UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS MATEMÁTICAS
Departamento de Estadística e Investigación Operativa

TESIS DOCTORAL

Algunas cuestiones notables sobre el modelo de Hopfield en optimización

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Lucas García Rodríguez

Directores

Francisco Javier Yáñez Gestoso
Pedro Martínez Talaván

Madrid, 2018

UNIVERSIDAD COMPLUTENSE DE MADRID

Facultad de Ciencias Matemáticas

Departamento de Estadística e Investigación Operativa

TESIS DOCTORAL

Algunas cuestiones notables sobre el modelo de Hopfield en optimización

Memoria para optar al grado de Doctor presentada por

Lucas García Rodríguez

Doctorado en Ingeniería Matemática, Estadística e Investigación Operativa

MADRID, 2017

Directores:

Dr. Francisco Javier Yáñez Gestoso
Universidad Complutense de Madrid

Dr. Pedro Martínez Talaván
Instituto Nacional de Estadística

A Esther y Marcos, por el camino recorrido
y el que queda por recorrer.

A mis padres, que me lo dieron todo.

Agradecimientos

Quizá sea ésta la parte de la memoria que más me ha costado comenzar a escribir y posiblemente será la más leída, en algunos casos la única. Con el fin de animar al lector a continuar, quisiera brevemente expresar mi gratitud:

A mis directores, Javier y Pedro, por sus enseñanzas y su ayuda desinteresada, por descubrirme el complejo mundo del modelo de Hopfield y guiarme por él. Gracias por la confianza y la paciencia todos estos años.

Al profesor Francisco Javier Montero de Juan, quién me acogió en su grupo de investigación, y cuyos proyectos TIN2015-66471-P y S2013/ICE-2845 (CASI-CAM) han soportado parcialmente este trabajo.

A mi compañera de viaje, Esther, a quién tantas horas juntos ha robado esta tesis. Compaginar vida laboral y doctorado no parecía tan complicado a priori. Gracias por el apoyo incondicional, por la comprensión, por escucharme en los momentos de atasco. Creo que por fin he llegado al óptimo. Y a nuestro hijo Marcos, cuya alegría me ha impulsado en la recta final y quién quizá algún día resuelva su propio rompecabezas.

A mis padres, Maite y Lucas, quienes me enseñaron todo aquello que no sale en los libros.

A todos ellos, gracias.

Índice general

Abstract

The continuous Hopfield model can be used as a heuristic to solve combinatorial optimization problems, such as the Traveling Salesman Problem (TSP). It consists of a recurrent neural network with an associated differential equation, whose states evolve from an initial point to an equilibrium point by minimizing a Lyapunov function. By associating this function with the objective function and constraints of the optimization problem, the equilibrium points correspond to feasible solutions of the optimization problem. Applying Hopfield's original formulation to solve the TSP, this process leaves a free parameter. Historically, researchers in the field have used small values of this free parameter, without explaining why better solutions are obtained.

This dissertation analyzes the relation between the free parameter and the saddle point of the Hopfield model. Whereas in small problems this result guarantees that the global optimum is always obtained, in more complex instances, such as the TSP, this is far more complicated. However, in the surroundings of the saddle point, the attractor basins for the best solutions grow as the free parameter decreases. Thus, saddle point neighbors become excellent starting point candidates for the simulation.

In order to continue improving the quality of Hopfield model solutions, the *Divide-and-Conquer* model is proposed, a model based in two consecutive Hopfield models. This model connects neighboring cities in the first phase and, in a second phase, the connected groups of cities from the previous phase. In both cases, the corresponding parametrizations of the problem are deduced and solved with the Hopfield model.

Finally, it is studied how a particular case of the Hopfield model used in the second phase of the *Divide-and-Conquer* model actually behaves as a *2-opt* algorithm, provided that the starting point is chosen appropriately. All these results improve the performance of the Hopfield model as a heuristic, matching it to the *2-opt* algorithm in terms of quality of the solution.

Resumen

El modelo de Hopfield continuo puede utilizarse como heurística para resolver problemas de optimización combinatoria, como es el caso del problema del viajante (TSP). Consiste en una red neuronal recurrente con una ecuación diferencial asociada, cuyos estados evolucionan de un punto inicial a un punto de equilibrio mediante la minimización de una función de Lyapunov. Asociando dicha función con la función objetivo y restricciones del problema de optimización, se consigue que los puntos de equilibrio se correspondan con soluciones factibles del problema de optimización. Aplicando la formulación original de Hopfield al TSP, este proceso deja un parámetro libre. Históricamente, los investigadores en el campo han utilizado valores pequeños de dicho parámetro, sin explicar por qué se obtienen mejores soluciones.

Esta memoria analiza la relación entre el parámetro libre y el punto de silla del modelo de Hopfield. Mientras que en problemas pequeños este resultado garantiza poder obtener siempre la solución óptima, en problemas más complejos como el TSP resulta más complicado. No obstante, las cuencas de atracción para las mejores soluciones se hacen más amplias en las proximidades del punto de silla cuando el parámetro libre decrece. Así, los puntos situados en las proximidades del punto de silla serán excelentes candidatos a punto inicial de la simulación.

Con el objetivo de continuar mejorando la calidad de las soluciones del modelo de Hopfield, se propone el modelo *Divide-y-Vencerás*, un modelo basado en dos modelos de Hopfield consecutivos. Este modelo conecta en la primera fase ciudades vecinas y en una segunda fase los grupos de ciudades conectados en la fase anterior. En ambos casos, se deducen las correspondientes parametrizaciones del problema y se resuelven con el modelo de Hopfield.

Finalmente, se estudia cómo un caso particular del modelo de Hopfield utilizado en la segunda fase del modelo *Divide-y-Vencerás*, se comporta en realidad como un algoritmo *2-opt*, siempre y cuando se elija adecuadamente el punto inicial. Todos estos resultados mejoran el rendimiento del modelo de Hopfield como heurística, equiparándolo al algoritmo *2-opt* en calidad de solución.

Prólogo

Esta memoria sintetiza el trabajo y las aportaciones desarrolladas en torno al estudio de la red o modelo de Hopfield aplicado a resolver problemas de optimización combinatoria.

En los últimos 15 años, los avances en Inteligencia Artificial apoyados en el proceso de miniaturización de los dispositivos electrónicos y en el acceso a grandes fuentes de datos, han permitido resolver problemas que en origen se antojaban como imposibles en campos como la visión artificial, el procesado del lenguaje natural o la robótica. Y es que estas técnicas no sólo están siendo utilizadas en éstas y otras disciplinas, sino que las están transformando. De entre todas estas técnicas destacan las redes neuronales artificiales, por su flexibilidad o capacidad de resolver problemas muy diversos.

Inspiradas en el funcionamiento del cerebro, las redes neuronales artificiales consisten en un conjunto o red de neuronas conectadas entre sí que trabajan conjuntamente para obtener un resultado. Al construir el modelo matemático, se modelizan tanto las conexiones neuronales, que transmiten información de una neurona a otra, como la respuesta de una neurona a un determinado estímulo. Esta característica, que permite a la neurona modificar sus conexiones, dota a las redes neuronales artificiales de la capacidad de aprender en base a la información proporcionada.

Las redes neuronales artificiales destacan por tener una gran habilidad para resolver tres grandes tipos de problemas: de regresión (por ejemplo, predecir con precisión la demanda energética diaria para cubrir las necesidades de una determinada población en las próximas 24 horas), de clasificación (como la clasificación de imágenes de tráfico en el contexto del desarrollo del vehículo autónomo) y de clustering (como es el caso del grado de riesgo que asume una entidad financiera a la hora de conceder un crédito). Todas estas redes neuronales buscan minimizar algún tipo de error en el modelo, para ser lo más precisas posibles. Pero pese a minimizar la función del error, el objetivo de la red no es resolver un problema de optimización, sino de regresión, clasificación o clustering. La pregunta planteada sería: ¿podría una red neuronal artificial ser diseñada para resolver un problema de optimización?

La red de Hopfield, creada por el científico americano John J. Hopfield, surge a principios de los años 80 para resolver un problema de memoria asociativa con unidades binarias. Esta red recurrente es capaz de recordar un número de patrones, reproduciendo el patrón adecuado a partir de una determinada entrada. Un par de años más tarde, el mismo Hopfield crearía una generalización, el modelo continuo, trabajo que además pondría el "apellido" discreto a su modelo anterior.

El modelo de Hopfield continuo, cuya variable de estado permite tomar todos los valores en el intervalo $[0, 1]$ y cuya dinámica se comporta como una ecuación diferencial ordinaria, podía resolver problemas de optimización combinatoria, como es el caso del problema del viajante.

Esta memoria expone algunas cuestiones notables acerca del uso del modelo de Hopfield como heurística para resolver problemas de optimización combinatoria. Está organizada en seis capítulos y un apéndice.

El capítulo 1 presenta las redes neuronales artificiales, comenzando con la motivación biológica, su estructura básica y un breve recorrido por su historia. A continuación, se analizan los principales procesos de aprendizaje de una red neuronal, lo que llevará a introducir el modelo de Hopfield, tanto en su versión discreta como continua, objeto de análisis en esta memoria. Finalmente, se abordan las investigaciones recogidas en la literatura hasta la fecha sobre el modelo de Hopfield, especialmente en el contexto de su aplicación al problema del viajante.

El capítulo 2 se centra en la importancia del punto inicial de la simulación del modelo de Hopfield y su relación con el parámetro libre del modelo, lo cual permite llevar a cabo un análisis de sus cuencas de atracción. Para ello, se introducen formalmente tanto el problema de asignación cuadrática generalizado como el problema del viajante, con sus correspondientes modelos de Hopfield. El cálculo analítico del punto de silla (resultado que aporta una importante reducción en la complejidad del cálculo del mismo para el modelo de Hopfield aplicado al TSP) y el análisis de las cuencas de atracción para cada uno de los dos problemas, se abordan en las secciones 2.3 y 2.5, siendo estos aportaciones originales.

El resto de capítulos y el apéndice A, son también aportaciones originales, a excepción de la conocida heurística *2-opt* introducida en la sección 4.2 y los métodos de simulación recogidos en la sección 5.2.

El capítulo 3 presenta una heurística llamada *Divide-y-Vencerás* que consiste en el uso de dos modelos de Hopfield sucesivos aplicados a resolver el problema del viajante, y que permite mejorar notablemente los resultados del modelo original. El primer modelo se centra en conectar ciudades próximas mientras que el segundo se ocupa de conectar los extremos de núcleos de ciudades alejadas resultado del primer modelo. Se realizan para ello las correspondientes proyecciones y análisis de estabilidad de cada uno de los dos modelos.

El capítulo 4 muestra como el modelo de Hopfield se comporta exactamente igual que el algoritmo *2-opt*, reconocido como una de las heurísticas más populares para resolver el problema del viajante. Los *intercambios 2-opt* que realiza esta heurística, pueden modelizarse como un modelo de Hopfield. Más concretamente, un caso particular de la segunda fase realizada en la heurística *Divide-y-Vencerás* se comporta igual que un *intercambio 2-opt*, deshaciendo los cruces obtenidos en una solución inicial.

El capítulo 5 detalla la implementación de la estrategia *Divide-y-Vencerás* y recoge todas las experiencias computacionales llevadas a cabo en el desarrollo de esta memoria. Se incluyen la comparación del modelo *Divide-y-Vencerás* con el modelo de Hopfield tradicional, la búsqueda de parámetros óptimos del modelo *Divide-y-Vencerás* y el uso del modelo de Hopfield como *2-opt*. Se aportan también experiencias computacionales utilizando la GPU,

especialmente apropiadas para resolver instancias grandes del TSP. En este aspecto, se resuelven las instancias más grandes del TSP abordadas con el modelo de Hopfield hasta donde se tiene constancia, siendo el problema más grande de 13509 ciudades.

El capítulo 6 expone las conclusiones y líneas futuras de investigación resultado de esta memoria. El uso del modelo de Hopfield como optimizador, en origen criticado por su incapacidad de garantizar soluciones factibles, y más recientemente una vez resuelto ese problema, por la baja calidad de sus soluciones, se aborda desde varios ángulos: la selección adecuada del parámetro libre del modelo, y el uso de heurísticas basadas en el propio modelo. De este modo, se mejora la calidad de la solución del modelo de Hopfield hasta el punto de hacerlo competitivo con otros modelos y equipararlo con el *2-opt*, abriendo líneas futuras de trabajo que permitirían mejorar aún más la calidad de las soluciones.

Finalmente, el apéndice A incluye una guía de uso de la librería *Hopfield Network Toolbox*, desarrollada en MATLAB® como complemento a esta memoria para poder abordar todas las experiencias computacionales que aquí se recogen. La librería está alojada como código abierto en la plataforma de desarrollo colaborativo GitHub.

Abreviaturas

ADALINE	*ADAptive LInear NEuron*
AI	Inteligencia Artificial (*Artificial Intelligence*)
ANN	Red Neuronal Artificial (*Artificial Neural Network*)
BCM	Matriz Circulante por Bloques (*Block Circulant Matrix*)
CHN	Red de Hopfield Continua (*Continuous Hopfield Network*)
CNN	Red Neuronal Convolucional (*Convolutional Neural Network*)
CPU	Unidad Central de Procesamiento (*Central Processing Unit*)
GPU	Unidad de Procesamiento Gráfico (*Graphics Processing Unit*)
GQKP	Problema de Asignación Cuadrática Generalizado (*Generalized Quadratic Assignment Problem*)
LMS	*Least Mean Square*
MADALINE	*Multiple ADAptive LInear NEuron*
MLP	Perceptrón Multicapa (*Multi-Layer Perceptron*)
QAP	Problema de Asignación Cuadrática (*Quadratic Assignment Problem*)
SOM	Mapa Auto-Organizado (*Self-Organizing Map*)
SVM	Máquinas de Vector Soporte (*Support Vector Machines*)
TLU	*Threshold Logic Unit*
TSP	Problema del Viajante (*Traveling Salesman Problem*)

Notación

Esta memoria utiliza, en la medida de lo posible, notación estándar. De este modo, texto, fórmulas y figuras siguen la siguiente notación:

- Letras en *cursiva* representan escalares: A, B, C, D, N, N', m, $v_{x,i}$, $T_{xi,yj}$, $i^b_{x,i}$, ...
- Letras minúscula en **negrita** denotan vectores (columna): $\mathbf{v}$, $\mathbf{i}^b$, $\mathbf{1}$, ...
- Letras mayúscula en **NEGRITA** representan matrices: $\mathbf{T}$, $\mathbf{Q_k}$, $\mathbf{I}$, $\mathbf{D}$, $\mathbb{1}$, ...

Se resaltan a continuación algunos casos aclaratorios:

$\mathbf{v}^t$		traspuesta del vector $\mathbf{v}$
$\mathbf{1}$	$= \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N\times 1}$	vector de unos de tamaño N
$\mathbb{1}$	$= \mathbf{1}\cdot\mathbf{1}^t = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}_{N\times N}$	matriz de unos de tamaño $N \times N$
$\mathbf{D}$	$= (d_{x,y})_{x,y\in\{1,\dots,N\}}$	matriz de distancias $N \times N$ del TSP
$\mathbf{V}$	$= (v_{x,i})_{x,i\in\{1,\dots,N\}}$	matriz de estados $N \times N$ de la CHN aplicado al TSP
$\mathbf{v}_{\cdot i}$	$= \begin{bmatrix} v_{1,i} \\ v_{2,i} \\ \vdots \\ v_{N,i} \end{bmatrix}_{N\times 1}$	vector de estados i de tamaño N de la CHN aplicado al TSP
$\mathbf{v}$	$= \begin{bmatrix} \mathbf{v}_{\cdot,1} \\ \mathbf{v}_{\cdot,2} \\ \vdots \\ \mathbf{v}_{\cdot,N} \end{bmatrix}_{N\times 1}$	vector de estados de tamaño $N^2 \times 1$ de la CHN aplicado al TSP

En ocasiones se indicará, para evitar confusión, los tamaños de las matrices cuadradas utilizando paréntesis: $(\mathbb{1})_N$, $(\mathbb{1})_2$, $(\mathbf{D})_N$, etc.

CAPÍTULO 1

Redes Neuronales Artificiales y el modelo de Hopfield

Este capítulo presenta las redes neuronales artificiales desde sus orígenes e introduce las redes de Hopfield, objeto de estudio en esta tesis. Comenzando con la motivación biológica, se define qué se entiende por red neuronal artificial y se lleva a cabo un breve recorrido por su historia, desde la década de 1940 hasta la época actual. Se presentan los distintos tipos de aprendizaje para centrarse en un modelo concreto de aprendizaje no supervisado, el modelo de Hopfield y en su aplicación a la resolución de problemas de optimización combinatoria, como es el caso del problema del viajante.

Contenidos del capítulo

1.1 Redes Neuronales Artificiales

Las redes neuronales artificiales tienen una gran importancia dentro del campo de la Inteligencia Artificial. Inspirándose en el comportamiento del cerebro humano, las redes neuronales artificiales tienen como objetivo resolver problemas que son complicados de resolver utilizando técnicas convencionales. Cabe señalar que pese a los grandes avances en el Aprendizaje Automático (*Machine Learning*) , las redes neuronales artificiales están todavía muy lejos de poder modelizar la complejidad del cerebro humano. Sin embargo, son capaces de abordar problemas sencillos para un humano, pero difíciles de resolver para un ordenador, como es el caso de la clasificación y detección de objetos en imágenes, identificación de caracteres, reconocimiento de voz, etc.

Los fundamentos e inicios de las redes neuronales artificiales están muy estrechamente relacionados con el funcionamiento del cerebro humano y, en concreto, con las redes neuronales biológicas.

1.1.1 Motivación biológica

El cerebro humano consiste, de modo simplificado, en un gran número de elementos altamente interconectados, llamados neuronas. El concepto de neurona fue introducido a finales del siglo XIX por Santiago Ramón y Cajal [37], quien pensaba en las neuronas como estructuras básicas constituyentes del cerebro. A día de hoy, su aportación sigue considerándose la mayor revolución en el campo de la neurociencia de todos los tiempos. Defendió la teoría de que las neuronas se interconectaban entre sí de forma paralela (ver figura 1.1), y no formando un circuito cerrado como es el caso del sistema sanguíneo.

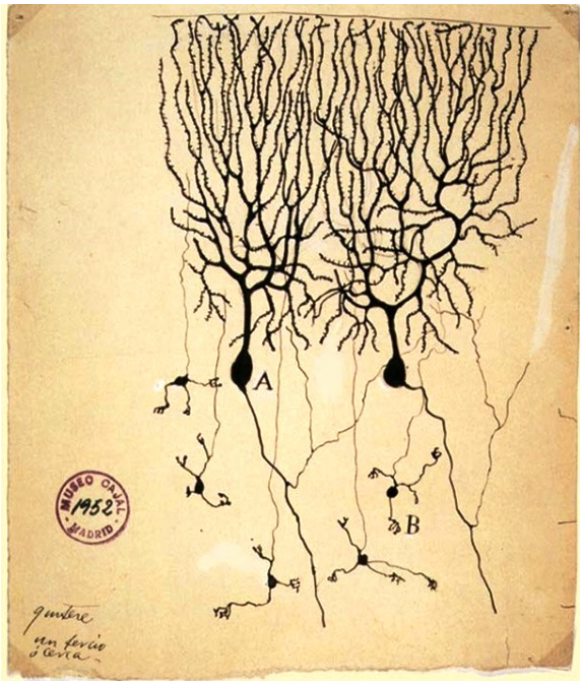


Figura 1.1: Dibujo de neuronas realizado por Ramón y Cajal

Las neuronas son un tipo de células del sistema nervioso (ver figura 1.2) formadas por

un cuerpo celular central denominado *soma*, del que surge un denso árbol de ramificaciones denominadas *dendritas* y una larga prolongación tubular llamada *axón*.

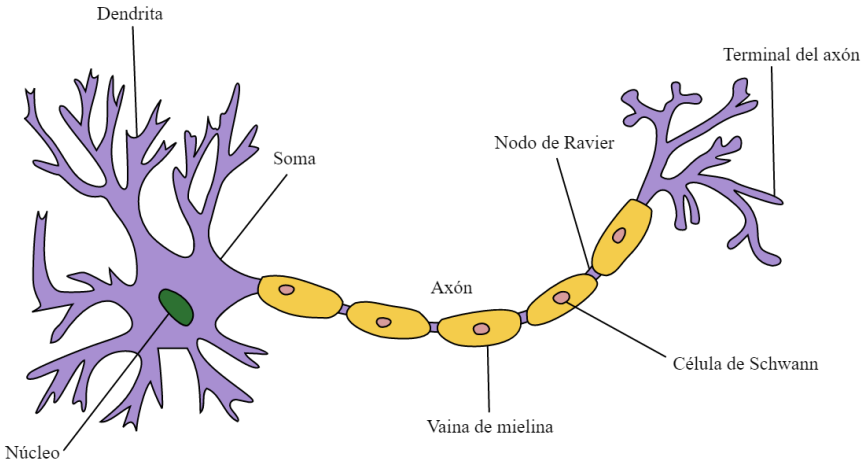


Figura 1.2: Estructura básica de una neurona biológica

Cada neurona recibe impulsos electroquímicos procedentes de otras neuronas en las dendritas. Si la suma de las entradas eléctricas es suficiente para activar la neurona, ésta transmite una señal electromecánica a través del axón, que a su vez se ramifica en ramificaciones axonales que van a conectarse con otras neuronas por sus dendritas. El conjunto de elementos que hay entre la ramificación axonal y la dendrita forman una conexión llamada *sinapsis* que regula la transmisión del impulso eléctrico mediante unos elementos bioquímicos, los *neurotransmisores*.

En estado de reposo, la carga de una célula inactiva se mantiene en valores negativos (en torno a -70 mV) variando dentro de unos estrechos márgenes. Cuando el potencial de membrana se despolariza más allá de un cierto umbral, la célula dispara un potencial de acción consistente en un cambio rápido en la polaridad de la membrana de negativo a positivo y vuelta a negativo, proceso que dura unos milisegundos. Así, las neuronas transmiten ondas de naturaleza eléctrica cuya propagación se debe a la existencia de una diferencia de potencial entre la parte interna (*plasma intracelular*) y externa (*fluido intersticial*) de la célula. Una de las características más importantes de las conexiones sinápticas es la plasticidad: la capacidad del sistema nervioso para modificar la intensidad de las conexiones a partir de su interacción con el entorno. Los mecanismos anteriores constituyen el fundamento del aprendizaje en el cerebro.

El aprendizaje asociativo más básico, conocido como *condicionamiento clásico*, fue demostrado por primera vez [34] por el fisiólogo ruso Iván Petróvich Pávlov (1849-1936) mientras estudiaba los procesos de digestión. Se dio cuenta que al ponerle la comida a un perro, este salivaba (ver figura 1.3). De este modo, se daba una respuesta incondicionada a un estímulo incondicionado. Su experimento consistió en hacer sonar una campana cada vez que se le pusiera la comida, de modo que, cuando el perro la escuchara, asociara ese sonido con la comida y salivara. Así, al cabo del tiempo, el estímulo de la campana (originalmente neutro) era asociado, independientemente de si iba unido a la comida, con la salivación. Se producía de este modo una respuesta condicionada a partir de un estímulo condicionado.

Figura 1.3: Ivan Pavlov y su personal, demostrando el fenómeno del reflejo de la condición con un perro

En 1949, Donald O. Hebb resumió sus dos décadas de investigación en el libro *The Organization of Behavior* [14]. La principal premisa del libro es que el comportamiento podía explicarse mediante la acción de las neuronas. Este postulado llegó a ser conocido como aprendizaje Hebbiano:

> *Cuando el axón de una célula A está lo suficientemente cerca como para excitar a una célula B y repetidamente toma parte en la activación, ocurren procesos de crecimiento o cambios metabólicos en una o ambas células de manera que tanto la eficiencia de la célula A, como la capacidad de excitación de la célula B son aumentadas.*

Este postulado sugiere un mecanismo físico para el aprendizaje a nivel celular. Aunque Hebb nunca afirmó tener evidencia fisiológica firme para su teoría, la investigación posterior ha demostrado que algunas células exhiben el aprendizaje de Hebb. Las teorías de Hebb continúan influyendo en la investigación actual en neurociencia.

Típicamente, las neuronas son entre cinco y seis órdenes de magnitud más lentas que las puertas lógicas de silicio. Los eventos en un chip de silicio tienen lugar en el ámbito de los nanosegundos, mientras que los eventos neuronales ocurren en el rango de milisegundos. Sin embargo, el cerebro es capaz de compensar la tasa relativamente lenta de operación de la neurona mediante un número realmente asombroso de neuronas con interconexiones masivas entre ellas. Se estima que en el córtex cerebral humano hay aproximadamente 10,000 millones (10×10^9) de neuronas y 60 billones (60×10^{12}) de sinapsis o conexiones [45].

Además, esta idea de ponderación entre las conexiones de las neuronas propuesta por Hebb ha servido para sentar las bases del aprendizaje de las *Redes Neuronales Artificiales*.

1.1.2 ¿Qué es una Red Neuronal Artificial?

Una *Red Neuronal Artificial* –ANN– (en inglés, *Artificial Neural Network*), es un modelo matemático abstracto inspirado en las estructuras, mecanismos y funciones cerebrales.

La *neurona artificial* (que en adelante, siempre y cuando no acarree confusión con la neurona biológica, llamaremos, abusando del lenguaje, *neurona*) consiste en la unidad de

procesamiento de la información que es fundamental para el funcionamiento de la red neuronal artificial, capaz de, a partir de una serie de entradas, producir una salida o respuesta. En su estructura más básica, la neurona consta de los siguientes elementos (ver figura 1.4):

- Las *entradas*, que reciben los datos provenientes de otras neuronas. En la neurona biológica (ver figura 1.2) se corresponderían con las dendritas.
- Un conjunto de *pesos* (o conexiones sinápticas).
- Una *suma ponderada*.
- Una *función de activación* o *de transferencia*.

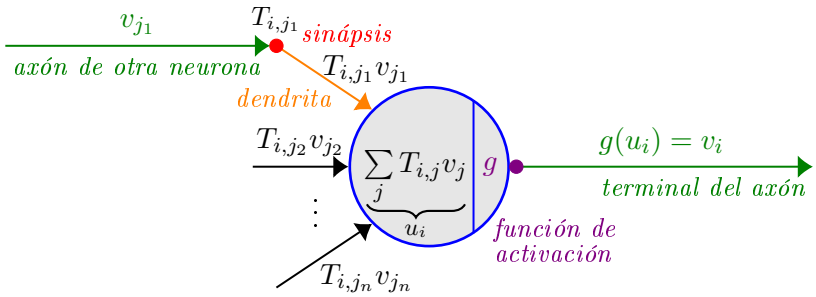


Figura 1.4: Neurona artificial

1.1.3 Breve historia de las Redes Neuronales Artificiales

Está ampliamente reconocido que el enfoque moderno de las *Redes Neuronales Artificiales* tiene su origen a comienzo de la década de 1940, con el trabajo de Warren McCulloch y Walter Pitts [29]. Desarrollaron un modelo neuronal, conocido como *Threshold Logic Unit* (TLU), en el que la suma ponderada de señales de entrada es comparada con un umbral con el fin de calcular la salida de la neurona. Si el valor calculado es mayor o igual que el umbral, la salida es 1. En caso contrario, la salida es 0. Consistía, en realidad, en aplicar como función de activación una función escalón o de Heaviside. Desde el comienzo se observó que utilizando este modelo de red neuronal podían implementarse fácilmente funciones lógicas como puertas AND y OR, y utilizarlas de modo conjuntivo o disyuntivo. El trabajo de McCulloch y Pitts sentó así las bases de las redes neuronales artificiales e inspiró modelos neuronales mucho más complejos. Sin embargo su trabajo carecía de un método de aprendizaje que permitiese modificar los pesos de la red.

No es hasta finales de la década de 1950 que Frank Rosenblatt [40] desarrolla el primer modelo de red neuronal (basada en la red de McCulloch y Pitts) con una clara aplicación práctica, *el Perceptrón*, que permite llevar a cabo reconocimiento de patrones. El Perceptrón de una sola capa podía clasificar un conjunto de entradas de valor continuo en una de dos clases. Acompañando a su modelo neuronal, Rosenblatt aporta también una regla de aprendizaje, inspirada en el trabajo de Donald O. Hebb (introducido en la sección 1.1.1). Demostró que la regla de aprendizaje para el Perceptrón siempre convergería a los valores adecuados de los pesos, siempre que existieran pesos que resolvieran el problema. El Perceptrón se

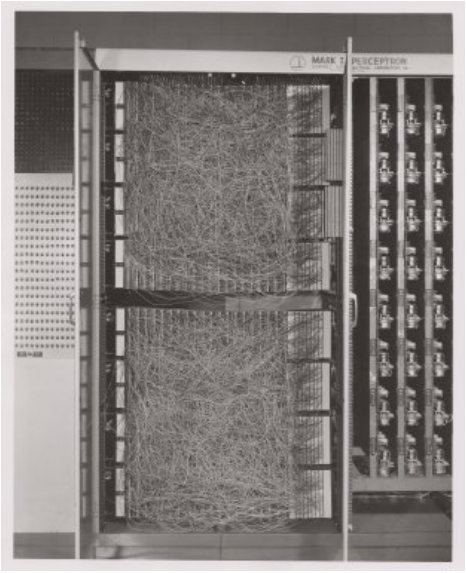


Figura 1.5: La máquina Mark I Perceptron, un sistema para reconocimiento de patrones, del Cornell Aeronautical Laboratory.

desarrolló en realidad con la intención de ser una máquina en vez de un algoritmo, y aunque su primera implementación fue en software para el IBM 704, posteriormente se implementó en hardware, el *Mark I Perceptron* (ver figura 1.5). La máquina fue diseñada para llevar a cabo reconocimiento de imágenes, disponía de una matriz de 20×20 fotocélulas conectadas al azar a las neuronas. Los pesos se codificaron en potenciómetros y las actualizaciones de los pesos durante el proceso de aprendizaje tenían lugar mediante motores eléctricos.

El interés en el campo de las redes neuronales artificiales crece significativamente. A comienzo de la década de 1960, Bernard Widrow y Martian E. "Ted" Hoff [57] introdujeron la red neuronal *ADALINE (ADAptative LInear NEuron)* junto con una regla de aprendizaje, *Least Mean Square* (LMS). A diferencia del Perceptrón propuesto por Rosenblatt, la red de Widrow y Hoff eliminaba la función de activación escalón de Heaviside. Sin que la salida de la neurona fuera modificada mediante el umbral de la función de activación, la medida de cuánto cambia el error cuando cambia el peso (la derivada) puede utilizarse para bajar el error y encontrar los valores de pesos óptimos. Esta regla para llevar a cabo el aprendizaje, en la que se encuentran los pesos adecuados utilizando las derivadas del error de entrenamiento con respecto a cada peso, continúa siendo utilizada para entrenar redes neuronales hoy día. La regla de aprendizaje LMS resultó ser más potente y exitosa que la del Perceptrón. Pese a que la regla del Perceptrón tiene garantizada su convergencia a una solución que clasifica correctamente los patrones de entrenamiento, la red es muy sensible al ruido, dado que frecuentemente los patrones se encuentran próximos a una frontera de decisión. El algoritmo LMS minimiza el error cuadrático medio y por tanto intenta desplazar las fronteras de decisión tan lejos como sea posible de los patrones de entrenamiento. Widrow y Hoff desarrollaron también una extensión de su red llamada MADALINE (Multiple ADAptative LInear NEuron), consistiendo en un filtro adaptativo que permitía eliminar el eco en las llamadas telefónicas.

La época dorada por la que pasaban las redes neuronales se vio duramente golpeada en

1969, cuando Marvin Minsky y Seymour Papert (fundador y director respectivamente del MIT AI Lab) publicaron [31] su escepticismo acerca del Perceptrón, exponiendo mediante un análisis riguroso las limitaciones de las redes del momento. Entre estas limitaciones, el Perceptrón era incapaz de aprender la sencilla función lógica XOR, al tratarse de un problema que no puede separarse linealmente. Rosenblatt y Widrow, conocedores de esas limitaciones, propusieron redes más complejas que pudieran abordarlas, pero no fueron capaces de modificar sus reglas de aprendizaje. Un grandísimo número de investigadores, influenciados por el trabajo de Minsky y Papert, creyeron que continuar trabajando en redes neuronales era un callejón sin salida y, junto con el hecho de que no existieran ordenadores digitales potentes con los que experimentar, decidieron abandonar sus investigaciones. Durante una década, la investigación en redes neuronales estuvo prácticamente suspendida (época conocida como "invierno en Inteligencia Artificial" –*"AI winter"*–).

Pese al largo inviero en IA, algunos como Teuvo Kohonen [23] y James Anderson [3] continuaron sus investigaciones, desarrollando redes neuronales que podían actuar como memorias. Stephen Grossberg [12] también estuvo activo durante este periodo en el desarrollo de una red competitiva en tiempo continuo inspirada en la fisiología del desarrollo de la corteza visual. Una estructura similar sería utilizada por Kohonen años más tarde [24] en el desarrollo de los *Mapas Auto-Organizados (Self-Organizing Maps –SOM–)* o *redes de Kohonen*.

A comienzos de la década de 1980, coincidiendo con el desarrollo del ordenador personal y estaciones de trabajo, dos trabajos influyeron enormemente en el resurgir de las redes neuronales artificiales. El primero fue el desarrollo por parte de John J. Hopfield de una red recurrente [17] basada en el concepto de memoria asociativa. Los trabajos de Hopfield se caracterizaron por la practicidad, tanto en la implementación de sus redes, como en los problemas que podían resolver. Entre las aplicaciones que describió en sus trabajos destacan la memoria asociativa, la conversión analógico-digital y la optimización. La claridad, rigor y legibilidad de sus artículos, junto con su enorme carisma hizo que muchos investigadores se interesasen de nuevo por las redes neuronales artificiales.

El segundo desarrollo clave en el resurgir de las redes neuronales fue el algoritmo de retropropagación (*backpropagation*) para el entrenamiento de redes Perceptrón con múltiples capas (permitiendo resolver problemas de reconocimiento de patrones no linealmente separables), descubierto de manera independiente por varios investigadores. Sin embargo, la descripción por parte de David E. Rumelhart, Geoffrey E. Hinton y Ronald J. Williams [41] fue la más aceptada y extendida, haciéndose popular a través del libro *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* [42]. Pese a todo, la primera descripción de un algoritmo para entrenar redes con múltiples capas estaba contenida en la tesis doctoral de 1974 de Paul Werbos, que desafortunadamente no se difundió entre la comunidad de investigadores del momento trabajando en redes neuronales. El descubrimiento del algoritmo de retropropagación dio lugar a un torrente de investigaciones que ha continuado hasta la época actual. Las redes Perceptrón Multicapa (*Multi-Layer Perceptron –MLP–*), entrenadas mediante el algoritmo de retropropagación, siguen siendo a día de hoy las redes neuronales artificiales más utilizadas.

Desde entonces, el interés por las redes neuronales artificiales ha continuado su crecimiento ascendente, se han desarrollado una grandísima variedad de arquitecturas y algoritmos de

aprendizaje, así como encontrado una gran cantidad de aplicaciones. Uno de los avances que ha tenido un mayor impacto son las Máquinas de Vector Soporte (*Support Vector Machines –SVM–*) de Vapnik y Cortes [4], originalmente conocidas como *Support-Vector Networks*. Las Máquinas de Vector Soporte (muy próximas a las redes neuronales) buscan un hiperplano que separe de forma óptima a los puntos de una clase de otra, que eventualmente han podido ser previamente proyectados a un espacio de dimensionalidad superior.

Yann LeCun y sus colaboradores [27] ponen a finales de la década de 1990 los cimientos del Aprendizaje Profundo (*Deep Learning*), con un largo y detallado artículo sobre Redes Neuronales Convolucionales (*Convolutional Neural Networks –CNN–*), en el que se utilizan un número mayor de capas que permiten tanto aprender características como llevar a cabo la clasificación de las entradas, y describen aplicaciones para reconocimiento de caracteres escritos a mano y detección de caras. Sin embargo, tendrán que pasar algunos años para que la madurez de la tecnología (GPUs) junto con nuevos desarrollos como el de Alex Krizhevsky y sus colaboradores [26] permitan poder entrenar una CNN con un grandísimo conjunto de datos (1.2 millones de imágenes de 1000 clases de objetos distintos) y mejorar cualquier otra técnica de clasificación. La CNN propuesta por Alex Krizhevsky ganó la competición ImageNet Large Scale Visual Recognition Challenge en 2012 y sentó las bases de los desarrollos actuales. Actualmente, este tipo de redes neuronales profundas pueden mejorar la precisión que tendría un humano al clasificar dichas imágenes. El aprendizaje profundo se ha consolidado como uno de los métodos más precisos para el reconocimiento de objetos y es clave para sistemas punteros como el desarrollo de sistemas de ayuda a la conducción, el vehículo autónomo, equipos de rescate basados en imágenes o robots autónomos.

1.2 Procesos de aprendizaje

Del mismo modo en que hay diferentes maneras en las que aprendemos del entorno que nos rodea, resulta habitual en el aprendizaje automático, y en particular en las redes neuronales artificiales, el clasificar los algoritmos según el proceso de aprendizaje que realizan. El aprendizaje es el proceso o regla por la cual una red neuronal modifica sus pesos en respuesta a una información de entrada. El objetivo de la regla de aprendizaje no es otro que el de entrenar la red para llevar a cabo una determinada tarea.

Es frecuente encontrar diferentes clasificaciones del proceso de aprendizaje. En esta sección se distinguen dos tipos de reglas de aprendizaje: aprendizaje supervisado y aprendizaje no supervisado.

En ocasiones, también se habla de un tercer tipo de aprendizaje, el aprendizaje reforzado. El aprendizaje reforzado es similar al aprendizaje supervisado, excepto que en vez de proporcionar la salida correcta para cada una de las entradas a la red, se recibe una valoración (*score*) acerca de la idoneidad de la respuesta dada.

1.2.1 Aprendizaje supervisado

El *aprendizaje supervisado* (en ocasiones también llamado *aprendizaje con profesor*) se caracteriza porque el proceso de aprendizaje está controlado por un agente externo (super-

visor, profesor). Dado un conjunto de ejemplos (o datos de entrenamiento):

$$\{\mathbf{p_1}, \mathbf{t_1}\}, \{\mathbf{p_2}, \mathbf{t_2}\}, \ldots, \{\mathbf{p_m}, \mathbf{t_m}\}$$

donde $\mathbf{p_i}$ es la entrada de la red y $\mathbf{t_i}$ es su correspondiente salida correcta (*target*), el proceso de aprendizaje consiste en aplicar la entrada $\mathbf{p_i}$ en la red neuronal y comparar su salida con el valor real $\mathbf{t_i}$, obteniendo una medida de error. La regla de aprendizaje consistirá en ajustar los pesos de la red a partir de este error obtenido con el fin de acercar la salida obtenida a la salida correcta. De este modo, en el aprendizaje supervisado, el conocimiento del supervisor o profesor (las relaciones entre las entradas y salidas reales) es transferido a la red neuronal durante el proceso de entrenamiento.

Existen un gran número de arquitecturas neuronales que llevan a cabo un aprendizaje supervisado. De entre todas, la que más destaca tanto por su desarrollo histórico (presentado en la sección 1.1.3) como por su extendido uso en una gran variedad de aplicaciones, es el Perceptrón.

El Perceptrón

Partiendo de la definición de neurona de la sección 1.1.2, se construye la arquitectura del Perceptrón (ver figura 1.6) del siguiente modo:

Dada una red neuronal de una sola capa con n neuronas, y un conjunto de m entradas $(p_1, p_2, \ldots, p_m)$, utilizando una matriz de pesos $\mathbf{T}$ de tamaño $n \times m$ ($T_{i,j}$, con $i = 1, 2, \ldots, n$ y $j = 1, 2, \ldots m$), un vector de umbrales $\mathbf{i^b}$ (i_i^b, con $i = 1, 2, \ldots, n$) y la función de activación de McCulloch-Pitts:

$$v_i = g(u_i) = \begin{cases} 0 & \text{si} \quad u_i < 0 \\ 1 & \text{si} \quad u_i \geq 0 \end{cases} \qquad \forall i \in \{1, \ldots, n\}$$

la salida de la red Perceptrón vendrá determinada por $\mathbf{v} = g(\mathbf{u}) = g(\mathbf{Tp} + \mathbf{i^b})$

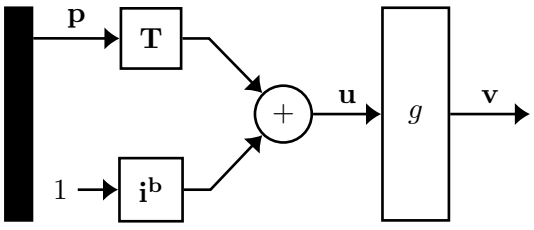


Figura 1.6: Arquitectura del Perceptrón

Observación 1.1. *Nótese que en el caso de la definición de neurona artificial (ver figura 1.4), las entradas a la neurona provienen de otra neurona. En este caso, se distingue entre las entradas a la red neuronal,* $\mathbf{p}$*, y las salidas de la misma,* $\mathbf{v}$*.*

Observación 1.2. *El umbral o potencial de entrada* $\mathbf{i^b}$ *junto con la función de activación, pueden interpretarse como el modelo neuronal desarrollado por McCulloch y Pitts (introducido en la sección 1.1.3) en el que la salida de la neurona toma su valor en función de la comparación de la suma ponderada de los pesos con un valor umbral.*

En otra interpretación diferente, el umbral o potencial de entrada $\mathbf{i^b}$ *podría incorporarse a la matriz* $\mathbf{T}$*, teniendo siempre como entrada el valor* 1.

La regla de aprendizaje, que permite clasificar patrones de clases linealmente separables convergiendo en un número finito de pasos, puede escribirse matricialmente como:

$$\mathbf{T}_{new} = \mathbf{T}_{old} + \alpha \mathbf{e}\mathbf{p}^t$$

$$\mathbf{i^b}_{new} = \mathbf{i^b}_{old} + \alpha \mathbf{e}$$

$$\mathbf{e} = \mathbf{t} - \mathbf{v}$$

Esta regla, conocida como *Regla Delta* al calcular la diferencia entre la salida real y la actual se conoce también como algoritmo de Widrow-Hoff, en honor a sus creadores.

Ejemplo 1.1. *Con el objetivo de ilustrar el desarrollo histórico de la sección 1.1.3, se muestra a continuación cómo se puede diseñar un Perceptrón que implemente una sencilla función lógica: la puerta AND (figura 1.7).*

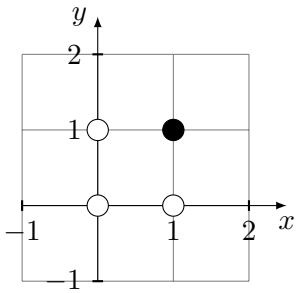


Figura 1.7: Puerta AND

En este caso, los pares de entradas y salidas correctas para la puerta AND son:

$$\{\mathbf{p_1} = \left[\begin{smallmatrix}0\\0\end{smallmatrix}\right], t_1 = 0\}, \{\mathbf{p_2} = \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right], t_2 = 0\}, \{\mathbf{p_3} = \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right], t_3 = 0\}, \{\mathbf{p_4} = \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right], t_4 = 1\}$$

Este problema puede abordarse a través de una red neuronal con sólo una neurona. Para simplificar el proceso de aprendizaje se utilizará $\alpha = 1$*. El proceso de aprendizaje seguido a continuación está detallado en la figura 1.8.*

Típicamente la matriz de pesos y umbrales son inicializados aleatoriamente. Supongamos que en este caso comenzamos con los valores:

$$\mathbf{T} = \begin{bmatrix} 3 & 3 \end{bmatrix}, \quad i^b = -2$$

1. *El primer paso consistirá en aplicar la primera entrada, el vector* $\mathbf{p_1}$*, a la red:*

$$v = g(\mathbf{T}\mathbf{p_1} + i^b) = g\Big(\begin{bmatrix} 3 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 2 \Big) = g(-2) = 0$$

Se calcula el error:

$$e = t_1 - v = 0$$

por lo que no es necesario actualizar los pesos y umbral, estando $\mathbf{p_1}$ *correctamente clasificado y siendo* $\mathbf{T}_{new} = \mathbf{T}_{old}$ *y* $i^b_{new} = i^b_{old}$.

2. *Se repite el proceso para la segunda entrada, el vector* $\mathbf{p_2}$*:*

$$v = g(\mathbf{T}\mathbf{p_2} + i^b) = g\Big(\begin{bmatrix}3 & 3\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} - 2\Big) = g(1) = 1$$

Se calcula el error:

$$e = t_2 - v = -1$$

Y se actualizan los pesos y umbral:

$$\mathbf{T}_{new} = \mathbf{T}_{old} + e\mathbf{p_2}^t = \begin{bmatrix}3 & 3\end{bmatrix} - 1\begin{bmatrix}0 & 1\end{bmatrix} = \begin{bmatrix}3 & 2\end{bmatrix}$$

$$i^b_{new} = i^b_{old} + e = -2 - 1 = -3$$

3. *El tercer paso consistirá en aplicar la tercera entrada, el vector* $\mathbf{p_3}$*, a la red:*

$$v = g(\mathbf{T}\mathbf{p_3} + i^b) = g\Big(\begin{bmatrix}3 & 2\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} - 3\Big) = g(0) = 1$$

Se calcula el error:

$$e = t_3 - v = -1$$

Y se actualizan los pesos y umbral:

$$\mathbf{T}_{new} = \mathbf{T}_{old} + e\mathbf{p_4}^t = \begin{bmatrix}3 & 2\end{bmatrix} - 1\begin{bmatrix}1 & 0\end{bmatrix} = \begin{bmatrix}2 & 2\end{bmatrix}$$

$$i^b_{new} = i^b_{old} + e = -3 - 1 = -4$$

4. *Se repite el proceso para la cuarta entrada, el vector* $\mathbf{p_4}$*:*

$$v = g(\mathbf{T}\mathbf{p_4} + i^b) = g\Big(\begin{bmatrix}2 & 2\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} - 4\Big) = g(0) = 1$$

Se calcula el error:

$$e = t_4 - v = 0$$

por lo que no es necesario actualizar los pesos y umbral, estando $\mathbf{p_4}$ *correctamente clasificado y siendo* $\mathbf{T}_{new} = \mathbf{T}_{old}$ *y* $i^b_{new} = i^b_{old}$.

La regla de aprendizaje consigue separar el conjunto de datos en dos, puesto que los conjuntos son linealmente separables. Sin embargo, la frontera obtenida no separa de manera equidistante los puntos de una clase de los de la otra.

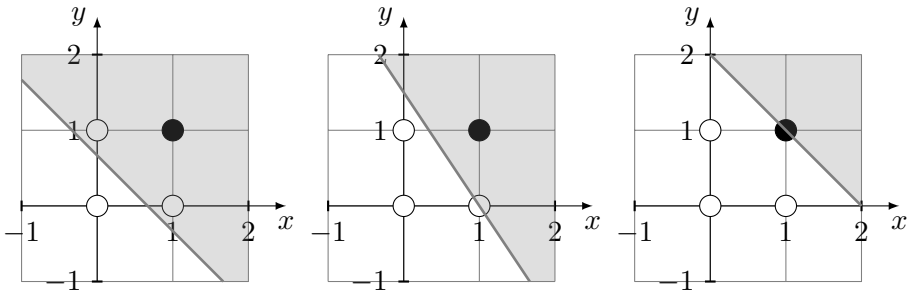


Figura 1.8: Proceso de aprendizaje para el ejemplo 1.1

El Perceptrón Multicapa

Minsky y Papert [31] expusieron la debilidad del Perceptrón de una sóla capa, que no podía resolver un sencillo problema XOR, al no ser las dos categorías linealmente separables.

A continuación, se considera la siguiente arquitectura de Perceptrón Multicapa (2 capas):

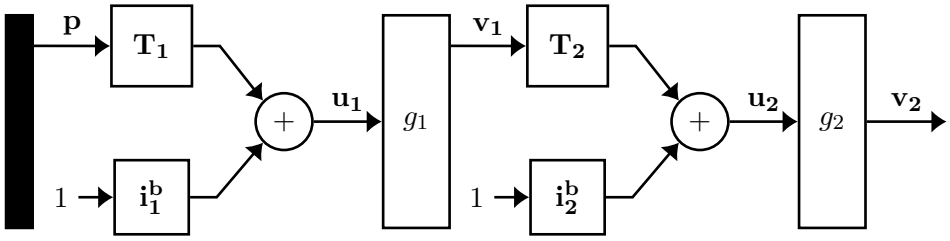


Figura 1.9: Arquitectura del Perceptrón Multicapa (2 capas)

Nótese que la notación utilizada para la construir la arquitectura (figura 1.9) se ha visto modificada ligeramente (con respecto a lo visto en la figura 1.6), indicando ahora las capas mediante el uso de subíndices.

Ejemplo 1.2. *Para ilustrar las limitaciones del Perceptrón de una sóla capa, se considera el clásico ejemplo de la puerta XOR, disyunción exclusiva (ver figura 1.10).*

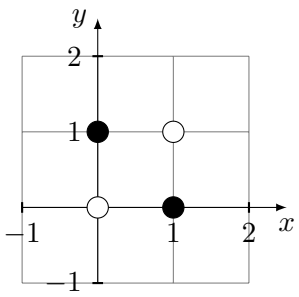


Figura 1.10: Puerta XOR

En este caso, los pares de entradas y salidas correctas para la puerta XOR son:

$$\{\mathbf{p_1} = \left[\begin{smallmatrix}0\\0\end{smallmatrix}\right], t_1 = 0\}, \{\mathbf{p_2} = \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right], t_2 = 1\}, \{\mathbf{p_3} = \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right], t_3 = 1\}, \{\mathbf{p_4} = \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right], t_4 = 0\}$$

Se construye una red de dos capas con dos neuronas en la primera capa y una neurona en la segunda. Eligiendo $\mathbf{T_1} = \begin{bmatrix} 2 & 2 \\ -1 & -1 \end{bmatrix}$, $\mathbf{i_1^b} = \begin{bmatrix} -1 \\ \frac{3}{2} \end{bmatrix}$, $\mathbf{T_2} = \begin{bmatrix} 1 & 1 \end{bmatrix}$, $i_2^b = -\frac{3}{2}$, *se consigue que*

en la primera capa cada neurona (cada fila de $\mathbf{T_1}$ *e* $\mathbf{i_1^b}$*) construya una frontera de decisión que permita clasificar correctamente dos puntos, siendo la segunda capa la que realiza la operación AND entre los resultados de la capa anterior (ver figura 1.11).*

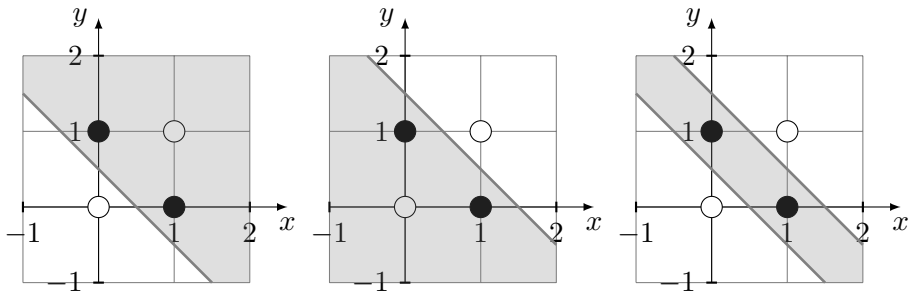


Figura 1.11: Fronteras de decisión del Perceptrón Multicapa para resolver el problema XOR

El algoritmo de retropropagación (*backpropagation*) permite llevar a cabo el aprendizaje en redes como la anterior, comparando de nuevo cada salida de la red con la salida real y calculando en esta ocasión el error cuadrático medio. Mediante el uso de la regla de la cadena, se calculan las derivadas parciales del error con respecto a cada peso y umbral, que servirán para actualizar los pesos y umbrales con nuevos valores.

Este tipo de redes se han utilizando extensamente (y lo siguen haciendo hoy día), para resolver una gran variedad de problemas de clasificación y regresión.

1.2.2 Aprendizaje no supervisado

El aprendizaje no supervisado se distingue del anterior por el hecho de que no hay un conocimiento a priori. Es decir, durante el proceso de aprendizaje, los pesos y umbrales de la red son modificados únicamente en respuesta a las entradas de la red. Una gran parte de estos algoritmos llevan a cabo algún tipo de operación de agrupación de la información (*clustering*).

En este tipo de aprendizaje destacan fundamentalmente dos tipos de redes: las redes de Kohonen (*Mapas Auto-organizados*) y las redes de Hopfield. En adelante, se hará un desarrollo profundo de estas últimas, las redes de Hopfield.

1.3 El modelo de Hopfield

El modelo de Hopfield, también frecuentemente conocido como red de Hopfield, está considerado como uno de los hitos [39] en el renacimiento de las redes neuronales a principios de los años 80 y como una de las redes neuronales más influyentes [13]. John J. Hopfield propuso dos modelos basados en el concepto de memoria asociativa, el modelo discreto [17], y una generalización que permite tomar todos los valores reales en el intervalo $[0, 1]$, el modelo continuo [18].

Se presenta a continuación el modelo discreto del modelo de Hopfield, que permite introducirse más fácilmente en el problema en cuestión, si bien será el modelo continuo el objeto de estudio detallado en esta memoria.

1.3.1 Modelo Discreto

El modelo de Hopfield discreto o red de Hopfield discreta es una red neuronal recurrente consistente en un conjunto de n neuronas totalmente interconectadas (al igual que en el caso continuo, ver figura 1.12) que se caracteriza por comportarse como un sistema de memoria asociativa con unidades binarias.

La activación de la neurona i-ésima en tiempo t es $v_i(t)$ y el peso de la conexión de la neurona j a la i es $T_{i,j}$. Además, el modelo discreto se caracteriza por disponer de una matriz de pesos entre las neuronas simétrica ($T_{i,j} = T_{j,i} \quad \forall i, j \in \{1, \dots n\}$), sin conexiones de cada neurona consigo misma ($T_{i,i} = 0$), pero disponiendo cada neurona de una entrada potencial constante denominada $i_i^b \quad \forall i \in \{1, \dots n\}$.

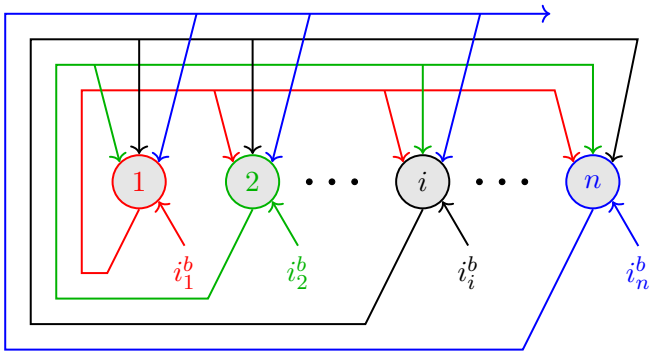


Figura 1.12: Arquitectura de la red de Hopfield (sin conexiones de cada neurona consigo misma, $T_{i,i} = 0$)

Uno de los primeros usos del modelo de Hopfield fue el de memoria asociativa, que permite reconstruir información de un patrón a partir de información parcial de él. Para ello, Hopfield propuso el siguiente proceso de aprendizaje no supervisado:

$$\begin{aligned} \mathbf{T} &= \sum_{s=0}^{k-1} (2\mathbf{q^s} - \mathbf{1})(2\mathbf{q^s} - \mathbf{1})^t - \mathbf{I} \\ \mathbf{i^b} &= \mathbf{0} \end{aligned} \tag{1.1}$$

siendo $\mathbf{q^s}$, con $s = 0, 1, \dots, k-1$, vectores de dimensión n que se desean almacenar, $\mathbf{1}$ y $\mathbf{0}$ los vectores con todas las componentes igual a 1 y 0 respectivamente, e $\mathbf{I}$ la matriz identidad de

tamaño n.

De este modo, el potencial de entrada $u_i(t+1)$ de cada neurona se calcula como:

$$u_i(t+1) = \sum_{j=1}^{n} T_{i,j} v_j(t) + i_i^b$$

calculándose la activación de la neurona i a partir de la función de activación de McCulloch-Pitts:

$$v_i(t+1) = g(u_i(t+1)) = \begin{cases} 0 & \text{si} \quad u_i(t+1) < 0 \\ v_i(t) & \text{si} \quad u_i(t+1) = 0 \\ 1 & \text{si} \quad u_i(t+1) > 0 \end{cases} \qquad \forall i \in \{1, \ldots, n\}$$

En resumidas cuentas, el modelo de Hopfield discreto consiste en un sistema dinámico que a partir de un vector de estados $\mathbf{v}(t)$ calcula el vector de potenciales $\mathbf{u}(t+1)$, obteniendo a continuación el vector de estados necesario en la siguiente etapa. El procedimiento continúa hasta que se alcanza un punto estable, de modo que el vector de estados $\mathbf{v}$ permanece invariante. La figura 1.13 ilustra el sistema dinámico del modelo discreto.

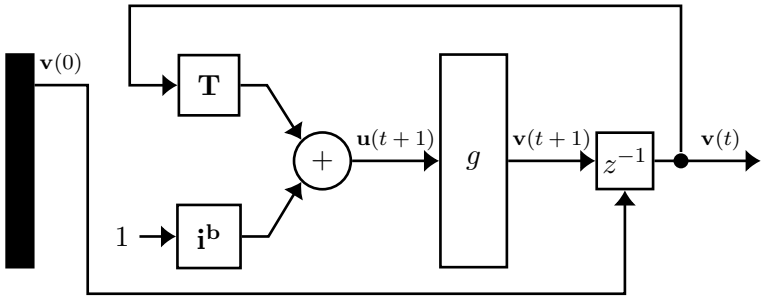
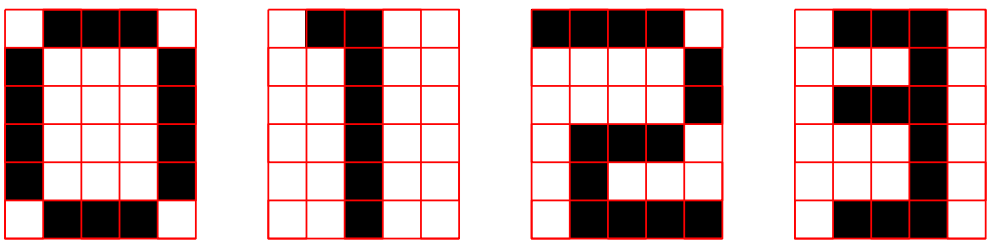


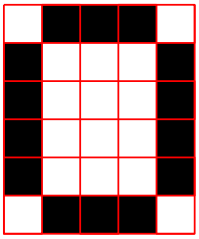
Figura 1.13: Sistema dinámico asociado al modelo de Hopfield discreto

A continuación se describe un ejemplo que ilustra el modelo de Hopfield discreto y que ayudará a enfocar el modelo continuo.

Ejemplo 1.3. *Se plantea una red de Hopfield que debe recordar 4 patrones (vectores), que dispuestos en formato matricial, se corresponden con las imágenes de los números 0, 1, 2 y 3.*



Los patrones en formato vectorial (y que deben ser recordados por la red) se construyen concatenando verticalmente cada una de las columnas de las imágenes anteriores, donde el recuadro escrito (negro) se corresponderá con un uno y el recuadro vacío (blanco) con un cero. De este modo, para la imagen del número 0 *el vector a recordar será:*



$\rightarrow$

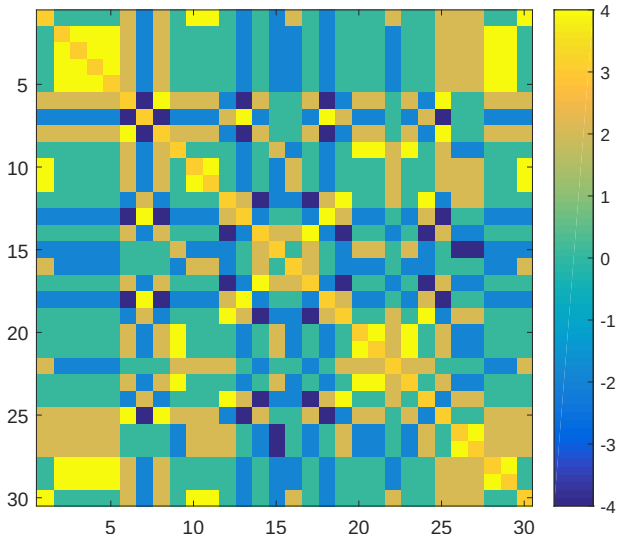
0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
0	1	1	1	0

$\rightarrow$ $\mathbf{q^0}$

Así, dados los cuatro patrones:

$$\mathbf{q^0} = \begin{bmatrix} 0\\1\\1\\1\\1\\0\\1\\0\\0\\0\\0\\0\\1\\1\\0\\0\\0\\0\\1\\1\\0\\0\\0\\0\\0\\1\\0\\1\\1\\1\\1\\0 \end{bmatrix}, \quad \mathbf{q^1} = \begin{bmatrix} 0\\0\\0\\0\\0\\0\\1\\0\\0\\0\\0\\0\\1\\1\\1\\1\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0 \end{bmatrix}, \quad \mathbf{q^2} = \begin{bmatrix} 1\\0\\0\\0\\0\\0\\1\\0\\0\\1\\1\\1\\1\\0\\0\\1\\0\\1\\1\\0\\0\\1\\0\\1\\0\\1\\1\\0\\0\\1 \end{bmatrix}, \quad \mathbf{q^3} = \begin{bmatrix} 0\\0\\0\\0\\0\\0\\0\\1\\0\\1\\0\\0\\1\\1\\0\\0\\1\\0\\0\\1\\1\\1\\1\\1\\1\\0\\0\\0\\0\\0 \end{bmatrix}$$

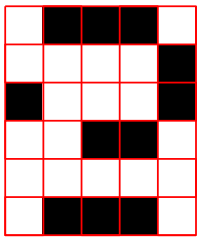
se construye la matriz $\boldsymbol{T}$ *a partir de la ecuación 1.1. El vector* $\mathbf{i^b}$ *se corresponde con el vector columna que tiene todas las coordenadas iguales a cero. Dado el tamaño considerable de la matriz* $\mathbf{T}$ (30×30)*, se muestra a continuación la matriz representada como imagen:*



Se resuelve el modelo discreto del problema asociado utilizando Simulink, que permite

comprender más fácilmente la simulación del modelo dinámico. Nótese la semejanza del modelo en Simulink (ver Apéndice A.3.1) con el diagrama de la figura 1.13, que ilustra el modelo discreto de Hopfield.

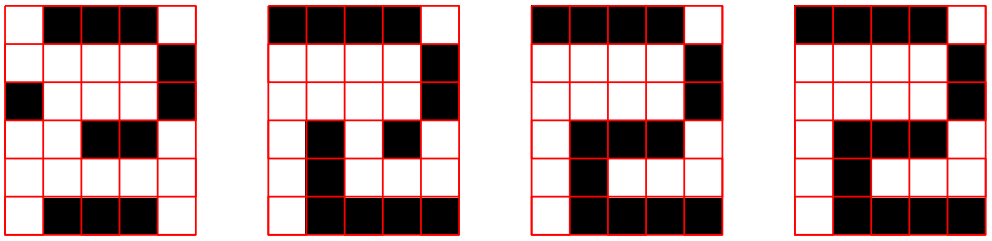
Arrancando la simulación a partir del siguiente patrón o punto inicial $\mathbf{v_0}$*:*



$$\rightarrow \begin{bmatrix} 0\\0\\1\\0\\0\\0\\1\\0\\0\\0\\0\\1\\1\\0\\0\\1\\0\\1\\1\\0\\0\\1\\0\\1\\0\\1\\1\\0\\0\end{bmatrix} = \mathbf{v_0}$$

se obtiene en $t = 2$ *uno de los patrones almacenados por la red, siendo en adelante un punto de equilibrio del sistema dinámico (*$\mathbf{v}(t) = \mathbf{v}(2), \forall t \geq 2$*):*

$t = 0$ $t = 1$ $t = 2$ $t = 3$



$$\mathbf{v}(0) = \begin{bmatrix} 0\\0\\1\\0\\0\\0\\1\\0\\0\\0\\0\\1\\1\\0\\0\\1\\0\\1\\1\\0\\0\\1\\0\\1\\0\\1\\1\\0\\0\end{bmatrix} \quad \mathbf{v}(1) = \begin{bmatrix} 1\\0\\0\\0\\0\\0\\1\\0\\0\\1\\1\\1\\1\\0\\0\\0\\1\\1\\0\\0\\1\\0\\1\\0\\1\\1\\0\\0\end{bmatrix} \quad \mathbf{v}(2) = \begin{bmatrix} 1\\0\\0\\0\\0\\0\\1\\0\\0\\1\\1\\1\\1\\0\\0\\1\\0\\1\\1\\0\\0\\1\\0\\1\\0\\1\\1\\0\\0\end{bmatrix} \quad \mathbf{v}(3) = \begin{bmatrix} 1\\0\\0\\0\\0\\0\\1\\0\\0\\1\\1\\1\\1\\0\\0\\1\\0\\1\\1\\0\\0\\1\\0\\1\\0\\1\\1\\0\\0\end{bmatrix}$$

1.3.2 Modelo Continuo

El modelo de Hopfield continuo o red de Hopfield continua (en inglés *Continuous Hopfield Network*, CHN) es una red neuronal recurrente con una ecuación diferencial asociada (generalización del modelo discreto que permite tomar todos los valores reales en el intervalo $[0, 1]$), cuyos estados evolucionan desde un punto inicial a un punto de equilibrio mediante la minimización de una función de Lyapunov. Si la función de Lyapunov está asociada con la función objetivo del problema de optimización (proceso conocido como proyección o *mapping*), el punto estable o de equilibrio coincide con un óptimo local del problema de optimización. El estudio de la CHN como optimizador de problemas combinatorios, en particular el TSP, es el objeto principal de estudio de esta tesis doctoral.

La arquitectura de red neuronal y sistema dinámico presentados por Hopfield se asemejan al comportamiento de las neuronas del cerebro. En su artículo fundacional [18], Hopfield propone una implementación eléctrica de su modelo a través del siguiente sistema dinámico. La CHN consiste en una red totalmente interconectada (ver figura 1.12) con n unidades (neuronas) evaluadas de modo continuo y una función de activación sigmoidal suave. La dinámica de la CHN se describe a partir de la ecuación diferencial:

$$\frac{d\mathbf{u}}{dt} = -\frac{\mathbf{u}}{\lambda} + \mathbf{T}\mathbf{v} + \mathbf{i}^{\mathbf{b}} \tag{1.2}$$

donde, $\forall i, j \in \{1, \ldots, n\}$:

u_i	es el estado actual de la neurona i
v_i	es la salida de la neurona i
$T_{i,j}$	es el peso de la conexión desde la neurona j a la neurona i
i_i^b	es la entrada externa (*offset bias*) de la neurona i

siendo la función de salida $g(u_i)$ una tangente hiperbólica:

$$v_i = g(u_i) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i}{u_0}\right)\right), \quad u_0 > 0 \tag{1.3}$$

donde el parámetro u_0 determina la pendiente de la función sigmoidal.

La figura 1.14 ilustra el sistema dinámico del modelo continuo.

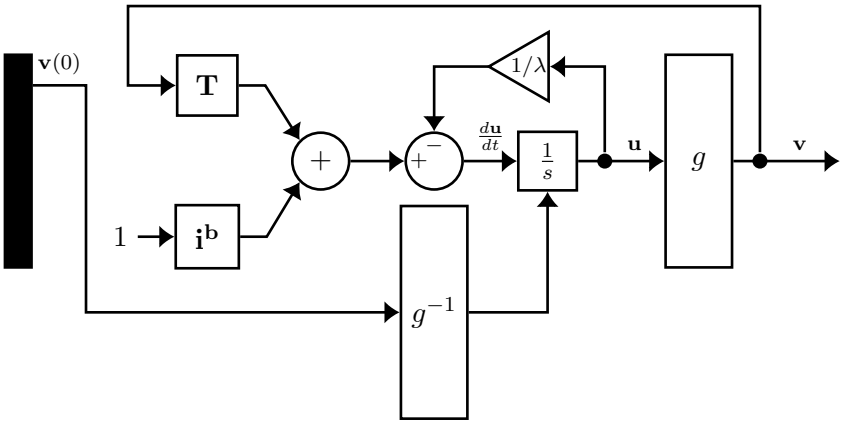


Figura 1.14: Sistema dinámico asociado al modelo de Hopfield continuo

La figura 1.15 ilustra la función de activación tangente hiperbólica $g(x)$ y su correspondiente inversa $g^{-1}(x)$ para distintos valores del parámetro u_0.

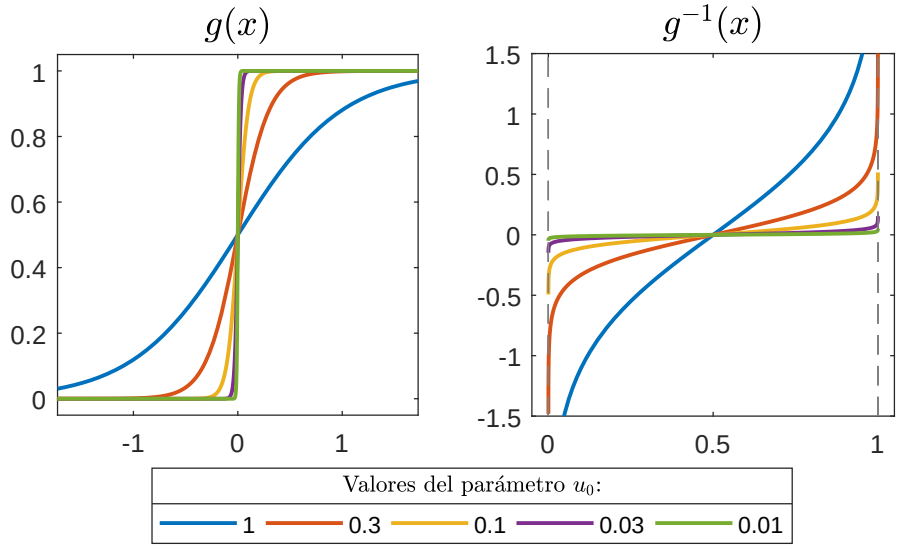


Figura 1.15: Función de activación, $g(x)$, y su inversa, $g^{-1}(x)$, para distintos valores de u_0

Condiciones de estabilidad

Definición 1.1. *Una función de Lyapunov para un sistema dinámico autónomo*

$$f : \mathbb{R}^n \to \mathbb{R}^n$$
$$\frac{d\mathbf{y}}{dt} = f(\mathbf{y})$$

con un punto de equilibrio en $\mathbf{y} = \mathbf{0}$ *es una función escalar* $E : G \subset \mathbb{R}^n \to \mathbb{R}$ *continua, con derivadas continuas, localmente definida positiva y con* $-\nabla E \cdot f$ *también definida positiva.*

El modelo continuo del sistema dinámico representado por la ecuación diferencial 1.2 tiene garantizada la existencia de un *punto de equilibrio* ($\mathbf{u^e}$ tal que $\mathbf{u}(t) = \mathbf{u^e}\ \forall t \geq t_e$ para algún $t_e \geq 0$) si existe una función de energía o de Lyapunov [11]. Tal y como mostró Hopfield [18], si $\mathbf{T}$ es simétrica, entonces existe la siguiente función de Lyapunov:

$$E(\mathbf{v}) = -\frac{1}{2}\mathbf{v}^t\mathbf{T}\mathbf{v} - (\mathbf{i^b})^t\mathbf{v} + \frac{1}{\lambda}\sum_{i=1}^{n}\int_0^{v_i} g^{-1}(x)dx \tag{1.4}$$

Se comprueba a continuación que la función es $E(\mathbf{v})$ es de Lyapunov (definición 1.1), para lo cual se calcula su derivada con respecto al tiempo:

$$\frac{dE}{dt} = -\sum_{i=1}^{n}\Big(\sum_{j=1}^{n} T_{i,j}v_j + i_i^b - \frac{u_i}{\lambda}\Big)\frac{dv_i}{dt} \tag{1.5}$$

El término entre paréntesis en la ecuación 1.5 se corresponde precisamente con $\frac{du_i}{dt}$ (ver

ecuación 1.2), quedando la ecuación anterior de la forma:

$$\frac{dE}{dt} = -\sum_{i=1}^{n} \frac{du_i}{dt}\frac{dv_i}{dt} \tag{1.6}$$

Teniendo en cuenta que $u_i = g^{-1}(v_i)$ (ver ecuación 1.3) y, aplicando la regla de la cadena, se obtiene:

$$\frac{du_i}{dt} = \Big[\frac{d}{dv_i} g^{-1}(v_i)\Big]\frac{dv_i}{dt}$$

que al sustituir en la ecuación 1.6 resulta en:

$$\frac{dE}{dt} = -\sum_{i=1}^{n} \Big[\frac{d}{dv_i} g^{-1}(v_i)\Big]\Big(\frac{dv_i}{dt}\Big)^2 \tag{1.7}$$

Dado que $g(u_i)$ es una función monótona creciente, su inversa $g^{-1}(v_i)$ también lo es. De este modo,

$$\Big[\frac{d}{dv_i} g^{-1}(v_i)\Big] > 0, \quad \forall i \in \{1, \ldots, n\}$$

Se tiene también que:

$$\Big(\frac{dv_i}{dt}\Big)^2 \geq 0, \quad \forall i \in \{1, \ldots, n\}$$

De este modo, todos los factores del sumatorio de la ecuación 1.7 son no negativos, por lo que la derivada de la función de energía con respecto al tiempo decrece siempre con el tiempo:

$$\frac{dE}{dt} \leq 0 \quad \forall t$$

Se demuestra así que la función introducida por Hopfield es de Lyapunov. Además, dado que $\frac{dE}{dt} = 0$ sólo si $\frac{dv_i}{dt} = 0$ puede decirse que:

$$\frac{dE}{dt} < 0 \quad \forall t, \quad \text{excepto en un punto fijo}$$

de modo que la función de energía $E(v)$ de la red de Hopfield es una función monótona decreciente.

Ejemplo 1.4. *Se considera a continuación el ejemplo original con $n = 2$ neuronas propuesto por Hopfield [18], basado en la dinámica de la CHN de la ecuación 1.2, donde se toman los valores:*

$$\mathbf{T} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{i^b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

siendo la función de activación la propuesta por Hopfield [19] y basada en la inversa de la función tangente: $v_i = g(u_i) = \frac{2}{\pi}\tan^{-1}\left(\frac{\pi}{2}u_i\right)$.

Dado que la función de activación puede ser escalada de la forma $v_i = g(\lambda u_i)$, de modo que $u_i = \frac{1}{\lambda}g^{-1}(v_i)$ (el valor $\frac{1}{\lambda}$ corresponde con el factor introducido en el término integral en la ecuación 1.4), se reescribe la función de activación:

$$v_i = g(u_i) = \frac{2}{\pi}\tan^{-1}\left(\frac{\lambda\pi}{2}u_i\right) \tag{1.8}$$

y se elige $\lambda = 1.4$.

Para este ejemplo, la función de Lyapunov (ecuación 1.4) tiene la parte cuadrática como:

$$-\frac{1}{2}\mathbf{v}^t\mathbf{T}\mathbf{v} - (\mathbf{i^b})^t\mathbf{v} = -\frac{1}{2}\begin{bmatrix} v_1 & v_2 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} - \begin{bmatrix} 0 & 0 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = -v_1 v_2$$

A partir de la función de activación (ecuación 1.8), podemos escribir:

$$u_i = g^{-1}(v_i) = \frac{2}{\lambda\pi}\tan\left(\frac{\pi}{2}v_i\right)$$

quedando la parte integral de la función de Lyapunov (ecuación 1.4), donde ya se está teniendo en cuenta el factor $\frac{1}{\lambda}$ *en* $g^{-1}(v_i)$ *para este ejemplo es:*

$$\begin{aligned}\sum_{i=1}^{2}\int_0^{v_i} g^{-1}(x)dx &= \sum_{i=1}^{2}\int_0^{v_i}\frac{2}{\lambda\pi}\tan\left(\frac{\pi}{2}x\right)dx = \frac{2}{\lambda\pi}\sum_{i=1}^{2}\int_0^{v_i}\tan\left(\frac{\pi}{2}x\right)dx \\ &= \frac{2}{\lambda\pi}\sum_{i=1}^{2}\Big[-\frac{2}{\pi}\log\Big[\cos\big(\frac{\pi}{2}x\big)\Big]\Big]_0^{v_i} \\ &= -\frac{4}{\lambda\pi^2}\Big[\log\Big[\cos\big(\frac{\pi}{2}v_1\big)\Big] + \log\Big[\cos\big(\frac{\pi}{2}v_2\big)\Big]\Big]\end{aligned}$$

quedando por tanto la función de energía de la forma:

$$E(\mathbf{v}) = -v_1 v_2 - \frac{4}{\lambda\pi^2}\Big[\log\Big[\cos\big(\frac{\pi}{2}v_1\big)\Big] + \log\Big[\cos\big(\frac{\pi}{2}v_2\big)\Big]\Big] \tag{1.9}$$

De este modo, el sistema dinámico de la CHN (ver ecuación 1.2) para el ejemplo es:

$$\begin{bmatrix}\frac{du_1}{dt} \\ \frac{du_2}{dt}\end{bmatrix} = -\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -u_1 + v_2 \\ -u_2 + v_1 \end{bmatrix} \tag{1.10}$$

siendo las salidas de las neuronas:

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \frac{2}{\pi}tan^{-1}\big(\frac{\lambda\pi}{2}u_1\big) \\ \frac{2}{\pi}tan^{-1}\big(\frac{\lambda\pi}{2}u_2\big) \end{bmatrix}$$

La figura 1.16 muestra la función de energía obtenida para el ejemplo 1.4 así como el plano de fases para el sistema dinámico obtenido (ecuación 1.10). El gráfico muestra también diversas curvas de nivel (las alturas $[-0.041, -0.023, -0.003, 0.017, 0.156, 0.449]$ *originalmente propuestas por Hopfield), así como algunas trayectorias y los puntos críticos del problema: los nodos inestables* ○, *el punto de silla* ● *y los nodos asintóticamente estables* ●.

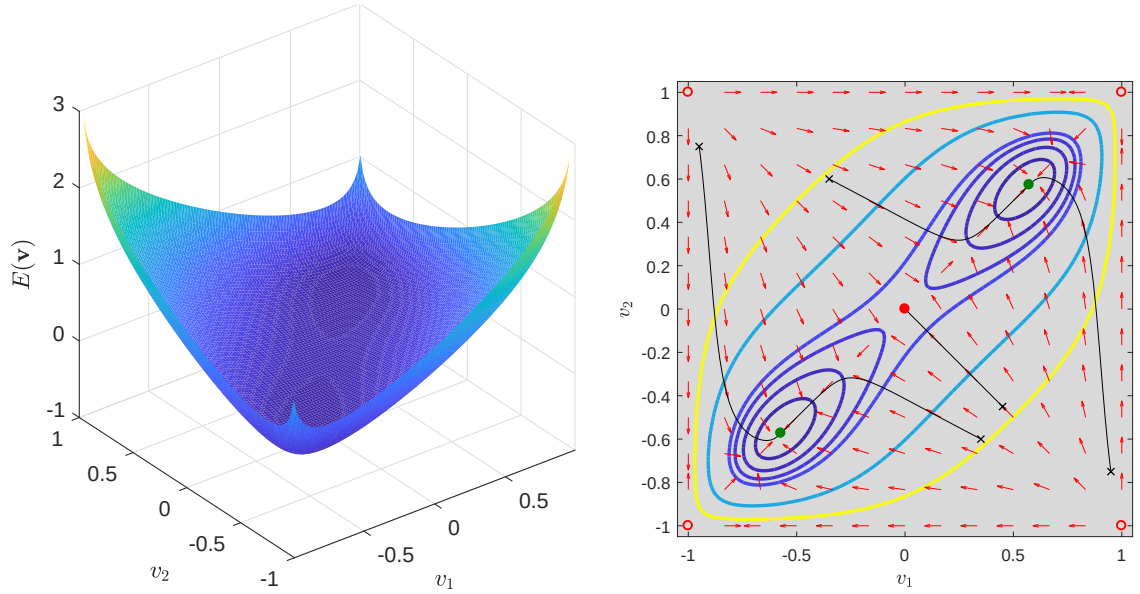


Figura 1.16: Función de energía (ecuación 1.9) y diagrama de fases del sistema dinámico (ecuación 1.10) del ejemplo 1.4

La consecuencia dinámica del término integral $\frac{1}{\lambda}\sum_{i=1}^{n}\int_0^{v_i} g^{-1}(x)dx$ en la ecuación 1.4, es la existencia de puntos de equilibrio estables en el interior del hipercubo de soluciones. La proximidad de estos puntos a los mínimos del término cuadrático $-\frac{1}{2}\mathbf{v}^t\mathbf{T}\mathbf{v} - (\mathbf{i}^\mathbf{b})^t\mathbf{v}$ estará controlada por el parámetro λ. Si $\lambda \to 0$, únicamente existirá un punto de equilibrio en el centro del hibercubo $[0,1]^n$ [55]. Por el contrario, cuando $\lambda \to \infty$, la función de Lyapunov de la red está asociada con la función objetivo a ser minimizada por el problema combinatorio (ver figura 1.17 para el ejemplo 1.4). Por tanto, la CHN resolverá aquellos problemas combinatorios que puedan expresarse como una minimización con restricciones de:

$$E(\mathbf{v}) = -\frac{1}{2}\mathbf{v}^t\mathbf{T}\mathbf{v} - (\mathbf{i}^\mathbf{b})^t\mathbf{v} \tag{1.11}$$

que tiene sus extremos en los vértices del hipercubo n-dimensional $[0,1]^n$. En adelante, se utilizará esta modificación de la función de energía del modelo Hopfield original, propuesta por Abe [1], que no incluye el término integral. Esta cuestión ha sido estudiada en profundidad por Joya et al. [21].

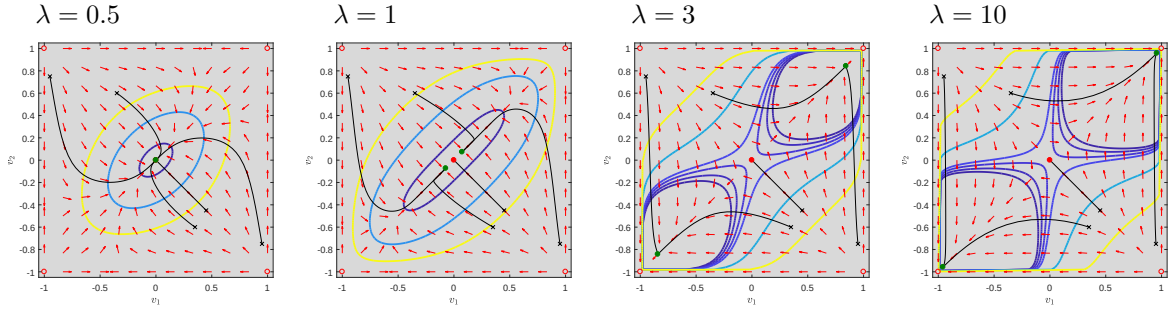


Figura 1.17: Diagramas de fases del sistema dinámico (ecuación 1.10) para el ejemplo 1.4 con distintos valores de λ

1.3.3 El problema del viajante y el modelo de Hopfield

El Problema del Viajante (en inglés, *Traveling Salesman Problem, TSP*) es uno de los problemas más profundamente estudiados en optimización combinatoria. Originalmente formulado en el siglo XIX por W. R. Hamilton y Thomas Kirkman, el problema se centra en un vendedor que tiene que visitar un conjunto finito de ciudades. El TSP busca encontrar el recorrido más corto a través de todas las ciudades de modo que cada ciudad sea visitada sólo una vez y el vendedor regrese al final del recorrido a la ciudad de partida.

El TSP es un problema de optimización *NP*-duro (ver [32] y [59]), esto es, no puede ser resuelto en tiempo polinomial. El problema *P vs. NP*, si todos los *problemas NP* son el realidad *problemas P*, sigue siendo una pregunta abierta. Si no son equivalentes, y ésta es la conjetura actualmente aceptada en el mundo científico, es necesaria una búsqueda exhaustiva para obtener una solución para *problemas NP*. Hoy en día, se utilizan heurísticas para obtener *buenas* soluciones para *problemas NP*, como es el caso del TSP, aunque su optimalidad no está garantizada. Una de estas heurísticas es la CHN.

Tanto la formulación matemática del TSP, como su resolución mediante la CHN se presentan respectivamente en las secciones 2.4 y 2.5. A continuación se proporciona simplemente una introducción histórica al problema.

Desde el trabajo seminal de Hopfield y Tank [19], muchos autores han estudiado el TSP, así como otros problemas de optimización, desde una aproximación neuronal. En su trabajo, la función de energía del TSP requiere dar valores a ciertos parámetros, asignándose de modo experimental: *"an anecdotal exploration of parameter values was used to find a good (but not optimized) operating point"*. A pesar del entusiasmo inicial, se obtuvieron algunos resultados decepcionantes, dado que los puntos de equilibrio encontrados por la CHN no correspondían con soluciones factibles del TSP. Wilson y Pawley [58] llevaron a cabo un estudio detallado del modelo de Hopfield, resaltando que Hopfield y Tank habían sido afortunados en sus resultados dado el número limitado de simulaciones del TSP que habían considerado. Pese a su actitud crítica, manifestaron que no sería prudente descartar el método dado que contenía muchas ideas que deberían ser continuadas.

A partir de este trabajo, los investigadores han adoptado dos estrategias:

1. Buscar una elección adecuada de parámetros que garantice la factibilidad de las soluciones obtenidas

2. Modificar la dinámica original del modelo de Hopfield

Centrándose en la primera línea de investigación, debido a la naturaleza de esta tesis doctoral, Hedge et al. [15] mostraron evidencia empírica de una hipotética interrelación entre los parámetros de la red de Hopfield. Platt y Barr [35] propusieron un método adaptativo que en cada iteración modificaba los términos de la función de energía aplicando gradualmente restricciones a lo largo del tiempo. Cuykendall y Reese [6] encontraron soluciones factibles para problemas del TSP de hasta 165 ciudades, ajustando la entrada $\mathbf{i^b}$ de cada neurona, e indicaron que los comentarios negativos de Wilson y Pawley acerca del trabajo de Hopfield y Tank no estaban justificados. También, Kamgar-Parsi et al. [22] aclararon algunas de

estas discrepancias e investigaron la escalabilidad de las soluciones encontradas por la red de Hopfield a medida que el tamaño del problema aumentaba. todavía sin un método para asignar valores a los parámetros de la red, Aiyer et al. [2] estudiaron el comportamiento de la CHN basándose en los autovalores de la matriz $\mathbf{T}$ y observaron que soluciones espurias pueden ocurrir en cualquier esquina del hipercubo y que la CHN puede con frecuencia converger a soluciones inválidas al aplicarse al TSP. Sin embargo, los parámetros utilizados en sus simulaciones fueron asignados experimentalmente. Abe [1] modificó la función original de la CHN aproximando la función tangente hiperbólica por una función lineal a trozos, introdujo un método para suprimir estados espurios y una función de energía modificada, obteniendo casi siempre soluciones factibles. De nuevo, los valores de los parámetros de la red son elegidos experimentalmente.

No es hasta el trabajo de Talaván y Yáñez [49] que se propone un método (no basado en *prueba y error*) para elegir los valores de los parámetros del modelo de Hopfield. Esta determinación de parámetros (o *parametrización*), que dejaba un parámetro libre relacionado con la *"inhibición global"* de la red, garantizaba que siempre se encontrara una solución factible para el TSP. En sus experiencias computacionales, que incluían la resolución de problemas del TSP de hasta 1000 ciudades, mostraron que valores pequeños (positivos) del parámetro libre C producían mejores resultados, fijándolo por tanto en $C = 0.001$.

A partir de este trabajo, Tan et al. [52] proponen un nuevo ajuste de parámetros utilizando una función de energía modificada, y dejando de nuevo el parámetro relacionado con la *"inhibición global"* de la red como parámetro libre. Tang et al. [53] compararon el modelo de Hopfield con enfoques alternativos como modelos competitivos y, de nuevo, mostraron que eran necesarios valores pequeños del parámetro libre en el modelo de Hopfield para producir mejores resultados. La razón por la que el valor del parámetro libre debía ser pequeño, quedaba como un problema abierto.

Más recientente, Jolai [20] utilizó técnicas de transformación de datos como paso previo a utilizar el modelo de Hopfield para encontrar mejores soluciones. Como en cualquier heurística, únicamente puede garantizarse el encontrar una solución óptima local, en vez de global. La CHN es, siguiendo a Di Marco et al. [8] multiestable, en el sentido que tiene múltiples puntos de equilibrio asintóticamente estables. Cualquier solución que converge a un punto de equilibrio depende en última instancia de las condiciones iniciales. Hernández-Solano et al. [16] desarrollaron un método que permite conservar propiedades favorables de la CHN al llevar a cabo una discretización, mostrando que el interés en el modelo de Hopfield sigue activo.

Otra línea de investigación relacionada con el modelo de Hopfield continuo como optimizador consiste en alterar aleatoriamente la red neuronal para conseguir escapar de los mínimos locales con el objetivo de llegar –si es posible- al mínimo global. Es lo que se conoce como *Chaotic Neural Networks* y se pueden citar en esta línea artículos recientes, Qin [36] Sun et al. [47], que demuestran el interés de conocer en profundidad la CHN, objeto de estudio de esta tesis doctoral.

CAPÍTULO 2

Influencia del punto de arranque del modelo de Hopfield

Uno de los principales inconvenientes a la hora de resolver problemas de optimización combinatoria utilizando el modelo de Hopfield, era la obtención de soluciones no factibles. Una vez resuelto el problema de la factibilidad por Talaván y Yáñez (tanto para resolver el TSP [49] como el GQKP [51]) mediante el uso de parametrizaciones que garanticen la factibilidad de soluciones válidas e infactibilidad de las inválidas, el objetivo es ahora mejorar la calidad de las soluciones que ofrece el modelo de Hopfield. En este sentido, se estudiarán las cuencas de atracción del problema de optimización así como su relación con los citados parámetros [9].

Contenidos del capítulo

2.1 El Problema de Asignación Cuadrática

El problema de asignación cuadrática (en inglés, *Quadratic Assignment Problem*, QAP) fue introducido por Koopmans y Beckmann [25] en 1957, para resolver el problema de asignación de plantas industriales a posibles localizaciones y es uno de los problemas de referencia en optimización e investigación operativa, dada su aplicabilidad a la resolución de problemas reales. Se trata de uno de los problemas de la clase *NP*-duros más difíciles.

El QAP, como cualquier problema de optimización combinatoria, puede resolverse mediante métodos exactos o aproximados. Sin embargo, los métodos exactos, como son el método *Branch and Bound*, el método de *planos de corte*, o la *programación dinámica* son incapaces de resolver el problema para instancias grandes debido a la necesidad de proporcionar una solución en tiempos razonables. Es por ello que se utilizan heurísticas que evitan la necesidad de llevar a cabo una enumeración total, llevando a cabo búsquedas parciales en el espacio de soluciones. Entre estas heurísticas destacan la *Búsqueda Tabú*, los *Algoritmos Genéticos*, el *Temple Simulado*, etc.

El QAP consiste en encontrar un asignación óptima de n instalaciones en n ciudades o localizaciones minimizando el coste de sus flujos. Este coste dependerá de las distancias y flujo entre las localizaciones, además de un coste adicional por instalar cierta instalación en una determinada localización. Así, se busca que el coste, en función de las distancias y flujos, sea mínimo. El QAP puede formularse como un QAP generalizado, más comunmente denominado *Generalized Quadratic Knapsack Problem* (GQKP), del siguiente modo:

$$\text{mín}\left\{\tfrac{1}{2}\sum_{i,j=1}^{n} v_i p_{i,j} v_j + \sum_{i=1}^{n} v_i q_i\right\} \tag{2.1}$$

$$\text{sujeto a}\left\{\begin{array}{ll} \displaystyle\sum_{i=1}^{n} r_{k,i} v_i \leq b_k & k = 1,\ldots,m_1 \\ \displaystyle\sum_{i=1}^{n} r_{k,i} v_i = b_k & k = m_1+1,\ldots,m \\ v_i \in \{0,1\} & i = 1,\ldots,n \end{array}\right\} \tag{2.2}$$

Con el objetivo de resolver el GQKP utilizando el modelo de Hopfield, se busca proyectar el GQKP en su correspondiente CHN, para lo cual las desigualdades anteriores deben convertirse en igualdades. Esto se consigue introduciendo las variables de holgura $v_{n+1},\ldots,v_{n+m_1}$, donde, $\forall k \in 1,\ldots,m_1$, cada v_{n+k} es ponderada por el coeficiente $r_{k,n+k} = b_k - \sum_{j/r_{k,j}<0} r_{k,j}$:

$$\text{mín}\left\{\tfrac{1}{2}\sum_{i,j=1}^{n} v_i p_{i,j} v_j + \sum_{i=1}^{n} v_i q_i\right\}$$

$$\text{sujeto a}\left\{\begin{array}{ll} e_k(\mathbf{v}) \equiv \displaystyle\sum_{i=1}^{n} r_{k,i} v_i + r_{k,n+k} v_{n+k} = b_k & k = 1,\ldots,m_1 \\ e_k(\mathbf{v}) \equiv \displaystyle\sum_{i=1}^{n} r_{k,i} v_i = b_k & k = m_1+1,\ldots,m \\ v_i \in \{0,1\} & i = 1,\ldots,n \\ v_{n+k} \in [0,1] & k = 1,\ldots,m_1 \end{array}\right\}$$

Así, el GQKP puede expresarse matricialmente de la siguiente forma:

$$\min_{\mathbf{v}} \left\{ \frac{1}{2}\mathbf{v}^t\mathbf{P}\mathbf{v} + \mathbf{q}^t\mathbf{v} \right\}$$

$$\text{sujeto a} \left\{ \begin{array}{ll} \mathbf{R}\mathbf{v} = \mathbf{b} & \\ v_i \in \{0,1\} & i = 1,\ldots,n \\ v_{n+k} \in [0,1] & k = 1,\ldots,m_1 \end{array} \right\}$$

De este modo, un GQKP queda caracterizado por sus parámetros: $(n, m, m_1, \mathbf{P}, \mathbf{q}, \mathbf{R}, \mathbf{b})$. Con el objetivo de proyectar el GQKP sobre la CHN, se definen los siguientes conjuntos:

- El hipercubo de Hamming, $H \equiv \{\mathbf{v} \in [0,1]^{(n+m_1)}\}$.
- El subconjunto, $H_C \equiv \{\mathbf{v} \in H : v_i \in \{0,1\} \quad i = 1,\ldots,n\}$, donde únicamente se tienen en cuenta las variables originales.
- Las soluciones factibles,

$$\begin{aligned} H_F &\equiv \{\mathbf{v} \in H_C : \mathbf{R}\mathbf{v} = \mathbf{b}\} \\ &= \left\{ \begin{array}{ll} e_k(\mathbf{v}) \equiv \sum_{i=1}^{n} r_{k,i}v_i + r_{k,n+k}v_{n+k} = b_k & k = 1,\ldots,m_1 \\ e_k(\mathbf{v}) = \sum_{i=1}^{n} r_{k,i}v_i = b_k & k = m_1+1,\ldots,m \\ v_i \in \{0,1\} & i = 1,\ldots,n \\ v_{n+k} \in [0,1] & k = 1,\ldots,m_1 \end{array} \right\} \end{aligned} \tag{2.3}$$

2.2 El modelo de Hopfield aplicado al GQKP

Se propone estudiar la proyección del siguiente problema de optimización sencillo en la CHN:

$$\begin{array}{c} \min \left\{ \frac{1}{2}(4\,{v_1}^2 - 2\,{v_2}^2) \right\} \\ \text{sujeto a} \;\; v_1 + v_2 = 1 \end{array} \tag{2.4}$$

que puede expresarse como un problema GQKP con los siguientes valores:

$$\mathbf{P} = \begin{bmatrix} 4 & 0 \\ 0 & -2 \end{bmatrix}, \; \mathbf{q} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \; \mathbf{R} = \begin{bmatrix} 1 & 1 \end{bmatrix}, \; \mathbf{b} = \begin{bmatrix} 1 \end{bmatrix} \tag{2.5}$$

Nótese que no son necesarias las variables de holgura en este ejemplo.

El conjunto de soluciones factibles para la ecuación 2.4 queda descrito por:

$$H_F \equiv \left\{ \begin{array}{l} e_1(\mathbf{v}) = v_1 + v_2 = 1 \\ v_1, \; v_2 \in \{0,1\} \end{array} \right\}$$

que tiene dos soluciones $\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$, siendo $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ la solución óptima.

Siguiendo el procedimiento introducido por Talaván y Yáñez [51], el problema de optimización $\min_{\mathbf{v}\in H} E(\mathbf{v})$ se considera la *proyección* del GQKP si

$$E(\mathbf{v}) = E^O(\mathbf{v}) + E^R(\mathbf{v}), \quad \forall \mathbf{v} \in H$$

verifica

- $E^O(\mathbf{v}) = \alpha(\frac{1}{2}\mathbf{v}^t\mathbf{P}\mathbf{v} + \mathbf{q}^t\mathbf{v})$, siendo directamente proporcional a la función objetivo
- $E^R(\mathbf{v})$ es una función cuadrática que penaliza las restricciones violadas por el problema y garantiza la factibilidad de la solución de la CHN

Utilizando el término para la función de energía propuesto por Talaván y Yáñez [51]:

$$E^R(\mathbf{v}) = \frac{1}{2}\left(\mathbf{R}\mathbf{v}\right)^t \boldsymbol{\Phi}\left(\mathbf{R}\mathbf{v}\right) + \mathbf{v}^t diag\left(\boldsymbol{\gamma}\right)\left(\mathbf{1}-\mathbf{v}\right) + \boldsymbol{\beta}^t\mathbf{R}\mathbf{v}$$

y los valores para el GQKP de la ecuación 2.5, la función de energía para el GQKP proyectado es:

$$\begin{aligned} E\left(\mathbf{v}\right) &= E^0(\mathbf{v}) + E^R(\mathbf{v}) \\ &= \alpha\left(\tfrac{1}{2}\mathbf{v}^t\mathbf{P}\mathbf{v} + \mathbf{q}^t\mathbf{v}\right) + \tfrac{1}{2}\left(\mathbf{R}\mathbf{v}\right)^t\boldsymbol{\Phi}\left(\mathbf{R}\mathbf{v}\right) + \mathbf{v}^t diag\left(\boldsymbol{\gamma}\right)\left(\mathbf{1}-\mathbf{v}\right) + \boldsymbol{\beta}^t\mathbf{R}\mathbf{v} \\ &= \alpha\left(2\,{v_1}^2 - {v_2}^2\right) + \tfrac{1}{2}\phi_{1,1}\left(v_1+v_2\right)^2 - \gamma_1 v_1\left(v_1-1\right) - \gamma_2 v_2\left(v_2-1\right) \\ &\quad + \beta_1\left(v_1+v_2\right) \end{aligned} \tag{2.6}$$

Comparando la función de energía obtenida para el GQKP proyectado (ecuación 2.6) con la función de energía de la CHN $E(\mathbf{v}) = \frac{1}{2}\mathbf{v}^t\mathbf{T}\mathbf{v} - (\mathbf{i}^b)^t\mathbf{v}$ (ecuación 1.11), se obtienen los siguientes valores para $\mathbf{T}$ e $\mathbf{i}^b$:

$$\mathbf{T} = -\left(\alpha\mathbf{P} + \mathbf{R}^t\boldsymbol{\Phi}\mathbf{R} - 2diag\left(\boldsymbol{\gamma}\right)\right) = -\alpha\begin{bmatrix} 4 & 0 \\ 0 & -2 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix}\phi_{1,1}\begin{bmatrix} 1 & 1 \end{bmatrix} + 2\begin{bmatrix} \gamma_1 & 0 \\ 0 & \gamma_2 \end{bmatrix}$$

$$= \begin{bmatrix} -4\alpha - \phi_{1,1} + 2\gamma_1 & -\phi_{1,1} \\ -\phi_{1,1} & 2\alpha - \phi_{1,1} + 2\gamma_2 \end{bmatrix}$$

$$\mathbf{i}^b = -\left(\alpha\mathbf{q} + \mathbf{R}^t\boldsymbol{\beta} + \boldsymbol{\gamma}\right) = -\alpha\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\beta_1 - \begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix} = \begin{bmatrix} -\beta_1 - \gamma_1 \\ -\beta_1 - \gamma_2 \end{bmatrix}$$

2.2.1 Determinación de parámetros para el ejemplo del GQKP

Una vez determinada la función de energía, los parámetros $(\boldsymbol{\Phi}, \boldsymbol{\gamma}, \boldsymbol{\beta})$ deben ser elegidos de modo que cada mínimo local de la función de energía se corresponda con una solución factible del problema de optimización.

Análisis de estabilidad para el ejemplo del GQKP proyectado

A continuación se lleva a cabo un análisis de las soluciones válidas, buscando aquellos parámetros que garanticen que la CHN es estable en las soluciones factibles. De nuevo,

recordando a Talaván y Yáñez [51], la estabilidad de cualquier $\mathbf{v} \in H$ está garantizada si:

$$\underline{E}^0(\mathbf{v}) \geq 0 \tag{2.7}$$

$$\overline{E}^1(\mathbf{v}) \leq 0 \tag{2.8}$$

$$E_{n+k}(\mathbf{v}) = 0 \quad \forall k \in \{1, \ldots, m_1\} \ / \ v_{n+k} \in (0,1) \quad \forall \mathbf{v} \in H_F \tag{2.9}$$

donde

$$\begin{aligned} E_i(\mathbf{v}) &\equiv \frac{\partial E(\mathbf{v})}{\partial v_i} \\ \underline{E}^0(\mathbf{v}) &\equiv \min_{v_i=0} E_i(\mathbf{v}) \\ \overline{E}^1(\mathbf{v}) &\equiv \max_{v_i=1} E_i(\mathbf{v}) \end{aligned}$$

Pese a que realizar un análisis de todas las soluciones factibles sería una tarea muy costosa para un GQKP arbitrario, este ejemplo sencillo permite realizar un estudio riguroso. De este modo, las derivadas parciales de la función de energía serán:

$$\begin{aligned} E_1(\mathbf{v}) &= 4\alpha v_1 + \phi_{1,1} v_1 + \phi_{1,1} v_2 - 2\gamma_1 v_1 + \gamma_1 + \beta_1 \\ E_2(\mathbf{v}) &= -2\alpha v_2 + \phi_{1,1} v_1 + \phi_{1,1} v_2 - 2\gamma_2 v_2 + \gamma_2 + \beta_1 \end{aligned}$$

Se observa que se satisface la condición 2.7 si:

$$\begin{aligned} \underline{E}^0([\begin{smallmatrix}1\\0\end{smallmatrix}]) &= \min\{E_2([\begin{smallmatrix}1\\0\end{smallmatrix}])\} = \phi_{1,1} + \gamma_2 + \beta_1 \geq 0 \\ \underline{E}^0([\begin{smallmatrix}0\\1\end{smallmatrix}]) &= \min\{E_1([\begin{smallmatrix}0\\1\end{smallmatrix}])\} = \phi_{1,1} + \gamma_1 + \beta_1 \geq 0 \end{aligned}$$

y la condición 2.8 se verifica si:

$$\begin{aligned} \overline{E}^1([\begin{smallmatrix}1\\0\end{smallmatrix}]) &= \max\{E_1([\begin{smallmatrix}1\\0\end{smallmatrix}])\} = 4\alpha + \phi_{1,1} - \gamma_1 + \beta_1 \leq 0 \\ \overline{E}^1([\begin{smallmatrix}0\\1\end{smallmatrix}]) &= \max\{E_2([\begin{smallmatrix}0\\1\end{smallmatrix}])\} = -2\alpha + \phi_{1,1} - \gamma_2 + \beta_1 \leq 0 \end{aligned}$$

Este ejemplo no requiere utilizar la condición 2.9 dado que no usa variables de holgura.

La inestabilidad de cualquier punto interior $\mathbf{v} \in H \setminus H_C$ se garantiza si:

$$T_{i,i} \geq 0 \quad \forall i \in \{1, \ldots, n\}$$

y la estabilidad de las soluciones válidas se consigue si $\mathbf{\Phi}$ es semi-definida positiva:

$$\phi_{k,l} \geq 0 \quad \forall k, l \in \{1, \ldots, m\}$$

La inestabilidad de los vértices no factibles se obtiene creando una partición del conjunto $H_C \setminus H_F$ e imponiendo las condiciones de inestabilidad a cada uno de los elementos de la

partición:

$$\vee \begin{cases} \underline{E}^0(\mathbf{v}) & \leq & -\varepsilon \\ \overline{E}^1(\mathbf{v}) & \geq & \varepsilon \\ E_{n+k}(\mathbf{v}) & \neq & 0 \quad \text{para algún } v_{n+k} \in (0,1) \end{cases} \qquad \text{con } \varepsilon > 0$$

A continuación se construye la partición $H_C \backslash H_F$, mediante el *método directo* desarrollado por Talaván [48]. En resumen, se construye la partición considerando los distintos casos que pueden ocurrir cuando las restricciones para H_F son violadas. Por tanto, para un $\mathbf{v} \in H_C \setminus H_F$, se buscará la inestabilidad a partir de la primera restricción que no se verifique, distinguiendo las inecuaciones de las ecuaciones (que pueden no verificarse por exceso o por defecto):

$$H_C \setminus H_F = \bigcup_{k=1}^{m_1} W_{k,0} \bigcup_{k=1}^{m} W_{k,1} \bigcup_{k=1}^{m} W_{k,2}$$

donde,

$$W_{k,0} \equiv \bigcap_{l=1}^{k-1} \{e_l(\mathbf{v}) = b_l\} \cap \{e_k(\mathbf{v}) > b_k\} \cap \{e_k(\mathbf{v}) - r_{k,n+k}\ v_{n+k} \leq b_k\} \qquad \forall k = 1, \ldots, m_1$$

$$W_{k,1} \equiv \bigcap_{l=1}^{k-1} \{e_l(\mathbf{v}) = b_l\} \cap \{e_k(\mathbf{v}) > b_k\} \cap \{e_k(\mathbf{v}) - r_{k,n+k}\ v_{n+k} > b_k\} \qquad \forall k = 1, \ldots, m_1$$

$$W_{k,1} \equiv \bigcap_{l=1}^{k-1} \{e_l(\mathbf{v}) = b_l\} \cap \{e_k(\mathbf{v}) > b_k\} \qquad \forall k = m_1 + 1, \ldots, m$$

$$W_{k,2} \equiv \bigcap_{l=1}^{k-1} \{e_l(\mathbf{v}) = b_l\} \cap \{e_k(\mathbf{v}) < b_k\} \qquad \forall k = 1, \ldots, m$$

Para el GQKP estudiado en esta sección, la partición queda reducida a:

$$H_C - H_F = W_{1,1} \cup W_{1,2}$$

donde

- $W_{1,1} = \{e_1(\mathbf{v}) > 1\} = \{v_1 + v_2 > 1\}$
  que se verifica si $v_1 = 1$ y $v_2 = 1$. Por tanto:

$$\begin{cases} E_1(\mathbf{v}) > & 4\alpha + 2\phi_{1,1} - \gamma_1 + \beta_1 \\ E_2(\mathbf{v}) > & -2\alpha + 2\phi_{1,1} - \gamma_2 + \beta_1 \end{cases}$$

  garantizándose la inestabilidad si:

$$\vee \begin{cases} 4\alpha + 2\phi_{1,1} - \gamma_1 + \beta_1 \geq \varepsilon \\ -2\alpha + 2\phi_{1,1} - \gamma_2 + \beta_1 \geq \varepsilon \end{cases}$$

- $W_{1,2} = \{e_1(\mathbf{v}) < 1\} = \{v_1 + v_2 < 1\}$

que se verifica si $v_1 = 0$ y $v_2 = 0$. Por tanto:

$$\begin{cases} E_1(\mathbf{v}) \leq \gamma_1 + \beta_1 \\ E_2(\mathbf{v}) \leq \gamma_2 + \beta_1 \end{cases}$$

garantizándose la inestabilidad si:

$$\vee \begin{cases} \gamma_1 + \beta_1 \leq -\varepsilon \\ \gamma_2 + \beta_1 \leq -\varepsilon \end{cases}$$

Teniendo en cuenta las condiciones iniciales y eligiendo de entre las desigualdades obtenidas anteriormente, se obtiene el siguiente conjunto de desigualdades lineales que garantizan la estabilidad de las soluciones factibles e inestabilidad de las no factibles:

$$\begin{cases} T_{1,1} = & -4\alpha - \phi_{1,1} + 2\gamma_1 \geq & 0 \\ T_{2,2} = & 2\alpha - \phi_{1,1} + 2\gamma_2 \geq & 0 \\ & \phi_{1,1} \geq & 0 \\ W_{1,1}: & 4\alpha + 2\phi_{1,1} - \gamma_1 + \beta_1 \geq & \varepsilon \\ W_{1,2}: & \gamma_2 + \beta_1 \leq & -\varepsilon \end{cases}$$

Resolviendo el sistema de inecuaciones lineales anterior, se obtiene la siguiente posible parametrización, que deja α y $\phi_{1,1}$ como parámetros libres:

$$\gamma_1 = 2\,\alpha + \frac{\phi_{1,1}}{2}, \quad \gamma_2 = \frac{\phi_{1,1}}{2} - \alpha, \quad \beta_1 = -\frac{\alpha}{2} - \phi_{1,1}, \quad \varepsilon = \frac{3\,\alpha}{2} + \frac{\phi_{1,1}}{2} \tag{2.10}$$

2.3 Cuencas de atracción y análisis del punto de silla de la CHN aplicado al GQKP

Utilizando la parametrización obtenida en la sección 2.2.1 y la función de energía de la ecuación 2.6, la función de energía del GQKP para el ejemplo sencillo queda de la forma:

$$E^O(\mathbf{v}) = \alpha\big(2v_1^2 - v_2^2\big),\ E^R(\mathbf{v}) = \alpha\big(-2v_1^2 + v_2^2 + \frac{3}{2}(v_1 - v_2)\big) - \phi_{1,1}\big(\frac{1}{2}v_1 + \frac{1}{2}v_2 - v_1v_2\big)$$

$$E\,(\mathbf{v}) = E^O(\mathbf{v}) + E^R(\mathbf{v}) = \frac{3}{2}\alpha\,(v_1 - v_2) - \frac{1}{2}\phi_{1,1}\,(v_1 + v_2 - 2v_1v_2)$$

siendo las derivadas parciales de $E(\mathbf{v})$:

$$\begin{aligned} E_1(\mathbf{v}) = & \quad \frac{3\alpha}{2} - \frac{\phi_{1,1}}{2} + v_2\ \phi_{1,1} \\ E_2(\mathbf{v}) = & \quad v_1\phi_{1,1} - \frac{\phi_{1,1}}{2} - \frac{3\alpha}{2} \end{aligned}$$

Haciendo las derivadas iguales a cero se obtiene el punto de silla $\mathbf{v}^*$ de $E(\mathbf{v})$ en términos

de α y $\phi_{1,1}$:

$$(\mathbf{v}^*)^t = [v_1^*, v_2^*] = \left[\frac{1}{2}\left(1 + \frac{3\alpha}{\phi_{1,1}}\right),\ \frac{1}{2}\left(1 - \frac{3\alpha}{\phi_{1,1}}\right)\right]$$

Eligiendo un valor de $\phi_{1,1}$ que verifique que $\phi_{1,1} \leq 3\alpha$, se garantiza que el punto de silla se mueve fuera del hipercubo de Hamming, dejando todo el hipercubo dentro de la cuenca de atracción de la solución óptima, $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$. La figura 2.1 muestra la función de energía y su punto de silla para distintos valores de $\phi_{1,1}$ ($\alpha = 1$). Se muestra como al elegir valores de ϕ menores que 3α para la función de energía, se obtiene una única cuenca de atracción dentro del hipercubo de Hamming que lleva a la solución óptima.

Más formalmente, se observa que para un valor fijo de α, el límite del punto de silla cuando $\phi_{1,1}$ tiende a ∞ es:

$$\lim_{\phi_{1,1} \to \infty} (\mathbf{v}^*)^t = \left[\frac{1}{2}, \frac{1}{2}\right]$$

y queda fuera del hipercubo de Hamming cuando $\phi_{1,1}$ tiende a 0:

$$\lim_{\phi_{1,1} \to 0} (\mathbf{v}^*)^t = [\infty, -\infty]$$

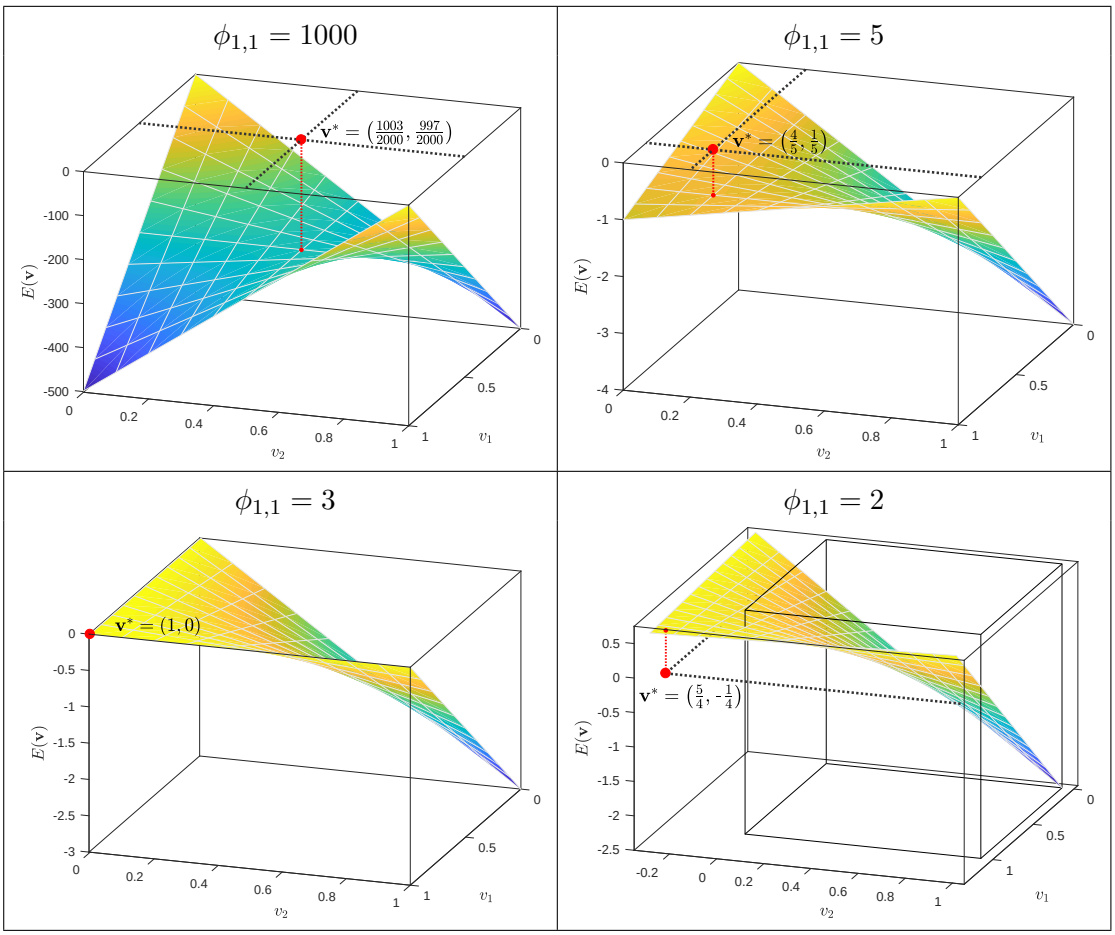


Figura 2.1: Función de energía para el ejemplo sencillo del GQKP con distintos valores de $\phi_{1,1}$

2.3.1 Resolución del ejemplo del GQKP utilizando el simulador de la CHN

Los resultados teóricos obtenidos en la sección 2.3 se han validado también empíricamente mediante un simulador del modelo de Hopfield (ver apéndice A) programado utilizando MATLAB (The MathWorks Inc., Natick, MA, USA).

La figura 2.2 muestra las cuencas de atracción para las funciones de energía mostradas en la figura 2.1, incluyendo algunos casos intermedios. Las cuencas se han obtenido mediante la resolución de múltiples CHNs utilizando diferentes puntos iniciales (muestreando el hipercubo de Hamming con un array 400×400), que se han coloreado de acuerdo con la solución a la que el correspondiente CHN converge. El método de resolución del sistema dinámico ha sido el *método de Runge-Kutta* (ver sección 5.2.2), utilizando la función de activación de la ecuación 1.3.

La cuenca oscura (azul) representa la solución óptima, $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$. Sobreimpreso, se ha graficado el plano de fases del problema junto con las curvas de nivel de la función de energía, así como algunas de las trayectorias que siguen determinados puntos iniciales durante la simulación. Los puntos coloreados en las gráficas representan los puntos de equilibrio de la CHN: las dos soluciones $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ y $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$, el punto de silla $\left[\begin{smallmatrix}v_1^*\\v_2^*\end{smallmatrix}\right]$ y los vértices $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$ y $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$, donde la CHN queda atrapada si se utilizan como puntos de arranque. Nótese como la cuenca de atracción para la mejor solución crece a medida que el parámetro libre decrece.

Eligiendo el valor adecuado para el parámetro libre ($\phi_{1,1} \leq 3\alpha$) se garantiza que siempre

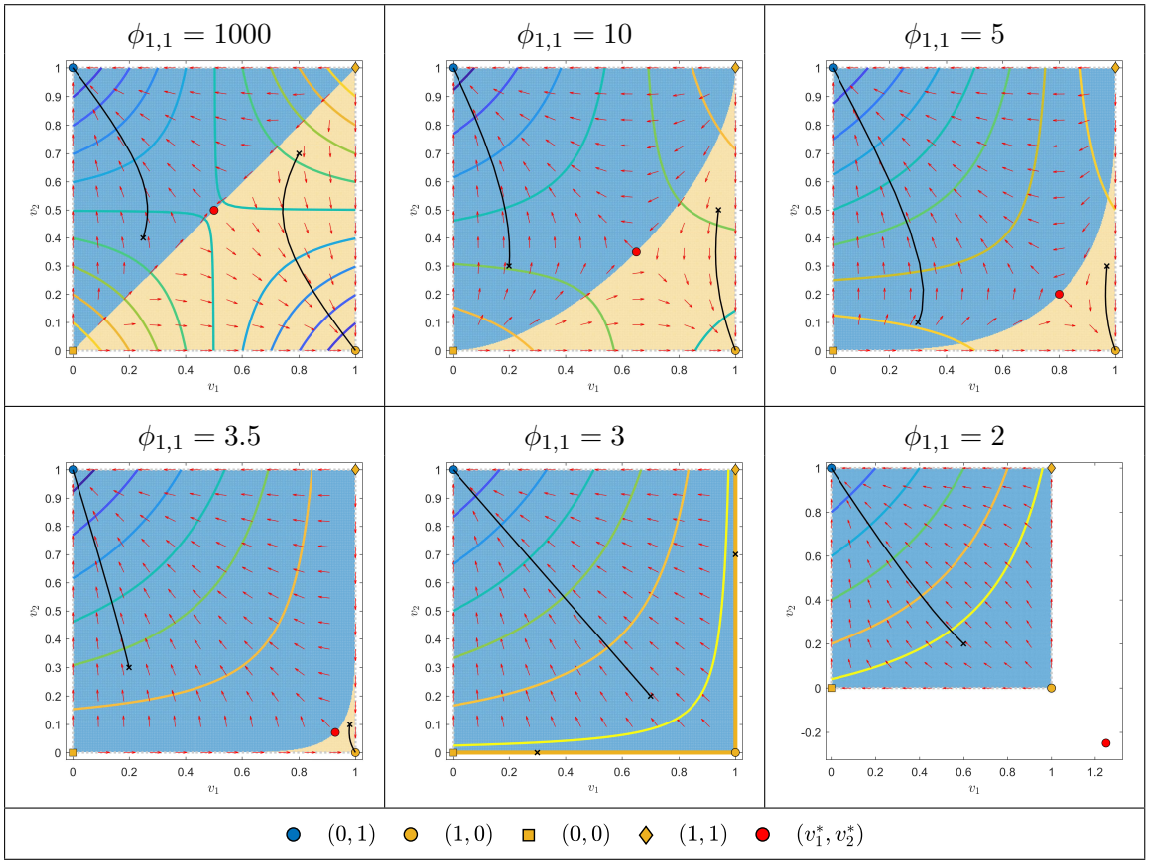


Figura 2.2: Cuencas de atracción de las funciones de energía consideradas para el ejemplo del GQKP utilizando la función de activación tangente hiperbólica de la ecuación 1.3

se obtiene la solución óptima con independencia del punto inicial.

Llevando a cabo el mismo proceso, pero utilizando en vez de la tangente hiperbólica de la ecuación 1.3 la siguiente función de activación

$$v_i = g(u_i) = \begin{cases} 0 & \text{si} \quad u_i \leq -u_0 \\ \frac{1}{2}(1 + \frac{u_i}{u_0}) & \text{si} \quad -u_0 < u_i < u_0 \\ 1 & \text{si} \quad u_i \geq u_0 \end{cases} \qquad \forall i \in \{1, \ldots, n\} \tag{2.11}$$

se obtienen los resultados de la figura 2.3. Se aprecia cómo en esta ocasión las fronteras delimitadas por los puntos iniciales que convergen a una u otra solución son rectilíneas, lo cual ayudará a la convergencia del modelo. Además, esta función de activación (ecuación 2.11) es mucho más simple y, desde un punto de vista computacional, más rápida que la tangente hiperbólica.

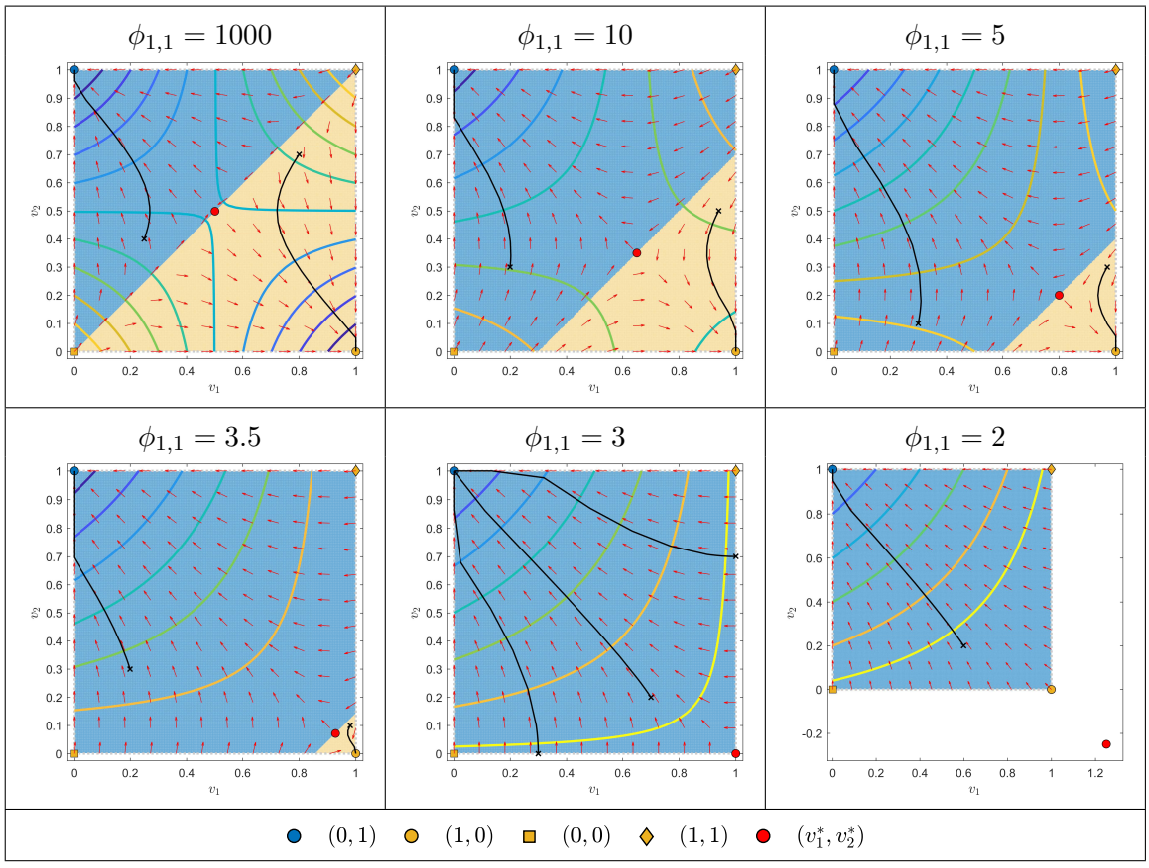


Figura 2.3: Cuencas de atracción de las funciones de energía consideradas para el ejemplo del GQKP utilizando la función de activación linea de la ecuación 2.11

2.4 El Problema del Viajante

Existen muchas formulaciones posibles para caracterizar el TSP, cuya introducción histórica se abordó en la sección 1.3.3. En adelante, se utilizará la siguiente formulación:

Sea N el número de ciudades y sea d_{xy} la distancia entre las ciudades x e y, $x, y \in \{1, 2, \ldots, N\}$. Además, sea $\mathbf{V}$ la matriz $N \times N$ de variable de estados:

$$v_{x,i} = \begin{cases} 1 & \text{si la ciudad } x \text{ se visita en la posición } i \\ 0 & \text{en otro caso} \end{cases} \qquad x, i \in \{1, \ldots, N\} \tag{2.12}$$

A menudo se referirá a la matriz $N \times N$ de variable de estados $\mathbf{V}$ como vector, $\mathbf{v}$, de tamaño $N^2 \times 1$ (ver notación, página XXI). Esta variable de estados $\mathbf{v}$ identifica un tour válido del TSP si se verifican las siguientes restricciones:

- Cada ciudad sólo se visita una vez:

$$S_x = \sum_{i=1}^{N} v_{x,i} = 1 \qquad \forall x \in \{1, 2, \ldots, N\} \tag{2.13}$$

- En cada posición del tour sólo se visita una ciudad:

$$S_i = \sum_{x=1}^{N} v_{x,i} = 1 \qquad \forall i \in \{1, 2, \ldots, N\} \tag{2.14}$$

La función objetivo es:

$$\text{mín} \left\{ \frac{1}{2} \sum_{x=1}^{N} \sum_{y \neq x} \sum_{i=1}^{N} d_{xy} v_{xi} (v_{y(i+1)} + v_{y(i-1)}) \right\} \tag{2.15}$$

(los subíndices $i+1$ e $i-1$ vienen dados módulo N)

El conjunto de soluciones factibles, a partir de lo visto en la ecuación 2.3, será:

$$H_F \equiv \left\{ \mathbf{v} \in H_C / \sum_{i=1}^{N} v_{x,i} = 1 \wedge \sum_{x=1}^{N} v_{x,i} = 1, \; x, i \in \{1, \ldots, N\} \right\}$$

2.5 El modelo de Hopfield aplicado al TSP

Con el objetivo de resolver el problema del viajante introducido en la sección 2.4, se propone una red neuronal recurrente con $N \times N$ neuronas, identificada cada una de ellas con el par (x, i) que identifica la ciudad y el orden en que se visita. El estado del modelo CHN se identifica por el conjunto de valores de cada una de las neuronas, siendo el hipercubo de Hamming es $H \equiv \{\mathbf{v} \in [0, 1]^{N \times N}\}$. La función de energía de la CHN propuesta por Hopfield

y Tank [19] para cualquier estado $\mathbf{v} \in H$ es:

$$\begin{aligned} E(\mathbf{v}) = \; & \frac{A}{2}\sum_{x}^{N}\sum_{i}^{N}\sum_{j\neq i}^{N} v_{x,i}v_{x,j} + \frac{B}{2}\sum_{i}^{N}\sum_{x}^{N}\sum_{y\neq x}^{N} v_{x,i}v_{y,i} + \frac{C}{2}\Big(\sum_{x}^{N}\sum_{i}^{N} v_{x,i} - N\Big)^2 \\ & + \frac{D}{2}\sum_{x}^{N}\sum_{y\neq x}^{N}\sum_{i}^{N} d_{x,y}v_{x,i}(v_{y,i-1} + v_{y,i+1}) \end{aligned} \tag{2.16}$$

(los subíndices $i+1$ e $i-1$ vienen dados módulo N)

Analizando los cuatro términos:

- $\sum_{x}^{N}\sum_{i}^{N}\sum_{j\neq i}^{N} v_{x,i}v_{x,j}$ se minimiza y toma el valor 0 cuando hay a lo sumo un uno por fila.
- $\sum_{i}^{N}\sum_{x}^{N}\sum_{y\neq x}^{N} v_{x,i}v_{y,i}$ se minimiza y toma el valor 0 cuando hay a lo sumo un uno por columna.
- $\Big(\sum_{x}^{N}\sum_{i}^{N} v_{x,i} - N\Big)^2$ se minimiza cuando hay exactamente N unos. Evita soluciones con ceros en algunas filas o columnas.
- $\sum_{x}^{N}\sum_{y\neq x}^{N}\sum_{i}^{N} d_{x,y}v_{x,i}(v_{y,i-1} + v_{y,i+1})$ identifica el valor la función objetivo.

Además, desarrollando los cuatro términos de la función de energía 2.16 y comparándolos con la forma general de la función de energía de la red de Hopfield (ecuación 1.11), se obtiene:

$$E(\mathbf{v}) = -\frac{1}{2}\sum_{x,i}\sum_{y,j} v_{x,i}T_{xi,yj}v_{y,j} - \sum_{x,i} i^b_{x,i}v_{x,i} \tag{2.17}$$

con

$$\begin{aligned} T_{xi,yj} &= -\left(A\delta_{x,y}(1-\delta_{i,j}) + B(1-\delta_{x,y})\delta_{i,j} + C + D(\delta_{i,j-1} + \delta_{i,j+1})d_{x,y}\right) \\ i^b_{x,i} &= CN \end{aligned} \tag{2.18}$$

donde $\delta_{i,j} = \begin{cases} 1 & \text{si} \quad i = j \\ 0 & \text{si} \quad i \neq j \end{cases}$ es una δ de Kronecker, $\quad x, y \in \{1,\ldots,N\}$, $i, j \in \{1,\ldots,N\}$.

La figura 2.4 representa las conexiones inhibitorias de la neurona (x,i), tal y como se establecen en $T_{xi,yj}$ e $i^b_{x,i}$, con el fin de garantizar las restricciones del problema y la minimización de la función objetivo. Se aprecia cómo la neurona (x,i) penaliza con el valor $-Av_{x,i}$ a todas las neuronas de su misma fila, con valor $-Bv_{x,i}$ a todas las de su misma columna, con $-Cv_{x,i}$ a todas las neuronas (incluida ella misma) y con $-Dd_{x,y}v_{x,i}$ a los elementos de las columnas anteriores y siguientes (salvo los de su misma fila).

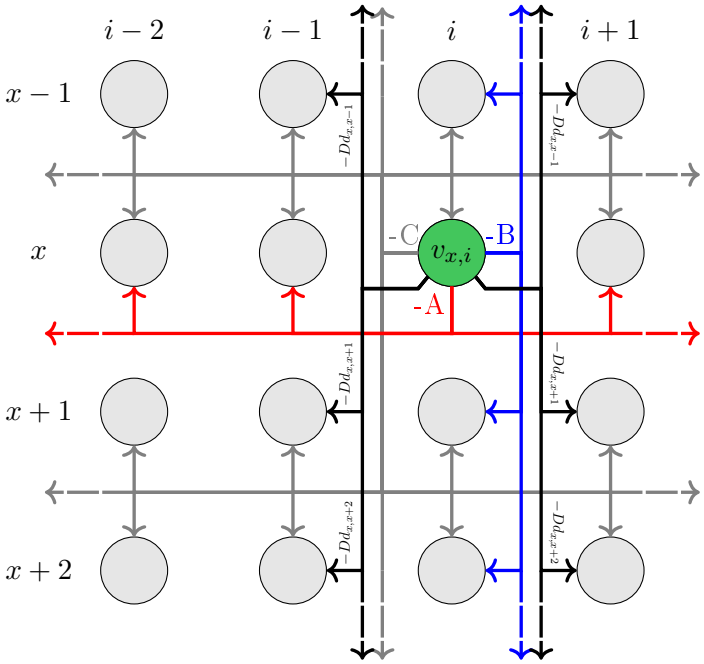


Figura 2.4: Disposición de las conexiones inhibitorias entre neuronas de la CHN aplicado al TSP

2.5.1 Determinación de parámetros para el TSP proyectado

Partiendo del trabajo de Talaván y Yáñez [49], la parametrización a utilizar será:

$$\begin{cases} B = 3 + C \\ A = B - Dd_L \\ N' = N + \frac{3}{C} \\ D = \frac{1}{d_U} \end{cases} \tag{2.19}$$

siendo d_U y d_L las cotas superior e inferior de $d_{x,y}$:

$$d_L \leq d_{x,y} \leq d_U \quad \forall x, y \in \{1, \dots N\}$$

Esta parametrización requiere modificar ligeramente la función de energía de la ecuación 2.16 por:

$$\begin{aligned} E(\mathbf{v}) = \; & \frac{A}{2} \sum_{x}^{N} \sum_{i}^{N} \sum_{j \neq i}^{N} v_{x,i} v_{x,j} + \frac{B}{2} \sum_{i}^{N} \sum_{x}^{N} \sum_{y \neq x}^{N} v_{x,i} v_{y,i} + \frac{C}{2} \Big(\sum_{x}^{N} \sum_{i}^{N} v_{x,i} - N' \Big)^2 \\ & + \frac{D}{2} \sum_{x}^{N} \sum_{y \neq x}^{N} \sum_{i}^{N} d_{x,y} v_{x,i} (v_{y,i-1} + v_{y,i+1}) \end{aligned} \tag{2.20}$$

(los subíndices $i+1$ e $i-1$ vienen dados módulo N)

y reemplazando por tanto $i^b_{x,i}$ en la ecuación 2.18 por:

$$i^b_{x,i} = CN' \quad \forall x, i \in \{1, \dots N\} \tag{2.21}$$

Observación 2.1. *Nótese que el problema puede ser planteado como un caso particular de GQKP, como ya fue mostrado por Talaván y Yáñez [51]. Sin embargo, en adelante se utilizará la formulación original de Hopfield y Tank [19], dado que utiliza menos parámetros.*

Esta parametrización se obtiene buscando la combinación de parámetros que garantice que cualquier punto de equilibrio del modelo caracterice un tour válido del TSP. Para ello se busca la estabilidad en las soluciones válidas ($\mathbf{v} \in H_F$) y la inestabilidad en las soluciones inválidas (vértices, $\mathbf{v} \in H_C \setminus H_F$ y puntos interiores ($\mathbf{v} \in H \setminus H_C$).

Siguiendo la misma idea que en las ecuaciones 2.7, 2.8 y 2.9, un punto $\mathbf{v} \in H$ será de equilibrio para la CHN si y sólo si se verifican las siguientes condiciones:

$$\underline{E}^0(\mathbf{v}) \geq 0$$

$$\overline{E}^1(\mathbf{v}) \leq 0$$

$$E_{x,i}(\mathbf{v}) = 0 \quad \forall (x,i) \in \{1,\ldots,N\}^2 \ / \ v_{x,i} \in (0,1)$$

donde

$$E_{x,i}(\mathbf{v}) \equiv \frac{\partial E(\mathbf{v})}{\partial v_{x,i}}, \qquad \underline{E}^0(\mathbf{v}) \equiv \min_{(x,i)/v_{x,i}=0} E_{x,i}(\mathbf{v}), \qquad \overline{E}^1(\mathbf{v}) \equiv \max_{(x,i)/v_{x,i}=1} E_{x,i}(\mathbf{v})$$

Sin llevar a cabo un análisis pormenorizado (ver detalles en Talaván y Yáñez [49]), se muestran a continuación las diferentes situaciones que es necesario penalizar con el objetivo de encontrar estabilidad de las soluciones válidas e inestabilidad de las soluciones inválidas.

Teniendo en cuenta las ecuaciones 2.13 y 2.14, se tiene

$$E_{x,i}(\mathbf{v}) = A(S_x - v_{x,i}) + B(S_i - v_{x,i}) + C(S - N') + D\sum_{y \neq x}^{N} d_{x,y}(v_{y,i-1} + v_{y,i+1})$$

donde $S = \sum_{x}^{N} S_x = \sum_{i}^{N} S_i$

- Estabilidad de las soluciones válidas, $\mathbf{v} \in H_F$

  - Si $v_{x,i} = 1$:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

    siendo $E_{x,i} = C(N - N') + 2D(d_{x,y} + d_{x,z}) \leq C(N - N') + 2Dd_U \leq 0$

  - Si $v_{x,i} = 0$:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

siendo posible elegir un valor $x = x_0$ tal que $v_{x_0,i+1} = 1$ (o equivalentemente con $i-1$), de modo que $\sum_{y \neq x_0} d_{x_0,y} v_{y,i+1} = 0$. Así,

$$E_{x_0,i} = A + B + C(N - N') + Dd_{x_0,z} \geq A + B + C(N - N') + Dd_L \geq 0$$

- Inestabilidad de las soluciones inválidas
  - Vértices, $\mathbf{v} \in H_C \setminus H_F$. Se distinguen 4 casos:
    - Una columna con (al menos) dos unos y N unos en total:
      $\exists i_0 \in \{1, \ldots, N\} / S_{i_0} \geq 2$, con $S = N$

      $$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

      Tomando $x_0 \,/\, v_{x_0,i_0} = 1$, y teniendo en cuenta que $S_{x_0} \geq v_{x_0,i_0} \geq 1$ y $S_i \geq v_{x,i}\, \forall (x,i) \in \{1, \ldots, N\}^2$

      $$\begin{aligned} \overline{E}^1(\mathbf{v}) \geq E_{x_0,i_0}(\mathbf{v}) \quad & \geq A(S_{x_0} - 1) + B(S_{i_0} - 1) + C(N - N') \\ & + Dd_L(S_{i_0-1} - v_{x_0,i_0-1} + S_{i_0+1} - v_{x_0,i_0+1}) \\ & \geq B + C(N - N') \end{aligned}$$

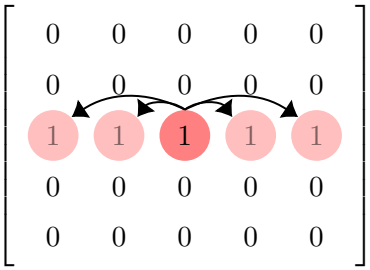
      forzando la inestabilidad si $\overline{E}^1(\mathbf{v}) \geq B + C(N - N') > 0$
    - Una fila con (al menos) dos unos, existiendo uno de ellos con un elemento anterior o posterior cero y N unos en total:
      $\exists x_0 \in \{1, \ldots, N\} / S_{x_0} \geq 2$, con $S_i = 1$, $\exists i_0 \in \{1, \ldots, N\} / v_{x_0,i_0} = 1 \wedge (v_{x_0,i_0-1} = 0 \vee v_{x_0,i_0+1} = 0)$, con $S = N$

      $$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

      $$\overline{E}^1(\mathbf{v}) \geq E_{x_0,i_0}(\mathbf{v}) \geq A + C(N - N') + Dd_L$$

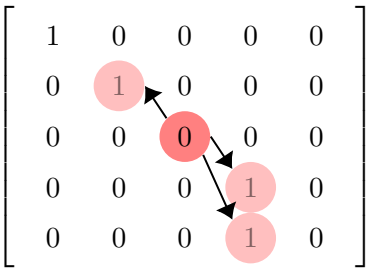
      forzando la inestabilidad si $\overline{E}^1(\mathbf{v}) \geq A + C(N - N') + Dd_L > 0$
    - Toda una fila con unos y N unos en total:

$\exists x_0 \in \{1,\dots,N\}\,/v_{x_0,i} = 1\,\forall i \in \{1,\dots,N\}$, i.e. $S_{x_0} = N$, con $S = N$



$$\overline{E}^1(\mathbf{v}) \geq E_{x_0,i}(\mathbf{v}) \geq (N-1)A + C(N-N')$$

– $N-1$ unos en total, (al menos) una columna con ceros y una fila con ceros:
$\exists x_0 \in \{1,\dots,N\}\,/S_{x_0} = 0$, con $S = N-1$. Sea $i_0 \in \{1,\dots,N\}\,/S_{i_0} = 0$



$$\underline{E}^0(\mathbf{v}) \leq E_{x_0,i_0}(\mathbf{v}) \leq C(N-1-N') + 3Dd_U$$

forzando la inestabilidad si $\underline{E}^0(\mathbf{v}) \leq C(N-1-N') + 3Dd_U < 0$

Así, la inestabilidad en los vértices puede ser forzada si se encuentran parámetros A, B, C, D y N' que verifiquen:

$$C(N-N') + \min\{B, A + Dd_L, (N-1)A\} > 0 \tag{2.22}$$

$$3Dd_U - C + C(N-N') < 0 \tag{2.23}$$

de modo que cualquier solución inválida $\mathbf{v} \in H_C \setminus H_F$ no pueda ser de equilibrio.

- Puntos interiores, $\mathbf{v} \in H \setminus H_C$
  En esta ocasión, se trabaja con un conjunto específico de puntos interiores, los puntos arista. Se llama punto arista a un punto interior $\mathbf{v} \in H \setminus H_C$ tal que

$$|\{(x,i) \in \{1,\dots,N\}^2 / v_{x,i} \in (0,1)\}| = 1$$

  Se consigue evitar que los puntos aristas sean estables [49] forzando $\underline{E}^0(\mathbf{v}) \leq -C$ y $\overline{E}^1(\mathbf{v}) \geq C$, convirtiendo las desigualdades 2.22 y 2.23 en:

$$C(N-N') + \min\{B, A + Dd_L, (N-1)A\} \geq C \tag{2.24}$$

$$3Dd_U - C + C(N-N') \leq -C \tag{2.25}$$

  El resto de puntos interiores son un punto de silla con respecto a la función de energía, teniendo probabilidad nula de ser un punto de convergencia de la CHN [33].

A partir de las desigualdades 2.24 y 2.25 se obtiene la parametrización de la ecuación 2.19.

2.6 Cuencas de atracción y análisis del punto de silla de la CHN aplicado al TSP

De un modo similar al realizado para el ejemplo del GQKP en la sección 2.3, esta sección se centrará en hacer un análisis del punto de silla para la CHN aplicado al TSP. Se presentan en primer lugar algunos resultados técnicos necesarios para llevar a cabo el análisis.

Proposición 2.1. *Dada una matriz circulante por bloques (cada fila está desplazada un bloque a la derecha con respecto a la fila que le precede) de tamaño N par ($N = 2m$), su matriz inversa tiene la siguiente estructura:*

$$\begin{bmatrix}
\mathbf{Q_1} & \mathbf{Q_2} & \mathbf{Q_3} & \cdots & \mathbf{Q_{\frac{N}{2}-1}} & \mathbf{Q_{\frac{N}{2}}} & \mathbf{Q_{\frac{N}{2}+1}} & \mathbf{Q_{\frac{N}{2}}} & \mathbf{Q_{\frac{N}{2}-1}} & \cdots & \mathbf{Q_3} & \mathbf{Q_2} \\
\mathbf{Q_2} & \ddots & & & & & & & & & \ddots & \mathbf{Q_3} \\
\mathbf{Q_3} & & \ddots & & & & & & & \ddots & & \vdots \\
\vdots & & & \ddots & & & & & \ddots & & & \mathbf{Q_{\frac{N}{2}-1}} \\
\mathbf{Q_{\frac{N}{2}-1}} & & & & \ddots & & & \ddots & & & & \mathbf{Q_{\frac{N}{2}}} \\
\mathbf{Q_{\frac{N}{2}}} & & & & & \ddots & \ddots & & & & & \mathbf{Q_{\frac{N}{2}+1}} \\
\mathbf{Q_{\frac{N}{2}+1}} & & & & & \ddots & \ddots & & & & & \mathbf{Q_{\frac{N}{2}}} \\
\mathbf{Q_{\frac{N}{2}}} & & & & \ddots & & & \ddots & & & & \mathbf{Q_{\frac{N}{2}-1}} \\
\mathbf{Q_{\frac{N}{2}-1}} & & & \ddots & & & & & \ddots & & & \vdots \\
\vdots & & \ddots & & & & & & & \ddots & & \mathbf{Q_3} \\
\mathbf{Q_3} & \ddots & & & & & & & & & \ddots & \mathbf{Q_2} \\
\mathbf{Q_2} & \mathbf{Q_3} & \cdots & \mathbf{Q_{\frac{N}{2}-1}} & \mathbf{Q_{\frac{N}{2}}} & \mathbf{Q_{\frac{N}{2}+1}} & \mathbf{Q_{\frac{N}{2}}} & \mathbf{Q_{\frac{N}{2}-1}} & \cdots & \mathbf{Q_3} & \mathbf{Q_2} & \mathbf{Q_1}
\end{bmatrix}$$

Demostración. Sea $\mathbf{T}$ una matriz circulante por bloques invertible. Entonces, utilizando el resultado de De Mazancourt et al. [7] acerca de las matrices circulantes por bloques (block-circulant matrix –BCM–), la inversa de una matriz BCM es también BCM. Por tanto, dado que $\mathbf{T^{-1}}$ es BCM, puede escribirse como (si $N = 2m$):

$$\mathbf{T^{-1}} = \begin{bmatrix}
\mathbf{Q_1} & \mathbf{Q_2} & & & & \mathbf{Q_{N-2}} & \mathbf{Q_{N-1}} & \mathbf{Q_N} \\
\mathbf{Q_N} & \mathbf{Q_1} & \mathbf{Q_2} & & & & \ddots & \mathbf{Q_{N-1}} \\
\mathbf{Q_{N-1}} & \mathbf{Q_N} & \mathbf{Q_1} & \mathbf{Q_2} & & & & \mathbf{Q_{N-2}} \\
\vdots & \ddots & \ddots & \ddots & \ddots & & & \vdots \\
 & & & & & & & \\
\mathbf{Q_3} & & & & & & & \mathbf{Q_2} \\
\mathbf{Q_2} & \mathbf{Q_3} & & & & \mathbf{Q_{N-1}} & \mathbf{Q_N} & \mathbf{Q_1}
\end{bmatrix}$$

Como $\mathbf{T^{-1}}$ es simétrica (puesto que $\mathbf{T}$ es simétrica), se deduce que:

$$\mathbf{Q_k} = \mathbf{Q_{N-(k-2)}} \quad \forall k \in \left\{2, \ldots, \frac{N}{2} + 1\right\}$$

□

Un resultado equivalente se obtiene si N es impar.

Proposición 2.2. *El punto de silla* $\mathbf{v}^*$ *de la función de energía definida en la ecuación 2.17 utilizando la entrada externa modificada de la ecuación 2.21 es:*

$$v^*_{x,i} = C\left(N + \frac{3}{C}\right)\sum_j M_{x,j}, \quad \forall x, i \in \{1, \ldots, N\},$$

$$\textit{con} \quad \mathbf{M} = \left[((N{+}1)\,C{+}3)\,\mathbb{1} + \left((N{-}2)\,(C{+}3) - (N{-}1)\,\tfrac{d_L}{d_U}\right)\mathbf{I} + \tfrac{2}{d_U}\mathbf{D}\right]^{-1}$$

$$\textit{donde} \quad \mathbb{1} \equiv \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}_{N\times N}, \; \mathbf{I} \equiv \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}_{N\times N} \quad y \quad \mathbf{D} \equiv (d_{x,y})_{x,y\in\{1,\ldots,N\}}$$

Demostración. El punto de silla de la función de energía definida en 2.17 se obtiene calculando las derivadas parciales de $E(\mathbf{v})$ con respecto a $\mathbf{v}$ e igualándolas a cero, $-\mathbf{T}\mathbf{v} - \mathbf{i}^b = 0$. Por tanto, el punto de silla $\mathbf{v}^*$ es:

$$\mathbf{v}^* = -\mathbf{T}^{-1}\mathbf{i}^b \tag{2.26}$$

La matriz $\mathbf{T}$ puede escribirse como una matriz $N^2 \times N^2$ (matriz por bloques $N \times N$) del siguiente modo:

$$\mathbf{T} = -\begin{bmatrix} (B{+}C)\mathbb{1}{-}B\mathbf{I} & C\mathbb{1}{+}A\mathbf{I}{+}D\mathbf{D} & C\mathbb{1}{+}A\mathbf{I} & \cdots & C\mathbb{1}{+}A\mathbf{I} & C\mathbb{1}{+}A\mathbf{I}{+}D\mathbf{D} \\ C\mathbb{1}{+}A\mathbf{I}{+}D\mathbf{D} & \ddots & \ddots & & \ddots & C\mathbb{1}{+}A\mathbf{I} \\ C\mathbb{1}{+}A\mathbf{I} & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & C\mathbb{1}{+}A\mathbf{I} \\ C\mathbb{1}{+}A\mathbf{I} & & & \ddots & \ddots & C\mathbb{1}{+}A\mathbf{I}{+}D\mathbf{D} \\ C\mathbb{1}{+}A\mathbf{I}{+}D\mathbf{D} & C\mathbb{1}{+}A\mathbf{I} & \cdots & C\mathbb{1}{+}A\mathbf{I} & C\mathbb{1}{+}A\mathbf{I}{+}D\mathbf{D} & (B{+}C)\mathbb{1}{-}B\mathbf{I} \end{bmatrix} \tag{2.27}$$

donde las matrices $\mathbb{1}$, $\mathbf{I}$ y $\mathbf{D}$ siguen las definiciones del enunciado. Además, el vector de umbrales $\mathbf{i^b}$ tiene la estructura:

$$\mathbf{i}^b = \begin{bmatrix} CN' \\ \vdots \\ CN' \end{bmatrix}_{N^2\times 1}$$

$\mathbf{T}$ es una matriz circulante por bloques. Si $\mathbf{T}$ es invertible y utilizando la proposición 2.1, $\mathbf{T}^{-1}$ es también circulante por bloques y tiene la siguiente estructura, en función de si N es par o impar:

Si $N = 2m$:

$$\mathbf{T}^{-1} = \begin{bmatrix}
\mathbf{Q_1} & \mathbf{Q_2} & \mathbf{Q_3} & \cdots & \mathbf{Q_{\frac{N}{2}-1}} & \mathbf{Q_{\frac{N}{2}}} & \mathbf{Q_{\frac{N}{2}+1}} & \mathbf{Q_{\frac{N}{2}}} & \mathbf{Q_{\frac{N}{2}-1}} & \cdots & \mathbf{Q_3} & \mathbf{Q_2} \\
\mathbf{Q_2} & & & & & & & & & & & \mathbf{Q_3} \\
\mathbf{Q_3} & & & & & & & & & & & \vdots \\
\vdots & & & & & & & & & & & \mathbf{Q_{\frac{N}{2}-1}} \\
\mathbf{Q_{\frac{N}{2}-1}} & & & & & & & & & & & \mathbf{Q_{\frac{N}{2}}} \\
\mathbf{Q_{\frac{N}{2}}} & & & & & & & & & & & \mathbf{Q_{\frac{N}{2}+1}} \\
\mathbf{Q_{\frac{N}{2}+1}} & & & & & & & & & & & \mathbf{Q_{\frac{N}{2}}} \\
\mathbf{Q_{\frac{N}{2}}} & & & & & & & & & & & \mathbf{Q_{\frac{N}{2}-1}} \\
\mathbf{Q_{\frac{N}{2}-1}} & & & & & & & & & & & \vdots \\
\vdots & & & & & & & & & & & \mathbf{Q_3} \\
\mathbf{Q_3} & & & & & & & & & & & \mathbf{Q_2} \\
\mathbf{Q_2} & \mathbf{Q_3} & \cdots & \mathbf{Q_{\frac{N}{2}-1}} & \mathbf{Q_{\frac{N}{2}}} & \mathbf{Q_{\frac{N}{2}+1}} & \mathbf{Q_{\frac{N}{2}}} & \mathbf{Q_{\frac{N}{2}-1}} & \cdots & \mathbf{Q_3} & \mathbf{Q_2} & \mathbf{Q_1}
\end{bmatrix}$$

donde $\mathbf{Q_k}$ son matrices $N \times N$ $\forall k \in \{1, \ldots, \frac{N}{2} + 1\}$.

Si $N = 2m + 1$:

$$\mathbf{T}^{-1} = \begin{bmatrix}
\mathbf{Q_1} & \mathbf{Q_2} & \mathbf{Q_3} & \cdots & \mathbf{Q_{\frac{N-3}{2}}} & \mathbf{Q_{\frac{N-1}{2}}} & \mathbf{Q_{\frac{N+1}{2}}} & \mathbf{Q_{\frac{N+1}{2}}} & \mathbf{Q_{\frac{N-1}{2}}} & \mathbf{Q_{\frac{N-3}{2}}} & \cdots & \mathbf{Q_3} & \mathbf{Q_2} \\
\mathbf{Q_2} & & & & & & & & & & & & \mathbf{Q_3} \\
\mathbf{Q_3} & & & & & & & & & & & & \vdots \\
\vdots & & & & & & & & & & & & \mathbf{Q_{\frac{N-3}{2}}} \\
\mathbf{Q_{\frac{N-3}{2}}} & & & & & & & & & & & & \mathbf{Q_{\frac{N-1}{2}}} \\
\mathbf{Q_{\frac{N-1}{2}}} & & & & & & & & & & & & \mathbf{Q_{\frac{N+1}{2}}} \\
\mathbf{Q_{\frac{N+1}{2}}} & & & & & & & & & & & & \mathbf{Q_{\frac{N+1}{2}}} \\
\mathbf{Q_{\frac{N+1}{2}}} & & & & & & & & & & & & \mathbf{Q_{\frac{N-1}{2}}} \\
\mathbf{Q_{\frac{N-1}{2}}} & & & & & & & & & & & & \mathbf{Q_{\frac{N-3}{2}}} \\
\mathbf{Q_{\frac{N-3}{2}}} & & & & & & & & & & & & \vdots \\
\vdots & & & & & & & & & & & & \mathbf{Q_3} \\
\mathbf{Q_3} & & & & & & & & & & & & \mathbf{Q_2} \\
\mathbf{Q_2} & \mathbf{Q_3} & \cdots & \mathbf{Q_{\frac{N-3}{2}}} & \mathbf{Q_{\frac{N-1}{2}}} & \mathbf{Q_{\frac{N+1}{2}}} & \mathbf{Q_{\frac{N+1}{2}}} & \mathbf{Q_{\frac{N-1}{2}}} & \mathbf{Q_{\frac{N-3}{2}}} & \cdots & \mathbf{Q_3} & \mathbf{Q_2} & \mathbf{Q_1}
\end{bmatrix}$$

donde $\mathbf{Q_k}$ son matrices $N \times N$ $\forall k \in \{1, \ldots, \frac{N+1}{2}\}$.

Teniendo en cuenta la ecuación 2.26 y la estructura para $\mathbf{T}^{-1}$, $\mathbf{v}^*$ puede calcularse como[1]:

$$\mathbf{v}^* = -\mathbf{T}^{-1}\mathbf{i}^b = \begin{bmatrix} \mathbf{v}^*_{\cdot 1} \\ \mathbf{v}^*_{\cdot 2} \\ \vdots \\ \mathbf{v}^*_{\cdot N} \end{bmatrix}_{N^2 \times 1} \tag{2.28}$$

[1] Nótese que la matriz de estados $\mathbf{V} = (v_{x,i})_{x,i \in \{1,\ldots,N\}}$ puede escribirse como vector de la forma: $\mathbf{v} = \begin{bmatrix} \mathbf{v}_{\cdot 1} \\ \mathbf{v}_{\cdot 2} \\ \vdots \\ \mathbf{v}_{\cdot N} \end{bmatrix}_{N^2 \times 1}$ con $\mathbf{v}_{\cdot i} = \begin{bmatrix} v_{1,i} \\ v_{2,i} \\ \vdots \\ v_{N,i} \end{bmatrix}_{N \times 1}$. Ver más detalles acerca de la notación en la página XXI.

con

$$\mathbf{v}^*_{\cdot i} = \begin{cases} -\Big(\mathbf{Q_1} + 2\sum_{k=2}^{\frac{N}{2}} \mathbf{Q_k} + \mathbf{Q}_{\frac{\mathbf{N}}{\mathbf{2}}+\mathbf{1}}\Big) CN'\mathbf{1} & \text{si} \quad N = 2m \\ -\Big(\mathbf{Q_1} + 2\sum_{k=2}^{\frac{N+1}{2}} \mathbf{Q_k}\Big) CN'\mathbf{1} & \text{si} \quad N = 2m{+}1 \end{cases}$$

donde $\mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{N\times 1}$

$$\text{Sea } \mathbf{Q}^{row} \equiv \begin{cases} \mathbf{Q_1} + 2\sum_{k=2}^{\frac{N}{2}} \mathbf{Q_k} + \mathbf{Q}_{\frac{\mathbf{N}}{\mathbf{2}}+\mathbf{1}} & \text{si} \quad N = 2m \\ \mathbf{Q_1} + 2\sum_{k=2}^{\frac{N+1}{2}} \mathbf{Q_k} & \text{si} \quad N = 2m+1 \end{cases}$$

entonces, teniendo en cuenta que $\mathbf{T}\mathbf{v}^* = \mathbf{i}^b$ y definiendo $\mathbf{q}^{row} \equiv \mathbf{Q}^{row}\mathbf{1}$:

$$\mathbf{T}\mathbf{v}^* = \mathbf{T}\begin{bmatrix} \mathbf{v}^*_{\cdot 1} \\ \mathbf{v}^*_{\cdot 2} \\ \vdots \\ \mathbf{v}^*_{\cdot N} \end{bmatrix}_{N^2\times 1} = \mathbf{T}\begin{bmatrix} -\mathbf{Q}^{row}CN'\mathbf{1} \\ -\mathbf{Q}^{row}CN'\mathbf{1} \\ \vdots \\ -\mathbf{Q}^{row}CN'\mathbf{1} \end{bmatrix}_{N^2\times 1} = -CN'\mathbf{T}\begin{bmatrix} \mathbf{q}^{row} \\ \mathbf{q}^{row} \\ \vdots \\ \mathbf{q}^{row} \end{bmatrix}_{N^2\times 1} = \begin{bmatrix} CN' \\ CN' \\ \vdots \\ CN' \end{bmatrix}_{N^2\times 1}$$

Puesto que todas las componentes son iguales, basta con desarrollar una de ellas:

$$-CN'\left(-\left[(B{+}C)\,\mathbb{1} - B\mathbf{I} + (N{-}3)\,(C\mathbb{1}{+}A\mathbf{I}) + 2\,(C\mathbb{1}{+}A\mathbf{I}{+}D\mathbf{D})\right]\right)\mathbf{q}^{row} = CN'\mathbf{1}$$

obteniendo:

$$\begin{aligned} \mathbf{q}^{row} &= \left[(B+C)\,\mathbb{1} - B\mathbf{I} + (N-3)\,(C\mathbb{1} + A\mathbf{I}) + 2\,(C\mathbb{1} + A\mathbf{I} + D\mathbf{D})\right]^{-1}\mathbf{1} \\ &= \left[(N{-}1)A\mathbf{I} + B(\mathbb{1} - \mathbf{I}) + NC\mathbb{1} + 2D\mathbf{D}\right]^{-1}\mathbf{1} \\ &= \left[(B{+}NC)\mathbb{1} + ((N{-}1)A - B)\mathbf{I} + 2D\mathbf{D}\right]^{-1}\mathbf{1} \end{aligned}$$

El punto de silla $\mathbf{v}^*$ de la función de energía $E(\mathbf{v})$ dada en la ecuación 2.20 en términos de los parámetros A, B, C, D y N':

$$\mathbf{v}^* = \begin{bmatrix} \mathbf{v}^*_{\cdot 1} \\ \mathbf{v}^*_{\cdot 2} \\ \vdots \\ \mathbf{v}^*_{\cdot N} \end{bmatrix}_{N^2\times 1}$$

donde

$$\begin{aligned}\mathbf{v}^*_{\cdot i} &= CN'\mathbf{q}^{row} \\ &= CN'\left[(B{+}NC)\mathbb{1} + ((N{-}1)A - B)\mathbf{I} + 2D\mathbf{D}\right]^{-1}\mathbf{1} \qquad \forall i \in \{1,\dots,N\}\end{aligned}$$

que puede escribirse de modo equivalente como:

$$v^*_{x,i} = CN'\sum_j m_{x,j}, \quad \forall x,i \in \{1,\dots,N\}, \tag{2.29}$$

con

$$\mathbf{M} = \left[(B{+}NC)\mathbb{1} + ((N{-}1)A - B)\mathbf{I} + 2D\mathbf{D}\right]^{-1} \tag{2.30}$$

Teniendo en cuenta la parametrización de la ecuación 2.19 para los parámetros A, B, C, D y N', se tiene que:

$$v^*_{x,i} = C\left(N + \frac{3}{C}\right)\sum_j m_{x,j}, \quad \forall x,i \in \{1,\dots,N\},$$

con

$$\mathbf{M} = \left[((N{+}1)\,C{+}3)\,\mathbb{1} + \left((N{-}2)\,(C{+}3) - (N{-}1)\,\frac{d_L}{d_U}\right)\mathbf{I} + \frac{2}{d_U}\mathbf{D}\right]^{-1}$$

□

Observación 2.2. *Merece la pena resaltar que gracias a la proposición 2.2 puede calcularse el punto de silla* $\mathbf{v}^*$ *sin tener que calcular la inversa de la matriz* $\mathbf{T}$*, de tamaño* $N^2 \times N^2$*. En su lugar, únicamente es necesario calcular la inversa de la matriz* $\mathbf{M}$*, de tamaño* $N \times N$*.*

A continuación, se introducen dos lemas técnicos relativos a la inversión de matrices que se utilizarán en la proposición 2.3.

Lema 2.1. *La inversa de la matriz* $N \times N$

$$\mathscr{M} = \left[(C{+}3)\mathbb{1} - \alpha_1^c\mathbf{I}\right]$$

es

$$\mathscr{M}^{-1} = \frac{1}{\alpha^c}\left[(C{+}3)\mathbb{1} - \alpha_2^c\mathbf{I}\right]$$

con α_1^c, α_2^c, $\alpha^c \in \mathbb{R}$ *verificando* $\alpha_2^c = \left((C{+}3)N{-}\alpha_1^c\right)$ *y* $\alpha^c = \alpha_1^c\alpha_2^c$*.*

Demostración. Nótese que $\mathbb{1}\cdot\mathbb{1} = N\mathbb{1}$ y $\alpha_1^c + \alpha_2^c = (C{+}3)N$. Entonces,

$$\begin{aligned}\mathscr{M}\cdot\mathscr{M}^{-1} &= \left[(C{+}3)\mathbb{1} - \alpha_1^c\mathbf{I}\right]\cdot\tfrac{1}{\alpha^c}\left[(C{+}3)\mathbb{1} - \alpha_2^c\mathbf{I}\right] \\ &= \tfrac{1}{\alpha^c}\left[(C{+}3)^2N\mathbb{1} - \alpha_2^c(C{+}3)\mathbb{1} - \alpha_1^c(C{+}3)\mathbb{1} + \alpha_1^c\alpha_2^c\mathbf{I}\right] \\ &= \tfrac{1}{\alpha^c}\left[(C{+}3)^2N\mathbb{1} - (\alpha_1^c + \alpha_2^c)(C{+}3)\mathbb{1} + \alpha_1^c\alpha_2^c\mathbf{I}\right] \\ &= \tfrac{1}{\alpha^c}\left[(C{+}3)^2N\mathbb{1} - (C{+}3)^2N\mathbb{1} + \alpha_1^c\alpha_2^c\mathbf{I}\right] \\ &= \tfrac{1}{\alpha^c}\left[\alpha_1^c\alpha_2^c\mathbf{I}\right] = \mathbf{I}\end{aligned}$$

□

Lema 2.2. *La inversa de la matriz $N \times N$*

$$\mathscr{M} = \Big[\big((N{+}1)C{+}3\big)\mathbb{1} + \big((N{-}2)(C{+}3){-}(N{-}1)\mu\big)\mathbf{I}\Big]$$

es

$$\mathscr{M}^{-1} = \frac{1}{\beta^c}\Big[-\big((N{+}1)C{+}3\big)\mathbb{1} + \beta_1^c\mathbf{I}\Big]$$

con β_1^c, β_2^c, $\beta^c \in \mathbb{R}$ verificando $\beta^c = \beta_1^c\beta_2^c$ y $\beta_1^c = (CN^2{+}\alpha_1^cN{-}\alpha_1^c)$, $\beta_2^c = (\alpha_2^c{-}\mu N)$, donde α_1^c, $\alpha_2^c \in \mathbb{R}$, $\alpha_1^c = \big(2C{-}\mu{+}6\big)$, $\alpha_2^c = \big((C{+}3)N{-}\alpha_1^c\big)$.

Demostración. Nótese que $\mathbb{1}\cdot\mathbb{1} = N\mathbb{1}$ y $\beta_1^c - \beta_2^c = \big((N{+}1)C{+}3\big)N$. Entonces,

$$\begin{aligned}
\mathscr{M}\cdot\mathscr{M}^{-1} &= \Big[\big((N{+}1)C{+}3\big)\mathbb{1} + \big((N{-}2)(C{+}3){-}(N{-}1)\mu\big)\mathbf{I}\Big]\cdot\tfrac{1}{\beta^c}\Big[-\big((N{+}1)C{+}3\big)\mathbb{1} + \beta_1^c\mathbf{I}\Big]\\
&= \tfrac{1}{\beta^c}\Big[\big((N{+}1)C{+}3\big)\mathbb{1} + \beta_2^c\mathbf{I}\Big]\cdot\Big[-\big((N{+}1)C{+}3\big)\mathbb{1} + \beta_1^c\mathbf{I}\Big]\\
&= \tfrac{1}{\beta^c}\Big[-\big((N{+}1)C{+}3\big)^2N\mathbb{1} + \beta_1^c\big((N{+}1)C{+}3\big)\mathbb{1} - \beta_2^c\big((N{+}1)C{+}3\big)\mathbb{1} + \beta_1^c\beta_2^c\mathbf{I}\Big]\\
&= \tfrac{1}{\beta^c}\Big[-\big((N{+}1)C{+}3\big)^2N\mathbb{1} + (\beta_1^c - \beta_2^c)\big((N{+}1)C{+}3\big)\mathbb{1} + \beta_1^c\beta_2^c\mathbf{I}\Big]\\
&= \tfrac{1}{\beta^c}\Big[-\big((N{+}1)C{+}3\big)^2N\mathbb{1} + \big((N{+}1)C{+}3\big)^2N\mathbb{1} + \beta_1^c\beta_2^c\mathbf{I}\Big]\\
&= \tfrac{1}{\beta^c}\Big[\beta_1^c\beta_2^c\mathbf{I}\Big] = \mathbf{I}
\end{aligned}$$

□

Proposición 2.3. *El límite del punto de silla $(v_{x,i}^*)$ del modelo de Hopfield aplicado al TSP cuando C tiende a infinito es $\frac{N}{N^2+2N-2}$, $\forall x, i \in \{1,\ldots,N\}$*

Demostración. Se considera la matriz $\mathbf{T}$ de la ecuación 2.27. A medida que $C > 0$ aumenta, la contribución de $D\cdot\mathbf{D}$ a $\mathbf{T}$ es prácticamente insignificante cuando se compara con $C\mathbb{1}+A\mathbf{I}$. Por lo tanto, $\mathbf{T}$ puede escribirse como:

$$\mathbf{T} \approx -\begin{bmatrix}
(B{+}C)\mathbb{1}{-}B\mathbf{I} & C\mathbb{1}{+}A\mathbf{I} & \cdots & C\mathbb{1}{+}A\mathbf{I}\\
C\mathbb{1}{+}A\mathbf{I} & \ddots & \ddots & \vdots\\
\vdots & \ddots & \ddots & C\mathbb{1}{+}A\mathbf{I}\\
C\mathbb{1}{+}A\mathbf{I} & \cdots & C\mathbb{1}{+}A\mathbf{I} & (B{+}C)\mathbb{1}{-}B\mathbf{I}
\end{bmatrix}$$

Utilizando la parametrización para A, B y C obtenida en la ecuación 2.19 y definiendo

$\mu = \dfrac{d_L}{d_U}$, $\mathbf{T}$ puede reescribirse como:

$$\mathbf{T} \approx -\begin{bmatrix} (2C{+}3)\mathbb{1}{-}(C{+}3)\mathbf{I} & C\mathbb{1}{+}(C{+}3{-}\mu)\mathbf{I} & \cdots & C\mathbb{1}{+}(C{+}3{-}\mu)\mathbf{I} \\ C\mathbb{1}{+}(C{+}3{-}\mu)\mathbf{I} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & C\mathbb{1}{+}(C{+}3{-}\mu)\mathbf{I} \\ (2C{+}3)\mathbb{1}{-}(C{+}3)\mathbf{I} & \cdots & (2C{+}3)\mathbb{1}{-}(C{+}3)\mathbf{I} & (2C{+}3)\mathbb{1}{-}(C{+}3)\mathbf{I} \end{bmatrix}$$

Teniendo en cuenta la estructura de $\mathbf{T}$ y la proposición 2.1, $\mathbf{T}^{-1}$ tiene la siguiente estructura:

$$\mathbf{T}^{-1} \approx \begin{bmatrix} \mathbf{Q_1} & \mathbf{Q_2} & \cdots & \mathbf{Q_2} \\ \mathbf{Q_2} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{Q_2} \\ \mathbf{Q_2} & \cdots & \mathbf{Q_2} & \mathbf{Q_1} \end{bmatrix} \tag{2.31}$$

Utilizando el hecho de que $\mathbf{T}\mathbf{T}^{-1} = \mathbf{I}$, se obtiene el siguiente sistema de ecuaciones lineales:

$$\begin{cases} \Big[(C{+}3)\mathbf{I} - (2C{+}3)\mathbb{1}\Big]\mathbf{Q_1} - (N{-}1)\Big[C\mathbb{1}{+}(C{+}3{-}\mu)\mathbf{I}\Big]\mathbf{Q_2} = \mathbf{I} & (2.32) \\ -\Big[C\mathbb{1}{+}(C{+}3{-}\mu)\mathbf{I}\Big]\mathbf{Q_1} - \Big[(2C{+}3)\mathbb{1}{-}(C{+}3)\mathbf{I}{+}(N{-}2)\big(C\mathbb{1}{+}(C{+}3{-}\mu)\mathbf{I}\big)\Big]\mathbf{Q_2} = \mathbf{0} & (2.33) \end{cases}$$

Restando la ecuación 2.33 a la ecuación 2.32:

$$\mathbf{Q_1} = \mathbf{Q_2} - \Big[(C{+}3)\mathbb{1} - \big(2C{-}\mu{+}6\big)\mathbf{I}\Big]^{-1}$$

y utilizando el resultado del lema 2.1 y fijando $\alpha_1^c = \big(2C{-}\mu{+}6\big)$, puede escribirse $\mathbf{Q_1}$ como:

$$\mathbf{Q_1} = \mathbf{Q_2} - \frac{1}{\alpha^c}\Big[(C{+}3)\mathbb{1} - \alpha_2^c\mathbf{I}\Big] \tag{2.34}$$

con

$$\alpha^c = \underbrace{\big(2C{-}\mu{+}6\big)}_{\alpha_1^c}\underbrace{\big((C{+}3)N{-}\alpha_1^c\big)}_{\alpha_2^c}$$

Sustituyendo $\mathbf{Q_1}$ de la ecuación 2.34 en la ecuación 2.32:

$$\begin{aligned} \mathbf{Q_2} = -&\Big[\big((N{+}1)C{+}3\big)\mathbb{1} + \big((N{-}2)(C{+}3){-}(N{-}1)\mu\big)\mathbf{I}\Big]^{-1} \\ &\cdot\Big[\mathbf{I} + \frac{1}{\alpha^c}\big((C{+}3)\mathbf{I} - (2C{+}3)\mathbb{1}\big)\big((C{+}3)\mathbb{1} - \alpha_2^c\mathbf{I}\big)\Big] \end{aligned}$$

y utilizando el resultado del lema 2.2, puede escribirse $\mathbf{Q_2}$ como:

$$\begin{aligned}\mathbf{Q_2} = &-\frac{1}{\beta^c}\Big[-\big((N{+}1)C{+}3\big)\mathbb{1} + \beta_1^c\mathbf{I}\Big] \\ &\cdot \Big[\mathbf{I} + \frac{1}{\alpha^c}\big((C{+}3)\mathbf{I} - (2C{+}3)\mathbb{1}\big)\big((C{+}3)\mathbb{1} - \alpha_2^c\mathbf{I}\big)\Big]\end{aligned} \tag{2.35}$$

con

$$\beta^c = \underbrace{\big(CN^2{+}\alpha_1^c N{-}\alpha_1^c\big)}_{\beta_1^c}\underbrace{\big(\alpha_2^c{-}\mu N\big)}_{\beta_2^c}$$

Recordando que $\mathbf{v}^* = -\mathbf{T}^{-1}\mathbf{i}^b$ (ecuación 2.28), la estructura para $\mathbf{T}^{-1}$ (ecuación 2.31) y el resultado obtenido para $\mathbf{Q_1}$ (ecuación 2.34), se obtiene el punto de silla como:

$$\mathbf{v}_{\cdot i}^* = -\mathbf{Q_1}CN'\mathbf{1} - (N{-}1)\mathbf{Q_2}CN'\mathbf{1} = -N\mathbf{Q_2}CN'\mathbf{1} + \frac{1}{\alpha^c}\big((C{+}3)\mathbb{1}{-}\alpha_2^c\mathbf{I}\big)CN'\mathbf{1}$$

Puesto que $\mathbb{1}\cdot\mathbf{1} = N\mathbf{1}$ y $\mathbf{I}\cdot\mathbf{1} = \mathbf{1}$, y utilizando el resultado obtenido para $\mathbf{Q_2}$ en la ecuación 2.35:

$$\begin{aligned}\mathbf{v}_{\cdot i}^* &= -CN'\Big[N\mathbf{Q_2}\mathbf{1} - \frac{1}{\alpha^c}\big((C{+}3)N\mathbf{1} + \alpha_2^c\mathbf{1}\big)\Big] \\ &= -CN'\Bigg[N\Bigg[-\frac{1}{\beta^c}\Big[-\big((N{+}1)C{+}3\big)\mathbb{1} + \beta_1^c\mathbf{I}\Big] \\ &\quad\cdot\Big[\mathbf{I} + \frac{1}{\alpha^c}\big((C{+}3)\mathbf{I} - (2C{+}3)\mathbb{1}\big)\big((C{+}3)\mathbb{1} - \alpha_2^c\mathbf{I}\big)\Big]\Bigg]\mathbf{1} - \frac{1}{\alpha^c}\big((C{+}3)N\mathbf{1} - \alpha_2^c\mathbf{1}\big)\Bigg] \\ &= CN'\frac{1}{\alpha^c\beta^c}\Bigg[N\Big[-\big((N{+}1)C{+}3\big)\mathbb{1} + \beta_1^c\mathbf{I}\Big] \\ &\quad\cdot\Big[\alpha^c\mathbf{1} + \big((C{+}3)\mathbf{I} - (2C{+}3)\mathbb{1}\big)\big((C{+}3)N\mathbf{1} - \alpha_2^c\mathbf{1}\big)\Big] + \beta^c\big((C{+}3)N\mathbf{1} - \alpha_2^c\mathbf{1}\big)\Bigg] \\ &= CN'\frac{1}{\alpha^c\beta^c}\Bigg[N\Big[-\big((N{+}1)C{+}3\big)\mathbb{1}\alpha^c\mathbf{1} + \beta_1^c\mathbf{I}\alpha^c\mathbf{1}\Big] \\ &\quad + N\Big[-\big((N{+}1)C{+}3\big)\mathbb{1} + \beta_1^c\mathbf{I}\Big]\big((C{+}3)\mathbf{I} - (2C{+}3)\mathbb{1}\big)\big((C{+}3)N\mathbf{1} - \alpha_2^c\mathbf{1}\big) \\ &\quad + \beta^c\big((C{+}3)N\mathbf{1} - \alpha_2^c\mathbf{1}\big)\Bigg] \\ &= CN'\frac{1}{\alpha^c\beta^c}\Bigg[\alpha^c N\Big[-N\big((N{+}1)C{+}3\big)\mathbf{1} + \beta_1^c\mathbf{1}\Big] \\ &\quad + N\Big[-\big((N{+}1)C{+}3\big)\mathbb{1} + \beta_1^c\mathbf{I}\Big] \\ &\quad\cdot\big((C{+}3)^2N\mathbf{1} - (C{+}3)\alpha_2^c\mathbf{1} - (2C{+}3)(C{+}3)N^2\mathbf{1} + (2C{+}3)\alpha_2^cN\mathbf{1}\big) \\ &\quad + \beta^c\big((C{+}3)N\mathbf{1} - \alpha_2^c\mathbf{1}\big)\Bigg] \\ &= CN'\frac{1}{\alpha^c\beta^c}\Bigg[\alpha^c N\Big[-N\big((N{+}1)C{+}3\big) + \beta_1^c\Big] \\ &\quad + N\Big[-N\big((N{+}1)C{+}3\big) + \beta_1^c\Big]\alpha_1^c(C{+}3{-}(2C{+}3)N) + \beta^c\big((C{+}3)N - \alpha_2^c\big)\Bigg]\mathbf{1}\end{aligned}$$

Puesto que todos los elementos de $\mathbf{v}_{\cdot i}^*$ son iguales y $\alpha_1^c + \alpha_2^c = (C{+}3)N$, $\beta_1^c - \beta_2^c =$

$\big((N{+}1)C{+}3\big)N$:

$$v^*_{x,i} = CN'\frac{1}{\alpha^c\beta^c}\Bigg[\alpha^c N\underbrace{\Big[-N\big((N{+}1)C{+}3\big)+\beta^c_1\Big]}_{\beta^c_2} + N\underbrace{\Big[-N\big((N{+}1)C{+}3\big)+\beta^c_1\Big]}_{\beta^c_2}\alpha^c_1(C{+}3{-}(2C{+}3)N)+\beta^c\underbrace{\big((C{+}3)N-\alpha^c_2\big)}_{\alpha^c_1}\Bigg]$$
$$= CN'\frac{\alpha^c_1\beta^c_2}{\alpha^c_1\alpha^c_2\beta^c_1\beta^c_2}\underbrace{\Big[\alpha^c_2 N + N(C{+}3{-}(2C{+}3)N)+\beta^c_1\Big]}_{\alpha^c_2} = \frac{CN'}{\beta^c_1}$$

Por tanto, considerando la parametrización para N' en la ecuación 2.19 y el valor para β^c_1 (ecuación 2.35), el límite de $v^*_{x,i}$ cuando C tiende a infinito es:

$$\lim_{C\to\infty} v^*_{x,i} = \lim_{C\to\infty}\frac{CN'}{\beta^c_1} = \lim_{C\to ß1nfty}\frac{C(N+\frac{3}{C})}{CN^2{+}(2C{-}\mu{+}6)N{-}(2C{-}\mu{+}6)} = \frac{1}{N^2+2N-2}$$

□

Observación 2.3. *Nótese que si la parametrización en la ecuación 2.19 hubiese sido $N' = N$, el límite cuando C tiende a ∞ se mantendría igual, pero entonces las soluciones válidas no serían necesariamente estables, motivo por el cual se introduce la ecuación 2.21.*

En la sección 2.3.1, se mostraba cómo al elegir un valor lo suficientemente ($\phi_{1,1} \leq 3\alpha$) pequeño para el parámetro libre $\phi_{1,1}$ se consigue mover el punto de silla fuera del hipercubo de Hamming, dejando todo el hipercubo en una única cuenca de atracción, lo cual permite a la CHN obtener siempre la solución óptima.

Con respecto al punto de silla $\mathbf{v}^*$ (ecuación 2.6) de la función de energía del modelo de Hopfield aplicado al TSP (ecuación 2.17), es relevante saber si el punto de silla puede ser también empujado hacia una esquina (en este caso el origen) del hipercubo de Hamming, con el objetivo de favorecer mejores soluciones.

En este caso, encontrar el valor del límite de $v^*_{x,i}$ no es tan sencillo, dado que se necesita obtener el límite de la inversa de una suma de matrices. Es por ello que se verifica empíricamente utilizando MATLAB que $\mathbf{v}^*$ es claramente un punto interior que está además significativamente separado del origen cuando $C \to 0$.

$$\lim_{C\to 0} v^*_{x,i} > 0$$

Adicionalmente, utilizando el resultado de la proposición 2.3:

$$\lim_{C\to\infty} v^*_{x,i} = \frac{N}{N^2+2N-2} < 1, \quad \forall N \geq 1$$

Por tanto, en el caso de la CHN aplicado al TSP, no es posible mover el punto de silla al origen del hipercubo de Hamming, y, para cualquier valor de C, $v^*_{x,i}$ es siempre un punto interior:

$$v^*_{x,i} \in \left(0, \frac{N}{N^2+2N-2}\right) \subset (0,1)$$

Éste es un resultado un tanto desalentador, en el sentido de que no es posible obtener una sola cuenca de atracción que contenga todo el hipercubo de Hamming.

Sin embargo, pese a que disminuir el valor de C no empuja el punto de silla al origen, sí que ayudará a que las cuencas de atracción sean mayores para las mejores soluciones, tal y como se mostrará en la siguiente sección.

Observación 2.4. *Nótese que si la parametrización en la ecuación 2.19 fuera $N' = N$, entonces:*

$$\lim_{C \to 0} v^*_{x,i} = 0$$

sería posible mover el punto de silla al origen, pero entonces las soluciones válidas no serían necesariamente estables, motivo por el cual se introduce la ecuación 2.21.

2.6.1 Resolución del ejemplo del TSP utilizando el simulador de la CHN

El punto de silla se caracteriza por ser el punto donde confluyen todas las cuencas de atracción. Recordando el ejemplo mostrado en la sección 2.3, el punto de silla se mueve a la esquina (e incluso fuera) del hipercubo de Hamming cuando el parámetro libre es lo suficientemente pequeño, $\phi_{1,1} \leq 3\alpha$ (ver figura 2.2). Como con cualquier heurística, el punto de arranque en la CHN juega un papel fundamental en la simulación. En el ejemplo de la sección 2.3.1, elegir un punto de arranque aleatoriamente dentro del hipercubo de Hamming parece ser una mejor opción que elegirlo próximo al punto de silla (ver de nuevo la figura. 2.2), las cuencas de atracción para las mejores soluciones se hacen mayores si se disminuye el valor del parámetro libre. Esta idea se vuelve especialmente relevante a la hora de elegir el punto de arranque para resolver el TSP usando una CHN.

En el caso de la CHN aplicado al TSP, no es posible visualizar las cuencas de atracción, ya que se trata de un problema de alta dimensionalidad $[0,1]^{N \times N}$ con un número muy elevado de soluciones ($N!$ posibles tours, $\frac{N!}{2N}$ tours distintos). Con el fin de comprender el impacto del punto de arranque en la simulación de la CHN, se realizan múltiples simulaciones utilizando los dos enfoques siguientes:

(a) Eligiendo el punto de arranque aleatoriamente en cualquier lugar dentro de $[0,1]^{N \times N}$

(b) Eligiendo el punto de arranque dentro de $[\mathbf{v}^* - \varepsilon, \mathbf{v}^* + \varepsilon]$, $\varepsilon > 0$

La calidad de las soluciones se mide calculando el cociente de rendimiento ρ de la CHN:

$$\text{cociente de rendimiento } (\rho) = \frac{\text{longitud del tour}}{\text{longitud del tour óptimo}}$$

Las figuras 2.5 y 2.6 muestran las distribuciones del cociente de rendimiento y el cociente medio para los problemas de TSPLIB [38] *ATT48* y *CH150* (N = 48 ciudades, i.e. 2304 neuronas y N = 150 ciudades, i.e. 225000 neuronas) utilizando distintos valores de C para los dos enfoques considerados, (a) –izquierda– y (b) –derecha–. Se aprecia cómo en ambos casos se obtienen mejores soluciones disminuyendo el valor de C, desde $C = 10$ a $C = 10^{-5}$. Asimismo, la calidad de las soluciones es considerablemente mejor cuando el punto de arranque se elige próximo al punto de silla $\mathbf{v}^*$ (b). Además, la forma de la distribución en (a) no cambia al variar C, mientras que la forma en (b) se hace más estrecha a medida que

el valor de C es más pequeño, confirmando que las cuencas para las mejores soluciones se hacen mayores cuando C disminuye.

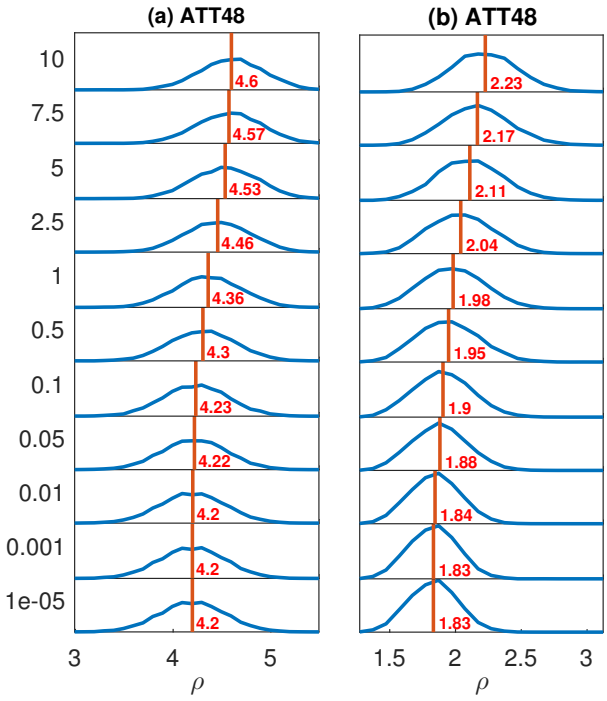


Figura 2.5: Distribución de las soluciones de la CHN para *ATT48*, eligiendo el punto de arranque: (a) aleatoriamente en cualquier lugar de $[0,1]^{N\times N}$; (b) aleatoriamente dentro de $[\mathbf{v}^* - \varepsilon, \mathbf{v}^* + \varepsilon]$, $\varepsilon > 0$

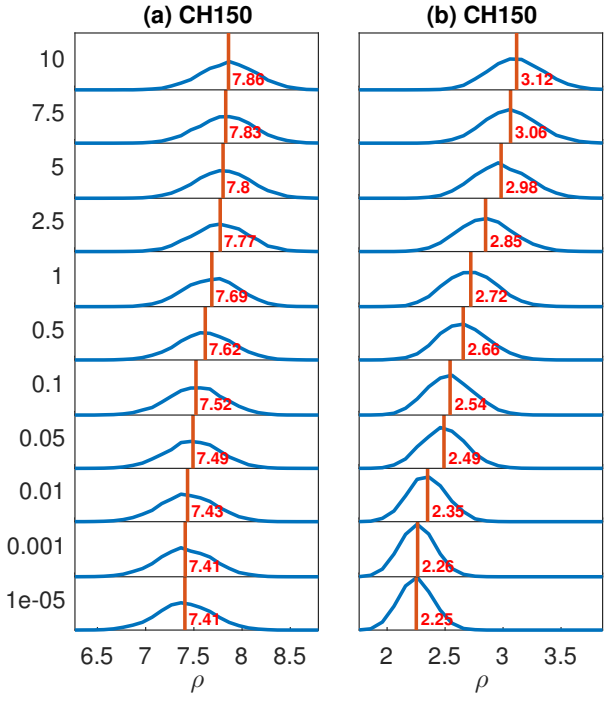


Figura 2.6: Distribución de las soluciones de la CHN para *CH150*, eligiendo el punto de arranque: (a) aleatoriamente en cualquier lugar den $[0,1]^{N\times N}$; (b) aleatoriamente dentro de $[\mathbf{v}^* - \varepsilon, \mathbf{v}^* + \varepsilon]$, $\varepsilon > 0$

Ejemplo 2.1. *Se considera a continuación un ejemplo sencillo consistente en un TSP con* $N = 4$ *ciudades, dispuestas en los vértices de un polígono regular centrado en el origen, con las ciudades a distancia 1 del mismo. Se consideran dos posibles valores del parámetro* C, 10^5 *y* 10^{-5}. *El objetivo es resolver el TSP con sendas CHNs y observar cómo la cuenca de atracción de la solución óptima (obviando simetrías) es más amplia en el caso en que* C *toma el valor más pequeño. Para ello, se resuelven las CHNs comenzando en un punto* $\mathbf{v} = \mathbf{s}^C$, *que se encuentra en ambos casos a la misma distancia del punto de silla, y que está en la cuenca de atracción del óptimo. Las simulaciones siguientes se hacen en la dirección del vector director de la primera neurona, incrementando únicamente el valor de* $v_{1,1}$.

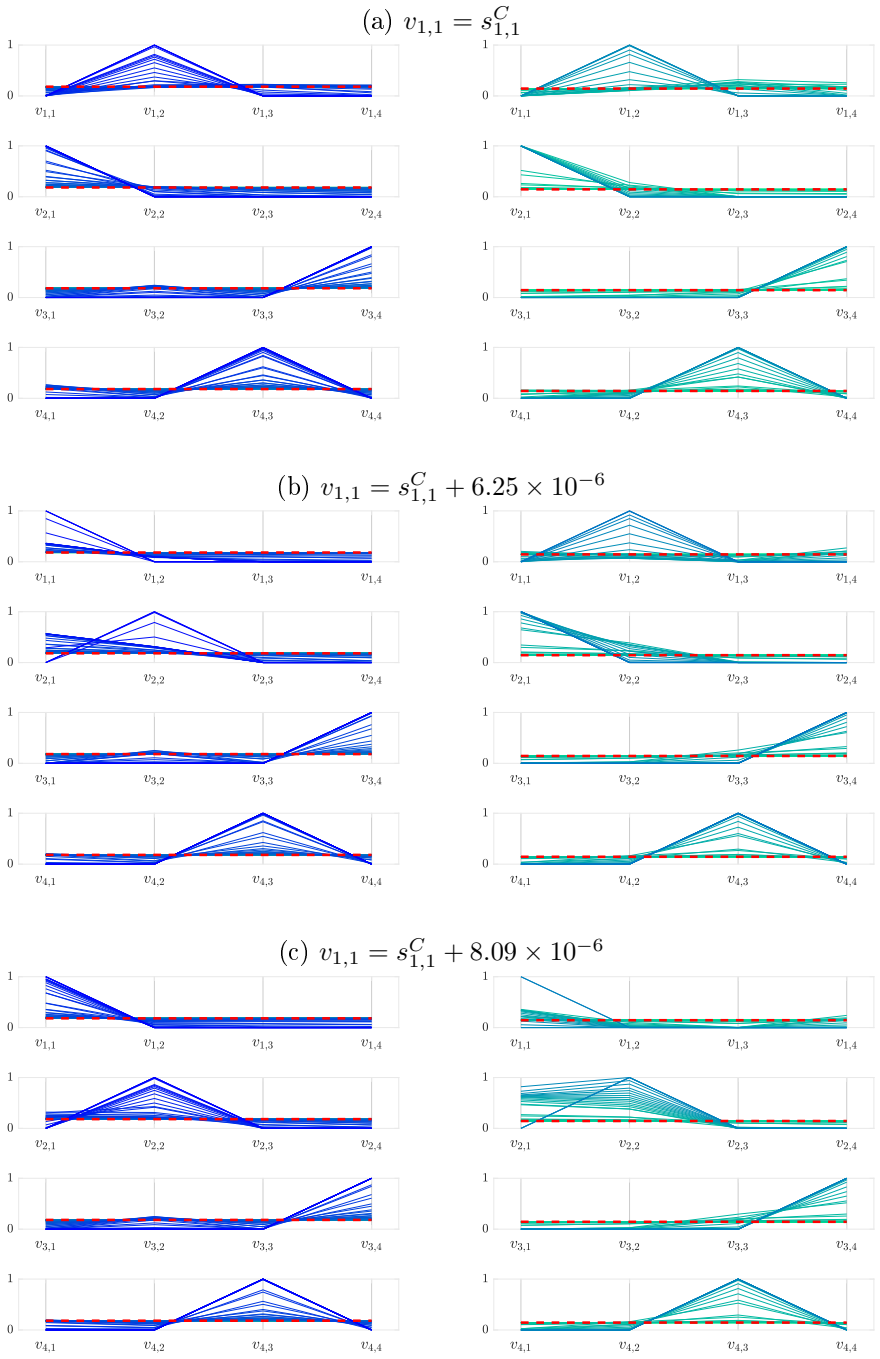


Figura 2.7: Trayectorias de la CHN con $C = 10^5$ (izquierda) y $C = 10^{-5}$ (derecha)

La figura 2.7 muestra en la columna de la izquierda las trayectorias correspondientes a tres simulaciones de la CHN con $C = 10^5$*, mientras que la columna de la derecha hace lo propio con* $C = 10^{-5}$*. En el caso en que* $v_{1,1} = s^C_{1,1}$ *(a), se aprecia cómo en ambos casos el valor inicial (marcado con línea discontinua roja) converge a la solución óptima. El salto de cuenca de atracción se produce para la CHN con* $C = 10^5$ *cuando* $v_{1,1} = s^C_{1,1} + 6.25 \times 10^{-6}$ *(b), mientras que la CHN con* $C = 10^{-5}$ *continua convergiendo a la solución óptima. Finalmente cuando* $v_{1,1} = s^C_{1,1} + 8.09 \times 10^{-6}$ *(c), se produce el salto de cuenca para la CHN con* $C = 10^{-5}$*. Se comprueba así, desde un punto de vista experimental que las cuencas de atracción para las mejores soluciones son más amplias cuando el parámetro libre es menor.*

CAPÍTULO 3

El modelo de Hopfield aplicado al TSP en dos fases

El tercer capítulo introduce una heurística basada en el modelo de Hopfield que utiliza una estrategia de Divide-y-Vencerás en dos fases para resolver el problema del viajante. Cada fase está modelizada utilizando una red de Hopfield continua (CHN), garantizando mediante sendas parametrizaciones, la factibilidad de las soluciones en cada fase. Esta estrategia permite mejorar el rendimiento del modelo de Hopfield cuando se aplica al TSP.

Contenidos del capítulo

3.1 Motivación e introducción al modelo de Hopfield en dos fases

Una estrategia frecuentemente utilizada para mejorar la optimalidad en métodos heurísticos consiste en la hibridación de varios métodos. En este sentido, la hibridación de algoritmos genéticos y el modelo de Hopfield ha probado ser una acertada estrategia para resolver problemas de optimización combinatoria con un alto número de restricciones (como es el caso del TSP). En estas asociaciones, el modelo de Hopfield gestiona algunas de las restricciones del problema [43].

Como ya se ha recogido en el capítulo anterior, la red o modelo de Hopfield continuo (CHN) puede utilizarse para resolver, entre otros problemas de optimización combinatoria, el problema del viajante (TSP). Este capítulo se centra en una *hibridación del modelo de Hopfield consigo mismo* con el objetivo de mejorar el rendimiento del modelo de Hopfield aplicado al TSP. Así, se presenta una estrategia basada en el procedimiento *Divide-y-Vencerás*, consistente en dos fases: se conectan en primer lugar aquellas ciudades que están próximas formando *cadenas* de ciudades, para después unir esas cadenas resultantes con las posiblemente restantes ciudades aisladas que hayan podido quedar. Ambas fases pueden expresarse como TSPs, resolviendo cada uno de ellos como una CHN.

La parametrización para la primera CHN es la misma que la introducida por Talaván y Yáñez [49] y ya presentada en la sección 2.5.1. Para la segunda CHN, siguiendo la sugerencia de Wen et al. [56] que sugiere que son necesarios nuevos avances en la parametrización del modelo de Hopfield para obtener resultados más fiables, se deduce una nueva parametrización.

La estrategia *Divide-y-Vencerás* resulta ser un enfoque muy útil para resolver problemas complejos y/o grandes, como es el caso de los problemas de optimización NP-duros. Una idea similar fue propuesta por Suh y Esat [46] para resolver el problema *sequence dependent set-up minimization problem*, que como muchos problemas de optimización combinatoria, puede ser transformado en el TSP. Su procedimiento consistía en utilizar una red neuronal Grossberg Regularity Detector (GDR) para particionar el problema y posteriormente utilizar la red de Hopfield para identificar la mejor de entre todas las soluciones. Sin embargo, su objetivo era encontrar soluciones factibles, cuando, por aquel entonces, todavía no estaba garantizado. La finalidad del procedimiento *Divide-y-Vencerás* detallado en este capítulo es otro, el de mejorar las calidad de las soluciones del modelo de Hopfield (puesto que la factibilidad está ya garantizada como se analizó en la sección 2.5.1).

3.1.1 Procedimiento Divide-y-Vencerás aplicado al TSP

El objetivo de esta red de Hopfield híbrida o modelo de Hopfield en dos fases es:

- en una primera fase unir las ciudades más próximas entre sí para definir un conjunto de *cadenas de ciudades* y,
- en una segunda fase, unir estas cadenas con posibles ciudades aisladas (no conectadas en la primera fase), para definir el tour final.

Cada una de estas fases se resuelve mediante la proyección de los dos TSPs en sus correspondientes CHNs, para lo que es necesario deducir sendas parametrizaciones (apartados 3.2

y 3.3). De este modo, se garantiza la factibilidad de los tours obtenidos, y es posible comparar esta heurística con el modelo continuo de Hopfield puro.

Teniendo en cuenta que la complejidad computacional de este problema *NP*-duro crece exponencialmente con el tamaño de la instancia, una estrategia clásica sería dividir el problema y después resolverlo. En el caso del TSP, el número de ciudades es el parámetro crítico y un modo natural de dividir el problema podría ser, por ejemplo, conectar las ciudades vecinas en una primera fase y después, en una segunda fase, unir las ciudades vecinas ya conectadas (o cadenas) para obtener el tour global.

Esta idea es diferente a una aproximación por clustering, en el que primero se agrupan un conjunto de ciudades en un número de grupos predefinido y después se unen los clusters resultantes. Con el enfoque propuesto, el mayor número de ciudades son conectadas en un número a priori desconocido de *cadenas*, teniendo únicamente en cuenta las distancias entre ciudades para la construcción de las mismas. Posteriormente, se conectarán las *cadenas* resultantes.

Recordando lo visto en la sección 2.4, a partir de la variable de decisión $\mathbf{v}$, las restricciones definidas a través de las ecuaciones 2.13 y 2.14, y la función objetivo (ecuación 2.15), se formula el TSP como un problema de optimización cuadrático binario. Partiendo de aquí, dado un TSP con N ciudades y matriz de distancias $\mathbf{D} = (d_{x,y})_{x,y\in\{1,\dots,N\}}$, el problema se resuelve en dos fases.

3.1.2 El problema de la Fase 1 (TSP_1^τ): Conectando vecinos

Dada una ciudad x, únicamente sus ciudades más próximas han de tenerse en cuenta, manteniendo las distancias originales entre ellas. Al mismo tiempo, el resto de ciudades son alejadas, fijando las distancias como d_U (una cota superior). El número de ciudades cercanas a considerar es el parámetro τ, el cual debe ser elegido de modo apropiado.

Formalmente, dado un parámetro entero $\tau \in \{0, 1, 2, \dots, N-1\}$ y una ciudad cualquiera $x \in \{1, 2, \dots, N\}$, la matriz de distancias original d_{xy} se modifica para cada ciudad que no está entre las τ-*ciudades vecinas* de x del siguiente modo:

$$d_{xy}^\tau = \begin{cases} & \text{si y es una de las } \tau\text{-}\textit{ciudades vecinas} \text{ de x} \\ d_{xy} & \qquad o \\ & \text{si x es una de las } \tau\text{-}\textit{ciudades vecinas}^2 \text{ de y} \\ d_U & \text{en otro caso} \end{cases} \tag{3.1}$$

donde d_U es la cota superior de distancias entre ciudades, $d_{x,y} \leq d_U \, \forall x, y \in \{1, \dots, N\}$.

Este nuevo TSP será denotado como TSP_1^τ y será considerado como la primera fase o fase 1 del método *Divide-y-Vencerás*.

Es importante resaltar los dos casos extremos:

[2]Una ciudad puede tener τ vecinos pero ser elegida como vecina por más de τ ciudades. Esta definición garantiza que la matriz de distancias es simétrica.

- Si $\tau = 0$, no ocurre nada en TSP_1^τ (todas las distancias son iguales a d_U) y el problema se resuelve en la siguiente fase (utilizando una CHN pura).
- Si $\tau = N-1$, el problema se resuelve en TSP_1^τ (utlizando una CHN pura) y no ocurre nada en la segunda fase.

El modelo matemático binario del TSP_1^τ

La variable de estados $\mathbf{v}$ es la misma que en la ecuación 2.12, teniendo también que verificar las ecuaciones 2.13 y 2.14. La función objetivo es de nuevo la misma que la introducida en la ecuación 2.15, pero cambiando la distancia $d_{x,y}$ por $d_{x,y}^\tau$:

$$\text{mín}\left\{\frac{1}{2}\sum_{x=1}^{N}\sum_{y\neq x}\sum_{i=1}^{N} d_{xy}^\tau v_{xi}(v_{y(i+1)}+v_{y(i-1)})\right\} \tag{3.2}$$

(los subíndices $i+1$ e $i-1$ vienen dados módulo N)

Sea $(z_1, z_2, \ldots, z_N)$ la permutación que identifica la solución óptima:

$$v_{z_1,1} = 1; v_{z_2,2} = 1; \cdots; v_{z_N,N} = 1$$

Esta solución define varias cadenas de ciudades *vecinas*

$$(z_k, z_{k+1}, \ldots, z_{k+h})$$

que verifican:

$$d_{z_{k-1},z_k}^\tau = d_U; d_{z_k,z_{k+1}}^\tau = d_{z_k,z_{k+1}}; \cdots; d_{z_{k+h-1},z_{k+h}}^\tau = d_{z_{k+h-1},z_{k+h}}; d_{z_{k+h},z_{k+h+1}}^\tau = d_U$$

Se denotará por K el número de cadenas no triviales (más de una ciudad), siendo las otras ciudades aisladas (no conectadas a ninguna cadena).

Ejemplo 3.1. *Sea $N = 10$ el número de ciudades del TSP original, y sea $d_{x,y}$ la distancia entre las ciudades x e y:*

$$\mathbf{D} = (d_{x,y})_{x,y\in\{1,\ldots,10\}} = \begin{bmatrix} 0.00 & 0.43 & 0.82 & 0.85 & 0.74 & 0.10 & 0.22 & 0.25 & 0.37 & 0.35 \\ 0.43 & 0.00 & 0.80 & 0.75 & 0.40 & 0.34 & 0.20 & 0.10 & 0.80 & 0.60 \\ 0.82 & 0.80 & 0.00 & 0.20 & 0.60 & 0.70 & 0.75 & 0.70 & 0.10 & 0.70 \\ 0.85 & 0.75 & 0.20 & 0.00 & 0.50 & 0.80 & 0.85 & 0.80 & 0.10 & 0.80 \\ 0.74 & 0.40 & 0.60 & 0.50 & 0.00 & 0.75 & 0.70 & 0.64 & 0.68 & 9.99 \\ 0.10 & 0.34 & 0.70 & 0.80 & 0.75 & 0.00 & 0.10 & 0.25 & 0.80 & 0.40 \\ 0.22 & 0.20 & 0.75 & 0.85 & 0.70 & 0.10 & 0.00 & 0.14 & 9.99 & 0.50 \\ 0.25 & 0.10 & 0.70 & 0.80 & 0.64 & 0.25 & 0.14 & 0.00 & 0.90 & 0.60 \\ 0.37 & 0.80 & 0.10 & 0.10 & 0.68 & 0.80 & 9.99 & 0.90 & 0.00 & 0.80 \\ 0.35 & 0.60 & 0.70 & 0.80 & 9.99 & 0.40 & 0.50 & 0.60 & 0.80 & 0.00 \end{bmatrix}$$

Sea $\tau = 3$, elegido de momento de modo arbitrario.

Teniendo en cuenta que $d_U = 9.99$ $d_{x,y} \leq 9.99\ \forall x, y \in \{1, \ldots, 10\}$, *la nueva matriz de distancias* $\mathbf{D}^\tau$ *para el problema modificado es:*

$$\mathbf{D}^\tau = (d^\tau_{x,y})_{x,y\in\{1,\ldots,10\}} = \begin{bmatrix} 0.00 & 9.99 & 9.99 & 9.99 & 9.99 & 0.10 & 0.22 & 0.25 & 0.37 & 0.35 \\ 9.99 & 0.00 & 9.99 & 9.99 & 0.40 & 0.34 & 0.20 & 0.10 & 9.99 & 9.99 \\ 9.99 & 9.99 & 0.00 & 0.20 & 0.60 & 9.99 & 9.99 & 9.99 & 0.10 & 9.99 \\ 9.99 & 9.99 & 0.20 & 0.00 & 0.50 & 9.99 & 9.99 & 9.99 & 0.10 & 9.99 \\ 9.99 & 0.40 & 0.60 & 0.50 & 0.00 & 9.99 & 9.99 & 9.99 & 9.99 & 9.99 \\ 0.10 & 0.34 & 9.99 & 9.99 & 9.99 & 0.00 & 0.10 & 0.25 & 9.99 & 0.40 \\ 0.22 & 0.20 & 9.99 & 9.99 & 9.99 & 0.10 & 0.00 & 0.14 & 9.99 & 0.50 \\ 0.25 & 0.10 & 9.99 & 9.99 & 9.99 & 0.25 & 0.14 & 0.00 & 9.99 & 9.99 \\ 0.37 & 9.99 & 0.10 & 0.10 & 9.99 & 9.99 & 9.99 & 9.99 & 0.00 & 9.99 \\ 0.35 & 9.99 & 9.99 & 9.99 & 9.99 & 0.40 & 0.50 & 9.99 & 9.99 & 0.00 \end{bmatrix}$$

Sea $(7, 8, 2, 3, 9, 4, 10, 5, 1, 6)$ *la permutación que identifica la solución de* TSP^τ_1*:*

$$\begin{array}{l} 7 \\ \quad | \ d^\tau_{7,8} \\ 8 \\ \quad | \ d^\tau_{8,2} \\ 2 \\ \quad \| \ d^\tau_{2,3} = d_U \\ 3 \\ \quad | \ d^\tau_{3,9} \\ 9 \\ \quad | \ d^\tau_{9,4} \\ 4 \\ \quad \| \ d^\tau_{4,10} = d_U \\ 10 \\ \quad \| \ d^\tau_{10,5} = d_U \\ 5 \\ \quad \| \ d^\tau_{5,1} = d_U \\ 1 \\ \quad | \ d^\tau_{1,6} \\ 6 \\ \quad | \ d^\tau_{6,7} \\ 7 \end{array}$$

y sean:

$$d^\tau_{7,8} = 0.14; d^\tau_{8,2} = 0.10; d^\tau_{2,3} = 9.99; d^\tau_{3,9} = 0.10; d^\tau_{9,4} = 0.10;$$
$$d^\tau_{4,10} = 9.99; d^\tau_{10,5} = 9.99; d^\tau_{5,1} = 9.99; d^\tau_{1,6} = 0.10; d^\tau_{6,7} = 0.10$$

las distancias consecutivas del tour. Se obtienen así dos cadenas, rompiendo el tour obtenido por las conexiones entre ciudades con distancia d_U*:*

$$(3, 9, 4) \ y \ (1, 6, 7, 8, 2)$$

Por tanto, $K = 2$ y ahora hay seis ciudades para el problema de la fase 2: los dos extremos de las cadenas $\{3, 4, 1, 2\}$ y las dos ciudades aisladas $\{5, 10\}$, tal y como se muestra en la figura 3.1.

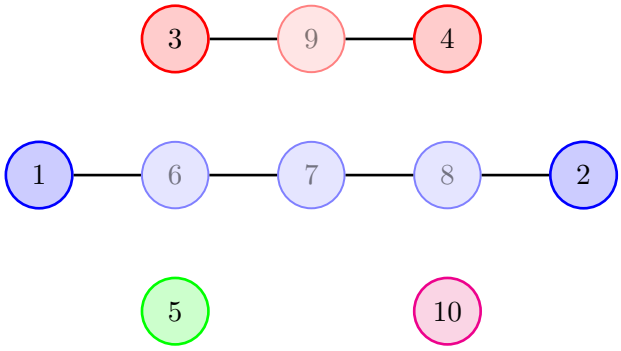


Figura 3.1: Las dos cadenas y dos ciudades aisladas del ejemplo 3.1

3.1.3 El problema de la Fase 2 ($TSP_2^{\not k}$): Conectando cadenas de vecinos

Dadas n nuevas ciudades $\{z_1, z_2, \ldots, z_n\} \subset \{1, 2, \ldots, N\}$ y las K cadenas resultado de la solución de TSP_1^{τ}, se plantea un nuevo TSP, denotado como $TSP_2^{\not k}$, del siguiente modo. El número n se obtiene sumando dos extremos de cadena para cada cadena y el número total de ciudades aisladas. Las nuevas ciudades, o conjunto de nodos son:

$$\{z_1, z_2, z_3, z_4, \ldots, z_{2K-1}, z_{2K}\} \cup \{z_{2K+1}, \ldots, z_n\}$$

donde $z_h \in \{1, 2, \ldots, N\}$ para $h \in \{1, 2, \ldots, n\}$.

Los pares $(z_{2k-1}; z_{2k})$ para $k \in \{1, 2, \ldots, K\}$ son las ciudades de los extremos de las K cadenas.

Se utiliza el orden canónico de los nodos: en primer lugar las cadenas con sus dos nodos extremos, seguidas de las ciudades aisladas.

En el ejemplo 3.1, $n = 6, K = 2$, el arreglo canónico de las ciudades es $\{3, 4, 1, 2, 5, 10\}$, con $z_1 = 3$, $z_2 = 4$; $z_3 = 1$, $z_4 = 2$ y $z_5 = 5$; $z_6 = 10$.

Ahora el problema se traduce en cómo conectar las cadenas con las otras cadenas y las ciudades aisladas. Dada una cadena, si el tour conecta por uno de sus extremos, debe salir por el otro extremo. Esta restricción debe satisfacerse para todas las cadenas, y, para la adecuada definición del modelo, se considerará que toda ciudad aislada ha de estar emparejada consigo misma. Esta idea se formaliza del siguiente modo.

Definición 3.1. Dado un problema $TSP_2^{\not k}$ con parámetros $(n, K, d^{\not k})$, para cualquier ciudad $z_h \in \{1, 2, \ldots, N\}$, con $h \in \{1, 2, \ldots, n\}$ se define su pareja o *couple* como:

$$z_h^c = \begin{cases} z_{2k-1} & \text{si } h = 2k \text{ para algún } k \in \{1, 2, \ldots, K\} \\ z_{2k} & \text{si } h = 2k - 1 \text{ para algún } k \in \{1, 2, \ldots, K\} \\ z_h & \text{si } h > 2K \end{cases}$$

En el ejemplo 3.1, las parejas son:

$$3^c = 4, \quad 4^c = 3, \quad 1^c = 2, \quad 2^c = 1, \quad 5^c = 5, \quad 10^c = 10$$

La nueva distancia $\mathbf{D}^{\not k} = (d^{\not k}_{x,y})_{x,y\in\{1,\ldots,n\}}$ es una matriz simétrica que sólo considera las distancias entre los extremos de las diferentes cadenas y las ciudades aisladas:

$$d^{\not k}_{xy} = \begin{cases} 0 & \text{si } x, y \in \{z_{2k-1}, z_{2k}\} \text{ para el mismo } k \in \{1, 2, \ldots, K\} \\ d_{xy} & \text{en otro caso} \end{cases}$$

En el ejemplo 3.1, si las nuevas ciudades se disponen como $\{3, 4, 1, 2, 5, 10\}$*, la matriz de distancias* $\mathbf{D}^{\not k}$ *queda de la forma:*

$$\mathbf{D}^{\not k} = \begin{bmatrix} 0 & 0 & d_{3,1} & d_{3,2} & d_{3,5} & d_{3,10} \\ 0 & 0 & d_{4,1} & d_{4,2} & d_{4,5} & d_{4,10} \\ d_{1,3} & d_{1,4} & 0 & 0 & d_{1,5} & d_{1,10} \\ d_{2,3} & d_{2,4} & 0 & 0 & d_{2,5} & d_{2,10} \\ d_{5,3} & d_{5,4} & d_{5,1} & d_{5,2} & 0 & d_{5,10} \\ d_{10,3} & d_{10,4} & d_{10,1} & d_{10,2} & d_{10,5} & 0 \end{bmatrix} = \begin{bmatrix} 0.00 & 0.00 & 0.82 & 0.80 & 0.60 & 0.70 \\ 0.00 & 0.00 & 0.85 & 0.75 & 0.50 & 0.80 \\ 0.82 & 0.85 & 0.00 & 0.00 & 0.74 & 0.35 \\ 0.80 & 0.75 & 0.00 & 0.00 & 0.40 & 0.60 \\ 0.60 & 0.50 & 0.74 & 0.40 & 0.00 & 9.99 \\ 0.70 & 0.80 & 0.35 & 0.60 & 9.99 & 0.00 \end{bmatrix}$$

El modelo propuesto añade restricciones adicionales: los extremos de cadena han de situarse consecutivamente en el tour final, ya que de otro modo se romperían las cadenas. Estas restricciones adicionales impiden que la variable de decisión $\mathbf{V}$ definida en la ecuación 2.12 ($v_{x,i} = 1$ si y sólo si la ciudad x se visita en la posición i, para $x, i \in \{1, 2, \ldots, n\}$) sea válida, como se muestra en el siguiente ejemplo:

En el ejemplo 3.1, las siguientes soluciones

$$\mathbf{V_1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{V_2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

son válidas, mientras que la solución

$$\mathbf{V_3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

no es válida, puesto que dos arcos entran en la cadena $(1, 6, 7, 8, 2)$*, lo que obligaría a romper la misma. En la figura 3.2 se muestran las tres soluciones* $\mathbf{V_1}$*,* $\mathbf{V_2}$ *y* $\mathbf{V_3}$*, donde se observa que claramente que* $\mathbf{V_3}$ *no puede ser una solución válida.*

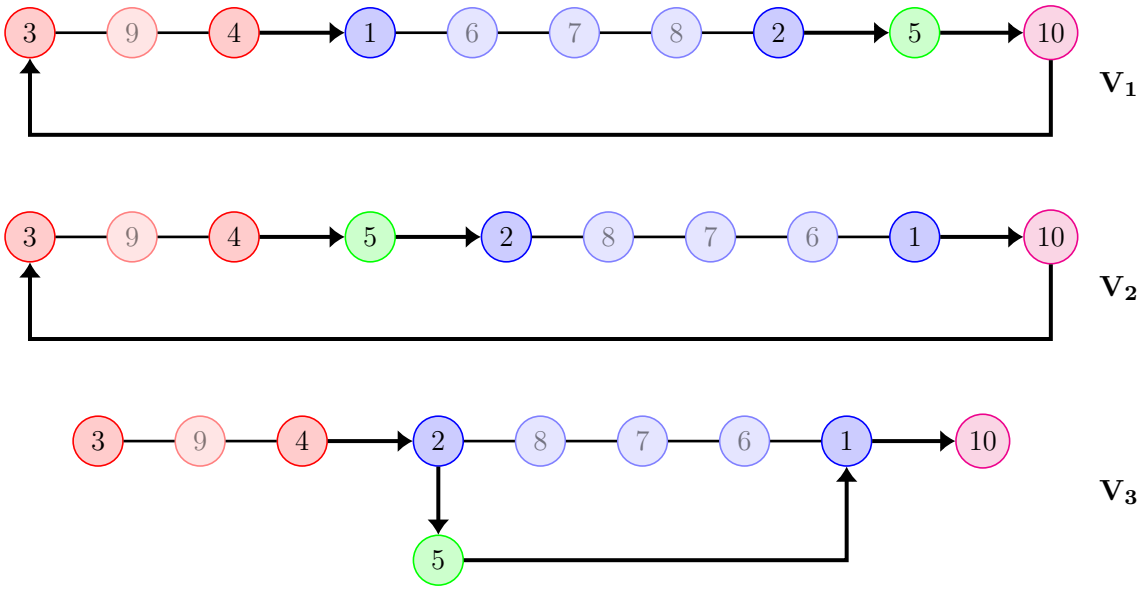


Figura 3.2: Dos posibles soluciones válidas $\mathbf{V_1}$, $\mathbf{V_2}$ y una solución inválida $\mathbf{V_3}$ del ejemplo 3.1

Teniendo en cuenta el análisis anterior y la definición 3.1, se propone la siguiente variable de decisión:

$$v_{x,i}^{\not k} = \begin{cases} 1 & \text{if } x \text{ es la ciudad de entrada, visitada en la posición } i, \\ & \text{y } x^c \text{ es la ciudad de salida} \\ 0 & \text{en otro caso} \end{cases} \tag{3.3}$$

para $x \in \{z_1, z_2, \ldots, z_n\}$ e $i \in \{1, 2, \ldots, n-K\}$.

Esta nueva variable de decisión es una matriz binaria de tamaño $n \times (n-K)$. Evidentemente, si $K = 0$ el problema $TSP_2^{\not k}$ se convierte en el TSP original con $n = N$. A medida que el parámetro K crece, el tamaño del problema $TSP_2^{\not k}$ decrece.

Volviendo al ejemplo 3.1, los dos tours válidos anteriores para el problema $TSP_2^{\not k}$ pueden caracterizarse, a partir de la nueva variable de decisión de la ecuación 3.3 como:

$$\mathbf{V_1^{\not k}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{V_2^{\not k}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

El modelo matemático binario del $TSP_2^{\not k}$

Con el objetivo de definir una nueva variable de estados adecuada, se considera la matriz $\mathbf{V}^{\not k} \in [0,1]^{n \times (n-K)}$, que debe verificar las siguientes restricciones:

- Cada posición del tour debe estar asociada con una única cadena o ciudad:

$$S_i^{\not k} = \sum_{x=1}^{n} v_{x,i}^{\not k} = 1 \qquad \forall i \in \{1, 2, \ldots, n-K\} \tag{3.4}$$

- Cada cadena debe ser visitada una única vez:

$$S_x^{\not k} = \sum_{i=1}^{n-K} (v_{x,i}^{\not k} + v_{x^c,i}^{\not k}) = 1 \qquad \forall x \in \{1, 3, \ldots, 2K-1\} \tag{3.5}$$

- Cada ciudad aislada debe ser visitada una única vez:

$$S_x^{\not k} = \sum_{i=1}^{n-K} v_{x,i}^{\not k} = 1 \qquad \forall x \in \{2K+1, 2K+2, \ldots, n\} \tag{3.6}$$

- El número total de ciudades visitadas es $n-K$:

$$\sum_{i=1}^{n-K} \sum_{x=1}^{n} v_{x,i}^{\not k} = n - K \tag{3.7}$$

Todo tour válido del $TSP_2^{\not k}$ se caracteriza por:

$$H_F^{\not k} = \left\{ V^{\not k} \in \{0,1\}^{n \times (n-K)} : \text{ se satisfacen las ecuaciones 3.4, 3.5, 3.6 y 3.7} \right\} \tag{3.8}$$

La función objetivo para $TSP_2^{\not k}$ es:

$$\text{mín} \left\{ \frac{1}{2} \sum_{x=1}^{n} \sum_{i=1}^{n-K} v_{x,i}^{\not k} \sum_{y \neq x} (v_{y,i-1}^{\not k} d_{y^c x}^{\not k} + v_{y,i+1}^{\not k} d_{x^c y}^{\not k}) \right\}$$

(los subíndices $i+1$ y $i-1$ vienen dados módulo $n-K$)

La estrategia *Divide-y-Vencerás* se ilustra en el siguiente ejemplo.

Ejemplo 3.2. *Sea $\{a, b, c, \ldots, r, s\}$ el conjunto de ciudades representado en la figura 3.3(a). Se presentan $N = 19$ ciudades, siendo las distancias entre ellas la euclídea. Se resuelve el problema TSP_1^{τ} utilizando el parámetro $\tau = 3$, resultando en $K = 4$ cadenas (las cadenas azules de la figura 3.3 (a)) y una ciudad aislada. El número de ciudades en el nuevo problema $TSP_2^{\not k}$ es ahora $n = 9$: dos ciudades por cada una de las cadenas ($\{[d, r]; [e, k]; [l, n]; [m, p]\}$) y una ciudad aislada ($\{o\}$). Se resuelve este problema y su solución se muestra en la figura 3.3 (b).*

A partir de la solución de la figura 3.3 (b), se expanden los extremos $\{\{d, r\}; \{e, k\}; \{l, n\}; \{m, p\}\}$ con las cadenas anteriormente calculadas para obtener la solución final del TSP original, que se muestra en la figura 3.4.

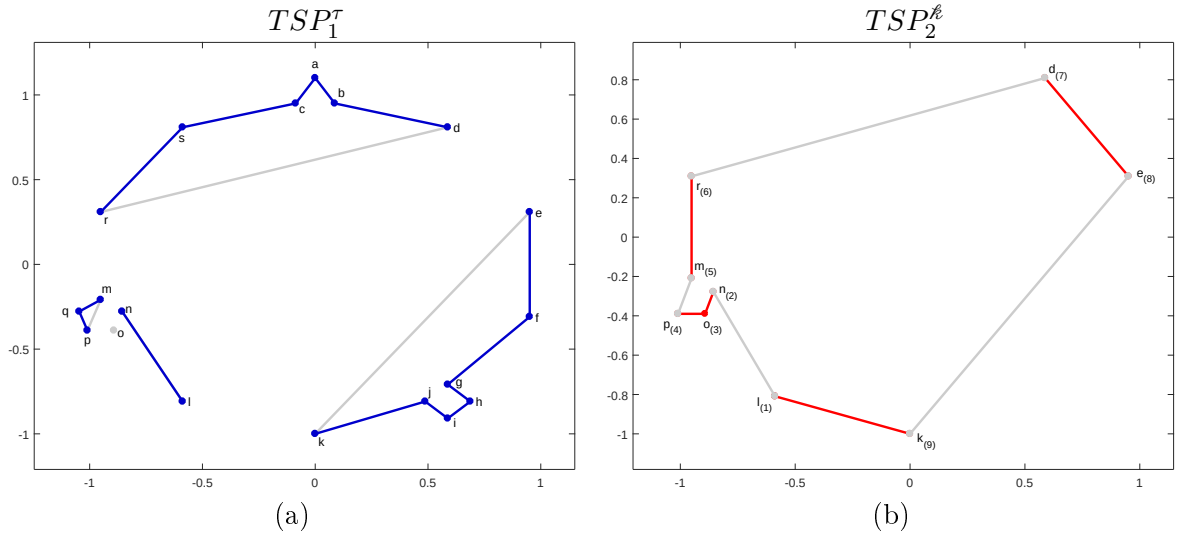


Figura 3.3: Las soluciones de las dos fases del ejemplo 3.2: TSP_1^τ (a) y $TSP_2^{\not k}$ (b)

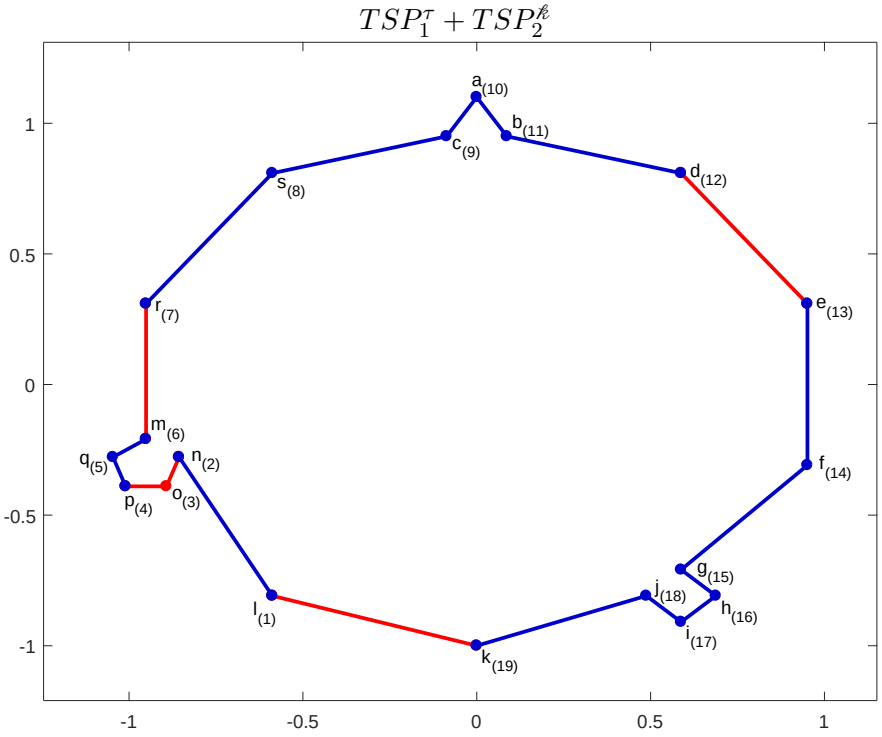


Figura 3.4: Solución del TSP para el ejemplo 3.2

3.2 Proyección del TSP_1^τ en el modelo de Hopfield

Cuando este problema de optimización, en nuestro caso el TSP_1^τ, se proyecta sobre una CHN, una solución estable de este sistema dinámico queda asociada con un tour válido del TSP_1^τ. Teniendo en cuenta que la función objetivo está incluida en la función de energía del sistema, este proceso puede considerarse como un procedimiento heurístico.

En el caso de TSP_1^τ, la matriz de variable de estados es $\mathbf{V} \in [0,1]^{N\times N}$, ver ecuación 2.12, las restricciones están definidas por las ecuaciones 2.13 y 2.14, y la función objetivo

está definida por la ecuación 3.2. De este modo, la función de energía de esta CHN es:

$$E(\mathbf{v}) = \frac{A}{2}\sum_{x}^{N}\sum_{i}^{N}\sum_{j\neq i}^{N} v_{x,i}v_{x,j} + \frac{B}{2}\sum_{i}^{N}\sum_{x}^{N}\sum_{y\neq x}^{N} v_{x,i}v_{y,i} + \frac{C}{2}\Big(\sum_{x}^{N}\sum_{i}^{N} v_{x,i} - N\Big)^2 + \frac{D}{2}\sum_{x}^{N}\sum_{y\neq x}^{N}\sum_{i}^{N} d^{\tau}_{x,y}v_{x,i}(v_{y,i-1} + v_{y,i+1}) \tag{3.9}$$

(los subíndices $i+1$ y $i-1$ vienen dados módulo N)

siendo el problema equivalente al de la sección 2.5, pero utilizando ahora la distancia modificada $d^{\tau}_{x,y}$ (ecuación 3.1).

De nuevo, desarrollando los cuatro términos de esta función de energía y comparándolos con la expresión general de la función de energía para la red de Hopfield:

$$E(\mathbf{v}) = -\frac{1}{2}\sum_{x,i}\sum_{y,j} v_{x,i}T_{xi,yj}v_{y,j} - \sum_{x,i} i^{b}_{x,i}v_{x,i}$$

se obtiene:

$$T_{xi,yj} = -\Big(A\delta_{x,y}(1-\delta_{i,j}) + B(1-\delta_{x,y})\delta_{i,j} + C + Dd^{\tau}_{x,y}(\delta_{i,j-1} + \delta_{i,j+1})\Big)$$

$$i^{b}_{x,i} = CN \tag{3.10}$$

con $x, y \in \{1, 2, \ldots, N\}$ e $i, j \in \{1, 2, \ldots, N\}$.

Como ya se analizó en la sección 2.5, la convergencia de la red neuronal a algún punto $\mathbf{V} \in [0,1]^{N\times N}$ está relacionada con la derivada parcial de su función de energía $E(\mathbf{v})$ con respecto a $v_{x,i}$. Así, teniendo en cuenta la ecuación 3.9, se puede calcular $E_{x,i}(\mathbf{v})$ como:

$$E_{x,i}(\mathbf{v}) = A\sum_{x}(S_x - v_{x,i}) + B\sum_{i}(S_i - v_{x,i}) + C(S-N) + Dd^{\tau}_{x,y}\sum_{y\neq x}(v_{y,i-1} + v_{y,i+1}), \qquad \forall x, i \in \{1, \ldots, N\} \tag{3.11}$$

Un punto $\mathbf{V} \in [0,1]^{N\times N}$, será un punto de equilibrio para la CHN del TSP_1^τ si y sólo si, se verifican las siguientes desigualdades:

$$\begin{aligned} &E_{x,i}(\mathbf{v}) \geq 0 \quad \forall (x,i) \in \{1,\ldots,N\}^2 \ / \ v_{x,i} = 0 \\ &E_{x,i}(\mathbf{v}) \leq 0 \quad \forall (x,i) \in \{1,\ldots,N\}^2 \ / \ v_{x,i} = 1 \\ &E_{x,i}(\mathbf{v}) = 0 \quad \forall (x,i) \in \{1,\ldots,N\}^2 \ / \ v_{x,i} \in (0,1) \end{aligned}$$

Recordando la definición introducida en la sección 2.5.1:

$$\underline{E}^0(\mathbf{v}) \equiv \min_{(x,i)/v_{x,i}=0} E_{x,i}(\mathbf{v})$$

$$\overline{E}^1(\mathbf{v}) \equiv \max_{(x,i)/v_{x,i}=1} E_{x,i}(\mathbf{v})$$

se puede concluir que cualquier punto $\mathbf{V}^{\not k} \in [0,1]^{N\times N}$ será un punto de equilibrio para la CHN si y sólo si se satisfacen las siguientes tres relaciones:

$$\underline{E}^0(\mathbf{v}) \geq 0 \tag{3.12}$$

$$\overline{E}^1(\mathbf{v}) \leq 0 \tag{3.13}$$

$$E_{x,i}(\mathbf{v}) = 0 \qquad \forall (x,i) \in \{1,\ldots,N\}^2 \ / \ v_{x,i} \in (0,1)$$

Cualquier punto $\mathbf{V} \in H_F = \left\{\mathbf{V} \in \{0,1\}^{N\times N} \ /\text{verifica las ecuaciones 2.13 y 2.14}\right\}$, caracterizando un tour factible, será un punto de equilibrio si y sólo si se verifican las ecuaciones 3.12 y 3.13. Este análisis es muy similar al realizado por Talaván y Yáñez [49]. En primer lugar, se necesita un nuevo parámetro $N' > N$ para que cualquier tour válido $\mathbf{V} \in H_F$ sea estable.

Este nuevo parámetro modifica la ecuación 3.10:

$$i^b_{x,i} = CN' \quad \forall x,i$$

y, consecuentemente, la ecuación 3.11 será sustituida por:

$$E_{x,i}(V) = A\sum_x (S_x - v_{x,i}) + B\sum_i (S_i - v_{x,i}) + C(S - N') + Dd^\tau_{x,y}\sum_{y\neq x}(v_{y,i-1} + v_{y,i+1})$$

Siguiendo la demostración de Talaván y Yáñez [49], o simplemente a partir del análisis de la sección 2.5.1 para las soluciones válidas e inválidas, se deducen condiciones suficientes para garantizar la estabilidad de las soluciones válidas y la inestabilidad de las soluciones inválidas para el TSP^τ_1. Los parámetros A, B, C, D y N' deben verificar las siguientes restricciones:

$$3Dd_U \leq C(N' - N) \leq \text{mín}\{B; A + Dd_L; (N-1)A\} - C$$

donde los parámetros N, d_L y d_U dependen de la instancia TSP^τ_1 ($d_L \leq d^\tau_{x,y} \leq d_U \quad \forall x,y \in \{1,2,\ldots,N\}$)

Esas restricciones son factibles, y se propone la siguiente parametrización para cualquier TSP^τ_1 que depende de un único parámetro $C > 0$:

1. $D = \frac{1}{d_U}$
2. $N' = N + \frac{3}{C}$
3. $B = C + 3$
4. $A = B - \frac{d_L}{d_U}$

Observación 3.1. *Para poder garantizar unas buenas propiedades computacionales en la convergencia de la CHN, y para evitar un número elevado de simetrías inducido por un alto*

número de valores iguales en la matriz de distancias, se modifica ligeramente el valor de d_U *en la definición de* $d^{\tau}_{x,y}$ *de la ecuación 3.1.*

$$d^{\tau}_{xy} = \begin{cases} & \text{si } y \text{ es una de las } \tau\text{-ciudades vecinas de } x \\ d_{xy} & \text{o} \\ & \text{si } x \text{ es una de las } \tau\text{-ciudades vecinas de } y \\ d_U - \left(\frac{d_L}{d_{x,y}} - \frac{d_L}{d_U}\right) & \text{en otro caso} \end{cases}$$

3.3 Proyección del $TSP_2^{\not k}$ en el modelo de Hopfield

La proyección del $TSP_2^{\not k}$ en la CHN es ligeramente diferente del caso mostrado en la sección anterior. Para cualquier estado $\mathbf{V}^{\not k} \in [0,1]^{n\times(n-K)}$, su función de energía es:

$$\begin{aligned} E(\mathbf{v}^{\not k}) = {} & \frac{A}{2}\sum_{x}^{n}\sum_{i}^{n-K}\sum_{j\neq i}^{n-K} v^{\not k}_{x,i}(v^{\not k}_{x,j} + v^{\not k}_{x^c,j}) + \frac{B}{2}\sum_{i}^{n-K}\sum_{x}^{n}\sum_{y\neq x}^{n} v^{\not k}_{x,i}v^{\not k}_{y,i} \\ & + \frac{C}{2}\Big(\sum_{x}^{n}\sum_{i}^{n-K} v^{\not k}_{x,i} - (n-K)\Big)^2 + \frac{D}{2}\sum_{x}^{n}\sum_{i}^{n-K}\sum_{y\neq x}^{n} v^{\not k}_{x,i}(v^{\not k}_{y,i-1}d^{\not k}_{y^c,x} + v^{\not k}_{y,i+1}d^{\not k}_{x^c,y}) \end{aligned}$$

(determinándose los subíndices $i+1$ e $i-1$ módulo $n-K$) (3.14)

Dada una matriz $\mathbf{V}^{\not k} \in H^{\not k} \equiv \{[0,1]^{n\times(n-K)}\}$, esta matriz será una solución válida para $TSP_2^{\not k}$ si $\mathbf{V}^{\not k} \in H_F^{\not k} \subset H^{\not k}$, que es equivalente a que todos los términos de $E(\mathbf{v}^{\not k})$, excepto el último, sean 0.

Desarrollando los cuatro términos de esta función de energía (ecuación 3.14) y comparándolos con la expresión general de la función de energía para la red de Hopfield:

$$E(\mathbf{v}^{\not k}) = -\frac{1}{2}\sum_{x,i}\sum_{y,j} v^{\not k}_{x,i}T_{xi,yj}v^{\not k}_{y,j} - \sum_{x,i} i^{b}_{x,i}v^{\not k}_{x,i} \tag{3.15}$$

se obtiene:

$$T_{xi,yj} = -\left(A\delta_{x,y}(1+\delta_{x,x^c})(1-\delta_{i,j}) + B(1-\delta_{x,y})\delta_{i,j} + C + D(\delta_{i,j-1}d^{\not k}_{y^c,x} + \delta_{i,j+1}d^{\not k}_{x^c,y})\right)$$

$$i^{b}_{x,i} = C(n-K) \tag{3.16}$$

con $x, y \in \{1, 2, \ldots, n\}$ y $i, j \in \{1, 2, \ldots, n-K\}$.

Una vez más, la convergencia de la red neuronal a algún punto $\mathbf{V}^{\not k} \in [0,1]^{n\times(n-K)}$ está directamente relacionada con la derivada parcial de la función de energía $E(\mathbf{v}^{\not k})$ con respecto a $v^{\not k}_{x,i}$ para todo x, i:

$$E_{x,i}(\mathbf{v}^{\not k}) \equiv \frac{\partial E(\mathbf{v}^{\not k})}{\partial v^{\not k}_{x,i}}$$

Teniendo en cuenta la ecuación 3.14, se puede calcular $E_{x,i}(\mathbf{v}^{\not k})$ como:

$$\begin{aligned} E_{x,i}(\mathbf{v}^{\not k}) = A\sum_{x}(S_x^{\not k} - v_{x,i}^{\not k} + (S_{x^c}^{\not k} - v_{x^c,i}^{\not k})) + B\sum_{i}(S_i^{\not k} - v_{x,i}^{\not k}) \\ + C(S^{\not k} - (n-K)) + D\sum_{y}(v_{y,i-1}^{\not k} d_{x,y^c}^{\not k} + v_{y,i+1}^{\not k} d_{x^c,y}^{\not k}) \end{aligned} \tag{3.17}$$

Un punto $\mathbf{V}^{\not k} \in [0,1]^{n\times(n-K)}$, será un punto de equilibrio para la CHN de $TSP_2^{\not k}$ si y sólo si se verifican las siguientes tres relaciones:

$$\begin{aligned} &E_{x,i}(\mathbf{v}^{\not k}) \geq 0, \qquad \forall (x,i) \in \{1,\dots,n\}\times\{1,\dots,(n-K)\} \ / \ v_{x,i}^{\not k} = 0 \\ &E_{x,i}(\mathbf{v}^{\not k}) \leq 0, \qquad \forall (x,i) \in \{1,\dots,n\}\times\{1,\dots,(n-K)\} \ / \ v_{x,i}^{\not k} = 1 \\ &E_{x,i}(\mathbf{v}^{\not k}) = 0, \qquad \forall (x,i) \in \{1,\dots,n\}\times\{1,\dots,(n-K)\} \ / \ v_{x,i}^{\not k} \in (0,1) \end{aligned}$$

Utilizando de nuevo la definición:

$$\begin{aligned} \underline{E}^0(\mathbf{v}^{\not k}) &\equiv \min_{(x,i)/v_{x,i}^{\not k}=0} E_{x,i}(\mathbf{v}^{\not k}) \\ \overline{E}^1(\mathbf{v}^{\not k}) &\equiv \max_{(x,i)/v_{x,i}^{\not k}=1} E_{x,i}(\mathbf{v}^{\not k}) \end{aligned}$$

se puede concluir que cualquier punto $\mathbf{V}^{\not k} \in [0,1]^{n\times(n-K)}$ será un punto de equilibrio para CHN si y sólo si se verifican las tres relaciones siguientes:

$$\underline{E}^0(\mathbf{v}^{\not k}) \geq 0 \tag{3.18}$$

$$\overline{E}^1(\mathbf{v}^{\not k}) \leq 0 \tag{3.19}$$

$$E_{x,i}(\mathbf{v}^{\not k}) = 0 \qquad \forall (x,i) \in \{1,\dots,n\}\times\{1,\dots,(n-K)\} \ / \ v_{x,i}^{\not k} \in (0,1)$$

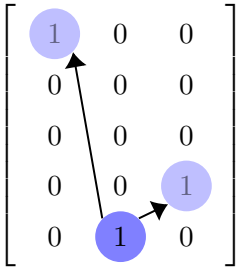
3.3.1 Convergencia de las soluciones válidas

Cualquier punto $\mathbf{V}^{\not k} \in H_F^{\not k}$ será un punto de equilibrio si y sólo si se verifican las desigualdades 3.18 y 3.19. Se analizan de modo exhaustivo los dos casos exclusivos siguientes:

1. Sea $v_{x,i}^{\not k} = 1$ un elemento del tour válido $\mathbf{V}^{\not k}$.

   Se han de considerar dos subcasos:

   a) $x > 2K$ (Una ciudad aislada, $x^c = x$)



   En este subcaso, teniendo en cuenta las ecuaciones 3.8 y 3.17, existen $y, z \neq x$ de

modo que $v^{\not k}_{y,i-1} = v^{\not k}_{z,i+1} = 1$ y, consecuentemente:

$$E_{x,i}(\mathbf{v}^{\not k}) = D(d^{\not k}_{x,y} + d^{\not k}_{x,z}) \geq 0 \quad \forall x, y \in \{1, 2, \ldots, n\}$$

De este modo, ningún tour $\mathbf{V}^{\not k} \in H_F^{\not k}$ puede ser un punto de equilibrio de la CHN. Este problema puede evitarse introduciendo un nuevo parámetro positivo N', que modifica todos los umbrales de la red. Así, la ecuación 3.16 queda reemplazada por:

$$i^b_{x,i} = CN' \quad \forall x \in \{1, \ldots, n\}, \forall i \in \{1, \ldots, (n-K)\} \tag{3.20}$$

Observación 3.2. *De ahora en adelante, con el umbral modificado en la ecuación 3.20, la derivada parcial de $E_{x,i}(\mathbf{v}^{\not k})$ se calcula como:*

$$\begin{aligned} E_{x,i}(\mathbf{v}^{\not k}) = A\sum_x (S^{\not k}_x - v^{\not k}_{x,i} + (S^{\not k}_{x^c} - v^{\not k}_{x^c,i})) + B\sum_i (S^{\not k}_i - v^{\not k}_{x,i}) \\ + C(S^{\not k} - N') + D\sum_y \left(v^{\not k}_{y,i-1} d^{\not k}_{x,y^c} + v^{\not k}_{y,i+1} d^{\not k}_{x^c,y} \right) \end{aligned}$$

donde $S^{\not k}_x$ y $S^{\not k}_i$ siguen las definiciones de las ecuaciones 3.4, 3.5, 3.6 y $S^{\not k} = \sum_x S^{\not k}_x = \sum_i S^{\not k}_i$.

Para un valor suficientemente grande de $N' > n - K$, la desigualdad:

$$E_{x,i}(\mathbf{v}^{\not k}) = C(n - K - N') + D(d^{\not k}_{x,y} + d^{\not k}_{x,z}) \leq 0 \quad \forall x, y, z \; \forall i$$

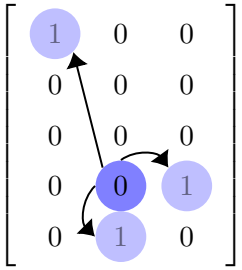
y la ecuación 3.19 se verificará siempre si se satisface la siguiente condición suficiente:

$$C(n - K - N') + 2\ D\ d_U \leq 0 \tag{3.21}$$

b) $x \leq 2K$ (El extremo de una cadena, $x^c \neq x$)

En este subcaso, la desigualdad anterior sigue siendo válida.

2. Sea $v^{\not k}_{x,i} = 0$ un elemento del tour $\mathbf{V}^{\not k} \in H_F^{\not k}$; entonces, para obtener una cota inferior de $E_{x,i}(\mathbf{v}^{\not k})$, no hay diferencias entre los casos de las cadenas y ciudades aisladas.



Se obtiene una cota inferior para $E_{x,i}(\mathbf{v}^{\not k})$ eligiendo un par (x, i) tal que $v^{\not k}_{x,i+1} = 1$ y la derivada parcial sea mayor o igual que cero, verificándose si:

$$A + B + C(n - K - N') + D\ d_L \geq 0 \tag{3.22}$$

Sean $d_U^{\not k}$ y $d_L^{\not k}$ respectivamente, las cotas superior e inferior de las distancias $d_{x,y}^{\not k}$, que están a su vez acotadas por d_U y d_L, de la distancia original $d_{x,y}$:

$$d_L \leq d_L^{\not k} \leq d_{x,y}^{\not k} \leq d_U^{\not k} \leq d_U \qquad \forall x, y \in \{1, \ldots, n\}$$

De este modo, a partir de las restricciones 3.21 y 3.22, se demuestra el siguiente teorema.

Teorema 3.1. *Si los coeficientes de la función de energía de la CHN verifican:*

$$C(n - K - N') + 2\ D\ d_U^{\not k} \leq 0$$

$$A + B + C\ (n - K - N') + D\ d_L^{\not k} \geq 0$$

Entonces cualquier tour $\mathbf{V}^{\not k} \in H_F^{\not k}$ *será un punto de equilibrio para la CHN.*

La demostración es equivalente a la del teorema 3.1 de Talaván y Yáñez [49].

Las dos condiciones anteriores pueden reducirse a la siguiente cadena de desigualdades:

$$2\ D\ d_U^{\not k} \leq C(N' - (n - K)) \leq A + B + D\ d_L^{\not k}$$

3.3.2 No convergencia de las soluciones inválidas

El análisis de la no convergencia de las soluciones inválidas para la CHN durante la fase $TSP_2^{\not k}$ se fundamenta en el trabajo de Talaván y Yáñez [49]. Siguiendo este artículo, se distinguen dos casos para una solución inválida $\mathbf{V}^{\not k} \in H^{\not k} \setminus H_F^{\not k}$: en los vértices del hipercubo de soluciones de Hamming $H_C^{\not k} \equiv \{0,1\}^{n \times (n-K)}$, y fuera de este conjunto.

En los vértices del hipercubo ($\mathbf{V}^{\not k} \in H_C^{\not k} \setminus H_F^{\not k}$)

Teorema 3.2. *Si los coeficientes de la función de energía son no-negativos y verifican:*

$$\text{mín}\{B; A + D\ d_L^{\not k}; (n - K - 1)A\} + C(n - K - N') > 0 \tag{3.23}$$

$$3\ D\ d_U^{\not k} + C(n - K - N') - C < 0 \tag{3.24}$$

Entonces, cualquier tour inválido $\mathbf{V}^{\not k} \in H_C^{\not k} \setminus H_F^{\not k}$ *no puede ser un punto de equilibrio.*

Esta demostración es equivalente a la del teorema 4.1 de Talaván y Yáñez [49].

Puntos interiores ($\mathbf{V}^{\not k} \in H^{\not k} \setminus H_C^{\not k}$)

El análisis de estos puntos es similar al del artículo de Talaván y Yáñez [49]. Las soluciones inválidas y las aristas estables deben ser evitadas mediante las restricciones de las ecuaciones 3.23 y 3.24, que son reemplazadas respectivamente por:

$$C(n - K - N') + \text{mín}\{B; A + Dd_L^{\not k}; (n - K - 1)A\} \geq C$$

$$3Dd_U^{\not k} + C(n - K - N') - C \leq -C$$

3.3.3 Parametrización

Como consecuencia de los resultados anteriores, se satisface la siguiente cadena de desigualdades:

$$3Dd_U^{\not k} \leq C(N' - n + K) \leq \min\{B; A + Dd_L^{\not k}; (n - K - 1)A\} - C \tag{3.25}$$

entonces, cualquier solución estable $\mathbf{V}^{\not k} \in H_C^{\not k}$ será una solución válida del $TSP_2^{\not k}$, y cualquier punto interior $\mathbf{V}^{\not k} \notin H_C^{\not k}$ será un punto inestable.

Observación 3.3. *Para el caso $K = 0$, la anterior cadena de desigualdades se reduce a la introducida en el artículo de Talaván y Yáñez [49].*

Se propone la siguiente parametrización, dependiente de un único parámetro $C > 0$:

$$\begin{aligned} D &= \tfrac{1}{d_U^{\not k}} \\ N' &= n - K + \tfrac{3}{C} \\ A &= 3 + C \\ B &= A + \tfrac{d_L^{\not k}}{d_U^{\not k}} \end{aligned} \tag{3.26}$$

Observación 3.4. *Fijándose en la desigualdad 3.25, la parametrización anterior se obtiene:*

1. *Igualando el término de la izquierda de la desigualdad, $3Dd_U^{\not k}$, y el término central, $C(N' - n + K)$.*
2. *Igualando el primer término del mínimo, B, y segundo, $A + Dd_L^{\not k}$.*
3. *Igualando el tercer término del mínimo, $(n - K - 1)A - C$ y el primer término de la desigualdad, $3Dd_U^{\not k}$. Para encontrar una parametrización válida para el caso general en el que $K > 0$, ésta debe tener en cuenta que el $\min\{B; A + Dd_L^{\not k}; (n - K - 1)A\}$ alcanza su menor valor en el tercer término, cuando $n - K - 1 = 1$ (e.g. el caso de una única cadena con tres ciudades y una ciudad aislada, $N = 4, K = 1, n = 3$).*

Siguiendo las reglas anteriores, se calcula la parametrización para el ejemplo 3.1 ($d_L^{\not k} = 0.35; d_U^{\not k} = 9.99$) para distintos valores de C:

C	D	N'	A	B
1	0.1001	11	4.0	4.035
0.001	0.1001	3008	3.001	3.036

CAPÍTULO 4

El modelo de Hopfield como 2-opt

Este capítulo presenta, partiendo de los resultados del capítulo anterior, el comportamiento del modelo de Hopfield como algoritmo 2-opt, en particular, como intercambio 2-opt cuando dadas 4 ciudades se fijan 2 cadenas. Este resultado ayudará a mejorar notablemente la calidad de las soluciones obtenidas por el modelo de Hopfield al resolver el TSP, resultados que se presentan en el capítulo 5.

Contenidos del capítulo

4.1 Análisis de las cuencas de atracción de la CHN aplicado al $TSP_2^{\not k}$

De un modo similar al análisis de las cuencas de atracción de la CHN aplicado al TSP que se abordó en la sección 2.5, se estudia a continuación la segunda fase del procedimiento Divide-y-Vencerás, denominada $TSP_2^{\not k}$, introducida en la sección 3.3.

En este caso, el objetivo es poder entender y calcular el punto de silla para la fase $TSP_2^{\not k}$, es decir, cuando se han fijado un conjunto de K cadenas. Ello permitirá determinar si es posible (y cuánto) mover el punto de silla, de modo que una adecuada selección del parámetro libre en la parametrización 3.26 favorezca la búsqueda de mejores soluciones. La siguiente proposición permite calcular dicho punto de silla y es además una generalización de la proposición 2.2.

Proposición 4.1. *Dado un TSP con n ciudades en el que se han fijado K cadenas, el punto de silla $\mathbf{v}^*$ de la función de energía para la segunda fase ($TSP_2^{\not k}$) del procedimiento Divide-y-Vencerás (definida en la ecuación 3.15 utilizando la entrada externa modificada de la ecuación 3.20) es:*

$$v^*_{x,i} = C\left(n - K + \frac{3}{C}\right)\sum_{j=1}^{n} m_{x,j}, \quad \forall x \in \{1,\dots,n\}, \forall i \in \{1,\dots,n-K\}$$

con

$$\mathbf{M} = \left[\left((n-K+1)\,C+3+\frac{d_L}{d_U}\right)\mathbb{1} + (n-K-1)\,(C+3)\,\mathbf{I_K} - \left(C+3+\frac{d_L}{d_U}\right)\mathbf{I} + \frac{1}{d_U}\left(\mathbf{D_K}+(\mathbf{D_K})^T\right)\right]^{-1}$$

donde

$$\mathbb{1} = (\mathbb{1})_n \equiv \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}_{n\times n}, \qquad \mathbf{I} = (\mathbf{I})_n \equiv \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}_{n\times n},$$

$$\mathbf{J_K} = (\mathbf{J_K})_n \equiv \begin{bmatrix} (\mathbb{1})_2 - (\mathbf{I})_2 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & (\mathbb{1})_2 - (\mathbf{I})_2 & 0 \\ 0 & \cdots & 0 & (\mathbf{I})_{n-2K} \end{bmatrix}_{n\times n},$$

$\mathbf{I_K} = (\mathbf{I_K})_n \equiv \delta_K \cdot \mathbf{J_K} + \mathbf{I}$, *donde* $\delta_K = \begin{cases} 0 & si \quad K = 0 \\ 1 & si \quad K > 0 \end{cases}$ *y* $\mathbf{D_K} = (\mathbf{D_K})_n \equiv \mathbf{D} \cdot \mathbf{J_K}$
con $\mathbf{D} = (\mathbf{D})_n \equiv (d_{x,y})_{x,y\in\{1,\dots,n\}}$

Se indican, para evitar confusión, los tamaños de las matrices cuadradas utilizando paréntesis.

Demostración. El punto de silla de la función de energía para la fase 2 ($TSP_2^{\not k}$) del procedimiento Divide-y-Vencerás, definida en la ecuación 3.15, se obtiene calculando las derivadas

parciales de $E(\mathbf{v})$ con respecto a $\mathbf{v}$ e igualándolas a cero, $-\mathbf{T}\mathbf{v} - \mathbf{i}^b = 0$. Por tanto, el punto de silla $\mathbf{v}^*$ es:

$$\mathbf{v}^* = -\mathbf{T}^{-1}\mathbf{i}^b$$

La matriz $\mathbf{T}$ puede escribirse como una matriz $(n \times (n-K)) \times (n \times (n-K))$ (matriz con $(n-K) \times (n-K)$ bloques de matrices cuadradas $n \times n$) del siguiente modo:

$$\mathbf{T} = -\begin{bmatrix} (B+C)\mathbb{1}-B\mathbf{I} & C\mathbb{1}+A\mathbf{I_K}+D(\mathbf{D_K})^T & C\mathbb{1}+A\mathbf{I_K} & \cdots & C\mathbb{1}+A\mathbf{I_K} & C\mathbb{1}+A\mathbf{I_K}+D\mathbf{D_K} \\ C\mathbb{1}+A\mathbf{I_K}+D\mathbf{D_K} & \ddots & \ddots & \ddots & \ddots & C\mathbb{1}+A\mathbf{I_K} \\ C\mathbb{1}+A\mathbf{I_K} & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & C\mathbb{1}+A\mathbf{I_K} \\ C\mathbb{1}+A\mathbf{I_K} & \ddots & \ddots & \ddots & \ddots & C\mathbb{1}+A\mathbf{I_K}+D(\mathbf{D_K})^T \\ C\mathbb{1}+A\mathbf{I}+(D\mathbf{D_K})^T & C\mathbb{1}+A\mathbf{I_K} & \cdots & C\mathbb{1}+A\mathbf{I_K} & C\mathbb{1}+A\mathbf{I_K}+D\mathbf{D_K} & (B+C)\mathbb{1}-B\mathbf{I} \end{bmatrix} \tag{4.1}$$

donde las matrices $\mathbb{1}$, $\mathbf{I}$, $\mathbf{I_K}$ y $\mathbf{D_K}$ siguen las definiciones del enunciado. Con el fin de poder considerar el caso trivial $K = 0$ en el que no hay cadenas fijadas y todas las ciudades aisladas son parejas de sí mismas, se introduce el término δ_K en la definición de la matriz $\mathbf{I_K}$.

Además, el vector de umbrales $\mathbf{i^b}$ tiene la estructura:

$$\mathbf{i}^b = \begin{bmatrix} CN' \\ \vdots \\ CN' \end{bmatrix}_{(n\times(n-K))\times 1}$$

$\mathbf{T}$ es una matriz circulante por bloques (ver De Mazancourt et al. [7]). Si $\mathbf{T}$ es invertible, utilizando la proposición 2.1, $\mathbf{T}^{-1}$ es también circulante por bloques y tiene la siguiente estructura, en función de si $n-K$ es par o impar:

Si $n - K = 2m$:

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{Q_1} & \mathbf{Q_2} & \mathbf{Q_3} & \cdots & \mathbf{Q_{\frac{n-K}{2}-1}} & \mathbf{Q_{\frac{n-K}{2}}} & \mathbf{Q_{\frac{n-K}{2}+1}} & \mathbf{Q_{\frac{n-K}{2}}} & \mathbf{Q_{\frac{n-K}{2}-1}} & \cdots & \mathbf{Q_3} & \mathbf{Q_2} \\ \mathbf{Q_2} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \mathbf{Q_3} \\ \mathbf{Q_3} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \mathbf{Q_{\frac{n-K}{2}-1}} \\ \mathbf{Q_{\frac{n-K}{2}-1}} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \mathbf{Q_{\frac{n-K}{2}}} \\ \mathbf{Q_{\frac{n-K}{2}}} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \mathbf{Q_{\frac{n-K}{2}+1}} \\ \mathbf{Q_{\frac{n-K}{2}+1}} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \mathbf{Q_{\frac{n-K}{2}}} \\ \mathbf{Q_{\frac{n-K}{2}}} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \mathbf{Q_{\frac{n-K}{2}-1}} \\ \mathbf{Q_{\frac{n-K}{2}-1}} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \mathbf{Q_3} \\ \mathbf{Q_3} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \mathbf{Q_2} \\ \mathbf{Q_2} & \mathbf{Q_3} & \cdots & \mathbf{Q_{\frac{n-K}{2}-1}} & \mathbf{Q_{\frac{n-K}{2}}} & \mathbf{Q_{\frac{n-K}{2}+1}} & \mathbf{Q_{\frac{n-K}{2}}} & \mathbf{Q_{\frac{n-K}{2}-1}} & \cdots & \mathbf{Q_3} & \mathbf{Q_2} & \mathbf{Q_1} \end{bmatrix}$$

donde $\mathbf{Q_s}$ son matrices $n \times n$ $\forall s \in \{1, \ldots, \frac{n-K}{2} + 1\}$.

Si $n - K = 2m + 1$:

$$\mathbf{T}^{-1} = \begin{bmatrix}
\mathbf{Q_1} & \mathbf{Q_2} & \mathbf{Q_3} & \cdots & \mathbf{Q_{\frac{n-K-3}{2}}} & \mathbf{Q_{\frac{n-K-1}{2}}} & \mathbf{Q_{\frac{n-K+1}{2}}} & \mathbf{Q_{\frac{n-K+1}{2}}} & \mathbf{Q_{\frac{n-K-1}{2}}} & \mathbf{Q_{\frac{n-K-3}{2}}} & \cdots & \mathbf{Q_3} & \mathbf{Q_2} \\
\mathbf{Q_2} & & & & & & & & & & & & \mathbf{Q_3} \\
\mathbf{Q_3} & & & & & & & & & & & & \vdots \\
\vdots & & & & & & & & & & & & \mathbf{Q_{\frac{n-K-3}{2}}} \\
\mathbf{Q_{\frac{n-K-3}{2}}} & & & & & & & & & & & & \mathbf{Q_{\frac{n-K-1}{2}}} \\
\mathbf{Q_{\frac{n-K-1}{2}}} & & & & & & & & & & & & \mathbf{Q_{\frac{n-K+1}{2}}} \\
\mathbf{Q_{\frac{n-K+1}{2}}} & & & & & & & & & & & & \mathbf{Q_{\frac{n-K+1}{2}}} \\
\mathbf{Q_{\frac{n-K+1}{2}}} & & & & & & & & & & & & \mathbf{Q_{\frac{n-K-1}{2}}} \\
\mathbf{Q_{\frac{n-K-1}{2}}} & & & & & & & & & & & & \mathbf{Q_{\frac{n-K-3}{2}}} \\
\mathbf{Q_{\frac{n-K-3}{2}}} & & & & & & & & & & & & \vdots \\
\vdots & & & & & & & & & & & & \mathbf{Q_3} \\
\mathbf{Q_3} & & & & & & & & & & & & \mathbf{Q_2} \\
\mathbf{Q_2} & \mathbf{Q_3} & \cdots & \mathbf{Q_{\frac{n-K-3}{2}}} & \mathbf{Q_{\frac{n-K-1}{2}}} & \mathbf{Q_{\frac{n-K+1}{2}}} & \mathbf{Q_{\frac{n-K+1}{2}}} & \mathbf{Q_{\frac{n-K-1}{2}}} & \mathbf{Q_{\frac{n-K-3}{2}}} & \cdots & \mathbf{Q_3} & \mathbf{Q_2} & \mathbf{Q_1}
\end{bmatrix}$$

donde $\mathbf{Q_s}$ son matrices $(n-K) \times (n-K)$ $\forall s \in \{1, \ldots, \frac{n-K+1}{2}\}$.

Teniendo en cuenta la ecuación 2.26 y la estructura para $\mathbf{T}^{-1}$, $\mathbf{v}^*$ puede calcularse como[3]:

$$\mathbf{v}^* = -\mathbf{T}^{-1}\mathbf{i}^b = \begin{bmatrix} \mathbf{v}^*_{\cdot 1} \\ \mathbf{v}^*_{\cdot 2} \\ \vdots \\ \mathbf{v}^*_{\cdot (n-K)} \end{bmatrix}_{(n\times(n-K))\times 1} \tag{4.2}$$

con

$$\mathbf{v}^*_{\cdot i} = \begin{cases} -\Big(\mathbf{Q_1} + 2\sum_{s=2}^{\frac{n-K}{2}} \mathbf{Q_s} + \mathbf{Q_{\frac{n-K}{2}+1}}\Big) CN'\mathbf{1} & \text{si} \quad n-K = 2m \\ -\Big(\mathbf{Q_1} + 2\sum_{s=2}^{\frac{n-K+1}{2}} \mathbf{Q_s}\Big) CN'\mathbf{1} & \text{si} \quad n-K = 2m+1 \end{cases}$$

donde $\mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{n\times 1}$

[3] Nótese que la matriz de estados $\mathbf{V} = (v_{x,i})_{x\in\{1,\ldots,n\}, i\in\{1,\ldots,n-K\}}$ puede escribirse como vector de la forma: $\mathbf{v} = \begin{bmatrix} \mathbf{v}_{\cdot 1} \\ \mathbf{v}_{\cdot 2} \\ \vdots \\ \mathbf{v}_{\cdot (n-K)} \end{bmatrix}_{(n\times(n-K))\times 1}$ con $\mathbf{v}_{\cdot i} = \begin{bmatrix} v_{1,i} \\ v_{2,i} \\ \vdots \\ v_{n,i} \end{bmatrix}_{n\times 1}$. Ver más detalles acerca de la notación en la página XXI.

$$\text{Sea } \mathbf{Q}^{row} \equiv \begin{cases} \mathbf{Q_1} + 2\sum_{s=2}^{\frac{n-K}{2}} \mathbf{Q_s} + \mathbf{Q}_{\frac{\mathbf{n-K}}{\mathbf{2}}+\mathbf{1}} & \text{si} \quad n-K=2m \\ \mathbf{Q_1} + 2\sum_{s=2}^{\frac{n-K+1}{2}} \mathbf{Q_s} & \text{si} \quad n-K=2m+1 \end{cases}$$

entonces, teniendo en cuenta que $\mathbf{Tv}^* = \mathbf{i}^b$ y definiendo $\mathbf{q}^{row} \equiv \mathbf{Q}^{row}\mathbf{1}$:

$$\mathbf{Tv}^* = \mathbf{T}\begin{bmatrix} \mathbf{v}^*_{\cdot 1} \\ \mathbf{v}^*_{\cdot 2} \\ \vdots \\ \mathbf{v}^*_{\cdot (n-K)} \end{bmatrix}_{(n\times(n-K))\times 1} = \mathbf{T}\begin{bmatrix} -\mathbf{Q}^{row}CN'\mathbf{1} \\ -\mathbf{Q}^{row}CN'\mathbf{1} \\ \vdots \\ -\mathbf{Q}^{row}CN'\mathbf{1} \end{bmatrix}_{(n\times(n-K))\times 1}$$

$$= -CN'\mathbf{T}\begin{bmatrix} \mathbf{q}^{row} \\ \mathbf{q}^{row} \\ \vdots \\ \mathbf{q}^{row} \end{bmatrix}_{(n\times(n-K))\times 1} = \begin{bmatrix} CN' \\ CN' \\ \vdots \\ CN' \end{bmatrix}_{(n\times(n-K))\times 1}$$

Puesto que todas las componentes son iguales, basta con desarrollar una de ellas:

$$-CN'\left(-\left[(B+C)\,\mathbb{1} - B\mathbf{I} + (n-K-3)\,(C\mathbb{1}+A\mathbf{I_K}) + 2\,(C\mathbb{1}+A\mathbf{I_K}) + D\left(\mathbf{D_K}+(\mathbf{D_K})^T\right)\right]\right)\mathbf{q}^{row} = CN'\mathbf{1}$$

obteniendo:

$$\begin{aligned} \mathbf{q}^{row} &= \left[(B+C)\,\mathbb{1} - B\mathbf{I} + (n-K-3)\,(C\mathbb{1}+A\mathbf{I_K}) + 2\,(C\mathbb{1}+A\mathbf{I_K}) + D\left(\mathbf{D_K}+(\mathbf{D_K})^T\right)\right]^{-1}\mathbf{1} \\ &= \left[(B+(n-K)C)\mathbb{1} + (n-K-1)A\mathbf{I_K} - B\mathbf{I} + D\left(\mathbf{D_K}+(\mathbf{D_K})^T\right)\right] \end{aligned}$$

El punto de silla $\mathbf{v}^*$ de la función de energía $E(\mathbf{v})$ dada en la ecuación 3.14 en términos de los parámetros A, B, C, D y N' es:

$$\mathbf{v}^* = \begin{bmatrix} \mathbf{v}^*_{\cdot 1} \\ \mathbf{v}^*_{\cdot 2} \\ \vdots \\ \mathbf{v}^*_{\cdot (n-K)} \end{bmatrix}_{(n\times(n-K))\times 1}$$

donde

$$\begin{aligned} \mathbf{v}^*_{\cdot i} &= CN'\mathbf{q}^{row} \\ &= CN'\left[(B+(n-K)C)\mathbb{1} + (n-K-1)A\mathbf{I_K} - B\mathbf{I} + D\left(\mathbf{D_K}+(\mathbf{D_K})^T\right)\right]^{-1}\mathbf{1} \\ &\forall i \in \{1,\ldots,n-K\} \end{aligned}$$

que puede escribirse de modo equivalente como:

$$v^*_{x,i} = CN'\sum_{j=1}^{n} m_{x,j}, \quad \forall x \in \{1,\ldots,n\}, \forall i \in \{1,\ldots,n-K\}, \tag{4.3}$$

$$\mathbf{M} = \left[(B+(n-K)C)\mathbb{1} + (n-K-1)A\mathbf{I_K} - B\mathbf{I} + D\left(\mathbf{D_K}+(\mathbf{D_K})^T\right)\right]^{-1} \tag{4.4}$$

Teniendo en cuenta la parametrización de la ecuación 3.26 para los parámetros A, B, C, D y N', se tiene que:

$$v^*_{x,i} = C\left(n - K + \frac{3}{C}\right)\sum_{j=1}^{n} m_{x,j}, \quad \forall x \in \{1,\ldots,n\}, \forall i \in \{1,\ldots,n-K\},$$

con

$$\mathbf{M} = \left[\left((n{-}K{+}1)\,C{+}3 + \frac{d_L}{d_U}\right)\mathbb{1} + (n{-}K{-}1)\,(C{+}3)\,\mathbf{I_K} - \left(C + 3 + \frac{d_L}{d_U}\right)\mathbf{I} + \frac{1}{d_U}\left(\mathbf{D_K}{+}(\mathbf{D_K})^T\right)\right]^{-1}$$

□

Observación 4.1. *Pese a que la proposición 4.1 es una generalización de la proposición 2.2 (es decir, si $K = 0$, entonces la demostración se reduce a la demostración de la proposición 2.2), los resultados pueden no parecer equivalentes debido a la utilización de una parametrización ligeramente diferente en el TSP sin fijar cadenas, la cual modifica ligeramente la función de energía y, por consiguiente, el punto de silla. Esto es así dado que se quieren poder comparar resultados con los trabajos de otros autores, Talaván y Yáñez [49] entre otros. Esta generalización puede verificarse comparando las ecuaciones 2.29 y 2.30 con las ecuaciones 4.3 y 4.4 (tomando $K = 0$), respectivamente.*

Observación 4.2. *Al igual que se destacó a través la observación 2.2 que no era necesario invertir la matriz de pesos $\mathbf{T}$ para calcular el punto de silla de la proposición 2.2, de nuevo, cuando se han fijado K cadenas en la proposición 4.1, puede obtenerse su punto de silla sin necesidad de calcular la inversa de la matriz de pesos de tamaño $(n\times(n-K))\times(n\times(n-K))$, siendo suficiente calcular la inversa de la matriz $\mathbf{M}$, de tamaño $n \times n$*

Observación 4.3. *Nótese que la matriz $\mathbf{T}$ (ver ecuación 4.1) de la demostración de la proposición 4.1 tiene, para el caso extremo $n - K = 2$, una estructura ligeramente diferente:*

$$\mathbf{T} = \begin{bmatrix} (B{+}C)\mathbb{1}{-}B\mathbf{I} & C\mathbb{1}{+}A\mathbf{I_K}{+}D(\mathbf{D_2}{+}(\mathbf{D_2})^T) \\ C\mathbb{1}{+}A\mathbf{I_2}{+}D(\mathbf{D_2}{+}(\mathbf{D_2})^T) & (B{+}C)\mathbb{1}{-}B\mathbf{I} \end{bmatrix}$$

Este resultado se utilizará má adelante en la demostración del lema 4.2.

4.2 La heurística 2-opt

El *2-opt* consiste en una heurística de optimización local introducida por Croes [5] en 1958 para resolver el TSP. Partiendo de una solución inicial del TSP y considerando la distancia euclídea, el objetivo consiste en mejorar la solución eliminando los posible cruces que se puedan haber originado en la solución inicial.

El procedimiento consiste en considerar todo par de arcos no consecutivos de la siguiente forma: dado un par de arcos, si son eliminados, existe un único modo de reconectar los mismos de tal manera que se obtenga un nuevo tour válido. Este movimiento se conoce en ocasiones como *intercambio 2-opt*. Si la longitud del nuevo tour obtenido es menor que la longitud del tour original, entonces se decide reemplazar el tour original por el nuevo tour. En caso contrario, se mantiene el tour original. Repetido este proceso para cada par de arcos no adyacentes, cuando no se puede mejorar la solución obtenida, se dice que la solución es 2-óptima y el proceso termina.

La figura 4.1 muestra un ejemplo en el que partiendo de una solución inicial, se deshacen los cruces para obtener una solución con menor valor en la función objetivo, alcanzando la solución óptima tras dos *intercambios 2-opt*.

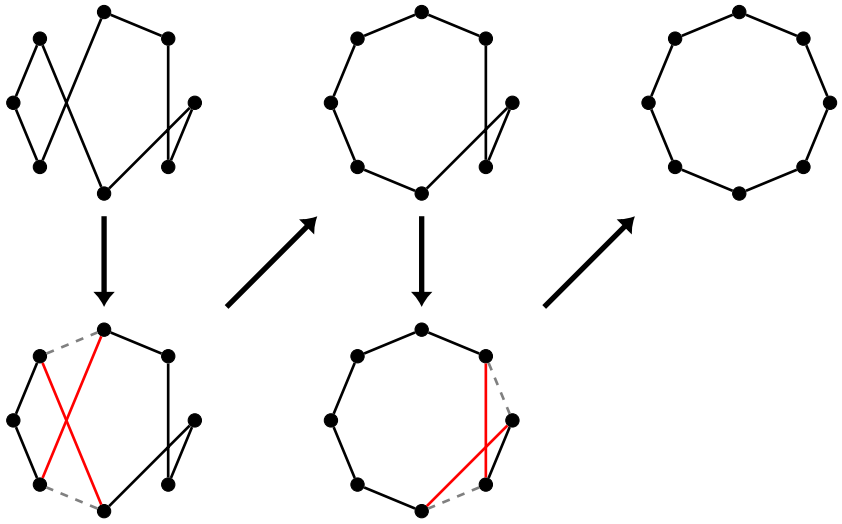


Figura 4.1: Mejora de solución inicial utilizando la heurística 2-opt

La heurística *2-opt* es un algoritmo de búsqueda de óptimos locales siendo altamente dependiente de la solución inicial. Nótese en la figura 4.2 las tres soluciones óptimas propuestas, con dos óptimos locales (izquierda y centro) en los que la heurística queda atrapada y el óptimo global (derecha).

El *2-opt* es en realidad un caso particular del *k-opt*, heurística introducida por Lin y Kernighan [28]. En el caso del *k-opt*, a partir de una solución del TSP, se eliminan k aristas mutuamente disjuntas. Los subtours obtenidos se combinan de la mejor manera posible, llamando a este movimiento *intercambio k-opt*.

El algoritmo de la heurística *2-opt* se resume a través del siguiente pseudo-código:

Observación 4.4. *Nótese que no es necesario calcular con cada intercambio 2-opt el tour total de cara a valorar si el intercambio es favorable para obtener una mejor solución. Bastará*

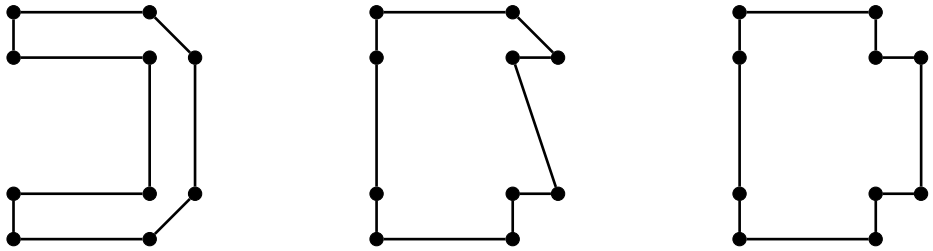


Figura 4.2: Óptimos locales (izq. y centro) y global (derecha), resultado de aplicar el *2-opt*

Algoritmo 1 Heurística 2-opt

```
procedure 2-OPT(N, tour, D)
    improvement = 0
    bestTourLength = computeTourLength(tour, D)
    do
        for i = 1 to N do
            for j = i + 2 to N do
                a = tour[ i ]
                b = tour[ (j + 1) mod N ]
                c = tour[ j ]
                d = tour[ (i + 1) mod N ]
                newImprovement =
                if newImprovement > improvement then
                    improvement = newImprovement
                    best = [ a, c ]
                end if
            end for
        end for
        if improvement > 0 then
            bestTourLength = bestTourLength − improvement
            tour = swap2opt(tour, best[1], best[2])
        end if
    while improvement ≠ 0
    return: tour
end procedure
```

con comparar, en cada caso, la suma de las dos distancias.

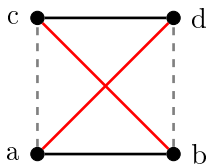


Figura 4.3: Intercambio 2-opt

produciéndose, en este caso, el *intercambio 2-opt* al verificarse que $d_{a,c} + d_{b,d} < d_{a,d} + d_{b,c}$.

4.3 El modelo de Hopfield como algoritmo 2-opt

Se plantea a continuación el modelo de Hopfield con $n = 4$ y $K = 2$, es decir, un problema con 2 cadenas y 4 ciudades (sus extremos). En esta sección, se estudiará el comportamiento de este CHN como *2-opt*. Para ello, son necesarios algunos resultados previos.

Lema 4.1. *Dada una matriz* $\mathbf{X}$*, definida positiva, de tamaño* 4×4 *con la siguiente estructura:*

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_2 & x_1 & x_4 & x_3 \\ x_3 & x_4 & x_1 & x_2 \\ x_4 & x_3 & x_2 & x_1 \end{bmatrix}_{4\times4}$$

entonces, su matriz inversa $\mathbf{M}$ *verifica que*

$$\sum_{j=1}^{4} m_{i,j} = \frac{1}{x_1 + x_2 + x_3 + x_4}, \quad \forall i = 1, 2, 3, 4.$$

Demostración. Al ser $\mathbf{X}$ definida positiva, existe su matriz inversa. Sea $\mathbf{M}$ la matriz inversa de $\mathbf{X}$. Por ser su inversa, se verifica que

$$\mathbf{m}_{i,\cdot}.\mathbf{x}_{\cdot,j} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{resto} \end{cases}$$

siendo

$$\mathbf{m}_{i,\cdot} = \begin{bmatrix} m_{i,1} & m_{i,2} & m_{i,3} & m_{i,4} \end{bmatrix}, \quad \forall i = 1, 2, 3, 4.$$

y

$$\mathbf{x}_{\cdot,j} = \begin{bmatrix} x_{1,j} \\ x_{2,j} \\ x_{3,j} \\ x_{4,j} \end{bmatrix}, \quad \forall j = 1, 2, 3, 4.$$

de modo que se tiene que:

$$\sum_{j=1}^{4} \mathbf{m}_{i,\cdot}.\mathbf{x}_{\cdot,j} = \mathbf{m}_{i,\cdot} \sum_{j=1}^{4} \mathbf{x}_{\cdot,j} = 1, \quad \forall i = 1, 2, 3, 4.$$

Teniendo en cuenta la definición de $\mathbf{X}$ del enunciado, se tiene

$$\sum_{j=1}^{4} \mathbf{x}_{\cdot,j} = \begin{bmatrix} x_1 + x_2 + x_3 + x_4 \\ x_1 + x_2 + x_3 + x_4 \\ x_1 + x_2 + x_3 + x_4 \\ x_1 + x_2 + x_3 + x_4 \end{bmatrix}$$

que no es el vector $\mathbf{0}$ por ser $\mathbf{X}$ invertible.

Así,

$$\mathbf{m}_{i,\cdot}\sum_{j=1}^{4}\mathbf{x}_{\cdot,j}=\sum_{j=1}^{4}m_{i,j}(x_1+x_2+x_3+x_4)=1,\quad \forall i=1,\,2,\,3,\,4.$$

y, por tanto,

$$\sum_{j=1}^{4}m_{i,j}=\frac{1}{x_1+x_2+x_3+x_4},\quad \forall i=1,\,2,\,3,\,4.$$

□

Lema 4.2. *Como consecuencia de la proposición 4.1, se tiene que para el caso de 4 ciudades y dos cadenas ($n=4$, $K=2$) el punto de silla es:*

$$v^*_{x,i}=(2C+3)\frac{d_U}{(13C+15)d_U+3d_L+d_{1,3}+d_{1,4}+d_{2,3}+d_{2,4}},$$
$$\forall x\in\{1,\,2,\,3,\,4\},\quad \forall i\in\{1,\,2\}.$$

Demostración. Considerándose el caso $n=4$, $K=2$, se tiene que, teniendo en cuenta las matrices de la proposición 4.1

$$\mathbb{1}=(\mathbb{1})_4\equiv\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix}_{4\times4},\ \mathbf{I}=(\mathbf{I})_4\equiv\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}_{4\times4},$$

$$\mathbf{J_2}=(\mathbf{J_2})_4\equiv\begin{bmatrix}(\mathbb{1})_2-(\mathbf{I})_2&0\\0&(\mathbb{1})_2-(\mathbf{I})_2\end{bmatrix}_{4\times4}=\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}_{4\times4},$$

$$\mathbf{I_2}=(\mathbf{I_2})_4\equiv(\mathbf{J_2})_4+(\mathbf{I})_4=\begin{bmatrix}1&1&0&0\\1&1&0&0\\0&0&1&1\\0&0&1&1\end{bmatrix}_{4\times4}\quad\text{y}\quad\mathbf{D_2}=(\mathbf{D_2})_4\equiv\mathbf{D}\cdot\mathbf{J_2}=$$

$$\begin{bmatrix}0&0&d_{1,3}&d_{1,4}\\0&0&d_{2,3}&d_{2,4}\\d_{1,3}&d_{2,3}&0&0\\d_{1,4}&d_{2,4}&0&0\end{bmatrix}_{4\times4}\cdot\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}_{4\times4}=\begin{bmatrix}0&0&d_{1,4}&d_{1,3}\\0&0&d_{2,4}&d_{2,3}\\d_{2,3}&d_{1,3}&0&0\\d_{2,4}&d_{1,4}&0&0\end{bmatrix}_{4\times4}$$

la matriz $\mathbf{M}$ necesaria para el cálculo del punto de silla (ver ecuación 4.4 y observación 4.3) queda de la forma:

$$\mathbf{M} = \left[(B+2C)\mathbb{1} + A\mathbf{I_2} - B\mathbf{I} + D\left(\mathbf{D_2}+(\mathbf{D_2})^T\right)\right]^{-1} =$$

$$= \begin{bmatrix} 2C+A & A+B+2C & B+2C+D(d_{1,4}+d_{2,3}) & B+2C+D(d_{1,3}+d_{2,4}) \\ A+B+2C & 2C+A & B+2C+D(d_{1,3}+d_{2,4}) & B+2C+D(d_{1,4}+d_{2,3}) \\ B+2C+D(d_{1,4}+d_{2,3}) & B+2C+D(d_{1,3}+d_{2,4}) & 2C+A & A+B+2C \\ B+2C+D(d_{1,3}+d_{2,4}) & B+2C+D(d_{1,4}+d_{2,3}) & A+B+2C & 2C+A \end{bmatrix}^{-1}_{4\times 4}$$

Teniendo en cuenta el resultado de la proposición 4.1 y el lema 4.1, se obtiene:

$$v^*_{x,i} = (2C+3)\sum_{j=1}^{4} m_{x,j} = \frac{2C+3}{13C+15+3\frac{d_L}{d_U}+\frac{d_{1,3}+d_{1,4}+d_{2,3}+d_{2,4}}{d_U}}$$
$$= \frac{(2C+3)d_U}{(13C+15)d_U+3d_L+d_{1,3}+d_{1,4}+d_{2,3}+d_{2,4}}, \quad \forall x \in \{1,\,2,\,3,\,4\}, \quad \forall i \in \{1,\,2\}.$$

□

Observación 4.5. *A partir del punto de silla encontrado en el lema 4.2, se puede concluir que:*

$$\lim_{C\to\,\infty} v^*_{x,i} = \frac{2}{13}, \quad \forall x \in \{1,\,2,\,3,\,4\}, \quad \forall i \in \{1,\,2\}$$

y

$$\lim_{C\to\,0} v^*_{x,i} = \frac{3d_U}{15d_U+3d_L+d_{1,3}+d_{1,4}+d_{2,3}+d_{2,4}}, \quad \forall x \in \{1,\,2,\,3,\,4\}, \quad \forall i \in \{1,\,2\}.$$

Teorema 4.1. *El modelo de Hopfield con $n = 4$ ciudades y $K = 2$ cadenas se comporta como un 2-opt al considerar como punto inicial la proyección del punto de silla sobre una de las caras del Hipercubo de Hamming, es decir, fijando la posición de la primera ciudad de una de las dos cadenas.*

Demostración. Sea $\{1,\,2,\,3,\,4\}$ el conjunto de las $n = 4$ ciudades del enunciado y sean $[1,2]$, $[3,4]$ las $K = 2$ cadenas fijadas. A partir del punto de silla $\mathbf{v}^*$ calculado en el lema 4.2, puede fijarse, sin pérdida de generalidad, cualquiera de las 4 ciudades como primera ciudad visitada.

Se considera la ciudad 1 como primera ciudad visitada, lo cual permite obtener el siguiente punto inicial:

$$\mathbf{V}(0) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & v^*_{3,2} \\ 0 & v^*_{4,2} \end{bmatrix}$$

donde $v^*_{3,2} = v^*_{4,2} = \dfrac{(2C+3)d_U}{(13C+15)d_U+3d_L+d_{1,3}+d_{1,4}+d_{2,3}+d_{2,4}}$

Con esta proyección, se consiguen eliminar todas las simetrías del problema. Es decir, únicamente hay dos posibles soluciones: el tour $1-2-3-4$ o el tour $1-2-4-3$, cuyas soluciones son respectivamente:

$$\mathbf{V^1} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \mathbf{V^2} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

El modelo de Hopfield con $n = 4$ y $K = 2$ se comportará como un *2-opt* si el punto inicial $\mathbf{V}(0)$ converge[4] a la solución óptima, es decir, de entre las soluciones $\mathbf{V^1}$ y $\mathbf{V^2}$, converge a aquélla que tenga menor valor en la función objetivo.

Para saber a qué solución converge el punto $\mathbf{V}(0)$, simplemente se calculará $\mathbf{V}(\Delta t)$, $\Delta t > 0$ (siguiendo la ecuación diferencial 1.2 con $\lambda \to \infty$) y se comprobará si el punto está más cerca[5] de la solución $\mathbf{V_1}$ ó $\mathbf{V_2}$.

$$\frac{d\mathbf{u}(0)}{dt} = \mathbf{T} \cdot \mathbf{v}(0) + \mathbf{i^b} \tag{4.5}$$

donde el vector $\mathbf{v}(0)$ es la matriz $\mathbf{V}(0)$ dispuesta en formato vectorial. Con el objetivo de simplificar los cálculos, se escribirá la matriz $\mathbf{V}$ como:

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_{\cdot,1} & \mathbf{v}_{\cdot,2} \end{bmatrix}$$

quedando dispuesta en formato vectorial de la forma:

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_{\cdot,1} \\ \mathbf{v}_{\cdot,2} \end{bmatrix}$$

Según lo visto en la demostración de la proposición 4.1, la matriz de pesos $\mathbf{T}$ y el vector de entradas externas $\mathbf{i^b}$ son de la forma:

$$\mathbf{T} = -\begin{bmatrix} (B+C)\mathbb{1}-B\mathbf{I} & C\mathbb{1}+A\mathbf{I_2}+D(\mathbf{D_2}+(\mathbf{D_2})^T) \\ C\mathbb{1}+A\mathbf{I_2}+D(\mathbf{D_2}+(\mathbf{D_2})^T) & (B+C)\mathbb{1}-B\mathbf{I} \end{bmatrix}, \quad \mathbf{i}^b = CN'\begin{bmatrix} (\mathbf{1})_{4\times 1} \\ (\mathbf{1})_{4\times 1} \end{bmatrix} = CN'\begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix}$$

Desglosando la ecuación 4.5 se tiene:

$$\begin{aligned} \frac{d\mathbf{u}(0)}{dt} &= \mathbf{T} \cdot \mathbf{v}(0) + \mathbf{i^b} \\ &= -\begin{bmatrix} (B+C)\mathbb{1}-B\mathbf{I} & C\mathbb{1}+A\mathbf{I_2}+D(\mathbf{D_2}+(\mathbf{D_2})^T) \\ C\mathbb{1}+A\mathbf{I_2}+D(\mathbf{D_2}+(\mathbf{D_2})^T) & (B+C)\mathbb{1}-B\mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v}_{\cdot,1}(0) \\ \mathbf{v}_{\cdot,2}(0) \end{bmatrix} + CN'\begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix} \end{aligned}$$

[4] Nótese que $\mathbf{V}(0)$ no es un punto de equilibrio del sistema dinámico, dado que la proyección del punto de silla es un punto diferente que el punto de silla de la función de energía proyectada.

[5] En caso de seguir a la misma distancia, es decir, $v_{3,2}(\Delta t) = v_{4,2}(\Delta t)$, se vuelve a proyectar $\mathbf{V}(\Delta t)$ sobre la misma cara, y se repite el procedimiento.

$$= - \begin{bmatrix} ((B+C)\mathbb{1} - B\mathbf{I})\,\mathbf{v}_{\cdot,1}(0) + \left(C\mathbb{1} + A\mathbf{I_2} + D(\mathbf{D_2} + (\mathbf{D_2})^T)\right)\mathbf{v}_{\cdot,2}(0) + CN'\mathbf{1} \\ \left(C\mathbb{1} + A\mathbf{I_2} + D(\mathbf{D_2} + (\mathbf{D_2})^T)\right)\mathbf{v}_{\cdot,1}(0) + ((B+C)\mathbb{1} - B\mathbf{I})\,\mathbf{v}_{\cdot,2}(0) + CN'\mathbf{1} \end{bmatrix}$$

El valor del potencial en $t = 0$ puede obtenerse aplicando la función inversa de la función de activación al vector $\mathbf{v}(0)$:

$$\mathbf{u}(0) = g^{-1}(\mathbf{v}(0))$$

Utilizando la función de activación definida en la ecuación 2.11, se tiene que su función inversa es:

$$u_i = g^{-1}(v_i) = \begin{cases} -u_0 & \text{si} \quad v_i \leq 0 \\ u_0(2v_i - 1) & \text{si} \quad 0 < v_i < 1 \\ u_0 & \text{si} \quad v_i \geq 1 \end{cases} \qquad \forall i \in \{1, \ldots, n\}$$

de modo que:

$$\mathbf{u}(0) = g^{-1}(\mathbf{v}(0)) = g^{-1}\left(\begin{bmatrix} \mathbf{v}_{\cdot,1}(0) \\ \mathbf{v}_{\cdot,2}(0) \end{bmatrix}\right) = \begin{bmatrix} \mathbf{u}_{\cdot,1}(0) \\ \mathbf{u}_{\cdot,2}(0) \end{bmatrix}$$

donde

$$\mathbf{u}_{\cdot,1}(0) = \begin{bmatrix} u_0 \\ -u_0 \\ -u_0 \\ -u_0 \end{bmatrix} \text{ y } \mathbf{u}_{\cdot,2}(0) = \begin{bmatrix} -u_0 \\ -u_0 \\ g^{-1}(v^*_{3,2}) \\ g^{-1}(v^*_{4,2}) \end{bmatrix}$$

con $g^{-1}(v^*_{3,2}) = g^{-1}(v^*_{4,2}) = u_0\left(\dfrac{2(2C+3)d_U}{(13C+15)d_U + 3d_L + d_{1,3} + d_{1,4} + d_{2,3} + d_{2,4}} - 1\right)$

A continuación, se calcula el potencial en $t = \Delta t$:

$$\mathbf{u}(\Delta t) = \mathbf{u}(0) + \frac{d\mathbf{u}(0)}{dt} \cdot \Delta t$$

siendo Δt el paso de integración.

Finalmente, se calcula la salida del sistema dinámico en $t = 1$ (utilizando la función de activación definida en la ecuación 2.11):

$$\mathbf{v}(\Delta t) = g(\mathbf{u}(\Delta t)) = g\left(\mathbf{u}(0) + \frac{d\mathbf{u}(0)}{dt} \cdot \Delta t\right)$$
$$= \begin{bmatrix} g\Big(\mathbf{u}_{\cdot,1}(0) - \big(\,((B+C)\mathbb{1} - B\mathbf{I})\,\mathbf{v}_{\cdot,1}(0) + \left(C\mathbb{1} + A\mathbf{I_2} + D(\mathbf{D_2} + (\mathbf{D_2})^T)\right)\mathbf{v}_{\cdot,2}(0) + CN'\mathbf{1}\big)\Delta t\Big) \\ g\Big(\mathbf{u}_{\cdot,2}(0) - \big(\,\left(C\mathbb{1} + A\mathbf{I_2} + D(\mathbf{D_2} + (\mathbf{D_2})^T)\right)\mathbf{v}_{\cdot,1}(0) + ((B+C)\mathbb{1} - B\mathbf{I})\,\mathbf{v}_{\cdot,2}(0) + CN'\mathbf{1}\big)\Delta t\Big) \end{bmatrix}$$

Se calculan cada uno de los dos vectores (por cajas) obtenidos para $\mathbf{v}(\Delta t)$:

$$
\begin{aligned}
\mathbf{v}_{\cdot,1}(\Delta t) &= g\left(\mathbf{u}_{\cdot,1}(0) - \left(((B+C)\mathbb{1}-B\mathbf{I})\,\mathbf{v}_{\cdot,1}(0) + \left(C\mathbb{1}+A\mathbf{I_2}+D(\mathbf{D_2}+(\mathbf{D_2})^T)\right)\mathbf{v}_{\cdot,2}(0) + CN'\mathbf{1}\right)\Delta t\right)\\
&= \frac{1}{2}\left(\mathbf{1} + \frac{1}{u_0}\left(\mathbf{u}_{\cdot,1}(0) - \left(((B+C)\mathbb{1}-B\mathbf{I})\,\mathbf{v}_{\cdot,1}(0) + \left(C\mathbb{1}+A\mathbf{I_2}+D(\mathbf{D_2}+(\mathbf{D_2})^T)\right)\mathbf{v}_{\cdot,2}(0) + CN'\mathbf{1}\right)\Delta t\right)\right)\\
&= \frac{1}{2}\Bigg(\begin{bmatrix}1\\1\\1\\1\end{bmatrix} + \frac{1}{u_0}\Bigg(\begin{bmatrix}u_0\\-u_0\\-u_0\\-u_0\end{bmatrix} - \Bigg(\begin{bmatrix}C & B+C & B+C & B+C\\ B+C & C & B+C & B+C\\ B+C & B+C & C & B+C\\ B+C & B+C & B+C & C\end{bmatrix}\cdot\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\\
&\quad + \begin{bmatrix}A+C & A+C & C+D(d_{1,4}+d_{2,3}) & C+D(d_{1,3}+d_{2,4})\\ A+C & A+C & C+D(d_{1,3}+d_{2,4}) & C+D(d_{1,4}+d_{2,3})\\ C+D(d_{1,4}+d_{2,3}) & C+D(d_{1,3}+d_{2,4}) & A+C & A+C\\ C+D(d_{1,3}+d_{2,4}) & C+D(d_{1,4}+d_{2,3}) & A+C & A+C\end{bmatrix}\cdot\begin{bmatrix}0\\0\\v^*_{3,2}\\v^*_{3,2}\end{bmatrix}\\
&\quad + \begin{bmatrix}CN'\\CN'\\CN'\\CN'\end{bmatrix}\Bigg)\Delta t\Bigg)\Bigg) = -\frac{1}{2u_0}\begin{bmatrix}\Big((2v^*_{3,2}+N'+1)C + v^*_{3,2}(d_{1,3}+d_{1,4}+d_{2,3}+d_{2,4})D\Big)\Delta t - 2u_0\\ \Big(B + (2v^*_{3,2}+N'+1)C + v^*_{3,2}(d_{1,3}+d_{1,4}+d_{2,3}+d_{2,4})D\Big)\Delta t\\ \Big(2v^*_{3,2}A + B + (2v^*_{3,2}+N'+1)C\Big)\Delta t\\ \Big(2v^*_{3,2}A + B + (2v^*_{3,2}+N'+1)C\Big)\Delta t\end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{v}_{\cdot,2}(\Delta t) &= g\Big(\mathbf{u}_{\cdot,2}(0) - \left(\left(C\mathbb{1}+A\mathbf{I_2}+D(\mathbf{D_2}+(\mathbf{D_2})^T)\right)\mathbf{v}_{\cdot,1}(0) + ((B+C)\mathbb{1}-B\mathbf{I})\,\mathbf{v}_{\cdot,2}(0) + CN'\mathbf{1}\right)\Delta t\Big)\\
&= \frac{1}{2}\left(\mathbf{1} + \frac{1}{u_0}\left(\mathbf{u}_{\cdot,2}(0) - \left(\left(C\mathbb{1}+A\mathbf{I_2}+D(\mathbf{D_2}+(\mathbf{D_2})^T)\right)\mathbf{v}_{\cdot,1}(0) + ((B+C)\mathbb{1}-B\mathbf{I})\,\mathbf{v}_{\cdot,2}(0) + CN'\mathbf{1}\right)\Delta t\right)\right)\\
&= \frac{1}{2}\Bigg(\begin{bmatrix}1\\1\\1\\1\end{bmatrix} + \frac{1}{u_0}\Bigg(\begin{bmatrix}-u_0\\-u_0\\g^{-1}(v^*_{3,2})\\g^{-1}(v^*_{3,2})\end{bmatrix}\\
&\quad - \Bigg(\begin{bmatrix}A+C & A+C & C+D(d_{1,4}+d_{2,3}) & C+D(d_{1,3}+d_{2,4})\\ A+C & A+C & C+D(d_{1,3}+d_{2,4}) & C+D(d_{1,4}+d_{2,3})\\ C+D(d_{1,4}+d_{2,3}) & C+D(d_{1,3}+d_{2,4}) & A+C & A+C\\ C+D(d_{1,3}+d_{2,4}) & C+D(d_{1,4}+d_{2,3}) & A+C & A+C\end{bmatrix}\cdot\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\\
&\quad + \begin{bmatrix}C & B+C & B+C & B+C\\ B+C & C & B+C & B+C\\ B+C & B+C & C & B+C\\ B+C & B+C & B+C & C\end{bmatrix}\cdot\begin{bmatrix}0\\0\\v^*_{3,2}\\v^*_{3,2}\end{bmatrix} + \begin{bmatrix}CN'\\CN'\\CN'\\CN'\end{bmatrix}\Bigg)\Delta t\Bigg)\Bigg)\\
&= -\frac{1}{2u_0}\begin{bmatrix}(A + 2v^*_{3,2}B + (2v^*_{3,2}+N'+1)C)\Delta t\\ (A + 2v^*_{3,2}B + (2v^*_{3,2}+N'+1)C)\Delta t\\ (v^*_{3,2}B + (2v^*_{3,2}+N'+1)C + (d_{1,4}+d_{2,3})D)\Delta t - u_0 - g^{-1}(v^*_{3,2})\\ (v^*_{3,2}B + (2v^*_{3,2}+N'+1)C + (d_{1,3}+d_{2,4})D)\Delta t - u_0 - g^{-1}(v^*_{3,2})\end{bmatrix}
\end{aligned}
$$

El punto de silla deja a todas las soluciones del problema a la misma distancia (ver observación 4.5). Habiendo proyectado el punto de silla $\mathbf{v}^*$ sobre una de sus caras para obtener $\mathbf{v}(0)$, este punto ya no será un punto de equilibrio. Sin embargo, la distancia de $\mathbf{v}(0)$ a las dos soluciones del problema seguirá siendo la misma. Para saber a qué solución converge el sistema dinámico partiendo de $\mathbf{v}(0)$, bastará con saber a qué solución está más próximo $\mathbf{v}(\Delta t)$. A ella convergerá el sistema dinámico.

Supongamos que $\mathbf{v}(\Delta t)$ se encuentra más próximo a la primera solución. El caso en que $\mathbf{v}(2)$ está más próximo a la segunda solución es equivalente. Considerando las soluciones escritas en formato vectorial y suponiendo distancia euclídea, se tiene que:

$$dist(\mathbf{v}(\Delta t), \mathbf{v^1}) < dist(\mathbf{v}(\Delta t), \mathbf{v^2})$$

Considerando los cuadrados de las distancias para simplificar el análisis:

$$dist(\mathbf{v}(\Delta t), \mathbf{v^1})^2 < dist(\mathbf{v}(\Delta t), \mathbf{v^2})^2$$

y teniendo en cuenta que se cancelan todos los términos salvo los dos últimos:

$$\begin{gathered}\Big(-\tfrac{1}{2u_0}\big((v^*_{3,2}B + (2v^*_{3,2} + N' + 1)C + (d_{1,4} + d_{2,3})D)\Delta t - u_0 - g^{-1}(v^*_{3,2})\big) - 1\Big)^2 \\ +\Big(-\tfrac{1}{2u_0}\big((v^*_{3,2}B + (2v^*_{3,2} + N' + 1)C + (d_{1,3} + d_{2,4})D)\Delta t - u_0 - g^{-1}(v^*_{3,2})\big)\Big)^2 \\ < \\ \Big(-\tfrac{1}{2u_0}\big((v^*_{3,2}B + (2v^*_{3,2} + N' + 1)C + (d_{1,4} + d_{2,3})D)\Delta t - u_0 - g^{-1}(v^*_{3,2})\big)\Big)^2 \\ +\Big(-\tfrac{1}{2u_0}\big((v^*_{3,2}B + (2v^*_{3,2} + N' + 1)C + (d_{1,3} + d_{2,4})D)\Delta t - u_0 - g^{-1}(v^*_{3,2}) - 1\big)\Big)^2\end{gathered}$$

La desigualdad $(a-1)^2 + b^2 < a^2 + (b-1)^2$ se verifica si y solo si $b < a$. Este resultado nos lleva a concluir que la desigualdad anterior se satisface si y solo si:

$$d_{1,3} + d_{2,4} < d_{1,4} + d_{2,3}$$

que coincide con la desigualdad que se verifica en un *intercambio 2-opt* cuando el tour que produce $\mathbf{v^1}$ es inferior al que produce $\mathbf{v^2}$ (ver observación 4.4 y figura 4.3, tomando $a = 1$, $b = 2$, $c = 3$, $d = 4$). Por tanto, utilizando como punto inicial en el modelo de Hopfield la proyección del punto de silla sobre una de las caras (para fijar la primera ciudad visitada), la simulación del modelo de Hopfield se comporta exactamente igual que un *intercambio 2-opt.* □

Observación 4.6. *Nótese que la demostración del teorema 4.1 muestra como el modelo de Hopfield con $n = 4$ y $K = 2$ se comportará como un intercambio 2-opt con independencia de la parametrización utilizada, siempre y cuando sea una parametrización válida que verifique las desigualdades 3.25.*

Observación 4.7. *El resultado del teorema 4.1 indica además que el punto elegido converge siempre a la solución óptima con independencia del valor que tome C. Sin embargo, como se muestra en la siguiente sección, la cuenca de atracción para el óptimo global crece a medida que el parámetro libre decrece, tal y como ocurría en el ejemplo del GQKP de la sección 2.3.1.*

4.4 Cuencas de atracción del modelo de Hopfield como algoritmo 2-opt utilizando el simulador de la CHN

Una vez demostrado que el modelo de Hopfield con $n = 4$ ciudades y $K = 2$ cadenas se comporta exactamente igual que un *intercambio 2-opt*, se completa el estudio analizando cómo son las cuencas de atracción de la CHN en este caso.

Se considera el siguiente TSP con $n = 4$ ciudades emplazadas en las siguientes coordenadas:

$$a = (1, 1),\ b = (1, 8),\ c = (1, 2),\ d = (2, 8) \tag{4.6}$$

de modo que la matriz de distancias es:

$$\mathbf{D} = \begin{bmatrix} 0 & 7 & 1 & 5\sqrt{2} \\ 7 & 0 & 5\sqrt{2} & 1 \\ 1 & 5\sqrt{2} & 0 & 7 \\ 5\sqrt{2} & 1 & 7 & 0 \end{bmatrix}$$

Se fijan además, las $K = 2$ cadenas: $[a, b]$; $[c, d]$,

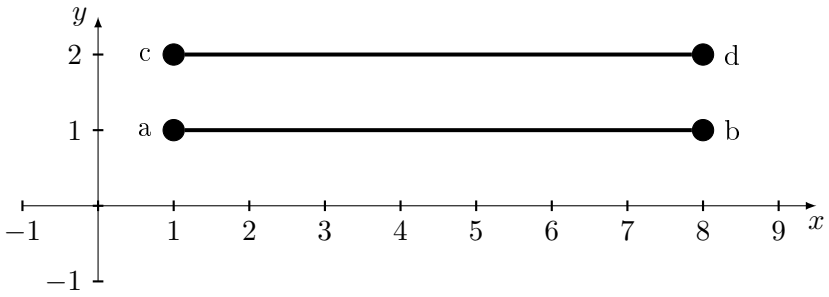


Figura 4.4: TSP con dos cadenas fijadas $a - b$ y $c - d$

quedando la matriz de distancias queda de la forma:

$$\mathbf{D}^{\not{k}} = \begin{bmatrix} 0 & 0 & 1 & 5\sqrt{2} \\ 0 & 0 & 5\sqrt{2} & 1 \\ 1 & 5\sqrt{2} & 0 & 0 \\ 5\sqrt{2} & 1 & 0 & 0 \end{bmatrix}$$

Se construye un modelo de Hopfield asociado ($N = 4$, $K = 2$) y se resuelve con ayuda del simulador de la CHN desarrollado en MATLAB, utilizando diferentes puntos de arranque. Los puntos iniciales han sido seleccionados muestreando la cara en la cual se proyecta el punto de silla con un array 400×400. Es decir, habiendo fijado la primera ciudad visitada (en este caso se elije la ciudad a), se dan valores a $v_{3,2}$ y $v_{4,2}$ de modo que:

$$\mathbf{V_0} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & v_{3,2} \\ 0 & v_{4,2} \end{bmatrix},\ \text{con } v_{3,2}, v_{4,2} \in (0, 1)$$

Se aprecia en la figura 4.4 que la solución óptima se alcanza cuando $v_{3,2} = 0$ y $v_{4,2} = 1$. Al igual que se mostró en la secciones 2.3 y 2.3.1 para el ejemplo sencillo del GQKP, se visualizan en la figura 4.5 la función de energía y cuencas de atracción para el TSP para distintos valores del parámetro libre.

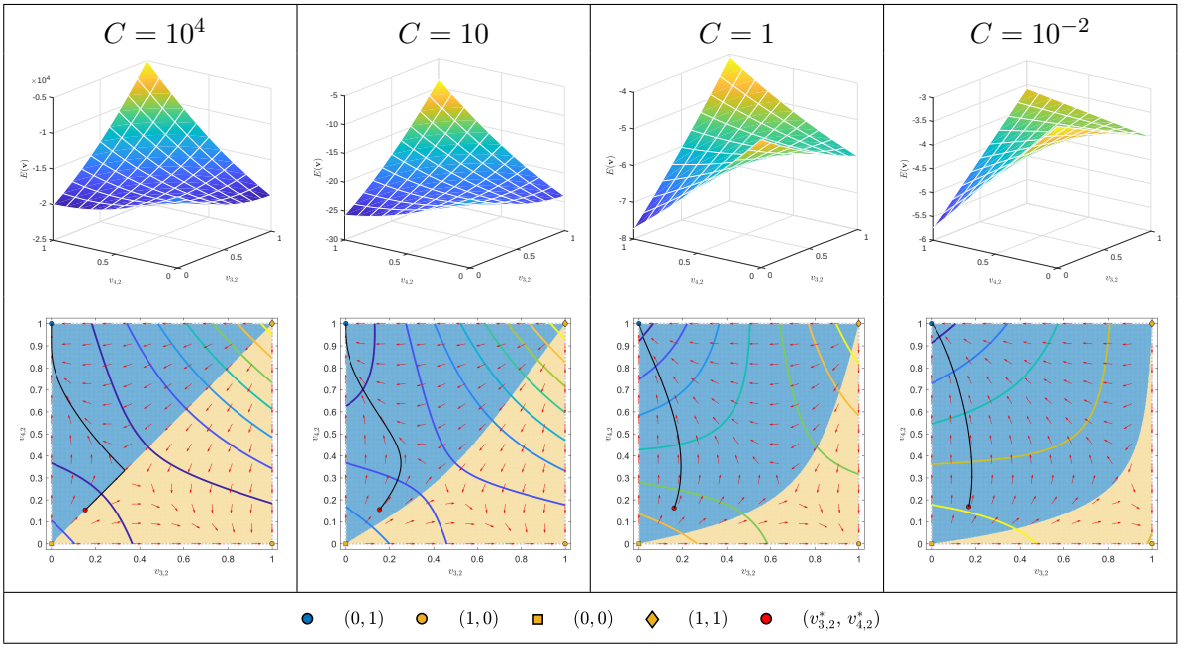


Figura 4.5: Proyección de la función de energía de la CHN para el ejemplo definido por las ciudades de la ecuación 4.6 (fijadas las cadenas $a - b$ y $c - d$ y la ciudad a como primera ciudad), junto con sus cuencas de atracción para distintos valores del parámetro libre C

Nótese que, en esta ocasión, no se trata de la función de energía real del problema, sino de una proyección sobre una de sus caras. De nuevo, se colorean los puntos iniciales de acuerdo con el color de la solución a la cual convergen. Se aprecia que, como ya se estudió en el caso del GQKP, la cuenca de atracción para la mejor solución crece a medida que el parámetro libre decrece, pero con una importante diferencia con respecto al ejemplo del GQKP: eligiendo como punto inicial la proyección del punto de silla sobre una de las caras, se garantiza el poder obtener siempre la solución óptima, lo que lleva a la CHN con $N = 4$ y $K = 2$ a comportarse como un *intercambio 2-opt*.

En el ejemplo anterior, no parece existir una alta competencia entre las neuronas $v_{3,2}$ y $v_{4,2}$, en parte debido al suficiente margen entre las sumas de distancias de las dos posible soluciones $2 = d_{a,c} + d_{b,d} < d_{a,d} + d_{b,c} = 10\sqrt{2}$. Por ello, se considera a continuación el caso más desfavorable: se fijan las cadenas $[a, c]$; $[b, d]$ (ver figura 4.6).

En este caso, el orden utilizado en las filas de la matriz $\mathbf{V}$, así como en la matriz $\mathbf{D}^{\not k}$ es $\{a, b, c, d\}$. De este modo, la matriz de distancias del problema es:

$$\mathbf{D}^{\not k} = \begin{bmatrix} 0 & 0 & 7 & 5\sqrt{2} \\ 0 & 0 & 5\sqrt{2} & 7 \\ 7 & 5\sqrt{2} & 0 & 0 \\ 5\sqrt{2} & 7 & 0 & 0 \end{bmatrix}$$

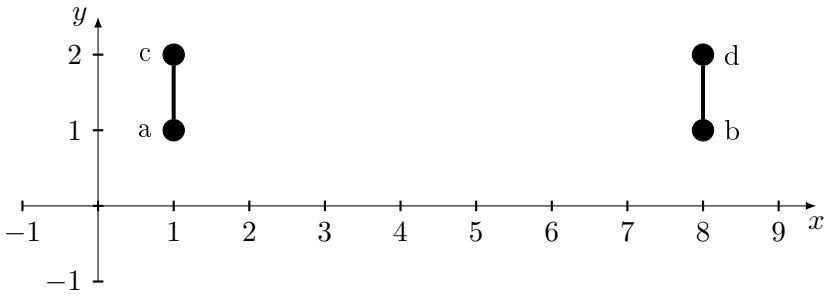


Figura 4.6: TSP con dos cadenas fijadas $a-c$ y $b-d$

Llevando a cabo de nuevo simulaciones de la CHN con diferentes puntos iniciales, se observa en la figura 4.7 cómo la ahora mayor competencia entre las neuronas $v_{3,2}$ y $v_{4,2}$ (debido a la mayor similitud entre las distancias del problema), hace que al disminuir el valor del parámetro libre C, el aumento de la cuenca de atracción sea prácticamente imperceptible, siendo necesario aproximarse al punto de silla para ver cómo aumenta la cuenca de atracción. Se aprecia además, cómo la trayectoria del punto que arranca en la proyección del punto de silla, se ve modificada con la disminución del parámetro libre.

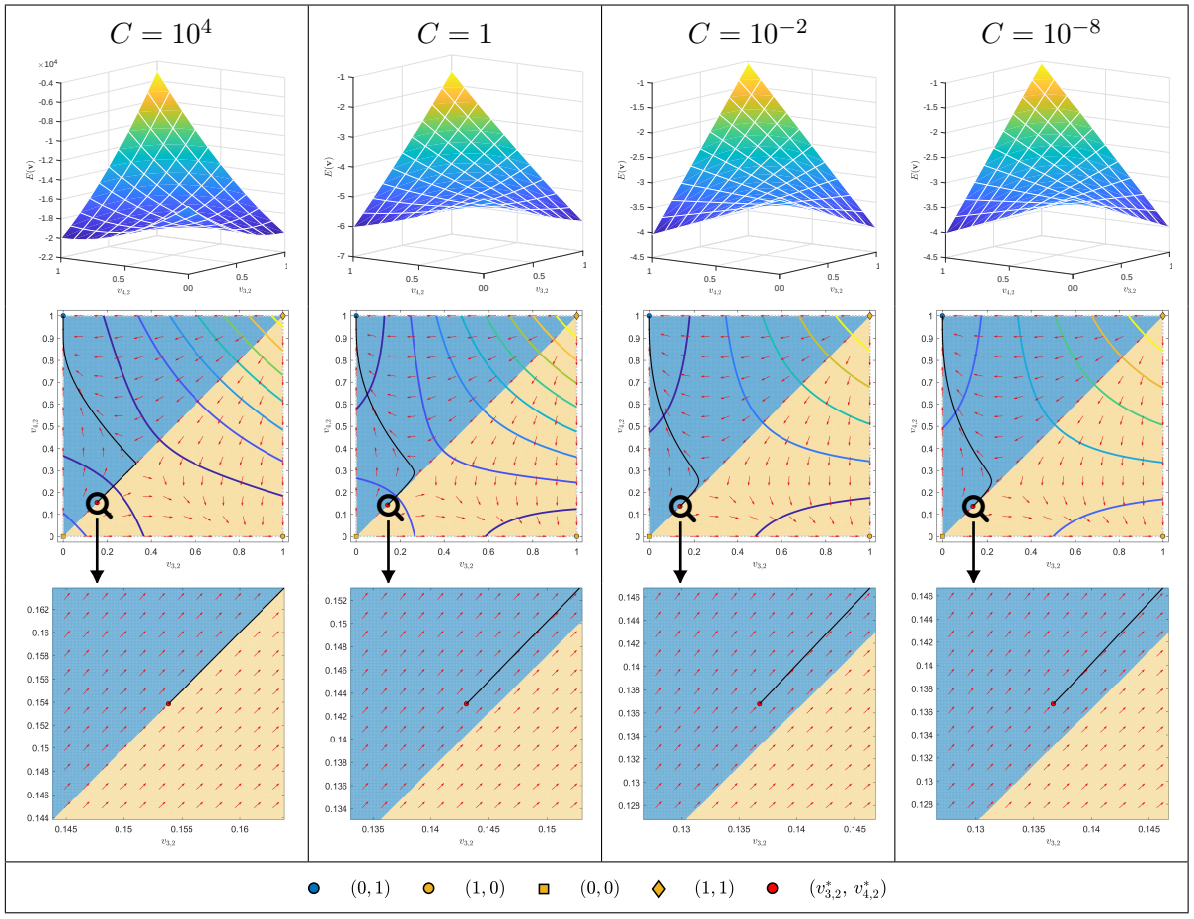


Figura 4.7: Proyección de la función de energía de la CHN para el ejemplo definido por las ciudades de la ecuación 4.6 (fijadas las cadenas $a-c$ y $b-d$ y la ciudad a como primera ciudad), junto con sus cuencas de atracción para distintos valores del parámetro libre C

CAPÍTULO 5

Implementación y experiencias computacionales

Este capítulo recoge todos los detalles de implementación y experiencias computacionales necesarios para apoyar los resultados teóricos presentados en esta memoria. Apoyándose en métodos de simulación ya existentes, Euler, Runge-Kutta y Talaván Yáñez, se detalla el algoritmo necesario para implementar la estrategia Divide-y-Vencerás así como la manipulación de matrices necesaria para poder resolver instancias grandes. Se recogen y comparan los resultados de los diferentes modelos considerados para resolver el TSP: la CHN Tradicional, Divide-y-Vencerás y el 2-opt, utilizando en todos los casos el modelo de Hopfield para resolverlos.

Contenidos del capítulo

5.1 Estructura de la matriz de pesos

Una de las grandes críticas que ha recibido el modelo de Hopfield desde su concepción es su incapacidad para resolver problemas grandes. Esto se debe a fundamentalmente a cómo crece la matriz de pesos a medida que crece el tamaño del problema. Nótese que en el caso del TSP, por ejemplo, la matriz de pesos tiene tamaño $N^2 \times N^2$, donde N es el número de ciudades. En palabras de Serpen [44], "el tamaño de la matriz de pesos hace que sea prácticamente imposible resolver instancias grandes del TSP, incluso haciendo uso de plataformas de supercomputación", como aparentemente fue su caso utilizando el *Ohio Supercomputing Center* para resolver instancias de hasta 400 ciudades (cuya matriz de pesos ocupa unos 200 GB en doble precisión). Un año antes, Talaván y Yáñez [49] resolvieron instancias de hasta $1,002$ ciudades utilizando la CHN. Esta matriz de pesos, de almacenarse completa en memoria (en doble precisión), ocuparía un total de aproximadamente 8 TB. Más recientemente, García et al. [10], han resuelto instancias de hasta $13,509$ ciudades (resultados que se presentan en la sección 5.4). El modelo de Hopfield asociado tiene más de 180 millones de neuronas ($13,509^2$) y su matriz de pesos, la cual es densa, ocuparía más de 260 PB (en doble precisión) en caso de almacenarla.

Por todo lo anterior, es necesario una computación muy eficiente conocida la estructura (ver ecuación 2.27) de la matriz de pesos, que permita no tener que almacenar la matriz por completo en memoria. Para ello, se calcula directamente el producto $\mathbf{T} \cdot \mathbf{v}$, sin necesidad de calcular $\mathbf{T}$ como matriz intermedia, tal y como se muestra en el siguiente pseudo-código:

Algoritmo 2 Cálculo eficiente de $\mathbf{T} \cdot \mathbf{v}$ para el modelo de Hopfield tradicional

procedure TTIMESV(N, $\mathbf{V}$, $\mathbf{D}$, *params*)
  $S_x = \sum_{i=1}^{N} v_{x,i} = 1$, $S_i = \sum_{x=1}^{N} v_{x,i} = 1$, $S = \sum_{x}^{N} S_x = \sum_{i}^{N} S_i$
  return: $-A(S_x - v_{x,i}) - B(S_i - v_{x,i}) - CS - D\sum_y (d_{x,y}(v_{y,(i-1) \bmod N} + v_{y,(i+1) \bmod N}))$
end procedure

5.2 Métodos de simulación

Recordando lo visto en la sección 1.3.2, el sistema dinámico asociado a la CHN se describe a partir de la ecuación diferencial (ver ecuación 1.2, con $\lambda \to \infty$):

$$\frac{du_i(t)}{dt} = \sum_{j=1}^{n} T_{i,j} v_j(t) + i_i^b, \quad \forall i \in \{1, \ldots, n\}$$
$$u_i(0) = u_i^0, \quad \forall i \in \{1, \ldots, n\}$$
$$v_i(t) = g(u_i(t)) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i(t)}{u_0}\right)\right), \quad u_0 > 0$$

Obtener una solución para cualquier problema de optimización combinatoria como es el caso del TSP, implica encontrar el punto estable para este *problema de valor inicial* o *de Cauchy*, con $u_i(0) = u_i^0 \quad \forall i \in \{1, \ldots, n\}$.

Con el objetivo de resolver el sistema dinámico asociado a la CHN (y al no ser posible resolver la ecuación diferencial de forma analítica), se presentan a continuación los tres

métodos implementados, si bien ha sido el último de ellos (el método de Talaván y Yáñez [50]), el utilizado en la realización de todas las experiencias computacionales, debido a su más rápida convergencia.

5.2.1 El método de Euler

El *método de Euler* es el más conocido y sencillo de los procedimientos de integración numérica para la resolución de ecuaciones diferenciales ordinarias. Dado el problema de valor inicial:

$$\begin{cases} \mathbf{y}'(t) = f(\mathbf{y}(t), t) \\ \mathbf{y}(0) = \mathbf{y}^0 \end{cases} \tag{5.1}$$

donde $\mathbf{y}$ se corresponde con el vector solución de n componentes e $\mathbf{y_0}$ es un vector de condiciones iniciales, se calcula el valor de la solución en $t + \Delta t$ a partir del valor de la solución en t y utilizando la información de la derivada en tiempo t:

$$\mathbf{y}(t + \Delta t) = \mathbf{y}(t) + f(\mathbf{y}, t) \cdot \Delta t$$

Así, utilizando este método de integración numérica para la resolución del sistema dinámico asociado a la CHN, se obtiene:

$$\begin{aligned} \mathbf{u}(t + \Delta t) &= \mathbf{u}(t) + (\mathbf{T}\mathbf{v} + \mathbf{i^b})\Delta t \\ \mathbf{v}(t + \Delta t) &= \frac{1}{2}\left(1 + \tanh\left(\frac{\mathbf{u}(t + \Delta t)}{u_0}\right)\right), \quad u_0 > 0 \end{aligned}$$

donde $\mathbf{T}$ y $\mathbf{i^b}$ son respectivamente la matriz de pesos y vector de umbrales de la CHN.

El *método de Euler* tiene un error de truncamiento de $\mathrm{O}(\Delta t^2)$, para comprobarlo basta con estudiar la expansión de Taylor de $\mathbf{y}(t)$, por lo que será necesario tomar en ocasiones un paso de integración excesivamente pequeño, lo cuál tendrá un impacto directo en la velocidad de convergencia del sistema dinámico. Otras limitaciones incluyen el error por redondeo acumulado cometido por el ordenador así como la sensibilidad a cambios de escala (un escalado de la matriz de pesos y vector de umbrales no varía los mínimos del problema pero las cuencas son más pronunciadas, de modo que el proceso tiene mayor velocidad [50]). Pese a todo, el *método de Euler* es uno de los métodos de integración más utilizados para la resolución de la CHN.

5.2.2 El método de Runge-Kutta

Los *métodos de Runge-Kutta* sustituyen el problema de valor inicial (ecuación 5.1) por la ecuación integral equivalente:

$$\int_n^{n+1} d\mathbf{y} = \int_{t_n}^{t_{n+1}} f(\mathbf{y}, t)dt \implies \mathbf{y_{n+1}} = \mathbf{y_n} + \int_{t_n}^{t_{n+1}} f(\mathbf{y}(t), t)dt$$

En el desarrollo de esta tesis doctoral se ha implementado también el *método de Runge-Kutta de orden 4*, donde, basándose en el método de integración de Simpson, se obtiene:

$$\begin{aligned}
\mathbf{k_1}(t) &= \Delta t \cdot f(\mathbf{y}(t), t) \\
\mathbf{k_2}(t) &= \Delta t \cdot f\left(\mathbf{y}(t) + \frac{\mathbf{k_1}(t)}{2}, t + \frac{\Delta t}{2}\right) \\
\mathbf{k_3}(t) &= \Delta t \cdot f\left(\mathbf{y}(t) + \frac{\mathbf{k_2}(t)}{2}, t + \frac{\Delta t}{2}\right) \\
\mathbf{k_4}(t) &= \Delta t \cdot f\left(\mathbf{y}(t) + \mathbf{k_3}(t), t + \Delta t\right) \\
\mathbf{y}(t + \Delta t) &= \mathbf{y}(t) + \frac{1}{6}(\mathbf{k_1}(t) + 2\mathbf{k_2}(t) + 2\mathbf{k_3}(t) + \mathbf{k_4}(t))
\end{aligned}$$

Pese a que el *método de Runge-Kutta de orden 4* mejora notablemente el error de truncamiento (en este caso es $O(\Delta t^5)$) con respecto al *método de Euler*, su utilización para resolver la CHN será poco adecuado debido a su lenta convergencia, aunque será muy útil en problemas pequeños cuando se deseen conocer las trayectorias de manera precisa (como en el caso de la resolución del GQKP utilizando la CHN en la sección 2.3.1).

5.2.3 El método de Talaván-Yáñez

El *método de Euler*, el *método de Runge-Kutta de orden 4*, así como muchos otros métodos de integración numérica no recogidos en este capítulo, resultan excesivamente lentos a la hora de resolver problemas de optimización combinatoria mediante el modelo de Hopfield, especialmente cuando se desea resolver problemas de tamaño medio o grande.

Esta problemática es identificada por Talaván y Yáñez [50], quienes proponen un método en el que el paso de integración será variable y determinado por la función de energía en cada paso. El objetivo es poder garantizar un rápido descenso de ésta y así acelerar la convergencia al punto de equilibrio.

De modo resumido, el objetivo es encontrar en cada paso de integración el valor adecuado de Δt. En este caso, se considera, a diferencia del *método de Euler* y *Runge-Kutta de orden 4*, al vector de estados $\mathbf{v}(t)$ como variable del sistema dinámico:

$$\begin{aligned}
\frac{dv_i(t)}{dt} &= \frac{2}{u_0} v_i(t) \cdot (1 - v_i(t)) \cdot \frac{du_i(t)}{dt}, \quad \forall i \in \{1, \ldots, n\} \\
\mathbf{v}(0) &= \mathbf{v^0}
\end{aligned}$$

con

$$\frac{d\mathbf{u}(t)}{dt} = \mathbf{T} \cdot \mathbf{v}(t) + \mathbf{i^b}$$

calculándose el valor del vector de estados en el siguiente paso como:

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \frac{d\mathbf{v}(t)}{dt} \Delta t$$

Así, se calcula el valor de la función de energía en $t + \Delta t$ como:

$$E(t + \Delta t) = E(t) - S_1 \Delta t + \frac{1}{2} S_2 \Delta t^2$$

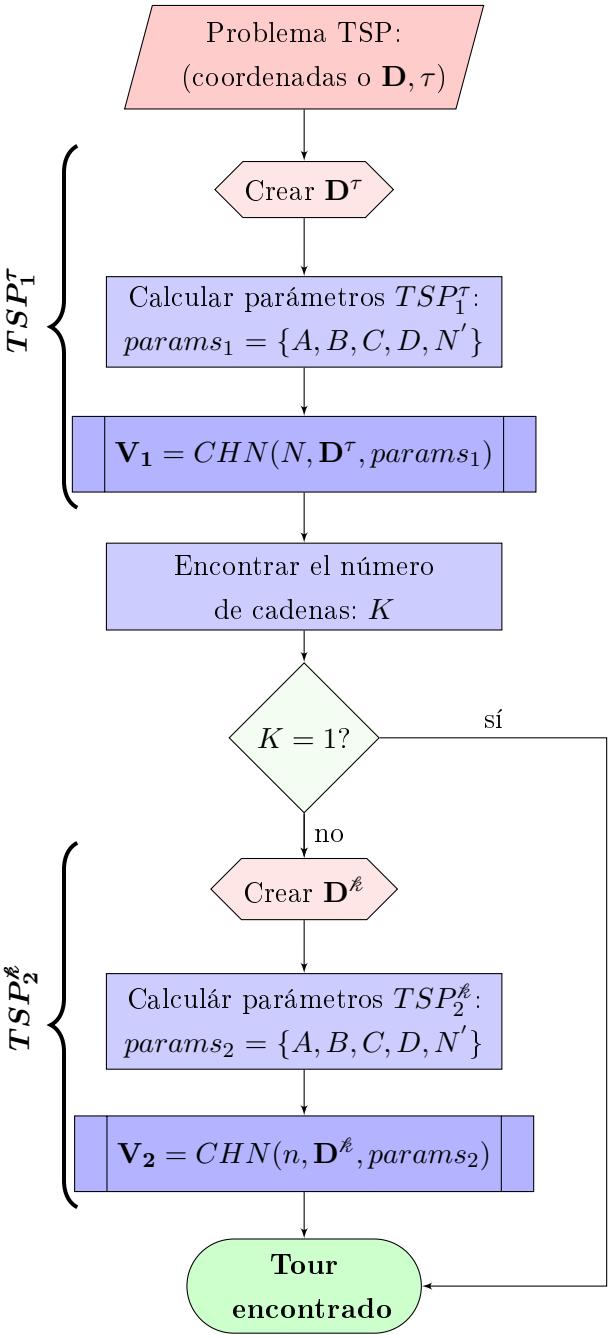
siendo una función parabólica con respecto a Δt, con

$$S_1 \equiv \sum_{i=1}^{n} \frac{dv_i(t)}{dt}\Big(\sum_{j=1}^{n} T_{i,j}v_j(t) + i_i^b\Big), \quad S_2 \equiv -\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{dv_i(t)}{dt} T_{i,j} \frac{dv_j(t)}{dt}$$

obteniendo el mayor descenso en la función de energía cuando $\Delta t = \frac{S_1}{S_2}$ (nótese que $S_1 \geq 0$). Se calcula por tanto Δt a partir de las aproximaciones discretas de S_1 y S_2. Los detalles técnicos acerca de este método están recogidos en Talaván y Yáñez [50].

5.3 Implementación de la estrategia Divide-y-Vencerás

La estrategia o esquema *Divide-y-Vencerás* de simulación propuesta en el capítulo 3, que utiliza dos CHNs para resolver el TSP se representa en el diagrama de flujo siguiente:



La subrutina CHN utilizada en el diagrama de flujo para la simulación del modelo de Hopfield queda resumida a través del siguiente pseudo-código:

Algoritmo 3 Simulador de la CHN utilizando el *método de Talaván-Yáñez*

procedure CHN(N, $\mathbf{D}$, *params*)
  u_0, $\mathbf{T}$, $\mathbf{i^b}$, $\mathbf{v_0}$
  $t = 0$, $iter = 0$
  while $iter < maxIter$ & $\text{máx}_i\{|v_{x,i}(t) - v_{x,i}(t+\Delta t)|\} \geq \epsilon$ **do**
    $\frac{du_{x,i}}{dt} = \sum_{y,j} T_{xi,yj} v_{y,j} + i^b_{x,i},\ \forall(x,i)$ ▷ Variación del potencial
    $\frac{dv_{x,i}}{dt} = \frac{2}{u_0} v_{x,i}(1 - v_{x,i}) \frac{du_{x,i}}{dt},\ \forall(x,i)$ ▷ Variación de los estados
    $\Delta t = max\{k >= 0 / v_{x,i} + k\frac{dv_{x,i}}{dt} \in [0,1], \forall(x,i) \in \{1,...,N\}^2\}$
    $S_1 = \sum_{x,i} \frac{dv_{x,i}}{dt} \frac{du_{x,i}}{dt}$
    $S_2 = \sum_{x,i} \sum_{y,j} \frac{dv_{x,i}}{dt} T_{xi,yj} \frac{dv_{y,j}}{dt}$
    if $S_2 > 0$ **then**
      $\Delta t = min\{\Delta t, \frac{S_1}{S_2}\}$ ▷ Cálculo del Δt óptimo
    end if
    $v_{x,i}(t + \Delta t) = v_{x,i}(t) + \frac{dv_{x,i}}{dt}$ ▷ Actualización de los estados
  end while
  return: v
end procedure

El algoritmo 3 considera conocida la matriz $\mathbf{T}$. Sin embargo, en la práctica se quiere evitar calcularla debido a la gran cantidad de memoria necesaria para almacenarla. En su lugar, se calcula directamente el producto de $\mathbf{T} \cdot \mathbf{v}$, según lo visto en el el algoritmo 2 de la sección 5.1. Cuando se quiere resolver el TSP en el que se han fijado K cadenas, como es el caso de la fase $TSP_2^{\not k}$ del esquema *Divide-y-Vencerás*, es necesario modificar el cálculo eficiente de $\mathbf{T} \cdot \mathbf{v}$ del siguiente modo:

Algoritmo 4 Cálculo eficiente de $\mathbf{T} \cdot \mathbf{v}$ para la segunda fase de *Divide-y-Vencerás*

procedure TTIMESV(N, K, $\mathbf{V}$, $\mathbf{D}$, *params*)
  $S_x = \sum_{i=1}^{N-K} v_{x,i} = 1$, $S_i = \sum_{x=1}^{N} v_{x,i} = 1$, $S = \sum_x^N S_x = \sum_i^{N-K} S_i$
  if $x \leq 2K$ **then**
    if $x \equiv 0 \pmod{2K}$ **then**
      $termA = -(S_{x-1} - v_{x-1,i}) - (S_x - v_{x,i})$
    else
      $termA = -(S_{x+1} - v_{x+1,i}) - (S_x - v_{x,i})$
    end if
  else
    $termA = -2(S_x - v_{x,i})$
  end if
  $termD = -\sum_y (d_{x,y^c} v_{y,(i-1) \bmod N-K} + d_{x^c,y} v_{y,(i+1) \bmod N-K})$
  return: $A \cdot termA - B(S_i - v_{x,i}) - CS + D \cdot termD$
end procedure

5.4 Experiencias computacionales

A continuación se muestran las experiencias computacionales que apoyan el estudio llevado a cabo en esta tesis doctoral. Los detalles de implementación de algoritmos se recogen en el apéndice A.

Todas las simulaciones se han llevado a cabo utilizando una estación de trabajo equipada con un procesador Intel Xeon E5-2620 @ 2GHz con 64 GB RAM. Las simulaciones de la sección 5.4.3 se han realizado con una GPU NVIDIA Tesla K80.

En todos los casos, el punto inicial de las simulaciones de la CHN ha sido elegido aleatoriamente en las proximidades del punto de silla, tal y como se vio en la sección 2.6.1, $\mathbf{v_0} \in [\mathbf{v}^* - \varepsilon, \mathbf{v}^* + \varepsilon]$, $\varepsilon > 0$, tomando $\varepsilon = 10^{-10}$. El método de simulación utilizado ha sido el *método de Talaván-Yáñez*, introducido en la sección 5.2.3.

5.4.1 Comparación del modelo Divide-y-Vencerás con la CHN tradicional

Con el objetivo de verificar la eficiencia de este nuevo esquema o arquitectura *Divide-y-Vencerás* y para comprobar si la calidad de las soluciones mejora con respecto al modelo de Hopfield tradicional aplicado al TSP, se han llevado a cabo diversas simulaciones mediante una aplicación desarrollada en MATLAB.

La parametrización considerada en todas las simulaciones utiliza un valor pequeño de $C = 10^{-5}$, donde el parámetro C representa la *inhibición global* de todas las neuronas introducido por Hopfield y Tank [19] (y presentado en la sección 1.3.2). Recordando lo visto en la sección 2.5, un valor pequeño del parámetro libre favorece a que las cuencas de atracción de las mejores soluciones sean mayores.

La Tabla 5.1 compara la calidad de las soluciones obtenidas entre el método clásico de resolución del TSP (utilizando la parametrización de Talaván y Yáñez) y la estrategia de *Divide-y-Vencerás* propuesta en el capítulo 3, donde el valor de τ utilizado se corresponde con el entero más cercano que representa al 10 % de las ciudades, $\tau = [\frac{N}{10}]$. Para medir la calidad de las soluciones, se ha utilizado el cociente de rendimiento (ρ) ya introducido en la sección 2.6.1:

$$\rho = \frac{\text{longitud del tour}}{\text{longitud del tour óptimo}}$$

Cada fila en la Tabla 5.1 comienza con la identificación del TSP (donde se han seleccionado 25 problemas de TSPLIB [38]), el tamaño del problema (N), seguido del número de simulaciones para cada problema. Las columnas restantes incluyen las medidas del cociente de rendimiento (medido a través del mínimo, media y moda) y los tiempos medios de computación (en segundos) para cada problema. Se aprecia cómo el esquema *Divide-y-Vencerás* mejora notablemente los resultados obtenidos con la arquitectura clásica del modelo de Hopfield (lo cuál se hace más evidente a medida que el tamaño del problema crece), siendo los tiempos de ejecución ligeramente superiores en el modelo *Divide-y-Vencerás* (aunque en algunos casos como *GR202* y *GR666* se mejora el tiempo de computación en el modelo tradicional). En ambos casos, el método de simulación de la CHN elegido (en el caso de *Divide-y-Vencerás* cada una de sus fases) ha sido el *método Talaván-Yáñez*.

Tabla 5.1: Experiencias computacionales para instancias de TSPLIB utilizando los modelos *CHN tradicional* y *Divide-y-Vencerás*, con C = 10^{-5}

Identificación del TSP	N	Núm. de Sim.	Modelo de Hopfield tradicional				Modelo Divide-y-Vencerás $\tau = [\frac{N}{10}]$			
			ρ			Tiempo CPU	ρ			Tiempo CPU
			Mínimo	Media	Moda		Mínimo	Media	Moda	
GR24	24	1000	1.04	1.40	1.44	0.08	1.04	1.29	1.23	0.16
FRI26	26	1000	1.19	1.46	1.43	0.09	1.04	1.29	1.26	0.13
BAYG29	29	1000	1.13	1.47	1.42	0.10	1.03	1.24	1.20	0.17
BAYS29	29	1000	1.10	1.52	1.56	0.10	1.05	1.29	1.27	0.16
ATT48	48	1000	1.36	1.80	1.74	0.19	1.17	1.43	1.42	0.32
GR48	48	1000	1.36	1.78	1.74	0.19	1.16	1.45	1.39	0.31
EIL51	51	1000	1.15	1.48	1.46	0.22	1.13	1.45	1.42	0.36
EIL76	76	1000	1.24	1.61	1.65	0.43	1.31	1.57	1.53	0.56
PR76	76	1000	1.72	2.15	2.03	0.44	1.35	1.67	1.67	0.64
GR96	96	1000	1.86	2.33	2.29	0.59	1.43	1.76	1.75	0.95
KROA100	100	1000	2.06	2.60	2.60	0.61	1.50	1.88	1.82	1.02
KROC100	100	1000	2.22	2.74	2.64	0.60	1.49	1.94	1.99	0.94
KROD100	100	1000	1.98	2.47	2.41	0.63	1.45	1.85	1.83	1.07
RD100	100	1000	1.47	1.99	1.89	0.67	1.52	1.82	1.82	1.06
EIL101	101	1000	1.40	1.74	1.68	0.73	1.40	1.62	1.58	1.06
LIN105	105	1000	1.94	2.65	2.62	0.66	1.64	1.98	1.95	1.05
GR120	120	1000	1.88	2.34	2.28	0.85	1.51	1.79	1.80	1.34
CH130	130	1000	1.70	2.20	2.10	1.15	1.55	1.92	1.93	1.32
CH150	150	1000	1.83	2.19	2.15	1.35	1.66	2.01	1.98	1.94
GR202	202	500	1.71	1.98	1.99	4.50	1.58	1.87	1.81	3.38
A280	280	500	2.75	3.21	3.13	4.49	2.03	2.39	2.33	5.78
PCB442	442	500	2.48	2.85	2.88	21.24	2.28	2.53	2.50	25.38
PA561	561	250	2.53	2.91	2.90	39.48	2.32	2.52	2.55	40.74
GR666	666	250	3.10	3.43	3.46	76.03	2.62	2.88	2.89	57.16
PR1002	1002	125	4.23	4.60	4.61	152.17	3.02	3.37	3.32	165.00

Pese a que utilizar valores pequeños de τ para la fase TSP_1^τ puede parecer intuitivamente más apropiado, no está claro cuál debería ser el valor óptimo de τ, o incluso si pudiera existir alguna relación entre el valor óptimo de τ y el número total de ciudades del problema. Para responder a estas preguntas, se lleva a cabo a continuación un estudio más detallado sobre τ, considerando todos sus posibles valores para cada uno de los problemas de TSPLIB.

5.4.2 Modelo Divide-y-Vencerás: optimización de los parámetros

Se han llevado a cabo más de 365,000 simulaciones de CHNs con el objetivo de encontrar los valores óptimos de τ para cada uno de los problemas considerados. En todas las simulaciones, se obtuvo una solución factible. La figura 5.1 muestra el cociente de rendimiento para todos los posibles valores de τ para los problemas de TSPLIB *ATT48* y *CH150* (con 48 y 150 ciudades respectivamente).

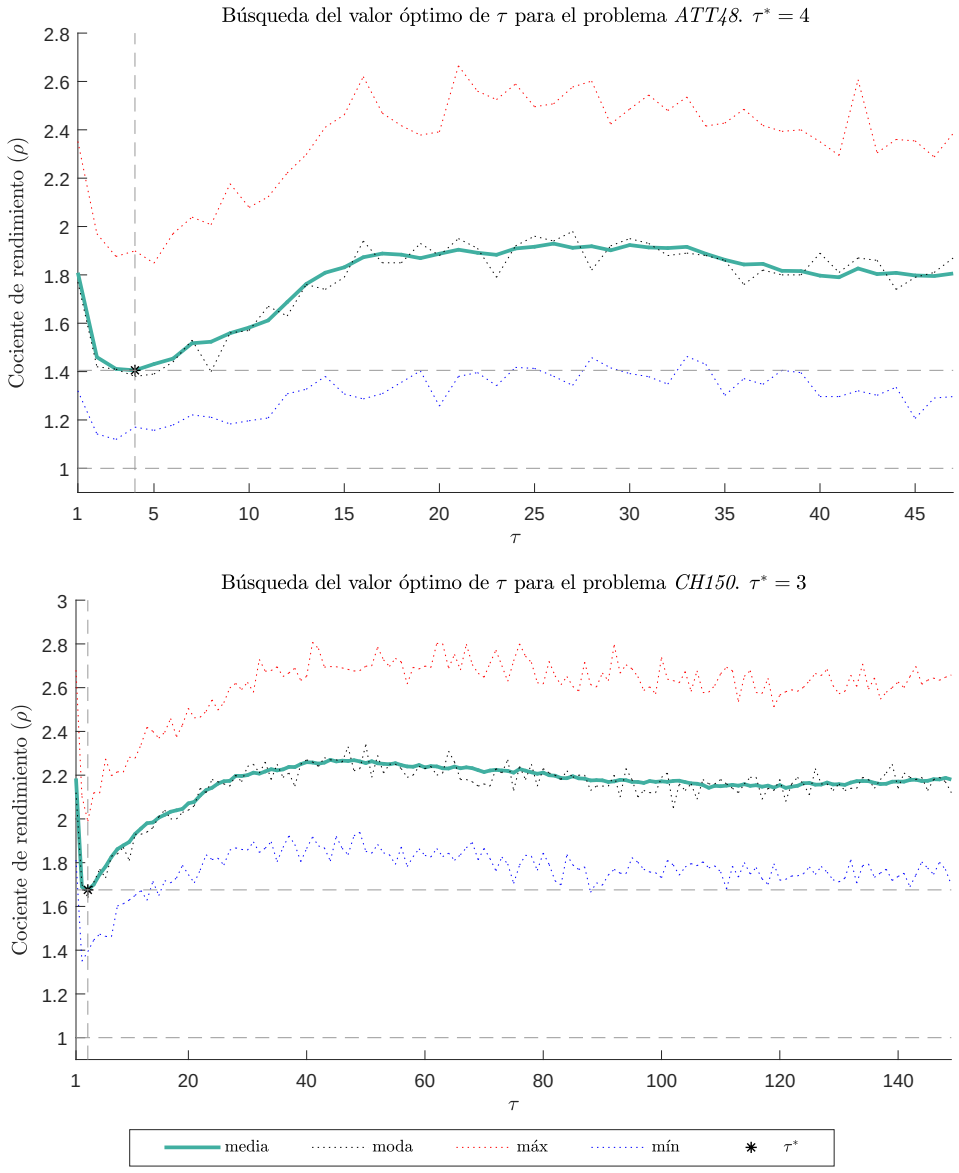


Figura 5.1: Las estadísticas para el cociente de rendimiento de la CHN muestran su dependencia de τ. La figura muestra que, para los dos problemas de TSPLIB, cada problema tiene su valor adecuado de τ.

La tabla 5.2 compara la calidad de la solución y el rendimiento computacional para cada problema utilizando el modelo *Divide-y-Vencerás*, considerando tanto el valor de τ más próximo al entero que representa el 10 % de las ciudades, $\tau = [\frac{N}{10}]$, como el valor óptimo de τ, τ^*, para la fase 1 del modelo, TSP_1^τ. Se observa cómo la calidad de la solución mejora considerablemente a expensas del aumento en el tiempo medio de cálculo, cuando se utilizan valores de τ próximos a su valor óptimo. En este caso, la complejidad de la red en términos del número de simetrías del problema aumenta (ya que las distancias son más parecidas), obligando a la red a producir más iteraciones para llegar a la solución. El hecho de que la calidad de las soluciones se pueda mejorar mediante el aumento del tiempo de cálculo se convierte en una nueva característica para la CHN. Otro punto interesante es que no parece haber una relación directa entre τ^* y el número total de ciudades consideradas, sino que son valores pequeños de τ los que producen mejores resultados.

Tabla 5.2: Experiencias computacionales para instancias de TSPLIB utilizando el modelo *Divide-y-Vencerás* utilizando distintos valores de τ, con C = 10^{-5}

Identificación del TSP	N	Núm. de Sim.	Modelo Divide-y-Vencerás $\tau = [\frac{N}{10}]$				Modelo Divide-y-Vencerás $\tau = \tau^*$				
			ρ			Tiempo CPU	ρ				Tiempo CPU
			Mínimo	Media	Moda		Mínimo	Media	Moda	τ^*	
GR24	24	1000	1.04	1.29	1.23	0.16	1.04	1.29	1.23	2	0.16
FRI26	26	1000	1.04	1.29	1.26	0.13	1.04	1.29	1.26	3	0.13
BAYG29	29	1000	1.03	1.24	1.20	0.17	1.03	1.24	1.20	3	0.17
BAYS29	29	1000	1.05	1.29	1.27	0.16	1.05	1.29	1.27	3	0.16
ATT48	48	1000	1.17	1.43	1.42	0.32	1.17	1.41	1.38	4	0.35
GR48	48	1000	1.16	1.45	1.39	0.31	1.14	1.39	1.39	4	0.33
EIL51	51	1000	1.13	1.45	1.42	0.36	1.12	1.44	1.46	4	0.39
EIL76	76	1000	1.31	1.57	1.53	0.56	1.16	1.49	1.46	2	1.01
PR76	76	1000	1.35	1.67	1.67	0.64	1.26	1.62	1.58	2	1.11
GR96	96	1000	1.43	1.76	1.75	0.95	1.33	1.66	1.61	3	1.50
KROA100	100	1000	1.50	1.88	1.82	1.02	1.38	1.84	1.90	4	1.40
KROC100	100	1000	1.49	1.94	1.99	0.94	1.38	1.81	1.80	3	1.69
KROD100	100	1000	1.45	1.85	1.83	1.07	1.39	1.75	1.74	5	1.07
RD100	100	1000	1.52	1.82	1.82	1.06	1.20	1.58	1.54	2	1.85
EIL101	101	1000	1.40	1.62	1.58	1.06	1.25	1.51	1.52	3	1.70
LIN105	105	1000	1.64	1.98	1.95	1.05	1.44	1.85	1.85	3	1.69
GR120	120	1000	1.51	1.79	1.80	1.34	1.37	1.70	1.67	4	2.08
CH130	130	1000	1.55	1.92	1.93	1.32	1.34	1.63	1.61	3	2.56
CH150	150	1000	1.66	2.01	1.98	1.94	1.39	1.68	1.64	3	3.36
GR202	202	500	1.58	1.87	1.81	3.38	1.31	1.51	1.53	2	8.94
A280	280	500	2.03	2.39	2.33	5.78	1.67	2.02	2.02	2	18.28
PCB442	442	500	2.28	2.53	2.50	25.38	1.60	1.87	1.84	3	77.54
PA561	561	250	2.32	2.52	2.55	40.74	1.66	1.87	1.86	3	186.08
GR666	666	250	2.62	2.88	2.89	57.16	1.76	1.96	1.98	4	293.64
PR1002	1002	125	3.02	3.37	3.32	165.00	2.10	2.29	2.24	6	819.57

De nuevo, cada fila de la tabla 5.2 comienza con la identificación del TSP (utilizando la misma selección de problemas de TSPLIB que en la tabla 5.1) y el número de ciudades, seguida del número de simulaciones para cada problema. El resto de columnas incluyen estadísticos relativos al cociente de rendimiento (mínimo, media y moda) y tiempo medio de cálculo (en segundos) para cada problema. También se incluyen los valores óptimos de τ encontrados para cada problema.

Se han resuelto también problemas de mayor tamaño, como el problema de TSPLIB *USA13509*, que representa las 13509 ciudades de los Estados Unidos continentales que tienen una población de 500 o más habitantes, se resolvió utilizando tanto el modelo tradicional de la CHN como con el modelo *Divide-y-Vencerás* ($\tau = [\frac{N}{10}]$), con cocientes de rendimiento de 13.84 y 8.87, respectivamente (ver tabla 5.3). Recuérdese, tal y como se indicó en la sección 5.1 que la correspondiente red de Hopfield tiene más de 180 millones de pesos y su matriz de pesos, de almacenarla en memoria (en doble precisión), ocuparía más de 260 PB.

Tabla 5.3: Comparación de la calidad de la solución para TSPs de gran tamaño utilizando los algoritmos *Talaván-Yáñez* y *Divide-y-Vencerás*

Identificación del TSP	N	Núm. de Sim.	Modelo de Hopfield tradicional			Modelo Divide-y-Vencerás $\tau = [\frac{N}{10}]$		
			ρ		Tiempo CPU	ρ		Tiempo CPU
			Mínimo	Media		Mínimo	Media	
PR2392	2392	25	5.44	5.70	21m 18s	4.32	4.50	27m 12s
PLA7397	7397	5	14.50	14.77	15h 58m	7.04	7.15	16h 06m
USA13509	13509	1	13.84	13.84	7d 02h 15m	8.87	8.87	6d 11h 32m

El algoritmo propuesto se ha comparado también con el problema de 10 ciudades introducido por Wilson y Pawley [58] y utilizado por Jolai [20], obteniendo no sólo mejores soluciones, sino la solución óptima:
Valor de la función objetivo = 2.6908, Tour = *e–d–a–c–b–j–i–h–g–f*.

Mientras que otros algoritmos basados en el modelo de Hopfield (que utilizan o no métodos híbridos) podrían quizá obtener mejores soluciones para el TSP (ver por ejemplo Mérida-Casermeiro et al. [30] y Wang et al. [54]), el método propuesto explota por completo las características de la red de Hopfield continua y la simula en consecuencia. La factibilidad de las soluciones obtenidas queda garantizada por la apropiada parametrización de la red.

5.4.3 Computación en la GPU

La *computación en la GPU* o *GPU Computing* consiste en el uso de la unidad de procesamiento gráfico (*Graphics Processing Unit –GPU–*) en combinación con la CPU para la resolución de problemas en el ámbito científico y de simulación. La *computación en la GPU* traslada las secciones del código con mayor carga computacional a la GPU, ejecutando el resto de la aplicación y comandando la misma desde la CPU. Las GPUs fueron originalmente diseñadas para la aceleración y renderización gráfica (proceso de generación de imágenes), pero su *array* masivo de procesadores en paralelo (con varios cientos o miles de procesadores) las convierte en herramientas muy adecuadas para llevar a cabo operaciones con matrices. Por contra, la CPU está formada por varios núcleos, los cuales están optimizados para el procesamiento en serie.

En el caso del modelo de Hopfield, el uso de la GPU se hace especialmente interesante para la resolución de problemas de gran tamaño, donde el coste de la transferencia de datos entre la CPU y GPU (y de vuelta a la GPU) se vuelve insignificante con respecto al rendimiento adicional obtenido, por llevar a cabo los cálculos en la GPU.

La tabla 5.4 muestra una comparativa de tiempos medios de ejecución del modelo de Hopfield en la CPU y GPU para instancias grandes de TSPLIB. En ambos casos, se utiliza la misma semilla de generación de números aleatorios, de modo que el resultado obtenido por la CHN es el mismo en ambos casos y se pueden comparar los tiempos de ejecución. En las simulaciones se ha utilizado una GPU NVIDIA K80, con 4992 núcleos de procesamiento paralelo CUDA® y 12 GB de memoria GDDR5.

Tabla 5.4: Comparación de tiempos medios de ejecución de la CHN en la CPU y GPU para instancias medias y grandes de TSPLIB

Identificación del TSP	N	Núm. de Sim.	Modelo de Hopfield tradicional	
			Tiempo CPU	Tiempo GPU
GR202	202	500	4.50s	6.53s
A280	280	500	4.49s	4.54s
PCB442	442	500	21.24s	6.95s
PA561	561	250	39.48s	9.55s
GR666	666	250	1m 16s	14.87s
PR1002	1002	125	2m 45s	17.75s
PR2392	2392	25	21m 18s	2m 07s
PLA7397	7397	5	15h 58m	1h 12m 22s
USA13509	13509	1	7d 02h 15m	8h 39m 43s

La figura 5.2 compara los resultados de tiempo de ejecución en la CPU y GPU. Para problemas pequeños del TSP, $N < 280$, no compensa realizar la ejecución en la GPU. Esto se debe a que el número de operaciones se realizan en la GPU no compensa el coste de transferencia de datos de la memoria de la CPU a la GPU. Se aprecia cómo la mejora computacional es notable a medida que el tamaño del problema es cada vez mayor.

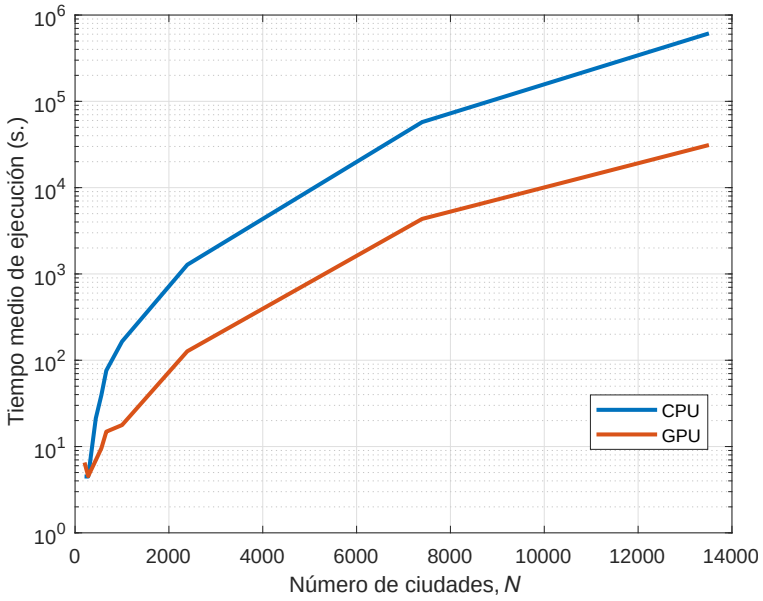


Figura 5.2: Comparación de tiempos medios de ejecución (en escala logarítmica) recogidos en la tabla 5.4 para instancias medias y grandes utilizando la CPU y GPU

5.4.4 El modelo de Hopfield como 2-opt

El *2-opt* es una de las heurísticas más utilizadas para resolver el TSP, tanto como heurística por sí sola, como formando parte de un modelo híbrido en el que su cometido es la mejora de la solución obtenida. En el capítulo 4 se analizó en detalle cómo el modelo de Hopfield con 4 ciudades y 2 cadenas fijadas se comporta exactamente igual que un *intercam-*

bio 2-opt, siempre y cuando el punto inicial sea escogido convenientemente. Puede mejorarse así la solución de la CHN deshaciendo todos los cruces del tour obtenido mediante *intercambios 2-opt* utilizando el modelo de Hopfield, encadenando sucesivamente segundas fases ($TSP_2^{\not\Delta}$) del modelo *Divide-y-Vencerás* hasta deshacer todos los cruces de la solución de la CHN original.

La tabla 5.5 muestra las experiencias computacionales del modelo de Hopfield tradicional (mostradas en la tabla 5.1) sobre las que posteriormente se ha aplicado el *2-opt* utilizando el modelo de Hopfield, aplicando sucesivamente *intercambios 2-opt* consistentes en modelos de Hopfield con 4 ciudades en los que se han fijado 2 cadenas (utilizando como punto inicial la proyección del punto de silla del problema sobre una de las caras).

Tabla 5.5: Experiencias computacionales para instancias de TSPLIB utilizando los modelos *CHN tradicional* y *CHN tradicional + 2-opt*, con C = 10^{-5}

Identificación del TSP	N	Núm. de Sim.	Modelo de Hopfield tradicional			Modelo de Hopfield tradicional + 2-opt		
			ρ			ρ		
			Mínimo	Media	Moda	Mínimo	Media	Moda
GR24	24	1000	1.04	1.40	1.44	1	1.03	1.01
FRI26	26	1000	1.19	1.46	1.43	1	1.04	1.02
BAYG29	29	1000	1.13	1.47	1.42	1	1.03	1.02
BAYS29	29	1000	1.10	1.52	1.56	1	1.03	1.02
ATT48	48	1000	1.36	1.80	1.74	1	1.04	1.04
GR48	48	1000	1.36	1.78	1.74	1	1.04	1.03
EIL51	51	1000	1.15	1.48	1.46	1	1.06	1.06
EIL76	76	1000	1.24	1.61	1.65	1.02	1.08	1.07
PR76	76	1000	1.72	2.15	2.03	1	1.05	1.04
GR96	96	1000	1.86	2.33	2.29	1.01	1.07	1.06
KROA100	100	1000	2.06	2.60	2.60	1	1.08	1.07
KROC100	100	1000	2.22	2.74	2.64	1	1.08	1.08
KROD100	100	1000	1.98	2.47	2.41	1.01	1.07	1.06
RD100	100	1000	1.47	1.99	1.89	1.01	1.09	1.08
EIL101	101	1000	1.40	1.74	1.68	1.02	1.08	1.08
LIN105	105	1000	1.94	2.65	2.62	1	1.07	1.08
GR120	120	1000	1.88	2.34	2.28	1.02	1.08	1.08
CH130	130	1000	1.70	2.20	2.10	1.02	1.08	1.07
CH150	150	1000	1.83	2.19	2.15	1.03	1.10	1.08
GR202	202	500	1.71	1.98	1.99	1.03	1.08	1.07
A280	280	500	2.75	3.21	3.13	1.06	1.12	1.12
PCB442	442	500	2.48	2.85	2.88	1.06	1.11	1.11
PA561	561	250	2.53	2.91	2.90	1.08	1.12	1.12
GR666	666	250	3.10	3.43	3.46	1.08	1.11	1.10
PR1002	1002	125	4.23	4.60	4.61	1.07	1.10	1.10
PR2392	2392	25	5.44	5.70	5.63	1.11	1.13	1.13
PLA7397	7397	5	14.50	14.77	14.50	1.10	1.11	1.11
USA13509	13509	1	13.84	13.84	13.84	1.11	1.11	1.11

CAPÍTULO 6

Conclusiones y líneas futuras

Este capítulo presenta las conclusiones fruto del desarrollo de esta tesis doctoral así como las líneas futuras de investigación.

Contenidos del capítulo

6.1 Conclusiones

Esta memoria se centra en el uso del *modelo de Hopfield continuo* o *red de Hopfield continua* (CHN) para la resolución de problemas de optimización combinatoria, haciendo especial hincapié en el problema del viajante (TSP), al ser éste uno de los problemas de referencia en los ámbitos de la optimización y la computación. El modelo de Hopfield consiste en una red recurrente totalmente interconectada con una ecuación diferencial asociada cuyos estados evolucionan desde un punto inicial a un punto de equilibrio mediante la minimización de una función de Lyapunov. El modelo de Hopfield podrá resolver el problema de optimización combinatoria si se es capaz de escoger los pesos y entradas externas de la red de modo que la función de Lyapunov coincida con la función objetivo y restricciones del problema de optimización. Este proceso se conoce como proyección del problema de optimización en la CHN. En el caso del TSP, Hopfield propuso una proyección dependiente de 4 parámetros, de los cuales, con el fin de garantizar que la CHN sea estable en las soluciones factibles e inestable en las no factibles, únicamente uno de los parámetros originales de Hopfield queda libre (relacionado con la "inhibición global" de la red). Históricamente, investigadores en este campo han utilizado valores pequeños del parámetro libre, dado que aparentemente aportaban mejores soluciones, pero sin explicar por qué razón.

En esta memoria, se expone y demuestra el motivo por el cual valores pequeños del parámetro libre producen mejores soluciones. Esto se debe a que el parámetro libre (y en general todos los parámetros del problema) están directamente relacionados con el punto de silla de la función de energía, punto donde confluyen todas las cuencas de atracción. El valor del parámetro libre modifica no sólo el punto de silla, sino también las cuencas de atracción. Por esta razón, seleccionar el punto inicial de la simulación en las proximidades del punto de silla (sin escogerlo, al ser un punto de equilibrio), será una buena estrategia para resolver el problema de optimización mediante la CHN.

Además, cuando se resuelven problemas de optimización sencillos, como es el caso del ejemplo del problema de asignación cuadrática generalizado (GQKP) presentado en la sección 2.2, es posible obtener siempre la solución óptima del problema, elegido un valor adecuado del parámetro libre, en base a su relación con el punto de silla (ver sección 2.3). Esto es gracias a que el punto de silla puede ser movido fuera del Hipercubo de Hamming, para un valor pequeño de dicho parámetro, lo cual deja todo el Hipercubo de Hamming dentro de la cuenca de atracción de la solución óptima (ver sección 2.3.1). El hecho de que la CHN pueda garantizar, para problemas sencillos, obtener siempre la solución óptima, se convierte en una nueva característica de la CHN. En problemas con un alto número de simetrías, como es el caso del TSP, esto se vuelve más complicado. Sin embargo, se encuentra la relación entre el parámetro libre y el punto de silla de la CHN aplicado al TSP (ver sección 2.5), no pudiendo, en este caso, mover el punto de silla lo suficiente al disminuir el parámetro libre. No obstante, se muestra que en los alrededores del punto de silla las cuencas de atracción para las mejores soluciones crecen a medida que el parámetro libre decrece.

De la relación entre el parámetro libre y el punto de silla de la CHN aplicado al TSP se obtiene también un resultado importante, y es que no es necesario calcular la inversa de la matriz de pesos, de tamaño $N^2 \times N^2$. En su lugar, bastará con calcular la inversa de

una matriz $N \times N$, lo cual reduce considerablemente la exigencia computacional asociada a calcular el punto de silla.

Con el objetivo de continuar mejorando la calidad de la solución de la CHN aplicado al TSP (elegir un valor pequeño del parámetro libre y seleccionar el punto inicial en las proximidades del punto de silla ya mejora notablemente la calidad de la solución), se propone una nueva estrategia *Divide-y-Vencerás*. Del mismo modo que un viajero puede planear su viaje organizando primero visitas a ciudades vecinas y después organizando viajes entre ciudades con distancias más largas, esta estrategia consiste en unir primero las ciudades más cercanas entre sí para formar un conjunto de cadenas y ciudades aisladas, para después combinar las ciudades extremo de cadena y ciudades aisladas para construir el recorrido final. Cada fase de la estrategia *Divide-y-Vencerás* es un TSP diferente que puede ser resuelto independientemente, utilizando algoritmos exactos o heurísticos.

En esta memoria, se utiliza la CHN para resolver ambos problemas, para lo cual se deducen las dos parametrizaciones asociadas a cada una de las CHNs (ver secciones 3.2 y 3.3). Además, cabe destacar la idea de conectar ciudades vecinas en la primera fase de la heurística mediante la modificación de la matriz de distancias, puesto que la CHN es capaz de resolver un caso particular del TSP: conectar el subconjunto de ciudades que están más próximas entre sí.

El algoritmo *2-opt* es una de las heurísticas más utilizadas para resolver el TSP, tanto por sí sola, como formando parte de un procedimiento híbrido que permita mejorar la solución del problema. La idea principal de esta heurística es reordenar los tramos del recorrido de modo que la ruta no se cruce sobre sí misma. Para ello, se llevan a cabo sucesivos intercambios en el orden del tour, procedimiento conocido como *intercambio 2-opt*. En este sentido, se demuestra que la CHN con 4 ciudades y 2 cadenas fijadas (caso particular de la segunda fase de la heurística *Divide-y-Vencerás*), se comporta exactamente igual que un *intercambio 2opt* seleccionando adecuadamente el punto inicial de la simulación (ver secciones 4.3 y 4.4). Para ello, es necesario estudiar el punto de silla de la segunda fase de la estrategia *Divide-y-Vencerás* (sección 4.1)

Desde su concepción, el modelo de Hopfield continuo ha tenido muchas críticas por su, en origen, incapacidad de garantizar soluciones factibles. Una vez resuelto este problema por Talaván y Yáñez (cuya solución fue presentada en la sección 2.5) mediante una parametrización que garantiza la factibilidad, la mayor parte de las críticas se han centrado en que la calidad de las soluciones del modelo de Hopfield no eran competitivas con respecto a otras heurísticas. Pese a todo, el interés por el modelo de Hopfield permanece activo.

Esta memoria puede resumirse en 3 resultados principales, que ayudan a mejorar la calidad de la solución de la CHN como heurística:

1. El análisis del punto de silla de la CHN y su relación con el parámetro libre, que permite elegir un punto inicial adecuado.

2. La estrategia *Divide-y-Vencerás*, que conecta en primer lugar ciudades próximas y posteriormente núcleos de ciudades mediante el uso de sendas CHNs.

3. Y el modelo de Hopfield como *2-opt*, caso particular de la segunda fase de la estrategia *Divide-y-Vencerás* que permite a la CHN comportarse como un intercambio *2-opt*.

Estos resultados teóricos han sido contrastados mediante exhaustivas experiencias computacionales (capítulo 5), complementando esta memoria con la librería *Hopfield Network Toolbox* (apéndice A).

6.2 Líneas futuras de investigación

Los resultados aportados en esta memoria han abierto algunas líneas de investigación que deben ser continuadas:

1. La elección del punto inicial en las proximidades del punto de silla, para un valor pequeño del parámetro libre, ayuda a mejorar notablemente la calidad de la solución de la CHN aplicado al TSP. Esto es debido a que las cuencas de atracción para las mejores soluciones se agrandan en las proximidades del punto de silla cuando el valor del parámetro libre decrece. Estudiar la amplitud de las cuencas (o ángulos entre ellas –utilizando la función de activación lineal saturada las fronteras de las cuencas de atracción son rectilíneas–) podrá ayudar a seleccionar puntos iniciales más apropiados, que los elegidos aleatoriamente en las proximidades del punto de silla.

   Adicionalmente, se considera modificar o re-escalar la matriz de distancias del TSP de modo que las distancias pequeñas sumen potencia y las largas resten, lo cual parece que favorecería para obtener mejores soluciones.

2. Habiendo resuelto la incógnita acerca de la importancia del parámetro libre en la CHN, se abre una nueva pregunta acerca del valor óptimo del parámetro τ, que conecta vecinos en la primera fase de *Divide-y-Vencerás*. Estudiar la relación entre el TSP y el valor óptimo de τ para cada problema, o si existe algún tipo de métrica relacionada con el desorden o entropía de las ciudades del problema, queda como una pregunta abierta. Además, la automatización de las dos CHNs de la estrategia *Divide-y-Vencerás* como una única CHN queda también también como una línea futura de trabajo.

3. El uso de la CHN como *2-opt* requiere resolver sucesivos CHNs con 4 ciudades y 2 cadenas fijadas para poder deshacer cada uno de los cruces de la solución inicial. Construir una CHN que permita resolver todos los intercambios *2-opt* de una sola vez podrá convertir en competitiva (estrictamente desde el punto de vista computacional) a la CHN con respecto al *2-opt* tradicional.

4. Por último, la extensión de la CHN para resolver *intercambios 3-opt*, o más generalmente *intercambios k-opt* se abre como una prometedora futura línea de investigación.

APÉNDICE A

Hopfield Network Toolbox

El Apéndice A recoge una guía de uso de la librería Hopfield Network Toolbox. Esta librería ha sido desarrollada para complementar esta memoria y aportar resultados prácticos que ayuden a comprender los resultados teóricos.

Contenidos del capítulo

A.1 Introducción y motivación

Todas las simulaciones así como los algoritmos necesarios para el desarrollo de esta tesis doctoral han sido llevados a cabo con MATLAB® y Simulink®, herramientas desarrolladas por The MathWorks, Inc. (3 Apple Hill Drive, Natick, MA 01760, USA).

Entre sus múltiples librerías, MathWorks dispone de Neural Network Toolbox™. En ella, la función `newhop` permite diseñar una red neuronal de Hopfield recurrente. Sin embargo, esta función está únicamente incluida por razones históricas y fines didácticos, tal y como se menciona en su documentación. Es más, la red permite únicamente resolver el modelo de Hopfield discreto.

Por este motivo, durante el transcurso de esta tesis doctoral se ha desarrollado una librería para MATLAB especialmente enfocada en el modelo de Hopfield continuo, llamada *Hopfield Network Toolbox*.

A.2 Descarga e instalación

La librería *Hopfield Network Toolbox* está alojada como código abierto en la plataforma de desarrollo colaborativo GitHub:

`https://github.com/mathinking/HopfieldNetworkToolbox`

La totalidad del código se encuentra licenciado bajo *BSD-2-Clause*. Ello permite su redistribución y uso (tanto código fuente como ficheros binarios), con o sin modificación siempre y cuando se mantengan las condiciones de la licencia, detalladas en el siguiente enlace:
`https://github.com/mathinking/HopfieldNetworkToolbox/blob/master/LICENSE`
Se recomienda siempre utilizar la versión más reciente de la librería, que a su vez tendrá como requisito utilizar cierta versión de MATLAB (la cual estará indicada en el fichero `README.md` disponible en el repositorio).

Opciones de instalación

Podrá descargar la totalidad del código fuente o un único fichero instalador que permita añadir la librería directamente a su instalación de MATLAB a través del siguiente enlace:

`https://github.com/mathinking/HopfieldNetworkToolbox/releases/latest`

1. En caso de descargar el código fuente, basta con ejecutar:

   ```
   >> setup_hopfieldNetwork
   ```

   para completar la instalación de ficheros y añadir todos los directorios al *path* de MATLAB.

2. Alternativamente, si ha descargado el fichero `.mltbx`, bastará con hacer doble click en el propio fichero para instalar la Toolbox.

A.3 Funcionalidades básicas

En primer lugar, se detallan paso a paso algunas funcionalidades básicas de *Hopfield Network Toolbox*, como son sendos modelos de Simulink para simular el modelo de Hopfield en su versión discreta y continua, una interfaz gráfica de usuario para la resolución del TSP y la resolución de problemas del viajante en los que las coordenadas del problema ocupan los vértices de un polígono regular.

A.3.1 Modelos de Simulink

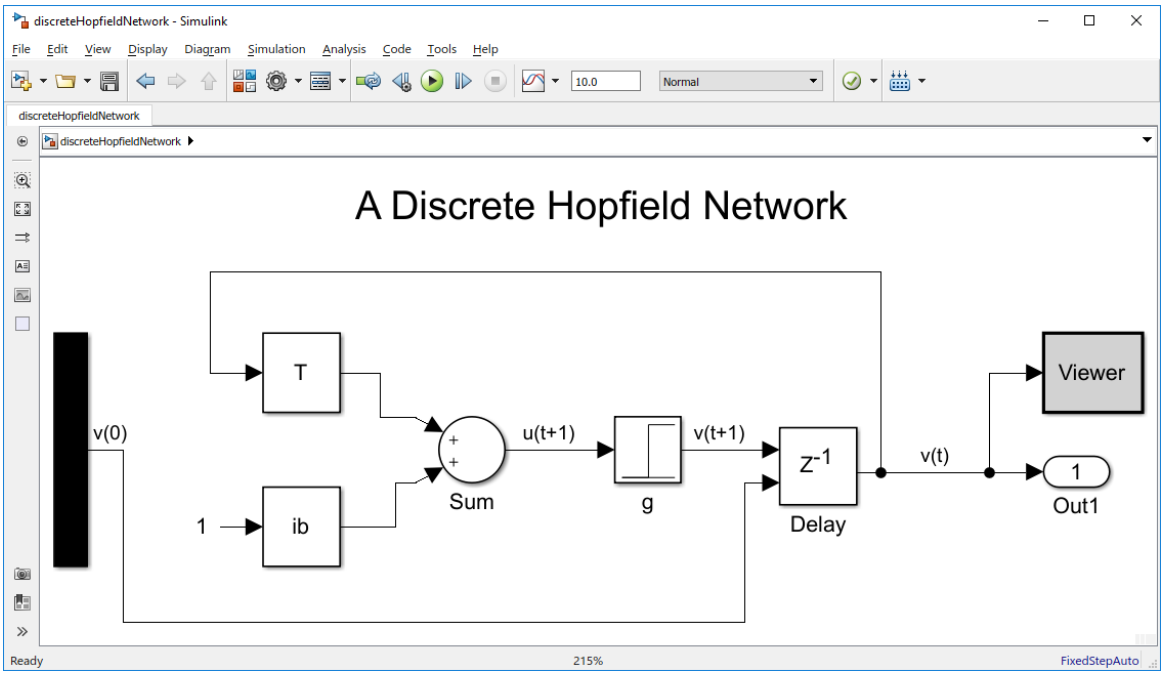
Se presentan a continuación dos modelos de Simulink para resolver tanto el modelo de Hopfield discreto y como el modelo de Hopfield continuo y, que principalmente, ayudarán a entender el sistema dinámico asociado a la red neuronal.

Modelo de Simulink del modelo de Hopfield discreto

En primer lugar, se abre el modelo de Hopfield discreto en Simulink, escribiendo:

```
1 discreteHopfieldNetwork
```

obteniendo el siguiente diagrama (nótese la semejanza con la figura 1.13):



A continuación, se plantea resolver en Simulink el ejemplo 1.3 mostrado en la sección 1.3.1, que debe recordar los 4 patrones $\mathbf{q^0}$, $\mathbf{q^1}$, $\mathbf{q^2}$ y $\mathbf{q^3}$. Para ello, es necesario construir la matriz $\mathbf{T}$ y el vector $\mathbf{i^b}$ (ver ecuación 1.1), requeridos para la simulación del sistema dinámico.

```
q0 = [0;1;1;1;1;0;1;0;0;0;0;1;1;0;0;0;0;1;1;0;0;0;0;1;0;1;1;1;1;0];
q1 = [0;0;0;0;0;0;1;0;0;0;0;0;1;1;1;1;1;1;0;0;0;0;0;0;0;0;0;0;0;0];
q2 = [1;0;0;0;0;0;1;0;0;1;1;1;1;0;0;1;0;1;1;0;0;1;0;1;0;1;1;0;0;1];
q3 = [0;0;0;0;0;0;1;0;1;0;0;1;1;0;1;0;0;1;1;1;1;1;1;1;0;0;0;0;0;0];
patterns = [q0,q1,q2,q3];
n = length(q0);
T = (2*q0 - 1)*(2*q0 - 1)' + (2*q1 - 1)*(2*q1 - 1)' + ...
    (2*q2 - 1)*(2*q2 - 1)' + (2*q3 - 1)*(2*q3 - 1)' - eye(n);
ib = zeros(n,1);
```

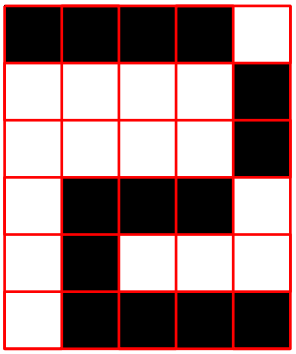
Se utiliza como valor inicial de la simulación el siguiente vector $\mathbf{v_0}$:

```
v0 = [0;0;1;0;0;0;1;0;0;0;0;1;1;0;0;1;0;1;1;0;0;1;0;1;0;1;1;0;0;0];
```

para a continuación simular el sistema dinámico, o bien pulsando sobre el botón de *Play* del modelo o bien ejecutando:

```
[t,y] = sim('discreteHopfieldNetwork');
```

obteniéndose como salida el patrón:



Adicionalmente, puede comprobarse programáticamente el patrón obtenido:

```
patternFound = find(all(patterns - y(end,:)' == 0))-1
```

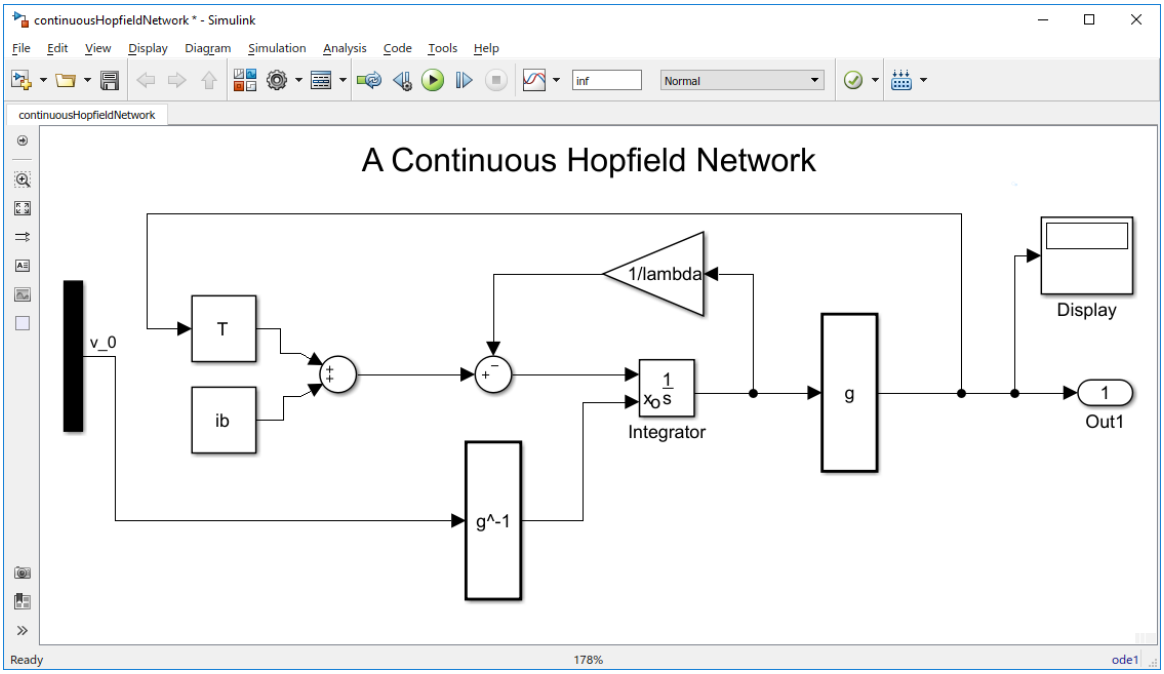
```
patternFound =

     2
```

Modelo de Simulink del modelo de Hopfield continuo

Hopfield Network Toolbox incorpora, además del caso discreto, un modelo de Simulink para la resolución del modelo de Hopfield continuo.

En primer lugar, para abrir el modelo de Hopfield continuo (basado en el sistema dinámico de la ecuación 1.2) en Simulink, basta escribir:

```
continuousHopfieldNetwork
```



Se pretende resolver el GQKP introducido en la sección 2.2, cuya función de energía está caracterizada (ver ecuación 2.5) por:

```
P = [4,0;0,-2];
q = [0;0];
R = [1,1];
b = 1;
```

A continuación, en base a la parametrización obtenida en la ecuación 2.10, se construye la matriz de pesos $\mathbf{T}$ y vector de entradas externas, $\mathbf{i^b}$ necesarios para la simulación de la CHN.

```
alpha = 1;
Phi = 3;
eps = 3*alpha/2 + Phi/2;
beta = -alpha/2 - Phi;
Gamma(1,1) = (2*alpha + Phi/2);
Gamma(2,2) = Phi/2 - alpha;

T  = -(alpha * P + R'*Phi*R - 2*Gamma);
ib = -(alpha * q + R'*beta + diag(Gamma));
```

Finalmente, se indica el punto inicial, el paso de integración, el valor de λ y se simula el sistema dinámico:

```
v0 = [0.6;0.2];
lambda = 1e5;
dt = 0.001;
output = sim('continuousHopfieldNetwork','StopTime','10');
output.yout(end,:)'
```

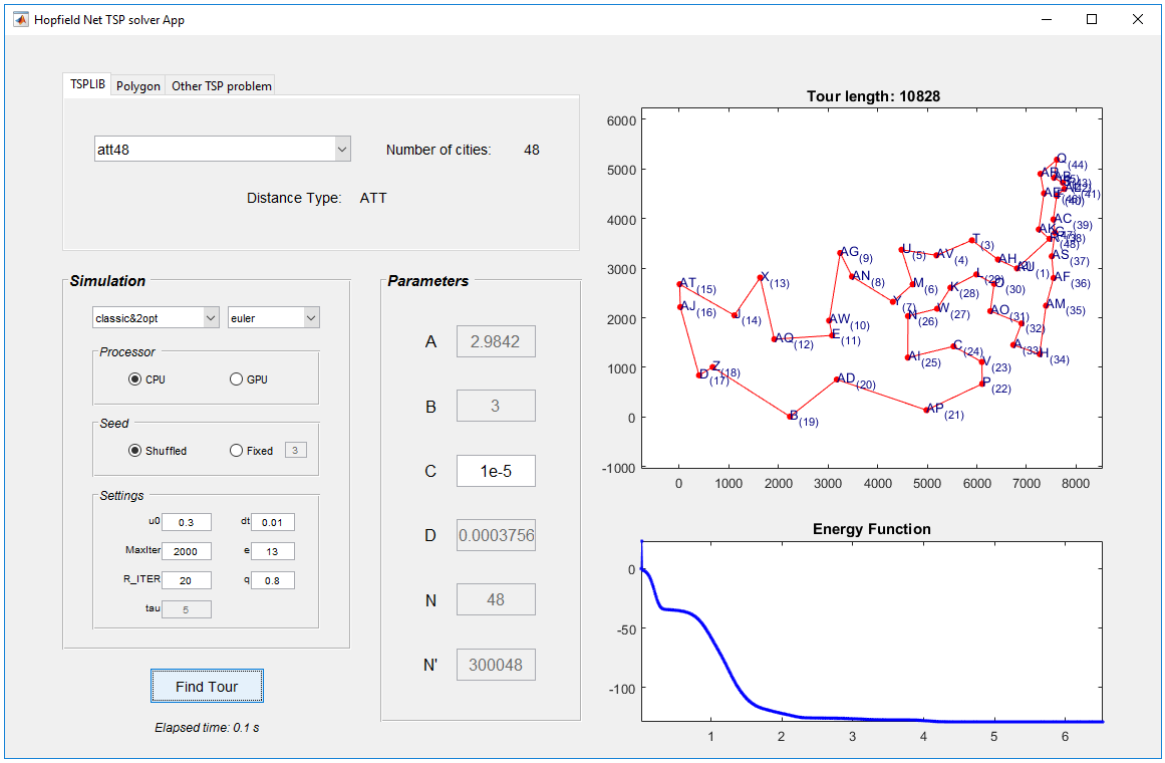
obteniendo la solución óptima del problema:

```
ans =

     0
     1
```

A.3.2 Hopfield Net TSP Solver App

Se incluye en *Hopfield Network Toolbox* una interfaz gráfica de usuario o *App* para resolver interactivamente el TSP utilizando el modelo de Hopfield. El *App* puede abrirse seleccionando *Hopfield Net TSP Solver App* desde la sección de APPs en MATLAB, o escribiendo en la ventana de comandos:

```
>> HopfieldNetworkTSPApp
```



La aplicación permite resolver el TSP utilizando varios esquemas o arquitecturas de la CHN: el modelo tradicional (`'classic'`), el modelo *Divide-y-Vencerás* (`'divide-conquer'`) y las combinaciones de estos con el *2-opt* (`'classic&2opt'` y `'divide-conquer&2opt'`). En cada caso, el método de simulación podrá ser *el método de Euler* (`'euler'`, ver sección 5.2.1), *el método de Runge-Kutta* (`'runge-kutta'`, ver sección 5.2.2) o el *método de Talaván-Yáñez* (`'talavan-yanez'`, ver sección 5.2.3).

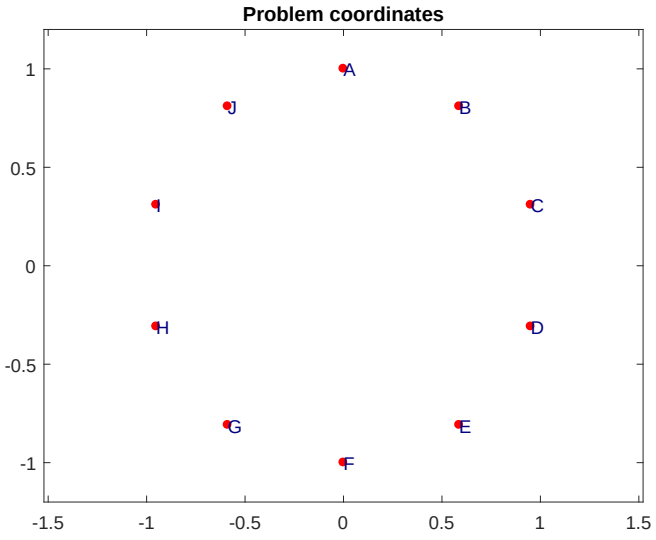
Los TSPs que puede resolver la aplicación son: los recogidos en TSPLIB [38], aquellos en los que las ciudades se sitúan a distancia 1 del origen de coordenadas en los vértices de un

polígono regular y cualquier problema arbitrario definido a partir de sus coordenadas o matriz de distancias (asumiendo distancia euclídea), siempre y cuando estén convenientemente creadas en el *Workspace* de MATLAB.

A.3.3 CNH aplicado al TSP con coordenadas en los vértices de un polígono regular

Se consideran en primer lugar el conjunto de problemas del viajante sencillos, en los que las coordenadas del problema se sitúan a distancia 1 del origen de coordenadas en los vértices de un polígono regular. Para construir la correspondiente CHN asociada al TSP anterior, basta con construir el objeto `net` a través de la función `tsphopfieldnet`, que recibe el número de ciudades del polígono regular, N y el valor del parámetro libre C:

```
rng(2);     % Para poder reproducir el problema
N = 10;     % Número de ciudades. Tamaño de la red: NxN
C = 1e-5;   % Parámetro libre C

 % Creación del objeto de la clase HopfieldNetworkTSP
net = tsphopfieldnet(N, C);

 % Visualización de las coordenadas antes de llevar a cabo la simulación
plot(net)
```



Construida la red neuronal, el siguiente paso consiste en entrenarla. A efectos de la CHN, el entrenamiento consiste en encontrar la parametrización adecuada que permita construir la matriz de pesos y vector de entradas externas:

```
 % Entrenamiento de la red
train(net);  % Cálculo de parámetros de la red
getTrainParam(net)  % Parametrización calculada
```

obteniendo en la ventana de comandos de MATLAB:

```
ans =

  struct with fields:

        A: 2.6910
        B: 3.0000
        C: 1.0000e-05
        D: 1
        K: 0
        N: 10
       Np: 300010
       dL: 0.3090
       dU: 1
    dUaux: 2
      rho: 0.3090
```

Finalmente, se simula la CHN mediante llamada a la función `sim`, obteniendo el resultado de la simulación llamando a `getResults`:

```
% Simulación de la CHN
sim(net); % El algoritmo de simulación por defecto es 'talavan-yanez'

% Visualización de resultados
getResults(net)
```

Así, para el TSP planteado, la solución obtenida mediante la CHN es:

```
ans =

struct with fields:

    CheckpointFilename: ''
              CompTime: 0.022236
                Energy: [1x64 double]
              ExitFlag: 1
         ItersReached: 64
                  Time: [1x64 double]
           TourLength: 6.1803
            ValidPath: 1
           VisitOrder: [6 7 8 9 10 1 2 3 4 5]
```

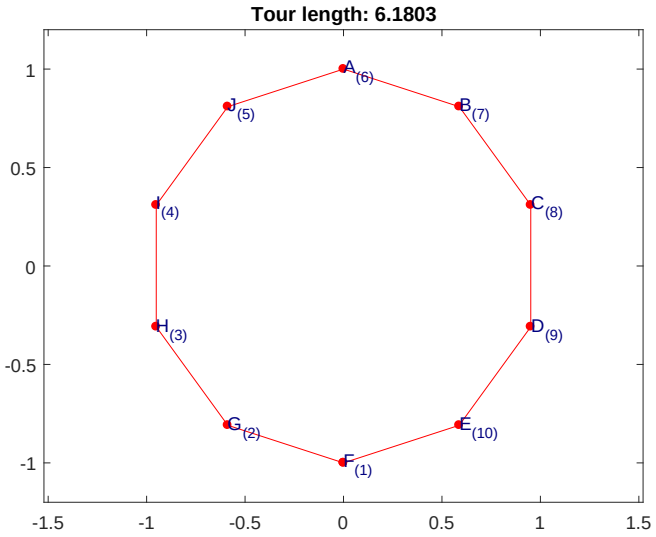
de modo que el tour obtenido es:

```
city(net, getResults(net,'VisitOrder'))
```

```
ans =

  1x10 cell array

    'F'    'G'    'H'    'I'    'J'    'A'    'B'    'C'    'D'    'E'
```

Finalmente, se visualiza el resultado de la simulación de la CHN llamando de nuevo a la función `plot`:

```
19 % Visualización del tour obtenido
20 plot(net)
```



A.4 Funcionalidades avanzadas

A continuación se exponen funcionalidades más avanzadas de la librería *Hopfield Network Toolbox*, que permiten tener un mayor control sobre la simulación de la CHN.

Se presentan la clase `TSPLIB`, para la construcción de problemas de la librería TSPLIB, `HopfieldNetworkGQKPOptions` y `HopfieldNetworkTSPOptions`, para la configuración de opciones del proceso de simulación de la CHN, y las clases `HopfieldNetworkGQKP` y `HopfieldNetworkTSP`, que permiten construir modelos de Hopfield para resolver respectivamente el GQKP y el TSP.

A.4.1 La clase TSPLIB

La clase `TSPLIB` permite construir un TSP para problemas de la librería TSPLIB [38], de modo que facilite el pasar la información relevante de dicho TSP a la CHN para resolverla. Para obtener un listado completo de todos los problemas de TSPLIB en los que se dispone del tour óptimo (de manera que tengamos con qué comparar la calidad de la CHN), bastará con ejecutar:

```
1 problems = utils.TSPLIB.problemNames(true);
```

Se construye a continuación un objeto de la clase `TSPLIB` que haga referencia al problema *BERLIN52*:

```
2 tsp = tsplib({'berlin52'})
```

El objeto `tsp` alberga toda la información relevante del problema como es el nombre, número de ciudades, coordenadas, tipo de distancia y matriz de distancias:

```
tsp =

  TSPLIB with properties:

              Name: 'berlin52'
    NumberOfCities: 52
       Coordinates: [52x2 double]
      DistanceType: 'EUC_2D'
    DistanceMatrix: [52x52 double]
```

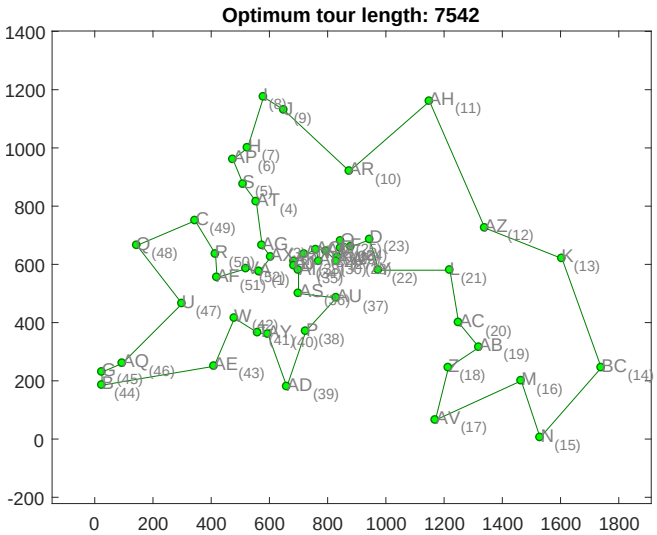
La función `findOptimumTourLength` devuelve la longitud del tour óptimo del problema:

```
3 findOptimumTourLength(tsp)
```

```
ans =

     7542
```

y para visualizar el tour óptimo, se llamará a la función `plot`:

```
4 plot(tsp)
```



A.4.2 Opciones de la CHN para resolver el GQKP

Con el objetivo de resolver el GQKP utilizando la CHN, se crea una clase que permita pasar una estructura de opciones como argumento de entrada a la CHN. Esta clase se llama `HopfieldNetworkOptionsGQKP`. La clase se construye a partir de la función `hopfieldnetOptions`, a la que se le pasan pares de parámetro-valor del siguiente modo:

```
options = hopfieldnetOptions('param1',value1,'param2',value2,...)
```

Los posibles pares de parámetro-valor se detallan en la siguiente tabla (donde, entre paréntesis se indica el valor por omisión):

Parámetro	Valor	Descripción
`'Scheme'`	`'classic'`	Arquitectura o esquema del modelo de Hopfield
`'SimFcn'`	`'euler'` | `'runge-kutta'` | `'talavan-yanez'` (omisión)	Método de simulación de la CHN
`'TrainFcn'`	`'traingty'`	Método de entrenamiento de la CHN para el GQKP
`'CheckpointPath'`	`''` (omisión) | vector de caracteres	Directorio en el que almacenar la trayectoria de la simulación
`'Dt'`	`0.01` (omisión) | valor real positivo	Paso de integración para los métodos de `'euler'` y `'runge-kutta'`
`'E'`	`13` (omisión) | valor escalar entero positivo	Exponente de la tolerancia para el criterio de parada, `1e-E`
`'ExecutionEnvironment'`	`'cpu'` (omisión) | `'gpu'`	Entorno de ejecución de la CHN
`'MaxIter'`	`2000` (omisión) | valor escalar entero positivo	Número máximo de iteraciones
`'Q'`	`0.8` (omisión) | valor real positivo	Disminución del paso de integración en `'talavan-yanez'` durante las primeras `'R_Iter'` iteraciones
`'R_Iter'`	`20` (omisión) | valor escalar entero positivo	Primeras iteraciones de `'talavan-yanez'` en las que se disminuye el paso de integración
`'SimulationPlot'`	`false` (omisión) | `true`	Indicador para visualizar la simulación de la CHN
`'SimulationPlotPauseTime'`	`0.3` (omisión) | valor real positivo	Pausa entre iteraciones en el proceso de visualización de la simulación de la CHN

'TransferFcn'	'satlin' | 'tanh' (omisión)	Función de activación o de transferencia
'U0'	0.3 (omisión) | valor real positivo	Pendiente de la función de transferencia
'Verbose'	false (omisión) | true	Indicador para mostrar información de la simulación de la CHN
'VerboseFrequency'	25 (omisión) | valor escalar entero positivo	Frecuencia de actualización de la información durante la simulación de la CHN

A.4.3 CHN aplicado al GQKP

Para construir una CHN con el objetivo de resolver el GQKP, basta con llamar a la función `hopfieldnet`, pasándole como argumentos de entrada: la matriz $\mathbf{P}$ y el vector $\mathbf{q}$, que se corresponden respectivamente con la parte cuadrática y lineal de la función objetivo (ver ecuación 2.1), las matrices de restricciones $\mathbf{A}$, $\mathbf{A_{eq}}$ y vectores de restricciones, $\mathbf{b}$ y $\mathbf{b_{eq}}$, que se corresponden con las matrices de desigualdades e igualdades de la ecuación 2.2. Adicionalmente, puede también recibir un argumento de entrada opcional que se corresponde con las opciones de simulación recogidas en la sección A.4.2:

```
net = hopfieldnet(P, q, A, b, Aeq, beq, options)
```

A continuación, se resuelve el ejemplo introducido en la sección 2.2:

```
P = [4,0;0,-2];
q = [0;0];
Aeq = [1,1];
beq = [1];
A = [];
b = [];
options = hopfieldnetOptions('SimFcn','talavan-yanez','TransferFcn','tanh');
net = hopfieldnet(P, q, A, b, Aeq, beq, options);
```

Utilizando la parametrización obtenida en la ecuación 2.10, eligiendo $\alpha = 1$ y $\phi_{1,1} = 2$:

```
alpha = 1;
Phi = 2;
eps = 3*alpha/2 + Phi/2;
beta = -alpha/2 - Phi;
Gamma(1,1) = (2*alpha + Phi/2);
Gamma(2,2) = Phi/2 - alpha;
net = train(net,Phi,alpha,beta,eps,Gamma);
```

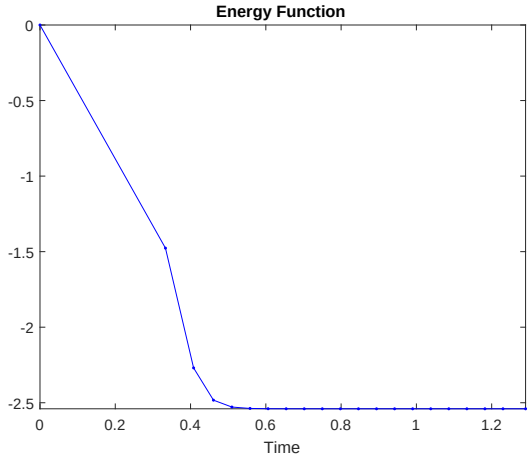
Tomando como punto inicial el punto $\mathbf{v_0} = \left[\begin{smallmatrix}0.6\\0.2\end{smallmatrix}\right]$, o cualquier otro punto (recuérdese que el punto de silla deja todo el Hipercubo de Hamming dentro de la cuenca de atracción del óptimo, ver figura 2.2), se tiene que la solución del problema es:

```
16 V0 = [0.6;0.2];
17 V = sim(net, V0)
```

```
V =

     0
     1
```

Finalmente, se visualiza el descenso de la función de energía:

```
18 energyplot(net)
```



A.4.4 Opciones de la CHN para resolver el TSP

Al igual que se hizo en la sección A.4.2, creando una clase de opciones para configurar la ejecución de la CHN aplicado al GQKP, a continuación se detalla la clase que permite configurar la CHN para resolver el TSP. Esta clase se llama `HopfieldNetworkOptionsTSP`. La clase se construye a partir de la función `tsphopfieldnetOptions`, a la que se le pasan pares de parámetro-valor del siguiente modo:

```
options = tsphopfieldnetOptions('param1',value1,'param2',value2,...)
```

Los posibles pares de parámetro-valor se detallan en la siguiente tabla (donde, entre paréntesis se indica el valor por omisión):

Parámetro	Valor	Descripción
`'Scheme'`	`'classic'` (omisión) | `'divide-conquer'` | `'2opt'` | `'classic&2opt'` | `'divide-conquer&2opt'`	Arquitectura o esquema del modelo de Hopfield

`'SimFcn'`	`'euler'` | `'runge-kutta'` | `'talavan-yanez'` (omisión)	Método de simulación de la CHN
`'TrainFcn'`	`'trainty'`	Método de entrenamiento de la CHN para el TSP
`'K'`	`0` (omisión) | valor escalar entero mayor o igual que 0	Número de cadenas fijadas
`'Coordinates'`	`[]` (omisión) | matriz de números reales con 2 columnas	Coordenadas de las ciudades del TSP
`'DistanceMatrix'`	`[]` (omisión) | matriz cuadrada de números reales	Matriz de distancias del TSP
`'DistanceType'`	`'geo'` | `'euc_2d'` | `'euc'` (omisión) | `'att'` | `'ceil_2d'` | `'explicit'`	Tipo de distancia entre las ciudades del TSP
`'Names'`	`''` (omisión) | cell array de cadenas de caracteres	Nombres de las ciudades del TSP
`'PlotPhases'`	`false` (omisión) | `true`	Indicador para visualizar el resultado de las fases en `'divide-conquer'`
`'Subtours'`	`''` (omisión) | cell array de cadenas de caracteres	Cadenas con las ciudades pre-fijadas estando estas separadas por guiones
`'SubtoursPositions'`	`[]` (omisión) | vector de números enteros con tantas columnas como elementos en `'Subtours'`	Posición que ocupa la primera ciudad de cada una de las cadenas pre-fijadas en `'Subtours'`
`'Tau'`	`[]` (omisión) | valor escalar entero positivo	Número de ciudades vecinas consideradas en la primera fase de `'divide-conquer'`
`'CheckpointPath'`	`''` (omisión) | vector de caracteres	Directorio en el que almacenar la trayectoria de la simulación
`'Dt'`	`0.01` (omisión) | valor real positivo	Paso de integración para los métodos de `'euler'` y `'runge-kutta'`

`'E'`	`13` (omisión) | valor escalar entero positivo	Exponente de la tolerancia para el criterio de parada, `1e-E`
`'ExecutionEnvironment'`	`'cpu'` (omisión) | `'gpu'`	Entorno de ejecución de la CHN
`'MaxIter'`	`2000` (omisión) | valor escalar entero positivo	Número máximo de iteraciones
`'Q'`	`0.8` (omisión) | valor real positivo	Disminución del paso de integración en `'talavan-yanez'` durante las primeras `'R_Iter'` iteraciones
`'R_Iter'`	`20` (omisión) | valor escalar entero positivo	Primeras iteraciones de `'talavan-yanez'` en las que se disminuye el paso de integración
`'SimulationPlot'`	`false` (omisión) | `true`	Indicador para visualizar la simulación de la CHN
`'SimulationPlotPauseTime'`	`0.3` (omisión) | valor real positivo	Pausa entre iteraciones en el proceso de visualización de la simulación de la CHN
`'TransferFcn'`	`'satlin'` | `'tanh'` (omisión)	Función de activación o de transferencia
`'U0'`	`0.3` (omisión) | valor real positivo	Pendiente de la función de transferencia
`'Verbose'`	`false` (omisión) | `true`	Indicador para mostrar información de la simulación de la CHN
`'VerboseFrequency'`	`25` (omisión) | valor escalar entero positivo	Frecuencia de actualización de la información durante la simulación de la CHN

A.4.5 CHN aplicado al TSP

Para construir una CHN con el objetivo de resolver el TSP, basta con llamar a la función `tsphopfieldnet`, que recibirá como argumentos de entrada el número de ciudades del problema, N, el valor del parámetro libre, C y el vector de opciones (introducido en la sección A.4.4) de la clase `HopfieldNetworkOptionsTSP`, con el resto de parámetros necesarios

para construir la CHN.

Se muestran a continuación 3 ejemplos utilizando los modelos `'classic'`, `'divide-conquer'` y `'classic&2opt'`:

Modelo `'classic'` aplicado un problema de TSPLIB

A continuación se resolverá el TSP *BERLIN52*, de la librería TSPLIB, ya presentado en la sección A.4.1. Utilizando las clases `tsplib` y `HopfieldNetworkOptionsTSP` introducidas respectivamente en las secciones A.4.1 y A.4.4, se puede construir el objeto de opciones necesario para configurar la CHN:

```
rng(3); % Para poder reproducir el problema
problem = tsplib({'berlin52'});

 % Estructura de opciones para configurar la CHN
options = tsphopfieldnetOptions('Coordinates', problem.Coordinates, ...
                                'DistanceType', problem.DistanceType, ...
                                'DistanceMatrix', problem.DistanceMatrix);
```

Se crea a continuación el objeto `net` de la clase `HopfieldNetworkTSP` a partir del número de ciudades del problema, N, el valor elegido para el parámetro libre C y la estructura de opciones calculada:

```
N = problem.NumberOfCities;
C = 1e-5;
net = tsphopfieldnet(N,C,options);
```

Para calcular la matriz de pesos y vector de entradas externas, se entrena la red neuronal, entrenamiento consistente en encontrar la parametrización adecuada para el problema:

```
 % Entrenamiento de la red
train(net); % Cálculo de parámetros de la red
getTrainParam(net) % Parametrización calculada
```

obteniendo en la ventana de comandos de MATLAB:

```
ans =

  struct with fields:

         A: 2.9913
         B: 3.0000
         C: 1.0000e-05
         D: 1
         K: 0
         N: 52
        Np: 300052
        dL: 0.0087
        dU: 1
     dUaux: 1716
       rho: 0.0087
```

Se calcula a continuación el punto de silla del problema, pasando, como argumento de entrada a la función `sim` un punto en las cercanías de dicho punto de silla:

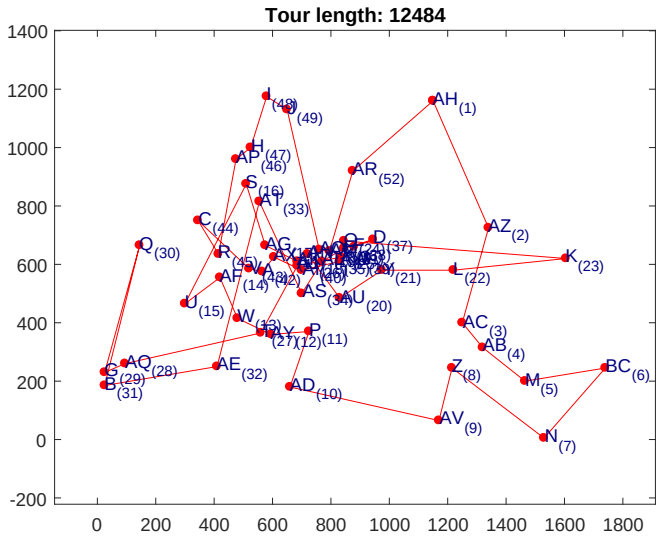
```
14 S = saddle(net);
15 V = S + (rand(N) - 0.5)*1e-10;
16 V = sim(net,V);
```

Pueden recogerse los resultados en la ventana de comandos escribiendo `disp(net)` o `getResults(net)`, obteniendo:

```
ans =

  struct with fields:

    CheckpointFilename: ''
              CompTime: 0.2673
                Energy: [1x352 double]
              ExitFlag: 1
         ItersReached: 352
                  Time: [1x352 double]
            TourLength: 12484
             ValidPath: 1
            VisitOrder: [1x52 double]
```

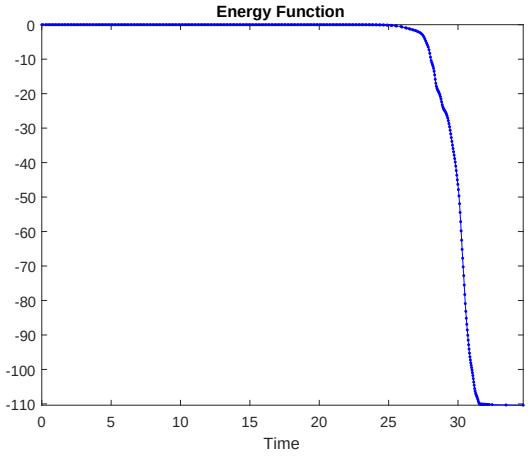
Para visualizar el tour obtenido, se llamará a la función `plot`:

```
17 plot(net)
```



pudiendo visualizar también el descenso de la función de energía:

```
18 energyplot(net)
```



Modelo 'divide-conquer' aplicado un problema de TSPLIB

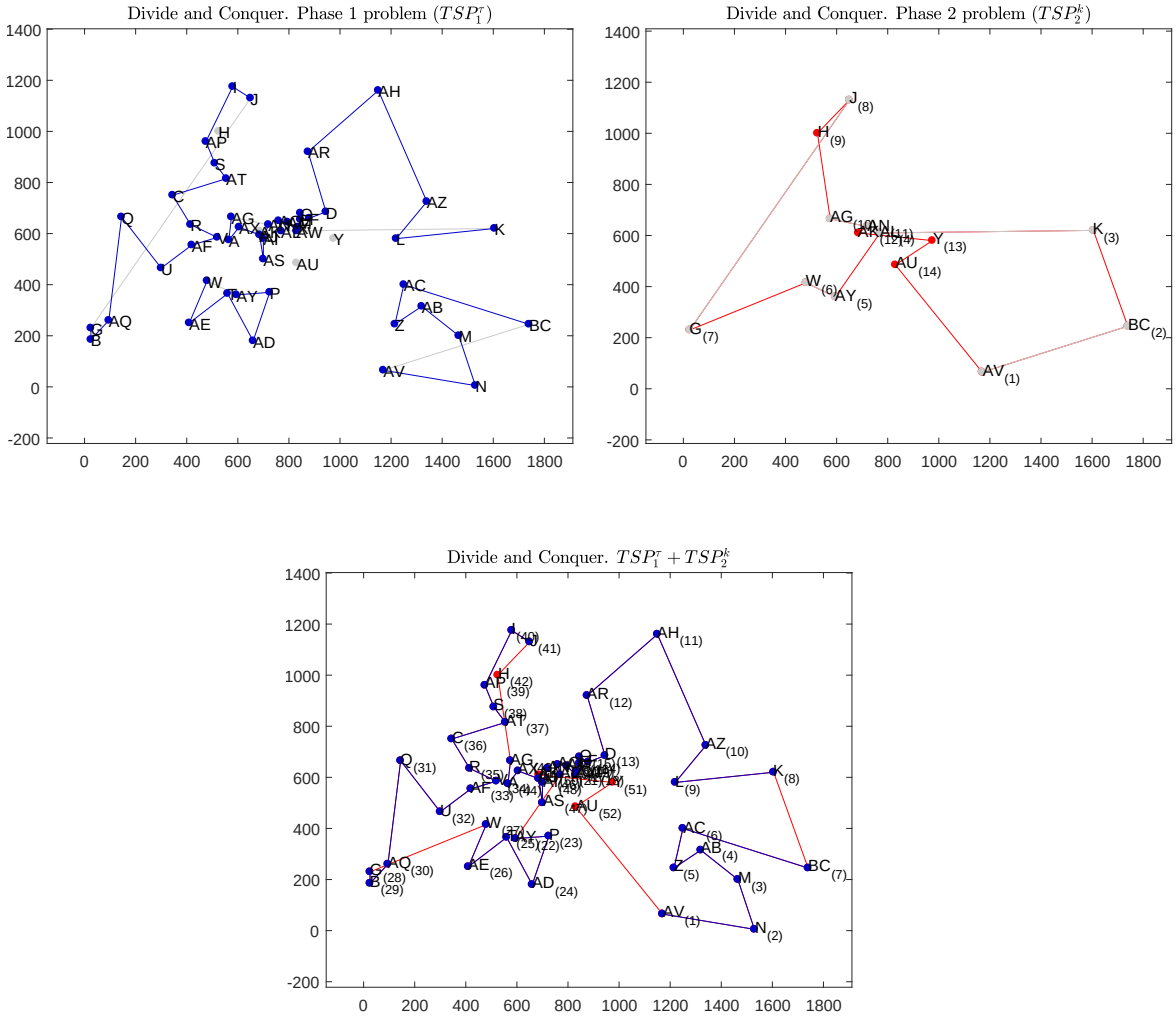
Se resuelve a continuación la misma instancia de TSPLIB mostrada en ejemplo anterior, *BERLIN52*, utilizando el modelo `'divide-conquer'`. La principal diferencia con respecto a lo visto anteriormente está en las opciones necesarias para configurar el modelo. En esta ocasión, el valor del parámetro `'Scheme'` es `'divide-conquer'` (en el ejemplo anterior se utilizaba el valor por omisión, `'classic'`), el valor de `'Tau'`, que ha de tomar un valor entre 1 y el número de ciudades del problema, y el indicador `'PlotPhases'`, que tomando el valor `true` permite obtener la visualización de la simulación tras cada una de las fases.

```
rng(3); % Para poder reproducir el problema
problem = tsplib({'berlin52'});

% Estructura de opciones para configurar la CHN
options = tsphopfieldnetOptions('Scheme', 'divide-conquer', ...
                                'Tau', 5, ...
                                'PlotPhases', true, ...
                                'Coordinates', problem.Coordinates, ...
                                'DistanceType', problem.DistanceType, ...
                                'DistanceMatrix', problem.DistanceMatrix);
```

El resto del código se mantiene igual con respecto a lo visto en el ejemplo anterior:

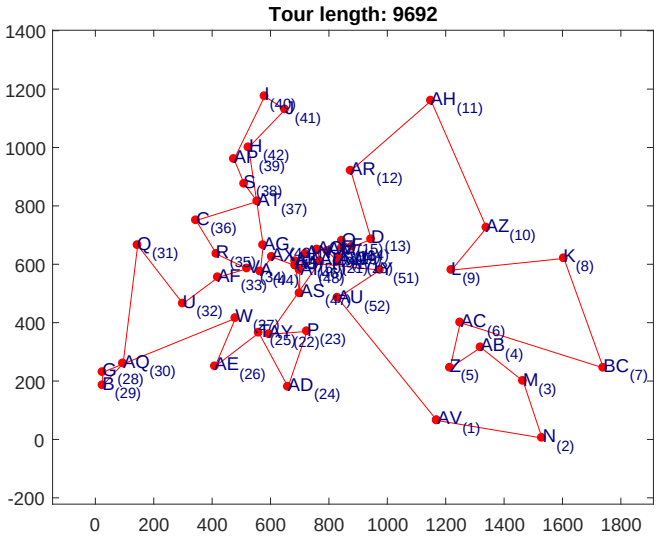
```
N = problem.NumberOfCities;
C = 1e-5;
net = tsphopfieldnet(N,C,options);
% Entrenamiento de la red
train(net); % Cálculo de parámetros de la red
S = saddle(net);
V = S + (rand(N) - 0.5)*1e-10;
V = sim(net,V);
```

obteniéndose, puesto que se ha solicitado la visualización de las distintas fases en la opciones de configuración:



De nuevo, se visualiza el resultado final llamando a la función `plot`:

```
11  plot(net)
```



Modelo `'classic&2opt'` aplicado un problema de TSPLIB

El ejemplo final consiste en intentar de nuevo mejorar la calidad de la solución, pero aplicando en esta ocasión *intercambios 2-opt* sobre la solución que produce el modelo `'classic'`, utilizando los resultados del capítulo 4.

Al igual que en el ejemplo anterior, debemos cambiar las opciones de configuración de la CHN, donde en este caso, en único cambio sustancial con respecto al modelo `'classic'` es la utilización en su lugar del modelo `'classic&2opt'`:

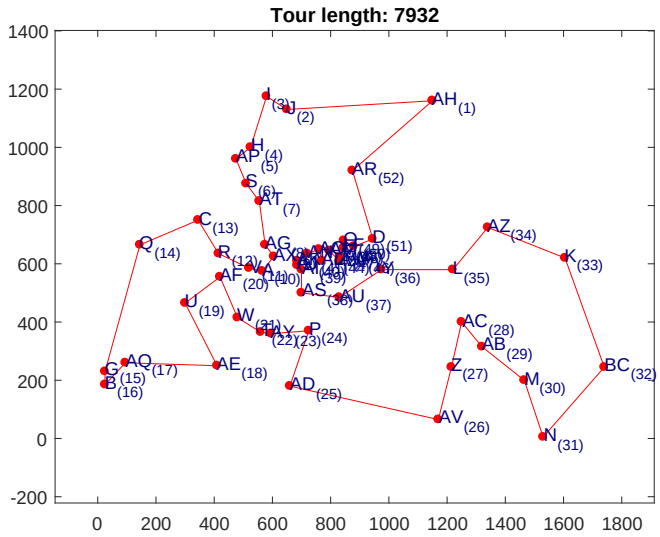
```
rng(3);  % Para poder reproducir el problema
problem = tsplib({'berlin52'});

 % Estructura de opciones para configurar la CHN
options = tsphopfieldnetOptions('Scheme', 'classic&2opt', ...
                                'Coordinates', problem.Coordinates, ...
                                'DistanceType', problem.DistanceType, ...
                                'DistanceMatrix', problem.DistanceMatrix);
```

Una vez más, el resto del código es exactamente igual:

```
N = problem.NumberOfCities;
C = 1e-5;
net = tsphopfieldnet(N,C,options);
 % Entrenamiento de la red
train(net);  % Cálculo de parámetros de la red
S = saddle(net);
V = S + (rand(N) - 0.5)*1e-10;
V = sim(net,V);
```

concluyendo el ejemplo visualizando el tour obtenido:

```
plot(net)
```



El tour obtenido mejora notablemente los resultados del modelo `'classic'` (12848) y `'divide-conquer'` (9692).

Referencias

[1] Abe, S. Global convergence and suppression of spurious states of the Hopfield neural networks. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 40*, 4 (1993), 246–257.

[2] Aiyer, S. V., Niranjan, M., and Fallside, F. A theoretical investigation into the performance of the Hopfield model. *IEEE Transactions on Neural Networks 1*, 2 (1990), 204–215.

[3] Anderson, J. A. A simple neural network generating an interactive memory. *Mathematical biosciences 14*, 3-4 (1972), 197–220.

[4] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[5] Croes, G. A. A method for solving traveling-salesman problems. *Operations research 6*, 6 (1958), 791–812.

[6] Cuykendall, R., and Reese, R. Scaling the neural TSP algorithm. *Biological Cybernetics 60*, 5 (1989), 365–371.

[7] De Mazancourt, T., and Gerlic, D. The inverse of a block-circulant matrix. *IEEE transactions on antennas and propagation 31*, 5 (1983), 808–810.

[8] Di Marco, M., Forti, M., Grazzini, M., and Pancioni, L. Necessary and sufficient condition for multistability of neural networks evolving on a closed hypercube. *Neural Networks 54* (2014), 38–48.

[9] García, L., Talaván, P. M., and Yáñez, J. Attractor basin analysis of the Hopfield model: The Generalized Quadratic Knapsack Problem. In *International Work-Conference on Artificial Neural Networks* (2017), Springer, pp. 420–431.

[10] García, L., Talaván, P. M., and Yáñez, J. Improving the Hopfield model performance when applied to the traveling salesman problem. *Soft Computing 21*, 14 (2017), 3891–3905.

[11] Gopal, M. *Modern control system theory*. New Age International, 1993.

[12] Grossberg, S. Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. *Biological cybernetics 23*, 3 (1976), 121–134.

[13] Hagan, M. T., Demuth, H. B., Beale, M. H., and De Jesús, O. *Neural network design*, vol. 20. PWS publishing company Boston, 1996.

[14] Hebb, D. O. *The organization of behavior: A neuropsychological theory*. John Wiley & Sons Inc., 1949.

[15] Hedge, S. U., Sweet, J. L., and Levy, W. B. Determination of parameters in a Hopfield/Tank computational network. In *Neural Networks, 1988., IEEE International Conference on* (1988), IEEE, pp. 291–298.

[16] Hernández-Solano, Y., Atencia, M., Joya, G., and Sandoval, F. A discrete gradient method to enhance the numerical behaviour of Hopfield networks. *Neurocomputing 164* (2015), 45–55.

[17] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences 79*, 8 (1982), 2554–2558.

[18] Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences 81*, 10 (1984), 3088–3092.

[19] Hopfield, J. J., and Tank, D. W. "Neural" computation of decisions in optimization problems. *Biological cybernetics 52*, 3 (1985), 141–152.

[20] Jolai, F., and Ghanbari, A. Integrating data transformation techniques with Hopfield neural networks for solving travelling salesman problem. *Expert Systems with Applications 37*, 7 (2010), 5331–5335.

[21] Joya, G., Atencia, M., and Sandoval, F. Hopfield neural networks for optimization: study of the different dynamics. *Neurocomputing 43*, 1-4 (2002), 219–237.

[22] Kamgar-Parsi, B., and Kamgar-Parsi, B. On problem solving with Hopfield neural networks. *Biological Cybernetics 62*, 5 (1990), 415–423.

[23] Kohonen, T. Correlation matrix memories. *IEEE transactions on computers 100*, 4 (1972), 353–359.

[24] Kohonen, T. Self-organized formation of topologically correct feature maps. *Biological cybernetics 43*, 1 (1982), 59–69.

[25] Koopmans, T. C., and Beckmann, M. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society* (1957), 53–76.

[26] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[27] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[28] Lin, S., and Kernighan, B. W. An effective heuristic algorithm for the traveling-salesman problem. *Operations research 21*, 2 (1973), 498–516.

[29] McCulloch, W. S., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*, 4 (1943), 115–133.

[30] Mérida-Casermeiro, E., Galán-Marín, G., and Munoz-Perez, J. An efficient multivalued Hopfield network for the traveling salesman problem. *Neural Processing Letters 14*, 3 (2001), 203–216.

[31] Minsky, M., and Papert, S. *Perceptrons*. Oxford, England: MIT Press, 1969.

[32] Papadimitriou, C. H. The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science 4*, 3 (1977), 237–244.

[33] Park, S. Signal space interpretations of Hopfield neural network for optimization. In *IEEE International Symposium on Circuits and Systems, 1989.* (1989), IEEE, pp. 2181–2184.

[34] Pavlov, I. P., and ANREP, G. V. *Conditioned Reflexes. An Investigation of the Physiological Activity of the Cerebral Cortex. Translated and Edited by GV Anrep*. London, 1927.

[35] Platt, J. C., and Barr, A. H. Constrained differential optimization for neural networks.

[36] Qin, K. On chaotic neural network design: A new framework. *Neural Processing Letters* (2016), 1–19.

[37] Ramón y Cajal, S. *Sur la structure de l'écorce cérébrale de quelques mammifères*. Typ. de Joseph van In & Cie.; Aug. Peeters, lib, 1891.

[38] Reinelt, G. TSPLIB. A traveling salesman problem library. *ORSA journal on computing 3*, 4 (1991), 376–384.

[39] Rojas, R. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[40] Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review 65*, 6 (1958), 386.

[41] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Cognitive modeling 5*, 3 (1988), 1.

[42] Rumelhart, D. E., McClelland, J. L., Group, P. R., et al. Parallel distributed processing: Explorations in the microstructures of cognition. volume 1: Foundations, 1986.

[43] Salcedo-Sanz, S., Ortiz-García, E. G., Pérez-Bellido, Á. M., Portilla-Figueras, A., and López-Ferreras, F. On the performance of the LP-guided Hopfield network-genetic algorithm. *Computers & Operations Research 36*, 7 (2009), 2210–2216.

[44] Serpen, G. *Adaptive Hopfield Network*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 3–10.

[45] Shepherd, G. M. *The synaptic organization of the brain*. Oxford University Press, 2003.

[46] Suh, T., and Esat, I. I. Solving large scale combinatorial optimisation problems based on a divide and conquer strategy. *Neural Computing & Applications 7*, 2 (1998), 166–179.

[47] Sun, Y., Wang, Z., and Van Wyk, B. J. Chaotic Hopfield neural network swarm optimization and its application. *Journal of Applied Mathematics 2013* (2013).

[48] Talaván, P. M. *El modelo de Hopfield aplicado a problemas de optimización combinatoria.* PhD thesis, Universidad Complutense de Madrid, 2003.

[49] Talaván, P. M., and Yáñez, J. Parameter setting of the Hopfield network applied to TSP. *Neural Networks 15*, 3 (2002), 363–373.

[50] Talaván, P. M., and Yáñez, J. A continuous Hopfield network equilibrium points algorithm. *Computers & operations research 32*, 8 (2005), 2179–2196.

[51] Talaván, P. M., and Yáñez, J. The Generalized Quadratick Knapsack Problem. A neuronal network approach. *Neural networks 19*, 4 (2006), 416–428.

[52] Tan, K. C., Tang, H., and Ge, S. S. On parameter settings of Hopfield networks applied to traveling salesman problems. *IEEE Transactions on Circuits and Systems I: Regular Papers 52*, 5 (2005), 994–1002.

[53] Tang, H., Tan, K. C., and Yi, Z. A columnar competitive model for solving combinatorial optimization problems. *IEEE Transactions on Neural Networks 15*, 6 (2004), 1568–1574.

[54] Wang, R. L., Tang, Z., and Cao, Q. P. A learning method in Hopfield neural network for combinatorial optimization problem. *Neurocomputing 48*, 1 (2002), 1021–1024.

[55] Wasserman, P. D., and Meyer-Arendt, J. R. Neural computing, theory and practice. *Applied Optics 29* (1990), 2503.

[56] Wen, U.-P., Lan, K.-M., and Shih, H.-S. A review of Hopfield neural networks for solving mathematical programming problems. *European Journal of Operational Research 198*, 3 (2009), 675–687.

[57] Widrow, B., Hoff, M. E., et al. Adaptive switching circuits. In *IRE WESCON convention record* (1960), vol. 4, New York, pp. 96–104.

[58] Wilson, G., and Pawley, G. On the stability of the travelling salesman problem algorithm of Hopfield and Tank. *Biological Cybernetics 58*, 1 (1988), 63–70.

[59] Woeginger, G. J. Exact algorithms for NP-hard problems: A survey. In *Combinatorial Optimization—Eureka, You Shrink!* Springer, 2003, pp. 185–207.

Índice alfabético