MPCNNs. Para analizar el desempeño de este modelo se utilizó un grafo ponderado.

Analysis of autowave characteristics for competitive pulse coupled neural Network and its application [Zhao, 2009]. Se presenta un nuevo algoritmo que incluye la característica de la propagación de la onda automática de la Competitive Pulse Coupled Neural Network / PCNN competitiva ($CPCNN_2$) para la solución del enrutamiento más corto en una red ponderada y SPT, el algoritmo está basado en el modelo PCNN de Johnson.

Applications of Pulse-Coupled Neural Networks [Ma, 2010]. Compendio de trabajos donde en conjunto se proponen dos algoritmos basados en PCNN para resolver problemas de optimización combinatoria como el de la ruta más corta (SP) y el del agente viajero (TSP). El Algoritmo AWNN (Figura 2.7) se propone para resolver el problema del trayecto más corto, representado por una matriz de adyacencia y al mismo tiempo utiliza caracteres de procesamiento paralelo y la onda automática de la PCNN por lo que es más eficiente para resolver problemas de optimización de trayectorias.

El segundo algoritmo se propuso para resolver el problema del camino más corto, que realiza la onda automática de la PCNN, también adquiere su propio procesamiento longitudinal. El Tristate Cascading Pulse Couple Neural Network / Red Neuronal Pulso-Acoplada de tri-estado en cascada (TCPCNN) mostrado en la Figura 2.8 realiza un procesamiento paralelo tanto en la dirección de ensanchamiento como en la dirección transversal. Aumenta la velocidad de búsqueda y asegura la precisión a través de la competencia acumulativa. Tiene poca dependencia de condiciones y parámetros iniciales.

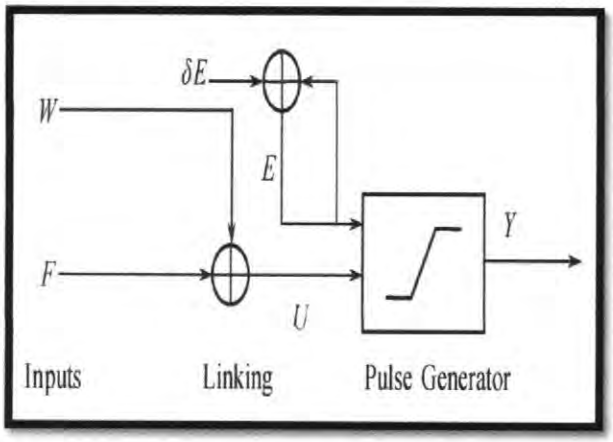


Figura 2.7 Modelo de la Neurona AWNN [Ma, 2010].

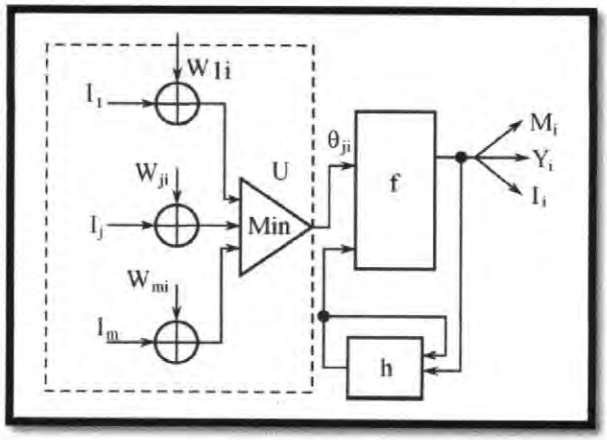


Figura 2.8 Modelo TCPCNN [Ma, 2010].

A Novel Algorithm for APSP Problem via a Simplified Delay Pulse Coupled Neural Network [Zhang, 2011]. Se introduce una Simplified Delay PCNN / Red Neuronal Pulso-Acoplada de Retardo Simplificado (SDPCNN) que simplifica el establecimiento de parámetros tradicionales de la red neuronal e introduce el concepto de retardo para el "all pairs shortest path" problem / problema de todos los pares más cortos (APSP) por sus siglas en ingles.

The application of the combinatorial optimization problems based on Preventive Feedback Pulse Coupled Neural Network [Ma, 2011]. Se introduce la retroalimentación preventiva basada en el teorema de desigualdad triangular para evitar

malas soluciones y se presenta la Preventive Feedback Pulse Coupled Neural Network/ Red Neuronal Acoplada por Pulsos de Respuesta Preventiva (PFPCNN) mostrado en la Figura 2.9, con el cual se reduce la complejidad en el espacio de búsqueda y aumenta la eficiencia y confiabilidad de la solución de búsqueda, dicho algoritmo se aplica a SP y la simulación de TSP.

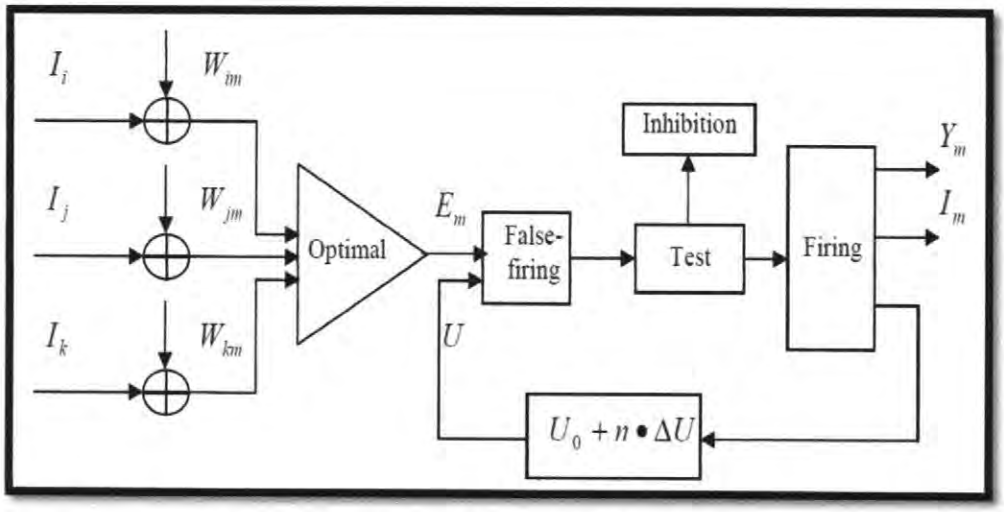


Figura 2.9 Modelo de la Neurona PFPCNN [Ma, 2011].

A novel neural network method for shortest path tree computation [Qu, 2012]. Se presenta un modelo de red neuronal acoplada a impulsos modificados (MCPCNN) para resolver problemas Shortest Path Tree (SPT). El modelo está organizado topológicamente con sólo conexiones laterales locales entre las neuronas. Esto demuestra que la onda generada en la red propuesta se propaga a velocidad constante, lo que garantiza que la onda se propague a lo largo de la ruta más corta desde un sólo nodo "inicio" a los otros nodos. La complejidad computacional sólo está relacionada con la longitud de la ruta más corta y es independiente del número de rutas existentes en el mapa.

Efficient Shortest-Path-Tree Computation in Network Routing Based on Pulse-Coupled Neural Networks [Qu, 2013]. Se propone un modelo modified of pulse-coupled neural networks / Modificado PCNN (M-PCNN) para el cálculo de Shortest path tree / Árbol del trayecto más corto (SPT) de manera eficiente para problemas de gran escala, además se propone un algoritmo dinámico que hace uso de la estructura del SPT previamente calculado para resolver problemas de enrutamiento cuando el estado de enlace se está actualizando dinámicamente en la red.

Self-adaptive autowave pulse-coupled neural network for shortest-path problem [Ma, 2013]. Se presenta un modelo de red neuronal pulso acoplada para el problema del trayecto más corto (SP) y el problema de los k caminos más cortos (KSP). Se propone el modelo Self-adaptive autowave pulse-coupled neural network / Red Neuronal Pulso-Acoplada de la Onda Automática Auto-adaptable (SAPCNN) para este problema, la onda automática generada por SAPCNN es propagada adaptativamente de acuerdo con el estado de la red actual, con la garantía de una propagación más efectiva en la búsqueda del trayecto más corto.

En el modelo SAPCNN su índice es de una dimensión, por lo que este algoritmo requiere de menos neuronas y no sólo encuentra soluciones globales sino también supera el algoritmo Dijkstra y el algoritmo MPCNN en el consumo de tiempo.

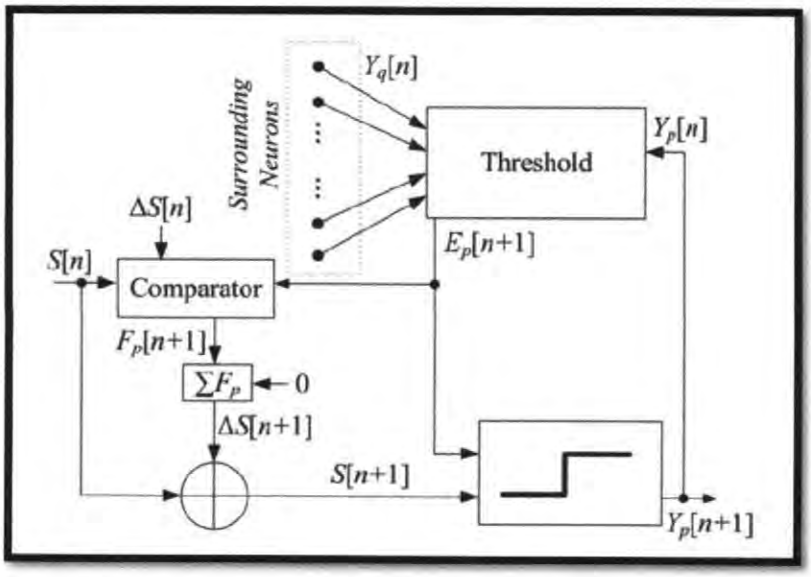


Figura 2.10 Estructura de SAPCNN [Ma, 2013].

Computing k shortest paths using modified pulse-coupled neural network [Qu, 2015]. Se presenta un modelo de Red Neuronal Modificada Continua Pulso-Acoplada (MCPCNN) para resolver los dos tipos de problemas KSP (k caminos más cortos) utilizando la característica de trasmisión de impulsos paralelos de las PCNN, donde se permiten los bucles en las trayectorias de solución: problemas de KSP de un sólo par y de fuente única. El modelo propuesto está organizado topológicamente con sólo conexiones laterales locales entre las neuronas. La onda *spiking* generada en la red se extiende hacia afuera con tiempos de recorrido proporcionales al peso de conexión entre las neuronas. Esto garantiza que la onda se propaga a lo largo de los caminos desde la fuente hasta todos los destinos. La complejidad computacional de los algoritmos sólo se relaciona con la longitud del camino más largo, independientemente del número de rutas existentes en el grafo.

Shortest path computation using pulse-coupled neural networks with restricted autowave [Qu, 2016]. Se propone una red neuronal acoplada a impulsos modificada para calcular trayectos más cortos (SP) y superar algunas deficiencias de los modelos tradicionales de PCNN. En dicho modelo la innovación principal es la nueva unidad de

control diseñada, que produce un mecanismo de control local, llamado en adelante / fuera-hacia atrás, para ajustar la propagación de la auto onda. Como resultado, su onda automática está restringida a propagarse en cada dirección diferente con la velocidad correspondiente diferente, que puede buscar caminos más cortos de manera más eficiente. Una serie de experimentos mostraron que el modelo $CPCNN_1$ puede reducir significativamente el espacio de búsqueda y el costo computacional para encontrar los caminos más cortos.

Debido al mecanismo especial, el modelo de CPCNN usualmente, pero no siempre, produce en una trayectoria globalmente óptima. Mientras tanto, la pérdida de óptimalidad es aceptable.

A time-delay neural network for solving time-dependent shortest path problem [Huang, 2017]. Se propone un entorno de trabajo de red neural de retardo de tiempo (TDNN) para resolver el problema de trayecto más corto dependiendo del tiempo (TD-SP). A diferencia de la red neural convencional, la red neural de retardo de tiempo es una red de realimentación con neuronas de onda automática sin necesidad de ningún entrenamiento. Los resultados experimentales muestran que el TDNN propuesto conduce a un mejor rendimiento que el PCNN convencional. El TDNN propuesto también se comparó con la calidad de los enfoques clásicos como Dijkstra y PCNN. En general la complejidad del TDNN depende únicamente del número de neuronas cuando se ejecuta el TDNN en la simulación por ordenador. Prácticamente, la aparente limitación de TDNN es que es difícil tratar con la red independiente del tiempo en los entornos de incertidumbre.

2.3. Discusión del estado del arte

El proyecto desarrollado en [Cárdenas, 2015] se logra la primera implementación de PCNN en CENIDET, donde realiza una comparación del desempeño de la segmentación utilizando PCNN y técnicas tradicionales. En [Zarate, 2015] se implementa la PCNN y algunas variantes como ICM y SCM para la extracción de características en imágenes digitales para realizar un estudio comparativo. En [Ortiz, 2017] se realiza una comparación entre técnicas tradicionales y PCNN para llevar a cabo la detección y el filtrado de Ruido Impulsivo y Gaussiano en imágenes digitales y compara a través de métricas con las técnicas tradicionales. En [Hernández, 2017] se trabaja con el modelo Izhikevich adaptado para la clasificación, se propone un algoritmo de aprendizaje este modelo y se experimenta con distintos tipos de bases de datos en el área de Visión Computacional.

A continuación, en la Tabla 2.1 se muestra un resumen de las investigaciones realizadas en CENIDET:

Tabla 2.1 Resumen de antecedentes institucionales.

Investigación	Objetivo	Técnicas	Observaciones
[Cárdenas, 2015]	Estudiar, analizar, implementar, experimentar y evaluar la RNA tipo PCNN, dentro del dominio de Visión Artificial en el área de segmentación de imágenes.	PCNN ICM Canny Sobel	Utiliza una base de datos propia.
[Zarate, 2015]	Estudiar, analizar, implementar, experimentar y evaluar la RNA tipo Pulsante, dentro del dominio de Visión Artificial para problemas de reconocimiento de patrones a través de la extracción de características de una imagen.	PCNN ICM SCM	Utiliza una base de datos generada en el CENIDET, trabaja con 30 imágenes rotadas, escaladas, con ruido Gaussiano y ruido impulsivo.

[Ortiz, 2017]	Implementar y evaluar la RNA tipo PCNN para problemas de detección de ruido impulsivo y gaussiano en imágenes digitales.	PCNN ICM PCNNI SCM	Utiliza una base de datos de prueba estándar en formato BMP y escala de grises como Lena, Peppers, Babuino, Cameraman y Grass.
[Hernández, 2017]	Analizar, comprender, implementar y experimentar la RNA tipo pulsante, así como aplicarlas dentro del dominio de Visión Artificial para problemas de reconocimiento de patrones y clasificación, implementado un par de modelos, en particular el modelo de Izhikevich.	Izhekevich ICM PCNN SCM	Se utilizan bases de datos: Académicas, de imágenes y del cerebro.

Del análisis de los trabajos institucionales se estudiaron los modelos PCNN, ICM y SCM reportados en [Cárdenas, 2015], [Zarate, 2015], [Ortiz, 2017], [Hernández, 2017] de los cuales al inicio de la implementación de los modelos variantes de PCNN se eligió el modelo ICM para replicación y observación de la característica principal de este tipo de RNAP (la propagación de la onda automática) en el procesamiento de imágenes.

De [Cárdenas, 2015] se tomó la idea de utilizar una base de datos propia para ampliar el número de diferentes bases de datos en la fase de experimentación que complemento la creación del "Generador de casos de prueba" basado en [Qu, 2016].

En general los trabajos institucionales permitieron comprender la gran diversidad de aplicaciones a problemas de segmentación, extracción de características, detección de ruido impulsivo y gaussiano, así como problemas de reconocimiento de patrones y clasificación dentro del área de Visión Artificial.

En conclusión, estos trabajos no abordan problemas de optimización con RNAP, ya que las implementaciones de PCNN se aplican al área de Visión Artificial por lo que la

presente investigación se orientó a la solución del problema SP como parte de problemas de optimización de trayectorias.

A continuación, en la Tabla 2.2 se muestra un resumen de los trabajos relacionados con el proyecto:

Tabla 2.2 Resumen de los trabajos relacionados

Artículo	**Problema**	**Algoritmo propuesto**	**Algoritmo de comparación**	**Observaciones**
[John, 1999]	Laberinto	PCNN	----------	Se descubre que se puede utilizar la PCNN en el problema del laberinto de forma no determinística.
[Qu, 2007]	Laberinto	MPCNN	----------	Uso de un grafo ponderado para resolver problemas de optimización de trayectorias.

Tabla 2.2 Resumen de los trabajos relacionados (continuación de la Tabla 2.2)

Artículo	**Problema**	**Algoritmo propuesto**	**Algoritmo de comparación**	**Observaciones**
[Zhao, 2009]	SPT	$CPCNN_2$	Dijkstra	Se compara el algoritmo propuesto con el número de iteraciones contra Dijkstra.
[Qu, 2009]	SP	MPCNN	Dijkstra A* AG	Se comparan los resultados con algoritmos genéticos.
[Ma, 2010]	SP TSP	AWNN TCPCNN	----------	Se incorporan dos algoritmos para problemas de optimización.

[Zhang, 2011]	APSP	SDPCNN	K*	Se introduce el concepto de retardo para la simplificación del establecimiento de parámetros.
[Ma, 2011]	SP TSP	PFPCNN	TCPCNN	Incorporan la retroalimentación preventiva basada en el teorema de la desigualdad triangular.
[Qu,2012]	TSP	MCPCNN	Dijkstra	El algoritmo calcula el SPT en un grafo estático.
[Qu, 2013]	SPT	M-PCNN	Dijkstra	Se propone un algoritmo el cual recalcule el SPT en un grafo dinámico. Experimentación con grafos de hasta 8000 nodos.
[Ma, 2013]	SP KSP	SAPCNN	Dijkstra MPCNN	Integran la auto-adaptación de la propagación de la onda automática de la PCNN.
[Qu, 2013]	SPT	M-PCNN	Dijkstra	Se propone un algoritmo el cual recalcule el SPT en un grafo dinámico. Experimentación con grafos de hasta 8000 nodos.

Tabla 2.2 Resumen de los trabajos relacionados (continuación de la Tabla 2.2)

Artículo	**Problema**	**Algoritmo propuesto**	**Algoritmo de comparación**	**Observaciones**
[Ma, 2013]	SP KSP	SAPCNN	Dijkstra MPCNN	Integran la auto-adaptación de la propagación de la onda automática de la PCNN.
[Qu, 2015]	KSP	MCPCNN	Kn	Se experimenta con un caso real perteneciente al grafo de New York.

[Qu, 2016]	SP	$CPCNN_1$	A* MPCNN [Qu, 2009]	Experimentación con grafos de 1000 hasta los 8000 nodos. Se menciona la idea de la creación de un "Generador de grafos". Bases de datos reales.
[Huang, 2017]	TD-SP	TDNN	Dijkstra PCNN	Se incorpora una unidad de ventana de tiempo y una unidad de auto onda.

Del análisis de los trabajos relacionados se detectaron 4 algoritmos propuestos derivados de PCNN para el problema del camino más corto (SP): [Qu, 2016], [Ma, 2013], [Ma, 2011] y [Ma, 2010], dentro de los cuales el modelo $CPCNN_1$ en [Qu, 2016] no siempre produce una trayectoria óptima global por lo cual se excluye fuera para ser implementado.

De los 3 modelos restantes para el problema de interés (SP) se analizaron de acuerdo a lo reportado en las referencias:

- En [Ma, 2013] se propone el algoritmo SAPCNN para el problema SP y KSP y se realiza una comparación contra Dijkstra y el algoritmo MPCNN [Qu, 2007].
- En [Ma, 2011] se propone el algoritmo PFPCNN para el problema SP y TSP, se comparó con el TCPCNN de [Ma, 2010] para TSP.
- En [Ma, 2010] se proponen dos algoritmos (AWNN y TCPCNN) para el problema SP y TSP, donde se detecta que el algoritmo AWNN no ha sido comparado para SP con respecto a las técnicas tradicionales de optimización de trayectorias.

En base a la falta de un estudio comparativo del algoritmo AWNN [Ma, 2010] se eligió para ser implementado y experimentado para el SP.

Para el diseño e implementación de pruebas se tomaron algunas ideas de las siguientes referencias:

- De [Qu, 2016] se tomó la idea de crear el "Generador de casos de prueba" de la sección 7.1.1 (Dataset generation), que fue desarrollado en lenguaje C++ el cual escribe en archivos ".txt" matrices de costos pertenecientes a Grafos No Dirigidos

descritos ampliamente en el capítulo 5 de la tesis (EXPERIMENTACIÓN Y RESULTADOS), además de mencionar la base de datos de casos reales de la referencia [Brinkhoff, 2002] utilizados como el tercer tipo de base de datos para la experimentación de este proyecto de tesis.

- De [Huang, 2017], [Qu, 2013], [Qu, 2012] y [Zhao, 2009] se justifica la selección del algoritmo Dijkstra como técnica tradicional de problemas de optimización de trayectorias para la comparación con el algoritmo de RNAP, para el problema SP al ser usualmente utilizado debido a sus buenos resultados en tiempo de procesamiento, así como en la calidad de la ruta óptima global.

Para la fase de experimentación se tomaron casos de prueba planteados en la literatura perteneciente a grafos no dirigidos, los cuales se mencionan en las siguientes referencias:

- De [Ma, 2010] el grafo de 5 nodos de la Figura 8.9 (A simple undirected weighted graph).
- De [Qu, 2012] el grafo de 9 nodos de la Figura 3. (An example network).
- De [Ma, 2013] el grafo de 10 nodos de la Figura 4. (A symmetric weighted graph with ten nodes and twenty edges).
- De [Ma, 2011] el grafo de 12 nodos de la Figura 4. (An undirected weighted gragh).
- De [Qu, 2016] el grafo de 13 nodos de la Figura 1. (An example of network).

Capítulo 3

Marco teórico

En este capítulo se incluyen los conceptos recolectados durante la investigación, se mencionan y describen tres temas principales, que son optimización de trayectorias, teoría de grafos y Redes Neuronales Artificiales Pulsantes (RNAP) mediante los cuales se desarrolló el proyecto.

3.1 Optimización de trayectorias

La optimización consiste en la selección de una alternativa mejor, en algún sentido, que las demás alternativas posibles. Los problemas de optimización se componen generalmente de estos tres componentes [Ramos, 2010]:

- Función objetivo. - es la medida cuantitativa del funcionamiento del sistema que se desea optimizar (maximizar o minimizar).
- Variables. - representan las decisiones que se pueden tomar para afectar el valor de la función objetivo.
- Restricciones. - representan el conjunto de relaciones (expresadas mediante ecuaciones o inecuaciones) que ciertas variables están obligadas a satisfacer.

Resolver un problema de optimización consiste en encontrar el valor que debe tomar las variables para hacer óptima la función objetivo satisfaciendo el conjunto de restricciones.

La optimización combinatoria es una rama de la optimización, dentro de esta rama se encuentran los problemas de optimización de trayectorias.

Este tipo de problemas son, dadas las condiciones restringidas, considerar todos los riesgos sintéticamente, y encontrar las variables que hace la función objetivo sea mayor o menor. La teoría de grafos siempre se utiliza como la base matemática para resolver este problema [Ma, 2010].

Algunos problemas de optimización de trayectorias se presentan de la siguiente manera [Ma, 2010]:

- **Problema del Clique máximo**. - En un grafo es un conjunto de vértices y cada par de ellos está conectado por un borde a condición de que el clique máximo no sea la parte de ningún otro clique. El clique máximo es el problema de encontrar el clique más grande que contiene la mayoría de los vértices en un grafo dado.

- **Traveling Salesman Problem (TSP)**. - Este problema se define como un agente viajero que quiere visitar cada una de las ciudades de la lista exactamente una vez y regresar a la ciudad de partida (Figura 3.1). El objetivo de este problema es encontrar la ruta menos costosa que toma el agente viajero. Dado el número de ciudades n, la matriz D representa la distancia entre dos ciudades, y resolver el problema significa encontrar la mejor combinación para hacer la menor distancia desde la ciudad de partida hasta la ciudad final.

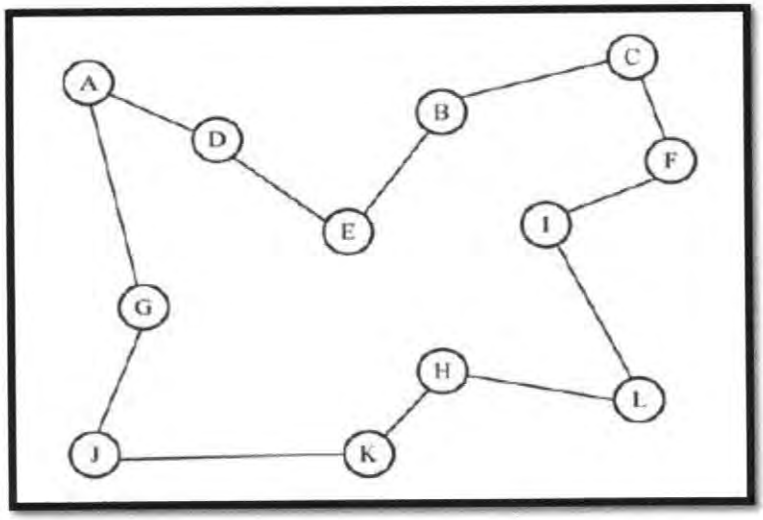


Figura 3.1 Ejemplo de TSP en un grafo no dirigido [Ma, 2010]

- **Shortest Path (SP) Problem**. - El problema del camino más corto es encontrar el camino más corto entre la posición inicial y el destino en una red dada (Figura 3.2).

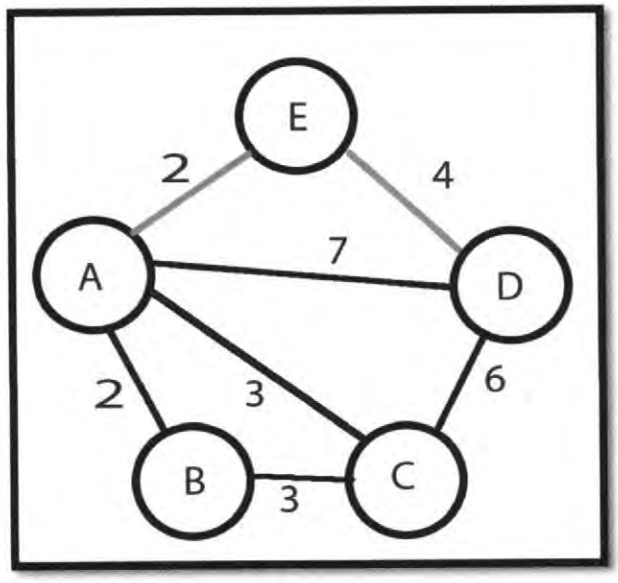


Figura 3.2 Ejemplo de SP de A-E en un grafo.

3.1.1 Algoritmos tradicionales para SP

Dentro de los algoritmos para la búsqueda de rutas se encuentran los exhaustivos y los "inteligentes" que permiten determinar el camino más corto o al menos uno optimizado, sin tener que comprobar todas las rutas posibles mostrados en la Tabla 3.1

Tabla 3.1 Algoritmos tradicionales de búsquedas de rutas.

Algoritmo ---------- Característica	Profundidad	Anchura	Bellman-Ford	Dijkstra	A*
Tipo	Exhaustivo	Exhaustivo	"Inteligente"	"Inteligente"	"Inteligente"
Funcionamiento	Si se bloquea se vuelve a la última intersección que se ha encontrado y probamos una nueva ruta.	La búsqueda en anchura permite comprobar nivel por nivel los distintos nodos accesibles.	Se aplica, a tantas veces como número de nodos menos 1 el mismo bucle.	Permite escoger de manera más inteligente el orden de aplicación de las distancias	Devuelve generalmente el mejor camino o uno de los mejores.
Ventajas	No utilizan Conocimiento Previo.	No utilizan Conocimiento Previo.	Permiten determinar la trayectoria más corta sin comprobar todas las rutas posibles.	Permiten determinar la trayectoria más corta sin comprobar todas las rutas posibles.	No comprueba todas las rutas.
Desventajas	Exploran todas las rutas posibles. No permite saber si se ha encontrado la trayectoria más corta.	Exploran todas las rutas posibles. No permite saber si se ha encontrado la trayectoria más corta.	Todos los caminos más cortos se calculan, y no solo el que nos interesa.	Funciona solamente si todas las distancias son positivas.	No puede asegurar que la trayectoria encontrada sea la más corto.

3.2 Teoría de grafos

La teoría de grafos es considerada a menudo una de las ramas más modernas de las matemáticas, aunque tiene un origen en 1736 cuando Euler publicó el primer trabajo de lo que hoy se define como Teoría de grafos [Vieites, 2014].

El ejemplo clásico es el problema de los siete puentes [Bogomolny, 2017], se dice que hay un río llamado Pregel que divide la ciudad de Konigsberg en cuatro masas de tierra $A, B, C, y\ D$; siete puentes conectan las diversas partes de la ciudad y algunos ciudadanos curiosos que viven en esta ciudad jugaron como una forma de recreación en la que intentaron tomar un viaje a través de los siete puentes y volver a la masa de tierra en que comenzaban sin tener que cruzar cualquier puente más de una vez. Todos los que intentaron terminaron con el fracaso, incluyendo al matemático suizo Leonhard Euler (1707-1783), que es un genio famoso del siglo XVIII. Euler consideró las varias partes de la ciudad como un punto, y los puentes que conectaban dos partes como una línea y dedujo el grafo [Ma, 2010]. El método de Euler había sentado una base para la topología y la optimización combinatoria (Figura 3.3).

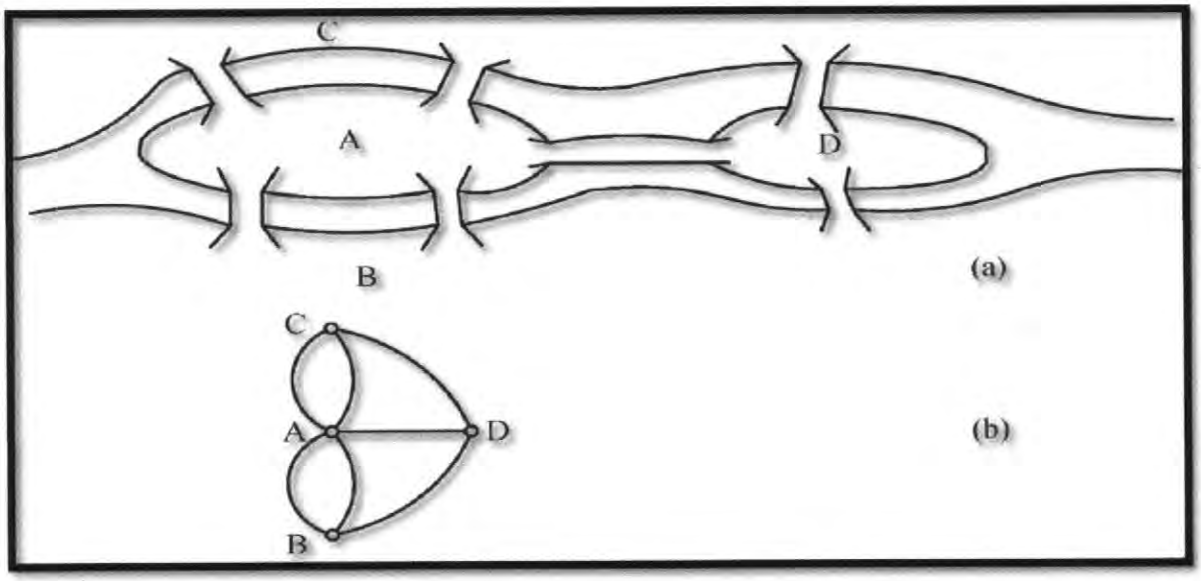


Figura 3.3 A) Problema de los siete puentes, (b) Grafo basado en siete problemas puentes [Ma, 2010]

Según Euler se puede definir un grafo como $G = (V(G), E(G))$ para describir el problema de siete puentes en la teoría de grafos:

$$G = \big(V(G), E(G)\big). \quad (1)$$

$$V(G) = \{A, B, C, D\}. \quad (2)$$

$$E(G) = \{AC, CA, AB, BA, AD, BD, CD\}. \quad (3)$$

3.2.1 Representación de los grafos

Aunque en general al representar datos de la vida real mediante un grafo los vértices suelen llevar asociada información, por lo que se supone que esa información se almacena en una lista indexada y por lo tanto se hace referencia a los vértices utilizando únicamente a la manera de almacenar las aristas.

Las dos representaciones principales de grafos son las siguientes [Aichholzer, 2005]:

- Matriz de adyacencia (MA).- Se utiliza una matriz de tamaño $n\,x\,n$ donde las filas y las columnas hacen referencia a los vértices para almacenar en cada casilla la longitud entre cada par de vértices del grafo. La celda $MA\,(i,j)$ almacena la longitud entre el vértice i y el vértice j, si su valor es infinito significa que no existe arista entre esos vértices, y $MA\,(i,j) = 0$.

- Lista de adyacencia (LA).- Se utiliza un vector de tamaño n (un elemento por cada vértice) donde $LA\,(i,j)$ almacena la referencia a una lista de los vértices adyacentes a i en una red esta lista almacenará también la longitud de la aristas que va desde i al vértice adyacente.

Se muestra un ejemplo (Figura 3.4 A), B) y C)) de un grafo no dirigido y su representación en una MA y una LA:

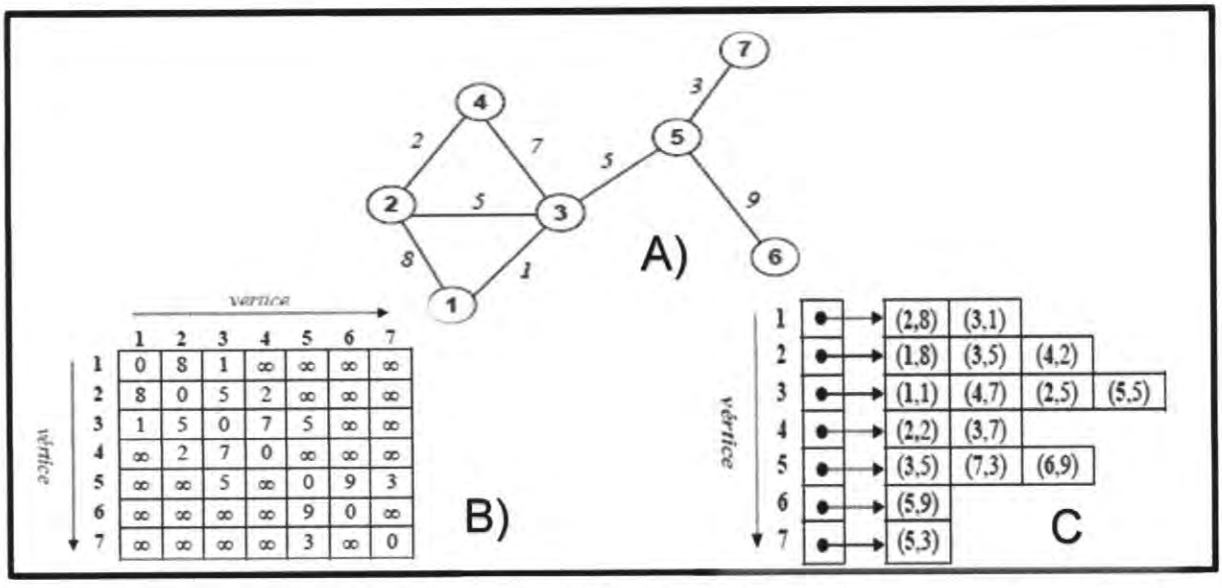


Figura 3.4 A) Grafo no dirigido, B) MA del grafo no dirigido y C) LA del grafo no dirigido [Aichholzer, 2005]

3.2.2 Tipos de grafos

Los grafos se pueden clasificar dependiendo de si el orden de los vértices en las aristas importa o no [Aichholzer, 2005], se dice que:

- Grafo dirigido.- el orden importa, (i,j) es diferente de (j,i), el que el vértice i esté conectado con el vértice j no implica que el vértice j esté conectado con i.
- Grafo no dirigido. El orden no importa, (i,j) es igual a (j,i).

La representación de un grafo dirigido y no dirigido se puede observar la Figura 3.5.

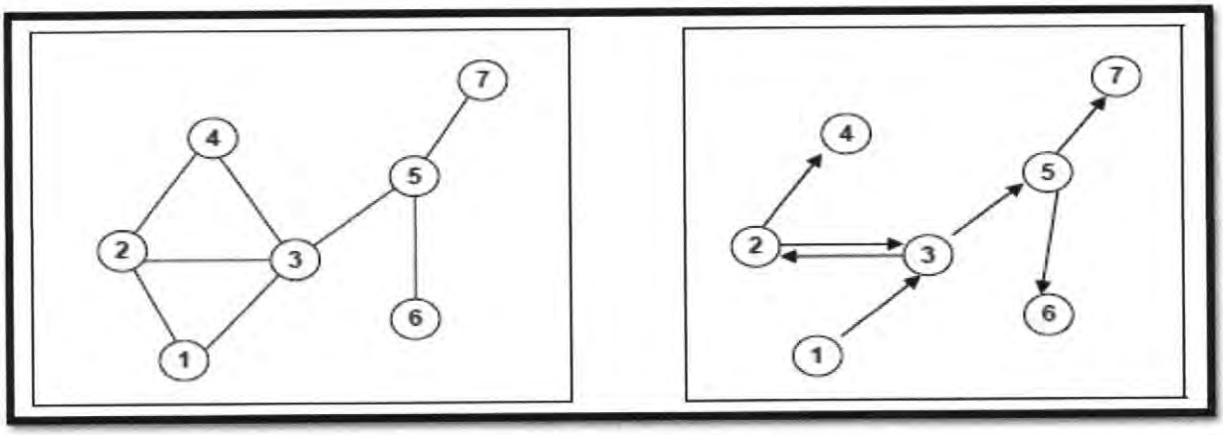


Figura 3.5 Grafo no dirigido y grafo dirigido.

3.3. Redes Neuronales Artificiales Pulsantes

La inteligencia Artificial ha tenido como objetivo, durante mucho tiempo, simular la inteligencia humana y obtener un sistema artificial con capacidad de reflexionar, de tomar decisiones y de aprendizaje. Los investigadores se han interesado rápidamente en el funcionamiento del cerebro para reproducirlo. De este modo es como aparecen las primeras neuronas artificiales por Mac Culloch y Pitts en 1943 [Mathivet, 2015].

Las RNA se definen como una estructura compuesta por unidades básicas las cuales están interconectadas mediante distintos pesos que le dan la capacidad de almacenar conocimiento experimental y hacerlo utilizable [Arista, 2012].

Durante el estudio de las RNA se han tenido diferentes filosofías de diseño, reglas de aprendizaje y funciones de construcción de las respuestas para estas, por lo que de acuerdo a su aparición y comportamiento se dividen en diferentes generaciones.

Las RNA de tercera generación o también llamadas "RNAP" se asemejan mucho más a las neuronas del sistema biológico, ya que incorporan el concepto del tiempo [Vázquez, 2010]. Los comienzos de las RNAP se remonta al año 1952 cuando Hodking y Huxley desarrollaron un modelo matemático que describe la dinámica neuronal en términos de

la activación e inactivación de conductividad de voltaje; dentro de este concepto se han desarrollado diferentes modelos neuronales que tienen un realismo más cercano a las redes neuronales biológicas ya que producen la codificación de la información en forma de pulsos en el tiempo, los cuales son llamados "patrones de disparo", a continuación se muestran algunos de estos patrones encontrados en las neuronas biológicas (Figura 3.6):

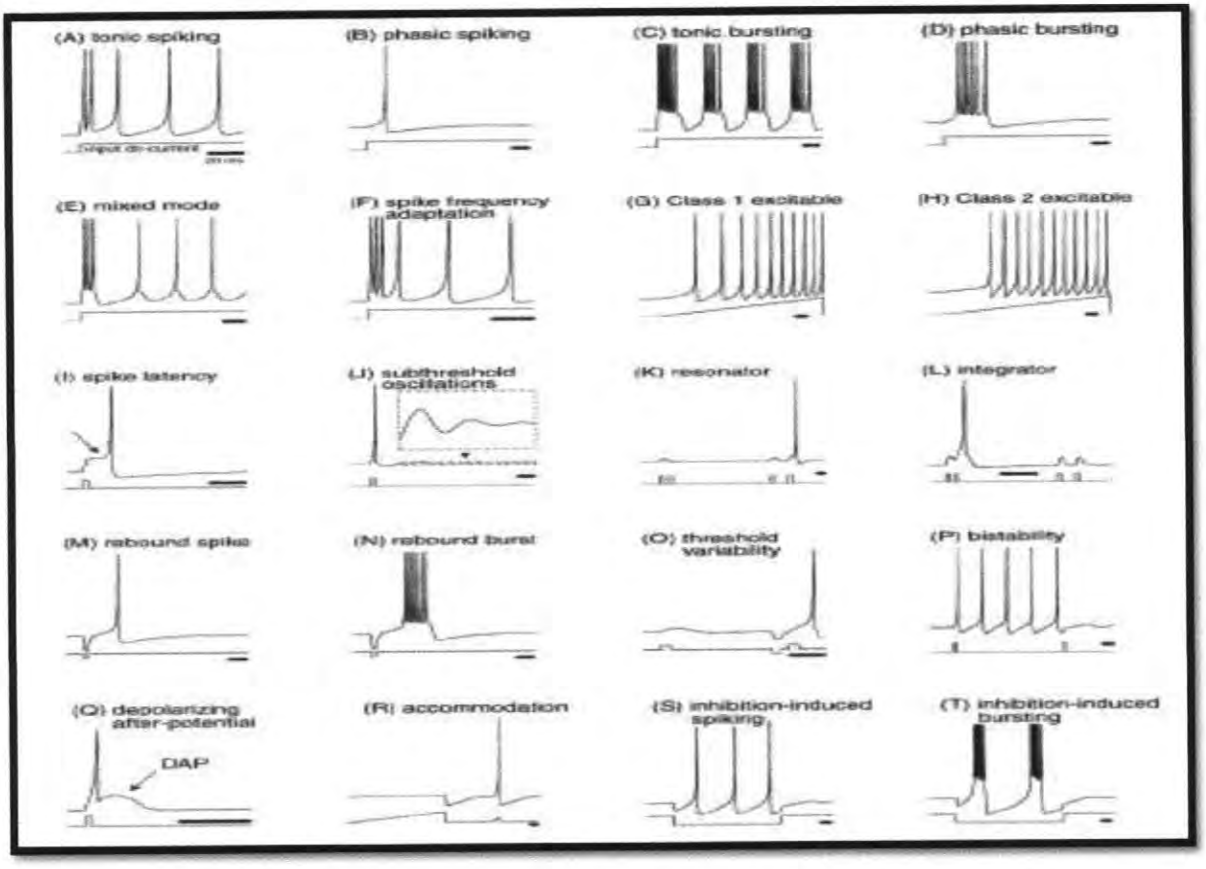


Figura 3.6 Ejemplos de algunos patrones de disparo de una neurona [Izhikevich, 2004].

3.3.1. Modelo de Red Neuronal Pulso-Acoplada (PCNN)

En 1989 Eckhorn descubrió que las oscilaciones en la corteza visual del gato generan imágenes binarias de las que se pueden extraer diferentes características de las impresiones visuales para generar una imagen.

En este modelo (Figura 3.7), cada neurona posee dos entradas: *feeding* (alimentación) y *linking* (conexión). El *feeding* recibe tanto los estímulos externos como estímulos locales, mientras que el *linking* únicamente recibe estímulos locales. Después ambas entradas son combinadas para crear el voltaje de membrana Um, el cual es comparado con un Umbral local θ.

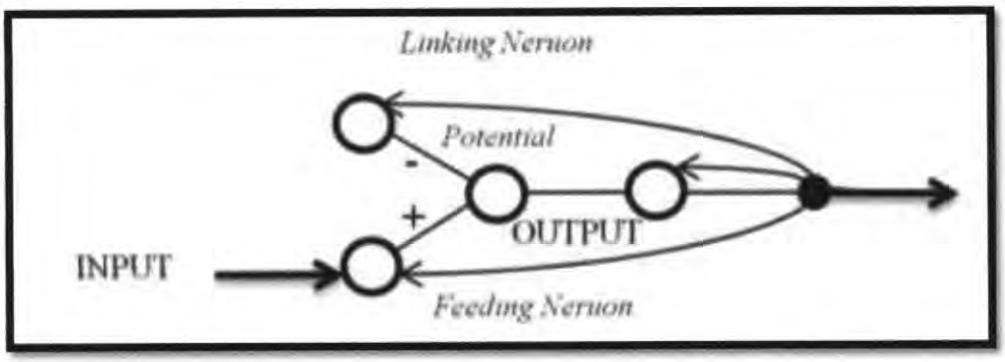


Figura 3.7 Neurona tipo Eckhorn [Lindblad, 2005]

Cada neurona de la red neuronal recibe entradas de sus propios estímulos y también de sus vecinos, definiéndose un así un radio de alimentación. Adicionalmente, los datos del *linking*, como las salidas de otras neuronas, son agregadas a la entrada.

El modelo introducido se expresa por medio de las siguientes ecuaciones [Eckhorn, 1990]:

$$U_{m,k}(t) = F_k(t)[1 + L_k(t)] \quad (4)$$

$$F_k(t) = \sum_{i=1}^{N}\left[w_{ki}^{f} Y_i(t) + S_k(t) + N_k(t)\right] \otimes I(V^a, \tau^a,$$

(5)

$$L_k(t) = \sum_{i=1}^{N}\left[w_{ki}^{l} Y_i(t) + N_k(t)\right] \otimes I(V^l, \tau^l, t) \quad (6)$$

$$Y_k(t) = \begin{cases} 1 \text{ si } U_{m,k}(t) \geq \Theta_k(t) \\ 0 \qquad \text{en otro caso} \end{cases} \quad (7)$$

Donde en general

$$X(t) = Z(t) \otimes I(v, \tau, t) \quad (8)$$

es

$$X[n] = X[n-1]e^{-t/\tau} + V\,Z[n] \tag{9}$$

En el que Um,k representa el voltaje de membrana, Fk la señal de *feeding*, Lk la señal de linking, I es el estímulo externo, N es el número de neuronas, w son los pesos sinápticos, Y es la salida binaria, Θ es el umbral, V es una constante de ajuste y τ es una constante de tiempo. Los rangos típicos de los valores dados en [Lindblad, 2005] son $\tau^a = [10,15]$, $\tau^l = [0.1, 1.0]$, $V^a = 0.5$, $V^l = [5,30]$, $V^s = [50,70]$, $\Theta_o = [0.5,1.8]$.

Este modelo original de Red Neuronal Pulsante o PCNN tiene algunas limitaciones en la práctica para el procesamiento de imágenes, por ejemplo, el gran número de conexiones de retroalimentación existentes entre las neuronas y que los parámetros del filtro integrador son difíciles de establecer. Para reducir estos problemas, Ranganath y Kuntimad en 1995, removieron el filtro integrador del campo receptivo de *feeding* y obtuvieron un modelo simplificado que se puede aplicar al procesamiento de imágenes [Ranganath, 1995].

El modelo de PCNN modificado para efectos de simulación se expresa como [Ma, 2010]:

$$F_{ij}[n] = e^{-\alpha F}F_{ij}[n-1] + V_F \textstyle\sum_{kl} M_{ijkl}Y_{kl}[n-1] + S_{ij} \tag{10}$$

$$L_{ij}[n] = e^{-\alpha L}L_{ij}[n-1] + V_L \textstyle\sum_{kl} W_{ijkl}Y_{kl}[n-1] \tag{11}$$

$$U_{ij}[n] = F_{ij}[n](1 + \beta L_{ij}[n]) \tag{12}$$

$$E_{ij}[n] = E_{ij}[n-1]e^{-\alpha E} + V_E Y_{ij}[n-1] \tag{13}$$

$$Y_{ij}[n] = \begin{cases} 1 \text{ si } U_{ij}[n] > \mathrm{E}_{ij}[n] \\ 0 \quad \text{en otro caso} \end{cases} \tag{14}$$

donde $\alpha F, \alpha L, y\ \alpha E$ son constantes de tiempo; $VF, VL\ y\ VE$ son constantes de ajuste y β es la fuerza de relación de la PCNN. Cada neurona es denotada con los índices (i,j), y una de sus neuronas vecinas con (k,l). F es el *feeding* combinado con L el *linking*, juntos originan la actividad interna de la neurona U. La neurona recibe la señal de entrada $I(i,j)$ por el *feeding* y la sinapsis M, y cada neurona está conectada con sus vecinas de modo tal que la señal de salida de una neurona modula la actividad de sus vecinas por el *linking* y la sinapsis W. El pulso es capaz de retroalimentar y modular el umbral E por medio del filtro integrador, incrementando el umbral por medio de la magnitud de VE que decrece con la constante de tiempo αE. De este modo se pueden distinguir tres partes del modelo, la receptiva, la de modulación y la de generación del pulso como se observa a continuación (Figura 3.8).

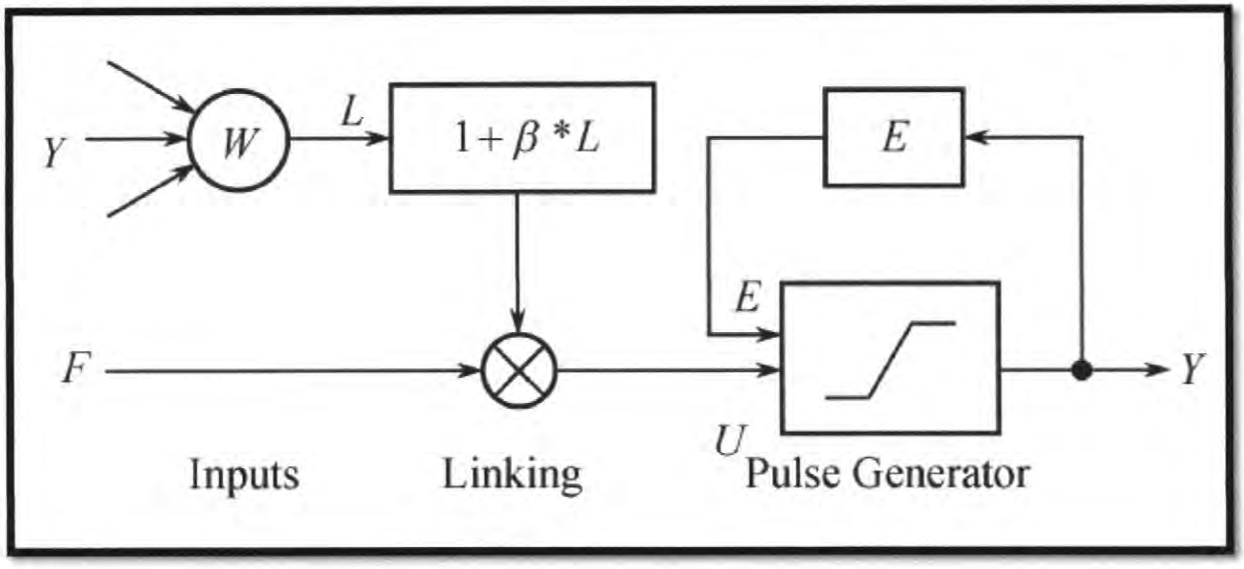


Figura 3.8 Modelo de PCNN [Zhang, 2007]

Las PCNN no requieren entrenamiento y su única función es clasificar los pixeles por niveles de intensidad, en ella cada pixel de la imagen corresponde a una neurona, tienen una fuerte ventaja en el suavizado de imágenes y han sido utilizadas en combinación con el filtro de mediana, el morfológico y el filtro promedio con resultados superiores a diversas técnicas tradicionales de filtrado.

De esta manera el modelo de las PCNN es un sistema biónico que emula las neuronas de la corteza visual de los mamíferos y ha sido aplicada en variedad de dominios del procesamiento de imágenes tales como la remoción de ruido, la detección de objetos, la optimización, el adelgazamiento de la imagen, la segmentación y la remoción de sombras, por lo que ha atraído la atención de muchos investigadores del área de la Visión Artificial [Shao-Fa, 2010].

3.3.2. Modelo ICM

El modelo cortical de intersección (ICM) es una versión simplificada de la PCNN [Ekblad, 2004].

El modelo matemático del ICM se describe como:

$$F_{ij}[n] = fF_{ij}[n-1] + \sum_{k,j} w_{i,j,k,l} Y_{i,j}[n-1] + S_{i,j} \tag{15}$$

$$Y_{ij}[n] = \begin{cases} 1 \text{ si } F_{ij}[n] > \mathrm{T}_{ij}[n-1] \\ 0 \qquad \text{en otro caso} \end{cases} \tag{16}$$

$$T_{ij}[n] = gT_{ij}[n-1] + hY_{ij}[n-1] \tag{17}$$

Donde f, g y h son constantes, y $f, g < 1.0$, con $f < g, y\ h$ es un valor grande.

El modelo ICM comparado con el modelo PCNN es más rápido debido a su reducido número de ecuaciones.

3.3.3. Modelo AWNN

El modelo AWNN (Figura 3.9) el cual está especialmente diseñado para resolver los problemas de las trayectorias óptimas tales como el problema SP y TSP. En AWNN el problema se representa por una matriz de adyacencia de grafos y al mismo tiempo, utiliza caracteres de procesamiento paralelo y la onda automática de la PCNN.

En AWNN cada neurona individual esta denotada por los índices (i,j) y puede ser activada solo una vez por los vecinos cuando la onda automática avanza. Como se muestra en las ecuaciones cada neurona está dividida en tres componentes: entradas $F_{ij}[n]$ y W_{ij}, la actividad interna $U_{ij}[n]$ y el umbral dinámico $E[n]$.

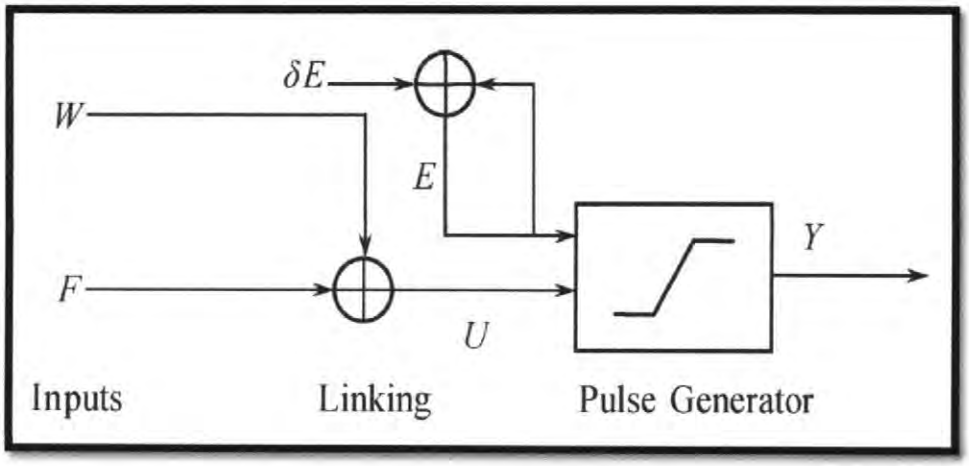


Figura 3.9 Modelo de la Neurona AWNN [Ma, 2010].

$$F_i[n] = F_h[n] + W_{hj}; \tag{18}$$

$$U_{ij}[n] = F_i[n] + W_{ij}; \tag{19}$$

$$Y_{ij}[n] = \begin{cases} 1 & E[n] \geq U_{ij}[n], \\ 0 & Otherwise; \end{cases} \tag{20}$$

$$E[n] = E[n-1] + \delta E. \quad (21)$$

El tamaño de F es el mismo que el número de nodos de problemas de optimización combinatoria, F se utiliza para registrar las longitudes acumuladas de la trayectoria de la neurona del comienzo al ser activada.

W_{ij}, que es descrito por una matriz de adyacencia, es el estímulo de entrada. Los valores en la matriz son las distancias entre todos los nodos, $U_{ij}[n]$ registra las longitudes de ruta desde la neurona de inicio hasta la neurona de disparo, $E[n]$ sólo tiene relación con los tiempos iterativos, y se incrementa paso a paso naturalmente, $Y_{ij}[n]$ es la salida de la neurona (i,j) y juega un papel de ruta-registro. Si $Y_{ij}[n]$ es 1, significa que la ruta ya ha pasado o acaba de llegar a la neurona (i,j). Finalmente se utiliza para encontrar el recorrido de la ruta.

Para decidir la condición de terminación iterativa del AWNN, se introduce una secuencia M . En M, los nodos disparados se definen como 1, mientras que los nudos sin activar se establecen en 0. Como resultado, la condición de terminación iterativa de SP es que el nodo final se dispara.

3.4. Discusión

Los problemas de optimización de trayectorias tienen ciertas características las cuales los hacen un problema de gran complejidad, entre los algoritmos tradicionales para el problema SP se encuentran los exhaustivos y los "inteligentes", entre los segundos se han propuesto métodos basados en RNAP los cuales están especialmente diseñados para combinar la teoría de grafos al incorporar la matriz de adyacencia y al mismo tiempo, utilizar caracteres de procesamiento paralelo y la onda automática de la PCNN. Mediante

la presente investigación se pretende evaluar las RNAP en el dominio de optimización de trayectorias para la T.Ó.S-Ó (el problema SP) en una red dada.

Capítulo 4

Red Neuronal Pulsante especializada para el problema del camino más corto

En este capítulo se describe el modelo AWNN, una adaptación del algoritmo para el problema del camino más corto, detalles de la implementación en grafos no dirigidos, la explicitación del conocimiento para la reconstrucción de la ruta más corta y una discusión sobre el modelo AWNN.

4.1. Modelo AWNN

El modelo AWNN (Figura 4.1) está especialmente diseñado para resolver los problemas de optimización de trayectorias tales como SP y TSP. En AWNN el problema se representa por una matriz de adyacencia de grafos y al mismo tiempo, utiliza caracteres de procesamiento paralelo y la onda automática de la PCNN.

En AWNN cada neurona individual esta denotada por los índices (i,j) y puede ser activada sólo una vez por los vecinos cuando la onda automática avanza. Como se

muestra en las ecuaciones cada neurona está dividida en tres componentes: entradas $F_{ij}[n]$ y W_{ij}, la actividad interna $U_{ij}[n]$ y el umbral dinámico $E[n]$.

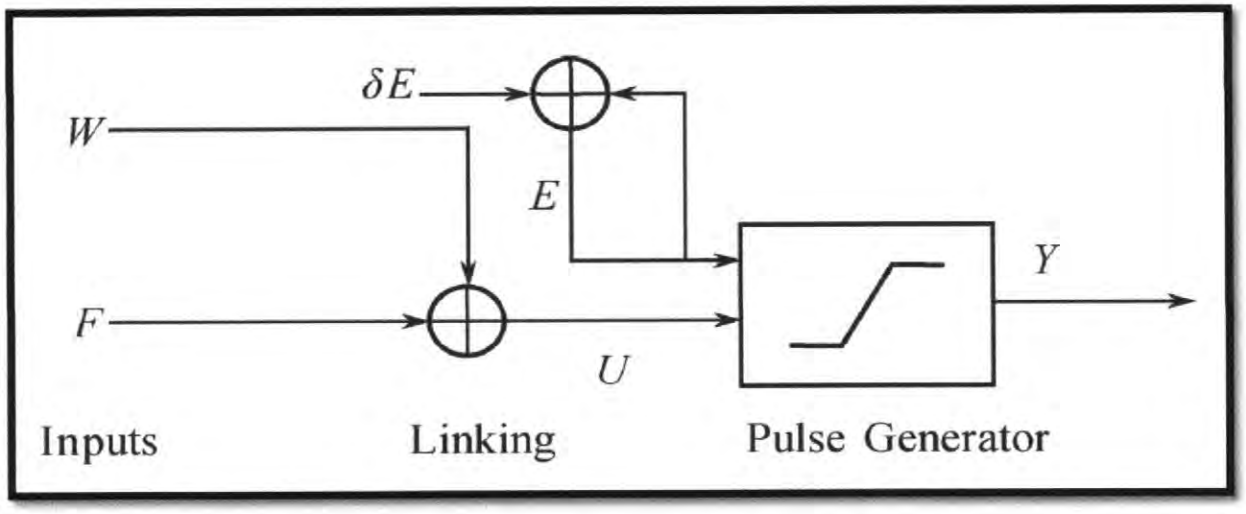


Figura 4.1 Modelo de la Neurona AWNN [Ma, 2010].

$$F_i[n] = F_h[n] + W_{hj}; \tag{18}$$

$$U_{ij}[n] = F_i[n] + W_{ij}; \tag{19}$$

$$Y_{ij}[n] = \begin{cases} 1 \; E[n] \geq U_{ij}[n], \\ 0 \quad Otherwise; \end{cases} \tag{20}$$

$$E[n] = E[n-1] + \delta E. \tag{21}$$

Donde cada componente se describe en la Tabla 4.1.

Tabla 4.1 Descripción de los componentes del modelo AWNN.

Símbolo	Significado	Descripción
W	Matriz de costos.	Matriz que contiene los costos de conexión entre los nodos del grafo.
F	Registro de longitudes.	Vector de registro de la longitud desde el nodo inicio hasta el nodo activado.
U	Actividad interna de la red.	Matriz de los valores que se generan en la actividad interna de la red.
$\&E$	Delta umbral.	Incremento iterativo al umbral dinámico de la red.
E	Umbral dinámico.	Umbral dinámico que comparado con la actividad interna genera la salida de la red.
Y	Salida de la red.	Matriz de salida de la red donde se activan los nodos disparados. Aquí es donde se da la explicitación del conocimiento de la red.
M	Secuencia de nodos disparados.	Vector de registro de activación de los nodos.

4.2. Descripción del algoritmo AWNN

A continuación, se presenta el algoritmo AWNN como una adaptación del modelo AWNN presentado en la sección 4.1, el algoritmo esta seccionado en cuatro componentes correspondientes a las ecuaciones de dicho modelo base.

Algoritmo 1	**AWNN**
Input	Matriz de costos $\boldsymbol{W}$, nodo de inicio $\boldsymbol{n\_I}$, nodo destino $\boldsymbol{n\_D}$, delta umbral $\&\boldsymbol{E}$, umbral dinámico $\boldsymbol{E}$.
Output	Matriz de salida $\boldsymbol{Y}$, vector de registro de longitudes $\boldsymbol{F}$, secuencia de nodos $\boldsymbol{M}$, matriz de la actividad interna de la neurona $\boldsymbol{U}$.
1:	Inicializar estructura AWNN ← $[M_i, F_i, U_{ij}, Y_{ij}]$ en 0.
2:	Establecer número de iteraciones n en 1.
3:	Establecer el nodo de inicio n_I en F como ∞.
4:	Establecer el nodo destino n_D en M como 1.
5:	Establecer la fila de inicio de U como en W.
6:	**Repeat**
7:	Actualización de U (**Algoritmo 1.1**).
8:	Actualización de Y (**Algoritmo 1.2**).
9:	Actualización de M (**Algoritmo 1.3**).
10:	Actualización de F (**Algoritmo 1.4**).
11:	Actualizar $E(\text{n})$ ← $E(n-1) + \&E$.
12:	Incrementar el número de iteraciones n en 1.
13:	**Until** que el nodo destino en el vector de secuencia de nodos $(M(n\_D - 1) == 0)$ no se haya activado.
14:	Explicitación del conocimiento (**Algoritmo 2**: sección 4.4).

4.2.1 Algoritmo 1.1 AWNN: Actualización de actividad interna de la neurona

El algoritmo de actualización de la actividad interna de la neurona correspondiente a la ecuación (19) del modelo AWNN se presenta a continuación:

Algoritmo 1.1	**AWNN: Actualización de U**
Input	Número de iteración n, vector de nodos activados M, vector de registro de longitudes F, matriz de costos W.
Output	Matriz de la actividad interna de la neurona U.
1:	**If** $n! = 1$ **then**
2:	**For** $k = 0$ **to** $k < número\_nodos\_W$ **do**
3:	**If** $(M(l) == 1)\&\&\ (k! = (n_l - 1))$ **then**
4:	**For** $l == 0$ **to** $l < número\_nodos\_W$ **do**
5:	**If** $(M(k,l)! = 0)\&\&(\ U(k,l) == 0)$ **then**
6:	**If** $M(l) == 1$ **then**
7:	Marcar la actividad interna de la neurona ya activada anteriormente en infinito $U(k,l) \leftarrow \infty$
8:	**Else if** $M(l) == 0$ **then**
9:	Actualizar la actividad interna sumando el costo del registro de longitudes y el costo de la actividad interna actual $U(k,l) \leftarrow F(l) + W(k,l)$
10:	**End if**
11:	**End for**
12:	**End if**
13:	**End for**
14:	**End if**

4.2.2 Algoritmo 1.2 AWNN: Actualización de la matriz de salida

El algoritmo de actualización de la actividad interna de la neurona correspondiente a la ecuación (19) del modelo AWNN se presenta a continuación:

Algoritmo 1.2	**AWNN: Actualización de Y**
Input	Matriz de la actividad interna de la neurona U, vector de nodos activados M, umbral dinámico E.
Output	Matriz de salida Y.
1:	**For** k= 0 **to** k< $número\_nodos\_W$ **do**
2:	**For** l= 0 **to** l< $número\_nodos\_W$ **do**
3:	**If** $(U(k,l)! = 0)$&& $(M(l) == 0)$ **then**
4:	**If** $E > U(k,l)$ **then**
5:	Establecer la matriz de salida en 1. $Y(k,l) \leftarrow 1$
6:	**Else**
7:	Establecer la matriz de salida en 0. $Y(k,l) \leftarrow 0$
8:	**End if**
9:	**End if**
10:	**End for**
11:	**End for**

4.2.3 Algoritmo 1.3 AWNN: Actualización de la secuencia de nodos disparados

El algoritmo de actualización de la secuencia de nodos disparados durante el proceso de búsqueda de la ruta más corta del modelo AWNN se presenta a continuación:

Algoritmo 1.3	**AWNN: Actualización de M**
Input	Matriz de la actividad interna de la neurona U, vector de nodos activados M, umbral dinámico E.
Output	Matriz de salida Y.
1:	**For** k= 0 **to** k< $número\_nodos\_W$ **do**
2:	**For** l= 0 **to** l< $número\_nodos\_W$ **do**
3:	**If** $(U(k,l)! = 0)\&\&\ (M(l) == 0)$ **then**
4:	**For** $l == 0$ **to** $l < número\_nodos\_W$ **do**
5:	**If** $(M(k,l)! = 0)\&\&(\ U(k,l) == 0)$ **then**
6:	**If** $E > U(k,l)$ **then**
7:	$Y(k,l) \leftarrow 1$
8:	**Else**
9:	$Y(k,l) \leftarrow 0$
10:	**End if**
11:	**End if**
12:	**End for**
13:	**End if**
14:	**End for**
15:	**End for**

4.2.4 Algoritmo 1.4 AWNN: Actualización del vector de registro de longitudes

El algoritmo de actualización del vector de registro de longitudes durante el proceso de búsqueda de la ruta más corta del modelo AWNN se presenta a continuación:

Algoritmo 1.3	**AWNN: Actualización de F**
Input	Matriz de salida Y, vector de nodos activados M, Matriz de costos W.
Output	Vector de registro de longitudes F.
1:	**If** $n == 1$ **then**
2:	**For** $k == 0$ **to** $k < número\_nodos\_W$ **do**
3:	**If** $(h == n\_l - 1)$ **then**
4:	**For** $l == 0$ **to** $l < número\_nodos\_W$ **do**
5:	**If** $(M(l) == 1)\&\&(l\,! = n_l - 1)$ **then**
6:	Actualizar el registro de longitudes $F(l) \leftarrow W(k,l)$
7:	**End if**
8:	**End for**
9:	**End if**
10:	**End for**
11:	**Else**
12:	**For** $k == 0$ **to** $k < número\_nodos\_W$ **do**
13:	**For** $l == 0$ **to** $l < número\_nodos\_W$ **do**
14:	**If** $(M(l) == 1)$
15:	**If** $(Y(k,l)! = 0)$
16:	Actualizar el registro de longitudes $F(l) \leftarrow U(k,l)$
17:	**End if**

18:	**End if**
19:	**End for**
20:	**End for**
21:	**End if**

4.3. Implementación del algoritmo AWNN en un grafo no dirigido.

La implementación de este modelo especialmente diseñado para el problema SP se puede ejemplificar con el siguiente ejemplo:

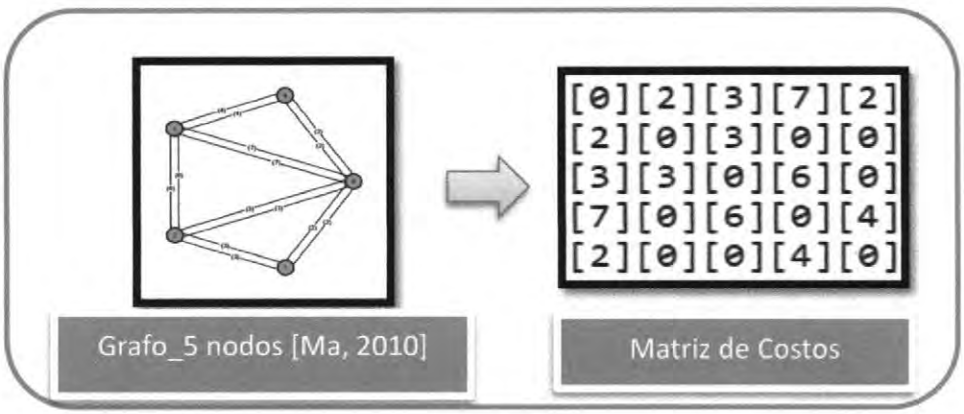


Figura 4.2 Ejemplo de matriz de costos de un grafo no dirigido.

Dada la matriz de costos de la Figura 4.2 se puede crear la estructura de la red neuronal de tipo AWNN mostrada en la Figura 4.3:

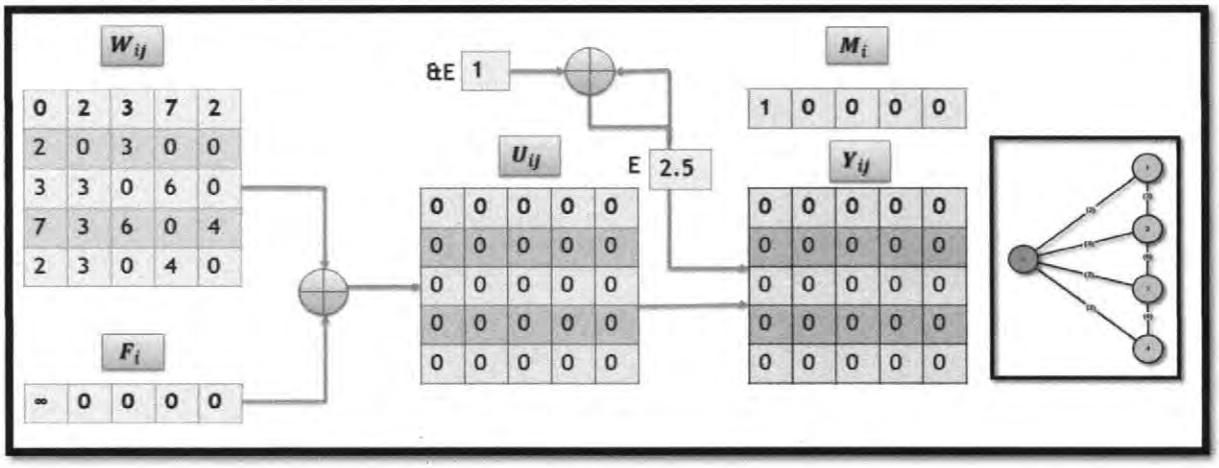


Figura 4.3 Estructura de la red Neuronal AWNN para la matriz de costos de la figura 4.2

Para la inicialización de la red de ejemplo se seleccionan los siguientes parámetros del experimento:

- Nodo Inicio: el nodo 0 (marcado en rojo).
- Nodo Destino 3.
- En el vector F la posición del nodo de inicio se marca con ∞ para que la neurona no se vuelva a activar (no se llegue al mismo nodo) y el resto en 0.
- La actividad interna U y la salida Y de la red se inicializan en 0.
- El vector M se marca con un 1 la posición del nodo inicio y el resto en 0.
- El delta umbral se inicializa en 1 ya que todos los valores de W son enteros.
- El umbral dinámico se inicializa en un número mayor al costo menor de W (en este caso 2.5).

Algunos puntos a consideración en el ejemplo son:

- El término de las iteraciones será cuando en el vector M se active en 1 la posición del nodo destino seleccionado.
- En cada iteración el delta umbral incrementara el umbral dinámico.
- En el vector F cada posición cambia una sola vez al igual que el vector M.

La iteración 1 de la red se puede visualizar en la Figura 4.4:

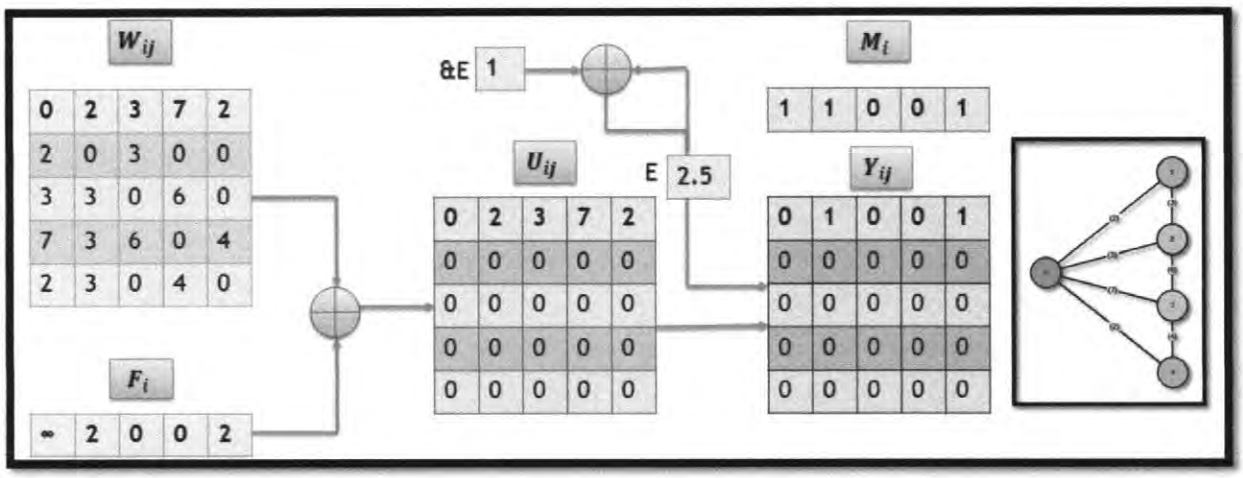


Figura 4.4 Iteración 1 de la red en el ejemplo del grafo de 5 nodos.

En donde se puede notar que en base a las ecuaciones correspondientes:

- Una vez seleccionado el nodo inicio la actividad interna U de esa neurona es igual a los valores de la matriz de costos W.
- El umbral dinámico E (2.5) rebasa los costos entre el nodo inicio y el nodo 1 y 4, por lo que en la matriz de salida Y se activan dichos nodos.
- Para actualizar el vector M se hace un recorrido en vertical de la matriz Y y los valores se cambian a 1, lo que podemos notar en el grafo de la parte derecha.

La iteración 2 de la red se puede visualizar en la Figura 4.5:

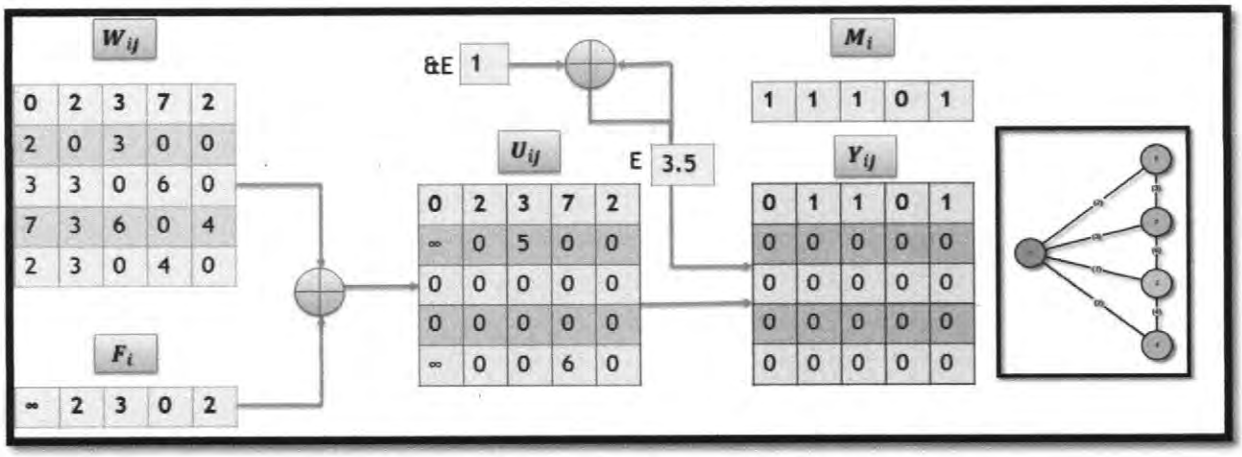


Figura 4.5 Iteración 2 de la red en el ejemplo del grafo de 5 nodos.

En donde podemos notar que en base a las ecuaciones correspondientes:

- El vector F ha cambiado su registro en las posiciones de los anteriores nodos activados.
- La actividad interna U se ve afectada por los nodos anteriores activados del registro de longitudes F y la matriz de costos W.
- El umbral dinámico E (3.5) ha sido incrementado en 1 por el delta umbral.
- Se ha activado el nodo 2 en la matriz de salida Y en base a que el umbral supera el costo de conexión entre el nodo inicio y el nodo 2.
- Se actualiza el vector M en la posición 2 como se puede notar en el grafo de la parte derecha.

La iteración 3 de la red se puede visualizar en la Figura 4.6:

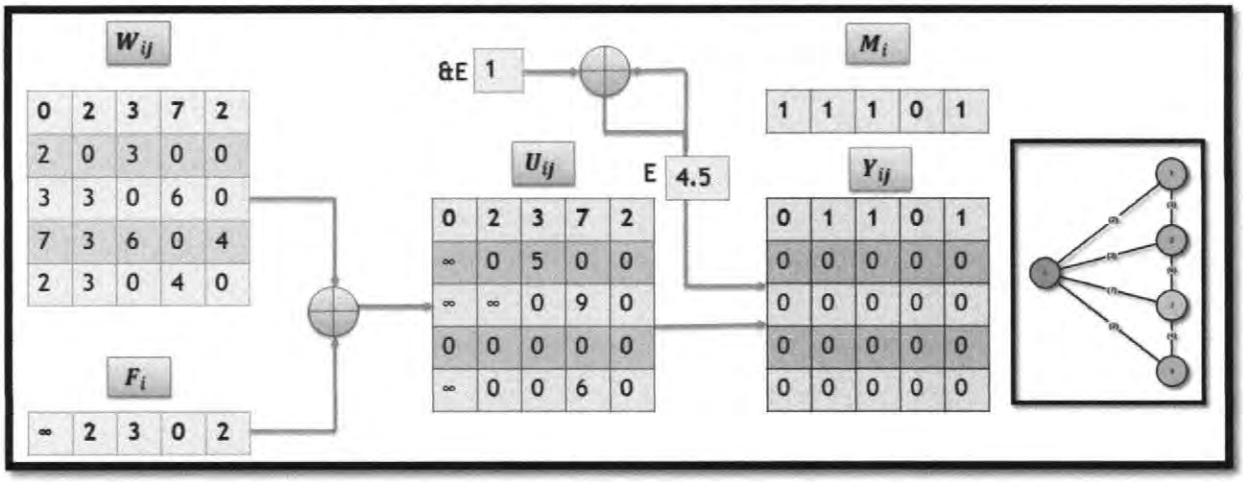


Figura 4.6 Iteración 3 de la red en el ejemplo del grafo de 5 nodos.

En donde podemos notar que en base a las ecuaciones correspondientes:

- El vector F no ha cambiado su registro.
- La actividad interna U se ve afectada por el registro de longitudes F y el nuevo umbral dinámico E (4.5).
- No se ha activado ningún nodo en la matriz de salida Y ya que el umbral dinámico no rebasa algún costo de la matriz U, por lo tanto no se actualiza el vector M como podemos notar en el grafo de la parte derecha.

La iteración 4 de la red se puede visualizar en la Figura 4.7:

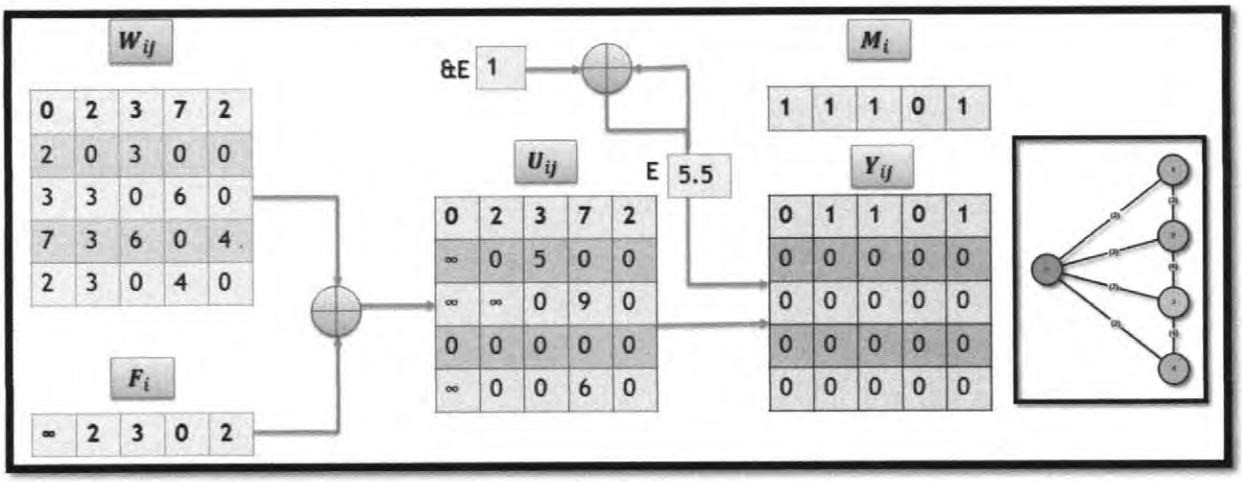


Figura 4.7 Iteración 4 de la red en el ejemplo del grafo de 5 nodos.

En donde se puede notar que en base a las ecuaciones correspondientes:

- El vector F no ha cambiado su registro.
- La salida Y no se ve afectada ya que el nuevo umbral dinámico E (5.5) no rebasa algún costo de la matriz U por lo tanto no se actualiza el vector M como podemos notar en el grafo de la parte derecha.

La iteración 5 de la red se puede visualizar en la Figura 4.8:

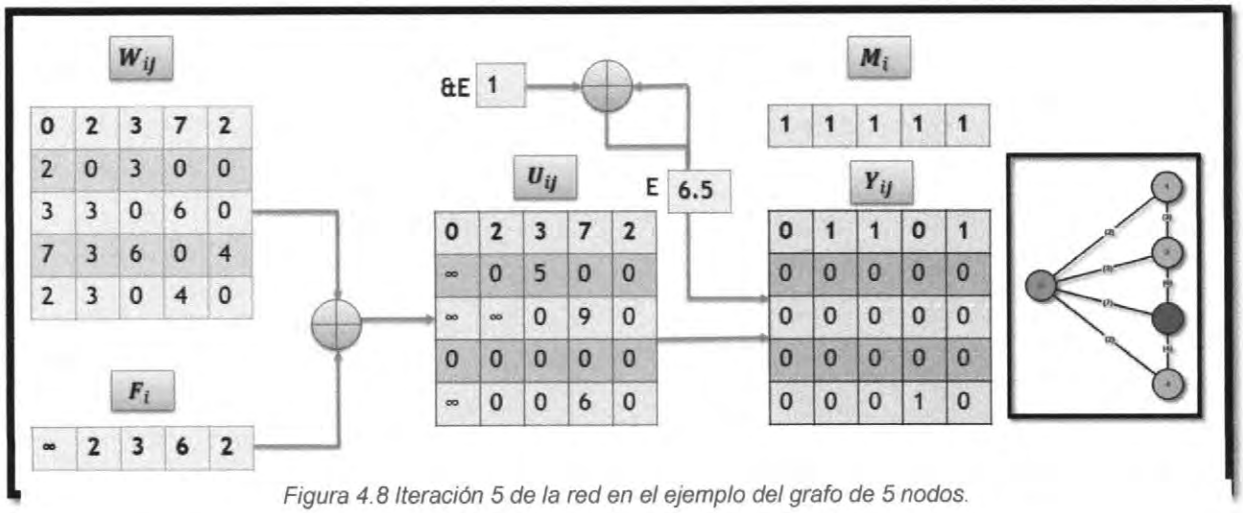


Figura 4.8 Iteración 5 de la red en el ejemplo del grafo de 5 nodos.

En donde se puede notar que en base a las ecuaciones correspondientes:

- El umbral dinámico al incrementarse a 6.5 se puede rebasar el costo 6 de la actividad interna U y esto a su vez activa el nodo 3 en la matriz de salida Y.
- El vector M se actualiza en la posición del nodo destino y esto provoca la finalización del algoritmo, activándose el nodo 3 como se puede notar en el grafo de la parte derecha.

Una parte importante del algoritmo es la explicitación del conocimiento de la red para poder reconstruir la ruta arrojada intrínsecamente dentro de la red, en la siguiente sección se detalla el algoritmo aportado para dicha actividad.

4.4. Explicitación del conocimiento del algoritmo AWNN

La explicitación del conocimiento de la red neuronal tipo AWNN se da al analizar la matriz de salida Y. Esta matriz permite con respecto a su ecuación correspondiente en [Ma, 2010] crear un algoritmo secuencial (algoritmo 2) para la "Reconstrucción de la ruta", el cual se puede aplicar una vez concluido el proceso de búsqueda de la ruta más corta.

Algoritmo 2	**Explicitación del conocimiento**
Input	Matriz de salida $\boldsymbol{Y}$, nodo de inicio $\boldsymbol{n\_I}$, nodo destino $\boldsymbol{n\_D}$.
Output	Ruta de nodos $\boldsymbol{R}$.
1:	Asignar nodo destino como Nodo Objetivo $n\_D \leftarrow n\_O$.
2:	Añadir a la ruta de nodos el nodo objetivo $R \leftarrow n\_D$.
3:	**Repeat**
4:	Establecer la posición del nodo encontrado en cero $p\_nE \leftarrow 0$.
5:	Establecer el nodo encontrado en falso $n\_E \leftarrow False$.
6:	**Repeat while** el nodo encontrado sea igual a falso $n_E == false$.

7:	**If** $(Y(p_{nE}, n\_O == 1))$ **then**
8:	Establecer el nodo anterior como la posición del nodo encontrado $n\_A \leftarrow p\_nE$.
9:	Establecer el nodo encontrado en verdadero $n\_E \leftarrow true$.
10:	**Else**
11:	Establecer el nodo encontrado en falso $n\_E \leftarrow False$.
12:	**End fin**
13:	Incrementar la posición del nodo encontrado en 1: p_nE++.
14:	Añadir a la ruta de nodos el nodo anterior $R \leftarrow n\_A$.
15:	Establecer el nodo objetivo como el nodo anterior $n\_O \leftarrow n\_A$
16:	**Until** que el nodo anterior sea diferente del nodo de inicio (n_A ! = n_I).

La implementación de este algoritmo se puede dar a partir de la última iteración (Figura 4.9).

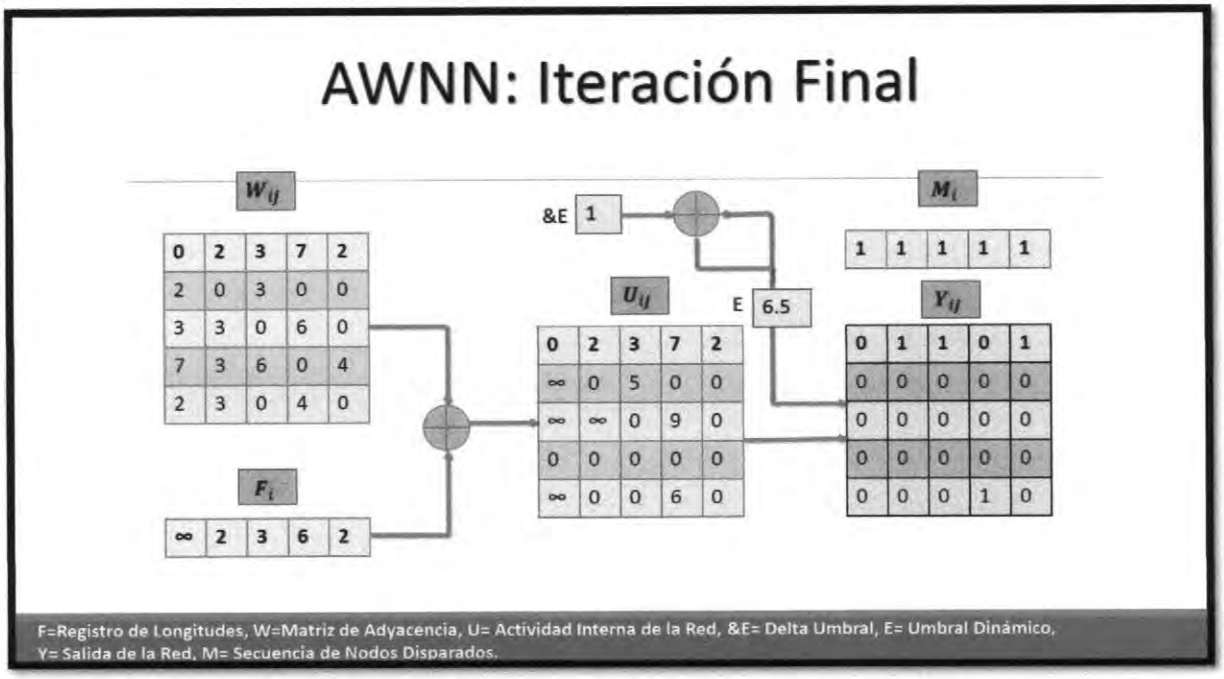
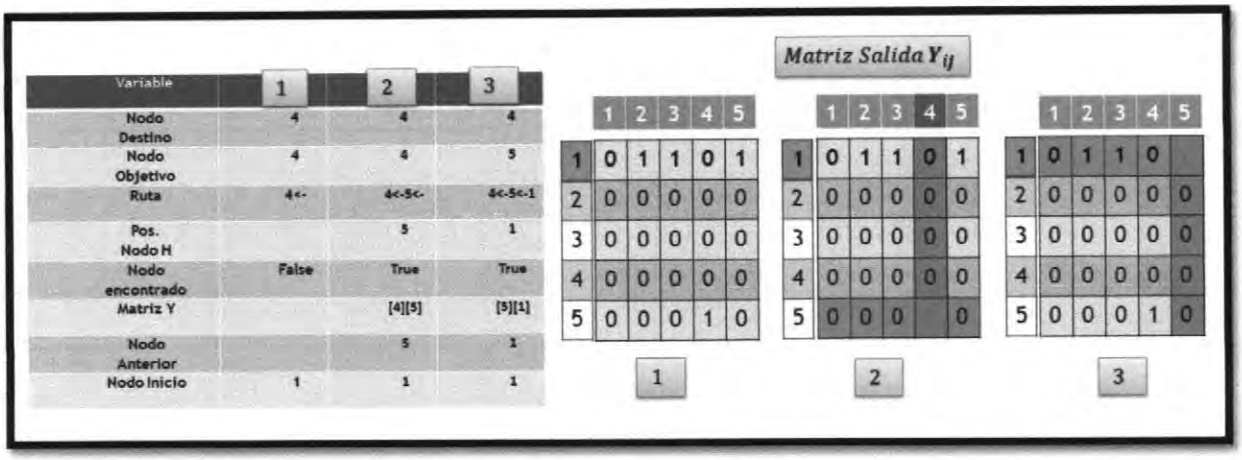


Figura 4.9 Ultima iteración del ejemplo del grafo de 5 nodos.

Para llevar a cabo la implementación del algoritmo en el ejemplo presentado se siguen los siguientes pasos:

- En el índice 1 de la Figura 4.10 correspondiente a la matriz de salida Y se marca con rojo el "Nodo inicio" y el "Nodo destino" (4) se agrega a "Ruta" (4 <-).
- En el índice 2 se busca en la columna del "Nodo Objetivo" (4) la casilla activada en 1 desde donde se llegó a este nodo y se guarda la posición de la casilla encontrada (5) en "Pos. Nodo H" que corresponde a "Nodo Anterior" y este se agrega a la "Ruta" (4 <- 5<-), posteriormente la el "Nodo Objetivo" se actualiza con el valor de "Pos. Nodo H" (5).

- En el índice 3 se repite el proceso buscando en la columna del "Nodo Objetivo" (5) la casilla activada en 1 desde donde se llegó a este nodo y se guarda la posición de la casilla encontrada (1) en "Pos. Nodo H" que corresponde a "Nodo Anterior" y este se agrega a la "Ruta" (4 <- 5<- 1).
- El proceso continuará siempre que "Nodo Anterior" sea diferente de "Nodo inicio".



Variable	1	2	3
Nodo Destino	4	4	4
Nodo Objetivo	4	4	5
Ruta	4<-	4<-5<-	4<-5<-1
Pos. Nodo H		5	1
Nodo encontrado	False	True	True
Matriz Y		[4][5]	[5][1]
Nodo Anterior		5	1
Nodo Inicio	1	1	1

	1	2	3	4	5
1	0	1	1	0	1
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	1	0

1

	1	2	3	4	5
1	0	1	1	0	1
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0		0

2

	1	2	3	4	5
1	0	1	1	0	
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	1	0

3

Figura 4.10 Ejemplo de la implementación de la "Reconstrucción de la ruta" a partir de la matriz de salida Y en la figura 4.9.

4.5 Complejidad algorítmica

Para un grafo G con N nodos y E vértices, en la adaptación del modelo propuesto AWNN[Ma,2010] el número de neuronas necesarias es N^2, en donde su complejidad algorítmica para encontrar la ruta más corta entre dos nodos depende del número de iteraciones (M) ya que en cada iteración debe actualizar su umbral dinámico incrementado constantemente por el delta-umbral, por lo que la complejidad computacional se define como $O(MN^2)$.

En AWNN las neuronas pueden ser estimuladas al mismo tiempo siempre y cuando el umbral dinámico rebase su actividad interna, por lo cual la complejidad algorítmica del procesamiento paralelo entre neuronas en este paradigma seria $O(M)$ si se tuviera un procesador por neurona requerida, esto a su vez disminuiría el tiempo de procesamiento T en T/N^2 .

4.6. Discusión

La adaptación del algoritmo AWNN para el problema del camino más corto resultó exitosa al lograr su implementación en lenguaje C++, una vez implementado se logró la explicitación del conocimiento para extraer el camino más corto en un grafo no dirigido a través de la matriz de salida Y de la red AWNN.

Ya que este modelo de RNA es una variante de las Redes Neuronales Artificiales Pulsantes hereda la característica de su naturaleza paralela, resaltando también su característica de pertenecer a la tercera generación la cual no requiere de entrenamiento previo, por lo cual tiene gran potencial para ser implementada en hardware especializado para su desempeño paralelo a diferencia de los algoritmos tradicionales por naturaleza secuenciales.

Capítulo 5

Experimentación y resultados

En este capítulo se presentan los experimentos realizados con la implementación del algoritmo propuesto, con la API de OpenMP y Dijkstra, así como la experimentación con el software "Grafos" sobre las bases de datos propuestas.

Además, se muestran los resultados entre los algoritmos experimentados basados en las medidas de comparación propuestas en el diseño de las pruebas.

5.1. Introducción

En los primeros experimentos se utilizaron casos planteados en la literatura los cuales por su baja complejidad se utilizaron para comprobar el buen funcionamiento de los algoritmos, entre ellos el caso de implementación mostrado en la sección 4.3 y a partir del cual se analizó la matriz Y de salida del modelo AWNN para lograr la explicitación del conocimiento del algoritmo.

En el estado del arte se menciona la creación de un generador de casos de prueba [Qu, 2016] en la sección 7.1.1 a partir del cual se dio a la tarea de recrear el "Generador de casos de prueba" para ampliar la experimentación siendo de mayor complejidad con respecto al tamaño de los grafos utilizados en los casos planteados en la literatura. Para

esta sección de pruebas se generaron dos bases de datos para realizar la experimentación:

- La primera consistió en experimentar 10 veces (promediando el tiempo de procesamiento) cada uno de los 32 grafos generados de la primera base de datos con un rango de dimensiones de 6 hasta 9000 nodos con distinto porcentaje de conexión y un rango de costos entre sus nodos de 1 a 100.
- La segunda consistió en experimentar 100 veces (promediando el tiempo de procesamiento) cada uno de los 30 grafos generados de la segunda base de datos con un rango de dimensiones de 6 hasta 9000 nodos con un porcentaje de conexión estandarizado de 50% y un rango de costos entre sus nodos de 1 a 100.

Para complementar la experimentación se utilizó una base de datos de caso real [Brinkhoff, 2002] mencionada en la referencia [Qu, 2016] la cual consta de 6105 nodos perteneciente a la red de carreteras de la ciudad de Oldenburg, Alemania.

La experimentación fue basada en evaluar los algoritmos utilizados y/o implementados en cada tipo de base de datos con respecto a las medidas de comparación planteadas para el problema SP.

La experimentación se realizó en una computadora con doble sistema operativo de 64 bits; Windows 10 Home Single Languaje y Linux Xubuntu 16.04 con procesador AMD A10-5745M APU with Radeom (tm) HD Graphics a 2.10 GHz, con memoria RAM de 6GB. Se utilizó el Software "Grafos" versión: 1.3.5 para Windows.

5.2. Diseño e implementación de pruebas

A continuación, se describe el entorno de desarrollo utilizado durante la experimentación, las bases de datos y las medidas de comparación utilizadas.

5.2.1 Entorno de desarrollo

El desarrollo del software y pruebas se realizaron bajo las siguientes especificaciones:

- Sistema Operativo: Windows 10 y Xubuntu 16. 04.
- Plataformas de Desarrollo: Eclipse IDE for C/C++ Developers Versión: Neon.2 Release (4.6.2) para Windows y Gcc para XUbuntu 16. 04.
- Software "Grafos" versión: 1.3.5 para Windows.

5.2.2 Bases de datos

Las bases de datos utilizadas para la experimentación estuvieron compuestas por tres tipos:

- **Casos de prueba planteados en la literatura. -**

De los artículos reportados en el estado del arte se tomaron los casos planteados en la literatura (Tabla 5.1) correspondientes a grafos no dirigidos para replicar los experimentos y comprobar el buen funcionamiento del algoritmo AWNN.

Tabla 5.1 Casos planteados en la literatura

Dimensión (nodos)	Tipo	Articulo
5	No dirigido	[Ma, 2020]
12	No dirigido	[Ma, 2011]
9	No dirigido	[Qu, 2012]
10	No dirigido	[Ma, 2013]
13	No dirigido	[Qu, 2016]

- **"Generador de casos de prueba". -**

El generador de casos de prueba se desarrolló en C++ el cual genera matrices de costos en archivos .txt de manera aleatoria que representan grafos no dirigidos.

Los parámetros que el generador permite seleccionar son:

- El número de nodos en el grafo a generar.
- Un rango que va de 1 al número limite seleccionado.
- El umbral para el porcentaje de conexión del grafo.

- **Casos de prueba reales. -**

De la base de catos de casos reales se tomó la red de carreteras de una ciudad de Alemania:

- Red de carreteras de la ciudad de Oldenburg [Brinkhoff, 2002] con 6105 nodos.

De la cual se planearon dos tipos de experimentos:

- La primera prueba consiste en utilizar los costos originales de entre los nodos de la red de carreteras.
- La segunda prueba consiste en utilizar los costos divididos entre 10 de la red de carreteras.

5.2.3. Medidas de comparación

Las medidas de comparación utilizadas para las pruebas fueron las siguientes:

- Tiempo de procesamiento.
- Calidad del resultado: correspondiente al costo total de la ruta.
- Espacio de búsqueda explorado: correspondiente al porcentaje de nodos explorados por cada experimento.

5.3. Casos planteados en la literatura

En este tipo de base de datos se conjunto de los grafos no dirigidos encontrados en el estado del arte mencionados en la sección 2.3 (Discusión del estado del arte) del estado del arte y enlistados en la sección 5.2.2 (Bases de datos).

5.3.1. AWNN

La Tabla 5.2 se muestra un resumen de la experimentación realizada con el algoritmo AWNN.

Tabla 5.2 Resumen de la experimentación de los casos de prueba planteados en la literatura con AWNN.

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
5 [Ma,2010]	N_I=0 / N_D=3	0-4-3	4	5	0.000179
12 [Ma,2011]	N_I=0 / N_D=8	0-1-8	17	14	0.000999667
9 [Qu,2012]	N_I=0 / N_D=8	0-6-8	3.4	17	0.001481
10 [Ma,2013]	N_I=0 / N_D=2	0-6-1-7-2	19	15	0.000999667
13 [Qu,2016]	N_I=3 / N_D=12	3-6-9-12	4.2	32	0.00150867

5.3.2. AWNN_OMP

La Tabla 5.3 muestra un resumen de la experimentación realizada con el algoritmo AWNN implementado con la API de OpenMP.

Tabla 5.3 Resumen de la experimentación de los casos de prueba planteados en la literatura con AWNN_OMP.

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
5 [Ma,2010]	N_I=0 / N_D=3	0-4-3	4	5	0.00266733
12 [Ma,2011]	N_I=0 / N_D=8	0-1-8	17	14	0.00697167
9 [Qu,2012]	N_I=0 / N_D=8	0-6-8	3.4	17	0.00916433
10 [Ma,2013]	N_I=0 / N_D=2	0-6-1-7-2	19	15	0.0153357
13 [Qu,2016]	N_I=3 / N_D=12	3-6-9-12	4.2	32	0.0183463

5.3.3. Dijkstra

La Tabla 5.4 muestra un resumen de la experimentación realizada con el algoritmo Dijkstra.

Tabla 5.4 Resumen de la experimentación de los casos de prueba planteados en la literatura con Dijkstra.

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
5 [Ma,2010]	N_I=0 / N_D=3	0-4-3	4	4	0.00133
12 [Ma,2011]	N_I=0 / N_D=8	0-1-8	17	8	0.000999667
9 [Qu,2012]	N_I=0 / N_D=8	0-6-8	3.4	5	0.001481
10 [Ma,2013]	N_I=0 / N_D=2	0-6-1-7-2	19	8	0.001632
13 [Qu,2016]	N_I=3 / N_D=12	3-6-9-12	4.2	12	0.004895

5.3.4. Software Grafos

En la experimentación de los algoritmos Dijkstra y Bellman-Ford del software "Grafos" se observó que los resultados arrojados eran similares, por lo que la Tabla 5.5 muestra como resumen de los dos algoritmos:

Tabla 5.5 Resumen de la experimentación Dijkstra/Bellman-Ford de los casos de prueba planteados en la literatura con el software Grafos [arodrigu, 2003-2012].

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Tiempo de solución (s)	Visualización

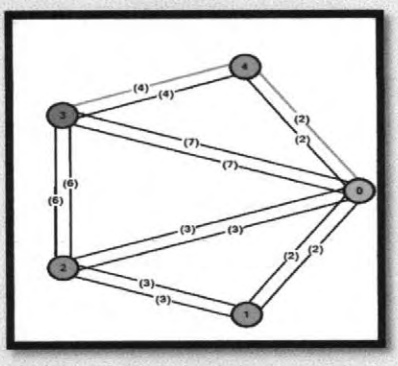
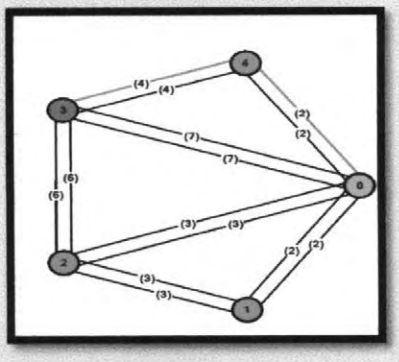
5 [Ma,2010]	N_I=0 / N_D=3	0 ----(2)---> 4 4 ----(4)---> 3	6	--------	

Tabla 5.5 Resumen de la experimentación Dijkstra/Bellman-Ford de los casos de prueba planteados en la literatura con el software Grafos [arodrigu, 2003-2012] (continuación de la Tabla 5.5).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Tiempo de solución (s)	Visualización
5 [Ma,2010]	N_I=0 / N_D=3	0 ----(2)---> 4 4 ----(4)---> 3	6	--------	

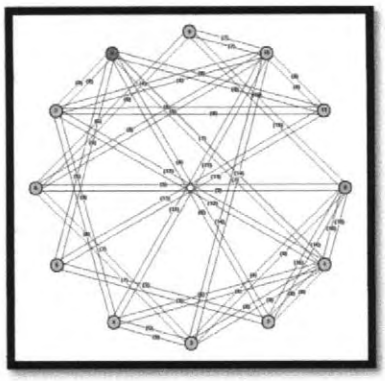
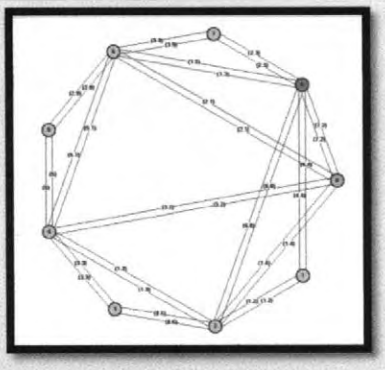
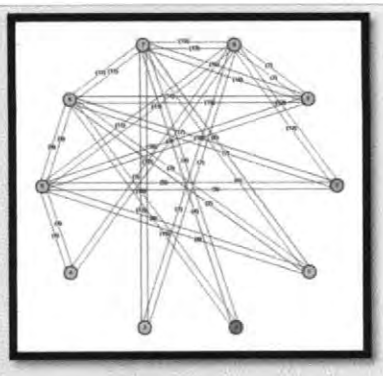
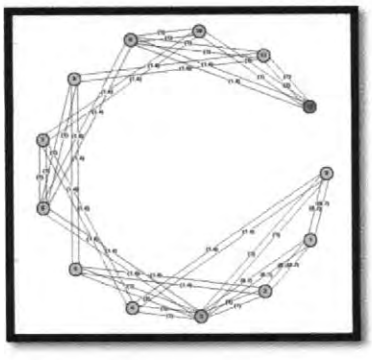
12 [Ma,2011]	N_I=0 / N_D=8	0 --(10)---> 1 1---(7)---> 8	17	---------	
9 [Qu,2012]	N_I=0 / N_D=8	0 --(2.1)--> 6 6 --(1.3)--> 8	3.4	---------	

Tabla 5.5 Resumen de la experimentación Dijkstra/Bellman-Ford de los casos de prueba planteados en la literatura con el software Grafos [arodrigu, 2003-2012] (continuación de la Tabla 5.5).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Tiempo de solución (s)	Visualización

10 **[Ma,2013]**	N_I=0 / N_D=2	0 ----(7)---> 6 1 ----(6)---> 7 6 ----(2)---> 1 7 ----(4)---> 2	19	---------	
13 **[Qu,2016]**	N_I=3 / N_D=12	3 --(1.4)-->6 6 --(1.4)-->9 9 -(1.4)-->12	4.2	---------	

5.4. Generador de casos de prueba

En este tipo de base de datos se utilizaron dos conjuntos de grafos no dirigidos generados; el primero de 32 grafos con dimensiones desde los 6 hasta los 9000 nodos con un porcentaje de conexión variado y el rango de costos de 1 hasta 100 entre sus nodos (se promedió el tiempo de procesamiento de 10 experimentos por grafo), y el segundo de 30 grafos con dimensiones desde los 5 hasta los 9000 nodos con un porcentaje de conexión estandarizado de 50% y el rango de costos de 1 hasta 100 entre sus nodos (se promedió el tiempo de procesamiento de 100 experimentos por grafo).

5.4.1. AWNN

La Tabla 5.6 muestra un resumen de la experimentación realizada con la primera base de datos generada de 32 grafos y el algoritmo AWNN.

Tabla 5.6 Resumen de la experimentación de la primera base de datos de 32 casos de prueba con AWNN.

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
6	N_I=0 N_D=5	0-(2)-4-(4)-3-(2)-5	8	5	0.000475
30	N_I=0 N_D=29	0-(1)-17-(1)-23-(4)-29	6	6	0.00075
50	N_I=0 N_D=49	0-(4)-25-(1)-36-(1)-9-(3)-49	9	7	0.00125
100	N_I=0 N_D=99	0-(12)-10-(4)-99	16	15	0.00475
200	N_I=0 N_D=199	0-(1)-162-(2)-7-(1)-199	4	4	0.0049995
300	N_I=0 N_D=299	0-(2)-92-(3)-79-(2)-126-(2)-284-(2)-299	11	10	0.0290325
400	N_I=0 N_D=399	0-(2)-172-(2)-269-(4)-399	8	7	0.033804
500	N_I=0 N_D=499	0-(2)-330-(2)-73-(2)-10-(2)-499	8	7	0.0635005
600	N_I=0 N_D=599	0-(2)-191-(3)-21-(2)-599	7	6	0.073088
700	N_I=0 N_D=699	0-(2)-141-(1)-138-(2)-181-(2)-699	7	7	0.10977

Tabla 5.6 Resumen de la experimentación de la primera base de datos de 32 casos de prueba con AWNN (continuación de la Tabla 5.6).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)

800	N_I=0 N_D=799	0-(4)-448-(1)-799	5	5	0.10825
900	N_I=0 N_D=899	0-(4)-292-(4)-899	8	7	0.216592
1000	N_I=0 N_D=999	0-(1)-100-(1)-200-(1)-300-(1)-400-(1)-500-(1)-600-(1)-700-(1)-800-(1)-900-(1)-950-(1)-999	11	11	0.281664
1300	N_I=1299 N_D=29	0-(2)-805-(2)-97-(2)-1299	6	5	0.298292
1500	N_I=0 N_D=1499	0-(2)-715-(2)-1368-(2)-1499	6	5	0.404062
1800	N_I=0 N_D=1799	0-(2)-981-(2)-418-(2)-1799	6	5	0.592084
2000	N_I=0 N_D=1999	0-(2)-1184-(2)-361-(2)-1999	6	5	0.708589
2400	N_I=0 N_D=2399	0-(2)-2256-(2)-55-(2)-2399	6	5	1.0128
2600	N_I=0 N_D=2599	0-(2)-117-(2)-223-(2)-2599	6	5	1.29977
3000	N_I=0 N_D=2999	0-(3)-1619-(2)-2588-(2)-2999	7	6	1.85208
3500	N_I=0 N_D=3499	0-(3)-580-(2)-3499	5	4	2.18869
4000	N_I=0 N_D=3999	0-(2)-2124-(2)-1286-(2)-3999	6	5	4.39908
4500	N_I=0 N_D=4499	0-(2)-2828-(2)-248-(2)-4499	6	5	3.953
5000	N_I=0 N_D=4999	0-(2)-4490-(4)-4999	6	5	3.92988
5500	N_I=0 N_D=5499	0-(2)-4552-(3)-5499	5	4	7.35715
6000	N_I=0 N_D=5999	0-(2)-2248-(2)-2741-(2)-5999	6	5	6.69575

Tabla 5.6 Resumen de la experimentación de la primera base de datos de 32 casos de prueba con AWNN (continuación de la Tabla 5.6).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
6500	N_I=0 N_D=6499	0-(2)-4236-(2)-1913-(2)-6499	6	5	7.77046
7000	N_I=0 N_D=6999	0-(2)-1250-(2)-6999	4	3	7.64294
7500	N_I=0 N_D=7499	0-(2)-6645-(2)-2939-(2)-20-(2)-7499	8	7	16.8965
8000	N_I=0 N_D=7999	0-(2)-7755-(2)-1334-(4)-7999	8	7	18.639
8500	N_I=0 N_D=8499	0-(2)-8184-(2)-707-(2)-8499	6	5	13.2688
9000	N_I=0 N_D=8999	0-(2)-1339-(2)-8999	4	3	11.0126

La Tabla 5.7 muestra un resumen de la experimentación realizada con la segunda base de datos generada de 30 grafos y el algoritmo AWNN.

Tabla 5.7 Resumen de la experimentación de la segunda base de datos de 30 casos de prueba con AWNN.

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
5	N_I=0 N_D=4	0-2-4	48	36	0.0031131
10	N_I=0 N_D=9	0-7-9	95	72	0.00629304
30	N_I=0 N_D=29	0-29	38	23	0.00189962
50	N_I=0 N_D=49	0-3-49	43	22	0.0024152
100	N_I=0 N_D=99	0-99	13	11	0.00274196
200	N_I=0 N_D=199	0-62-199	10	8	0.00738027
300	N_I=0 N_D=299	0-31-40-136-287-299	11	10	0.02711824

Tabla 5.7 Resumen de la experimentación de la segunda base de datos de 30 casos de prueba con AWNN (continuación de la Tabla 5.7).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
400	N_I=0 N_D=299	0-6-29-399	9	8	0.03577503
500	N_I=0 N_D=499	0-104-309-499	7	6	0.03971285
600	N_I=0 N_D=599	0-477-599	4	3	0.02440350
700	N_I=0 N_D=699	0-538-401-699	8	7	0.10315381
800	N_I=0 N_D=799	0-761-252-799	7	6	0.11563010
900	N_I=0 N_D=899	0-464-391-532-899	10	9	0.26801492
1000	N_I=0 N_D=999	0-989-708-922-999	8	7	0.23897439
1500	N_I=0 N_D=1499	0-648 -389-1499	7	6	0.45130058
2000	N_I=0 N_D=1999	0-368-1999	7	6	0.82105977
2500	N_I=0 N_D=2499	0-13-2499	6	5	0.92695587
3000	N_I=0 N_D=2999	0-1915-431-2999	7	6	1.93382510
3500	N_I=0 N_D=3499	0-2182-3499	6	5	1.94671940
4000	N_I=0 N_D=3999	0-504-3999	4	3	1.28367570
4500	N_I=0 N_D=4499	0-750-4411-4499	6	5	3.13612740
5000	N_I=0 N_D=4999	0-1553-4999	4	3	2.14409130
5500	N_I=0 N_D=5499	0-955-5499	5	4	6.50858590
6000	N_I=0	0-2731-1920-5999	6	5	5.82523070

	N_D=5999				

Tabla 5.7 Resumen de la experimentación de la segunda base de datos de 30 casos de prueba con AWNN (continuación de la Tabla 5.7).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
6500	N_I=0 N_D=6499	0-4256-6499	5	4	5.53602390
7000	N_I=0 N_D=6999	0-2187-61-6999	6	5	8.60693290
7500	N_I=0 N_D=7499	0-5645-58-7499	6	5	10.8559050
8000	N_I=0 N_D=7999	0-2121-7999	4	3	6.21174740
8500	N_I=0 N_D=8499	0-1313-420-8499	6	5	14.3948770
9000	N_I=0 N_D=8999	0-1797-8999	4	3	8.71657180

5.4.2. AWNN_OMP

La Tabla 5.8 muestra un resumen de la experimentación realizada con la primera base de datos generada de 32 grafos experimentados con AWNN_OMP.

Tabla 5.8 Resumen de la experimentación de la primera base de datos de 32 casos de prueba con AWNN_OMP.

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
6	N_I=0 N_D=5	0-(2)-4-(4)-3-(2)-5	8	5	0.007999
30	N_I=0 N_D=29	0-(1)-17-(1)-23-(4)-29	6	6	0.004501
50	N_I=0 N_D=49	0-(4)-25-(1)-36-(1)-9-(3)-49	9	7	0.004533
100	N_I=0 N_D=99	0-(12)-10-(4)-99	16	15	0.010521

200	N_I=0 N_D=199	0-(1)-162-(2)-7-(1)-199	4	4	0.004999

Tabla 5.8 Resumen de la experimentación de la primera base de datos de 32 casos de prueba con AWNN_OMP (continuación de la Tabla 5.8).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
300	N_I=0 N_D=299	0-(2)-92-(3)-79-(2)-126-(2)-284-(2)-299	11	10	0.022498
400	N_I=0 N_D=399	0-(2)-172-(2)-269-(4)-399	8	7	0.025000
500	N_I=0 N_D=499	0-(2)-330-(2)-73-(2)-10-(2)-499	8	7	0.039061
600	N_I=0 N_D=599	0-(2)-191-(3)-21-(2)-599	7	6	0.042122
700	N_I=0 N_D=699	0-(2)-141-(1)-138-(2)-181-(2)-699	7	7	0.064020
800	N_I=0 N_D=799	0-(4)-448-(1)-799	5	5	0.066540
900	N_I=0 N_D=899	0-(4)-292-(4)-899	8	7	0.124165
1000	N_I=0 N_D=999	0-(1)-100-(1)-200-(1)-300-(1)-400-(1)-500-(1)-600-(1)-700-(1)-800-(1)-900-(1)-950-(1)-999	11	11	0.190501
1300	N_I=1299 N_D=29	0-(2)-805-(2)-97-(2)-1299	6	5	0.164425
1500	N_I=0 N_D=1499	0-(2)-715-(2)-1368-(2)-1499	6	5	0.222167
1800	N_I=0 N_D=1799	0-(2)-981-(2)-418-(2)-1799	6	5	0.311682
2000	N_I=0 N_D=1999	0-(2)-1184-(2)-361-(2)-1999	6	5	0.413162
2400	N_I=0 N_D=2399	0-(2)-2256-(2)-55-(2)-2399	6	5	0.614582
2600	N_I=0 N_D=2599	0-(2)-117-(2)-223-(2)-2599	6	5	0.709077

3000	N_I=0 N_D=2999	0-(3)-1619-(2)-2588-(2)-2999	7	6	2.010290

Tabla 5.8 Resumen de la experimentación de la primera base de datos de 32 casos de prueba con AWNN_OMP (continuación de la Tabla 5.8).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
3500	N_I=0 N_D=3499	0-(3)-580-(2)-3499	5	4	1.504710
4000	N_I=0 N_D=3999	0-(2)-2124-(2)-1286-(2)-3999	6	5	3.280560
4500	N_I=0 N_D=4499	0-(2)-2828-(2)-248-(2)-4499	6	5	2.208790
5000	N_I=0 N_D=4999	0-(2)-4490-(4)-4999	6	5	2.284290
5500	N_I=0 N_D=5499	0-(2)-4552-(3)-5499	5	4	4.220040
6000	N_I=0 N_D=5999	0-(2)-2248-(2)-2741-(2)-5999	6	5	3.778000
6500	N_I=0 N_D=6499	0-(2)-4236-(2)-1913-(2)-6499	6	5	4.215710
7000	N_I=0 N_D=6999	0-(2)-1250-(2)-6999	4	3	3.145750
7500	N_I=0 N_D=7499	0-(2)-6645-(2)-2939-(2)-20-(2)-7499	8	7	11.55660
8000	N_I=0 N_D=7999	0-(2)-7755-(2)-1334-(4)-7999	8	7	12.57450
8500	N_I=0 N_D=8499	0-(2)-8184-(2)-707-(2)-8499	6	5	10.08780

9000	N_I=0 N_D=8999	0-(2)-1339-(2)-8999	4	3	6.763040

La Tabla 5.9 muestra un resumen de la experimentación realizada con la segunda base de datos generada de 30 grafos y el algoritmo AWNN implementado con la API de OpenMP.

Tabla 5.9 Resumen de la experimentación de la segunda base de datos de 30 casos de prueba con AWNN_OMP.

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
5	N_I=0 N_D=4	0-2-4	48	36	0.00144782
10	N_I=0 N_D=9	0-7-9	95	72	0.00194815
30	N_I=0 N_D=29	0-29	38	23	0.00111028
50	N_I=0 N_D=49	0-3-49	43	22	0.00239188
100	N_I=0 N_D=99	0-99	13	11	0.00252795
200	N_I=0 N_D=199	0-62-199	10	8	0.00632163
300	N_I=0 N_D=299	0-31-40-136-287-299	11	10	0.01958683
400	N_I=0 N_D=299	0-6-29-399	9	8	0.02593995
500	N_I=0 N_D=499	0-104-309-499	7	6	0.03007743
600	N_I=0 N_D=599	0-477-599	4	3	0.02063412
700	N_I=0 N_D=699	0-538-401-699	8	7	0.06672316
800	N_I=0 N_D=799	0-761-252-799	7	6	0.08054877
900	N_I=0	0-464-391-532-899	10	9	0.16163840

	N_D=899				
1000	N_I=0 N_D=999	0-989-708-922-999	8	7	0.15274258
1500	N_I=0 N_D=1499	0-648 -389-1499	7	6	0.30846870

Tabla 5.9 Resumen de la experimentación de la segunda base de datos de 30 casos de prueba con AWNN_OMP (continuación de la Tabla 5.9).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
2000	N_I=0 N_D=1999	0-368-1999	7	6	0.56399680
2500	N_I=0 N_D=2499	0-13-2499	6	5	0.65740061
3000	N_I=0 N_D=2999	0-1915-431-2999	7	6	1.25499060
3500	N_I=0 N_D=3499	0-2182-3499	6	5	1.29684890
4000	N_I=0 N_D=3999	0-504-3999	4	3	0.96486804
4500	N_I=0 N_D=4499	0-750-4411-4499	6	5	2.12112240
5000	N_I=0 N_D=4999	0-1553-4999	4	3	1.51334820
5500	N_I=0 N_D=5499	0-955-5499	5	4	5.48662260
6000	N_I=0 N_D=5999	0-2731-1920-5999	6	5	3.84615720
6500	N_I=0 N_D=6499	0-4256-6499	5	4	3.36899040
7000	N_I=0 N_D=6999	0-2187-61-6999	6	5	5.43450810

7500	N_I=0 N_D=7499	0-5645-58-7499	6	5	6.24692770
8000	N_I=0 N_D=7999	0-2121-7999	4	3	4.10466560
8500	N_I=0 N_D=8499	0-1313-420-8499	6	5	8.32035980
9000	N_I=0 N_D=8999	0-1797-8999	4	3	5.83508210

5.4.3 Dijkstra

La Tabla 5.10 muestra un resumen de la experimentación realizada con la primera base de datos generada de 32 grafos y el algoritmo Dijkstra.

Tabla 5.10 Resumen de la experimentación de la primera base de datos de 32 casos de prueba con Dijkstra.

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
6	N_I=0 N_D=5	0-(2)-4-(4)-3-(2)-5	8	5	0.00066566
30	N_I=0 N_D=29	0-(1)-17-(1)-23-(4)-29	6	20	0.00063866
50	N_I=0 N_D=49	0-(4)-25-(1)-36-(1)-9-(3)-49	9	46	0.00148100
100	N_I=0 N_D=99	0-(12)-10-(4)-99	16	85	0.00182200
200	N_I=0 N_D=199	0-(1)-162-(2)-7-(1)-199	4	119	0.0030000
300	N_I=0 N_D=299	0-(2)-92-(3)-79-(2)-126-(2)-284-(2)-299	11	263	0.0073336
400	N_I=0 N_D=399	0-(2)-172-(2)-269-(4)-399	8	236	0.0074930
500	N_I=0 N_D=499	0-(2)-330-(2)-73-(2)-10-(2)-499	8	446	0.0156663
600	N_I=0 N_D=599	0-(2)-191-(3)-21-(2)-599	7	440	0.0154900

700	N_I=0 N_D=699	0-(2)-141-(1)-138-(2)-181-(2)-699	7	698	0.0276377
800	N_I=0 N_D=799	0-(4)-448-(1)-799	5	753	0.0341760
900	N_I=0 N_D=899	0-(4)-292-(4)-899	8	881	0.0391533
1000	N_I=0 N_D=999	0-(1)-100-(1)-200-(1)-300-(1)-400-(1)-500-(1)-600-(1)-700-(1)-800-(1)-900-(1)-950-(1)-999	11	11	0.0071393

Tabla 5.10 Resumen de la experimentación de la primera base de datos de 32 casos de prueba con Dijkstra (continuación de la Tabla 5.10).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
1300	N_I=1299 N_D=29	0-(2)-805-(2)-97-(2)-1299	6	1084	0.0576707
1500	N_I=0 N_D=1499	0-(2)-715-(2)-1368-(2)-1499	6	1055	0.184679
1800	N_I=0 N_D=1799	0-(2)-981-(2)-418-(2)-1799	6	1524	0.103228
2000	N_I=0 N_D=1999	0-(2)-1184-(2)-361-(2)-1999	6	1665	0.130505
2400	N_I=0 N_D=2399	0-(2)-2256-(2)-55-(2)-2399	6	1946	0.225071
2600	N_I=0 N_D=2599	0-(2)-117-(2)-223-(2)-2599	6	2563	0.247078
3000	N_I=0 N_D=2999	0-(3)-1619-(2)-2588-(2)-2999	7	1973	0.261566
3500	N_I=0 N_D=3499	0-(3)-580-(2)-3499	5	1219	0.202193
4000	N_I=0 N_D=3999	0-(2)-2124-(2)-1286-(2)-3999	6	3709	0.818422
4500	N_I=0 N_D=4499	0-(2)-2828-(2)-248-(2)-4499	6	4492	0.982235
5000	N_I=0	0-(2)-4490-(4)-4999	6	1311	0.388679

	N_D=4999				
5500	N_I=0 N_D=5499	0-(2)-4552-(3)-5499	5	3013	0.854427
6000	N_I=0 N_D=5999	0-(2)-2248-(2)-2741-(2)-5999	6	5968	1.630870
6500	N_I=0 N_D=6499	0-(2)-4236-(2)-1913-(2)-6499	6	5193	1.425790
7000	N_I=0 N_D=6999	0-(2)-1250-(2)-6999	4	3509	1.290580
7500	N_I=0 N_D=7499	0-(2)-6645-(2)-2939-(2)-20-(2)-7499	8	7278	1.992450
8000	N_I=0 N_D=7999	0-(2)-7755-(2)-1334-(4)-7999	8	7969	2.107390

Tabla 5.10 Resumen de la experimentación de la primera base de datos de 32 casos de prueba con Dijkstra (continuación de la Tabla 5.10).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
8500	N_I=0 N_D=8499	0-(2)-8184-(2)-707-(2)-8499	6	8406	2.644670
9000	N_I=0 N_D=8999	0-(2)-1339-(2)-8999	4	5892	2.543930

La Tabla 5.11 muestra un resumen de la experimentación realizada con la segunda base de datos generada de 30 grafos y el algoritmo Dijkstra.

Tabla 5.11 Resumen de la experimentación de la segunda base de datos de 30 casos de prueba con Dijkstra.

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
5	N_I=0 N_D=4	0-2-4	48	36	0.0001060
10	N_I=0 N_D=9	0-7-9	95	72	0.0009189
30	N_I=0 N_D=29	0-29	38	23	0.0016063

50	N_I=0 N_D=49	0-3-49	43	22	0.00311696
100	N_I=0 N_D=99	0-99	13	11	0.00492834
200	N_I=0 N_D=199	0-62-199	10	8	0.00520426
300	N_I=0 N_D=299	0-31-40-136-287-299	11	10	0.03053287
400	N_I=0 N_D=299	0-6-29-399	9	8	0.03100640
500	N_I=0 N_D=499	0-104-309-499	7	6	0.02218956
600	N_I=0 N_D=599	0-477-599	4	3	0.00274702
700	N_I=0 N_D=699	0-538-401-699	8	7	0.06648954

Tabla 5.11 Resumen de la experimentación de la segunda base de datos de 30 casos de prueba con Dijkstra (continuación de la Tabla 5.11).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
800	N_I=0 N_D=799	0-761-252-799	7	6	0.05471446
900	N_I=0 N_D=899	0-464-391-532-899	10	9	0.11582321
1000	N_I=0 N_D=999	0-989-708-922-999	8	7	0.11956752
1500	N_I=0 N_D=1499	0-648 -389-1499	7	6	0.16036582
2000	N_I=0 N_D=1999	0-368-1999	7	6	0.25176729
2500	N_I=0 N_D=2499	0-13-2499	6	5	0.24915797
3000	N_I=0 N_D=2999	0-1915-431-2999	7	6	0.43279199

3500	N_I=0 N_D=3499	0-2182-3499	6	5	0.49068130
4000	N_I=0 N_D=3999	0-504-3999	4	3	0.17079954
4500	N_I=0 N_D=4499	0-750-4411-4499	6	5	0.68546443
5000	N_I=0 N_D=4999	0-1553-4999	4	3	0.25239616
5500	N_I=0 N_D=5499	0-955-5499	5	4	0.59944513
6000	N_I=0 N_D=5999	0-2731-1920-5999	6	5	1.15558500
6500	N_I=0 N_D=6499	0-4256-6499	5	4	0.86603372
7000	N_I=0 N_D=6999	0-2187-61-6999	6	5	1.63661080

Tabla 5.11 Resumen de la experimentación de la segunda base de datos de 30 casos de prueba con Dijkstra (continuación de la Tabla 5.11).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)
7500	N_I=0 N_D=7499	0-5645-58-7499	6	5	1.85198550
8000	N_I=0 N_D=7999	0-2121-7999	4	3	0.73196661
8500	N_I=0 N_D=8499	0-1313-420-8499	6	5	2.39115980
9000	N_I=0 N_D=8999	0-1797-8999	4	3	1.01064778

5.4.4. Software Grafos

La Tabla 5.12 muestra los experimentos de la primera base de datos en el software "Grafos" de los algoritmos Dijkstra y Bellman-Ford.

Tabla 5.12 Resumen de la experimentación de la primera base de datos de 32 casos de prueba con Dijkstra y Bellman-Ford.

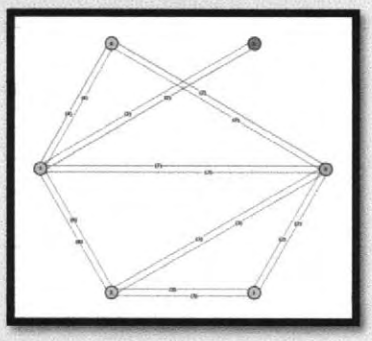
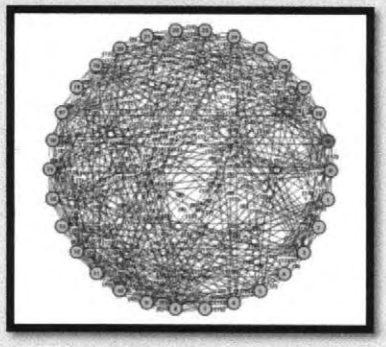
Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Tiempo de solución (s)	Visualización
6	N_I=0 N_D=5	0 ----(2)---> 4 4 ----(4)---> 3 3 ----(2)---> 5	8	---------	

Tabla 5.12 Resumen de la experimentación de la primera base de datos de 32 casos de prueba con Dijkstra y Bellman-Ford (continuación de la Tabla 5.12).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Tiempo de solución (s)	Visualización
30	N_I=0 N_D=29	0 --(1)-->17 17 --(1)-->23 23 --(4)-->29	6	---------	

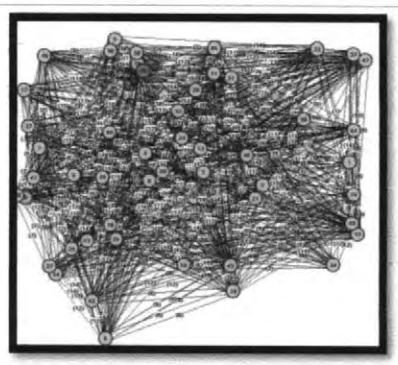
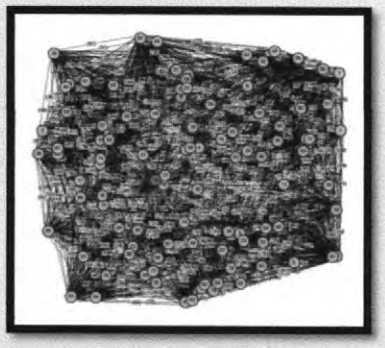
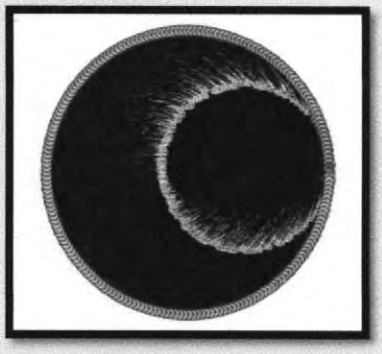
50	N_I=0 N_D=49	0 --(4)--> 25 25 --(1)-> 36 36 -(1)--> 9 9--(3)--> 49	9	---------	
100	N_I=0 N_D=99	0 --(12)-->10 10 --(4)-> 99	16	---------	

Tabla 5.12 Resumen de la experimentación de la primera base de datos de 32 casos de prueba con Dijkstra y Bellman-Ford (continuación de la Tabla 5.12).

Dimensión (Nodos)	N_Inicio N_Destino	Ruta	Costo Total	Tiempo de solución (s)	Visualización

200	N_I=0 N_D=199	0 --(1)-->162 162 --(2)-->7 7 --(1)-->199	4	--------	

5.5. Caso real

Para la experimentación se realizaron dos tipos de pruebas: la primera del nodo inicio 0 al nodo destino 32 con los costos divididos entre 10, y la segunda prueba del nodo inicio 0 al nodo 6104 con los costos divididos entre 10:

5.5.1. AWNN

La Tabla 5.13 muestra los experimentos del caso real con AWNN.

Tabla 5.13 Resumen de la experimentación de los casos reales con AWNN.

Dimensión (Nodos)/ Grafo	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)/(m)
6105Nodos-Oldenburg [Brinkhoff, 2002]	N_I=0 / N_D=32	0-(9)-1-(33)-3-(6)-4-(5)-6-(30)-8-(42)-10-(5)-12-(6)-16-(3)-20-(3)-25-(3)-32	145	137	151.247 / 2.5207

Tabla 5.13 Resumen de la experimentación de los casos reales con AWNN (continuación de la Tabla 5.13).

Dimensión (Nodos)/ Grafo	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)/(m)

6105Nodos-Oldenburg [Brinkhoff, 2002]	N_I=0 / N_D=6104	0-(9)-1-(33)-3-(6)-4-(5)-6-(37)-9-(49)-21-(3)-27-(4)-33-(42)-66-(30)-82-(49)-713-(60)-711-(20)-710-(37)-631-(44)-593-(13)-595-(3)-597-(6)-601-(2)-606-(12)-623-(2)-624-(5)-640-(3)-650-(15)-672-(16)-4295-(10)-4288-(5)-4285-(5)-4281-(14)-4292-(9)-4300-(20)-4317-(30)-2229-(11)-2204-(2)-2196-(14)-2166-(10)-2157-(18)-2149-(4)-2148-(1)-2150-(3)-2152-(1)-2154-(16)-2159-(3)-2162-(11)-2182-(4)-2193-(10)-2219-(4)-2227-(9)-2255-(6)-2262-(7)-6104	732	724	763.478 / 12.7246

5.5.2. AWNN_OMP

La Tabla 5.14 muestra los experimentos del caso real con AWNN_OMP.

Tabla 5.14 Resumen de la experimentación de los casos reales con AWNN_OMP.

Dimensión (Nodos)/ Grafo	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)/(m)
6105Nodos-Oldenburg [Brinkhoff, 2002]	N_I=0 / N_D=32	0-(9)-1-(33)-3-(6)-4-(5)-6-(30)-8-(42)-10-(5)-12-(6)-16-(3)-20-(3)-25-(3)-32	145	137	109.159 / 1.81931
6105Nodos-Oldenburg [Brinkhoff, 2002]	N_I=0 / N_D=6104	0-(9)-1-(33)-3-(6)-4-(5)-6-(37)-9-(49)-21-(3)-27-(4)-33-(42)-66-(30)-82-(49)-713-(60)-711-(20)-710-(37)-631-(44)-593-(13)-595-(3)-597-(6)-601-(2)-606-(12)-623-(2)-624-(5)-640-(3)-650-(15)-672-(16)-4295-(10)-4288-(5)-4285-(5)-4281-(14)-4292-(9)-4300-(20)-4317-(30)-2229-(11)-2204-(2)-2196-(14)-2166-(10)-2157-(18)-2149-(4)-2148-(1)-2150-(3)-2152-(1)-2154-(16)-2159-(3)-	732	724	583.156 / 9.7192

		2162-(11)-2182-(4)-2193-(10)-2219-(4)-2227-(9)-2255-(6)-2262-(7)-6104			

5.5.3. Dijkstra

La Tabla 5.15 muestra los experimentos del caso real con Dijkstra.

Tabla 5.15 Resumen de la experimentación de los casos reales con Dijkstra.

Dimensión (Nodos)/ Grafo	N_Inicio N_Destino	Ruta	Costo Total	Iteraciones	Tiempo de procesamiento (s)/(m)
6105Nodos-Oldenburg [Brinkhoff, 2002]	N_I=0 / N_D=32	0-(9)-1-(33)-3-(6)-4-(5)-6-(30)-8-(42)-10-(5)-12-(6)-16-(3)-20-(3)-25-(3)-32	145	33	0.344936
6105Nodos-Oldenburg [Brinkhoff, 2002]	N_I=0 / N_D=6104	0-(9)-1-(33)-3-(6)-4-(5)-6-(37)-9-(49)-21-(3)-27-(4)-33-(42)-66-(30)-82-(49)-713-(60)-711-(20)-710-(37)-631-(44)-593-(13)-595-(3)-597-(6)-601-(2)-606-(12)-623-(2)-624-(5)-640-(3)-650-(15)-672-(16)-4295-(10)-4288-(5)-4285-(5)-4281-(14)-4292-(9)-4300-(20)-4317-(30)-2229-(11)-2204-(2)-2196-(14)-2166-(10)-2157-(18)-2149-(4)-2148-(1)-2150-(3)-2152-(1)-2154-(16)-2159-(3)-2162-(11)-2182-(4)-2193-(10)-2219-(4)-2227-(9)-2255-(6)-2262-(7)-6104	732	3969	0.843272

5.6. Comparativa

La comparativa entre los algoritmos experimentados se realizó en base a las medidas de comparación mencionadas en la sección 5.2.3.

5.6.1 Casos planteados en la literatura

En la Figura 5.1 se muestra la gráfica comparativa entre AWNN [Ma, 2010], AWNN implementado con la API de OpenMP y Dijkstra en tiempo de procesamiento para los casos planteados en la literatura experimentados (sección 5.3).

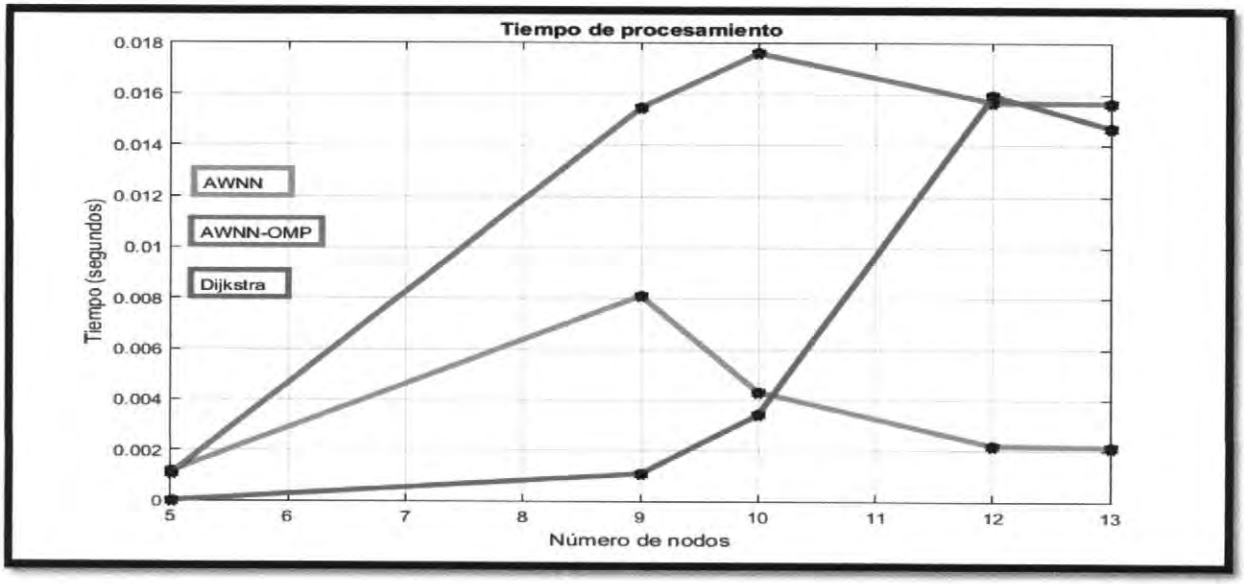


Figura 5.1 Gráfica comparativa en tiempo de procesamiento de la experimentación entre AWNN, AWNN_OMP y Dijkstra para los casos planteados en la literatura correspondientes a la sección 5.3.

En la Figura 5.2 se muestra la gráfica comparativa entre AWNN [Ma, 2010] y Dijkstra en la calidad de la ruta tomando como parámetro el costo total de la ruta óptima arrojada por AWNN contra Dijkstra en cada experimento de los casos plateados en la literatura (sección 5.3).

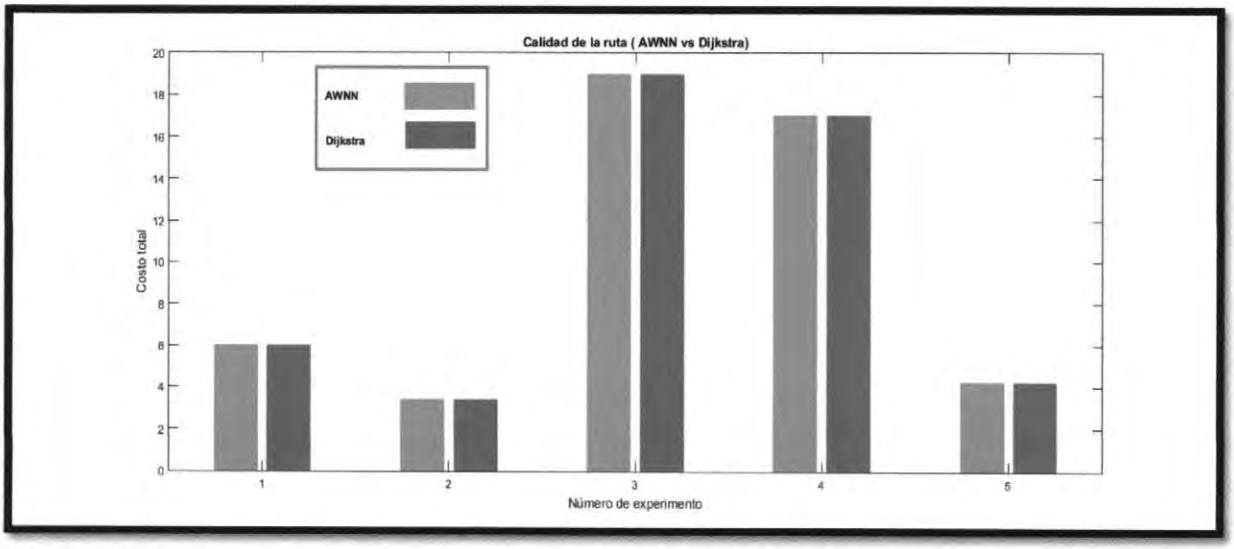


Figura 5.2 Gráfica comparativa en la calidad de la ruta óptima entre AWNN y Dijkstra para los casos planteados en la literatura correspondientes a la sección 5.3.

En la Figura 5.3 se muestra la gráfica comparativa del espacio de búsqueda explorado durante el proceso de obtención de la ruta óptima de AWNN contra Dijkstra en los casos planteados en la literatura (sección 5.3).

5.6.2. Generador de casos de prueba

En la Figura 5.4 se muestra la gráfica comparativa entre AWNN [Ma, 2010], AWNN implementado con la API de OpenMP y Dijkstra en tiempo de procesamiento para la primera base de datos de 32 casos por el "Generador de casos de prueba" (sección 5.3).

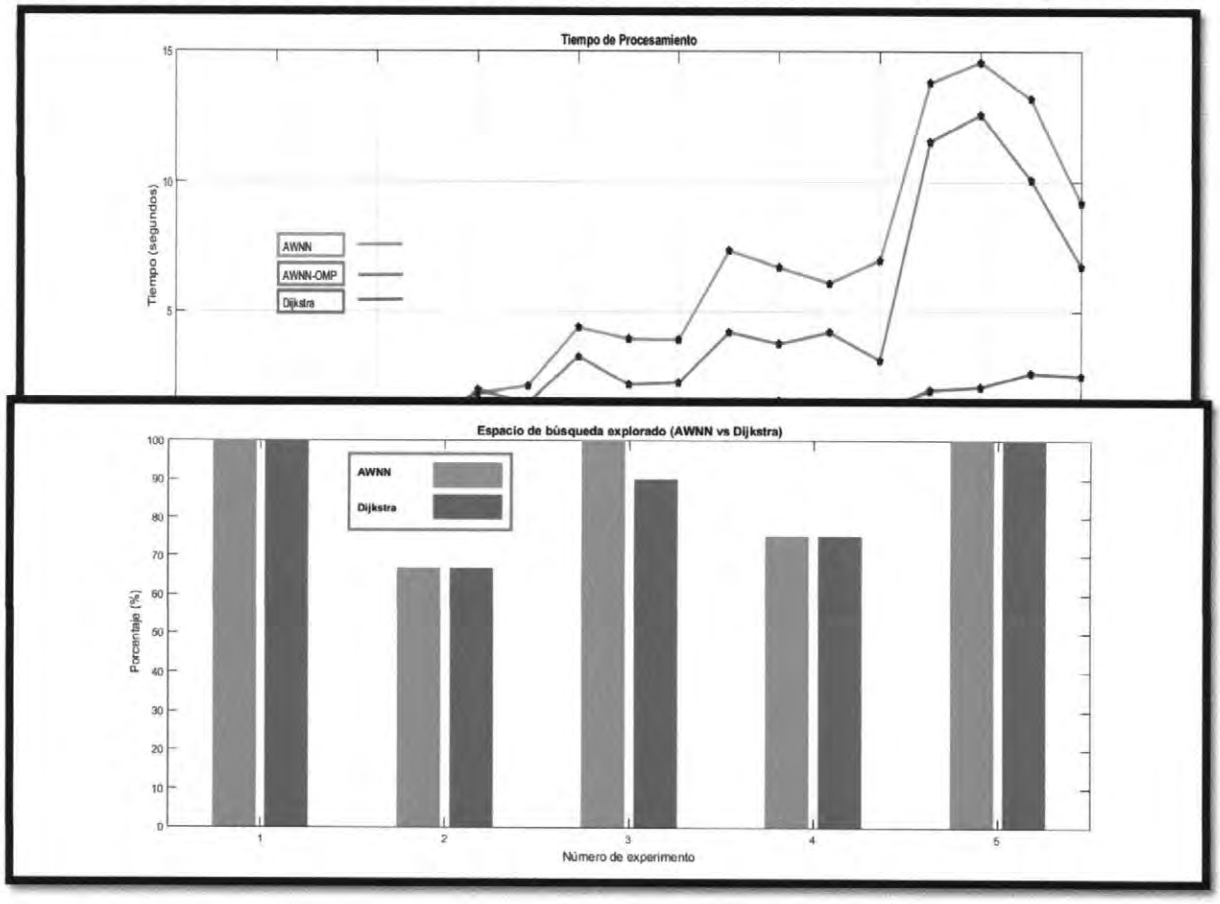


Figura 5.3 Gráfica comparativa del espacio de búsqueda explorado por AWNN contra Dijkstra en los casos planteados en la literatura.

En la Figura 5.5 se muestra la gráfica comparativa entre AWNN [Ma, 2010] y Dijkstra en la calidad de la ruta tomando como parámetro el costo total de la ruta óptima arrojada en cada experimento de la primera base de datos de los 32 casos generados (sección 5.3).

En la Figura 5.6 se muestra la gráfica comparativa del espacio de búsqueda explorado durante el proceso de obtención de la ruta óptima de AWNN contra Dijkstra en cada experimento de la primera base de datos de los 32 casos generados.

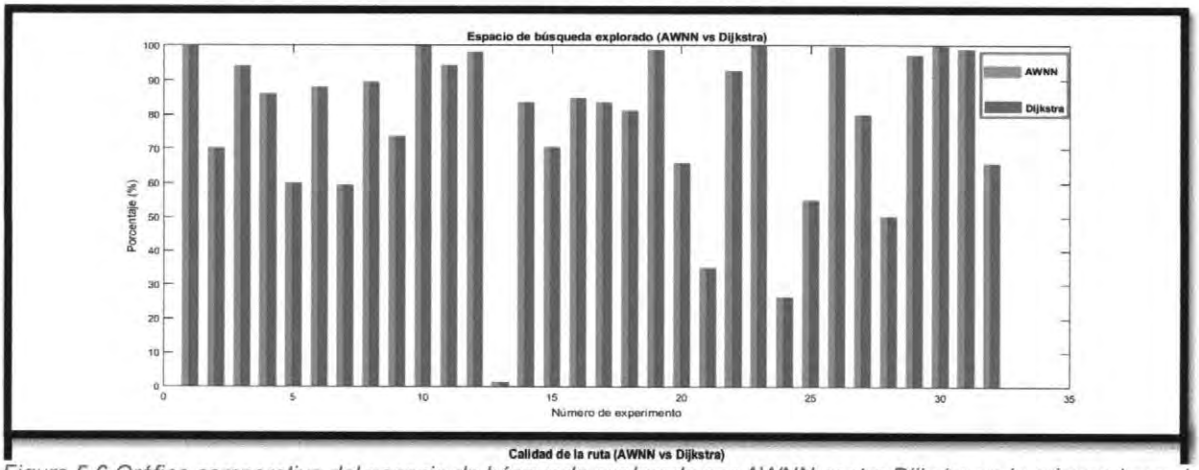


Figura 5.6 Gráfica comparativa del espacio de búsqueda explorado por AWNN contra Dijkstra en la primera base de datos de los 32 casos generados.

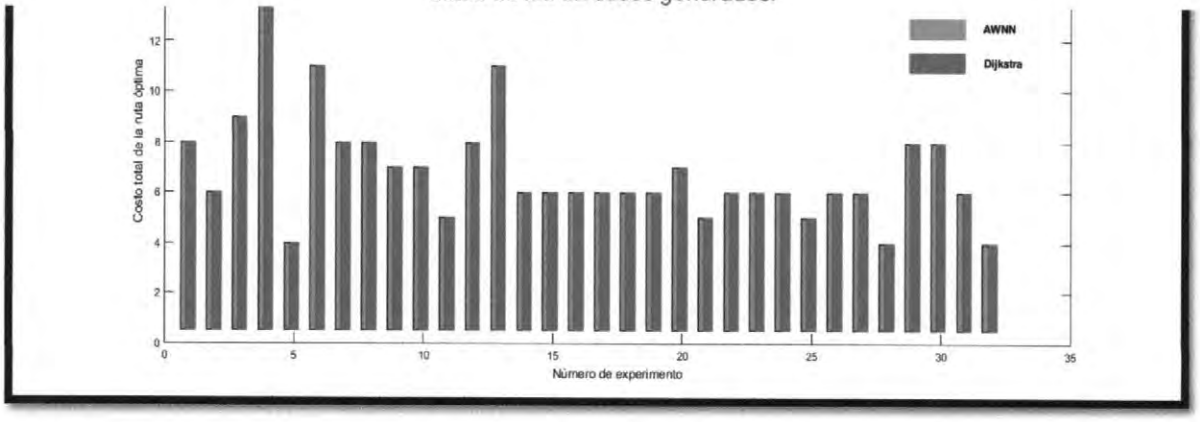


Figura 5.5 Gráfica comparativa en la calidad de la ruta óptima entre AWNN y Dijkstra para los 32 casos de la primera base de datos generada.

En la experimentación de la segunda base de datos de los 30 casos generados se ejecutaron 100 veces cada experimento con los algoritmos AWNN [Ma, 2010], AWNN con la API de OpenMP y Dijkstra.

En la Figura 5.7 se muestra la gráfica del tiempo mínimo, máximo y promedio de procesamiento al experimentar en AWNN la segunda base de datos.

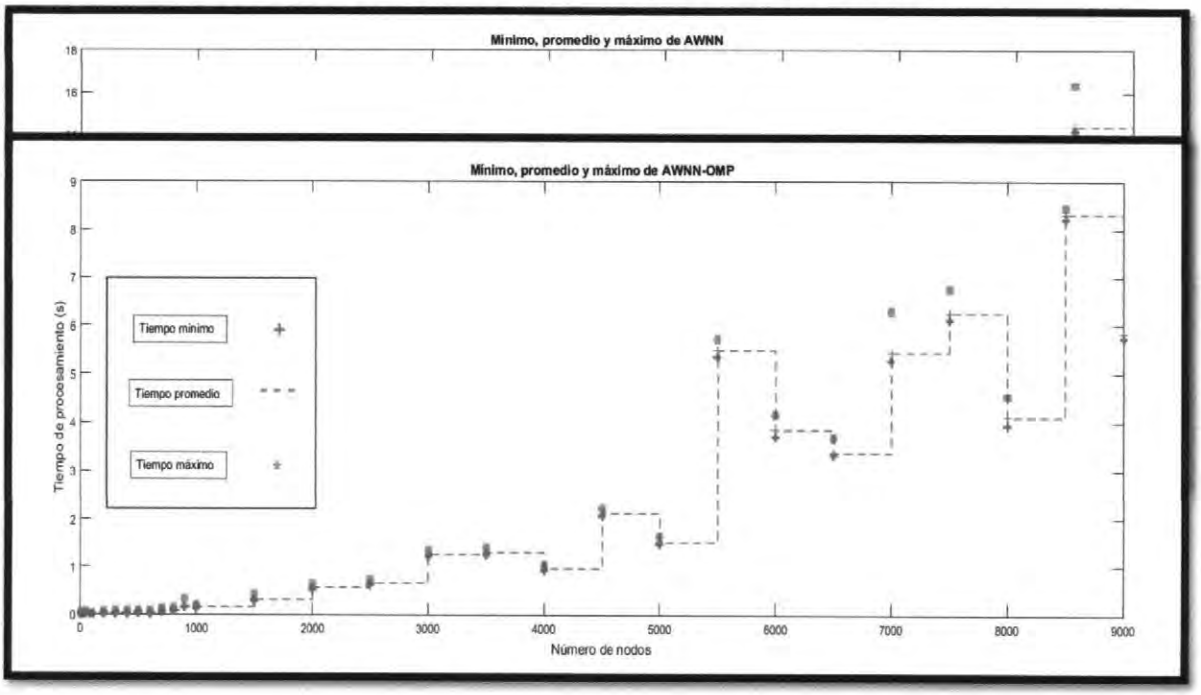


Figura 5.8 Mínimo, promedio y máximo del tiempo de procesamiento en la experimentación con AWNN_OMP de la segunda base de datos de los 30 casos generados.

En la Figura 5.8 se muestra la gráfica del tiempo mínimo, máximo y promedio de procesamiento al experimentar en AWNN_OMP la segunda base de datos.

En la Figura 5.9 se muestra la gráfica del tiempo mínimo, máximo y promedio de procesamiento al experimentar en Dijkstra la segunda base de datos.

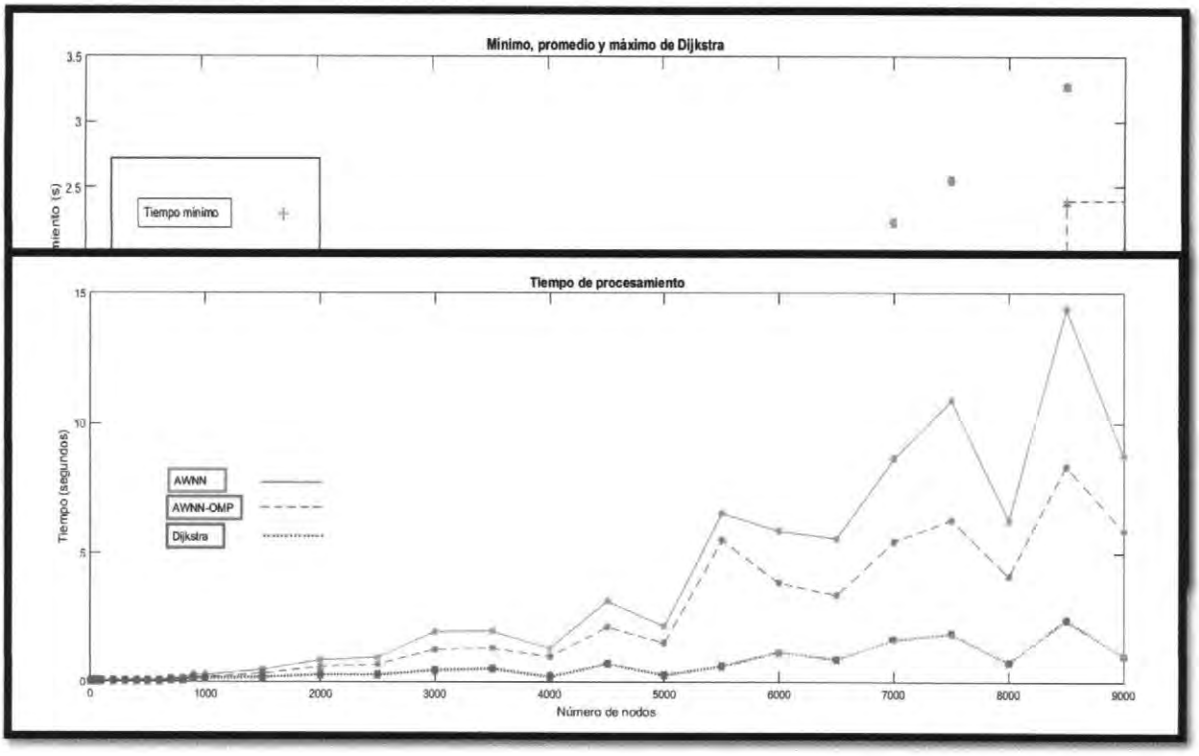


Figura 5.10 Gráfica comparativa en tiempo de procesamiento (tiempo promedio) entre AWNN, AWNN_OMP y Dijkstra para los 30 casos de la segunda base de datos generada.

En la Figura 5.10 se muestra la gráfica comparativa entre AWNN [Ma, 2010], AWNN implementado con la API de OpenMP y Dijkstra en tiempo de procesamiento (tiempo promedio) para la segunda base de datos de 30 casos por el "Generador de casos de prueba".

5.6.3. Caso real

En la experimentación se utilizó la red de carreteras de Oldenburg [Brinkhoff, 2002], la Figura 5.11 muestra el mapa correspondiente a la experimentación.

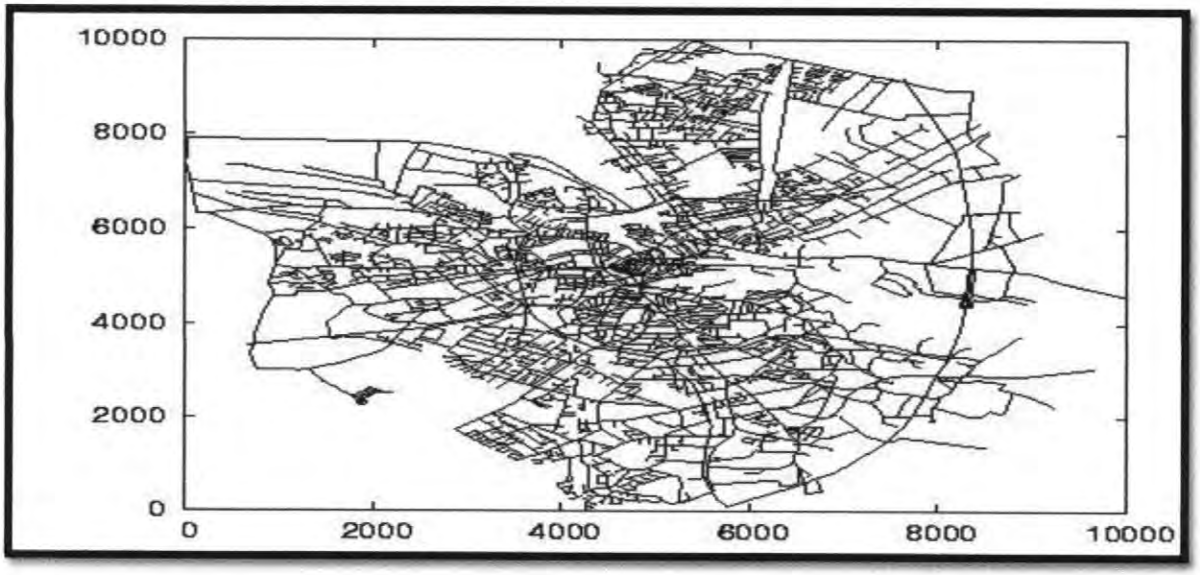


Figura 5.11 Red de carreteras de Oldenburg, Alemania [Brinkhoff, 2002].

5.7. Estimación de una implementación completamente paralela de AWNN

Tomando en cuenta la naturaleza paralela del modelo AWNN derivado de las PCNN se realiza la estimación de una completa implementación paralela en hardware especializado.

La estimación se realizó de acuerdo a lo mencionado en la literatura al dividir el tiempo de procesamiento entre el número de nodos de cada experimento al suponer que se tiene un núcleo por cada (Figura 5.12).

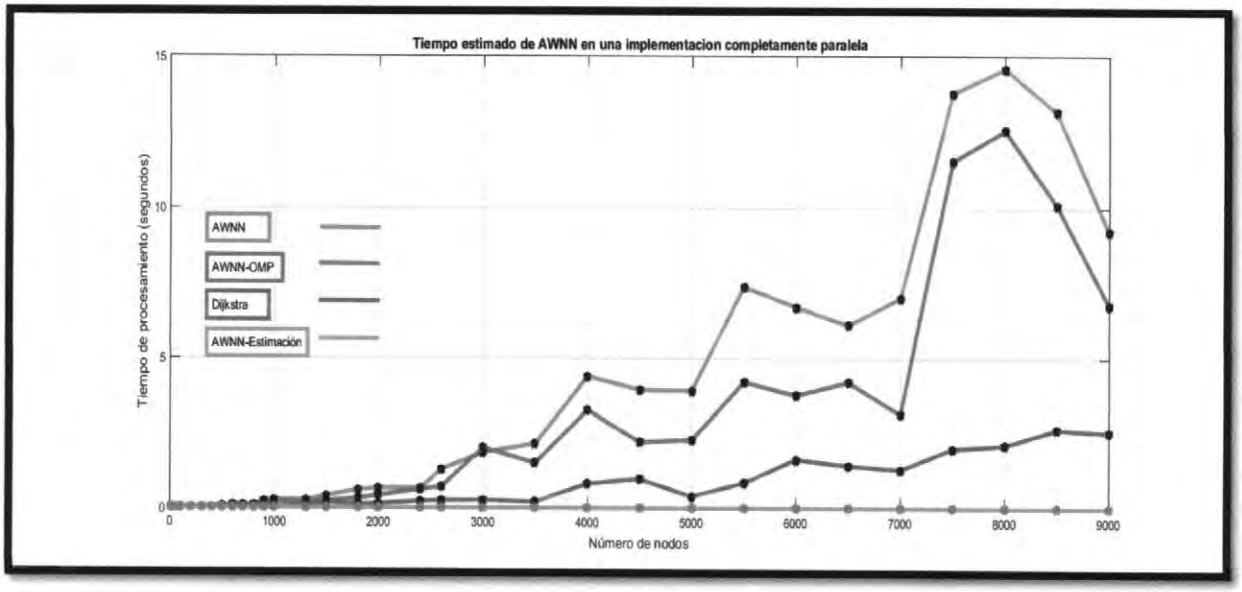


Figura 5.12 Gráfica del tiempo estimado de AWNN totalmente paralelizado comparado con AWNN, AWNN_OMP y Dijkstra.

En la Figura 5.13 se muestra una gráfica de estimación del tiempo de procesamiento obtenido (color cian) en una escala logarítmica con respecto al tiempo de procesamiento para apreciar el comportamiento de la experimentación y la disminución con respecto a Dijkstra.

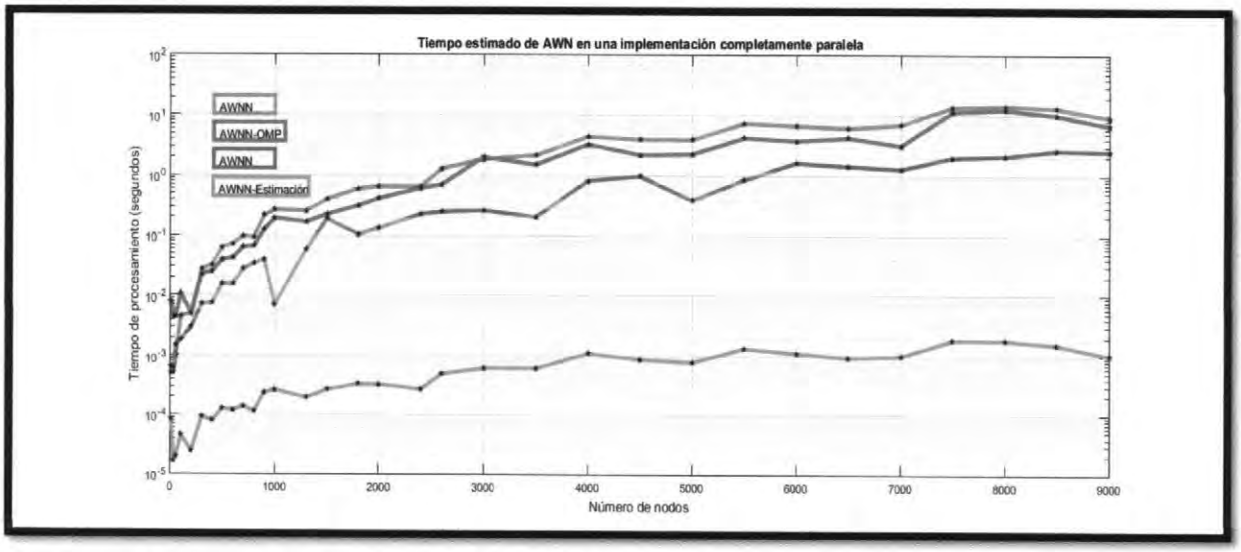


Figura 5.13 Gráfica del tiempo estimado de AWNN totalmente paralelizado en escala logarítmica del tiempo.

5.8. Discusión

Se puede observar que los resultados de la experimentación con la base de datos de los casos planteados en la literatura son afectados por factores externos al algoritmo como el hardware y software en el cual se experimentó ya que dichos casos están compuestos de dimensiones pequeñas que no sobrepasan los 15 nodos, pero por otro lado permitieron corroborar el buen funcionamiento del algoritmo antes de experimentar con las demás bases de datos.

En la comparación del tipo de bases de datos por el "Generador de casos de prueba" en tiempo de procesamiento se observa una clara ventaja del algoritmo Dijkstra con

respecto a AWNN y a AWNN implementado con la API de OpenMP (señalando que el hardware utilizado solo consta de 4 núcleos físicos).

En la experimentación de las tres bases de datos se muestra un comportamiento idéntico en los resultados entre AWNN y Dijkstra con respecto a la calidad de la ruta y el espacio de búsqueda explorado durante el proceso de búsqueda de la ruta óptima en cada experimento.

Para hacer un análisis a fondo del comportamiento de AWNN vs Dijkstra con respecto al tiempo de procesamiento se graficó el tiempo de procesamiento mínimo, máximo y promedio de la segunda base de datos generada de 30 casos.

Capítulo 6

Conclusiones y trabajo a futuro

En este capítulo se presentan las conclusiones del trabajo, a través de la discusión de los objetivos iniciales, los alcances, los productos y aportaciones del trabajo desarrollado, así como una comparación entre ventajas y desventajas observadas a lo largo del proyecto, y finalmente se muestran algunos puntos detectados como trabajo futuro a esta investigación.

6.1. Objetivos logrados

El objetivo principal del proyecto de tesis fue *implementar, experimentar y evaluar la RNAP en el dominio de la optimización de trayectorias para la obtención de la trayectoria óptima o sub-óptima*.

En la Tabla 6.1 se muestran las actividades que dan cumplimiento a los objetivos propuestos de la tesis y la Tabla 6.2 muestra cómo se lograron los alcances.

Tabla 6.1 Actividades realizadas para el logro de los objetivos.

OBJETIVOS	ACTIVIDADES
1.-Conocer los conceptos básicos y el funcionamiento de las RNAPs en el dominio de optimización de trayectorias.	Se estudiaron los antecedentes de trabajos con PCNN en Cenidet. Se desarrolló un programa en lenguaje C++ de la Red Neuronal ICM (variante de la PCNN) para conocer este tipo de RNA.

Tabla 6.1 Actividades realizadas para el logro de los objetivos (continuación de la Tabla 6.1).

OBJETIVOS	ACTIVIDADES
2.-Estudiar y comprender como las RNAPs permiten resolver problemas de optimización de trayectorias.	Se estudió el estado del arte de trabajos relacionados externos a CENIDET. Se estudió el Marco teórico comprendido por Optimización de trayectorias, Teoría de Grafos y RNAP.
3.- Implementar distintos modelos de las RNAPs para resolver problemas de optimización de trayectorias.	Se implementó en lenguaje C++ una adaptación del modelo AWNN para el problema de la ruta más corta. Se creó e implementó un algoritmo de "Reconstrucción de Ruta para la adaptación del modelo AWNN" como aporte al modelo original. Se implementó en lenguaje C++ la adaptación del modelo AWNN con la API de OpenMP para el problema de la ruta más corta. Se implementó en lenguaje C++ el algoritmo Dijkstra para el problema de la ruta más corta.
4.-Experimentar con los modelos de RNAPs seleccionados en bases de datos de trayectorias conocidas y con casos de problemas planteados en la literatura.	Se desarrolló en lenguaje C++ un programa "Generador de casos de Prueba" de Grafos No Dirigidos (basado en Qu [2016]). Se experimentó con tres tipos de bases de datos: ➢ Casos generados de manera aleatoria. ➢ Casos presentados en la literatura. ➢ Caso de real de la red de carreteras de Oldenburg. Se experimentó con las distintas bases de datos las implementaciones de AWNN, AWNN_OMP y Dijkstra.
5.-Evaluar el desempeño de las RNAPs en el dominio de optimización de trayectorias con respecto a las técnicas tradicionales.	Se realizó la comparativa y análisis de las implementaciones en los tres tipos de bases de datos. Se analizó el desempeño del software "Grafos" con el algoritmo Dijkstra.
6.-Redactar un artículo sobre los resultados obtenidos.	Se publicó y presento el artículo "Red Neuronal Pulsante Explicitada para el Problema del Camino más Corto" en CIINDET 2018. Se publicó y presentó el artículo "Red Neuronal Pulsante Adaptada al Problema del Camino más Corto" en CICOS 2018. Se redactó y publicó el artículo "Neural Network for Shortest Path Problems Accelerated with Parallel Multi-Core Architecture" en ICMEAE 2018.

Tabla 6.2 Actividades realizadas para el logro de los alcances.

ALCANCES	ACTIVIDADES
1.-Estudiar los modelos propuestos de RNAPs para optimización de trayectorias.	Se analizaron 3 modelos derivados de PCNN reportados en el estado del arte.
2.- Seleccionar al menos tres variantes e implementar una de PCNN para problemas de optimización de trayectorias.	Se seleccionaron las PCNN: AWNN, PFPCNN y SAPCNN. Se implementó una adaptación del modelo AWNN para el problema del camino más corto. Se implementó la adaptación del modelo AWNN con la API de OpenMP para el problema del camino más corto. Se creó e implementó un algoritmo de "Reconstrucción de Ruta para la adaptación del modelo AWNN" como aporte al modelo original.
3.- Experimentar con al menos cinco bases de datos de trayectorias conocidas.	Se experimentó con tres tipos de bases de datos: ➢ Casos generados de manera aleatoria (desde 5 hasta 9000 nodos). ➢ Casos presentados en la literatura. ➢ Caso de real de la red de carreteras de Oldenburg.

6.2. Productos

Los entregables que originalmente se propusieron para el desarrollo de la tesis fueron: un documento del estado del arte, presentaciones de avance, el software de implementación de algoritmos de optimización de trayectorias con PCNN, reporte de resultados, documento del estudio comparativo, y un artículo de los resultados obtenidos y la tesis.

En marco de referencia se construyó con el contenido del estado del arte, que consistió en revisar los antecedentes institucionales en el CENIDET y los trabajos relacionados externos, los cuales involucran algún modelo derivados de las PCNN para problemas de optimización de trayectorias; además del marco teórico el cuál se elaboró de tres temas principales que engloban conceptos de Optimización de Trayectorias, Teoría de Grafos y Redes Neuronales Artificiales Pulsantes.

En base al desarrollo de la investigación se presentaron avances los cuales incluyeron un reporte del algoritmo implementado y el análisis del mismo, así como las bases de datos utilizadas para la experimentación.

Durante el desarrollo se generó software en lenguaje C++ de la implementación del algoritmo AWNN [Ma, 2010], un "Generador de casos de prueba" de Grafos No Dirigidos (basado en Qu [2016]), la implementación del algoritmo Dijkstra y de la API de OpenMP en el algoritmo AWNN, así como un algoritmo complementario de "Reconstrucción de la ruta óptima" para el modelo AWNN.

Al terminar la experimentación se generó un reporte de resultados en el cual se evalúa el algoritmo de PCNN implementado con respecto a Dijkstra y la implementación del modelo con OpenMP en las distintas bases de datos, para así conformar un estudio comparativo del desempeño en el problema del camino más corto.

Como productos de la investigación se generaron 3 artículos:

- Se publicó y presento el artículo "Red Neuronal Pulsante Explicitada para el Problema del Camino más Corto" en el XIV CIINDET 2018 con el registro ISBN 978-607-95255-8-3.
- Se publicó y presentó el artículo "Red Neuronal Pulsante Adaptada al Problema del Camino más Corto" en el XII CICOS 2018, también será publicado en la revista Programación Matemática y Software con registro ISSN 207-3283.
- Se publicó y presentó el artículo "Neural Network for Shortest Path Problems Accelerated with Parallel Multi-Core Architecture" en ICMEAE 2018 con ISBN 978-1-5386-9191-5/18.

Además, se realizaron actividades complementarias desarrolladas durante la maestría como:

- La asistencia y exhibición de posters en "Escuela de Inteligencia Artificial y Robótica 2017".
- Asistencia y exhibición de posters en el congreso ICMEAE 2017.
- La participación en el concurso 3M/T 2018 en CENIDET.

6.3. Aportaciones

Anteriormente las Redes Neuronales Artificiales de tercera generación han sido ampliamente exploradas dentro de CENIDET en los ámbitos de clasificación, seguimiento de señales, aproximación de funciones, predicción y reconocimiento de patrones.

Por lo que este trabajo aporta el entendimiento e implementación de una adaptación del modelo AWNN para comprender como la característica de la auto-onda de las PCNN ayudan a resolver problemas de optimización de trayectorias.

Al realizar el análisis del funcionamiento de AWNN se aporta la determinación de la inicialización del parámetro Delta-Umbral, el cual es el incremento del Umbral dinámico a través de las iteraciones de la red, este parámetro se puede establecer en 1 si los costos entre los nodos de la matriz de costos son enteros, de lo contrario se debe establecer en decimales de acuerdo al número encontrado en la matriz de costos previamente revisada (por lo que en este caso se recomienda una normalización de todos los costos para que sean enteros positivos).

Se aporta también la determinación de la inicialización del parámetro Umbral dinámico, el cual se define como el costo menor del nodo de inicio seleccionado a todos los demás nodos conectados a él, por lo que se hace una previa revisión de la fila del nodo de inicio en la matriz de costos.

Finalmente se hace una aportación al algoritmo original AWNN, la cual consiste en un algoritmo de "Reconstrucción de la ruta" a través de la explicitación del conocimiento de esta red, extrayendo el conocimiento que queda oculto en la matriz de salida Y, esta aportación consiste en un proceso secuencial inverso saltando del nodo final al nodo inicial.

6.4. Lecciones aprendidas

La realización de un proyecto de tesis permitió el desarrollo de habilidades como el análisis, identificación del problema, abstracción de conocimiento, planteamiento de objetivos y propuestas de solución.

Se aprendió un modelo de Red Neuronal Artificial de tercera generación especialmente creado para problemas de optimización utilizando como base la teoría de grafos y las características de la PCNN.

Se obtuvo conocimiento sobre el área de optimización de trayectorias, las características de sus problemas, en especial sobre el problema de la ruta más corta.

Se aprendió a desarrollar la capacidad de plantear algoritmos, como lo fue el algoritmo de "Reconstrucción de la ruta" para la adaptación del modelo.

6.5. Conclusiones finales

Se presentó un modelo de Red Neuronal Artificial que no se había estudiado en CENIDET para problemas de optimización de trayectorias, en donde se logró un entendimiento de cómo se resuelve el problema de la ruta más corta entre dos nodos y

se logró una aplicación a distintas bases de datos; además se evaluó su desempeño con respecto al algoritmo de Dijkstra.

Entre los puntos más importantes concluidos durante el desarrollo del proyecto son:

- Los patrones de conexión del modelo AWNN son apropiados para resolver problemas de Optimización de Trayectorias, en especial para el problema del camino más corto.
- La elección de los valores iniciales de los parámetros Umbral dinámico y Delta-Umbral en el modelo AWNN son de gran importancia para asegurar la obtención de la ruta óptima.
- La matriz de salida Y del modelo implementado permite la explicitación del conocimiento de la red; por lo cual se logró generar un algoritmo secuencial inverso de "Reconstrucción de la ruta" como aportación al modelo original.
- La naturaleza paralela en la que se basa el modelo AWNN permite implementar la API de OpenMP y lograr la optimización de este algoritmo obteniendo mejores resultados en tiempo de procesamiento.
- Se espera obtener resultados ampliamente eficientes contra los algoritmos secuenciales como Dijkstra en una implementación completamente paralela en hardware especializado (GPU).

6.5.1. Ventajas

Al final de este trabajo de maestría se llegó a identificar claramente las ventajas del modelo AWNN para el problema de la ruta más corta con respecto a Dijkstra:

- En la calidad de la ruta obtenida; si la elección de los parámetros iniciales (Delta-Umbral y Umbral dinámico) es la adecuada el algoritmo siempre se obtiene la ruta óptima, y en comparación con Dijkstra como se muestra en las gráficas 5.2 y 5.5 la calidad es la misma.

- En el porcentaje del espacio de búsqueda explorado durante el proceso de obtención de la ruta en comparación con el algoritmo Dijkstra es el mismo como se muestra en las gráficas 5.3 y 5.6.

6.5.2. Desventajas

Al final de este trabajo de maestría se llegó a identificar las desventajas del modelo AWNN para el problema de la ruta más corta con respecto a Dijkstra:

- El tiempo de procesamiento del modelo AWNN (implementado de forma SECUENCIAL) comparado con el algoritmo Dijkstra es mayor.
- Se necesita establecer la inicialización de los valores de los parámetros Delta-Umbral y umbral dinámico adecuados para garantizar el resultado de la ruta óptima.

Como conclusión final, la Red Neuronal presentada en esta Tesis, debido a su naturaleza paralela derivada de las PCNN, presenta una solución altamente competitiva si se realiza una implementación paralelizada usando miles de núcleos de la Unidad de Procesamiento de Gráficos (GPU), esto basado en:

- La etapa de experimentación en la base de datos de 32 grafos de 6 hasta 9000 nodos, al implementar la API de OpenMP utilizando 4 núcleos el tiempo de procesamiento disminuyo en un 32.6 de porcentaje general, por lo que en la estimación realizada de una implementación completamente paralela como se muestra en las Figuras 5.12 y 5.13 se obtendrían resultados ampliamente favorables para el paradigma presentado.
- El análisis de complejidad algorítmica de la sección 4.5 en donde se plantea que la complejidad algorítmica del modelo propuesto es $O(MN^2)$ siendo M el número de iteraciones necesarias para encontrar el camino más corto entre dos nodos y N el número de nodos en el grafo y gracias a su naturaleza paralela las neuronas pueden ser estimuladas al mismo tiempo, por lo cual la complejidad algorítmica del procesamiento paralelo entre neuronas de este paradigma seria $O(M)$ si se

tuviera un procesador por neurona requerida, a lo cual el tiempo de procesamiento T en disminuiría a T/N^2 .

6.6. Trabajos a futuro

El trabajo futuro de la investigación puede centrarse principalmente en tres aspectos:

- Inicialización automática del parámetro Delta-Umbral en caso de que los valores de la matriz de costos no sean enteros, o, dicho de otra forma, se genere la normalización automática de la matriz de costos.
- Otra forma para evitar la normalización seria proponer una mejora al ajuste del Umbral dinámico, es decir, que el Delta Umbral se autoajustara para incrementar el Umbral dinámico a través de las iteraciones de la red.
- Implementación de la adaptación del modelo AWNN en hardware especializado para paralelización aprovechando la naturaleza paralela derivada de PCNN y evaluar la optimización en tiempo de procesamiento.

Referencias

[Aichholzer, 2005]	**Oswin Aichholzer, Departamento de Informatica de la Universidad de Valladolid, "I. T. Informatica de Gestion, Curso "2005/2006". Apuntes de Grafos.**
[Arista, 2012]	R. V. A.Arista, «Implementación ConFigurable y Multipropósito de Redes neuronales de tercera Generación en GPUs,» Memorias del primer concurso de Investigación, desarrollo e inovación tecnológica, pp. 35-38, 2012.
[arodrigu, 2003-2012].	http://arodrigu.webs.upv.es/grafos/doku.php?id=software (08/02/2018).
[Brinkhoff,2002]	https://www.cs.utah.edu/~lifeifei/SpatialDataset.htm **(08/02/2018)**
[Bogomolny, 2017].	Bogomolny,A."gráficos".Disponible en:http://www.cut-the-knot.org/do_you_know/graphs.shtml. Consultado: 30/05/2017.
[Cárdenas, 2015]	S. Y. Cárdenas. Implementación y Evaluación de Redes Neuronales Artificiales tipo "Pulse-Coupled Neural Networks" (PCNN) Aplicadas a Visión Artificial. Tesis de Maestría en Ciencias de la Computación, CENIDET. 2015.
[Caulfield, 1999]	Caulfield HJ, Kinser M. Finding the path in the shortest time using PCNNs. IEEE Trans Neural Networks 1999; 10(3):604–6.
[Eckhorn,1990]	R.Eckhorn, H.J.Reitboeck, M.Arndt, P.Dicke, Feature linking via synchronization among distributed assemblies: simulation of result from cat visual cortex, Neutral Computation 2 (3) (1990) 293–307.
[Ekblad, 2004].	Ekblad, Ulf,: The intersecting cortical model in image processing. Nuclear Instruments and Methods in Physics Research subsection A: Accelerators, Spectrometers, Detectors and Associated Equipment 525.1: 392-396, 2004.
[Hernández, 2017]	C. Hernandez. B. , Aplicación del Descenso de Gradiente para el Aprendizaje de Neuronas Pulsantes de Izhikevich, Tesis de Maestría en Ciencias de la Computación, CENIDET. 2017.
[Hopfield, 1985]	Hopfield JJ, Tank DW. "Neural" computation of decisions in optimization problems. Biol Cybernet 1985; 52:141–52.
[Huang, 2017]	Huang, Wie., Yan, Chunwang. Y Jisong y Wie Wang. A time-delay neural network for solving time-dependent shortest path problem, Neural Networks 90 (2017) 21–28.
[Izhikevich, 2004]	Izhikevich, E.M., Which model to use for cortical spiking neurons?, Neural Networks, IEEE Transactions on 2004, Volume: 15, Issue: 5 Pages: 1063 – 1070.

[John, 1999] H, John. Y J. M, Kinser, Finding the Shortest Path in the Shortest Time Using PCNN's. IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 10, NO. 3, MAY 1999.

[Lindblad, 2005] T.Lindblad and J. M. Kinser, *Image Processing Using Pulse-coupled Neural Networks.* Springer, 2005.

[Ma, 2010] Yi. Ma, K. Zhan, and Z. Wang, *Applications of Pulse-Coupled Neural Networks*. Springer, 2010, pp. 147–166.

[Ma, 2011] Yi. Ma, K. Zhang y X. Feng, The application of the combinatorial optimization problems based on Preventive Feedback Pulse Coupled Neural Network, Lanzhou University Lanzhou China,2011.

[Ma, 2013] Y. Ma., Li.Xiaojun. Li., Feng. Xiaowen., Self-adaptive autowave pulse-coupled neural network for shortest-path problem, Neurocomputing 115(2013)63–71.

[Mathivet, 2015] Mathivet, Virginie. Inteligencia Artificial para Desarrolladores: Conceptos e Implementación en C#. Ediciones ENI,2015. ISBN: 9782747098855.

[Moustapha,2016] Moustapha. D., K. Mark, Advances in combinatorial optimization, World Scientific, 2016.

[Ortiz, 2017] E.Ortiz. Deteccion de Ruido Impulsivo o Gaussiano en Imagenes Monocromaticas mediante Redes Neuronales Pulso-Acopladas, Tesis de Maestría en Ciencias de la Computación, CENIDET. 2017.

[Qu, 2007] Qu, Hong. Y Yi, Zhang, A new algorithm for finding the shortest paths using PCNN, Chaos, Solitons and Fractals 33 (2007) 1220–1229, 2007.

[Qu, 2012] Qu, Hong., Yang, Simon X.,Yi, Zhang. Y Wang, Xiaobin, *A novel neural network method for shortest path tree computation*. Applied Soft Computing 12 (2012) 3246–3259.

[Qu, 2013] Qu. Hong., Yi, Zhang. Y Simon X. *Efficient Shortest-Path-Tree Computation in Network. Routing Based on Pulse-Coupled Neural Networks.* IEEE TRANSACTIONS ON CYBERNETICS, VOL. 43, NO. 3, JUNE 2013.

[Qu, 2015] Qu. Hong., Liu. Guisong., Qiu. Zhao., Ji. Luping., Computing k shortest paths using modified pulse-coupled neural network, Neurocomputing 149 (2015) 1162–1176.

[Qu, 2016 Qu, Hong., Sang, Y., Lv,Jiancheng., Y Yi, Zhang, Shortest path computation using pulse-coupled neural networks with restricted autowave.Knowledge-Based Systems 114 (2016) 1–11.

[Ranganath, 1995] H. S. Ranganath, G. Kuntimad, and J. L. Johnson, *Pulse coupled neural networks for image processing*. in Proc. IEEE Southeastcon, Raleigh, NC, 1995

[Ramos, 2010]. Ramos. Andres, Sanchez. Pedro, Ferrer. J. Maria, Baquín. Julián, Linares. Pedro, MODELOS MATEMÁTICOS DE OPTIMIZACIÓN, Alberto Aguilera 23- E28015 Madrid,, septiembre 2010.

[Shao-Fa, 2010] Shao-Fa L, Kai HE, Cheng W,. Modified PCNN Model and Its Application to Mixed-noise Removal. Int Conf Innov Comput Commun 2010 Asia-Pacific Conf Inf Technol Ocean Eng. 2010;213(2):213-216. doi:10.1109/CICC-ITOE.2010.61.

[Vázquez, 2010] Vázquez, R., «Pattern Recognition Using Spiking Neurons and Firing Rates,» Proceedings of the 12th Ibero-American conference on advances in artificial intelligence, pp. 423-432, 2010.

[Vieites, 2014] Veites. R. A.María, Aguado. M. Felicidad.,Teroría de grafos: Ejercicios ressueltos y propuestos, Ediciones parainfo S. A, 2014.

[Zarate, 2015]. M.B. Zarate. Extracción de Características de Imágenes Digitales mediante una Red Neuronal Artificial Pulsante. Tesis de Maestría en Ciencias de la Computación, CENIDET. 2014.

[Zhang, 2011] Yudong ZHANG, Lenan WU. A Novel Algorithm for APSP Problem via a Simplified Delay Pulse Coupled Neural Network. Journal of Computational Information Systems 7:3 (2011) 737-744

[Zhao, 2009] Dongming Zhou, Rencan Nie, Dongfeng Zhao. Analysisofautowavecharacteristicsforcompetitivepulsecoupledneural Networkanditsapplication. Neurocomputing72(2009)2331–2336.

Acrónimos

API	Application Programming Interface / Interfaz de programación de aplicaciones.
APSP	"all pairs shortest path" problem / el problema de todos los pares más cortos.
AWNN	Auto-wave Neural Network / Red Neural de la Onda Automática.
$CPCNN_1$	Controlled PCNN / Red Neuronal Pulso-Acoplada Controlada.
$CPCNN_2$	Competitive Pulse Coupled Neural Network / PCNN competitiva.
CPU	Central Processing Unit / Unidad Central de Procesamiento.
ICM	Intersecting Cortical Model / Modelo cortical intersecante.
KSP	K Shortest Paths / (K rutas más cortas).
LA	Lista de adyacencia.
MA	Matriz de adyacencia.
MCPCNN	Modified Continued Pulse Coupled Neural Network / (Red Neuronal Modificada Continua Pulso-Acoplada).
M-PCNN	Modified of Pulse-Coupled Neural Networks / Modificado PCNN.
MPCNNs	Multi-output model of pulse coupled neural networks / modelo de Múltiples-Salidas de Redes Neuronales Pulso-Acopladas.
OpenMP	Open Multi-Processing /Multi-proceso abierto.
PCM	Pixel Correspondance Metric / Métrica de correspondencia de pixel.
PCNN	Pulse-Coupled Neural Network/ Red Neural Pulso-Acoplada.
PFPCNN	Preventive Feedback Pulse Coupled Neural Network/ Red Neuronal Acoplada por Pulsos de Respuesta Preventiva.

RNA	Red Neuronal Artificial.
RNAP	Redes Neuronales Artificiales Pulsantes.
SAPCNN	Self-adaptive autowave pulse-coupled neural network / Red Neuronal Pulso-Acoplada de la Onda Automática Auto-adaptable.
SCM	Spiking Cortical Model / Modelo cortical Spiking.
SDPCNN	Simplified Delay PCNN / Red Neuronal Pulso-Acoplada de Retardo Simplificado.
SP	Shortest path / Ruta más corta.
SPT	Shortest path tree / Árbol del trayecto más corto.
TCPCNN	El Tristate Cascading Pulse Couple Neural Network / Red Neuronal Pulso-Acoplada de tri-estado en cascada.
TDNN	Time-Delay Neural Network / Red Neuronal de Retardo de Tiempo.
TD-SP	Time-Dependent Shortest Path / Trayecto más Corto Dependiendo del Tiempo.
T.Ó.S-Ó	Trayectoria óptima o sub-óptima.
TSP	The Traveling Salesman Problem / el problema del agente viajero.
VLSI	Very Large Scale Integration / Integración a escala muy grande.