REPORTE FINAL

Red neuronal Hopfield aplicada al problema del agente viajero

M. en C. José Alejandro Cornejo Acosta
Instituto Nacional de Astrofísica, Óptica y Electrónica
Coordinación de Ciencias Computacionales
alexcornejo@inaoep.mx

Mayo 2020

Resumen

Este documento corresponde al reporte final de la materia Inteligencia Computacional I - Parte II. Este trabajo se centra en la exploración y experimentación de un algoritmo de cómputo suave llamado Red Neuronal Hopfield. El problema abordado es un problema de optimización combinatoria llamado “El problema del agente viajero". La experimentación se realiza con instancias pequeñas generadas aleatoriamente y los resultados son comparados con un algoritmo heurístico llamado Algoritmo del vecino más cercano.
Adicionalmente, con el objetivo de experimentar con diferentes parámetros para la red neuronal Hopfield, se presenta un software desarrollado durante la realización de este trabajo el cual fue utilizado para realizar los experimentos presentados. Finalmente, se analizan y discuten los resultados, y se contrastan las diferencias entre de cada uno de los algoritmos mencionados.

Palabras clave: TSP, Red-Neuronal-Hopfield, vecino-más-cercano.

Instituto Nacional de Astrofísica, Óptica y Electrónica. Coordinación de Ciencias Computacionales. Luis Enrique Erro 1, Sta María Tonanzintla, 72840 San Andrés Cholula, Puebla, México.
Asignatura: Inteligencia Computacional I - Parte II
Catedrática: Dra. María del Pilar Gómez Gil (`https://ccc.inaoep.mx/~pgomez/`)

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Dado un conjunto de ciudades conectadas por carreteras, y cuya distancia entre cada par de ellas es conocida, ¿Cuál es la ruta más corta para visitar todas ellas exactamente una vez, iniciando y terminando el recorrido en la misma ciudad?. En resumen, este es el planteamiento del problema del agente viajero conocido como TSP por sus siglas en inglés (Traveling Salesman Problem). A pesar de que el planteamiento es simple, detrás del problema han habido grandes retos que incluso hasta hoy varios de ellos no han sido resueltos. Este problema fue inicialmente formulado por Hamilton en los años de 1800, y es uno de los problemas de optimización combinatoria más estudiados [Cook, 2012]. El TSP puede modelarse mediante un grafo ponderado completamente conectado $G = (V, E)$, en donde las ciudades pueden ser representadas por los *nodos* (conjunto V) y las carretas que conectan a las ciudades pueden ser representadas por las *aristas* (conjunto E). Utilizando términos formales, el problema consiste en encontrar el ciclo hamiltoniano de menor costo del grafo G. La Figura 1.1 muestra visualmente el objetivo del problema TSP asumiendo que la imagen de la izquiera es un grafo ponderado completamente conectado.

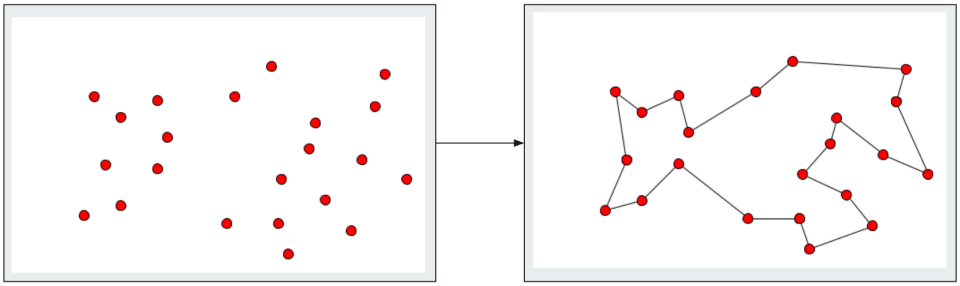


Figura 1.1: Ejemplo de TSP.

El TSP tiene aplicaciones en diferentes áreas, por ejemplo, en los problemas de transporte y ruteo, en los problemas de impresión de circuitos electrónicos, en problemas de programación de máquinas, entre otros [Punnen, 2007]. Respecto a su complejidad, el TSP corresponde a la categoría de problemas NP-Difícil, lo que significa que no puede resolverse en un tiempo de ejecución polinomial por una máquina de Turing determinista. Debido a esto, investigadores han optado por desarrollar técnicas y algoritmos que puedan aproximar soluciones a este problema en tiempos de ejecución factibles, como los algoritmos de cómputo suave. El cómputo suave es la rama que involucra a un conjunto de algoritmos y métodos que trabajan bajo cierta tolerancia a la incertidumbre y a la imprecisión [Chaturvedi, 2008]. Lógica difusa, redes neuronales, algoritmos genéticos y heurísticos son algunas de las áreas de cómputo suave.

En este trabajo se muestran dos algoritmos de cómputo suave para abordar el problema del agente viajero. Los cuales son una red neuronal y un algoritmo heurístico voraz. Se hace especial énfasis en el algoritmo de red neuronal debido a que el propósito de este documento es el de explorar dicho algoritmo. Adicionalmente, se menciona un algoritmo exacto que no

se describe de forma detallada puesto que solo se utiliza para obtener los óptimos de las instancias utilizadas y tener un mejor punto de referencia al comparar los dos algoritmos de cómputo suave.
Este trabajo es únicamente de carácter exploratorio en técnicas de cómputo suave para el problema del agente viajero, por lo que no hay un aporte al estado del arte como tal, tampoco se utilizan las últimas técnicas del estado del arte. El propósito es únicamente el estudio y compresión de dichas técnicas, y poder contrastar las principales diferencias entre ellas.

2. OBJETIVOS

Realizar un estudio comparativo entre dos técnicas clásicas de cómputo suave aplicadas al problema del agente viajero.

- Implementar un algoritmo heurístico para el problema de agente viajero.
- Implementar una red neuronal Hopfield para el problema del agente viajero.
- Utilizando instancias pequeñas, comparar ambos algoritmos mediante diferentes criterios, calidad de soluciones encontradas, facilidad de implementación, etc.
- Realizar la prueba estadística de Wilcoxon para contrastar si un algoritmo es significativamente mejor que el otro.

3. METODOLOGÍA

En esta sección se describen los algoritmos y el procedimiento en general que se realizó durante este trabajo. Es importante aclarar que el modelo de red neuronal Hopfield implementado en este trabajo, es el propuesto en [Hopfield and Tank, 1985]. El código del proyecto se realizó en el lenguaje de programación Java. El repositorio está publicado en `Github` y puede ser consultado en `https://github.com/alex-cornejo/HopfieldTSP`. Aquí se encontrará también una descripción de como utilizar el software.

3.1. RED NEURONAL HOPFIELD

Las redes neuronales han sido empleadas para diferentes propósitos como tareas de clasificación y reconocimiento. Algunas de ellas han sido utilizadas para abordar problemas de optimización, tal es el caso de la red neuronal Hopfield la cual fue desarrollada por John Hopfield en 1982. Esta red pertenece a la clase de redes neuronales de aprendizaje no supervisado. La estructura de esta red neuronal consiste en un conjunto de neuronas que están conectadas entre sí, formando un grafo completamente conectado. De esta forma, la activación de una neurona influye en todas las demás, por lo que también esta red neuronal es considerada una red recurrente.
La red Hopfield, ha sido utilizada para abordar el problema del agente viajero, tal es del caso de [Hopfield and Tank, 1985] [Mańdziuk, 1996][Talaván, 2002] en donde se muestra como

utilizar este modelo puede brindar resultados relativamente buenos en comparación a utilizar una búsqueda computacional exhaustiva. El modelo de red neuronal Hopfield utilizado en este trabajo, es el propuesto en [Hopfield and Tank, 1985]. La idea principal, es que la red neuronal Hopfield tenga una cantidad de neuronas igual a n, en donde n es cantidad de ciudades que el agente viajero tiene que visitar. Además, las neuronas están completamente conectadas unas con otras. Antes de entrar en profundidad sobre como funciona dicho modelo, primero es necesario introducir algunos conceptos que se emplearán más adelante.

Definición 1. *El estado de una red Hopfield es el conjunto de activaciones de todas las neuronas de la red.*

Definición 2. *Una red Hopfield converge a un estado estable cuando las activaciones las neuronas de la red ya no sufren cambios de una iteración a otra.*

Definición 3. *La función de energía de una red Hopfield es aquella que mapea de un estado de la red, a un número real.*

Utilizando el concepto de función de energía de una red Hopfield, es posible utilizar esta red para resolver problemas de optimización. La idea principal consiste en modelar la función objetivo del problema de optimización que deseamos abordar. Esta función objetivo, debe ser transformada en una función de energía. De esta forma, al hacer converger la red Hopfield a un estado estable, la función de energía será minimizada y por lo tanto, la función objetivo del problema de optimización también se minimizará. Sin embargo, modelar una función de objetivo como una función de energía muchas veces no es suficiente y no es una tarea trivial, ya que muchos problemas de optimización tienen ciertas restricciones. En el caso de TSP, el objetivo es encontrar una ruta con el menor costo posible para un agente viajero, en donde las restricciones son que la ruta visite n ciudades, y que lo haga únicamente una vez. Es posible representar dicha ruta mediante una matriz binaria. Por ejemplo, si tenemos cuatro ciudades c_1, c_2, c_3, c_4, una ruta puede representarse con la siguiente matriz en donde los renglones representan las ciudades y las columnas representan las posiciones de las ciudades en la ruta.

$$\begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Esta matriz representa una ruta en el orden c_3, c_1, c_4, c_2. Para que esta matriz represente una ruta válida que cumpla las restricciones del problema del agente viajero, dicha matriz debe respetar ciertas condiciones. Estas restricciones deben ser consideradas al momento de modelar la función de energía. La Ecuación 3.1 es la función de energía empleada para TSP que cumple con las restricciones del problema.

$$E(V) = \frac{A}{2}\sum_{x}^{n}\sum_{i}^{n}\sum_{j\neq i}^{n} V_{x,i}V_{x,j} + \frac{B}{2}\sum_{i}^{n}\sum_{x}^{n}\sum_{y\neq x}^{n} V_{x,i}V_{y,j} + \frac{C}{2}\left(\sum_{x}^{n}\sum_{i}^{n} V_{x,i} - n\right)^2 + \frac{D}{2}\sum_{x}^{n}\sum_{y\neq x}^{n}\sum_{i}^{n} d_{x,y}V_{x,i}(V_{y,i+1} + V_{y,i-1}) \tag{3.1}$$

El primer término de la ecuación es 0 únicamente cuando hay un valor de 1 en cada renglón de la matriz. De lo contrario el término tendrá un valor mayor a 0. El segundo termino es 0 únicamente cuando hay un 1 en cada columna de la matriz. El tercer término es 0 únicamente cuando hay n 1s en toda la matriz. El último término representa la longitud de la ruta, la cual deseamos sea minimizada. Es importante especificar que los valores V_i del último término están definidos en términos del módulo n, es decir $V_{i+n} = V_i$.
Los valores de $V_{i,j}$ se refieren a las activaciones de las neuronas. En donde la función de activación se define en la Ecuación 3.2

$$V_{x,i} = g(u_{x,i}) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_{x,i}}{u_0}\right)\right) \tag{3.2}$$

En donde $u_0 > 0$ y $u_{x,i}$ es la entrada de la neurona. Éste último es un valor que va cambiando durante las iteraciones de la red conforme a la Ecuación 3.3.

$$u_{x,i} = u_{x,i} + \left(\frac{\delta u_{x,i}}{\delta t}\right)\Delta t \tag{3.3}$$

$$\frac{\delta u_{x,i}}{\delta t} = -\frac{u_{x,i}}{\tau} - A\sum_{j\neq i}^{n} V_{x,j} - B\sum_{y\neq x}^{n} V_{y,i} - C\left(\sum_{x}^{n}\sum_{j}^{n} V_{x,j-n'}\right) - D\sum_{y}^{n} d_{x,y}(V_{y,i+1} + V_{y,i-1}) \tag{3.4}$$

En la Ecuación 3.4, $d_{x,y}$ representa la distancia de la ciudad x ala ciudad y. Según [Hopfield and Tank, 1985], el parámetro n' puede tomar un valor igual a n, pero se recomienda que sea un valor diferente de tal forma que represente un parámetro de ajuste para la ecuación. Los valores A, B, C y D son constantes. Desafortunadamente, la calidad de la soluciones obtenidas por la red dependen de estos parámetros, los cuales son difíciles de establecer. En la siguiente sección, se detallarán los valores específicos utilizados en la experimentación de este trabajo.

3.2. Algoritmo del vecino más cercano

Es un algoritmo heurístico voraz para construir soluciones al problema del agente viajero en una complejidad cuadrática [Kizilates and Nuriyeva, 2013]. El planteamiento es simple y es el siguiente:

1. Elegir una ciudad aleatoria para iniciar la ruta.
2. Moverse a la ciudad más cercana que no haya sido visitada.

3. Si aún hay ciudades que no han sido visitadas, volver al paso 2.
4. Regresar a la primer ciudad de la ruta.

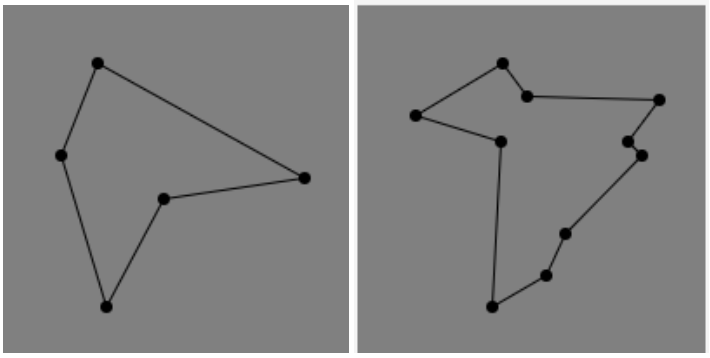
Al ser un algoritmo heurístico, este no garantiza el encontrar las soluciones óptimas al problema.

3.3. ALGORITMO HELD-KARP

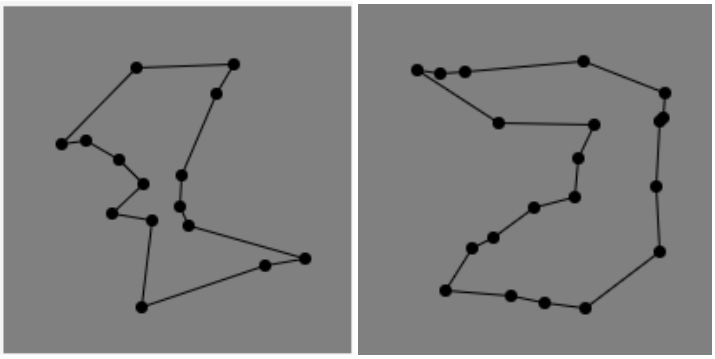
El algoritmo Held-Karp es un algoritmo de programación dinámica propuesto en 1962 por Held y Karp [Held and Karp, 1962] para resolver el problema del agente viajero de manera exacta. Este algoritmo tiene una complejidad de $O(2^n n^2)$ y es importante mencionar que aunque es mucho más eficiente que una búsqueda exhaustiva, al ser de complejidad exponencial, solo permite resolver instancias únicamente con unas cuantas decenas de nodos. La implementación de este algoritmo fue tomada de la librería de teoría de grafos para Java `JGrapht` `https://jgrapht.org/`.

4. EXPERIMENTACIÓN Y RESULTADOS

En esta sección se presenta una comparación experimental entre la red neuronal Hopfield con el algoritmo del vecino más cercano descrito previamente. Adicionalmente, se incluye también el método de generación de soluciones aleatorias que consiste en generar una solución de TSP mediante una permutación aleatoria. Para la experimentación se generaron aleatoriamente instancias de 5, 10, 15 y 20 nodos. Todas ellas fueron generadas en un espacio euclídeo con coordenadas en un rango [0, 1] con una distribución uniforme. La Figura 4.1 muestra las instancias mencionadas con las rutas óptimas obtenidas con el algoritmo Held-Karp.



(a) n = 5. Opt = 2.265 (b) n = 10. Opt = 2.702



(c) n = 15. Opt = 2.967 (d) n = 20. Opt = 3.792

Figura 4.1: Instancias de prueba generadas aleatoriamente.

Con el propósito de poder experimentar diferentes combinaciones de parámetros para la red neuronal Hopfield y poder visualizar el comportamiento en las soluciones generadas, se desarrolló un software que se encuentra en el repositorio de `Github` mencionado anteriormente. En este repositorio se encuentran las instrucciones para utilizarlo. La Figura 4.2 es una captura de pantalla que muestra gráficamente tres soluciones para la instancia de 20 nodos utilizada en la experimentación. La imagen muestra una de las soluciones de la red neuronal Hopfield, otra solución obtenida con el algoritmo del vecino más cercano, y una tercera solución que fue generada aleatoriamente. En las etiquetas `Length` se puede observar que la solución de la red Hopfield fue mejor que las encontradas por los otros métodos.

La configuración de parámetros para la red neuronal Hopfield se describe en la Tabla 4.1. La mayor parte de esta configuración fue tomada de [Hopfield and Tank, 1985]. El parámetro de ajuste n' es un parámetro dinámico y fue tomado como $n+1$, donde n es la cantidad de nodos de la instancia. Toma este valor porque en otro artículo donde se trabaja este mismo tema [Mańdziuk, 1996], el autor reporta que dicha configuración presentó buenos resultados.

Durante la experimentación, se observó que la red Hopfield no siempre converge a soluciones válidas, de hecho,la mayoría de las veces no lo hace. Por lo que para hacer un análisis de resultados, el algoritmo de la red Hopfield se ejecutó hasta obtener 10 soluciones válidas para cada instancia. Para hacer una comparación, también se obtuvieron 10 soluciones con

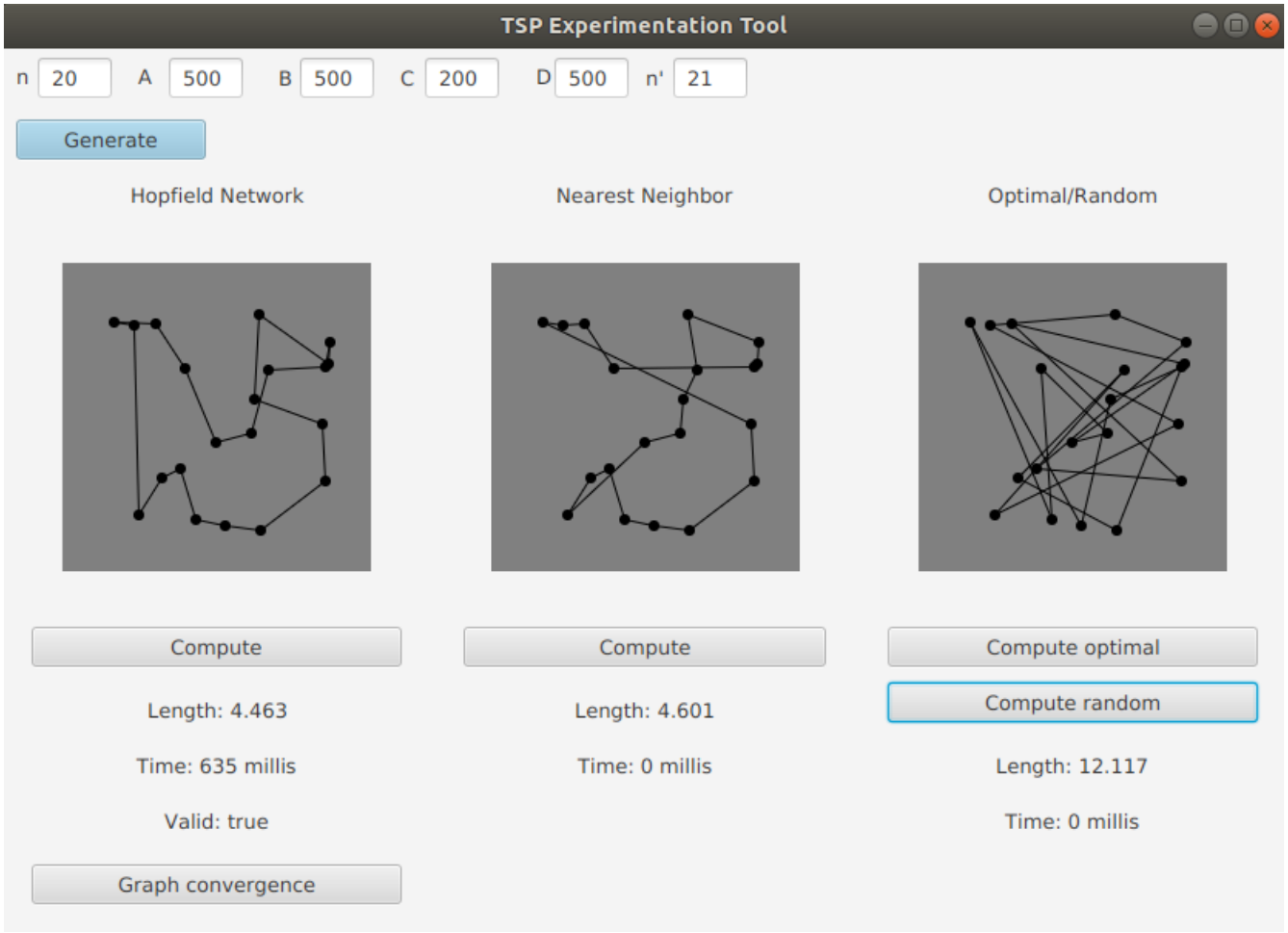


Figura 4.2: Software para experimentar con los algoritmos descritos en este documento.

Parámetro	**A**	**B**	**C**	**D**	**n'**	Δt	u_0	τ
Valor	500	500	200	500	$n+1$	1e-4	0.02	1

Tabla 4.1: Configuración de parámetros de red neuronal Hopfield.

Instancia	Opt	Hopfield			Vecino más cercano			Aleatorio		
		mejor	μ	σ	*mejor*	μ	σ	*mejor*	μ	σ
5	2.265	2.265	2.441	0.115	2.378	2.443	0.071	2.482	2.687	0.149
10	2.702	2.702	2.939	0.165	2.953	3.128	0.189	4.419	5.098	0.444
15	2.967	3.465	3.801	0.257	3.200	3.321	0.156	5.280	5.912	0.515
20	3.792	4.366	5.247	0.406	3.908	4.266	0.299	8.110	10.496	1.149

Tabla 4.2: Comparación sobre los diferentes algoritmos.

Instancia	Hopfield	Vecino más cercano	Aleatorio
5	1.077	1.079	1.186
10	1.088	1.158	1.887
15	1.281	1.119	1.993
20	1.384	1.125	2.768

Tabla 4.3: Factor de aproximación empírico de los algoritmos.

el algoritmo del vecino más cercano y 10 soluciones generadas aleatoriamente. En la Tabla 4.2 se reportan la mejor solución encontrada, el promedio de la 10 soluciones, y la desviación estándar. Además, en la columna **Opt** se reporte el óptimo de cada instancia obtenido con el algoritmo Held-Karp.

En la Tabla 4.2 se puede observar que en las instancias de 5 y 10 nodos, la red neuronal Hopfield fue la que obtuvo mejores resultados. Obtuvo una menor desviación estándar y un mejor promedio respecto a estas dos instancias. Incluso logró encontrar el valor óptimo para ambas instancias. En las últimas dos instancias de 15 y 20 nodos, fue el algoritmo del vecino más cercano el que obtuvo mejores resultados. Aunque en éstas últimas dos instancias ninguno de los algoritmos fue capaz de encontrar algún óptimo. El método de generación de soluciones aleatorias fue el que obtuvo los peores resultados en todas las instancias.

Utilizando el promedio de los resultados presentados en la Tabla 4.2, en la Tabla 4.3 se presenta el factor de aproximación empírico de cada uno de los algoritmos respecto a cada instancia. En esta tabla se puede observar que para las instancias de 5 y 10 nodos, la red Hopfield fue la que tuvo un mejor factor de aproximación, ya que las soluciones encontradas fueron en promedio solo 7.7 % lejanas del óptimo y 8.8 % respectivamente. En el caso de las instancias de 15 y 20 nodos, fue el algoritmo del vecino más cercano el que obtuvo un mejor factor de aproximación, ya que las soluciones encontradas en promedio fueron solo 11 % y 12 % lejanas al óptimo.

Para analizar los datos de forma más detallada, utilizando las 10 soluciones generadas por los algoritmos, se realizó la prueba de pares de Wilcoxon. Esta prueba consiste en hacer una prueba de hipótesis entre dos muestras en donde la hipótesis nula H_0 plantea que ambas muestras vienen de distribuciones con la misma mediana. La hipótesis alterna H_1 plantea que las muestras no vienen de distribuciones con la misma mediana. El valor de significancia utilizado fue $\alpha = 0{,}05$. La Tabla 4.4 muestra los resultados de la prueba de Wilcoxon entre los

diferentes métodos en donde los acrónimos son los siguientes:
H: Red Hopfield.
NN: Algoritmo del vecino más cercano.
R: Generación aleatoria.
Un valor de 1 indica que hay suficiente evidencia estadística para rechazar H_0 mientras que un valor 0 indica que H_0 es aceptada.

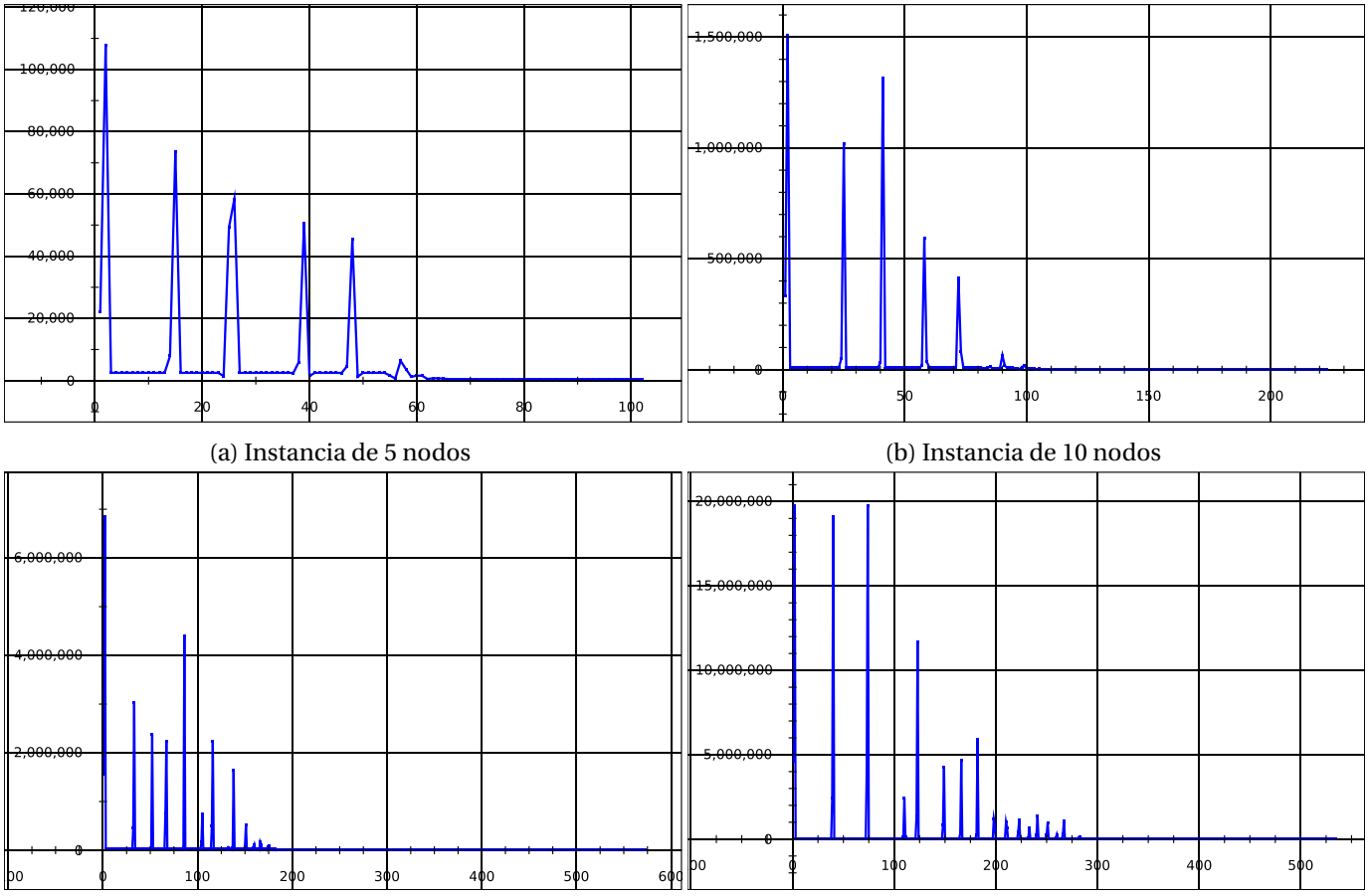
Instancia	**H:NN**	**H:R**	**NN:R**
5	0	1	1
10	0	1	1
15	1	1	1
20	1	1	1

Tabla 4.4: Resultados de la prueba de Wilcoxon.

Como se mencionó anteriormente, aparentemente la red Hopfield obtuvo mejores resultados en las Tablas 4.2 y 4.3 para las instancias de 5 y 10 nodos en comparación con el algoritmo del vecino más cercano. Sin embargo, utilizando la prueba de Wilcoxon, podemos concluir que no hay evidencia estadística para afirmar que dicha mejora es significativa.

Por otro lado, al comparar la red Hopfield con el algoritmo del vecino más cercano en las instancias de 15 y 20 nodos. Utilizando la prueba de Wilcoxon, se puede observar que en este caso la mejora del algoritmo del vecino más cercano si es estadísticamente significativa. Respecto al método de generación de soluciones aleatorias, con la prueba de Wilcoxon se verifica que tanto los resultados de la red Hopfield como los del algoritmo del vecino más cercano, son significativamente mejores que las soluciones aleatorias.

Como se mencionó anteriormente, para obtener una solución de TSP mediante la red de Hopfield, tiene que haber una convergencia de la función de energía. La Figura 4.3 muestra el comportamiento de la función de energía en ejecuciones arbitrarias de la red Hopfield para las instancias utilizadas. En esta figura se observa que una mayor cantidad de nodos implica que la red necesite más iteraciones en converger. También se observa que en todos los casos la convergencia de la red oscila y esta oscilación va disminuyendo hasta que logra estabilizarse. También es importante mencionar que la convergencia de la función de energía tendrá casi siempre un comportamiento similar, incluso cuando la red no encuentre un solución válida. Esto se debe a que la red puede converger hacia un mínimo local, encontrando una ruta que no cumpla con las restricciones necesarias.



(a) Instancia de 5 nodos

(b) Instancia de 10 nodos

(c) Instancia de 15 nodos

(d) Instancia de 20 nodos

Figura 4.3: Convergencia de instancias.

5. CONCLUSIONES Y COMENTARIOS FINALES

En este trabajo se realizó la exploración de la red neuronal Hopfield aplicada al problema del agente viajero. Utilizando instancias generadas aleatoriamente con *pocos* nodos, se observo que la red obtuvo mejores resultados en comparación con la generación de soluciones aleatorias. En comparación con el algoritmo del vecino más cercano, no podemos concluir que la red Hopfield es una mejor opción, ya que ésta solo obtuvo mejores resultados con las instancias de 5 y 10 nodos. Además de que esta mejor no es significativa según la prueba estadística realizada. Que la red Hopfield haya obtenido un buen desempeño con las instancias de 5 y 10 nodos tiene sentido, ya que los parámetros de la red Hopfield utilizados en este trabajo fueron tomados de [Hopfield and Tank, 1985], en donde fueron especialmente ajustados para instancias de 10 nodos. Por lo que utilizarlas para instancias con una mayor cantidad de nodos, podría representar una “desventaja". Una posible mejora de este trabajo sería realizar un ajuste de éstos parámetros dependiendo de la cantidad de nodos. Sin embargo, no

se encontró algún método propuesto que de forma sencilla permita realizar el ajuste de éstos parámetros obteniendo resultados favorables. En [Talaván, 2002] se presenta un método para ajustar estos parámetros logrando que la red converja casi el 100 % de las veces a soluciones válidas. Aunque este método fue probado de manera informal, las rutas encontradas parecían ser totalmente aleatorias. Por lo que en este trabajo se decidió optar por los mismos parámetros que [Hopfield and Tank, 1985]. Finalmente, otro criterio importante a considerar es la facilidad de implementación. La principal desventaja de utilizar la red Hopfield para problemas de optimización es el ajuste de los parámetros. En el caso del algoritmo del vecino más cercano, no se requiere ningún ajuste, al menos no en la versión presentada en este trabajo. Además de que implementar este algoritmo es relativamente fácil y requiere menos cantidad de líneas de código. Tomando en cuenta estos criterios, podemos decir que el algoritmo del vecino más cercano es la mejor opción, ya que obtuvo buenos resultados en cuanto a la calidad de las soluciones y además es sencillo de implementar. Sin embargo, no deja de ser interesante como las redes neuronales pueden ser empleadas en tareas muy diversas, y puede ser que en estas otras tareas las redes neuronales si sean la mejor opción. Aunque como lo dice el teorema de No Free Lunch, no es posible que un solo algoritmo obtenga los mejores resultados en todos los problemas.

REFERENCIAS

[Chaturvedi, 2008] Chaturvedi, D. K. (2008). *Soft Computing*. Springer-Verlag Berlin Heidelberg, 1 edition.

[Cook, 2012] Cook, W. J. (2012). *In Pursuit of the Traveling Salesman*. Princeton University Press, New Jersey, 1 edition.

[Held and Karp, 1962] Held, M. and Karp, R. M. (1962). A Dynamic Programming Approach to Sequencing Problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210.

[Hopfield and Tank, 1985] Hopfield, J. J. and Tank, D. W. (1985). "NeuralÇomputation of Decisions in Optimization Problems. *Biological Cybernetics*, 52.

[Kizilates and Nuriyeva, 2013] Kizilates, G. and Nuriyeva, F. (2013). On the Nearest Neighbor Algorithms for the Traveling Salesman Problem. In Nagamalai, D., Kumar, A., and Annamalai, A., editors, *Advances in Computational Science, Engineering and Information Technology*, pages 111–118, Heidelberg. Springer International Publishing.

[Mańdziuk, 1996] Mańdziuk, J. (1996). Solving the travelling salesman problem with a hopfield-type neural network. *Demonstratio Mathematica*, 29:219–231.

[Punnen, 2007] Punnen, A. P. (2007). The Traveling Salesman Problem: Applications, Formulations and Variations. In Gutin, G. and Punnen, A. P., editors, *The Traveling Salesman Problem and Its Variations*, pages 1–28. Springer US, Boston, MA.

[Talaván, 2002] Talaván, P. (2002). Parameter setting of the hopfield network applied to tsp. *Neural networks : the official journal of the International Neural Network Society*, 15:363–73.