ANÁLISIS DEL DESEMPEÑO DE UNA RED OBS DISTRIBUIDA MEDIANTE LA INTEGRACION DE UN METODO COGNITIVO BASADO EN METAHEURISTICA HIBRIDA PARA EL ENSAMBLE DE RAFAGAS

ANEXOS

Juan Camilo Bravo Flórez

Juan David Prado López

Director: Ph.D. Ing. José Giovanny López Perafán

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
Grupo de I+D Nuevas Tecnologías en Telecomunicaciones
GNTT
Línea de Investigación Señales y Sistemas de Telecomunicaciones
2015

JUAN CAMILO BRAVO FLÓREZ
JUAN DAVID PRADO LÓPEZ.

ANÁLISIS DEL DESEMPEÑO DE UNA RED OBS DISTRIBUIDA MEDIANTE LA INTEGRACION DE UN MÉTODO COGNITIVO BASADO EN METAHERÍSTICAS HIBRIDAS PARA EL ENSAMBLE DE RÁFAGAS

Trabajo de Grado presentado como requisito parcial para la obtención del título de:

Ingeniero en Electrónica y Telecomunicaciones.

Director:
Ph.D. Ing. José Giovanny López Perafán

Popayán
2015

TABLA DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

LISTA DE ACRÓNIMOS

ANN	*Artificial Neural Network,* Redes Neuronales Artificiales.
NP	*Non deterministic Polynomial*, Polinomial no Deterministico.
NP-C	*Non deterministic Polynomial Complete*, Polinomial no Deterministico Completo.
P	*Polynomial*, Polinomial.
PSO	*Particle Swarm Optimization*, Optimización por Enjambre de Particulas.

ANEXO A: EXTENSIÓN MARCO TEÓRICO

A.1 Problemas P, NP y NP-Completo

Existen problemas que tienen solución con orden lineal, los cuales se cuales clasifican de acuerdo a la complejidad o coste computacional polinómico que tienen que ejecutar los computadores para resolver los problemas mediante algoritmos. Los algoritmos de complejidad polinómica se dice que son tratables en el sentido que son abordables en la practica. están agrupados en la clase Polinómial (P, *Polynomial*).

Por otro lado se encuentran los problemas de tipo Tiempo Polinómial no Determinístico (NP, *Non deterministic Polynomial*), los cuales no tiene una solución practica, es decir, son el tipo de problemas que pueden ser resueltos en tiempo polinómico por una maquina de Turing[1] no determinista. El problema de tipo Polinómial no Determinístico–Completo (NP-C, *Non deterministic Polynomial–Complete*) no tiene solución practica en un tiempo razonable, por lo que lo convierte en problemas de extrema complejidad. En la figura 1 se muestra las clases de complejidad de los problemas [1].

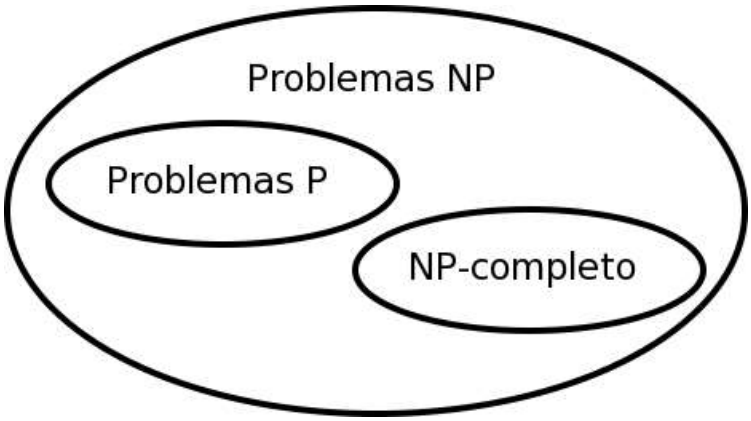

Figura A. 1. Diagrama de clases de complejidad. Tomada de

[1] Es un dispositivo que manipula símbolos sobre una tira de cinta de acuerdo a una tabla de reglas

A.2 Redes Neuronales Artificiales (ANN, Artificial Neural Network)

Las Redes Neuronales Artificiales (ANN, Artificial Neural Network) imitan la estructura hardware del sistema nervioso, con la intención de construir sistemas de procesamiento de la información paralelos, distribuido y adaptativos que puedan presentar un cierto comportamiento inteligente.

A pesar del gran desarrollo tecnológico de las ciencias de la computación ningún computador puede llevar a cabo tareas tan simples como los cerebros de animales hacen por naturaleza. En la Tabla A.1 se puede apreciar la diferencia de procesamiento entre un cerebro y un computador convencional [2] [3]:

Tabla A. 1. Cerebro frente a computador convencional. Tomada de[4]

	Cerebro	Computador
Velocidad de proceso	$\approx 10^{-2} seg\ (100\ Hz)$	$\approx 10^{-9} seg\ (GHz)$
Estilo de procesamiento	paralelo	Secuencial
Numero de procesadores	$10^{11} - 10^{14}$	pocos
Conexiones	10.000 por procesador	pocas
Almacenamiento del conocimiento	distribuido	direcciones fijas
Tolerancia a fallos	amplia	nula
Tipo de control del proceso	auto-organizada	centralizado

Como se puede observar el cerebro y el computador convencional varían de una manera sustancial. Mientras que el computador en escancia es una maquina von Neumann con un único procesador que ejecuta de modo secuencial un programa almacenado en memoria, el cerebro esta compuesto por miles de millones de procesadores (Neuronas).

Los elementos básicos de un sistema neuronal biológico son las neuronas, las cuales se agrupan en conjuntos compuestos por millones de estas organizadas en capas. Un conjunto de estos subsistemas da lugar a un sistema global. Un sistema neuronal puede establecerse de manera similar. El elemento principal es la neurona artificial que se organizará en capas; varias capas constituirán una red neuronal y por ultimo una red neuronal mas los módulos convencionales adicionales necesarios constituyen un sistema global de proceso, lo anterior se puede observar en la *figura 1.*

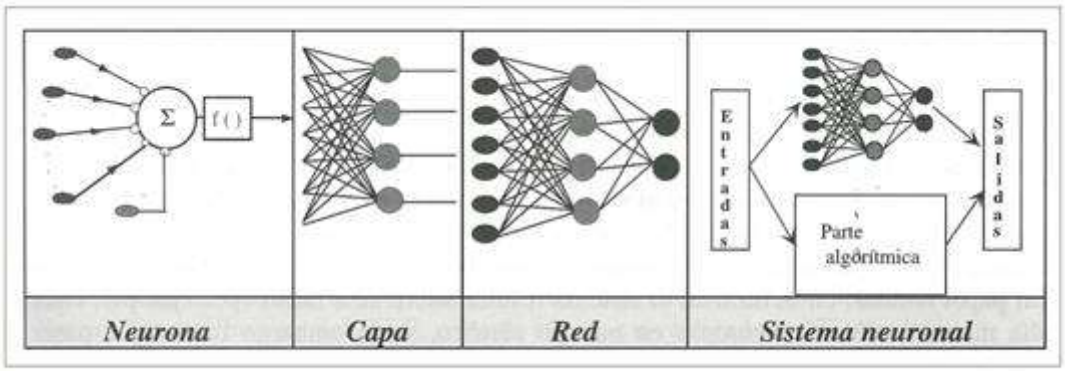

Figura A. 2. Estructura jerárquica de un sistema basado en ANN. Tomada de[4]

A.2.1 Modelo general de una neurona artificial

Las neuronas se definen como dispositivos de procesamiento que proporcionan una única respuesta a partir de un vector de entrada procedente del exterior. Esta neurona basa su funcionamiento en la regla de propagación, la cual permite obtener a partir de las entradas y los pesos, el valor del potencial postsináptico h_i de la neurona

$$h_i = \sigma_i\left(w_{ij}, x_j(t)\right) \qquad (A.1)$$

La función mas habitual es de tipo lineal, y se basa en la suma ponderada de los pesos sinápticos de las entradas

$$h_i = \sum_j w_{ij} x_j \qquad (A.2)$$

Los pesos sinápticos se definen como el grado de interacción que hay entre la neurona postsináptica i y la presináptica j, en el caso de ser positiva se tendrá que excitar a la neurona postsináptica y en caso contrario hay que inhibirla.

Estos son los conceptos mas básicos de cómo definir una neurona, en la practica es muy común utilizar a la que muchos autores denominan como neurona estándar, en donde se considera la suma ponderada expuesta en la ecuación (A.2) como regla de propagación. La neurona estándar esta compuesta por:

- Un conjunto de entradas $x_j(t)$ con sus correspondientes pesos sinápticos w_{ij}
- Una regla de propagación $h_i(t) = \sigma(w_{ij}, x_j(t); h_i(t) = \sum w_{ij}x_j$
- Una función de activación $y_i(t) = f_i(h_i(t))$, representa la salida de la neurona

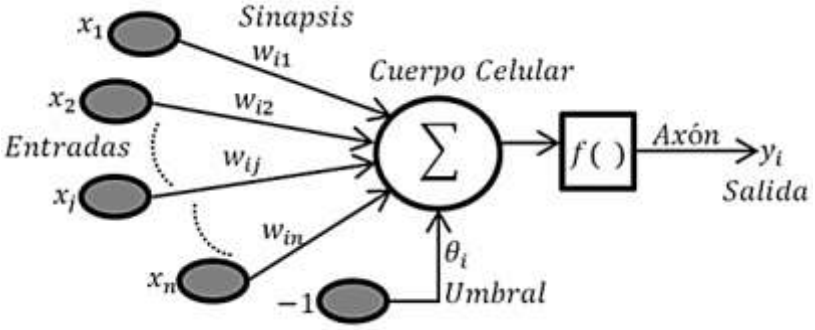

Figura A. 3. Modelo de neurona estándar. Tomada de [4]

La regla de propagación mas comúnmente utilizada consiste en combinar linealmente la entradas y los pesos sinápticos, en donde, suele ser habitual añadir al conjunto de pesos de la neurona un parámetro adicional denominado umbral θ_i, el cual se le resta al potencial postsináptico, en donde se tiene la siguiente ecuación

$$\sum_j w_{ij}x_j - \theta_i \qquad (A.3)$$

la denominada neurona estándar tiene como función de activación, en donde se debe establecer una función de activación mas adecuada para modelar el problema. En la siguiente ecuación se define la función de activación de una neurona estándar

$$y_i(t) = f_i(\sum_j w_{ij}x_j - \theta_i \qquad (A.4)$$

A partir de estos parámetros es como se hace el diseño de la arquitectura de una red neuronal, en donde se deben seleccionar los parámetros de aprendizaje del algoritmo, los puntos clave en una metodología de diseño y por ultimo la etapa de la consolidación de la red neuronal con la definición de cada una de sus capas.

A.2.2 Clasificación de los modelos neuronales

Dependiendo del modelo de neurona concreto que se utiliza de la arquitectura o topología de conexión y del algoritmo de aprendizaje los modelos de redes neuronales se pueden clasificar de la siguiente manera:

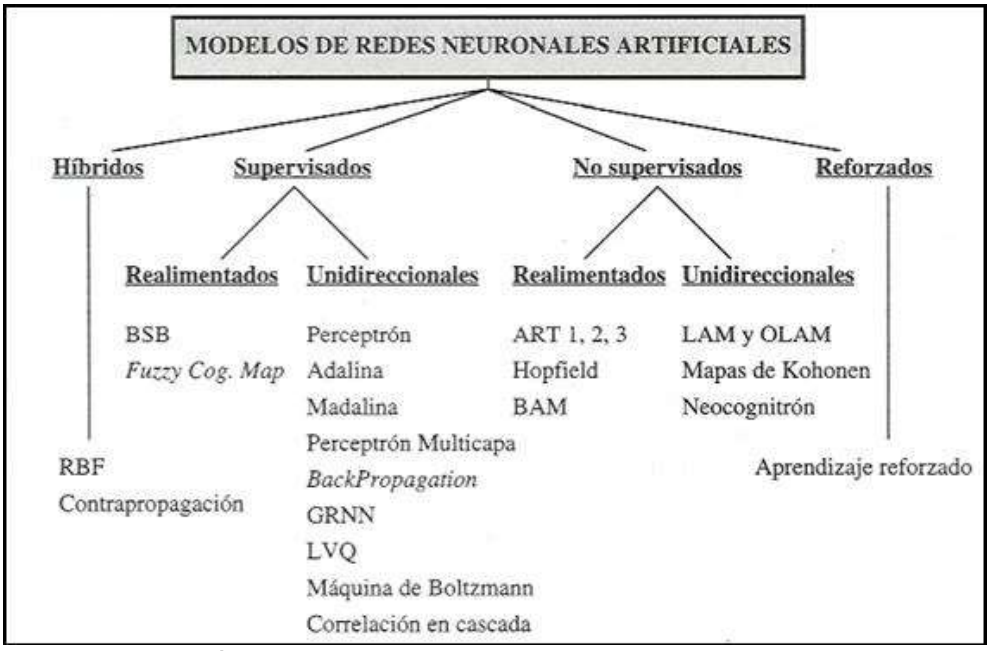

Figura A. 4. Clasificación de los ANN por el tipo de aprendizaje y arquitectura. Tomada de [4]

Para el presente trabajo de grado se selecciono la red neuronal Perceptrón Multicapa clasificado dentro de las redes unidireccionales.

A.2 Optimización por enjambre de partículas (PSO, Particle Swarm Optimization)

La optimización por enjambre de partículas es una técnica metaheurística poblacional basada en la naturaleza, más específicamente en el vuelo de las bandadas de aves y el movimiento de los bancos de peces.

Una breve descripción de dicho algoritmo se presenta a continuación: los individuos o partículas que conviven en una sociedad tienen una “opinión” que es parte del espacio de búsqueda, compartido por los individuos. Cada uno de estos individuos puede modificar su opinión de acuerdo a tres factores específicos, a saber:

- El conocimiento de su entorno o adaptación.
- Experiencias anteriores del individuo o memoria del individuo.
- Experiencias anteriores de los individuos del vecindario o memoria del vecindario.

La forma en que los individuos adaptan sus opiniones es identificando los individuos de más éxito en su entorno y modificando dichas opiniones por las más exitosas. Con el tiempo, los individuos de un entorno tienen un conjunto de opiniones bastante relacionado [5].

PSO es un sistema multi agente. Las partículas son agentes simples que se mueven en un espacio de búsqueda, guardan y posiblemente comunican la mejor solución que han encontrado.

Las principales características del algoritmo PSO son las siguientes [6]:

- Las partículas intercambian información. Estas modifican su dirección en función de las direcciones de las partículas de su vecindario.
- PSO almacena la experiencia propia de cada partícula y esta decide su nueva dirección de acuerdo a la mejor posición por la que pasó anteriormente.
- Tiene una rápida convergencia a buenas soluciones.
- La población inicial del algoritmo se genera de manera aleatoria y evoluciona al pasar las iteraciones.
- La búsqueda persigue la solución más óptima posible.
- La búsqueda se basa en los valores de la función objetivo
- PSO tiene operadores de movimiento pero no de evolución como la mutación o el cruce.
- PSO no crea nuevas partículas durante su ejecución sino que siempre son las mismas partículas iniciales modificadas a lo largo del proceso.

Para describir el algoritmo se debe estudiar las características de una partícula. Una partícula está compuesta por tres vectores y dos valores de aptitud con respecto al problema considerado. Los vectores se mencionan a continuación [7].

- El vector $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,N})$ almacena la posición actual de la partícula.
- El vector $mejorpos_i = (mejorpos_{i,1}, mejorpos_{i,2}, \dots, mejorpos_{i,N})$ almacena la posición de la mejor solución encontrada por la partícula hasta el momento.

- El vector velocidad $v_i = (v_{i,1}, v_{i,2}, \dots, v_{i,N})$ almacena la dirección según la cual se moverá la partícula.

Los dos valores de aptitud son los siguientes :

- El valor de $aptitud\_x_i$, almacena el valor de adaptación o adecuación de la posición actual correspondiente al vector $\mathbf{x_i}$.
- El valor de $aptitud\_mejorpos_i$, almacena el valor de adecuación de la partícula con mejor solución local encontrada hasta el momento correspondiente al vector $\boldsymbol{mejorpos_i}$.

Expuesto lo anterior, la descripción del algoritmo se describe a continuación [8]:

1. El enjambre se inicializa generando las posiciones de forma aleatoria.
2. Se generan las velocidades aleatoriamente en un intervalo establecido distinto de cero.
3. Se calcula la *aptitud* de cada partícula y se actualizan los valores de $aptitud_{x_i}$ y $aptitud\_mejorpos_i$.
4. Las partículas se mueven de una posición a otra en cada iteración. Al vector de posición x_i se le añade el vector velocidad v_i para obtener un nuevo vector x_i.
5. Con la nueva posición de la partícula se calcula y actualiza $aptitud_{x_i}$.
6. Si el nuevo valor de *aptitud* es el mejor encontrado por la partícula i hasta el momento, se actualizan los valores de $mejorpos_i$ y $aptitud\_mejorpos_i$.
7. Si el nuevo valor de $aptitud\_mejorpos_i$ es el mejor encontrado por el enjambre de partículas hasta el momento, se actualiza el valor de la mejor posición del enjambre *mejorpos* y su *aptitud_mejorpos.*
8. El vector velocidad de cada partícula es modificado en cada iteración utilizando la velocidad anterior, un componente cognitivo y un componente social. El modelo matemático resultante viene dado por la siguiente ecuación:

$$v_i^t \leftarrow w^{t-1} \cdot v_i^{t-1} + \varphi_1 \cdot rand_1 \cdot \left(mejorpos_i - x_i^{t-1}\right) + \varphi_2 \cdot rand_2 \cdot \left(mejorpos - x_i^{t-1}\right) \quad (A, 5)$$

$$x_i^t \leftarrow x_i^{t-1} \cdot v_i^t \qquad (A, 6)$$

Para $i = 1, 2, \dots, P$

Donde:

- x_i^t ≡ vector posición de la partícula i en la iteración t.
- v_i^t ≡ vector velocidad de la partícula i en la iteración t.
- w^t≡ factor de inercia en la iteración t.
- φ_1, φ_2 ≡ son pesos que controlan los componentes cognitivo y social.
- $rand_1$ ≡ número aleatorio entre 0 y 1.
- $rand_2$ ≡ número aleatorio entre 0 y 1.
- $mejorpos_i$ ≡ mejor posición encontrada por la partícula i hasta el momento que posee la mejor solución.
- $mejorpos$ ≡ representa la posición de la partícula con la mejor solución o aptitud.
- P ≡ número de partículas que componen la nube.

De la ecuación $(A.5)$ se actualiza el vector velocidad de cada partícula i en la iteración t. De la misma manera, la ecuación $(A.6)$ actualiza el vector posición de la partícula i para cada iteración.

De la ecuación $(A.5)$ se puede observar que el primer término es el vector velocidad de la iteración anterior, por lo que esto indica que el algoritmo tiene memoria y por tanto es cognitivo. Este componente cognitivo le permite a la partícula saber qué decisión tomar dependiendo de su propia experiencia, es decir, esto representa la distancia entre la posición actual y la mejor conocida por esa partícula. Dicho componente cognitivo es el factor: $\varphi_1 \cdot rand_1 \cdot (mejorpos_i - x_i^{t-1})$ de la ecuación $(A.5)$.

Por otra parte, la partícula también tomará una decisión de acuerdo a la influencia del resto de partículas que componen el enjambre, es decir evaluará la distancia entre la posición actual y la mejor posición encontrada por el vecindario. Este componente social viene dado por el factor $\varphi_2 \cdot rand_2 \cdot (mejorpos - x_i^{t-1})$ de la ecuación $(A.5)$.

En la siguiente figura se puede apreciar el movimiento de una partícula en el espacio de búsqueda, donde las flechas verdes discontinuas representan la dirección de los componentes cognitivo y social. La flecha azul discontinua representa la velocidad actual de la partícula para moverse de una posición a otra.

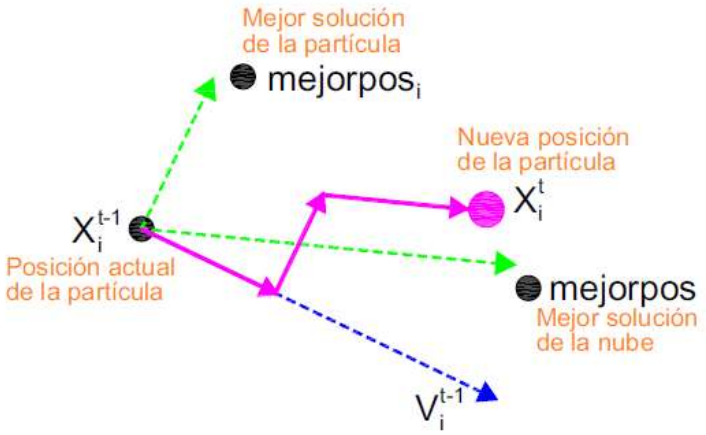

Figura A. 5. Representación gráfica del movimiento de una partícula. Tomada de [6]

Es de gran importancia determinar el tamaño adecuado de la nube, puesto que determina el equilibrio entre la calidad de las soluciones obtenidas y el número de iteraciones necesarias hasta llegar a una buena solución (tiempo computacional).

ANEXO B: MODELO FUNCIONAL DE RED

B.1 Enrutamiento de paquetes (Routing)

Este módulo posee la capacidad de realizar el enrutamiento y enviar las ráfagas hacia los nodos directamente conectados. Debido a que en el presente trabajo de grado no se requiere de ningún algoritmo específico de enrutamiento, este proceso se realiza encontrando siempre la ruta más corta entre el origen y el destino de la ráfaga enviada. En la figura xx se observa el módulo correspondiente a esta unidad. Su funcionamiento se describe a continuación:

1. Se debe saber el momento de llegada de una ráfaga. De esta manera, la unidad de enrutamiento asigna recursos al proceso de selección de ruta para dicha ráfaga.
2. Se obtienen las posibles rutas que se tienen para la ráfaga por medio de la función *cTopology* de OMNeT++.
3. Posterior a esto se envía la ráfaga por la interfaz adecuada hacia el siguiente nodo o hacia el app si es su destino final.
4. Se finaliza el proceso.

Figura B. 1. Modulo de enrutamiento de la red

B.2 Contención

Este módulo define en qué momento se pierde una ráfaga debido a contención. Para este caso se observa si más de una ráfaga llega al mismo tiempo a una misma interfaz, de ser así, alguna de las ráfagas se descarta.

Es de anotar que el número de ráfagas perdidas por contención se empieza a contar desde el tiempo de simulación 0.5 segundos. Esto debido a que antes de este tiempo se espera a que la red es “congestione” de ráfagas para observar el comportamiento de la misma de una manera más cercana a la realidad.

Por lo anterior, por contención solo se tendrá un tráfico de ráfagas debido a que los paquetes en ningún instante de tiempo pasan por dicho módulo. El módulo que correspondiente a esta unidad se muestra en la figura xx a continuación.

Figura B. 2. Modulo de contención de la red.

B.3 Dropp

Por medio de este módulo se define si el tiempo de offset aún no ha terminado. De ser así, la ráfaga, al igual que en el módulo de contención, se descarta. De la misma manera que se tiene para el módulo de contención, solo habrá tráfico de ráfagas y las ráfagas descartadas o perdidas empiezan a ser contadas desde el tiempo de simulación 0.5 segundos donde posterior a esto cada 10 milisegundos se cuenta el número de ráfagas perdidas para posteriormente realizar los diferentes cálculos necesarios para la obtención de uno de los parámetros del desempeño de la red.

Figura B. 3. Modulo perdida ráfagas por tiempo offset.

B.4 Clasific

Este módulo es el encargado de clasificar los paquetes provenientes del app en las diferentes direcciones de destino que tienen. De esta manera, se tendrán 11 clasificaciones diferentes puesto que la red EON cuenta con un total de 11 nodos. Cabe anotar que a pesar de que se cuenta con diferentes tipos de servicio, esta clasificación se realiza desde el módulo app, donde dependiendo del servicio, el app enviará de inmediato o con un retardo el paquete correspondiente.

Figura B. 4. Modulo clasificador de la red.

B.5 Ensamble

Ensambla los paquetes de acuerdo a la clasificación previamente hecha en el módulo clasific. En este punto, se tienen dos criterios para ensamblar la ráfaga, por número de paquetes o hasta completar un tamaño máximo en bytes de la misma, lo que primero ocurra. Hay que aclarar que este número de paquetes a ensamblar

o tamaño máximo de ráfaga viene dado por la metaheurística híbrida implementada en el presente trabajo de grado. De esta manera, de acuerdo al comportamiento de la red, las dos variables que se tienen de ensamble, están en constante cambio.

Figura B. 5. Modulo de ensamble de ráfagas de la red.

B.6 L2Queue

Realiza la reserva de recursos de acuerdo al protocolo de reserva de recursos configurado, es decir, el protocolo JET y al algoritmo de planificación utilizado en el presente trabajo de grado, el cual fue el de Horizonte de planificación - LAUC. De esta manera, si alguna interfaz de la red está ocupada, este módulo tiene la capacidad de detectar ese suceso y por tanto no asignarle recursos a la misma.

Figura B. 6. Modulo de reserva de recursos de la red.

B.6.1 Señales

Busy: enviada cuando una de las interfaces ópticas está ocupada.

Drop: enviada en caso de alguna pérdida de ráfaga.

txBytes: enviada para determinar el tamaño en bytes de una ráfaga enviada.

rxBytes: enviada para informar el tamaño en bytes de una ráfaga recibida en cada nodo**.**

B.7 Control

Este módulo es de gran importancia ya que por medio de este se hace la recolección de todos los datos de la red, las ráfagas generadas y perdidas y los retardos extremo a extremo que sufren las ráfagas cada 10 milisegundos como se mencionó anteriormente en este documento y en la monografía.

Posterior a obtener todos estos resultados, se procede a hacer el cálculo de la probabilidad de bloqueo, retardos y tráfico cursado sobre la red cada determinado tiempo periódicamente, para finalmente enviarlos a los módulos de las metaheurísticas PSO y Redes neuronales, los cuales deciden de acuerdo a ciertos parámetros, el número de paquetes a ensamblar en una ráfaga y su tamaño máximo ideal para tratar de disminuir los parámetros establecidos para analizar el desempeño de la red.

Figura B. 7. Modulo de control de la red.

B.8 Archivo de configuración .ini

En este archivo se configuran todas las variables predefinidas de la red. Es de aclarar que de acuerdo a los requerimientos de cada red, las variables necesarias en el archivo .ini van a cambiar. Para el presente trabajo de grado se tienen las siguientes variables configuradas:

- ***Sim-time-limit:*** Define el tiempo máximo de simulación. Para el trabajo de grado se simuló 1.5 segundos.

- ***SendIaTime:*** Define cada cuánto se va a generar un paquete para enviarlo a la red. Se configuraron 3 distribuciones para modelar 3 tipos de tráfico, distribución exponencial para tráfico alto, Normal para tráfico medio y uniforme para tráfico bajo.

- ***App.packetLength:*** Configura la longitud que tendrá cada paquete. Para las diferentes simulaciones se utilizó la distribución uniforme, para un tamaño entre 128 y 1024 bytes.

- ***Timeout:*** máximo tiempo que puede una ráfaga cursar la red. Si sobrepasa este tiempo, la ráfaga se descarta. El tiempo configurado fue de 0.01 segundos.

- ***numPackets:*** Número de paquetes máximo a ensamblar en la ráfaga.

- ***minOffset:*** Tiempo mínimo de offset. Se configuró de 0,0001 segundos.

- ***maxOffset:*** Tiempo máximo de offset. Se configuró de 0,0009 segundos.

- ***destAdresses:*** define cuantas direcciones habrán sobre la red. Para el trabajo de grado se configuró como un vector de 11 posiciones, cada una con un valor diferente para diferenciar un nodo de otro.

- ***overflowLastPacket:*** Determina la condición para ensamblar por número de paquetes o por tamaño de la ráfaga. Tiene un valor por defecto de *false.*

- ***"Direcciones de nodos":*** A cada nodo se lo nombró de acuerdo a la ciudad donde está ubicado y de igual manera se le asignó una posición del vector desadresses para diferenciar una dirección de un nodo de otra.

ANEXO C: ENTRENAMIENTO DE LA RED NEURONAL

C.1 Entrenamiento de la Red Neuronal con Trafico Medio

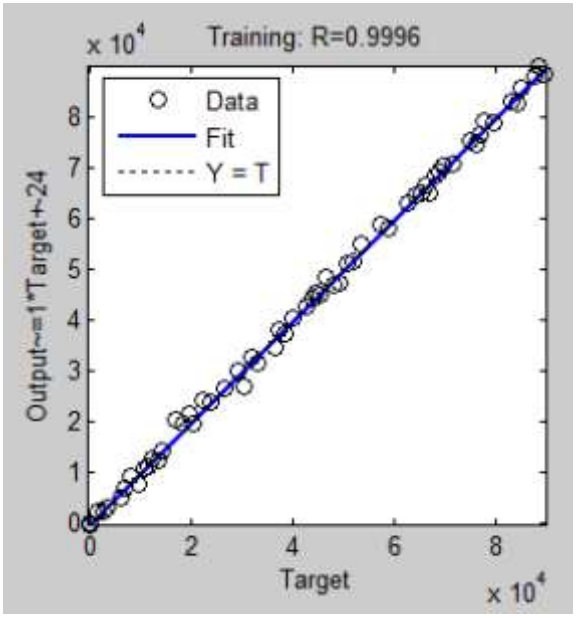

Figura C. 1. Entrenamiento de la red OBS/DWDM con el algoritmo HMCNA para trafico medio.

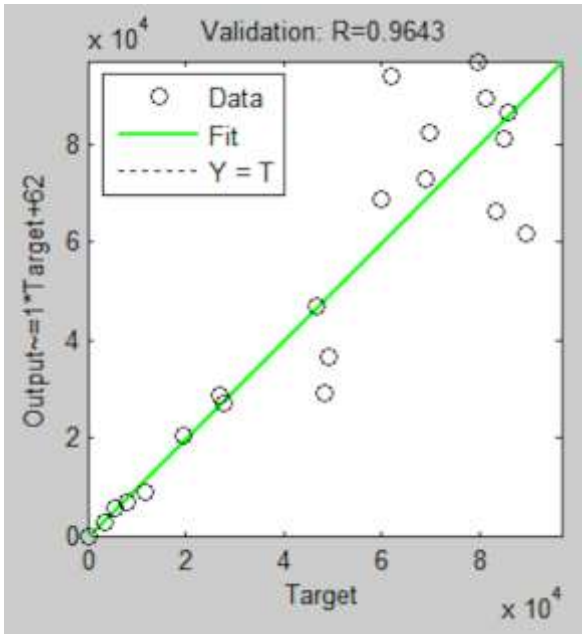

Figura C. 2. Validación de la red OBS/DWDM con el algoritmo HMCNA para trafico medio.

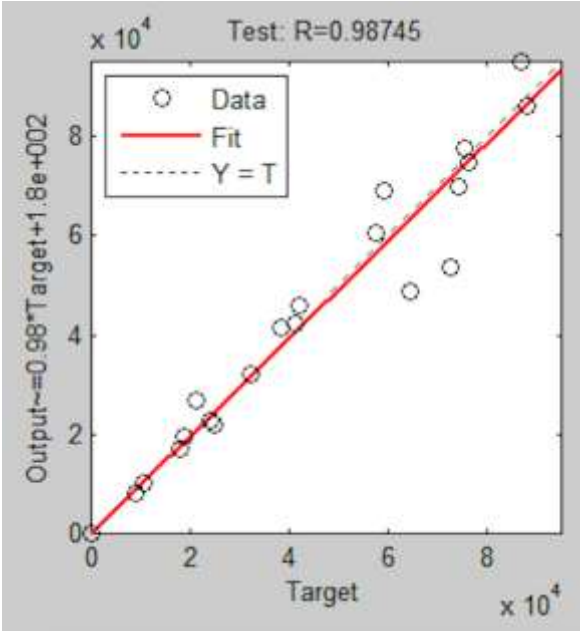

Figura C. 3. Prueba del entrenamiento de la red OBS/DWDM con el algoritmo HMCNA para trafico medio.

Al igual que los resultados obtenidos para el entrenamiento, validación y prueba de la Red Neuronal para un trafico alto, dado el caso de un trafico medio el entrenamiento de la red tal y como lo muestra las *figuras 13, 14 y 15* es satisfactorio debido a los coeficientes de correlación están dentro de un margen cercanos a uno, por lo que se puede deducir que la red ejecutó los procesos anteriormente mencionados de manera correcta.

C.2 Entrenamiento de la Red Neuronal con Trafico Bajo

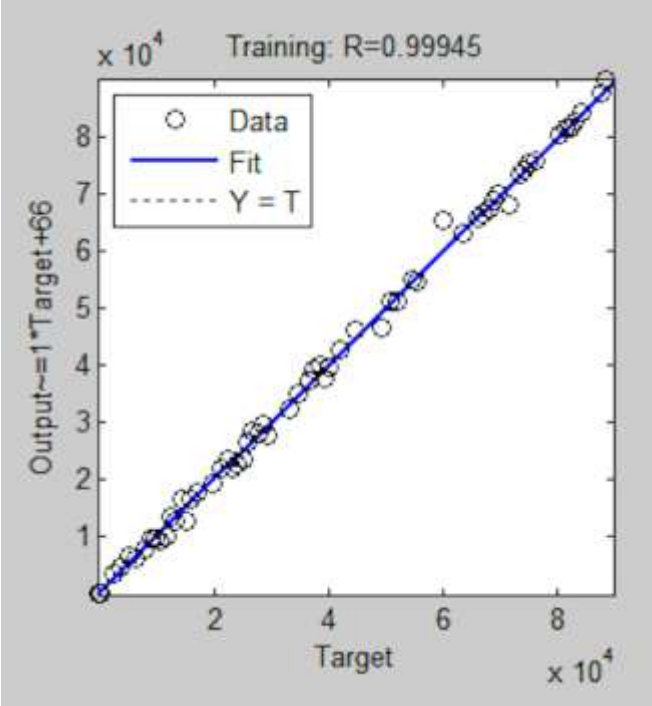

Figura C. 4. Entrenamiento de la red OBS/DWDM con el algoritmo HMCNA para trafico bajo.

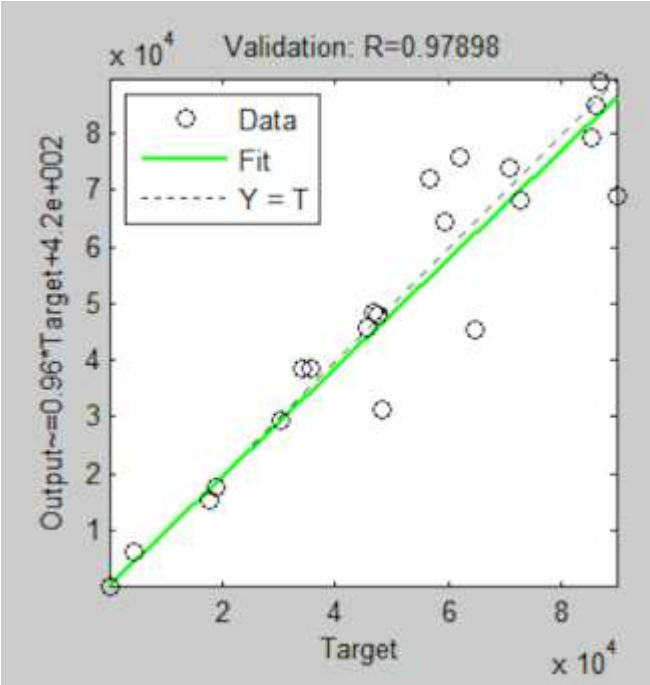

Figura C. 5. Validación de la red OBS/DWDM con el algoritmo HMCNA para trafico bajo.

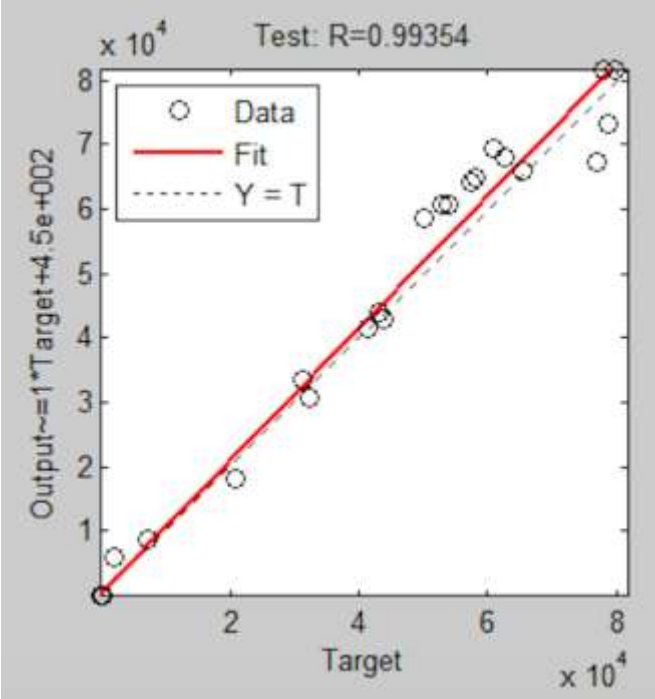

Figura C. 6. *Prueba del entrenamiento de la red OBS/DWDM con el algoritmo HMCNA para trafico bajo.*

Los resultados obtenidos para el entrenamiento de la Red Neuronal con trafico bajo se pueden observar en la *figura 16,17 y 18*, donde los coeficientes de correlación indican que la red se entrenó correctamente y que el vector salida es el adecuado para realizar el proceso de ensamble de las ráfagas.

Se puede observar que los resultaos del entrenamiento de la Red Neuronal de acuerdo al trafico de entrada varían, puesto que el vector de entrada correspondiente a la probabilidad de bloqueo y retardo extremo a extremo varían de acuerdo al trafico introducido en la red, y por tanto los pesos sinápticos de la red neuronal deben adecuarse a dichas entradas.

ANEXO D: TABLAS DE REDULTADOS DE LAS SIMULACIONES

D.1 Probabilidad de Bloqueo para un tráfico alto

Para el trafico alto, en las Tablas D.1 y D.2 se muestra los datos obtenidos correspondientes a la probabilidad de bloqueo, modelado con la distribución exponencial de probabilidad

Tabla D. 1 Probabilidad de Bloqueo para un tráfico alto sin la implementación del algoritmo HMCNA

Tráfico	Probabilidad de Bloqueo
2,8	0,2968
7,55	0,3551
10,35	0,41596
14,3	0,47857
16,6	0,5199
23,15	0,63271
34,75	0,7442

Tabla D. 2 Probabilidad de Bloqueo para un trafico alto con la implementación del algoritmo HMCNA

Tráfico	Probabilidad de Bloqueo
3	0,2
7,83889	0,20241
10,22333	0,23297
11,58889	0,24467
13,33929	0,27561
15,6125	0,31204
18,5125	0,3456
22	0,38016
26,575	0,42463
34,86429	0,46947

D.2 Probabilidad de Bloqueo para un tráfico medio

Para el trafico alto, en las Tablas D.3 y D.3 se muestra los datos obtenidos correspondientes a la probabilidad de bloqueo, modelado con la distribución normal de probabilidad

Tabla D. 3 Probabilidad de Bloqueo para un tráfico medio sin la implementación del algoritmo HMCNA

Tráfico	Probabilidad de Bloqueo
3,47727	0,275
6,6125	0,44494
10,9	0,51167
23,66	0,67984
29,934	0,68024

Tabla D. 4 Probabilidad de Bloqueo para un trafico medio con la implementación del algoritmo HMCNA

Tráfico	Probabilidad de Bloqueo
4,47727	0,14561
6,07857	0,14751
9,0125	0,20691
11,43	0,23971
13,595	0,27455
15,675	0,29999
18,50909	0,37858
22,84375	0,43925
24,783	0,4418
29,934	0,4486

D.3 Probabilidad de Bloqueo para un tráfico bajo

Para el trafico alto, en las Tablas D.5 y D.6 se muestra los datos obtenidos correspondientes a la probabilidad de bloqueo, modelado con la distribución exponencial de probabilidad

Tabla D. 5 Probabilidad de Bloqueo para un tráfico bajo sin la implementación del algoritmo HMCNA

Tráfico	Probabilidad de Bloqueo
5,8394	0,12273
10,94667	0,15223
15,10882	0,19002
17,53125	0,23644
18,79333	0,26355
27,37926	0,29342

Tabla D. 6 Probabilidad de Bloqueo para un trafico bajo con la implementación del algoritmo HMCNA

Tráfico	Probabilidad de Bloqueo
3,87692	0,23717
6,1	0,29196
7,55357	0,3458
10,43333	0,40913
13,09545	0,4214
21,725	0,62004
26,473	0,62005

Referencias

[1] S. Aaronson, “NP-complete Problems and Physical Reality,” *ACM Sigact News*, vol. 36, no. 1, pp. 30–52, 2005.

[2] J. McClelland, D. Rumelhart, and G. Hinton, “The Appeal of Parallel Distributed Processing,” *Parallel Distrib. Process. Explor. Microstruct. Cogn. Vol. 1 Found.*, pp. 3–44, 1986.

[3] J. L. Mcclelland, D. E. Rumelhart, and P. J. Hayes, *Explorations in Parallel Distributed Processing*. Cambridge Massachusetts, 1986.

[4] B. Martin del Brío and A. Sanz Molina, *Redes Neuronales y Sistemas Borrosos*, 3ª Edición. México D.F, 2007.

[5] J. Kennedy and R. C. Eberhart, *Swarm intelligence*, Morgan Kau., vol. 5, no. 11. Indiana University, 2010.

[6] M. Gomez González, “Sistema de generación eléctrica con pila de combustible de óxido sólido alimentado con residuos forestales y su optimización mediante algoritmos basados en nubes de partículas,” Universidad Nacional de Educación a distancia, 2008.

[7] J. Kennedy and R. C. Eberhart, “A discrete binary version of the particle swarm algorithm,” *1997 IEEE Int. Conf. Syst. Man, Cybern. Comput. Cybern. Simul.*, vol. 5, pp. 4–8, 1997.

[8] J. Kennedy and R. Eberhart, “Particle swarm optimization,” *Neural Networks, 1995. Proceedings., IEEE Int. Conf.*, vol. 4, pp. 1942–1948 vol.4, 1995.