UNIVERSIDAD DE DEUSTO
FACULTAD DE INFORMÁTICA

TESIS DOCTORAL

ODE: UN NUEVO MODELO NEURONAL ESTOCÁSTICO PARA OPTIMIZACIÓN COMBINATORIA

José Fernando Díaz Martín

1996

UNIVERSIDAD DE DEUSTO
FACULTAD DE INFORMÁTICA

ODE: UN NUEVO MODELO NEURONAL ESTOCÁSTICO PARA OPTIMIZACIÓN COMBINATORIA

Tesis doctoral presentada por D. **José Fernando Díaz Martín**

Dirigida por el Dr. **Verónica Canivell Castillo**

El Director

El Doctorando

Bilbao, a 23 de Mayo de 1996

A mis padres.

Agradecimientos

Las ideas contenidas en esta tesis son el resultado de una gran dedicación personal a la investigación en el campo de las redes neuronales artificiales y la optimización combinatoria, investigación que no hubiera sido posible sin la colaboración y el apoyo que me han prestado muchas personas durante estos últimos cinco años. Principalmente, quiero agradecer a mi directora de tesis, Verónica Canivell Castillo, su labor de orientación sobre el tema para la tesis doctoral, así como el seguimiento realizado durante su desarrollo, junto a los valiosos consejos y comentarios aportados sobre la estructura y formalización de la misma.

Agradezco a todos mis compañeros del departamento de IA del centro de investigación LABEIN, y especialmente a los componentes de mi equipo de trabajo, José Mª Lázaro y José Maseda, por su decisiva influencia en mi interés por el mundo de la investigación en la disciplina de la Inteligencia Artificial, durante los años en que permanecí en dicho centro con una beca de investigación concedida por el Colegio de Ingenieros Industriales.

También quiero agradecer a Ulises Cortés el haberme proporcionado en un curso de doctorado sobre sistemas inteligentes una importante referencia sobre el área de redes neuronales artificiales y optimización combinatoria, la cual resultó decisiva en mi posterior elección del tema de tesis doctoral.

Deseo agradecer a todos mis compañeros profesores del departamento de Lenguajes y Sistemas Informáticos, especialmente a Charo de Godos, así como al becario colaborador Mario Sanz, su apoyo, sugerencias y consejos durante la elaboración de la tesis.

Por último, agradezco a mis amigos Alfredo Sanz y Abel Vélez, su generosidad y ayuda durante los momentos difíciles habidos en estos años de trabajo. Así mismo, debo agradecer a mi amigo Gerardo Valbuena su apoyo y ánimo, los cuales me ayudaron a realizar con mayor interés si cabe la investigación de la tesis.

El desarrollo de esta tesis ha sido posible gracias a la financiación concedida por el Gobierno Vasco - Eusko Jaurlaritza a través de la beca predoctoral para formación de investigadores BFI92.039.

ÍNDICE GENERAL

ABSTRACT

This thesis is concerned with the area of Artificial Neural Networks and Combinatorial Optimization, and its objective is the development of optimization techniques based on neural models, capable of solving real size problems, by achieving high quality solutions in feasible execution time. Following a common approach in the development of a connectionist system, a theoretical model suitable for combinatorial optimization problems is defined, analyzing its structure with respect to its dynamical behaviour (recall and stability). Three classic problems of combinatorial optimization are used in the practical analysis of the proposed model: the travelling salesman problem, TSP, the graph partition problem, GPP, and the quadratic assignment problem, QAP, evaluating the system with respect to temporal complexity, so as to the quality of the solutions. Finally, a practical application, developed on an important area of interest in discrete manufacturing, is described: NEUROPROG, a predictive job-shop scheduler.

Keywords: Artificial Neural Networks, Combinatorial Optimization, Resource Allocation, Job-shop Scheduling.

RESUMEN

Esta tesis se enmarca dentro del área de Redes Neuronales Artificiales y Optimización Combinatoria, y tiene como objetivo el desarrollo de nuevas técnicas de optimización basadas en modelos neuronales, que sean capaces de resolver problemas prácticos de dimensión real, obteniendo soluciones de alta calidad en tiempos de ejecución factibles. Siguiendo la aproximación general para la construcción de un sistema conexionista, se define un modelo teórico general apropiado para problemas de optimización combinatoria, analizándose tanto su arquitectura o estructura estática como su comportamiento dinámico (modo de operación y estabilidad). El estudio práctico del modelo propuesto se realiza sobre tres problemas clásicos de optimización combinatoria: el problema del viajante comercial, PVC, el problema de la partición de grafos, PPG, y el problema de la asignación cuadrática, PAC, evaluando el sistema con respecto a su complejidad temporal y a la calidad de las soluciones obtenidas. Finalmente, se hace una descripción de la aplicación práctica que se ha desarrollado sobre una importante área de interés en el entorno de fabricación industrial: NEUROPROG, un programador predictivo para sistemas de producción discreta.

Palabras clave: Redes Neuronales Artificiales, Optimización Combinatoria, Asignación de recursos, Programación de Producción Discreta.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1

Introducción

Esta introducción está dividida en cuatro apartados: un planteamiento general del problema, una breve descripción del método propuesto para su resolución, las principales aportaciones realizadas y un esquema de los restantes capítulos de la tesis.

1.1 PLANTEAMIENTO GENERAL DEL PROBLEMA.

Desde la antigüedad, los seres humanos han tratado de duplicar, de una forma u otra, el comportamiento inteligente. Este interés se ha incrementado especialmente desde la aparición de los primeros computadores hace algunas décadas, dando origen a una nueva ciencia denominada Inteligencia Artificial, IA, cuya finalidad principal es encontrar modelos de proceso capaces de emular las actividades inteligentes de los seres humanos.

Los computadores son máquinas capaces de realizar tareas que resultan imposibles para las personas: pueden almacenar sin ningún esfuerzo cantidades enormes de datos, y pueden realizar millones de operaciones aritméticas sin cometer un solo error en tiempos increíblemente pequeños. Por esta razón, los computadores son especialmente adecuados para multitud de aplicaciones tales como mantenimiento de bases de datos, resolución de problemas de cálculo matemático, comunicaciones electrónicas, procesamiento de textos y gráficos, etc.

Sin embargo, los seres humanos realizan rutinariamente otras tareas de características diferentes como, por ejemplo, caminar, percibir objetos sobre escenas naturales, hablar, y razonar con sentido común, las cuales son imposibles de implementar con igual eficacia en los computadores actuales. Así mismo, las personas son capaces de aprender a hacer todas estas cosas de forma cada vez más precisa y efectiva a través de la experiencia.

Todas estas tareas, aparentemente simples, tienen una importante característica en común: la existencia de múltiples fuentes de información y restricciones actuando simultáneamente durante su desarrollo. Por ejemplo, realizar una acción motora, como capturar con la mano un objeto en movimiento, implica tratar un gran número de restricciones que interaccionan a la vez de forma dinámica: la posición y velocidad del objeto, su forma, tamaño, peso, la postura de la mano, los obstáculos que pudiera haber en la trayectoria, etc.

La implementación de estas tareas cognitivas y motoras propias de los seres humanos y de algunos animales no es un problema irresoluble en la mayoría de los casos, sino que su dificultad está en la utilización de computadores secuenciales, los cuales no parecen apropiados para resolver este tipo de problemas. Así, si se dispone únicamente de computadores secuenciales es lógico que se intenten resolver todos los problemas mediante algoritmos secuenciales, con el inconveniente de que en determinados casos se inviertan grandes esfuerzos en el desarrollo de sofisticados algoritmos, incluso sin que se llegue a alcanzar una solución admisible.

Inicialmente, dentro de la Inteligencia Artificial hubo dos corrientes de trabajo claramente diferenciadas. La primera de ellas, denominada comúnmente IA simbólica, estaba dirigida a modelizar los procesos cognitivos de la mente humana desde una perspectiva funcional o de alto nivel, empleando fundamentalmente para ello algoritmos secuenciales. La segunda estaba encaminada a emular la inteligencia de forma más directa, construyendo máquinas basadas en los sistemas nerviosos biológicos: las *redes neuronales artificiales, RNA*.

En un principio, las redes neuronales artificiales crearon grandes expectativas entre los científicos e investigadores, por lo que fueron estudiadas activamente. No obstante, la investigación en redes neuronales sufrió desde finales de los años sesenta hasta los ochenta un periodo de vacío en el que apenas se consiguieron avances notables. Esta situación fue debida fundamentalmente a la falta de arquitecturas neuronales y algoritmos de aprendizaje eficientes, junto a consideraciones negativas con respecto a este área por parte de la comunidad científica de la IA simbólica, entonces predominante.

Posteriormente, en especial a partir de la segunda mitad de la década de los ochenta, se produjo un gran incremento en el interés generado por las redes neuronales artificiales. Este nuevo auge fue debido a varias razones, entre las que pueden incluirse la aparición de computadores digitales más rápidos en los que simular grandes redes, el interés por

contruir computadores con procesamiento masivamente paralelo, y, sobre todo, el descubrimiento de nuevas arquitecturas de redes neuronales y algoritmos de aprendizaje más eficientes.

Las redes neuronales artificiales pueden verse como sistemas computacionales masivamente paralelos de control distribuido, cuyos principios teóricos se han inspirado en los conocimientos actuales de la estructura y funcionamiento de los sistemas nerviosos biológicos. Su funcionalidad principal es la de reconocer y clasificar conjuntos de patrones, con la característica excepcional de ser capaces de *aprender* a realizar dicha tarea sin una programación previa. Esta capacidad, inherente a la mayoría de los modelos neuronales desarrollados, permite su aplicación a actividades para las que no existen reglas o modelos matemáticos precisos, algunas de las cuales han sido ampliamente estudiadas en la Inteligencia Artificial simbólica, como el reconocimiento de patrones, la visión artificial, el procesamiento de voz, el control de acciones motoras, el procesamiento de lenguaje natural, etc. Así mismo, las redes neuronales artificiales pueden aplicarse a otras áreas igualmente estudiadas en la Inteligencia Artificial clásica, en las que interesa más la *arquitectura masivamente paralela* y el *modo de procesamiento distribuido* de las redes, que su potencial de aprendizaje. Dentro de esta clase de aplicaciones, denominadas generalmente aplicaciones de búsqueda heurística, una de las más importantes es la resolución de problemas de optimización combinatoria.

La *optimización combinatoria* tiene por objeto el estudio y resolución de problemas de optimización con restricciones en los que las variables constituyentes son de tipo discreto. Los problemas de optimización combinatoria se clasifican de acuerdo con el tiempo de ejecución que es necesario emplear para su resolución óptima, siendo de especial relevancia aquellos que precisan algoritmos de complejidad temporal superpotencial (exponencial, factorial o potencial-exponencial) con respecto a la dimensión del problema, denominados también problemas NP-completos. Este tipo de problemas ha sido objeto de extenso estudio en las áreas de Matemática Discreta, y de Computación e Inteligencia Artificial, especialmente a partir del desarrollo de la teoría de la NP-completitud en los años setenta [GARE79].

Dada su sencillez y su importancia desde un punto de vista teórico, el ejemplo más característico de problema de optimización combinatoria es el problema del viajante comercial, en el que un viajante debe recorrer cíclicamente N ciudades, visitando una sola vez cada ciudad, de forma que se minimice la distancia total recorrida. Todos los métodos exactos conocidos para determinar la solución de este problema requieren un

tiempo de ejecución que crece exponencialmente con respecto al número de ciudades, por lo que se trata de un problema NP-completo.

Una clase de problemas de optimización combinatoria, muy importante por su carácter eminentemente práctico, es la formada por los problemas NP-completos de *programación* o *asignación de recursos*. Pueden definirse brevemente de la siguiente forma: Se debe realizar una serie de actividades o tareas, y se dispone para ello de un conjunto de recursos materiales y/o humanos. El problema consiste en encontrar un programa que asigne a cada tarea los recursos necesarios, de forma que se minimice una función de coste determinada, y se satisfagan las restricciones que puedan existir entre los diferentes elementos del sistema. Entre los problemas clásicos de asignación de recursos pueden citarse la programación de tareas de producción, el problema del transporte, la programación de turnos de trabajo, y la planificación de proyectos.

El principal esfuerzo de los investigadores en problemas de optimización, y especialmente en problemas de asignación de recursos, se centra actualmente en encontrar un método general con el que puedan tratarse problemas de dimensión y características reales, y que al mismo tiempo sea eficiente con respecto al tiempo de ejecución y a la calidad de la solución obtenida.

Las técnicas generales de resolución desarrolladas pertenecen a tres campos de investigación diferentes. En primer lugar, pueden mencionarse los *algoritmos de búsqueda heurística*, tales como los algoritmos deterministas de búsqueda local, y el algoritmo estocástico del temple simulado (simulated annealing)[1]. Un segundo tipo de técnicas de resolución son los algoritmos de búsqueda basados en la evolución natural, denominados *algoritmos genéticos*. Finalmente, a raíz del descubrimiento de las propiedades de computación colectiva que poseen algunos modelos conexionistas recurrentes, el área de las *redes neuronales artificiales* ha aportado un tercer enfoque general a la resolución de problemas de optimización combinatoria.

El inicio de la investigación en el campo de las redes neuronales y la optimización combinatoria fue realizado por John Hopfield con el estudio del comportamiento

[1]El término inglés *annealing* se traduce literalmente como *recocido*, proceso térmico utilizado en la industria metalúrgica, que se caracteriza por un calentamiento del metal a temperatura elevada (próxima a la fusión), seguido de un enfriamiento lento, con el objeto de disminuir la fragilidad de la masa metálica. En la redacción de la tesis se ha elegido el término *temple* o *templado* por ser el utilizado habitualmente en las traducciones y en la literatura en lengua española sobre el tema [RUME92], [FREE95], aunque en realidad éste corresponde al proceso térmico opuesto, caracterizado por el enfriamiento brusco del metal.

dinámico de un modelo neuronal denominado *red de Hopfield continua* [HOPF84], resultado de la evolución de un modelo previo, el *autocorrelador discreto* o *red de Hopfield discreta* [HOPF82]. El campo se amplió más tarde con otros modelos entre los que destacan la *máquina de Boltzmann* [HINT86], la *máquina de Cauchy* [SZU87], la *máquina de Gauss* [AKIY89] y la *red de temple determinista* [BOUT89].

El fundamento de la aplicación de las redes neuronales artificiales al campo de la optimización combinatoria está en el paradigma de minimización de energía. Este paradigma interpreta a las redes neuronales como sistemas termodinámicos cuyos estados de energía mínima representan soluciones a determinados problemas de optimización. Bajo este punto de vista, no interesa el potencial de aprendizaje de los modelos conexionistas, sino su arquitectura masivamente paralela y su modo de procesamiento recurrente. El objetivo consiste en definir un esquema de interconexión y un modo de procesamiento de la red que permita resolver el problema tratado a través de un proceso denominado *relajación paralela*, en el que el estado de la red se desplaza hacia un estado estable de energía mínima.

La aproximación conexionista a la optimización combinatoria es, en principio, muy interesante, porque las redes neuronales artificiales son intrínsecamente paralelizables, y podrían, por esta razón, usarse con gran ventaja en problemas de dimensión real. Ahora bien, los modelos conexionistas actuales, basados en la red de Hopfield continua y en la máquina de Boltzmann, han resultado insuficientes para la resolución de problemas de optimización [GEE95], debido fundamentalmente al carácter local del procesamiento utilizado en la implementación de la relajación paralela. Sin embargo, las importantes características, tanto estructurales como dinámicas, de las redes neuronales recurrentes bastan para motivar la investigación y desarrollo de nuevos modelos conexionistas que superen las limitaciones de los anteriores y exploten eficientemente el paralelismo masivo propio de las RNA en la resolución de problemas de optimización combinatoria.

Por ello, el objetivo principal de esta tesis, enmarcada dentro del área de las redes neuronales y la optimización combinatoria, es el desarrollo de un modelo neuronal estocástico que sea capaz de resolver problemas generales de optimización combinatoria de dimensión real, obteniendo soluciones de alta calidad en tiempos de ejecución razonables, mejorando las prestaciones de los métodos actuales de optimización tanto heurísticos como conexionistas.

El estudio y análisis del modelo neuronal propuesto se realiza sobre tres problemas clásicos de optimización combinatoria: el problema del viajante comercial, el problema de la partición de grafos, y el problema de la asignación cuadrática, evaluando estadísticamente el sistema con respecto a su complejidad temporal, así como a la calidad de las soluciones obtenidas.

Adicionalmente, como ejemplo práctico sobre el estudio realizado, esta tesis plantea el desarrollo de una aplicación basada en el modelo neuronal propuesto. Para ello se ha elegido un problema de asignación de recursos enmarcado en una importante área de interés en la industria de fabricación: *la programación de producción discreta.*

1.2 DESCRIPCIÓN GENERAL DEL MODELO CONEXIONISTA DESARROLLADO.

El modelo neuronal desarrollado en esta tesis, denominado *optimizador discreto estocástico, ODE*, es una red recurrente no simétrica, resultado de la generalización de la máquina de Boltzmann [HINT86], que posee estados de activación con valores discretos, e incorpora un modo de procesamiento recurrente síncrono basado en la técnica del temple simulado [KIRK83]. La representación de un problema de optimización sobre el modelo ODE se realiza siguiendo un método general consistente en fijar los diferentes parámetros de la red de forma que su función de energía represente correctamente los objetivos y restricciones del problema. A continuación se describen brevemente los elementos y características más importantes de este modelo.

El optimizador discreto estocástico es una red recurrente con 2N + 1 nodos, donde N es el tamaño de los patrones a procesar. Los nodos se agrupan en dos capas, una visible con N nodos principales más N nodos auxiliares, y otra oculta formada por un solo nodo. Utiliza un modo de procesamiento recurrente síncrono en el que una entrada se procesa hasta que la red alcanza un estado estable que minimiza la función de energía del sistema. Al comenzar el procesamiento de la red, tanto los N nodos principales como los N nodos auxiliares de la capa visible se inicializan con los N valores del patrón de entrada. A continuación, esta entrada se procesa síncronamente de acuerdo con el siguiente criterio. Se actualizan al azar K nodos auxiliares de la capa visible y se calcula el estado de activación del nodo de la capa oculta, utilizando la regla de combinación aditiva-multiplicativa de segundo orden y una función de umbral estocástica. Si el nuevo valor

del nodo de la capa oculta es 1, se acepta la transición producida y se copian los estados de los nodos auxiliares sobre los nodos principales de la capa visible. En caso contrario, se rechaza la transición y se refrescan los nodos auxiliares con los estados de los nodos principales. Esta operación se repite sucesivamente hasta que la red alcance un estado en el que cualquier nuevo cálculo no implique un cambio en los valores de activación de los nodos principales de la capa visible.

La característica principal de este modo de procesamiento es que implementa una técnica de minimización basada en el algoritmo del temple simulado. Así, en el optimizador discreto estocástico la actualización del estado del nodo de la capa oculta no es determinista con respecto a las entradas ponderadas del mismo, sino que, de forma similar a otras redes como la máquina de Boltzmann o la máquina de Gauss, utiliza una regla de aleatorización. Esta forma de procesamiento permite aceptar en determinados casos un cambio que genera un incremento parcial de energía en el estado actual de la red, con el fin de escapar de los mínimos locales y obtener una mejor solución final.

La forma en que la red ODE se aplica a la optimización combinatoria es similar a la de otros modelos conexionistas. El proceso general se describe a continuación:

- En primer lugar, se debe establecer una disposición arquitectónica de la red que identifique de forma específica los nodos con las variables del problema de optimización.
- Después, se debe transformar la función objetivo y las restricciones del problema en una función de energía, de manera que al minimizarse esta última se encuentre la solución del problema de optimización inicial.
- Por último, hay que determinar los parámetros asociados a la función de energía, para así obtener los valores de los pesos de las conexiones de cada nodo. Estos valores deberán representar adecuadamente la instancia del problema específico a resolver.

De esta manera, a partir de un patrón de entrada que representa la situación inicial del sistema a optimizar, y a través del proceso de minimización de energía que caracteriza al modelo, la red construida genera una respuesta cercana a la solución óptima del problema.

1.3 PRINCIPALES APORTACIONES.

Las principales aportaciones del trabajo realizado al área de las redes neuronales artificiales y la optimización combinatoria se resumen a continuación.

- *Definición de un modelo conexionista general para resolución de problemas de optimización combinatoria que mejora la eficiencia de los métodos actuales.* Al igual que otros modelos de red, como la máquina de Gauss [AKIY89] o la red de temple determinista [BOUT89], el optimizador discreto estocástico presenta un modo de procesamiento recurrente que intenta combinar las ventajas de varios modelos conexionistas utilizados en la resolución de problemas de optimización combinatoria. Por una parte, la operación en tiempo discreto y la utilización de un criterio estocástico de transición de estados, similares a los de la máquina de Boltzmann [HINT86], permiten superar el comportamiento de optimización local de las redes deterministas basadas en la técnica de la escalada (hill-climbing). Por otra parte, el procesamiento recurrente específico del modelo ODE evita la limitación intrínseca del procesamiento de la máquina de Boltzmann, caracterizado por la actualización de un solo nodo cada vez, con lo que la búsqueda se realiza sobre un espacio de soluciones mayor. Finalmente, el sincronismo del procesamiento, propio de la red de Hopfield continua [HOPF84], permite la paralelización inmediata del modelo.

- *Desarrollo de un método general de representación de problemas de optimización combinatoria sobre redes neuronales recurrentes.* La determinación de los parámetros asociados a la función de energía del sistema es uno de los mayores inconvenientes actuales de la implementación de problemas de optimización sobre redes recurrentes, y, generalmente, se realiza de forma empírica para cada instancia de un determinado problema. Estos parámetros pueden dividirse en cuatro clases: parámetros destinados a obtener K nodos activos de N en la estabilización del sistema, parámetros destinados a implementar las restricciones estructurales de la red, parámetros destinados a implementar las restricciones del problema, y parámetros destinados a implementar los objetivos de la optimización. Tras un estudio del comportamiento dinámico de los diferentes modelos conexionistas aplicados a problemas de optimización, se proporciona un procedimiento de diseño para la determinación exacta de los pesos de las conexiones y de la tendencia de cada nodo, ante cualquier instancia de un problema de optimización combinatoria.

- *Análisis comparativo de la eficiencia de los diferentes métodos de optimización combinatoria actuales con respecto al modelo conexionista propuesto.* Para obtener una evaluación correcta de la eficiencia de un método de resolución de problemas de optimización combinatoria es necesario, en primer lugar, establecer un criterio estadístico de evaluación de la eficiencia, tanto en la calidad de la solución obtenida como en el tiempo de ejecución empleado. Además, es necesario determinar un conjunto representativo de técnicas o métodos adicionales que sirvan de referencia en la comparación, y una serie de instancias de diferentes problemas de optimización a resolver con cada uno de los métodos. En la evaluación del modelo propuesto se utilizan como técnicas de comparación los dos algoritmos generales de optimización mediante búsqueda heurística más importantes, la búsqueda local y el temple simulado, y los dos modelos conexionistas más utilizados en este tipo de problemas, la red de Hopfield continua y la máquina de Boltzmann. Cada uno de los cinco métodos se evalúa mediante la ejecución de tres problemas clásicos, el problema del viajante sobre un recorrido de 15/20 ciudades, el problema de la partición de grafos sobre un grafo plano de 50 vértices que debe dividirse en 2/5 particiones, y el problema de la asignación cuadrática sobre un total de 15/20 plantas de producción a ubicar en 15/20 localizaciones distintas. Para efectuar las pruebas se ha realizado una implementación de cada método en lenguaje C, utilizando en su desarrollo una metodología de diseño y programación modular basada en máquinas abstractas. Las 20 pruebas efectuadas para cada método y problema se ejecutaron sobre una estación de trabajo UNIX HP-9000.

- *Desarrollo de una aplicación práctica sobre una importante área de gran interés en la industria: NEUROPROG, un programador predictivo para sistemas de producción discreta.* La problemática de la programación de tareas en producción suscita actualmente un gran interés en todo el mundo por diferentes razones, entre las que destaca por una parte la necesidad de incrementar la competitividad de las empresas, y por otra el desarrollo de computadores más potentes que permiten mejorar las prestaciones de los actuales programas de producción [GUID90]. En este trabajo se ha intentado demostrar la adecuación de los modelos basados en redes neuronales para la resolución de este tipo de problemas, teniendo en cuenta los sistemas conexionistas realizados previamente sobre el problema de la programación de producción [FOO88], [ZHOU91], [ZWIE91], así como sobre otros problemas de asignación de recursos [YU90], [PELL94]. A tal efecto, se ha diseñado un programador predictivo denominado NEUROPROG, que se basa en el modelo estocástico

ODE. NEUROPROG implementa un modelo de planta industrial simplificado, y se ha desarrollado íntegramente en lenguaje C, existiendo una versión para computadores PC con un interfaz gráfico de usuario basado en el uso de ventanas y menús desplegables.

1.4 ESQUEMA DE LOS RESTANTES CAPÍTULOS.

Esta tesis se ha estructurado en siete capítulos y cuatro apéndices. Los capítulos dos y tres de la tesis están dedicados a describir el *estado actual* de las técnicas de resolución de problemas en el área de la optimización combinatoria. Los capítulos cuatro al seis constituyen el *cuerpo principal* de la tesis, en el que se define y analiza el modelo neuronal ODE, y se desarrolla la aplicación NEUROPROG sobre el problema de la programación de producción discreta. Finalmente, el capítulo siete incluye las *conclusiones* sobre el trabajo realizado, así como las *líneas futuras de investigación*.

El contenido detallado de estos seis capítulos y los apéndices se expone brevemente a continuación:

- El capítulo dos introduce el área de la optimización combinatoria, y presenta los diferentes métodos de resolución basados en búsqueda heurística empleados en la actualidad. Para ello, se desarrolla un formalismo matemático del problema genérico de la optimización combinatoria, muy importante para la posterior definición de las diferentes técnicas de resolución, y se presentan bajo este formalismo los tres problemas ejemplo que se utilizarán más adelante en el análisis y evaluación del modelo neuronal desarrollado. El capítulo se completa con el estudio de los dos tipos de algoritmos generales de optimización combinatoria basados en búsqueda heurística más importantes: el algoritmo de búsqueda local, y el algoritmo del temple simulado.
- El capítulo tres resume brevemente los conceptos básicos del campo de las redes neuronales artificiales, y su relación con la resolución de problemas de optimización, proporcionando un estudio detallado de los dos modelos de red más empleados en este campo: la red de Hopfield continua, y la máquina de Boltzmann. También se describen otros modelos importantes, que han resultado de la evolución de los anteriores, como la máquina de Cauchy, la máquina de Gauss, y la red de temple determinista.

- El capítulo **cuatro** describe en detalle el optimizador discreto estocástico, ODE, modelo propuesto para la resolución de problemas de optimización combinatoria. En este capítulo se realiza un completo análisis de sus características estructurales, así como de su comportamiento dinámico, con el fin de obtener reglas generales que faciliten la determinación de los parámetros de la red en la implementación de problemas de optimización.

- El capítulo **cinco** presenta un estudio comparativo de la eficiencia del modelo neuronal propuesto, con respecto a las dos técnicas de búsqueda heurística presentadas en el capítulo dos, la búsqueda local y el temple simulado, y los dos modelos de red más importantes descritos en el capítulo tres, la red de Hopfield continua y la máquina de Boltzmann. Este estudio comparativo consiste en la evaluación estadística de la eficiencia de cada método en la resolución de tres problemas clásicos de optimización combinatoria: el problema del viajante comercial, el problema de la partición de grafos y el problema de la asignación cuadrática.

- El capítulo **seis** describe en detalle la aplicación NEUROPROG, desarrollada mediante el modelo ODE sobre el problema de la programación de producción discreta. En primer lugar, se introducen las características generales de los problemas de asignación de recursos, mostrando las particularidades y la complejidad del problema de la programación de producción (componentes de una planta industrial, restricciones y objetivos de la programación, modelos de planta). En segundo lugar, se hace un breve repaso de los métodos heurísticos que han sido utilizados para resolver este tipo de problema. A continuación, se describe detalladamente el modelo de planta industrial a utilizar en el desarrollo de la aplicación NEUROPROG, se define la red ODE necesaria para la resolución del problema, y se explican algunos detalles importantes de la implementación. Finalmente, se realiza un análisis de las pruebas efectuadas con el modelo.

- El capítulo **siete** incluye las conclusiones sobre el trabajo realizado, así como las líneas futuras de investigación que pueden desarrollarse como continuación del mismo.

- Por último, los **apéndices** se han concebido para contener los resultados de las pruebas de ejecución de los diferentes problemas utilizados en el análisis práctico del modelo propuesto, *problema del viajante*, *problema de la partición de grafos*, y *problema de la asignación cuadrática*, así como de la aplicación NEUROPROG sobre el *problema de la programación de producción discreta*.

2

Optimización Combinatoria

La ciencia de todo objeto se deriva de un conocimiento previo de las causas, la génesis y la construcción del mismo.

Tom Hobbes

2.1 INTRODUCCIÓN.

La *optimización* se puede considerar como la búsqueda de la mejor solución posible a un problema concreto, caracterizado por la existencia de un objetivo a lograr y un conjunto opcional de restricciones que deben satisfacerse para que la solución obtenida sea válida. Los problemas de optimización se presentan en una amplia variedad de ramas de la ciencia y tecnología. Además, muchas actividades cotidianas realizadas por las personas, como por ejemplo, elegir entre diferentes opciones la más adecuada, pueden verse como procesos mentales de optimización.

En muchas ocasiones, para poder determinar cuál es la mejor solución a un problema de optimización, es conveniente formularlo en términos matemáticos antes de abordar su resolución:

Objetivo:	$f(x_1, ..., x_n)$
Restricciones:	$r_1(x_1, ..., x_n)$
	.....
	$r_m(x_1, ..., x_n)$

Una vez traducido el problema al lenguaje matemático, es preciso disponer de técnicas que permitan conocer si éste tiene o no solución y, en caso de tenerla, cuáles son su localización y naturaleza. La teoría que proporciona los métodos y herramientas

precisos para estudiar este tipo de problemas es la *optimización matemática*. Atendiendo al carácter de las variables que intervienen en el problema, el estudio de la optimización matemática se divide en *optimización de tipo continuo* y *optimización discreta o combinatoria*:

- Los problemas de optimización matemática continua son problemas en los que las variables constituyentes $x_1, ..., x_n$ son de tipo continuo, es decir, son problemas en los que tanto la función objetivo f, como los grafos r_i asociados a las restricciones poseen como conjunto original un subconjunto continuo de R^n. Consecuentemente, las técnicas de resolución empleadas habitualmente se basan en el estudio de las propiedades de continuidad, derivabilidad y convexidad de las funciones reales.
- Por otra parte, la optimización combinatoria tiene igualmente por objeto el estudio de problemas de optimización con restricciones, pero en los que las variables constituyentes $x_1, ..., x_n$ toman valores discretos y finitos. Por tanto, estos problemas se caracterizan por la existencia de un gran, aunque finito, conjunto de soluciones posibles entre las que se debe encontrar aquella que optimice globalmente la función objetivo, y son objeto de estudio principalmente en las áreas de matemática discreta, y de computación e inteligencia artificial.

Los problemas de optimización combinatoria se dividen en clases de acuerdo con el tiempo computacional que es necesario emplear para su resolución. Un importante logro en este campo, obtenido al final de la década de los sesenta, es la conjetura de que existe una clase de problemas de optimización combinatoria de dificultad tal que sólo pueden resolverse mediante algoritmos de complejidad superpotencial con respecto a la dimensión del problema. Los problemas de este tipo se denominan problemas de complejidad temporal potencial no determinista completa o *NP-completos* (nondeterminis[illegible]polynomial time complete problems), y se caracterizan porque, en la práctica, su resolución exacta es imposible en tiempos de ejecución razonables. Durante los años setenta, el desarrollo de la teoría de la computación ha proporcionado una formulación rigurosa a esta conjetura, resultando la llamada teoría de la NP-completitud [GARE79].

De entre todos los problemas NP-completos de optimización combinatoria, el problema del viajante comercial (travelling salesman problem, TSP) es probablemente el que se conoce con mayor profundidad [LAWL85]. En este problema, un viajante debe visitar una lista de ciudades pasando sólo una vez por cada una de ellas, de forma que,

empezando y terminando por la misma ciudad, la distancia total recorrida sea mínima. La importancia de este problema está en el hecho de que reúne las características típicas de una amplia clase de problemas NP-completos de optimización combinatoria y contiene los dos elementos que llaman la atención a los matemáticos: simplicidad en su descripción y dificultad de resolución.

La resolución de problemas de optimización NP-completos se basa en la construcción de algoritmos apropiados sobre varias posibilidades:

- Algoritmos que obtienen soluciones óptimas, con el riesgo de emplear un tiempo computacional impracticable, excepto para problemas de tamaño muy reducido. Esta opción constituye la clase de *algoritmos de optimización*, y son de gran importancia desde un punto de vista teórico en el estudio y análisis de los problemas. Ejemplos de este tipo de algoritmos son los algoritmos de enumeración sistemática de soluciones y las técnicas de programación dinámica.
- La segunda opción constituye la clase de *algoritmos heurísticos*, entre los que están los algoritmos de búsqueda local y los algoritmos estocásticos o de aleatorización. Adicionalmente, pueden distinguirse en esta clase dos tipos diferentes de algoritmos: los algoritmos generales y los algoritmos específicos. Los algoritmos de tipo general son aplicables a una amplia variedad de problemas por lo que se les denomina algoritmos independientes del problema. Los algoritmos específicos utilizan información relativa al problema concreto a resolver, por lo que su aplicación se restringe a unos pocos casos, si bien generalmente permiten obtener soluciones de mayor calidad.
- Una tercera opción está en los *algoritmos genéticos*, basados en procesos de evolución simulada. No aplicables en grandes problemas de optimización por la capacidad de memoria que precisan, estos algoritmos son más apropiados en problemas de pequeño tamaño en los que intervienen un gran número de restricciones diferentes.
- Finalmente, la cuarta opción constituye la clase de *algoritmos conexionistas*, basados en las características de paralelismo y dinámica de procesamiento propias de las redes neuronales artificiales recurrentes.

Ante la imposibilidad práctica de emplear algoritmos que garanticen la obtención de soluciones óptimas globales, lo más deseable sería obtener un algoritmo de aproximación de alta calidad, que fuese aplicable a una amplia gama de problemas de optimización combinatoria.

En los restantes apartados del capítulo se proporciona una definición formal del problema de optimización combinatoria y se estudian en detalle los dos tipos de algoritmos heurísticos de aproximación más importantes: el algoritmo de búsqueda local y el algoritmo del temple simulado. Los algoritmos conexionistas se tratarán en el capítulo siguiente, completándose así una visión global de las técnicas actuales de resolución aproximada de grandes problemas de optimización combinatoria.

2.2 DEFINICIÓN FORMAL DEL PROBLEMA DE OPTIMIZACIÓN COMBINATORIA.

Definición. Un problema de optimización combinatoria es una cuádrupla (X, S, f, R), donde:

- X es el conjunto de variables del problema, $X = \{x_1, ..., x_n\}$, de dominios discretos y finitos $D_1, ..., D_n$.
- S es el espacio de soluciones del problema, formado por todas las combinaciones de valores de las variables, es decir, $S = D_1 \times ... \times D_n$.
- f es la función objetivo a optimizar, $f : S \rightarrow R$, que asigna un número real a cada combinación de valores de las variables del problema.
- R es el conjunto de restricciones, $R = \{r_1, ..., r_m\}$, cada una de las cuales determina los valores de compatibilidad entre un subconjunto específico de las variables del problema. Por tanto, cada r_i es un grafo del producto cartesiano de los dominios de las variables relacionadas por la restricción.

El objetivo del problema consiste en encontrar una asignación simultánea de valores de las variables, $s_o \in S$, que optimice (maximice o minimice) globalmente la función objetivo f, de tal forma que se satisfagan todas las restricciones impuestas entre las variables. Tal solución s_o se denomina solución óptima global, siendo $f(s_o)$ el objetivo óptimo a conseguir.

Ejemplo 1: Problema del Viajante Comercial. El problema del viajante comercial (Travelling Salesman Problem, TSP) consiste en encontrar el camino más corto de recorrido cíclico de n ciudades de forma que cada ciudad se visite una sola vez. Formalmente, el problema puede describirse mediante la cuádrupla (X, S, f, R):

- Conjunto de variables del problema $X = \{x_1, ..., x_n\}$, donde cada x_i representa la posición i del recorrido. El dominio finito D para cada una de las n variables es el conjunto de las n ciudades a visitar.
- Espacio de soluciones $S = D^n$.
- Función objetivo a minimizar, $f: S \rightarrow R$, determinada por la expresión:

$$f(s) = \sum_{i=1}^{n} d\left(x_i, x_{(i \bmod n)+1}\right)$$

donde $d(x_i,x_j)$ es la distancia existente entre las ciudades x_i y x_j.

- Conjunto de restricciones del problema $R = \{r\}$, donde r determina todos los recorridos válidos, es decir, aquellos en los que todas las ciudades se visitan una sola vez.

Ejemplo 2: Problema de la Partición de Grafos. El problema de la partición de grafos (Grapb Partition Problem, GPP) puede formularse de la siguiente forma: Dado un grafo plano G(V,E) con n vértices $V = \{v_1, ..., v_n\}$ y un conjunto E de conexiones entre los vértices, el problema consiste en dividir el grafo en p particiones equilibradas, es decir, con n/p vértices cada una, de forma que el número de conexiones entre particiones diferentes sea mínimo. Formalmente, el problema puede describirse mediante la cuádrupla (X, S, f, R):

- Conjunto de variables del problema $X = \{x_1, ..., x_n\}$, donde cada x_i representa el vértice i del grafo G. El dominio finito D para cada una de las n variables es el conjunto de p particiones.
- Espacio de soluciones $S = D^n$.
- Función objetivo a minimizar, $f: S \rightarrow R$, determinada por la expresión:

$$f(s) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \gamma\left(x_i, x_j\right) \cdot \phi(i,j)$$

donde γ es una función binaria que devuelve 1 si $x_i \neq x_j$ (los vértices i y j pertenecen a diferentes particiones) y 0 en caso contrario, y ϕ es una función binaria que devuelve 1 si el vértice i está conectado al j, y 0 en caso contrario.

- Conjunto de restricciones del problema $R = \{r\}$, donde r es el conjunto de combinaciones equilibradas de valores de las variables.

Ejemplo 3: Problema de la Asignación Cuadrática. El problema de la asignación cuadrática (Quadratic Assignment Problem, QAP) puede interpretarse como la ubicación óptima de n plantas de proceso en p lugares posibles ($p \geq n$), teniendo en cuenta que entre cada dos plantas debe transportarse una cantidad determinada de material con un coste unitario distinto según el lugar en que se hallen éstas. El objetivo es minimizar el coste total del transporte de materiales entre las diferentes plantas de proceso. Formalmente, el problema puede describirse mediante la cuádrupla (X, S, f, R):

- Conjunto de variables del problema $X = \{x_1, ..., x_n\}$, donde cada x_i representa la planta de proceso i a ubicar. El dominio finito D para cada una de las n variables es el conjunto de los p lugares posibles.
- Espacio de soluciones $S = D^n$.
- Función objetivo a minimizar, $f: S \rightarrow R$, determinada por la expresión:

$$f(s) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c\left(x_i, x_j\right) \cdot q\left(i, j\right)$$

  donde $c(x_i,x_j)$ es el coste unitario del transporte del material entre los lugares en que se hallan las plantas i y j, y q(i, j) es la cantidad de material que debe transportarse entre ambas.
- Conjunto de restricciones del problema R = {r}, donde r determina todas las combinaciones de valores de las variables en que ninguna variable toma el mismo valor que otra, es decir, cuando todas las plantas se asignan a lugares diferentes.

2.3 GENERACIÓN DE SOLUCIONES.

Antes de comenzar el estudio de los algoritmos de búsqueda local y temple simulado es necesario introducir previamente dos conceptos necesarios para su definición: la *estructura de soluciones k-próximas* del problema y el *mecanismo de generación de soluciones*.

Definición. Dado un problema de optimización combinatoria (X, S, f, R), se denomina *estructura de soluciones k-próximas* del problema a la aplicación $\lambda: S \rightarrow \mathcal{P}(S)$ que asocia cada solución $s_i \in S$ con un conjunto $S_i \subseteq S$ de soluciones que se diferencian de s_i como máximo en el valor de k variables. El conjunto S_i se llama conjunto de soluciones k-próximas a la solución s_i, y cada $s_j \in S_i$ se llama solución k-próxima a s_i.

Definición. Dado un problema de optimización combinatoria (X, S, f, R), una estructura de soluciones k-próximas δ, y una solución en curso s_i, se dice que s_i es una *solución óptima local* de f con respecto a δ si para todo $s_j \in S_i$ se cumple que $f(s_i) \leq f(s_j)$ en caso de minimización, o $f(s_i) \geq f(s_j)$ en caso de maximización.

Definición. Dado un problema de optimización combinatoria (X, S, f, R) y una estructura de soluciones k-próximas δ, se dice que δ es una *estructura exacta* si para todo $s_i \in S$ que sea óptimo local de f con respecto a δ, s_i es también óptimo global.

Observación. Para todo problema de optimización combinatoria (X, S, f, R), si el número de variables del problema es n, la estructura de soluciones n-próximas es exacta.

Definición. Dado un problema de optimización combinatoria (X, S, f, R), una estructura de soluciones k-próximas δ, y una solución en curso s_i, se denomina *mecanismo de generación de soluciones* al medio de selección de una nueva solución s_j no procesada previamente a partir de la estructura de soluciones k-próximas a la solución actual s_i.

2.4 BÚSQUEDA LOCAL.

2.4.1 Algoritmo de búsqueda local.

Los algoritmos de búsqueda local constituyen una clase interesante de algoritmos generales de optimización aproximada, basados en la conjunción de dos técnicas deterministas de búsqueda heurística: la escalada (hill-climbing), y la satisfacción de restricciones (constraint satisfaction).

Algoritmo. Dado un problema de optimización combinatoria (X, S, f, R), una estructura de soluciones k-próximas δ, y un mecanismo arbitrario de generación de soluciones, el algoritmo de búsqueda local puede describirse de la siguiente forma:

1. Se establece como solución inicial en curso una solución posible $s_i = s_0$, elegida normalmente de forma aleatoria.

2. Se inicializa el conjunto S_i de soluciones k-próximas a la solución actual s_i.

3. Mediante el mecanismo de generación de soluciones se extrae una solución s_j del conjunto de soluciones k-próximas a la actual.

4. Si la nueva solución s_j verifica todas las restricciones del problema y es de mejor calidad que la actual s_i, entonces s_j pasa a ser la solución en curso, inicializándose su correspondiente conjunto de soluciones k-próximas.

5. Si existen soluciones k-próximas a la actual aún no procesadas, se reiteran los pasos 3 y 4. El algoritmo termina cuando el conjunto de soluciones k-próximas a la actual se ha procesado completamente sin obtenerse mejoras en la calidad de las soluciones.

2.4.2 Análisis del comportamiento práctico.

A pesar de su sencillez, generalidad y flexibilidad, los algoritmos de búsqueda local presentan una serie de inconvenientes importantes:

- Por definición, estos algoritmos obtienen soluciones óptimas locales, y no globales, excepto cuando la estructura de soluciones k-próximas utilizada es exacta. Sin embargo, las estructuras exactas son inaplicables en la mayoría de los casos, ya que conllevan la enumeración completa de las soluciones.
- La calidad de la solución obtenida por un algoritmo de búsqueda local depende directamente de la solución inicial elegida, pero en la mayoría de los problemas de optimización combinatoria no existen normas generales para la apropiada elección de la misma, por lo que suele elegirse al azar. De esta forma, la desviación típica que presentan las soluciones obtenidas mediante búsqueda local con respecto a la media es bastante grande. Además, el carácter local de la búsqueda hace que la calidad media de las soluciones con respecto a la óptima empeore a medida que crece el tamaño del problema [JOHN85].
- La complejidad temporal del caso más desfavorable, cota superior del tiempo de computación necesario para resolver el problema, no se ha podido determinar en muchos casos. Así, el conocido algoritmo de búsqueda local propuesto originalmente por Lin para el problema del viajante [LIN65], basado en una estructura de soluciones 2-próximas, posee una complejidad temporal para el caso más desfavorable aún no determinada, si bien más recientemente se ha demostrado que, para una clase limitada de instancias del problema, la complejidad temporal media es de orden potencial [KERN86].

Para evitar estos inconvenientes, sin renunciar a las características básicas de los algoritmos de búsqueda local, pueden considerarse dos alternativas posibles:

- Ejecutar el algoritmo de búsqueda local un gran número de veces con diferentes soluciones iniciales, eligiendo al final la mejor solución obtenida. De esta forma, y bajo la condición de que todas las soluciones posibles sean utilizadas como soluciones iniciales, el algoritmo converge asintóticamente hacia la solución óptima global.
- Introducir una estructura de soluciones próximas más compleja, de forma que el algoritmo efectúe la búsqueda por una parte mayor del espacio de soluciones. Sin embargo, esta alternativa requiere un conocimiento específico del problema de optimización combinatoria a resolver, por lo que el algoritmo pierde generalidad y se transforma en un algoritmo dependiente del problema.

Una forma adicional de intentar resolver los inconvenientes del algoritmo de búsqueda local consiste en aceptar de forma estocástica un número limitado de transiciones de la solución en curso hacia soluciones factibles que supongan una pérdida de calidad en la función objetivo. De esta forma, el algoritmo puede escapar de los puntos de óptimo local, en beneficio de una solución final de mejor calidad. Esta alternativa da lugar al algoritmo del temple simulado, el cual se describe a continuación.

2.5 TEMPLE SIMULADO.

2.5.1 Fundamentos del temple simulado.

En la primera mitad de la década de los ochenta, Kirkpatrick, Gelatt y Vecchi [KIRK83], e independientemente Cerny [CERN85], introdujeron el concepto del temple simulado[1] en el campo de la optimización combinatoria. Este concepto se basa en la fuerte analogía que puede establecerse entre la evolución de un sistema físico sometido a una fuente de calor y la resolución de grandes problemas de optimización combinatoria. En los párrafos siguientes de este apartado se desarrollan de forma intuitiva algunos

[1]Traducción usual de *simulated annealing*, que significa literalmente *recocido simulado*. Ver nota de la página 4, capítulo 1.

conceptos básicos que constituyen el fundamento del temple simulado y permiten comprender correctamente dicha analogía.

La *mecánica estadística* es una rama de la física que estudia los sistemas formados por un número tan elevado de partículas, que el comportamiento del sistema no puede determinarse evaluando la contribución de cada partícula por separado. En lugar de esto, se asume que el sistema físico es compatible con una población estadística, cuyos parámetros coinciden con los parámetros observables del sistema físico. De esta forma, puede determinarse, por ejemplo, la energía media del sistema a partir de la media estadística de las energías de las partículas constituyentes. Si P_i es la probabilidad de que una partícula i tenga la energía E_i, entonces la energía media del sistema es:

$$\overline{E} = \sum_i P_i E_i$$

Una clase muy importante de este tipo de sistemas físicos es la formada por los que están en contacto con una *fuente de calor*. Las fuentes de calor se caracterizan porque toda interacción con el sistema en cuestión da lugar únicamente a cambios infinitesimales en las propiedades de la fuente, mientras que el sistema puede sufrir cambios de importancia hasta que se alcance el equilibrio térmico, el cual viene determinado por la denominada *distribución de Boltzmann* [TODA83]. Esta distribución proporciona la probabilidad de que el sistema físico se halle en un estado i con una energía E_i a la temperatura absoluta T, y viene dada por la expresión:

$$P(i,T) = \frac{e^{\frac{-E_i}{k_B T}}}{Z(T)}$$

donde k_B es una constante física conocida como *constante de Boltzmann*, el factor exponencial $e^{\frac{-E_i}{k_B T}}$ se denomina *factor de Boltzmann*, y Z(T) es la *función de partición*, definida como:

$$Z(T) = \sum_j e^{\frac{-E_j}{k_B T}}$$

donde el sumatorio se extiende sobre el conjunto de estados del sistema.

En la física de la materia condensada, el temple (annealing) es un proceso térmico destinado a obtener estados de baja energía en un sólido sometido a una fuente de calor. Este proceso consiste fundamentalmente en los dos pasos siguientes:

- Incrementar la temperatura de la fuente de calor hasta el punto de fusión del sólido, momento en el que las partículas del mismo se disponen aleatoriamente.
- Decrementar paulatinamente la temperatura de la fuente a fin de que las partículas del sólido se organicen en una estructura cristalina que representa el estado de mínima energía del sólido.

El estado de mínima energía se obtiene sólo si la fase de enfriamiento se realiza suficientemente despacio. En caso contrario, las partículas del sólido se disponen en estructura amorfa resultando un estado estable de alta energía en lugar del deseado de mínima energía. En otras palabras, se alcanza un mínimo local, pero no global.

El proceso del temple puede modelizarse con éxito mediante métodos de simulación por computador. Ya en 1953, N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller y E. Teller [METR53] desarrollaron un sencillo algoritmo basado en la mecánica estadística para simular la evolución de un sólido sometido a una fuente de calor. Este algoritmo utiliza técnicas de Monte Carlo, y genera una secuencia de L transiciones de estado del sólido para una temperatura dada T, hasta alcanzar el estado de equilibrio térmico. Cada una de las transiciones de estado se realiza de la siguiente forma. Dado un estado actual i del sólido con energía E_i, se aplica un mecanismo de perturbación que produce un pequeño cambio en el estado actual i, por ejemplo el desplazamiento de una partícula, para generar un siguiente estado j del sólido. Si la energía E_j del siguiente estado es menor que la del estado actual E_i, entonces se acepta el estado j como el estado actual del sólido. En caso contrario, el estado j se acepta como estado actual con una probabilidad dada por la expresión:

$$e^{\frac{E_i - E_j}{k_B T}}$$

donde k_B es la constante de Boltzmann y T es la temperatura absoluta de la fuente de calor. La regla de transición de estados descrita se denomina *criterio de Metropolis*, y el algoritmo en conjunto se llama *algoritmo de Metropolis*. Si se aplica el algoritmo de Metropolis en la simulación del proceso de enfriamiento del sólido, éste alcanzará el estado de mínima energía, bajo el supuesto de que para cada valor de la temperatura se consigue el equilibrio térmico.

2.5.2 Algoritmo del temple simulado.

El temple simulado es un algoritmo estocástico de resolución de problemas de optimización combinatoria, basado en el algoritmo de Metropolis para simulación de sistemas físicos sometidos a una fuente de calor. El fundamento de la analogía que se establece entre un problema de optimización combinatoria y un sistema físico de múltiples partículas viene dado por las siguientes equivalencias:

- Las soluciones del problema de optimización combinatoria son equivalentes a los estados del sistema físico.
- El coste de una solución es equivalente a la energía de un estado.
- La temperatura del sistema físico se considera en el problema de optimización combinatoria como un parámetro de control $T > 0$.

El algoritmo del temple simulado puede verse ahora como una iteración del algoritmo de Metropolis, para valores decrecientes del parámetro de control T.

Algoritmo. Dado un problema de optimización combinatoria (X, S, f, R), una estructura de soluciones k-próximas λ, y un mecanismo arbitrario de generación de soluciones, el algoritmo del temple simulado puede describirse de la siguiente forma:

1. Se establece como solución inicial en curso una solución posible $s_i = s_0$, elegida normalmente de forma aleatoria.
2. Se inicializa al azar el conjunto S_i de soluciones k-próximas a la solución actual s_i.
3. Se inicializa la temperatura o parámetro de control $T = T_0$.
4. Mediante el mecanismo de generación de soluciones se extrae una solución s_j del conjunto de soluciones k-próximas a la actual.
5. Si la nueva solución s_j satisface todas las restricciones del problema entonces verificar el siguiente criterio de aceptación: Si la nueva solución s_j es de mejor calidad que la actual s_i, entonces s_j pasa a ser la solución en curso, inicializándose su correspondiente conjunto de soluciones k-próximas. En caso contrario, si

$$e^{\frac{f(s_i)-f(s_j)}{T}} > \text{random}[0,1[$$

se acepta igualmente s_j como solución en curso, y se inicializa su correspondiente conjunto de soluciones k-próximas.

6. Si el número de transiciones de estado efectuadas para el valor actual de la temperatura es inferior a L volver al paso 4.

7. Se decrementa el valor de la temperatura y se reiteran los pasos 4 al 6. El algoritmo termina si la solución actual no se modificó en ninguna de las transiciones efectuadas para el último valor de la temperatura.

La característica más importante del algoritmo del temple simulado es que, además de aceptar transiciones que suponen una mejora en el coste de la solución, también permite aceptar, hasta un cierto límite, transiciones que suponen una pérdida de calidad de la solución. La probabilidad de aceptar transiciones con pérdida de calidad se implementa comparando el valor de la expresión

$$e^{\frac{f(s_i)-f(s_j)}{T}}$$

con un número aleatorio generado a partir de una distribución uniforme sobre el intervalo [0,1[. Inicialmente, para valores grandes de la temperatura T, se aceptan grandes pérdidas de calidad de la solución. Al decrecer T, sólo se aceptan pérdidas de calidad más pequeñas. Finalmente, cuando la temperatura se aproxima a cero, no se acepta ninguna pérdida de calidad. Esta característica significa que el algoritmo del temple simulado puede escapar de los mínimos locales, aceptando transiciones intermedias hacia soluciones de mayor coste en beneficio de una mejor solución final.

En virtud de la analogía con el algoritmo de Metropolis (simulación del proceso físico del temple), el algoritmo del temple simulado debe converger asintóticamente hacia el conjunto de soluciones óptimas globales del problema. No obstante, para garantizar esta convergencia asintótica es necesaria la existencia de una única distribución estacionaria para el algoritmo del temple simulado, equivalente a la existencia de la distribución de Boltzmann de equilibrio térmico en mecánica estadística.

Proposición. Dado un problema de optimización combinatoria (X, S, f, R), una estructura de soluciones k-próximas $\mathcal{N}$, y un mecanismo arbitrario de generación de soluciones, después de un número L de transiciones para una temperatura dada T aplicando la probabilidad de aceptación

$$e^{\frac{f(s_i)-f(s_j)}{T}} > \text{random}[0,1[$$

el algoritmo del temple simulado encontrará una solución $s_i \in S$ con una probabilidad que viene determinada por la siguiente distribución estacionaria:

$$P(s_i, T) = \frac{e^{\frac{-f(s_i)}{T}}}{N(T)}$$

siendo N(T) la *función de normalización*, definida como:

$$N(T) = \sum_j e^{\frac{-f(s_j)}{T}}$$

donde el sumatorio se extiende sobre el espacio de soluciones del problema.

La proposición anterior conjetura la existencia de una distribución estacionaria para el algoritmo del temple simulado equivalente a la distribución de Boltzmann del proceso físico análogo. E. Aarts y J. Korst proporcionan una demostración completa de esta proposición en su libro *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing* [AART89], utilizando para ello un formalismo matemático del algoritmo del temple simulado basado en la teoría de las cadenas de Markov. Así mismo, se proporcionan las condiciones suficientes de convergencia asintótica del temple simulado hacia el conjunto de soluciones óptimas globales del problema [AART89]. Otros autores han demostrado igualmente condiciones similares de convergencia asintótica del temple simulado [GEMA84], [MITR86], mientras que Hajek ha proporcionado condiciones necesarias y suficientes [HAJE88].

La principal consecuencia que se desprende del estudio matemático del temple simulado es que para converger asintóticamente hacia la solución óptima con probabilidad 1 es necesario un número L de transiciones de estado que tiende a infinito para cada valor de la temperatura. Incluso una buena aproximación al comportamiento asintótico requiere un número de transiciones de estado que en la mayoría de los problemas excede del tamaño del espacio de soluciones, lo que implica un tiempo exponencial de ejecución del algoritmo. Por esta razón, el temple simulado no es un algoritmo práctico de optimización global, sino que debe usarse como algoritmo de optimización aproximada.

2.5.3 Programas de templado.

La implementación práctica del algoritmo del temple simulado puede realizarse generando una secuencia finita de valores decrecientes de la temperatura T, y un número finito L de transiciones de estado para cada valor de la temperatura. Para ello debe especificarse un *programa de templado*, o conjunto de parámetros cuyo objetivo es controlar la evolución del algoritmo.

El siguiente programa de templado usado frecuentemente en la literatura es una variante del propuesto por Kirkpatrick, Gelatt y Vecchi [KIRK83], y se compone de cuatro parámetros. Los tres primeros especifican la secuencia finita de valores de la temperatura, y el último indica el número finito de transiciones para cada valor de la misma:

- *Valor inicial de la temperatura*, T_0. Este valor inicial de la temperatura debe ser suficientemente elevado como para que se acepten todas las transiciones de estado realizadas.
- *Función de decremento de la temperatura*. Generalmente, se utiliza una función de decremento exponencial de la forma $T_n = T_0 \cdot (\alpha)^n$, donde α es una constante un poco menor que la unidad. Los valores usuales de α oscilan entre 0.8 y 0.99.
- *Criterio de finalización*. Generalmente, el criterio de finalización consiste en verificar que no se aceptó ninguna de las transiciones efectuadas para el último valor de la temperatura.
- *Número de transiciones de estado*, L, para cada valor de la temperatura. Intuitivamente, el número de transiciones para cada temperatura debe ser suficientemente grande como para que, si no se aceptan transiciones, pueda recorrerse toda la estructura de soluciones k-próximas a la solución en curso. Por ello, L deberá ser igual al número máximo de soluciones k-próximas a una dada.

En la práctica, la temperatura inicial T_0 puede determinarse mediante el siguiente algoritmo simple:

Algoritmo. Cálculo de la temperatura inicial en el temple simulado.

1. Inicializar T_0 con un valor positivo (por ejemplo, T_0 puede tomar el valor absoluto del coste de una solución factible cualquiera).

2. Partiendo de una solución factible elegida al azar, realizar un número reducido de transiciones que supongan un deterioro de calidad de la solución (por ejemplo, 100), aplicando el criterio de aceptación.

3. Si el numero de transiciones aceptadas es inferior al total de transiciones efectuadas, entonces incrementar T_0 multiplicándolo por un factor mayor que la unidad y volver al paso 2. El algoritmo termina cuando todas las transiciones realizadas con pérdida de calidad se aceptan, lo que indica que la temperatura ya es suficientemente elevada.

En la analogía con el sistema fisico, este algoritmo coincide con el calentamiento del sólido hasta el punto de fusión, en el que todas las particulas del mismo se disponen al azar.

La búsqueda de programas de templado apropiados ha sido objeto de estudio en numerosos artículos de investigación [COLL87], [LAAR87]. E. Aarts y J. Korst han desarrollado una implementación del temple simulado basada en un programa de templado con un tiempo potencial de ejecución [AART89].

2.5.4 Análisis del comportamiento práctico.

La principal caracteristica diferencial del algoritmo del temple simulado frente al algoritmo de búsqueda local es que el criterio estocástico de aceptación de transiciones permite escapar de los mínimos locales parciales con el fin de obtener una solución final de mejor calidad, sin perder por ello la sencillez y generalidad de la búsqueda local. De hecho, la búsqueda local puede verse como un caso particular del temple simulado en el que la temperatura permanece constante a un valor próximo a cero.

El comportamiento práctico del temple simulado puede resumirse en los siguientes puntos [AART89]:

- El algoritmo permite obtener soluciones de alta calidad, aunque con un coste computacional importante, superior al del algoritmo de búsqueda local. No obstante, es posible comparar la eficiencia global de ambas técnicas simplemente reiterando el algoritmo de búsqueda local tantas veces como sea posible en el tiempo medio de ejecución del algoritmo del temple simulado. Si se realiza una implementación de este último con un programa de templado proporcional a la

inversa de la exponencial de la iteración en curso, la mejor solución obtenida mediante búsqueda local es de peor calidad que la solución media obtenida mediante el temple simulado, hecho que se ve acentúado cuando aumenta el tamaño del problema.

- El algoritmo es robusto, es decir, la solución final no depende directamente de la solución inicial elegida. En consecuencia, la desviación típica que presentan las soluciones obtenidas mediante este algoritmo con respecto a la media es mucho menor que en el caso de la búsqueda local.
- La calidad de la solución obtenida depende fundamentalmente de la función que determina la velocidad de decremento de la temperatura. Los valores inicial y final de la temperatura sólo juegan un papel secundario en este aspecto, debido principalmente a que se eligen con un razonable grado de exactitud.

El algoritmo del temple simulado se ha utilizado en la práctica en numerosas aplicaciones de optimización y asignación de recursos [ABRA92], [DAOU95]. En la actualidad se realizan investigaciones sobre posibles mejoras del algoritmo, basadas en la paralelización del mismo [AZEN92], y en su combinación con técnicas de computación evolutiva (algoritmos genéticos) [ONOD93], [LIN93], [WILH94].

3

Redes Neuronales Artificiales y Optimización Combinatoria

Todos los argumentos que se fundan en la experiencia están basados en la semejanza que descubrimos en los objetos naturales, la cual nos induce a esperar efectos semejantes a los que hemos visto seguir a tales objetos.

David Hume

3.1 INTRODUCCIÓN.

En el capítulo precedente se realizó una introducción al problema de la optimización combinatoria, estudiándose en detalle las dos técnicas de resolución más importantes basadas en algoritmos heurísticos: búsqueda local y temple simulado.

Otra clase diferente de técnicas de resolución es la que se originó en el campo de las redes neuronales artificiales, a partir del análisis de las propiedades de computación colectiva de algunas redes recurrentes simétricas, estudio comenzado por John Hopfield en 1982 sobre el autocorrelador discreto [HOPF82].

El presente capítulo tiene como objetivo completar la visión general del conjunto de métodos de resolución aproximada de problemas de optimización, mostrando los diferentes algoritmos conexionistas existentes en la actualidad. Para ello, se introducen en primer lugar los conceptos fundamentales de los modelos conexionistas, y a continuación se describen detalladamente las dos redes neuronales más utilizadas en la resolución de problemas combinatorios, la *red de Hopfield continua* [HOPF84], y la

máquina de Boltzmann [HINT86], junto a sus modelos derivados, la *máquina de Cauchy* [SZU87], la *máquina de Gauss* [AKIY89], y la *red de temple determinista* [BOUT89].

3.2 REDES NEURONALES. CONCEPTOS FUNDAMENTALES.

3.2.1 Definición de red neuronal.

Las *redes neuronales artificiales*, también conocidas como *sistemas neuronales artificiales*, *sistemas conexionistas*, *redes adaptativas*, y *procesadores paralelos distribuidos*, son modelos simplificados de los sistemas nerviosos naturales. La siguiente definición, dada por Hecht-Nielsen en 1988 [HECH88], formaliza el concepto de red neuronal artificial:

> "Una red neuronal artificial es un sistema computacional de procesamiento paralelo distribuido, formado por un conjunto de unidades procesadoras elementales dotadas de una pequeña memoria local y relacionadas entre sí en forma de red mediante conexiones con pesos asociados. Cada unidad de procesamiento posee una o varias conexiones de entrada, y una única conexión de salida que enlaza con tantas conexiones colaterales como se desee. Todo el procesamiento asociado a una unidad elemental es de tipo local, esto es, depende exclusivamente de los valores que tomen las señales de entrada de la unidad y del estado interno de la misma".

3.2.2 Elementos de un modelo neuronal.

La aproximación estándar a la construcción o diseño de un modelo de red neuronal consiste, en primer lugar, en proponer una arquitectura o estructura de la red, a continuación definir el comportamiento dinámico de la misma, y, finalmente, mostrar las propiedades interesantes que el modelo pueda poseer, junto con sus posibles aplicaciones. A continuación se muestran las características generales de los modelos

conexionistas, estudiando los diferentes aspectos estructurales y funcionales que los definen, así como otras propiedades relevantes.

La descripción general de un modelo conexionista viene dada por cinco componentes o elementos principales:

- Las *unidades de procesamiento* o *nodos*, que son las unidades computacionales básicas de una red neuronal artificial, y en ellos se efectúa todo el procesamiento asociado a la misma.
- La *arquitectura de la red*, definida por la disposición y la forma de interconexión de los nodos.
- El *esquema de procesamiento* de los patrones de entrada a través de la red.
- La *regla de aprendizaje*, por la cual se modifican los pesos de las conexiones con el fin de redefinir dinámicamente el comportamiento de la red.
- El *esquema de representación*, o significado de los patrones de entrada/salida dentro del entorno en el cual opera el sistema.

3.2.3 Unidades de procesamiento.

Esquema general de una unidad de procesamiento o nodo

Todo modelo conexionista tiene como base un conjunto de *unidades de procesamiento*, también llamadas *nodos*, *neuronas* o *unidades lógicas de umbral*. Definir el conjunto de unidades de procesamiento y lo que ellas representan es el primer estadio de especificación de un modelo conexionista. En algunos de los modelos los nodos pueden representar objetos conceptuales particulares, tales como rasgos, letras, palabras o conceptos, mientras que en otros son simplemente elementos abstractos que sólo tomados en conjunto dan origen a las entidades o patrones de actividad del sistema.

Cada nodo en una red neuronal posee un *estado de activación* que depende de las entradas asociadas al mismo. Todo el procesamiento de un modelo neuronal se lleva a cabo por estos nodos de forma local, es decir, no hay ejecutivos u otros supervisores. La misión de un nodo es simplemente recibir las entradas procedentes de otros nodos, y, en función de ellas, calcular su estado de activación, el cual se envía a otros nodos como valor de salida. El sistema es intrínsecamente paralelo en el sentido de que muchos

nodos pueden desarrollar sus cómputos al mismo tiempo. La figura 3.1 muestra el esquema general de una unidad de procesamiento o nodo:

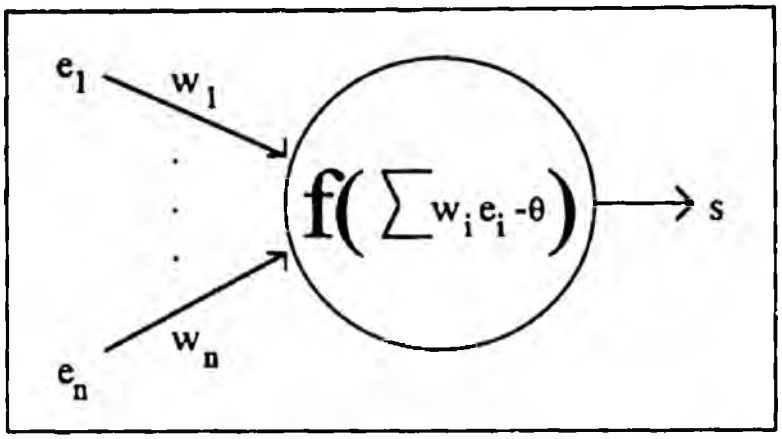

Figura 3.1: Unidad de procesamiento o nodo.

Las señales de entrada forman un vector de entrada $E = (e_1, ..., e_n)$, donde e_i es el estado de activación del i-ésimo nodo antecesor. Así mismo, es habitual que cada conexión de entrada posea un valor asociado denominado peso o fuerza de la conexión, formándose para cada nodo un vector de pesos $W = (w_1, ..., w_n)$, donde w_i es el peso de la conexión que une el nodo a su i-ésimo antecesor. En ocasiones se utiliza un parámetro adicional θ como valor interno de umbral. Las entradas E, sus pesos asociados W, y el parámetro θ se utilizan para calcular el estado de activación o valor de salida S del nodo. Este cálculo se realiza normalmente procesando los vectores E y W de acuerdo con una regla de combinación, restando el valor θ, y evaluando el resultado mediante una función de umbral f.

Reglas de combinación

En un modelo conexionista pueden aparecer tres tipos de *reglas de combinación* de las entradas de los nodos: la regla aditiva simple, la regla multiplicativa simple, y la regla aditiva-multiplicativa:

- La *regla aditiva simple* se utiliza en la mayoría de los modelos conexionistas desarrollados. Consiste en calcular la entrada global neta al nodo μ como la suma de las entradas individuales ponderadas por sus pesos respectivos:

$$\mu = \sum_{i=1}^{n} w_i e_i$$

Los nodos cuya entrada global se calcula mediante la regla aditiva simple se denominan nodos SIGMA (figura 3.2).

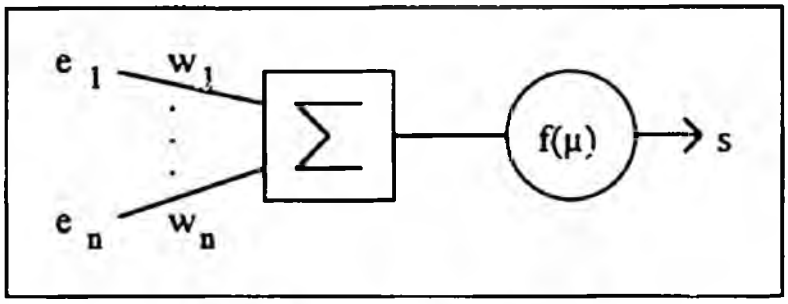

Figura 3.2: Regla aditiva simple.

- Existen ocasiones en que se precisan conexiones multiplicativas en las cuales los valores de las entradas son multiplicados en vez de sumados, antes de la operación de la función de umbral. Este es el caso de los nodos PI, que emplean la *regla de combinación multiplicativa simple* para el cálculo de su estado de activación. Esta regla se define matemáticamente de la siguiente forma:

$$\mu = w\prod_{i=1}^{n} e_i$$

Según este esquema, si una entrada de un par multiplicativo es 0, el otro miembro queda anulado sin importar que su valor sea incluso muy intenso, y por otra parte, si una entrada de un par es 1, la otra se transmite sin cambios al nodo receptor (figura 3.3):

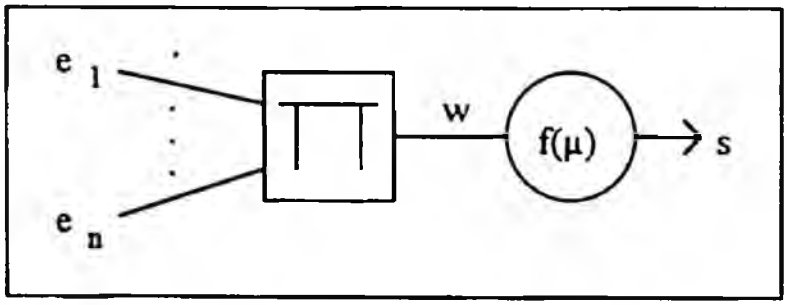

Figura 3.3: Regla multiplicativa simple.

- La *regla aditiva-multiplicativa* es una combinación de las anteriores, y da lugar a los nodos SIGMA-PI. La característica diferencial más importante de los nodos SIGMA-PI es que las entradas no se tratan individualmente, sino agrupadas en conjuntos o conjunciones, que son multiplicadas entre sí antes de su suma (figura 3.4).

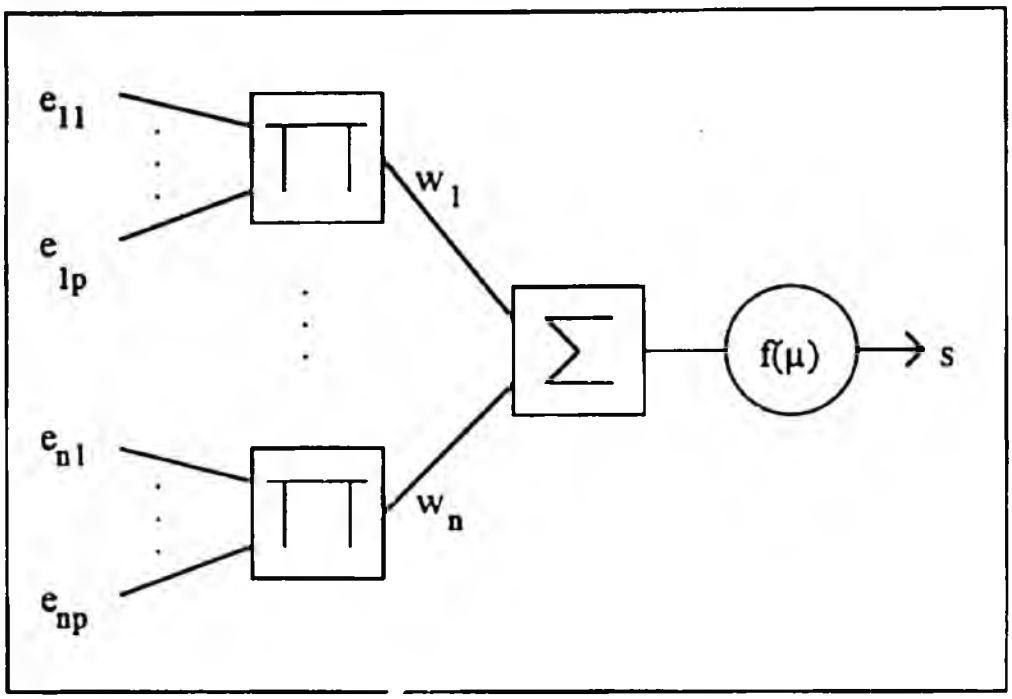

Figura 3.4: Regla aditiva-multiplicativa.

Matemáticamente, la regla aditiva-multiplicativa viene dada por la suma de las conjunciones o productos ponderados de las entradas individuales al nodo:

$$\mu = \sum_{i=1}^{n}\left(w_i \prod_{j=1}^{p} e_{ij} \right)$$

Esta regla permite representar cualquier función monótona, por lo que no es necesario desarrollar reglas de combinación más complejas. De hecho, la regla aditiva simple puede verse como un caso particular de la regla aditiva-multiplicativa, en la que cada conjunción de entradas a un nodo estaría formada por un único elemento. Igualmente, la regla multiplicativa simple sería un caso particular de la regla aditiva-multiplicativa, en la que cada nodo tendría una única conjunción de entradas individuales.

En general, el número de conexiones que se integran en una conjunción multiplicativa indica el orden de correlación entre los nodos conectados. Así, un nodo SIGMA, con conexiones independientes que no se multiplican previamente a la operación de suma ponderada, posee una correlación de primer orden. Por otra parte, un nodo PI o SIGMA-PI, con conjunciones multiplicativas formadas por varias conexiones, posee una correlación de orden superior. Esta correlación será de segundo orden si las conjunciones multiplicativas se forman por pares de conexiones, de tercer orden si las conexiones se unen en grupos de tres, y así sucesivamente.

Las correlaciones de orden superior se usan frecuentemente en aplicaciones de visión artificial en las que se desea que la red neuronal reconozca patrones de entrada aún cuando éstos hayan sufrido desplazamientos, escalados, y/o rotaciones.

Funciones de umbral

La *función de umbral*, f, también llamada *función de discriminación*, *función de activación*, o *función de señal*, tiene como misión normalizar la entrada global μ de un nodo a un rango predefinido en el que debe encontrarse su valor de activación o estado S. Este rango puede ser discreto (generalmente binario o bipolar), o continuo.

Existen cuatro tipos comunes de funciones de umbral: las funciones de salto, que producen valores de activación de tipo discreto, y las funciones de rampa, lineales, y sigmoidales, que producen valores de tipo continuo:

- La función de umbral más elemental es la *función de salto* (figura 3.5):

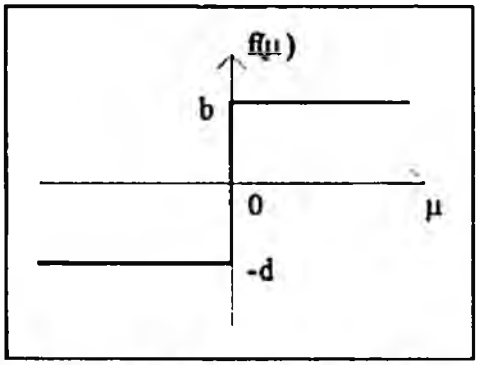

Figura 3.5: Función de salto.

Esta función responde sólo al signo de la entrada, obteniéndose un valor b si la entrada global μ al nodo es positiva, y un valor -d si ésta es negativa. Matemáticamente se define:

$$f(\mu) = b \qquad \text{si } \mu > 0$$

$$f(\mu) = -d \qquad \text{si } \mu \leq 0$$

Generalmente, la función de salto es de tipo binario, donde b = 1 y -d = 0. Así mismo, puede haber funciones de salto más complejas que produzcan más de dos valores discretos de activación (funciones multisalto).

- La *función de rampa* (figura 3.6), es similar a la función de salto, pero en ella el paso del valor mínimo al máximo no se produce bruscamente, sino de forma progresiva de acuerdo con una ecuación de tipo lineal.

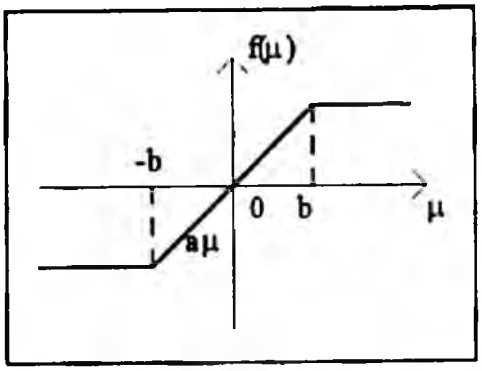

Figura 3.6: Función de rampa.

Matemáticamente se define:

$$f(\mu) = ab \qquad \text{si } \mu \geq b$$

$$f(\mu) = a\mu \qquad \text{si } -b < \mu < b$$

$$f(\mu) = -ab \qquad \text{si } \mu \leq -b$$

donde a es una constante positiva que regula la magnitud de la entrada μ, y b (-b) es el mayor (menor) valor posible de salida del nodo, también llamado valor de saturación.

- La *función lineal* está representada en la figura 3.7:

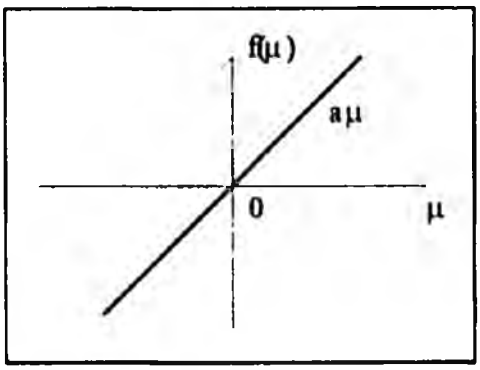

Figura 3.7: Función lineal.

Matemáticamente la función de umbral lineal viene determinada por la siguiente ecuación:

$$f(\mu) = a\mu$$

donde a es una constante positiva que regula la magnitud de entrada global μ del nodo. Es, por tanto, similar a la función de rampa, pero sin valores de saturación que limiten el crecimiento o decrecimiento del estado de activación del nodo.

- Las *funciones sigmoidales* (con forma de S) son funciones monótonas crecientes, acotadas, continuas y derivables, que proporcionan una respuesta no lineal, y sin saltos o cambios bruscos, a la entrada global μ (figura 3.8):

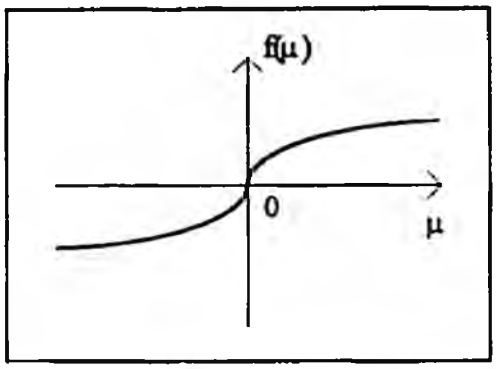

Figura 3.8 Función sigmoidal.

Existen varios prototipos de funciones sigmoidales, entre los que se hallan la función logística, con valores de saturación 0 y 1:

$$f(\mu)=\frac{1}{1+e^{-\mu}}$$

la función tangente hiperbólica, con saturación -1 y 1:

$$f(\mu)=th(\mu)=\frac{1-e^{-2\mu}}{1+e^{-2\mu}}$$

y la función racional cuadrática, con valores de saturación 0 y 1:

$$f(\mu)=\frac{\mu^2}{1+\mu^2} \quad si\ \mu>0$$
$$f(\mu)=0 \quad si\ \mu\le 0$$

3.2.4 Arquitectura de las redes.

Existen diferentes *esquemas de interconexión* o *arquitecturas de redes*, cada una adaptada a resolver un tipo de problema, pero subsisten algunos principios básicos comunes a todos los casos: en una red neuronal típica los nodos o unidades de procesamiento se disponen, por lo general, en capas sucesivas, con una capa de entrada, una de salida y ninguna o varias capas intermedias, llamadas capas ocultas. Los principales aspectos arquitectónicos a estudiar incluyen los tipos de conexiones, y las arquitecturas básicas.

Conexiones

Existen dos tipos de conexiones en cuanto a la función que realizan sobre el nodo receptor, las conexiones excitadoras, y las conexiones inhibidoras:

- Las conexiones *excitadoras* tienden a incrementar el valor del estado de activación del nodo receptor, y normalmente se representan por señales positivas, esto es, conexiones con pesos positivos.
- Por el contrario, las conexiones *inhibidoras* tienden a disminuir el valor de activación o salida del nodo receptor, y se representan por señales negativas (conexiones con pesos negativos).

Por tanto, en toda conexión el valor absoluto del peso asociado indica la fuerza con que el nodo emisor influye en el estado de activación del nodo receptor, siendo el signo el que determina si esta influencia es excitadora o inhibidora. Estos tipos de conexión no tienen por qué ser excluyentes, es decir, un nodo puede tener conexiones tanto excitadoras como inhibidoras.

Atendiendo a su disposición en la red, podemos clasificar las conexiones en conexiones intercapa, conexiones intracapa, y conexiones recurrentes (figura 3.9):

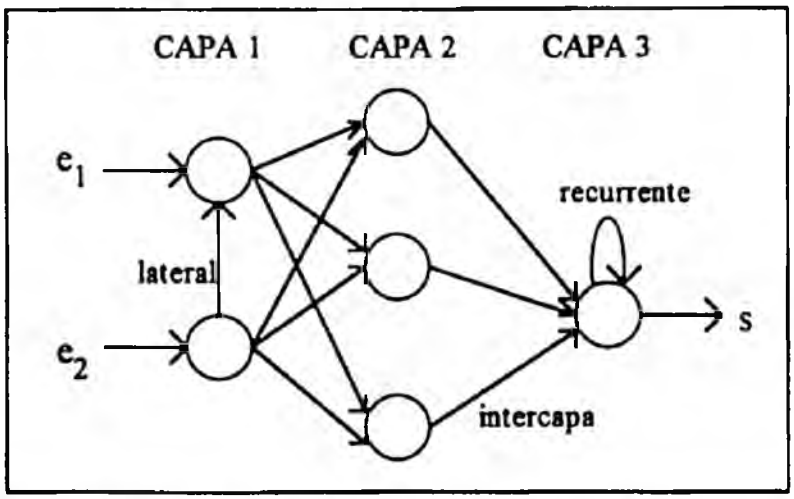

Figura 3.9: Tipos de conexiones según su disposición en la red.

- Las conexiones *intercapa*, también llamadas conexiones de capa, son las existentes entre nodos de diferentes capas.

- Las conexiones *intracapa*, también llamadas conexiones laterales, son aquellas que se establecen entre nodos de la misma capa.
- Las conexiones *recurrentes* son las conexiones de tipo lazo, es decir, las que conectan a un nodo consigo mismo.

Arquitecturas básicas

Las *arquitecturas de RNA* combinan los nodos, los tipos de conexiones y el flujo de información en configuraciones coherentes que proporcionan a las redes diferentes propiedades. Todas las arquitecturas de RNA poseen un aspecto estructural común: la disposición de las unidades de procesamiento en una o varias capas sucesivas. En general, una capa que recibe señales directamente del entorno se denomina capa de entrada, y una capa que emite señales al entorno se denomina capa de salida. Todas las capas de nodos que se hallen entre la capa de entrada y la de salida se llaman capas intermedias u ocultas, y no tienen contacto directo con el entorno.

Los tipos básicos de arquitecturas de RNA son las redes aciclicas o dirigidas hacia delante, las redes cíclicas o recurrentes y las redes simétricas:

- Las *redes aciclicas* o *dirigidas hacia delante* (figura 3.10) son generalmente de varias capas, y en ellas el flujo global de información se produce en un sólo sentido, desde la capa de entrada hacia la de salida, obteniéndose la respuesta en una sola pasada. Por tanto, no poseen conexiones intercapa que conduzcan la señal hacia capas precedentes, ni tampoco conexiones laterales o recurrentes.

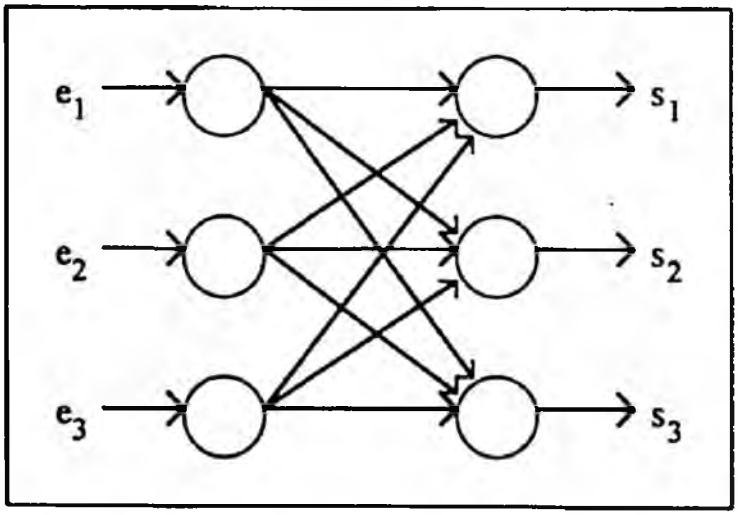

Figura 3.10: Red aciclica o dirigida hacia delante.

- Las *redes cíclicas* o *recurrentes* son generalmente de varias capas, y en ellas el flujo global de información se produce en ambos sentidos, desde la capa de entrada hacia las posteriores, y al contrario, desde la capa de salida hacia las anteriores a modo de realimentación. Por tanto, para que este efecto pueda producirse la red deberá poseer conexiones intercapa hacia atrás, laterales y/o recurrentes. La figura 3.11 muestra el esquema de una red cíclica o recurrente.

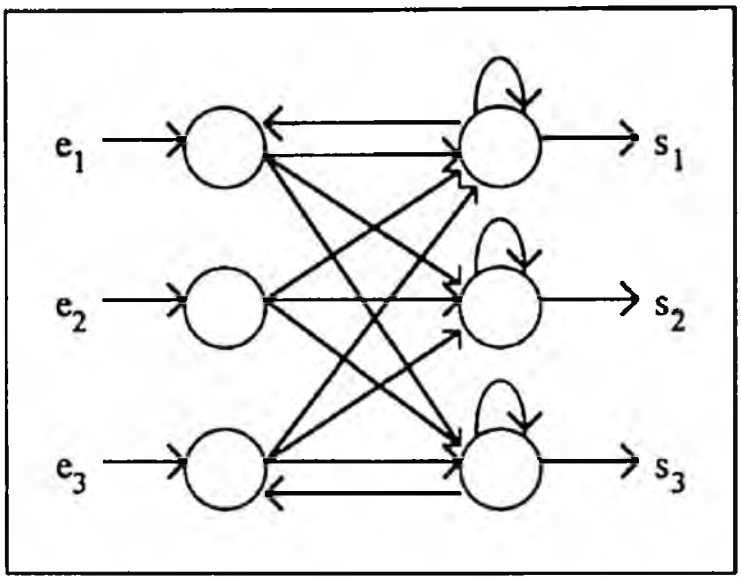

Figura 3.11: Red cíclica o recurrente.

- Las *redes simétricas* (figura 3.12) son un tipo especial de redes recurrentes de una sola capa en las que todas las conexiones son recurrentes (de lazo), o dobles y de sentido opuesto (simétricas).

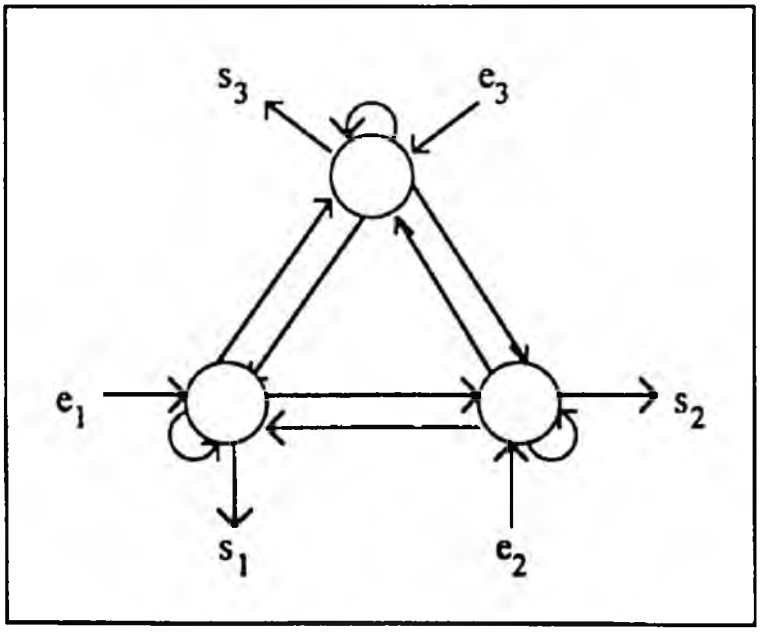

Figura 3.12: Red simétrica.

3.2.5 Procesamiento.

Definición de procesamiento

Los pesos de las conexiones de una RNA representan el conocimiento que la red posee del sistema, es decir, las relaciones existentes entre los patrones de entrada E_h y los patrones de salida respectivos S_h almacenados en la red, donde (E_h, S_h), para h = 1, ..., m, indica un par de patrones de entrada/salida asociados. El *procesamiento* o *recuperación* de una red neuronal puede definirse como el proceso de filtrado de un patrón de entrada a través de las conexiones de la red, para obtener el patrón de salida correspondiente. Matemáticamente, puede representarse mediante una función de recuperación, g, que obtiene un vector de salida S a partir de un vector de entrada E y de la matriz de pesos de la red W:

$$S = g(E, W)$$

Asociados al procesamiento de una RNA existen dos conceptos fundamentales: el modo de recuperación de los patrones de salida, y el sincronismo del procesamiento.

Modos de recuperación

La función de recuperación g de una red neuronal obtiene la respuesta S ante un estímulo E utilizando uno de los siguientes *modos de recuperación*: la recuperación hacia el patrón más próximo, y la recuperación interpolativa.

- La *recuperación hacia el patrón más próximo* consiste en encontrar el patrón de entrada E_h que se ajusta mejor al estímulo presentado E para responder con el correspondiente patrón de salida almacenado S_h:

  $$S_h = g(E, W)$$

  siendo

  $$d(E, E_h) = \min_{i=1}^{m} \left(d(E, E_i) \right)$$

  donde *d* es típicamente la función de distancia Euclídea, o la función de distancia de Hamming, y (E_h, S_h) es un par de patrones almacenados en la red.

- La *recuperación interpolativa* acepta un estímulo E y lo interpola a través del conjunto completo de patrones de entrada almacenados en la red $\{E_h\}$ para producir la correspondiente respuesta o salida S:

$$S = g(E, W)$$

donde $E_h \leq E \leq E_{h'}$, y $S_h \leq S \leq S_{h'}$, siendo (E_h, S_h) y $(E_{h'}, S_{h'})$ dos pares de patrones almacenados en la red.

Sincronismo del procesamiento

Otro aspecto importante a estudiar en un modelo conexionista es el *esquema temporal* o *sincronismo* con que se debe realizar el cálculo y actualización de los estados de activación de los nodos de la red. El sincronismo del procesamiento depende fundamentalmente de la arquitectura del modelo tratado, distinguiéndose dos clases: procesamiento en redes acíclicas, y procesamiento en redes recurrentes y simétricas.

El *procesamiento en redes acíclicas* o *dirigidas hacia delante* sigue un esquema muy sencillo, que consiste en actualizar el estado de activación de los nodos capa a capa, comenzando por la capa de entrada, siguiendo por las capas ocultas, y terminando por la capa de salida. Este esquema no plantea conflictos, ya que, al tener todas las conexiones el mismo sentido, el estado de cada nodo de una capa depende exclusivamente del estado de los nodos de la capa o capas anteriores.

En las *redes recurrentes y simétricas*, cada patrón de entrada se procesa de forma que los valores de activación de los nodos se actualizan continuamente hasta que la red alcanza un estado estable. Ahora bien, en estos modelos es necesaria la existencia de un mecanismo de control temporal que resuelva los conflictos provocados por las conexiones de realimentación a la hora de calcular el estado de activación de los nodos. Existen al respecto dos enfoques diferentes, el procesamiento síncrono y el procesamiento asíncrono:

- En algunos modelos, existe un temporizador central que cada cierto intervalo de tiempo provoca el cálculo de nuevos valores de activación para todos los nodos de forma simultánea. Este es un procedimiento *síncrono*, en el que, en general, el estado en el instante t+1 de un nodo depende del estado en el instante t de los nodos de los que recibe información.

- Sin embargo, en otros modelos las actualizaciones son *asíncronas* y al azar. En estos casos, se asume que en cada instante los nodos tienen una cierta probabilidad de evaluar y actualizar su estado de activación. Así, en un instante dado se calcula únicamente el estado de activación de un nodo escogido al azar. Esta operación se repite sucesivas veces, hasta que la red alcanza un estado en el que cualquier nuevo cálculo no implica un cambio de activación de ninguno de los nodos existentes. Este sistema evita las oscilaciones que en ocasiones pueden aparecer cuando se utiliza el procedimiento síncrono.

3.2.6 Aprendizaje.

Definición de aprendizaje

El *aprendizaje*, desde un punto de vista conexionista, se define como la modificación en el tiempo del patrón de interconexión de la red. En principio, esto puede suponer tres tipos de modificaciones:

- El desarrollo de nuevas conexiones.
- La pérdida de conexiones.
- La modificación de los pesos de las conexiones existentes.

Se ha realizado muy poco trabajo sobre los dos primeros aspectos. Sin embargo, ambos pueden considerarse casos particulares del tercero, ya que siempre que se cambie la fuerza de una conexión de cero a algún valor positivo o negativo, el efecto es el mismo que si se desarrollase una nueva conexión, y siempre que se cambie a cero el valor de un peso, tiene el mismo efecto que si se perdiese la conexión asociada. Por la razón anterior, el aprendizaje se define comúnmente como la modificación en el tiempo de los pesos asociados a las conexiones de la red. Matemáticamente esto supone que una red neuronal "aprende" en los instantes en que la derivada con respecto al tiempo de la función de modificación de pesos es distinta de cero.

Por tanto, el objetivo del aprendizaje consiste en la búsqueda del valor de los pesos de las conexiones que optimice la actuación deseada de la red. En general, esto puede interpretarse como el desplazamiento de la matriz de pesos W por un espacio de estados en la dirección que minimice o maximice la actuación de la red de acuerdo con algún criterio específico (función de coste, parámetros estadísticos, etc.).

Características del aprendizaje

Al describir los elementos característicos de todo modelo conexionista, se ha hecho referencia al aprendizaje o entrenamiento de la red y a su procesamiento como si fueran dos procesos independientes: en primer lugar se produce la fase de aprendizaje, y cuando ésta ha finalizado, se comienza la utilización normal de la red. Cuando esto sucede, se dice que el aprendizaje es un proceso *off-line*, y, aunque menos deseable, es el modo de aprendizaje más común de las redes neuronales artificiales. Por el contrario, si el aprendizaje y el procesamiento de la red se producen simultáneamente, entonces se dice que el aprendizaje es un proceso *on-line*.

Todas las técnicas de aprendizaje pueden agruparse en dos clases, aprendizaje supervisado y aprendizaje no supervisado:

- El *aprendizaje supervisado* es un proceso que incorpora un agente externo e información de carácter global, para decidir sobre aspectos tales como la duración y frecuencia de presentación de los patrones de entrada, el momento de finalización del aprendizaje, y, fundamentalmente, para proporcionar información sobre los errores cometidos en el proceso. En el aprendizaje supervisado la red ajusta sus pesos comparando cada patrón de salida obtenido con un patrón de salida objetivo proporcionado por el agente externo.
- El *aprendizaje no supervisado*, también llamado autoorganización, es un proceso que no incorpora un agente externo, y en él no se fuerzan los pesos a un ajuste entre la entrada y la salida, sino que el aprendizaje se produce gracias a la información de tipo local y al mecanismo interno de control que posee la red. El aprendizaje no supervisado permite que la red autoorganice su información interna y descubra sus propiedades emergentes de computación colectiva.

Reglas generales de aprendizaje

Prácticamente todos los modelos conexionistas poseen su propia regla de aprendizaje, supervisado o no supervisado, si bien la mayoría pueden considerarse variaciones de la regla de aprendizaje Hebbiano, sugerida por Donald Hebb en su libro *La Organización de la Conducta* [HEBB49]. Sin entrar en detalles acerca de sus características diferenciales, estas reglas se pueden agrupar en las siguientes clases generales:

- Aprendizaje por *corrección de error*.

- Aprendizaje por *correlación (Hebbiano).*
- Aprendizaje *estocástico.*
- Aprendizaje *competitivo.*
- Aprendizaje por *reforzamiento.*

3.2.7 Estabilidad y convergencia.

El procesamiento computacional de todo sistema conexionista está gobernado por dos conceptos matemáticos fundamentales: la estabilidad global del procesamiento, y la convergencia del aprendizaje.

Estabilidad del procesamiento

La *estabilidad global* de una RNA puede definirse como la característica que permite la estabilización en un tiempo finito de los estados de activación de los nodos de la red ante cualquier patrón de entrada. Es evidente que las redes acíclicas son siempre estables, por lo que la estabilidad global es un concepto generalmente asociado a los modelos recurrentes y simétricos. En consecuencia, una red recurrente será globalmente estable si ante un patrón de entrada cualquiera actualiza rápidamente su estado llevándolo a un punto fijo en el que cualquier nuevo cálculo no implica un cambio de activación de ninguno de los nodos. Estos puntos fijos o de equilibrio estable representan estados correspondientes a patrones almacenados en la red.

Existen tres teoremas fundamentales que utilizan el método directo de convergencia de Lyapunov para probar la estabilidad global de un amplio rango de modelos conexionistas recurrentes:

- El *teorema de Cohen-Grossberg* [COHE83], que se utiliza para demostrar la estabilidad global de redes simétricas no adaptativas de autoasociación de patrones.
- El *teorema de Cohen-Grossberg-Kosko*, que es una extensión del anterior realizada por Kosko [KOSK88] en el que se elimina la restricción de no aprendizaje, y se utiliza, por tanto, para demostrar la estabilidad global de redes adaptativas de autoasociación de patrones.

- El *teorema ABAM*, desarrollado igualmente por Kosko [KOSK88], que se utiliza para demostrar la estabilidad de redes adaptativas de heteroasociación de patrones, y representa una implementación del teorema general de Cohen-Grossberg-Kosko específica para el modelo de Memoria Asociativa Bidireccional Adaptativa (ABAM) de Kosko.

Convergencia del aprendizaje

El otro concepto matemático fundamental de la dinámica de un modelo conexionista es la *convergencia del aprendizaje*. Un sistema dinámico es convergente si la secuencia de estados X_i del mismo tiende hacia un límite X. En un sistema conexionista, la convergencia se refiere a la evolución de los pesos de las conexiones de la red durante el aprendizaje. Por tanto, la convergencia es un medio de analizar la habilidad de un procedimiento de aprendizaje para capturar apropiadamente la asociación entre el conjunto de patrones de entrada y el conjunto de patrones de salida deseados. Si esta asociación converge hacia un estado fijo, entonces el procedimento de aprendizaje es correcto, siendo la variación del error medido durante el proceso el que describe la bondad o calidad del algoritmo empleado.

3.2.8 Esquemas de representación.

El *esquema de representación* de una red neuronal hace referencia al significado de los patrones de entrada/salida dentro del entorno en el cual opera el sistema. Así, cada patrón puede representar una entidad o concepto diferente, o bien puede representar un conjunto de entidades. En este último caso, aparecen dos esquemas posibles de representación: la representación local y la representación distribuida.

Entorno de una red neuronal

En el desarrollo de cualquier modelo es crucial tener una idea clara del entorno en el que va a existir. En los modelos conexionistas el entorno se representa idealmente como una distribución de probabilidad que varía con el tiempo sobre el espacio de patrones de entrada. Es decir, se supone que en un instante dado, hay una cierta probabilidad de que cualquiera de los posibles patrones de entrada esté afectando a los nodos de entrada de la red. Esta función de probabilidad puede depender tanto de la historia de las entradas

como de la de las salidas del sistema. En la práctica, la mayor parte de los modelos conexionistas suponen una caracterización mucho más simple del entorno. Normalmente, este se define mediante una distribución estable de probabilidad sobre el conjunto de posibles patrones de entrada, independiente del tiempo, y, por tanto, de la historia del sistema. En este caso, el entorno puede definirse como el conjunto de probabilidades P_h, para h = 1, ..., m, donde m es el número de patrones posibles de entrada a la red.

Dado que cada patrón de entrada puede considerarse un vector, el conjunto de patrones de entrada con probabilidades distintas de cero constituye un conjunto de vectores en un espacio n-dimensional. Estos vectores pueden ser ortogonales, linealmente independientes, o arbitrarios. Ciertos modelos conexionistas están restringidos a los tipos de patrones que son capaces de aprender. Algunos son capaces de aprender a responder correctamente sólo si los vectores de entrada forman un conjunto ortogonal. otros si forman un conjunto de vectores linealmente independientes, y otros son capaces de aprender a responder a patrones de entrada esencialmente arbitrarios.

Patrones espaciales y espacio-temporales

Los patrones de entrada y de salida de una red pueden ser espaciales o espacio-temporales. Los *patrones espaciales* son estáticos o invariantes en el tiempo, y los *patrones espacio-temporales* son los formados por secuencias de patrones estáticos. Esta clasificación de los patrones de activación sugiere de forma inmediata cuatro tipos de funciones básicas implementables en los modelos conexionistas:

- Redes cuya funcionalidad sea reconocer un patrón de activación estático determinado.
- Redes cuya funcionalidad sea reconocer un patrón de activación espacio-temporal determinado.
- Redes cuya funcionalidad sea producir un patrón de salida estático determinado, ante la presencia de un patrón de entrada igualmente estático.
- Redes cuya funcionalidad sea producir un patrón de salida espacio-temporal determinado, ante la presencia de un patrón de entrada espacio-temporal.

Cada uno de estos mecanismos puede utilizarse en redes tanto adaptativas como no adaptativas para implementar con éxito diferentes tipos de aplicaciones.

Representaciones distribuidas

Dada una red neuronal artificial, formada por un conjunto de unidades simples de procesamiento o nodos, y un concepto general cuyas entidades particulares deben ser representadas, se denomina *representación local* a aquella que utiliza una unidad computacional o nodo de la red para representar cada entidad. Por el contrario, se denomina *representación distribuida* a aquella en la que cada entidad se representa mediante un patrón de actividad distribuido sobre varios nodos, y cada nodo participa en la representación de varias entidades diferentes.

3.3 RED DE HOPFIELD CONTINUA.

3.3.1 Introducción.

En un artículo publicado en 1982, John Hopfield realizó un importante estudio sobre una red recurrente simétrica de procesamiento asíncrono [HOPF82], que provocó un gran interés en la comunidad científica. Este modelo de red neuronal, denominado *autocorrelador discreto*, fue investigado inicialmente por Shun Ichi Amari, quién presentó el modelo y diversas variantes en 1972, y analizó la estabilidad del mismo usando un método denominado neurodinámica estadística [AMAR72] [AMAR74]. El trabajo de Hopfield es de particular importancia dado que introdujo un procedimiento de estabilidad alternativo, basado en las funciones de energía de Lyapunov. Hopfield mostró las propiedades de computación colectiva del autocorrelador discreto, utilizándolo para implementar un modelo de memoria autoasociativa. Posteriormente, diseñó un modelo evolucionado con valores de activación continuos en lugar de discretos [HOPF84], mostrándose posteriormente la capacidad del mismo para resolver problemas de optimización combinatoria [HOPF85], y de satisfacción de restricciones [TAGL87]. Gracias a Hopfield se popularizaron las características y capacidades del autocorrelador en la comunidad científica, reiniciando el interés en las redes neuronales artificiales recurrentes. Por esta razón, los autocorreladores discreto y continuo estudiados por Hopfield se denominan también *red de Hopfield discreta* y *red de Hopfield continua* respectivamente.

3.3.2 Red de Hopfield continua. Características.

Definición

La red de Hopfield continua (Continuous Hopfield, CH) es una red recurrente simétrica de una capa, que opera en tiempo continuo, y se caracteriza porque los pesos de las conexiones no se obtienen a partir de un procedimiento específico de aprendizaje (conexiones cableadas), si bien puede utilizarse para dicha tarea el aprendizaje por correlación. Desde el paradigma de la asociación de patrones, el modelo CH es un autoasociador hacia el patrón más próximo de patrones analógicos $X_h = (x_1^h, ..., x_n^h)$, para $h = 1, ..., m$.

El modelo CH, presentado por Hopfield en 1984 como una variante de la red discreta [HOPF84], supone una transición del autocorrelador discreto, un modelo que opera en tiempo discreto y procesa patrones bipolares/binarios, a un modelo que opera en tiempo continuo y procesa patrones cuyas componentes son valores definidos sobre el intervalo]0, 1[. Esta transición fue propuesta por Hopfield en un intento de aproximar aún más las unidades de procesamiento de la red a las neuronas biológicas, y abrió el camino a nuevas aplicaciones de las redes neuronales, siendo las más importantes la satisfacción de restricciones y la optimización combinatoria.

Estructura básica

La red de Hopfield continua posee una configuración de una sola capa de N nodos, en la que cada nodo se enlaza consigo mismo mediante una conexión recurrente, y con los restantes mediante conexiones laterales, es decir, se trata de una red simétrica típica (figura 3.13).

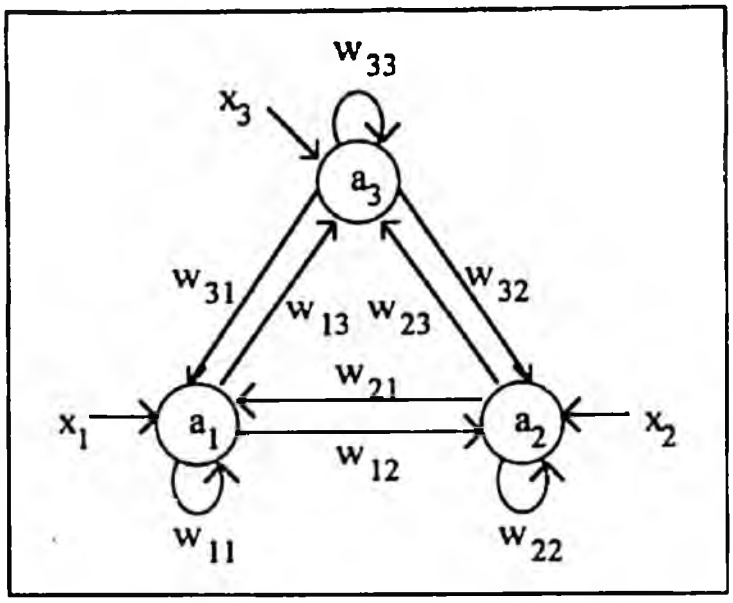

Figura 3.13: Red CH de tres nodos.

En este modelo los N nodos de la red corresponden a los N componentes analógicos con valores en el intervalo]0, 1[de los patrones de entrada X_h. Cada conexión entre dos nodos de la red posee un peso asociado w_{ij}. Este peso es un valor real considerado intuitivamente la fuerza de la conexión. En el modelo CH, al igual que en el discreto, los pesos de las conexiones laterales simétricas son iguales, es decir, $w_{ij} = w_{ji}$, y generalmente los pesos de las conexiones recurrentes son nulos, esto es, $w_{ii} = 0$. Opcionalmente, cada nodo j de la red puede poseer una señal de entrada adicional de valor constante, I_j, denominada tendencia (bias) del nodo.

Procesamiento

La red de Hopfield continua utiliza un mecanismo de procesamiento recurrente de tipo síncrono, en el que una entrada X se procesa hasta que la red alcanza un estado estable (los nodos cesan de cambiar). El procesamiento síncrono consiste en actualizar simultáneamente el estado de activación de todos los nodos de la red, repitiendo esta operación sucesivas veces hasta que la red alcance un estado en el que cualquier nuevo cálculo no implique un cambio de estado significativo de ninguno de los nodos existentes. Sin embargo, la actualización síncrona de los nodos de la red CH puede llegar a provocar oscilaciones infinitas en el procesamiento. Este tipo de procesamiento se debe fundamentalmente a las características de la implementación del modelo realizada originalmente por Hopfield [HOPF84].

Hopfield construyó el modelo utilizando un circuito electrónico analógico con amplificadores no lineales y resistencias (figura 3.14). Cada amplificador posee una resistencia de entrada r y una capacidad de entrada C, así como una señal externa I_i que suministra una corriente constante.

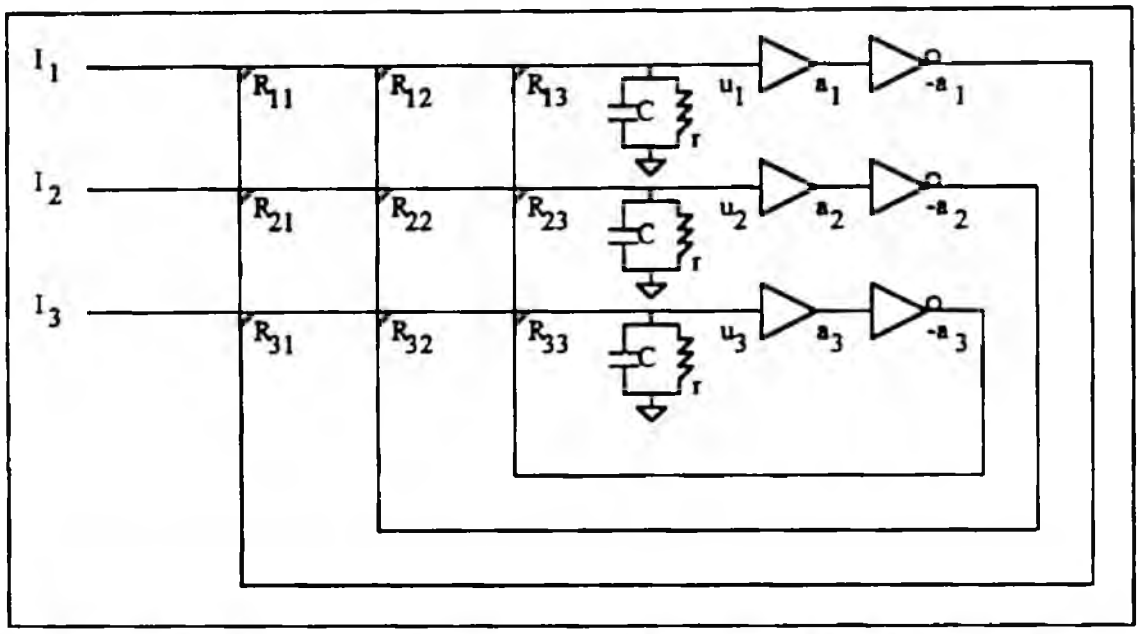

Figura 3.14: Circuito electrónico correspondiente a una red CH de tres nodos.

La corriente de entrada neta en cada amplificador no lineal es la suma de las contribuciones individuales de corriente de las otras unidades, más la corriente de la entrada externa, menos las pérdidas producidas a través de la resistencia de entrada r.

El amplificador no lineal implementa la función sigmoidal tangente hiperbólica, cuya salida es un valor en el intervalo]0, 1[:

$$a_j = f(u_j) = \frac{1}{2} + \frac{1}{2}\text{th}(\lambda u_j)$$

Las resistencias colocadas en los puntos de intersección de las líneas corresponden a los pesos de las conexiones de la red, y su valor viene determinado por la expresión:

$$R_{ij} = \frac{1}{|w_{ij}|}$$

Dado que los valores reales de todas las resistencias R_{ij} son positivos, se utilizan amplificadores inversores para simular las conexiones inhibitorias.

El circuito anterior es un transitorio RC, por lo que es posible hallar el valor de cada u_j a partir de la ecuación diferencial que describe la carga del condensador como resultado de la entrada de la corriente neta:

$$C\frac{du_j}{dt} = \sum_{i=1}^{N} w_{ij}a_i - \frac{u_j}{R_j} + I_j$$

donde $\frac{1}{R_j} = \frac{1}{r} + \sum_{i=1}^{N}\frac{1}{R_{ij}}$ es la combinación en paralelo de la resistencia de entrada r y de las resistencias R_{ij} de la matriz de pesos.

Las N ecuaciones diferenciales del circuito anterior describen por completo la evolución en el tiempo del sistema. No obstante, es posible simular por computador la red de Hopfield continua empleando el siguiente algoritmo de actualización de los estados de los nodos:

Algoritmo. El algoritmo de actualización de los nodos de la red CH puede describirse de la siguiente forma:

1. Inicialmente, los valores de activación de los nodos, a_i, corresponden a los componentes x_i del patrón de entrada X.

2. A continuación, se realiza una iteración de procesamiento síncrono. Cada nodo j de la red actualiza su estado de activación mediante la ecuación:

$$a_j(t+\Delta t) = a_j(t) + \left(f(\mu_j) - a_j(t)\right) \cdot \Delta t$$

donde $a_j(t + \Delta t)$ es el nuevo estado de activación del nodo j, Δt es el incremento temporal de la simulación, f es la función de umbral sigmoidal logística:

$$f(\mu_j) = \frac{1}{1+e^{-\mu_j}}$$

y μ_j es la entrada neta del nodo j, obtenida aplicando la regla de combinación aditiva:

$$\mu_j = \sum_{i=1}^{N} w_{ij}a_i(t) + I_j$$

Este cálculo se realiza síncronamente por iteración para todos los nodos de la red.

3. Si al menos un nodo ha cambiado de estado significativamente en la última iteración se repite el paso 2. En caso contrario, la red se considera estabilizada y se finaliza el algoritmo.

Hopfield demostró la estabilidad de la red CH bajo la hipótesis de que la matriz de pesos era simétrica y contenía ceros en su diagonal principal, utilizando para ello una función de energía de Lyapunov. Así, cada estado de la red de Hopfield continua posee un valor de energía asociado, definido por la función:

$$E(A) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}a_i a_j + \frac{1}{\lambda}\sum_{i=1}^{N}\frac{1}{R_i}\int_0^{a_i} f^{-1}(a)da - \sum_{i=i}^{N} I_i a_i$$

donde $A = (a_1, ..., a_n)$ es el vector de estados de los nodos de la red, y $f^{-1}(a)$ es la inversa de la función f de amplificación sigmoidal. En esta ecuación, la influencia del término

$$\frac{1}{\lambda}\sum_{i=1}^{N}\frac{1}{R_i}\int_0^{a_i} f^{-1}(a)da$$

queda determinada por el valor de la ganancia λ de la función f de amplificación sigmoidal. Suponiendo que el factor de ganancia λ es elevado (la sigmoide se aproxima a una función de salto binario), el término anterior es prácticamente nulo, por lo que puede eliminarse de la función de energía sin pérdida de generalidad, quedando:

$$E(A) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}a_i a_j - \sum_{i=1}^{N} I_i a_i$$

El siguiente teorema garantiza la estabilidad de la red de Hopfield continua bajo el mecanismo de procesamiento antes descrito.

Teorema. Dada un red de Hopfield continua de N nodos con función de energía

$$E(A) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}a_i a_j - \sum_{i=1}^{N} I_i a_i$$

se verifica que E es una función de Lyapunov, y el sistema es globalmente estable.

Demostración:

Se trata de comprobar que todo cambio en el estado de un nodo a_j durante el procesamiento de la red da lugar a una disminución de energía, y que dicha disminución de energía posee límite. En efecto, E es una función acotada que posee como cota inferior

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}|w_{ij}| - \sum_{i=1}^{N}|I_i|$$

Para demostrar que el sistema es estable basta verificar que la función de energía es monótona decreciente. En efecto, supóngase que el nodo a actualizar en el siguiente paso del procesamiento es el nodo j. La parte de la energía de la red afectada por el nodo j viene dada por la expresión:

$$E(a_j) = -a_j\left[\sum_{i=1}^{N} w_{ij}a_i + I_j\right]$$

Si el nodo j cambia de estado entonces el incremento de energía producido es:

$$\Delta E = -\Delta a_j\left[\sum_{i=1}^{N} w_{ij}a_i + I_j\right]$$

donde Δa_j es igual a la diferencia entre el nuevo estado del nodo j y el estado anterior. De acuerdo con el esquema de procesamiento Δa_j y el factor entre corchetes son ambos positivos o ambos negativos, por lo que cualquier cambio en la función de energía es siempre negativo, es decir, la función de energía de la red es monótona decreciente. Por tanto, la iteración del algoritmo tiende siempre a estabilizar la red, como se quería demostrar.

3.3.3 Red de Hopfield continua y optimización.

El inicio de la investigación en el campo de las redes neuronales y la optimización combinatoria fue realizado por John Hopfield con el estudio del comportamiento dinámico de la red de Hopfield continua [HOPF84] y su aplicación al problema clásico del viajante comercial [HOPF85].

El fundamento de la aplicación de la red de Hopfield continua al campo de optimización combinatoria está en el paradigma de minimización de energía. Este paradigma interpreta a la red neuronal como un sistema termodinámico cuyos estados de energía mínima representan soluciones a determinados problemas de optimización. Por tanto, el objetivo consiste en definir un esquema de interconexión y un modo de procesamiento de la red que permita resolver el problema tratado a través de un proceso de *relajación paralela*, en el que el estado de la red se desplaza hacia un estado estable de energía mínima.

Para utilizar la red CH en esta aplicación es necesario realizar las siguientes tareas:

- En primer lugar, se debe establecer una disposición arquitectónica de la red que identifique de forma específica los nodos con las variables del problema de optimización.
- Después, se debe transformar la función objetivo y las restricciones del problema en una función de energía apropiada para la red CH, de manera que al minimizarse esta última se encuentre la solución del problema de optimización inicial.
- Por último, hay que determinar los parámetros asociados a la función de energía, para así obtener los valores de los pesos de las conexiones de cada nodo. Estos valores deberán representar adecuadamente la instancia del problema específico a resolver.

De esta manera, a partir de un patrón de entrada que representa la situación inicial del sistema a optimizar, y a través del proceso de minimización de energía que caracteriza al modelo, la red construida se estabiliza en un estado solución del problema de optimización a resolver.

Hopfield y Tank [HOPF85] proporcionaron la siguiente implementación del problema del viajante utilizando el modelo continuo de Hopfield:

- La red que representa el problema del viajante posee una arquitectura de n^2 nodos, siendo n el número de ciudades a recorrer. La disposición de los nodos en forma de matriz cuadrada de orden n puede ayudar a interpretar el significado de cada nodo en relación con la solución del problema. Las n filas de la matriz representan las variables del problema, es decir, las posiciones del recorrido, y las n columnas los valores del dominio de las variables, esto es, las ciudades a visitar. Así, si un nodo situado en la posicion (i, j) de la red-matriz está activo, significa que en la posicion i del recorrido se debe visitar la ciudad j.
- La función de energía de la red que implementa los objetivos y restricciones del problema viene dada por la siguiente expresión:

$$E = \frac{A}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq j}}^{n} a_{ij}a_{ik} + \frac{B}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq j}}^{n} a_{ji}a_{ki} +$$

$$\frac{C}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} - n^2\right) + \frac{D}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} d_{ij}a_{ki}\left(a_{(k+1)j} + a_{(k-1)j}\right)$$

donde a_{ij} es el valor de activación del nodo situado en la fila i columna j, y A, B, C y D son parámetros que deben ajustarse convenientemente para que la actuación de la red sea correcta. Cada término de la función de energía representa una restricción o un objetivo del problema a resolver. El primer término se hace mínimo sólo si hay un nodo activo en cada fila de la matriz, lo que implica que cada variable toma un valor único (no hay dos ciudades en la misma posición del recorrido). El segundo término se hace mínimo sólo si hay un nodo activo en cada columna de la matriz, lo que implica que todas las variables toman valores distintos (ninguna ciudad se visita dos veces). El tercer término se hace mínimo si existen exactamente n nodos activos en la red, es decir, hay n ciudades en el itinerario. El cuarto término representa el objetivo de la optimización, la suma de las distancias entre las ciudades del recorrido o distancia total a minimizar.

- Los pesos de las conexiones de la red se establecen a partir de la función de energía anterior de la siguiente forma:

$$w_{xi,yj} = -A\delta_{xy}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{xy}) - C - Dd_{ij}(\delta_{y(x+1)} + \delta_{y(x-1)})$$

donde δ es la función delta de Kroenecker, definida:

$$\delta_{ij} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

Finalmente, la tendencia global externa I para todos los nodos es:

$$I = Cn$$

Los pesos de las conexiones determinados de acuerdo con los criterios anteriores permanecen invariables a lo largo del procedimiento de optimización. El estado inicial de la red se establece dando a los nodos valores de activación $a_j = 0.5 + \varepsilon$, donde ε es un valor aleatorio (p.e., $\varepsilon \in [-0.1, 0.1]$), utilizado para romper la simetría de los estados de activación e impedir la oscilación de la red. A continuación la red se procesa síncronamente hasta que se alcance un estado final estable. Durante el procesamiento los nodos se van actualizando simultáneamente de manera continua, estableciéndose entre ellos un proceso paralelo de optimización por competición, que termina cuando los K nodos ganadores de la competición quedan activos (con valores próximos a 1), mientras los restantes nodos quedan inactivos (con valores próximos a 0). Si la convergencia del procesamiento ha tenido éxito, entonces el estado final de la red corresponderá a la solución del problema, es decir, el recorrido que minimiza la distancia total.

Uno de los problemas de esta implementación es que el ajuste de los parámetros A, B, C y D de la función de energía debe realizarse de forma empírica. Así, Hopfield proporciona los siguientes valores para un recorrido de 10 ciudades: A = B = 500, C = 200, y D = 500. Sin embargo, estos valores no tienen por qué ser válidos para rutas de otros tamaños.

3.3.4 Análisis de la red de Hopfield continua.

La red de Hopfield continua y su utilidad en la resolución de problemas de búsqueda heurística han sido objeto de numerosos estudios en los últimos años [TAGL87] [HEDG88] [KAHN89] [AIYE90], siendo uno de los más completos el realizado por Wilson y Pawley acerca de la estabilidad del problema del viajante en la red CH [WILS88].

La mayor ventaja del modelo continuo de Hopfield es su arquitectura inherentemente paralela, fácilmente implementable en hardware mediante circuitos electrónicos analógicos. Además, las características de la relajación paralela que produce su esquema de procesamiento recurrente hacen que este modelo sea muy apropiado para la implementación de problemas combinatorios de restricciones. De hecho, estas características pueden extenderse al modelo discreto de Hopfield [HOPF82], por lo que las redes de Hopfield suelen denominarse comúnmente *redes neuronales de satisfacción de restricciones*. Sin embargo, la eficiencia del modelo continuo en la resolución de problemas de optimización combinatoria es bastante pobre, presentando las siguientes limitaciones importantes:

- Uno de los inconvenientes de la implementación de los problemas de optimización en la red de Hopfield continua, de acuerdo con los criterios indicados por Hopfield y Tank, es que el ajuste de los parámetros de la función de energía que representa los objetivos y restricciones del problema debe realizarse de forma empírica. De esta manera, si se da un valor excesivo a los parámetros que implementan los objetivos de la optimización en detrimento de los parámetros correspondientes a las restricciones del problema, puede hacer que la red se estabilice en un estado correspondiente a una solución no válida. Recíprocamente, si se da mucha más importancia a los parámetros asociados a las restricciones, frente a los parámetros asociados a los objetivos de optimización, la solución obtenida será válida, pero de muy baja calidad. En cualquier caso, es necesario determinar un procedimiento general de diseño que permita fijar los

parámetros de la función de energía, de forma que se garantice la obtención de una solución final válida, requisito fundamental de toda técnica de optimización.

- Por otra parte, el procesamiento de la red CH se caracteriza por una competición entre los nodos que la componen: inicialmente todos los nodos toman valores próximos a 0.5 y su estado se va actualizando síncronamente de manera continua hasta que los K nodos ganadores de la competición quedan activos, mientras los restantes nodos quedan inactivos. Con respecto a la optimización, este proceso competitivo es tanto menos eficiente cuantos más nodos posea la red, es decir, cuanto mayor sea el tamaño del problema implementado, y también cuanto mayor sea la rugosidad de la función de energía de la red que representa los objetivos y restricciones del problema. La rugosidad de la función de energía hace referencia a la diferencia media de energía existente entre dos estados próximos de la red (esto es, dos estados que se diferencien en el valor de activación de un solo nodo). Lógicamente, la rugosidad depende de dos factores, el valor que adopten los parámetros de la función de energía, y, fundamentalmente, las características del problema a implementar. Así, el comportamiento de la red de Hopfield continua en cuanto a la calidad media de la solución obtenida varía según el tipo de problema implementado.

- Además, el sincronismo del procesamiento de la red, generador del proceso competitivo antes descrito, requiere que los estados iniciales de los nodos sean valores próximos a 0.5, pero con una variación aleatoria ε, que rompa la simetría del procesamiento y evite una oscilación infinita de la red. El valor de ε debe elegirse con cuidado, ya que si es muy grande, anula el proceso competitivo y con ello el mecanismo de optimización, pero si es demasiado pequeño la red puede llegar a no estabilizarse nunca. El valor apropiado de ε depende nuevamente de la rugosidad de la función de energía de la red, por lo que debe encontrarse de forma empírica.

- El modo de procesamiento de la red de Hopfield continua, al igual que en el modelo discreto, implementa un proceso de minimización en sentido monótono decreciente que termina cuando la red alcanza un estado estable mínimo local de la función de energía. Por esta razón, aún cuando el ajuste de los parámetros de la red sea adecuado, el tamaño del problema sea pequeño, y la función de energía presente una variación suave, las soluciones obtenidas mediante el modelo CH son óptimos locales del problema, lo que no es muy deseable en problemas de optimización global.

- Finalmente, la calidad de la solución obtenida por la red de Hopfield continua depende directamente del estado inicial de la red, el cual se elige de forma aleatoria. Por esta razón, la desviación típica que presentan las soluciones obtenidas mediante la red de Hopfield con respecto a la media es bastante grande. Además el carácter local de la búsqueda implementada hace que la calidad media de las soluciones con respecto a la óptima empeore a medida que crece el tamaño del problema.

Una solución inmediata y sencilla a estos inconvenientes, aunque no demasiado eficiente, podría ser aplicar el criterio de la resolución iterativa, inicializando la red sucesivas veces de forma aleatoria y procesándola hasta obtener un mínimo local, eligiendo al final el mejor resultado obtenido. Una mejor solución consiste en añadir un factor de ruido estocástico a la función de cálculo de los estados de los nodos, que permita escalar los mínimos locales y posibilite alcanzar estados finales globalmente óptimos. Este ruido tendría un comportamiento decreciente, para evitar que, una vez que la red se haya ido aproximando al óptimo global, se produzcan oscilaciones sin fin alrededor del mismo. Esta técnica basada en los estudios de Kirkpatrick, Gelatt y Vecchi [KIRK83], se conoce como *temple simulado*, y ha sido aplicada en otros modelos conexionistas para la resolución de problemas de optimización combinatoria.

3.4 MÁQUINA DE BOLTZMANN.

3.4.1 Introducción.

Todos los modelos conexionistas poseen una característica en común: bien durante el aprendizaje, bien durante el procesamiento, utilizan un algoritmo que trata de minimizar una función. En el primer caso, esta función representa una medida de error utilizada en la búsqueda del valor de los pesos de las conexiones que optimiza la actuación deseada de la red. En el segundo caso, esta función supone una medida de la energía del sistema utilizada en la búsqueda del estado estable más próximo al estado inicial de la red.

La mayoría de los algoritmos empleados para minimizar las funciones de error o de energía son de carácter local. Esencialmente, se trata de algoritmos que implementan diversas versiones de la técnica heurística de optimización combinatoria denominada

escalada (hill-climbing). El algoritmo de la escalada sólo permite transiciones de estado que generan una mejora en la calidad de las solución, por lo que, si la función a optimizar no es convexa, la solución final obtenida será óptimo local, pero no necesariamente global, de dicha función.

Aproximadamente al mismo tiempo que Hopfield mostraba las propiedades de computación colectiva de su modelo discreto [HOPF82], Kirkpatrick, Gelatt y Vecchi, investigadores de IBM, introdujeron una nueva técnica de búsqueda heurística para resolver problemas de optimización combinatoria, denominada *temple simulado* [KIRK83]. El algoritmo del temple simulado tiene como base el algoritmo de la escalada, pero permite efectuar estocásticamente transiciones de estado que suponen una pérdida de calidad de la solución en curso, con el fin de escapar de los mínimos locales, y favorecer la obtención de una soluelón final mejor. El algoritmo del temple simulado fue utilizado por Hinton y Sejnowski como fundamento del diseño de la máquina de Boltzmann [HINT86].

3.4.2 Máquina de Boltmann. Características.

Definición

La máquina de Boltzmann (Boltzmann Machine, BM) introducida por Hinton y Sejnowski [HINT86] es una red recurrente, que opera en tiempo discreto, y emplea un procedimiento supervisado off-line de aprendizaje estocástico, basado en la técnica del temple simulado. Desde el paradigma de la asociación de patrones, el modelo BM es un asociador hacia el patrón más próximo de patrones binarios $X_h = (x_1^h, ..., x_n^h)$, para h = 1, ..., m.

Estructura básica

Existen diferentes arquitecturas para la organización de los nodos en la máquina de Boltzmann, siendo la más importante la denominada *red de completitud de Boltzmann*, bajo la cual el modelo se comporta como un autoasociador de patrones (memoria autoasociativa). La red de completitud de Boltzmann posee una configuración de dos capas de nodos, capa visible y capa oculta.

La capa visible está formada por los nodos correspondientes a los p componentes binarios (0, 1) de los patrones X_h a almacenar, y la capa oculta contiene un conjunto de q nodos cuya finalidad es la de soportar conexiones cuyos pesos representan interacciones complejas que no pueden expresarse con simples correlaciones directas entre los componentes de los patrones. La red está completamente interconectada, es decir, cada uno de los N = p + q nodos se conecta con todos los de su misma capa y con todos los de la otra capa (figura 3.15).

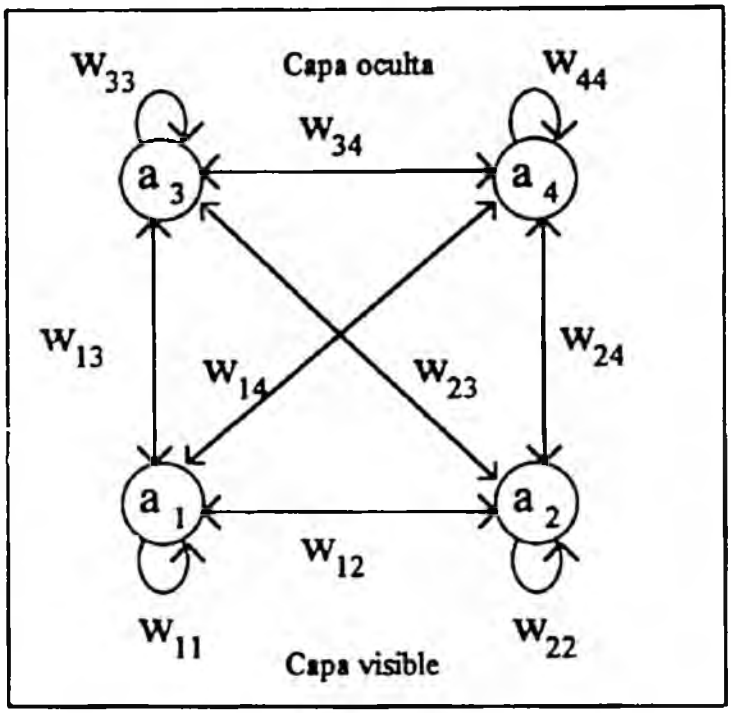

Figura 3.15: Arquitectura de completitud de dos capas para la máquina de Boltzmann.

Cada conexión entre dos nodos de la red posee un peso asociado. Este peso es un valor real considerado intuitivamente la fuerza de la conexión. Generalmente, se denota w_{ij} al peso de la conexión que parte del nodo i hacia el nodo j de la red. En el modelo BM, los pesos de las conexiones son simétricos, es decir, $w_{ij} = w_{ji}$, y generalmente, aunque no de forma necesaria, los pesos de las conexiones recurrentes son nulos, esto es, $w_{ii} = 0$. Opcionalmente, cada nodo j de la red puede poseer una señal de entrada adicional de valor constante, I_j, denominada tendencia (bias) del nodo.

Procesamiento

La máquina de Boltzmann utiliza un mecanismo de procesamiento recurrente de tipo asíncrono, en el que una entrada X se procesa hasta que la red alcanza un estado estable.

Sin embargo, en este modelo la salida de los nodos individuales es una función estocástica de las entradas, es decir, la salida de un nodo cualquiera se calcula de acuerdo con una distribución de probabilidad, en lugar de utilizar una función de umbral determinista.

Algoritmo. El algoritmo de actualización de los nodos de la red BM puede describirse de la siguiente forma:

1. Inicialmente, los valores de activación de los p nodos de la capa visible, a_i, corresponden a los componentes binarios x_i del patrón de entrada X, mientras que los q nodos de la capa oculta se inicializan con valores binarios aleatorios.

2. Se inicializa la temperatura o parámetro de control $T = T_0$.

3. A continuación, se realiza una iteración de procesamiento asíncrono. Se elige al azar un nodo j de la red y se calcula su estado de activación. Independientemente del valor que tenga dicho nodo se le asigna el valor $a_j = 1$ con una probabilidad que viene determinada por la distribución de Boltzmann:

$$\frac{1}{1+e^{-\mu_j/T}} > \text{random}[0,1[$$

En la expresión anterior T es la temperatura actual del sistema, y μ_j es la entrada neta del nodo j, calculada utilizando la regla de combinación aditiva:

$$\mu_j = \sum_{i=1}^{N} w_{ij} a_i + I_j$$

donde a_i es el estado del i-ésimo nodo antecesor al nodo j, w_{ij} es el peso de la conexión entre ambos nodos, e I_j es la tendencia del nodo j. Este cálculo se realiza tantas veces por iteración como sea necesario hasta conseguir el equilibrio térmico, teniendo en cuenta que los nodos no se eligen siguiendo un determinado orden, sino al azar. El número de actualizaciones de nodos por iteración depende del programa de templado utilizado, y será como mínimo igual al número de nodos existentes en la red.

4. Si al menos un nodo ha cambiado de estado en la última iteración se decrementa la temperatura T y se repite el paso 3. En caso contrario, la red se considera estabilizada y se finaliza el algoritmo.

En la práctica, la temperatura inicial T_0 puede determinarse mediante el siguiente algoritmo simple, similar al del temple simulado:

Algoritmo. Cálculo de la temperatura inicial en la máquina de Boltzmann.

1 Inicializar T_0 con un valor positivo (por ejemplo, T_0, puede tomar el valor absoluto de la energía de un estado aleatorio de la red).

2. Partiendo de un estado aleatorio de la red, realizar un número reducido de transiciones que supongan un incremento de energía (por ejemplo, 100), aplicando el criterio de aceptación.

3. Si el numero de transiciones aceptadas es inferior al total de transiciones efectuadas, entonces incrementar T_0 multiplicándolo por un factor mayor que la unidad y volver al paso 2. El algoritmo termina cuando todas las transiciones realizadas con pérdida de calidad se aceptan, lo que indica que la temperatura ya es suficientemente elevada.

El algoritmo de procesamiento de la red de Boltzmann posee un comportamiento estocástico variable de acuerdo con la temperatura T. Así, para temperaturas elevadas, la función sigmoidal de Boltzmann toma valores próximos a 0.5, al ser el término $-\mu_j / T$ muy pequeño en valor absoluto. En esta situación, todos los nodos toman valores binarios al azar, es decir, se aceptan transiciones de estado que representan tanto incrementos como decrementos de la energía del sistema. Sin embargo, para valores de la temperatura pequeños, la función sigmoidal toma valores próximos a 0 y 1, según sea negativo o positivo el valor de la entrada neta μ_j, con lo que el sistema se comporta prácticamente de forma determinista.

En el caso de la máquina de Boltzmann, cada estado de la red posee un valor de energía asociado, definido por la función de Lyapunov:

$$E(A) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} a_i a_j - \sum_{i=1}^{N} I_i a_i$$

donde $A = (a_1, ..., a_n)$ es el vector de estados de los nodos de la red.

El siguiente teorema, similar al referente a la red de Hopfield continua, garantiza la estabilidad de la máquina de Boltzmann con el mecanismo de procesamiento antes

descrito, y bajo la hipótesis de que todas las conexiones simétricas poseen pesos iguales, siendo nulos los pesos de las conexiones de lazo.

Teorema. Dada un máquina de Boltzmann de N nodos con función de energía

$$E(A) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}a_i a_j - \sum_{i=1}^{N} I_i a_i$$

se verifica que E es una función de Lyapunov, y el sistema es globalmente estable.

Demostración:

Se trata de comprobar que la función de energía de la red posee límite finito. Para ello hay que verificar que la función de energía está acotada y que es monotóna decreciente, al menos a partir de un cierto instante del procesamiento.

En efecto, una cota inferior de la función de energía viene dada por la expresión:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} |w_{ij}| - \sum_{i=1}^{N} |I_i|$$

Además, transcurrido un determinado número de iteraciones de procesamiento, cuando la temperatura T es suficientemente baja, el comportamiento de la red se vuelve determinista. A partir de entonces, cada nodo j que actualiza su estado, genera un incremento de energía igual a:

$$\Delta E = -\Delta a_j \left[\sum_{i=1}^{N} w_{ij}a_i + I_j \right]$$

donde Δa_j es igual a la diferencia entre el nuevo estado del nodo j y el estado anterior. De acuerdo con el esquema de procesamiento, Δa_j y el factor entre corchetes, igual a la entrada neta del nodo, son ambos positivos o ambos negativos, por lo que cualquier cambio en la función de energía es siempre negativo, es decir, la función de energía de la red, para una temperatura T próxima a cero, es monótona decreciente, como se quería demostrar.

Una observación importante acerca del procesamiento de la máquina de Boltzmann es que el incremento de energía producido por el cambio de estado de un nodo j, es igual en valor absoluto a la entrada neta μ_j del nodo, valor utilizado en el propio cálculo del estado de activación. Por tanto, el comportamiento estocástico de este modo de

procesamiento viene determinado por la razón entre el incremento de energía y la temperatura del sistema, como ocurre en el algoritmo de optimización del temple simulado. En conclusión, puede decirse que el procesamiento de la máquina de Boltzmann implementa el algoritmo del temple simulado en la búsqueda del estado de mínima energía de la red.

3.4.3 Máquina de Boltzmann y optimización.

Una aplicación en la que sólo tiene interés el modo de procesamiento de la red BM, sin tener en cuenta su algoritmo de aprendizaje, es la optimización combinatoria. La minimización de la función de energía a la que conduce el procesamiento de la máquina de Boltzmann hace pensar en su posible aplicación a problemas de optimización combinatoria, tanto más teniendo en cuenta que el mecanismo que fundamenta la máquina de Boltzmann, el temple simulado, permite superar el comportamiento de optimización local propio de otras redes como la red de Hopfield continua.

Al igual que en otros modelos conexionistas, para utilizar la red BM en la resolución de un problema de optimización combinatoria es necesario realizar las siguientes tareas:

- En primer lugar, se deben identificar de forma específica los nodos de la red con las variables del problema de optimización.
- Después, se debe transformar la función objetivo y las restricciones del problema en una función de energía apropiada para la máquina de Boltzmann, de forma que al minimizarse esta última se encuentre la solución del problema de optimización inicial.
- Por último, hay que determinar los parámetros asociados a la función de energía, para así obtener los valores de los pesos de las conexiones de cada nodo.

De esta forma, es posible realizar con la máquina de Boltzmann la siguiente implementación del problema del viajante, equivalente a la efectuada por Hopfield y Tank [HOPF85] mediante la red de Hopfield continua, y que se introdujo antes:

- La red BM que representa el problema del viajante posee una arquitectura sin nodos ocultos y con n^2 nodos visibles, dispuestos en forma de matriz cuadrada de orden n, siendo n el número de ciudades a recorrer. Las n filas de la matriz representan las variables del problema, es decir, las posiciones del recorrido, y las n columnas los valores del dominio de las variables, esto es, las ciudades a visitar.

Así, si un nodo situado en la posicion (i, j) de la red-matriz está activo significa que en la posicion i del recorrido se debe visitar la ciudad j.

- La función de energía de la red que implementa los objetivos y restricciones del problema es la misma que la presentada para el modelo de Hopfield, y cada término posee el significado correspondiente, asociado a una restricción o a un objetivo del problema:

$$E = \frac{A}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq j}}^{n} a_{ij}a_{ik} + \frac{B}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq j}}^{n} a_{ji}a_{ki} +$$

$$\frac{C}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} - n^2\right) + \frac{D}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} d_{ij}a_{ki}\left(a_{(k+1)j} + a_{(k-1)j}\right)$$

- Los pesos de las conexiones de la red se establecen a partir de la función de energía anterior de la siguiente forma:

$$w_{xi,yj} = -A\delta_{xy}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{xy}) - C - Dd_{ij}(\delta_{y(x+1)} + \delta_{y(x-1)})$$

donde δ es la función delta de Kroenecker, definida:

$$\delta_{ij} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

Finalmente, la tendencia global externa I para todos los nodos es:

$$I = Cn$$

El estado inicial de la red se establece dando a los nodos valores de activación binarios al azar. A continuación la red se procesa de acuerdo con el algoritmo de Boltzmann hasta que se alcance un estado final estable. Si la convergencia del procesamiento ha tenido éxito, entonces el estado final de la red deberá corresponder a la solución del problema, es decir, el recorrido que minimiza la distancia total.

3.4.4 Análisis de la máquina de Boltzmann.

Una característica importante del funcionamiento de la máquina de Boltzmann es su tremenda lentitud. Los estudios de Geman y Geman [GEMA84] acerca del temple simulado muestran que, para conseguir el óptimo global tanto en el procesamiento como en el aprendizaje, la temperatura debe reducirse de forma proporcional a la inversa del logaritmo de la iteración en curso, supuesto que el número de transiciones a efectuar en

cada iteración sea suficiente como para que todos los nodos de la red puedan cambiar de estado. Es decir:

$$T(n) = \frac{T_0}{1 + Ln(t)}$$

donde T_0 es la temperatura inicial, y t es el número de iteración del procesamiento. No obstante, este programa de templado es impracticable, ya que requiere un tiempo de computación exponencial.

Con respecto a la resolución de problemas de optimización combinatoria, es necesario indicar que el proceso de implementación del problema en la máquina de Boltzmann es idéntico al de la red de Hopfield continua, lo mismo que la forma de ajustar los parámetros de la función de energía para que la estabilización de la red sea correcta.

Sin embargo, el mecanismo de optimización empleado por la máquina de Boltzmann es totalmente distinto al de la red de Hopfield continua. La máquina de Boltzmann no puede implementar un proceso paralelo y continuo de optimización por competición como la red CH, porque actúa en tiempo discreto, posee nodos con valores de activación binarios, y el procesamiento es asíncrono. Por el contrario, fundamenta la optimización en la técnica probabilística del temple simulado, la cual permite obtener soluciones óptimas bajo un programa de templado proporcional a la inversa del logaritmo de la iteración en curso.

No obstante, la máquina de Boltzmann, bajo un programa práctico de templado como el propuesto por Kirkpatrick, Gelatt y Vecchi [KIRK83], no se comporta tan bien como el algoritmo original del temple simulado, resultando incluso menos eficiente que la red de Hopfield, especialmente en ciertos problemas con restricciones que generan funciones de energía muy rugosas. Esto se debe a un inconveniente intrinseco a su modo de procesamiento, el cual puede observarse estableciendo una sencilla analogía con el algoritmo del temple simulado.

En una máquina de Boltzmann las n variables discretas del problema de optimización se implementan mediante n vectores de nodos binarios, de tal forma que cada nodo del vector representa un valor del dominio de la variable correspondiente. Por otra parte, en el algoritmo del temple simulado cada transición de estado implica realizar un cambio sobre tantas variables como se indique en la estructura de soluciones k-próximas. Sin

embargo, el esquema de procesamiento de la máquina de Boltzmann permite transiciones de un solo nodo cada vez, cuando se precisa cambiar de estado al menos dos nodos del mismo vector para que cambie el valor de la variable asociada. Además, de acuerdo con la forma de implementar el problema en la máquina de Boltzmann, el cambio de un único nodo a partir de un estado válido de la red siempre genera un gran incremento de energía (se viola una restricción). Por esta razón, sólo para temperaturas altas en las que se permite cambiar varios nodos de forma consecutiva con incrementos de energía, es posible transformar un estado válido de la red en otro diferente, pero igualmente válido. Por tanto, a partir de cierto valor de la temperatura (no necesariamente bajo) la red queda atrapada en un estado estable del que no puede escapar, puesto que un cambio en un solo nodo supone un incremento de energía tan elevado que o no se aceptará, o si se acepta cualquier nueva transición devolverá la red al mismo estado estable anterior.

En resumen, puede decirse que la máquina de Boltzmann implementa el algoritmo del temple simulado sobre un espacio de estados inferior al de una estructura de soluciones 1-próximas, lo cual es definitivamente insuficiente para que el proceso de optimización tenga éxito.

3.5 MODELOS EVOLUCIONADOS.

3.5.1 Máquina de Cauchy.

La máquina de Cauchy (Cauchy Machine, CM) introducida por Szu como una variante de la máquina de Boltzmann [SZU87], es una red recurrente, que opera en tiempo discreto, y emplea un procedimiento supervisado off-line de aprendizaje estocástico, basado en la técnica del temple simulado rápido (fast simulated annealing). Desde el paradigma de la asociación de patrones, el modelo CM es un asociador hacia el patrón más próximo de patrones binarios $X_h = (x_1^h, ..., x_n^h)$, para h = 1, ..., m.

La disposición de los nodos, el modo de procesamiento y la regla de aprendizaje de la máquina de Cauchy son idénticos a los de la máquina de Boltzmann. Sin embargo, en la red CM se utiliza la distribución de probabilidad de Cauchy, en lugar de utilizar la distribución de Boltzmann. De esta manera, en el método propuesto por Szu, el cálculo de los estados de activación de los nodos durante el procesamiento o el aprendizaje se

realiza de la siguiente forma: independientemente del valor en curso de un nodo j elegido al azar, se asigna a dicho nodo el valor $a_j = 1$ con una probabilidad que viene determinada por la distribución de Cauchy:

$$\frac{1}{2}+\frac{1}{\pi}\text{arctg}\left(\frac{\mu_j}{T}\right) > \text{random}[0,1[$$

donde T es la temperatura actual del sistema, y μ_j es la entrada neta del nodo j.

La distribución de Cauchy tiene la misma forma general que la distribución de Boltzmann (sigmoide), pero no decae tan rápidamente para grandes energías. La consecuencia es que la máquina de Cauchy puede dar ocasionalmente un gran salto para salir de un mínimo local. De acuerdo con el trabajo de Szu, la ventaja de esta aproximación es que se puede alcanzar el mínimo global con un programa de templado más breve, inversamente proporcional a la iteración en curso:

$$T(n) = \frac{T_0}{1+t}$$

donde T_0 es la temperatura inicial, y t es el número de iteración del procesamiento. Este programa de templado es más rápido que el correspondiente a la máquina de Boltzmann, razón por la cual el algoritmo de Szu se denomina *temple simulado rápido*.

A pesar de la ventaja teórica del temple simulado rápido, en la práctica y bajo un programa de templado inversamente proporcional a la exponencial de la iteración en curso, el comportamiento de la máquina de Cauchy no es mejor que el de la máquina de Boltzmann en su aplicación a problemas de optimización combinatoria, dado que conserva su mismo esquema de procesamiento, en el que las transiciones de estado se producen considerando incrementos de energía debidos a la actualización de un solo nodo cada vez.

3.5.2 Máquina de Gauss.

La máquina de Gauss (Gaussian Machine, GM) introducida por Akiyama et al [AKIY89], es una red recurrente, que opera en tiempo continuo con un esquema de procesamiento basado en la técnica del temple simulado Gaussiano. Desde el paradigma de la asociación de patrones, la máquina de Gauss es un autoasociador hacia el patrón más próximo de patrones analógicos $X_h = (x_1^h, ..., x_n^h)$, para h = 1, ..., m.

El modelo GM, presentado por Akiyama et al en 1989, combina las características de dos modelos conexionistas previos con el fin de mejorar el comportamiento de la red en aplicaciones de optimización. Por una parte, opera en tiempo continuo y posee nodos con valores de activación analógicos sobre el intervalo]0, 1[como la red de Hopfield continua. Por otra parte, incorpora un componente estocástico en el procesamiento, como la máquina de Boltzmann, con el fin de que el sistema pueda escapar de los mínimos locales. El nombre de *máquina de Gauss* se debe a que el comportamiento estocástico del procesamiento se deriva de la distribución normal o de Gauss, utilizada en la generación de un ruido aleatorio que se añade a la entrada de los nodos de la red.

La disposición de los nodos y la función de energía de la máquina de Gauss son idénticos a los de la red de Hopfield continua. Con respecto al modo de procesamiento sólo se diferencia de ésta en un componente aleatorio que se incorpora a la entrada neta de cada nodo en el cálculo de su nuevo estado de activación. Así, cada nodo j de la red actualiza su estado de activación mediante la función de umbral sigmoidal tangente hiperbólica:

$$f(\mu_j) = \frac{1}{2} + \frac{1}{2}\,\mathrm{th}(\mu_j)$$

aplicada a la entrada neta μ_j del nodo, la cual se obtiene aplicando la regla de combinación aditiva:

$$\mu_j = \sum_{i=1}^{N} w_{ij} a_i + I_j + \varepsilon$$

donde ε es el error de la entrada causado por el ruido aleatorio.

La característica más novedosa de la máquina de Gauss es que la entrada neta μ_j de cada nodo está siempre influenciada por un ruido estocástico ε. Este ruido queda determinado por una distribución normal o de Gauss de media cero y desviación típica igual a:

$$\sigma = \sqrt{\tfrac{8}{\pi}}\,T$$

donde T es la temperatura en curso del sistema. De esta forma, para valores grandes de la temperatura, el ruido que se incorpora a la entrada neta de los nodos es grande, mientras que para valores de T próximos a cero, el valor de ε es prácticamente nulo, con lo que la red se comporta de forma determinista.

La máquina de Gauss fue pensada como una evolución de la red de Hopfield continua, mediante la cual se intentaba mejorar su comportamiento en aplicaciones de

optimización, al incorporar un componente estocástico basado en el temple simulado. No obstante, sigue presentando algunos de los inconvenientes de la red de Hopfield continua y la máquina de Boltzmann:

- El proceso de optimización competitivo, que comparte con la red de Hopfield continua, es igualmente menos eficiente cuanto mayor sea el tamaño del problema implementado, y cuanto mayor sea la rugosidad de la función de energía de la red que representa los objetivos y restricciones del problema.
- Así mismo, el sincronismo del procesamiento de la red, generador del proceso competitivo de optimización, requiere que los estados iniciales de los nodos sean valores próximos a 0.5, pero con una variación aleatoria ε que rompa la simetría del procesamiento y evite una oscilación infinita de la red. Sin embargo, en la máquina de Gauss, esta variación viene proporcionada por el ruido estocástico que se añade a la entrada neta de cada nodo, y al ser este inicialmente muy grande disminuye considerablemente el efecto competitivo del procesamiento.
- Además, en el esquema de procesamiento de la máquina de Gauss cada nodo se actualiza estocásticamente teniendo en cuenta sólo su propia entrada neta, como ocurre en la máquina de Boltzmann. Por tanto, la modificación del valor de activación de cada nodo se realiza en función del incremento de energía generado por el propio nodo, sin tener en cuenta las modificaciones simultáneas que se producen en los restantes nodos, es decir, es totalmente local. Esto implica que el espacio de estados de búsqueda está excesivamente limitado, y lo que es peor, que la técnica del temple simulado se aplica de forma incorrecta, ya que la aceptación estocástica de los nuevos estados de los nodos debería depender del incremento de energía global producido.

3.5.3 Red de temple determinista.

La red de temple determinista (Mean Field Annealing, MFA) introducida por Van den Bout y Miller como una variante de la red de Hopfield continua [BOUT89], es una red recurrente, que opera en tiempo continuo mediante un esquema de procesamiento basado en la teoría del temple de campo medio o temple determinista. Desde el paradigma de la asociación de patrones, la red de temple determinista es un autoasociador hacia el patrón más próximo de patrones analógicos $X_h = (x_1^h, \ldots, x_n^h)$, para $h = 1, \ldots, m$.

La red de temple determinista, al igual que la máquina de Gauss, es una evolución de la red de Hopfield continua pensada con el fin de mejorar el comportamiento de ésta en aplicaciones de optimización. Por una parte, opera en tiempo continuo y posee nodos con valores de activación analógicos sobre el intervalo]0, 1[como la red de Hopfield continua. Por otro lado, posee un mecanismo de procesamiento que incorpora una aproximación no estocástica al temple simulado, denominada temple de campo medio o temple determinista.

La disposición de los nodos y la función de energía de la red MFA son idénticos a los de la red de Hopfield continua. Con respecto al esquema de procesamiento, se diferencia de ésta en que en el cálculo del estado de activación a_j de cada nodo, la entrada neta μ_j se divide entre un parámetro de control T, equivalente a la temperatura del algoritmo del temple simulado. Así, cada nodo j de la red actualiza su estado de activación mediante una función de umbral sigmoidal, como la función tangente hiperbólica o la función logística:

$$f\left(\frac{\mu_j}{T}\right) = \frac{1}{1+e^{-\mu_j/T}}$$

Como se aprecia, este modo de procesamiento exige un programa de templado similar al de la máquina de Boltzmann, la máquina de Cauchy o la máquina de Gauss, pero es totalmente determinista y no probabilístico como ocurre con estos otros modelos.

La red de temple determinista fue también pensada como un medio de mejorar las prestaciones de la red de Hopfield continua en problemas de optimización, pero, aunque reduce notablemente el tiempo de procesamiento, posee varios de los inconvenientes de los modelos precedentes:

- Como ocurre en la red de Hopfield continua y en la máquina de Gauss, el proceso de optimización competitivo es tanto menos eficiente cuanto mayor sea el tamaño del problema implementado, y cuanto mayor sea la rugosidad de la función de energía de la red que representa los objetivos y restricciones del problema.
- La variación aleatoria ε, que se añade a los estados iniciales de los nodos para romper la simetría del procesamiento y evitar la oscilación infinita de la red, debe elegirse con cuidado, ya que si es muy grande, anula el proceso competitivo y con ello el mecanismo de optimización, pero si es demasiado pequeña la red puede llegar a no estabilizarse nunca. El valor apropiado de ε depende de la rugosidad de la función de energía de la red, por lo que debe encontrarse de forma empírica.

- Por otra parte, el temple determinista no representa más que una aproximación al temple estocástico, por lo que el nivel de optimización conseguido no es tan bueno como el de éste. Además, en el esquema de procesamiento de la red cada nodo se actualiza teniendo en cuenta sólo su propia entrada neta. Por tanto, la modificación del valor de activación de cada nodo se realiza sin tener en cuenta las modificaciones simultáneas que se producen en los restantes nodos, lo que hace que el espacio de búsqueda quede excesivamente limitado.

4

El Optimizador Discreto Estocástico, ODE

Sed omnia praeclara tam difficilia, quam rara sunt.

Benedicto de Spinoza

4.1 INTRODUCCIÓN.

4.1.1 Problemática de la resolución de problemas de optimización combinatoria mediante redes neuronales.

En el capítulo anterior se analizaron los diferentes tipos de redes neuronales artificiales empleados hasta el momento en la resolución de problemas de optimización combinatoria, centrando el estudio en los dos modelos más representativos, la red de Hopfield continua y la máquina de Boltzmann, junto a los modelos evolucionados a partir de éstos, la máquina de Cauchy, la máquina de Gauss y la red de temple determinista. En el breve análisis realizado para cada uno de estos modelos puede apreciarse que, en general, la eficiencia de los mismos en cuanto a la calidad de las soluciones obtenidas es bastante pobre en relación con los métodos heurísticos presentados en el capítulo 2, la búsqueda local y el temple simulado, y además en algunos casos esta eficiencia varía según el tipo de problema a resolver.

La causa de esta aparente falta de eficiencia se debe a varios factores, algunos comunes a todas las redes neuronales estudiadas, y otros específicos de cada modelo concreto. De todos estos factores, los más importantes se reducen a tres: el ajuste de los parámetros de la función de energía, la limitación del procesamiento competitivo de las redes de tipo continuo, y la limitación del procesamiento asíncrono y local de las redes discretas.

Ajuste de los parámetros de la función de energía

El inconveniente más general de la implementación de problemas de optimización mediante redes neuronales, aunque también el más fácil de resolver, viene dado por el *ajuste de los parámetros de la función de energía de la red*, la cual representa las restricciones y objetivos del problema.

En principio, el ajuste de los parámetros de la función de energía debe realizarse de forma empírica, con el inconveniente de que si se da un valor excesivo a los parámetros que implementan los objetivos de la optimización con respecto a los parámetros asociados a las restricciones, puede hacer que la red se estabilice en un estado correspondiente a una solución no válida. Por el contrario, si se da más importancia a los parámetros asociados a las restricciones, frente a los parámetros correspondientes a los objetivos de la optimización, la solución obtenida será válida en la mayoría de los casos, pero de baja calidad.

En los estudios realizados sobre los diferentes modelos conexionistas [KAHN89] [BOUT89] [TAKE89] [AKIY89] [ABE89] [JEON89] [AIYE90] [CULI90] [CHEN90], se documenta una buena actuación general de las redes neuronales cuando el ajuste favorece los objetivos de la optimización, siempre que la solución obtenida sea factible. Sin embargo, esta situación se da con poca frecuencia, ya que bajo este supuesto en la mayoría de los casos la solución obtenida no verifica las restricciones. Se trata de un inconveniente importante, dado que un algoritmo de optimización que no garantiza una solución factible parece poco aplicable en la práctica.

No obstante, es posible definir un método de diseño específico que permita determinar de forma exacta los parámetros de la función de energía, de manera que se garantice la estabilización de la red en un estado correspondiente a una solución factible del problema. Así, en el siguiente capítulo de la tesis se proporciona un procedimiento completo de diseño, válido para la implementación general de problemas de optimización combinatoria en cualquier modelo de red neuronal recurrente.

Procesamiento competitivo de las redes de tipo continuo

El procesamiento síncrono de las redes de tipo continuo, como la red CH, la máquina de Gauss, y la red de temple determinista, se caracteriza por una *competición paralela entre los nodos*: inicialmente todos los nodos toman valores próximos a 0.5 y su estado se va

actualizando síncronamente de manera continua hasta que quedan activos k nodos, los ganadores de la competición, mientras los restantes nodos quedan inactivos. Con respecto a la optimización, este proceso competitivo se hace tanto menos eficiente cuantos más nodos posea la red, y cuanto mayor sea la rugosidad de la función de energía de la red que representa los objetivos y restricciones del problema. Así mismo, la rugosidad depende de las características del problema, por lo que el comportamiento de este tipo de redes en cuanto a la calidad media de la solución obtenida varía según el tipo de problema implementado.

Además, el sincronismo del procesamiento de la red requiere que los estados iniciales de los nodos sean valores próximos a 0.5, pero con una variación aleatoria ε, que rompa la simetría del procesamiento y evite una oscilación infinita de la red. Un valor de ε grande puede anular el proceso competitivo y con ello el mecanismo de optimización, y un valor demasiado pequeño puede hacer que la red no llegue a estabilizarse nunca. El valor apropiado de ε depende nuevamente de la rugosidad de la función de energía de la red, por lo que debe encontrarse de forma empírica. Además, en las redes que utilizan una técnica estocástica basada en el temple simulado, como la máquina de Gauss, el valor de ε aparece en forma de ruido aleatorio que se añade a la entrada neta de cada nodo. En este caso, al ser ε inicialmente muy grande, la posible ventaja que debería aportar la incorporación del criterio estocástico desaparece al disminuir notablemente el efecto competitivo del procesamiento.

Procesamiento asíncrono y local de las redes discretas

Las redes discretas basadas en la técnica del temple simulado, como la máquina de Boltzmann y la máquina de Cauchy, presentan una limitación intrínseca a su modo de procesamiento asíncrono y local, con *transiciones que producen cambios de energía debidos a la modificación del estado de un solo nodo cada vez.* El efecto de esta limitación puede apreciarse fácilmente comparando este esquema de procesamiento con los algoritmos de tipo heurístico basados en una estructura de soluciones k-próximas.

En los algoritmos heurísticos de búsqueda local y del temple simulado cada transición de estado implica realizar un cambio sobre tantas variables como se indique en la estructura de soluciones k-próximas del problema. En el caso de determinados problemas con restricciones, como el del viajante, la mínima estructura que puede utilizarse es una estructura de soluciones 2-próximas, lo que implica un cambio simultáneo de dos variables cada vez. Esto es necesario porque, al efectuar una transición de estado, el

cambio de valor de una sola variable viola una restricción, generándose una solución intermedia no válida, que no puede aceptarse. Por ello, si se utiliza una estructura de soluciones 1-próximas, la solución final obtenida es la misma solución inicial factible elegida al azar (no se acepta ninguna transición), con lo que el proceso de optimización resulta nulo.

Por otra parte, en una red neuronal las n variables discretas del problema de optimización original se implementan mediante n vectores de nodos, de manera que cada nodo de un vector representa un valor distinto del dominio p de la variable correspondiente. Ahora bien, el esquema de procesamiento de estas redes permite transiciones de sólo un nodo cada vez, cuando se necesita cambiar de estado al menos dos nodos del mismo vector para que la variable correspondiente cambie de valor. Si el problema a resolver posee restricciones y los parámetros de la red se han ajustado de manera que se garantice una solución final válida, el cambio de un único nodo a partir de un estado válido de la red siempre genera un gran incremento de energía, ya que viola una restricción. De esta forma, la red queda atrapada en un estado estable, puesto que un cambio en un solo nodo supone un incremento de energía muy grande que no se aceptará, excepto cuando la temperatura del sistema sea elevada. Por lo tanto, puede decirse que las redes de Boltzmann y Cauchy implementan búsquedas sobre un espacio inferior al de una estructura de soluciones 1-próximas, es decir, el espacio de estados sobre el que se produce la búsqueda es totalmente insuficiente para que el proceso de optimización tenga éxito.

4.1.2 Generalización del modo de procesamiento de la máquina de Boltzmann.

El análisis realizado sobre las redes neuronales en problemas de optimización combinatoria indica que los modelos actuales, basados en la red de Hopfield continua y en la máquina de Boltzmann, son insuficientes para una resolución plausible de este tipo de problemas, por lo que se ha llegado a cuestionar su utilidad real en el campo de la optimización combinatoria [WILS88] [GEE95]. Sin embargo, las importantes características de las redes neuronales recurrentes bastan para motivar la investigación y desarrollo de nuevos modelos conexionistas que superen las limitaciones de los anteriores y exploten eficientemente el paralelismo masivo de las RNA en la resolución de problemas combinatorios de optimización.

Dado que el algoritmo del temple simulado es sin duda el mejor método heurístico general de optimización combinatoria conocido, lo más deseable sería definir un nuevo modo de procesamiento de red basado en este método estocástico.

Sin embargo, en el caso de los modelos de red continuos, la utilización de un ruido aleatorio basado en el temple simulado para el cálculo de los estados de los nodos resulta poco conveniente, dado que afecta de forma negativa al carácter competitivo del procesamiento, carácter en el que se fundamenta el mecanismo de optimización de la red.

Por otra parte, en el caso de las redes discretas, sería necesario mejorar su comportamiento introduciendo un nuevo esquema de procesamiento en el que se eliminara la limitación debida a las transiciones de estado nodo a nodo, con el fin de ampliar el espacio de estados de búsqueda y obtener así soluciones de alta calidad.

Con este objetivo, se define a continuación un nuevo modo de procesamiento que constituye una generalización directa del de la máquina de Boltzmann en el sentido indicado.

Al igual que en la red BM original, cada estado de la máquina de Boltzmann generalizada posee un valor de energía asociado, definido por la función de Lyapunov:

$$E(A) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} a_i a_j - \sum_{i=1}^{N} I_i a_i$$

Así mismo, para mejorar la actuación de la red en los problemas de optimización con restricciones, el nuevo procesamiento debe generar transiciones de estado que modifiquen 2k nodos simultáneamente, siendo k la característica de la estructura de soluciones próximas que se desea implementar. El siguiente algoritmo permite simular el comportamiento básico de este modelo en la actualización de los estados de los nodos:

Algoritmo. El algoritmo de actualización de los nodos de la máquina de Boltzmann generalizada puede describirse de la siguiente forma:

1. Inicialmente, los valores de activación de los nodos, a_i, corresponden a los componentes x_i del patrón aleatorio de entrada X.

2. A continuación, se realiza una iteración de procesamiento asíncrono. Se eligen 2k nodos cualesquiera de la red y se actualizan al azar los valores de activación respectivos. Al nuevo estado de la red se le aplica un criterio de aceptación

basado en el algoritmo de metrópolis. Si el incremento de energía producido en la red, ΔE, es negativo se acepta el nuevo estado de la red, y, en caso contrario, el nuevo estado se acepta con una probabilidad determinada por la distribución de Boltzmann:

$$e^{\frac{-\Delta E}{T}} > \text{random}[0,1[$$

donde T es la temperatura de la red. Cada iteración del procesamiento corresponde a un valor decreciente de la temperatura T, y al igual que en el algoritmo del temple simulado debe generar un número de transiciones suficiente como para que todas las combinaciones de 2k nodos de la red puedan haber sido afectadas por al menos una transición de estado.

3. Si al menos un nodo ha cambiado de estado en la última iteración se decrementa la temperatura y se repite el paso 2. En caso contrario, la red se considera estabilizada y se finaliza el algoritmo.

Este algoritmo permite teóricamente obtener soluciones con una calidad de optimización semejante a la del temple simulado. Sin embargo, si se considera el algoritmo básico anterior, entonces el tiempo de ejecución secuencial del modelo se incrementa notablemente con respecto al tiempo de ejecución del temple simulado, dado que el número de transiciones a efectuar para cada valor de la temperatura es mucho mayor. Para un problema de n variables con dominio p, y una estructura de soluciones k-próximas dada, el número mínimo de transiciones a efectuar para cada temperatura en el temple simulado es:

$$p^k\binom{n}{k} = \frac{p^k n!}{(n-k)!k!}$$

mientras que en la máquina de Boltzmann generalizada es igual a:

$$2^{2k}\binom{np}{2k} = \frac{2^{2k}(np)!}{(np-2k)!(2k)!}$$

No obstante, es posible reducir drásticamente el tiempo de ejecución del algoritmo introduciendo las siguientes modificaciones en el modo de procesamiento:

- El patrón de entrada X que representa el estado inicial de la red se elige al azar, pero dentro del conjunto de estados que corresponden a soluciones factibles del problema. Esto implica necesariamente que, del vector de nodos de la red correspondiente a cada variable del problema, sólo un nodo estará activo.

- Los 2k nodos cuyos valores de activación deben actualizarse en cada transición de estado se seleccionan de la siguiente forma: Se eligen al azar k vectores de nodos correspondientes a variables del problema, y cada vector se actualiza con todos los nodos inactivos excepto uno elegido al azar. De esta forma, el número total de transiciones a efectuar en cada valor de la temperatura se reduce al del temple simulado.
- Tanto en el cálculo de la temperatura inicial como en el procesamiento de la red, el criterio probabilístico de aceptación se aplica cuando la transición de estado aumenta la energía del sistema, pero sólo si ésta permanece por debajo de un cierto valor de umbral θ. Este umbral deberá ser una cota superior de la máxima energía de la red correspondiente a una solución factible. De esta forma, como los estados correspondientes a soluciones que no verifican las restricciones están muy penalizados, no afectarán al cálculo de la temperatura inicial. Así, la temperatura inicial será menos elevada, y permitirá efectuar transiciones al azar, pero sólo si el nuevo estado generado corresponde a una solución factible. Con ello, se elimina del algoritmo el periodo inútil correspondiente a temperaturas tan elevadas que permiten transiciones a estados arbitrarios que pueden no verificar las restricciones del problema, con lo que se disminuye considerablemente el tiempo de ejecución.

En conclusión, el nuevo modo de procesamiento propuesto para la máquina de Boltzmann generalizada permite optimizar problemas con restricciones con un nivel de calidad semejante al del algoritmo del temple simulado, empleando un tiempo en la simulación secuencial cercano al de este último.

Sin embargo, un inconveniente que presenta este modelo es que se pierde el mecanismo de control totalmente local de la máquina de Boltzmann original, dado que la aceptación de cada transición de estado depende de varios nodos a la vez. Este hecho puede no parecer excesivamente importante en un principio, pero con él la máquina de Boltzmann generalizada pierde una de las propiedades básicas que, por definición, caracterizan a una red neuronal artificial: Todo el procesamiento asociado a una red neuronal se realiza en paralelo *de forma distribuida y local*, es decir, el valor de activación de cada nodo o unidad elemental de la red depende exclusivamente de las señales de entrada del nodo y del estado interno del mismo. Esto significa que la generalización presentada de la máquina de Boltzmann ya no es una red neuronal artificial en sentido estricto.

Por la razón anterior, parece conveniente definir una arquitectura de red apropiada para el nuevo algoritmo de procesamiento que se acaba de describir, a fin de conseguir que la actualización de los nodos recupere el carácter local propio de las RNA.

Se propone por tanto, un nuevo modelo conexionista específico para resolución de problemas de optimización combinatoria, que se denomina *optimizador discreto estocástico, ODE*, y se estudia en detalle en los restantes apartados del capítulo.

4.2 EL OPTIMIZADOR DISCRETO ESTOCÁSTICO, ODE: DEFINICIÓN Y CARACTERÍSTICAS.

El optimizador discreto estocástico, ODE, es una red recurrente, que opera en tiempo discreto, e incorpora un modo de procesamiento síncrono basado en la técnica estocástica del temple simulado. Desde el paradigma de la asociación de patrones, el modelo ODE es un autoasociador hacia el patrón más próximo de patrones binarios $X_h = (x_1^h, ..., x_n^h)$, para h = 1, ..., m.

El modelo ODE puede considerarse una generalización de la máquina de Boltzmann [HINT86], con un esquema de procesamiento que implementa de forma paralela el algoritmo del temple simulado sobre una estructura de soluciones k-próximas. La principal característica del modelo es que adopta una nueva arquitectura o disposición de los nodos no simétrica, que permite generalizar el comportamiento estocástico de la máquina de Boltzmann, conservando la propiedad de procesamiento paralelo distribuido característico de toda red neuronal, en el que cada nodo se actualiza de forma local atendiendo únicamente a sus señales de entrada.

4.3 ESTRUCTURA BÁSICA DE LA RED ODE.

El optimizador discreto estocástico posee una configuración de 2N + 1 nodos, donde N indica el tamaño o número de componentes de los patrones a procesar. Los nodos se agrupan en dos capas, una visible con N nodos principales más N nodos auxiliares, y otra oculta formada por un único nodo (figura 4.1).

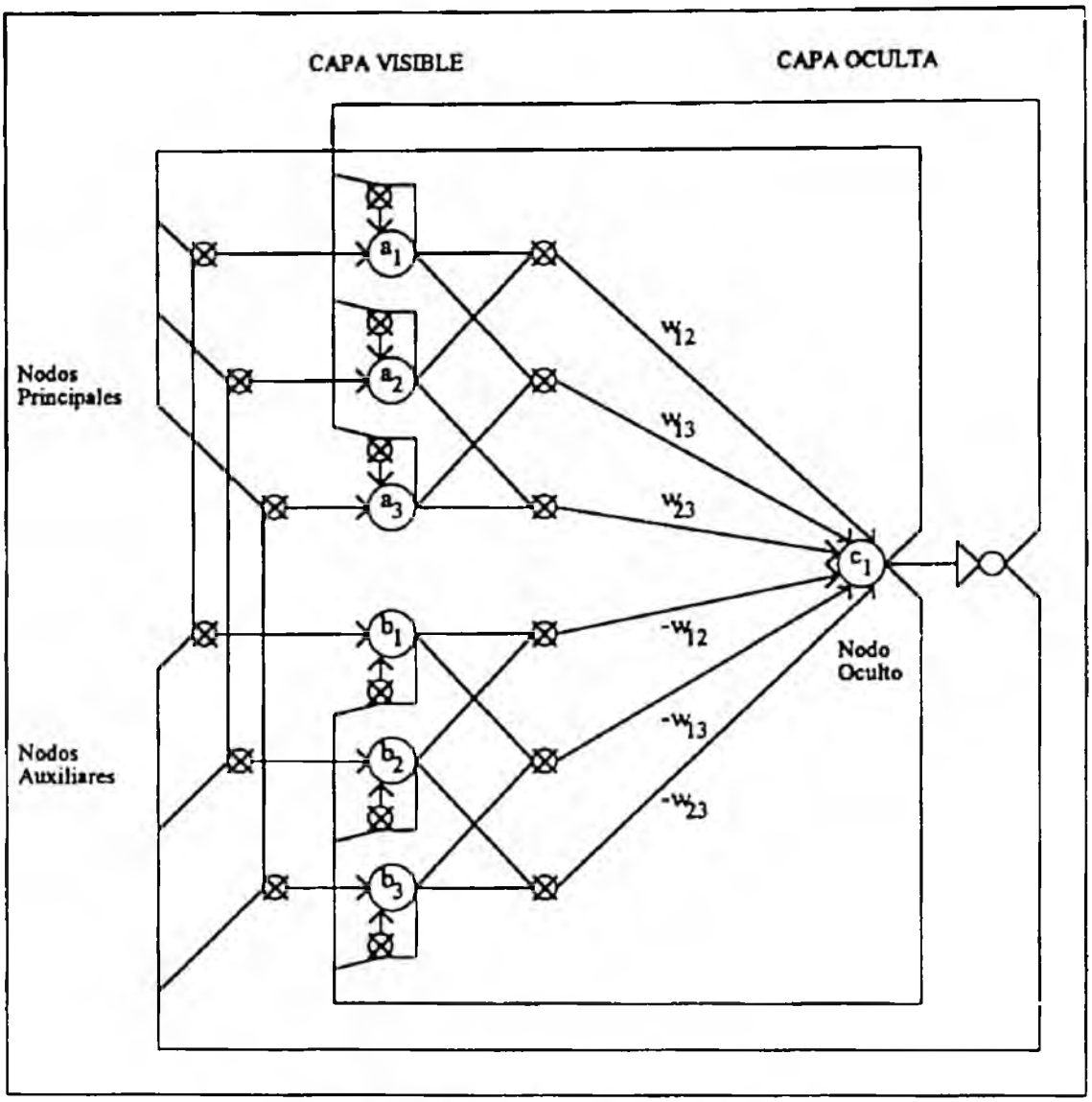

Figura 4.1: Red ODE de 7 nodos.

En este modelo, los N nodos principales de la capa visible corresponden a los N componentes binarios del patrón de entrada X_h. El patrón X_h representa la solución inicial aleatoria del problema de optimización a resolver, teniendo en cuenta que si el problema de optimización posee n variables discretas de dominio p, entonces N = np. Esto significa que los N nodos principales se agrupan en n vectores de p nodos cada uno, asociándose un vector a cada variable de optimización. Por otra parte, los N nodos auxiliares se utilizan para representar las nuevas soluciones que deben aceptarse o rechazarse en el procesamiento. El nodo oculto implementa el comportamiento estocástico del algoritmo del temple simulado, y su salida se utiliza para aceptar o rechazar la transición efectuada.

Todas las conexiones de la red son de segundo orden, es decir, poseen dos nodos emisores cuyas salidas se multiplican previamente a la operación de suma ponderada, de acuerdo con la regla de combinación aditiva-multiplicativa. Las correlaciones de orden superior se utilizan habitualmente en procesamiento de imágenes para implementar invarianza ante traslaciones, escalados y rotaciones, y también pueden usarse, como sucede en la red ODE, para implementar problemas de optimización combinatoria.

De los N nodos principales de la capa visible parten conexiones unidireccionales con pesos w_{ij} hacia el nodo oculto, y lo mismo ocurre con los nodos auxiliares, pero con pesos respectivos opuestos $-w_{ij}$. Estos pesos implementan las restricciones y objetivos del problema a resolver, y se determinan previamente de la misma forma que para cualquier otro modelo conexionista de optimización. Del nodo oculto retornan dos conexiones a cada nodo de la capa visible, una de ellas previamente negada. Así mismo, los nodos de la capa visible están conectados recursivamente, y cada nodo principal se conecta con su correspondiente nodo auxiliar en ambas direcciones, formando conexiones de segundo orden con las señales procedentes del nodo oculto (figura 4.2). Los pesos de estas conexiones son siempre iguales a la unidad.

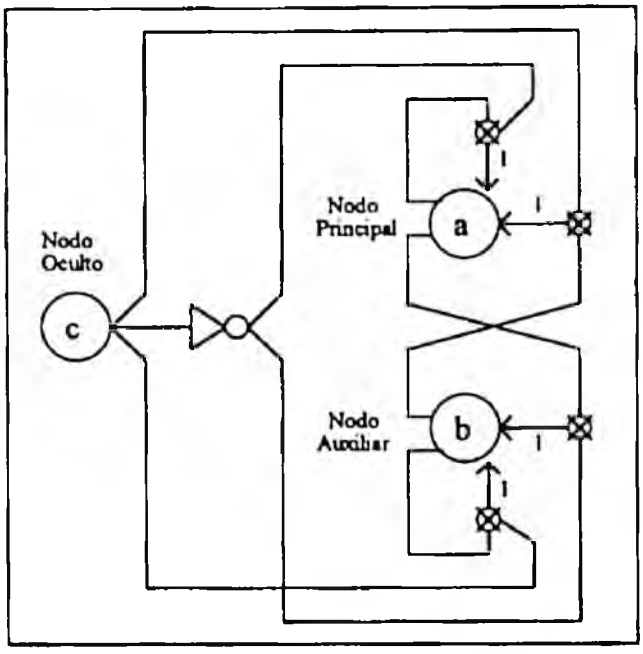

Figura 4.2: Detalle de los nodos principal y auxiliar en el modelo ODE.

4.4 ESQUEMA DE PROCESAMIENTO DE LA RED ODE.

4.4.1 Actualización de los estados de los nodos.

El optimizador discreto estocástico utiliza un mecanismo de procesamiento recurrente de tipo síncrono, en el que una entrada X se procesa hasta que la red alcanza un estado estable, es decir, hasta que cualquier nueva transición de estado no provoque un cambio en ninguno de los nodos principales de la capa visible. En este modelo la salida de los nodos visibles es determinista, mientras que la salida del nodo oculto es una función estocástica de las entradas, es decir, se calcula de acuerdo con la distribución de probabilidad de Boltzmann, en lugar de utilizar una función de umbral determinista.

A continuación se ofrece el algoritmo que simula el esquema de procesamiento del optimizador discreto estocástico:

Algoritmo. Dada una red ODE de 2N + 1 nodos, que implementa un problema de optimización de n variables discretas de dominio p, siendo N = np, entonces el algoritmo de actualización de los nodos puede describirse de la siguiente forma:

1. Inicialmente, los valores de activación a_i de los N nodos principales de la capa visible, así como los valores b_i de los N nodos auxiliares, corresponden a los N componentes binarios x_i del patrón de entrada X. Este patrón de entrada inicial X se elige al azar entre todos los que corresponden a soluciones factibles del problema a resolver. Esto implica que, de cada vector de nodos principales de la red asociado a una variable del problema, sólo un nodo estará activo.

2. Se inicializa la temperatura o parámetro de control $T = T_0$. En el cálculo de T_0 se emplea una variante del algoritmo simple visto para la máquina de Boltzmann, en el que el criterio probabilístico de aceptación se aplica cuando la transición de estado aumenta la energía del sistema, pero sólo si ésta permanece por debajo de un cierto valor de umbral θ. De esta forma, dado que los estados de la red que no verifican las restricciones poseen energías muy elevadas, no afectarán al cálculo de T_0. Así, la temperatura inicial permitirá efectuar transiciones al azar, pero sólo si el nuevo estado generado corresponde a una solución factible.

3. A continuación, se realiza una iteración de procesamiento sincrono, que consta de tres fases:

3.a Se eligen al azar k vectores de nodos auxiliares correspondientes a variables del problema, y cada vector se actualiza con todos los nodos inactivos, excepto uno elegido al azar que toma valor 1.

3.b Posteriormente, se calcula el nuevo valor de activación c_1 del nodo oculto, aplicando a la entrada neta del mismo un criterio basado en el algoritmo de Metropolis. Si la entrada neta μ es negativa, entonces el nuevo valor del nodo es $c_1 = 1$. En caso contrario, si μ es positiva, pero inferior al valor de umbral θ especificado, el nuevo estado del nodo oculto será $c_1 = 1$ con una probabilidad determinada por la distribución de Boltzmann:

$$e^{\frac{-\mu}{T}} > \text{random}[0,1[$$

En la expresión anterior T es la temperatura actual del sistema, y μ es la entrada neta del nodo oculto, calculada utilizando la regla de combinación aditiva-multiplicativa de segundo orden, de la forma siguiente:

$$\mu = \sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij} a_i a_j + \sum_{i=1}^{N}\sum_{j=i+1}^{N} -w_{ij} b_i b_j$$

donde a_i y b_i son respectivamente el estado del i-ésimo nodo principal y el estado del i-ésimo nodo auxiliar de la capa visible, w_{ij} es el peso de la conexión de segundo orden que parte de los nodos principales i y j, y $-w_{ij}$ es el peso de la conexión de segundo orden que parte de los nodos auxiliares i y j.

3.c A continuación, se calcula el nuevo estado de activación de los nodos visibles, aplicando a la entrada neta μ_j de cada uno la función de umbral de salto binario:

$$f(\mu_j) = \begin{cases} 1 & \text{si } \mu_j > 0 \\ 0 & \text{si } \mu_j \leq 0 \end{cases}$$

De acuerdo con la estructura de la red, si la salida del nodo oculto es 1, entonces los nodos auxiliares permanecen invariables y sus estados se copian sobre los nodos principales (se acepta la transición de estado), mientras que si la salida del nodo oculto es 0, entonces son los nodos principales los que no cambian y sus estados se copian sobre los nodos auxiliares (se rechaza la transición de estado).

Este cálculo o iteración de procesamiento se realiza tantas veces para cada valor de la temperatura como sea necesario hasta conseguir el equilibrio térmico. El número de iteraciones depende del programa de templado utilizado, y será como mínimo igual al tamaño de la estructura de soluciones k-próximas implementada.

4. Si al menos un nodo principal de la capa visible ha cambiado de estado en alguna de las iteraciones correspondientes al último valor de la temperatura, entonces se decrementa T y se repite el paso 3. En caso contrario, la red se considera estabilizada y se finaliza el algoritmo.

El algoritmo de procesamiento de la red ODE posee un comportamiento estocástico variable de acuerdo con la temperatura T. Así, para temperaturas elevadas, la función de distribución del modelo ODE toma valores próximos a 1, al ser el término $-\mu / T$ muy pequeño en valor absoluto. En esta situación, se aceptan todas las transiciones de estado producidas, correspondientes a soluciones factibles. Sin embargo, para valores de la temperatura pequeños, la función de Boltzmann toma valores próximos a 0, con lo que el sistema se comporta prácticamente de forma determinista, aceptándose sólo transiciones debidas a valores negativos de la entrada neta μ del nodo oculto.

4.4.2 Estabilidad del procesamiento.

En el optimizador discreto estocástico, cada estado de la red posee un valor de energía asociado, definido por la función de Lyapunov:

$$E(A) = -\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} a_i a_j$$

donde $A = (a_1, \ldots, a_n)$ es el vector de estados de los nodos visibles principales de la red.

Esta función de energía es equivalente a la función de energía de la máquina de Boltzmann, salvo porque no se consideran entradas adicionales de tendencia en los nodos. La tendencia de cada nodo, obligatoria en otras redes de optimización para garantizar la activación de un número determinado de nodos en la estabilización del sistema, no es necesaria en la red ODE debido a su esquema de procesamiento característico, en el que el número de nodos activos siempre permanece constante.

El siguiente teorema garantiza la estabilidad del optimizador discreto estocástico con el mecanismo de procesamiento antes descrito.

Teorema. Dada una red ODE de 2N + 1 nodos, que implementa un problema de optimización de n variables discretas de dominio p, y posee una función de energía dada por:

$$E(A) = -\sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij}a_i a_j$$

donde a_i es el estado del i-ésimo nodo visible principal de la red, se verifica que E es una función de Lyapunov, y el sistema es globalmente estable.

Demostración:

Se trata de comprobar que la función de energía de la red posee límite finito. Para ello hay que verificar que la función de energía está acotada y que su comportamiento es monotóno decreciente a partir de un cierto instante del procesamiento. En efecto, una cota inferior de la función de energía viene dada por la expresión:

$$E = -\sum_{i=1}^{N}\sum_{j=i+1}^{N} |w_{ij}|$$

Además, transcurrido un determinado número de iteraciones de procesamiento, cuando la temperatura T es suficientemente baja, el comportamiento de la red se vuelve determinista. A partir de entonces, si se acepta una transición de estado, el incremento de energía producido es igual a:

$$\Delta E = E(A(t+1)) - E(A(t)) = -\sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij}a_i(t+1)a_j(t+1) + \sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij}a_i(t)a_j(t)$$

De acuerdo con el esquema de procesamiento, A(t+1) es igual a B(t), siendo B el vector de estados de los nodos auxiliares, con lo que se tiene:

$$E(A(t+1)) = -\sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij}b_i(t)b_j(t) = \sum_{i=1}^{N}\sum_{j=i+1}^{N} -w_{ij}b_i(t)b_j(t)$$

y por tanto:

$$\Delta E = \sum_{i=1}^{N}\sum_{j=i+1}^{N} -w_{ij}b_i(t)b_j(t) + \sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij}a_i(t)a_j(t)$$

que es precisamente la entrada neta μ del nodo oculto en el instante t. Como μ es negativa por hipótesis, resulta que cualquier cambio en la función de energía es siempre negativo, es decir, la función de energía de la red, para una temperatura T próxima a cero, es monótona decreciente, como se quería demostrar.

Una observación importante acerca del procesamiento de la red ODE es que el cambio de energía producido por la aceptación de una transición, es igual a la entrada neta μ del nodo oculto. Por tanto, el comportamiento estocástico de este modo de procesamiento viene determinado por la razón entre el cambio de energía y la temperatura del sistema, como ocurre en el algoritmo de optimización del temple simulado. En conclusión, puede decirse que el procesamiento de la red ODE implementa una variante del algoritmo del temple simulado sobre una estructura de soluciones k-próximas en la búsqueda del estado de mínima energía de la red.

5

Análisis y Evaluación del Modelo ODE

Sine experientia, nihil sufficienter sciri potest.

Rogerio Bacon

5.1 IMPLEMENTACIÓN DE PROBLEMAS DE OPTIMIZACIÓN CON RESTRICCIONES MEDIANTE REDES NEURONALES.

5.1.1 Introducción.

La resolución de problemas de optimización combinatoria mediante redes neuronales recurrentes se fundamenta en un mecanismo, denominado *relajación paralela*, que se halla implícito en el modo de procesamiento de la red, y que está directamente relacionado con el paradigma conexionista de minimización de energía. Este paradigma interpreta a la red neuronal como un sistema energético que, partiendo de un estado inicial aleatorio, evoluciona en sentido decreciente, hasta alcanzar un estado estable de energía mínima.

En la resolución de problemas de optimización combinatoria no interesa la característica de aprendizaje propia de los modelos conexionistas, sino su arquitectura masivamente paralela y su modo de procesamiento recurrente. Es decir, en una aplicación de este tipo los pesos no se determinan mediante un algoritmo de aprendizaje, como suele suceder habitualmente, sino que se definen de forma explícita. Por tanto, el objetivo consiste en encontrar un esquema de interconexión, es decir, un conjunto de pesos, que permita resolver el problema a través del mecanismo de relajación paralela propio del procesamiento recurrente de la red.

La implementación de un problema de optimización combinatoria mediante una red neuronal recurrente implica realizar tres tareas importantes:

- La identificación de la arquitectura de la red con las variables y soluciones del problema.
- La transformación de la función objetivo y las restricciones del problema en una función de energía apropiada para el modelo neuronal utilizado.
- La derivación de los pesos de las conexiones a partir de dicha función de energía.

En las múltiples descripciones de sistemas conexionistas de optimización combinatoria aparecidas en la literatura [HEDG88], [TAGL89], [TAKE89], [AKIY89], [CULI90], [CHEN90], [LEE90], [LEE91], [ROFK92], [SCHA92], se aportan normas precisas sólo para algunos aspectos de este proceso de implementación. En general, los pesos de las conexiones, que representan los objetivos y restricciones del problema, se van ajustando de forma más o menos intuitiva a través de una serie de pruebas que se repiten hasta que la red se comporta satisfactoriamente, de manera similar a como ocurre con algunos algoritmos conexionistas de aprendizaje. Sin embargo, este proceso de ajuste no es realmente útil, dado que los pesos obtenidos no son válidos, por lo general, más que para la instancia del problema utilizada en la obtención de los mismos.

Por la razón anterior, uno de los objetivos de la investigación realizada en esta tesis se ha centrado en el estudio del comportamiento dinámico de las redes recurrentes a fin de obtener criterios exactos para la implementación de problemas de optimización combinatoria.

En los siguientes apartados se describe el conjunto de tareas necesarias para la implementación del problema genérico de optimización combinatoria en una red neuronal recurrente. Es importante destacar que esta descripción se presenta como un *procedimiento metódico de diseño*, en lugar de ser una técnica ad-hoc de ensayo y error.

Así mismo, con el fin de clarificar el proceso de diseño descrito, cada una de las tareas a desarrollar se concreta en los ejemplos del problema del viajante comercial, el problema de la partición de grafos, y el problema de la asignación cuadrática, los cuales se definieron formalmente en el capítulo 2.

5.1.2 Representación del problema en la estructura de la red.

El primer paso en el proceso de implementación del problema de optimización combinatoria en una red neuronal recurrente es la identificación específica de los nodos de la red con las variables del problema y sus dominios respectivos. Dado que las variables de un problema de optimización combinatoria suelen poseer el mismo dominio D, en la siguiente descripción se asume esta hipótesis, sin que por ello el método pierda generalidad.

La representación del problema en la estructura de la red implica establecer una correspondencia o asociación entre los diferentes estados de la red y las soluciones del problema de optimización. Esta representación se realiza de la siguiente forma: Si el problema posee n variables $X=\{x_1, ..., x_n\}$ de dominio común D, entonces se define una red con n vectores de nodos, un vector por cada variable, y con tantos nodos por vector como valores diferentes posea el dominio común D. Por tanto, el número total N de nodos de la red será igual a:

$$N = n \cdot D$$

De esta forma, cada nodo de la red representa un posible valor de una de las variables del problema. Generalmente, los nodos se disponen en forma de matriz, de manera que las filas representan las variables del problema, y las columnas los posibles valores del dominio.

Ejemplo 1. Sea una instancia del *problema del viajante comercial*, en el que se trata de minimizar la distancia de recorrido cíclico de n ciudades. Las n variables del problema $X=\{x_1, ..., x_n\}$ son las posiciones del recorrido, y cada variable posee un dominio D de cardinalidad n, es decir, puede tomar n valores diferentes, correspondientes a las n ciudades a recorrer. La arquitectura apropiada para representar este problema es la de una red de $N = n^2$ nodos dispuestos en forma de matriz cuadrada, en la que las n filas representan las variables del problema, es decir, las posiciones del recorrido, y las n columnas los valores posibles de las variables, esto es, las ciudades a visitar. En la figura 5.1 se representa la matriz de nodos correspondiente al problema del viajante, indicando los nodos activos mediante círculos negros y los nodos inactivos mediante círculos blancos.

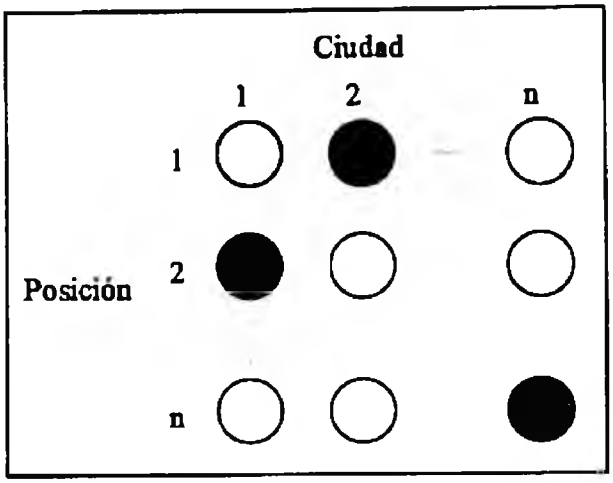

Figura 5.1: Arquitectura de red para el problema del viajante.

Ejemplo 2. Sea una instancia del *problema de la partición de grafos*, en el que se trata de dividir un grafo plano de n vertices en p particiones equilibradas (con n/p vértices cada una) de forma que se minimice el número de conexiones entre vértices de particiones distintas. Las n variables del problema $X=\{x_1, ..., x_n\}$ son los vértices del grafo, y cada variable posee un dominio D de cardinalidad p, es decir, puede tomar p valores diferentes, correspondientes a las p particiones a formar. La arquitectura apropiada para representar este problema es la de una red de N = n x p nodos dispuestos en forma de matriz, en la que las n filas representan las variables del problema, es decir, los vértices del grafo, y las p columnas los valores posibles de las variables, esto es, las particiones a formar (figura 5.2).

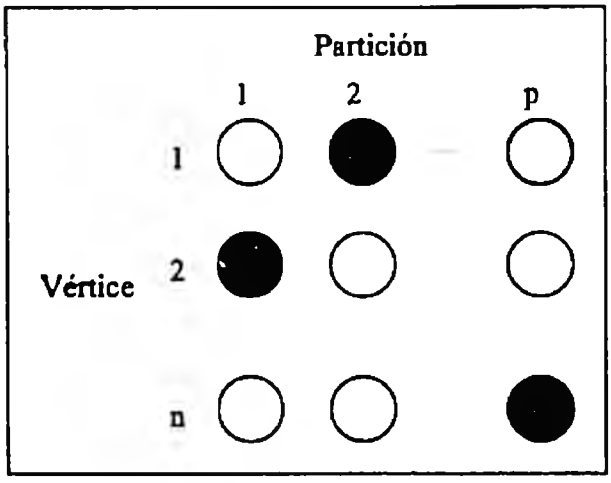

Figura 5.2: Arquitectura de red para el problema de la partición de grafos.

Ejemplo 3. Sea una instancia del *problema de la asignación cuadrática*, en el que se trata de situar n plantas de proceso en p lugares posibles ($p \geq n$), sabiendo que entre cada

dos plantas de proceso se debe transportar una cantidad diferente de material con un coste unitario distinto según el lugar donde se localicen éstas, de forma que se minimice el coste total del transporte de materiales. Las n variables del problema $X=\{x_1, ..., x_n\}$ son las plantas de proceso, y cada variable posee un dominio D de cardinalidad p, es decir, puede tomar p valores diferentes, correspondientes a los p lugares de ubicación. La arquitectura apropiada para representar este problema es la de una red de N = n x p nodos dispuestos en forma de matriz, en la que las n filas representan las variables del problema, es decir, las plantas de proceso, y las p columnas los valores posibles de las variables, esto es, los lugares de ubicación (figura 5.3).

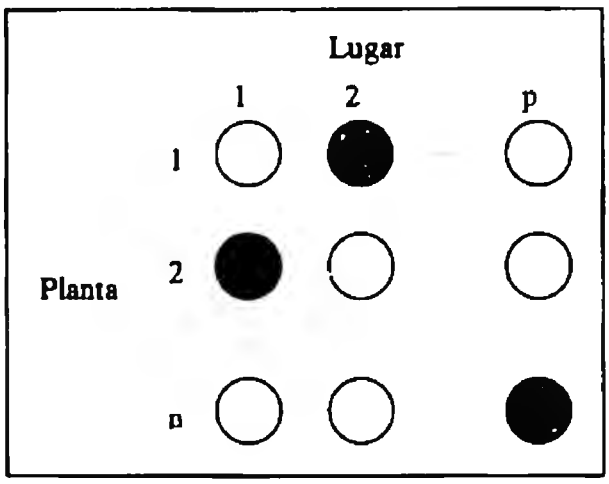

Figura 5.3: Arquitectura de red para el problema de la asignación cuadrática.

5.1.3 Determinación de la función de energía.

La *principal* tarea a desarrollar en el diseño e implementación de un problema de optimización combinatoria en una red neuronal recurrente consiste en la determinación de la función de energía E que mejor represente los objetivos y restricciones de la instancia del problema a resolver. En general, la función de energía de los modelos neuronales utilizados en optimización combinatoria adopta la forma siguiente:

$$E(A) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}a_i a_j - \sum_{i=1}^{N} I_i a_i$$

donde $A = (a_1, ..., a_n)$ es el vector de estados de los nodos de la red, w_{ij} es el peso de la conexión entre los nodos i y j, e I_i es la tendencia del nodo i. Los estados a_i de los nodos varían en el tiempo de acuerdo con la evolución del sistema, mientras que los pesos w_{ij} de las conexiones son constantes, es decir, constituyen los coeficientes fundamentales de

la función de energía, por lo que deben fijarse explícitamente de forma previa a la utilización de la red.

Como condición necesaria para la correcta derivación de los pesos de la red, se debe descomponer la función de energía E en varios términos, identificando cada uno de ellos con uno de los factores a considerar en la implementación del problema de optimización. Los diferentes factores que deben tenerse en cuenta en la definición de la función de energía son:

- La obtención de un estado final estable de la red con el número exacto de nodos activos que se precise para que la solución sea coherente.
- Las restricciones a implementar en la red.
- Los componentes de la función objetivo del problema.

La determinación de la función de energía de la red depende de todos estos factores, por lo que se estudian en detalle en los párrafos siguientes.

Obtención de K nodos activos sobre N en la estabilización de la red

En la implementación de un problema de optimización mediante una red neuronal, normalmente los pesos w_{ij} de las conexiones adoptan valores negativos (pesos inhibidores), mientras que la tendencia I_i de cada nodo es positiva para contrarrestar el efecto inhibidor de los anteriores. Así, para conseguir que en la estabilización de la red existan K nodos activos de un total de N es necesario establecer un equilibrio entre los valores de los pesos de las conexiones que recibe cada nodo y la tendencia del mismo.

Basándose en los estudios de Tagliarini y Page sobre la implementación de problemas de restricciones en la red de Hopfield continua [TAGL87] [TAGL89], Kahng proporciona la siguiente fórmula que determina la relación entre los pesos de las conexiones y la tendencia de cada nodo para conseguir K nodos activos de N en la estabilización de la red [KAHN89]:

$$I = -WK - 1$$

donde I es la tendencia global común a todos los nodos la red, y W es el peso común a todas las conexiones. Sin embargo, esta fórmula es incompleta, como se demuestra en la siguiente proposición relativa a la activación final de K nodos de N en la estabilización de

cualquier red recurrente simétrica que posea una función de umbral con valores de saturación 0-1.

Proposición. Dada una red neuronal simétrica de N nodos que posea una función de umbral con niveles de saturación 0-1, si los pesos de las conexiones son negativos y constantes, $w_{ij} = W$, y la tendencia es igualmente constante para todos los nodos, $I_i = I$, la condición necesaria para que K nodos ($0 < K < N$) queden activos en la estabilización del sistema es que se verifique la relación:

$$-W(K-1) < I < -WK$$

Demostración:

Por reducción al absurdo, supóngase que la relación anterior no se verifica. Pueden darse dos casos:

a) Sea $I < -W(K-1)$, y supóngase que ya existen K nodos activos, con estado igual o próximo a 1, y N-K nodos inactivos, con estado igual o próximo a 0. Para que la red sea estable, el cálculo del estado de cualquier nodo no debe suponer un cambio significativo con respecto al estado actual. Sin embargo, si se calcula el estado a_j de uno de los nodos activos mediante la ecuación:

$$a_j = f(\mu) = f\left(\sum_{i=1}^{N} W a_i + I\right)$$

como el sumatorio de los estados de ios nodos distintos de j ponderados por los pesos W es igual o próximo a W(K-1), se tiene que:

$$\sum_{i=1}^{N} W a_i \cong W(K-1) \Rightarrow \mu = \sum_{i=1}^{N} W a_i + I \cong W(K-1) + I < W(K-1) - W(K-1) = 0$$

es decir, la entrada neta μ del nodo j es negativa, con lo que el nuevo estado de activación del nodo sería inactivo, obteniéndose un total de K-1 nodos activos en la red. Por tanto, $I > -W(K-1)$.

b) Sea $I > -WK$, y supóngase que ya existen K nodos activos, con estado igual o próximo a 1, y N-K nodos inactivos, con estado igual o próximo a 0. Para que la red sea estable, el cálculo del estado de cualquier nodo no debe suponer un cambio significativo con respecto al estado actual. Sin embargo, si se calcula el estado a_j de uno de los nodos inactivos mediante la ecuación:

$$a_j = f(\mu) = f\left(\sum_{i=1}^{N} Wa_i + I\right)$$

como el sumatorio de los estados de los nodos distintos de j ponderados por los pesos W es igual o próximo a WK, se tiene que:

$$\sum_{i=1}^{N} Wa_i \cong WK \Rightarrow \mu = \sum_{i=1}^{N} Wa_i + I \cong WK + I > WK - WK = 0$$

es decir, la entrada neta μ del nodo j es positiva, con lo que el nuevo estado de activación del nodo sería activo, obteniéndose un total de K+1 nodos activos en la red. Por tanto, $I < -WK$.

En conclusión, para que en la estabilización de la red queden K nodos activos de N, es necesario que se verifique la relación:

$$-W(K-1) < I < -WK$$

Observación. Como consecuencia de la proposición anterior se observa intuitivamente que el valor ideal de I con respecto a W para conseguir K nodos activos de N en la estabilización de la red sería la media de los valores -W(K-1) y -WK:

$$I = \frac{-W(2K-1)}{2}$$

Puede observarse que la fórmula proporcionada por Tagliarini y Page [TAGL89], y Kahng [KAHN89], también es válida en la mayoría de los casos, pero se distancia del valor medio ideal de I cuanto mayor es el peso W de las conexiones de la red. Esto puede provocar una estabilización incorrecta de la red, especialmente en los modelos con función de umbral sigmoidal y valores de activación reales, como la red de Hopfield continua, la máquina de Gauss o la red de temple determinista.

La proposición anterior proporciona condiciones necesarias para la discriminación de K nodos activos en la estabilización de la red. Además, para cualquier W negativo la relación

$$I = \frac{-W(2K-1)}{2}$$

es también condición suficiente de estabilización correcta en las redes discretas con valores de activación binarios, como la máquina de Boltzmann o la máquina de Cauchy. Sin embargo, no garantiza dicha discriminación en las redes con valores de activación reales, sino que en este caso se precisa de una condición adicional sobre el valor de

inhibición W de los pesos. En efecto, en las pruebas realizadas mediante la simulación de la red de Hopfield continua (figuras 5.4, 5.5 y 5.6) se observa que, para valores de W negativos y próximos a cero, la red se estabiliza en un estado en que todos los nodos toman un mismo valor en el intervalo]0,1[. Al disminuir el valor de W, el estado final de los nodos de la red va aproximándose al valor K/N, punto a partir del cual sí se produce la discriminación en K nodos activos de la red.

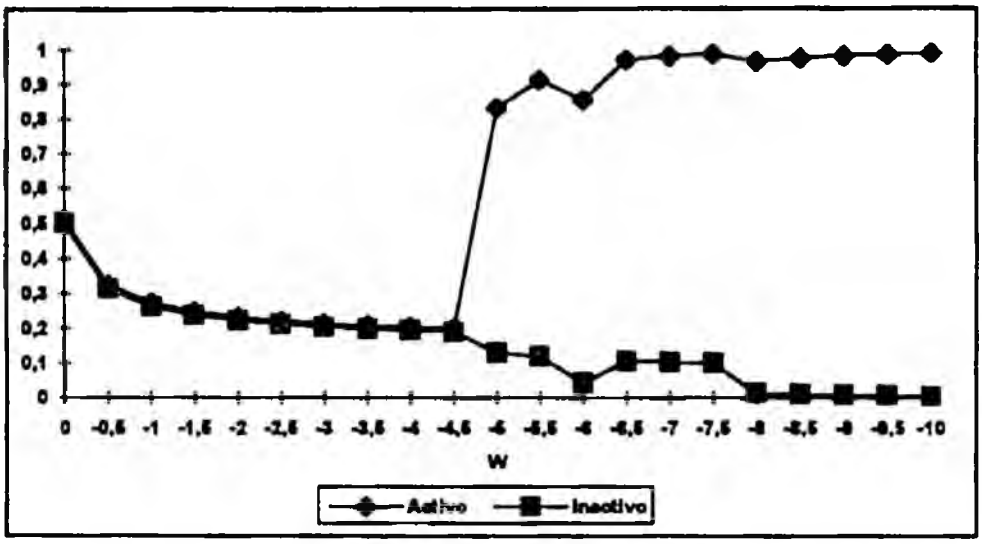

Figura 5.4: Discriminación de estados activo-inactivo 2/10 en una red CH de 10 nodos.

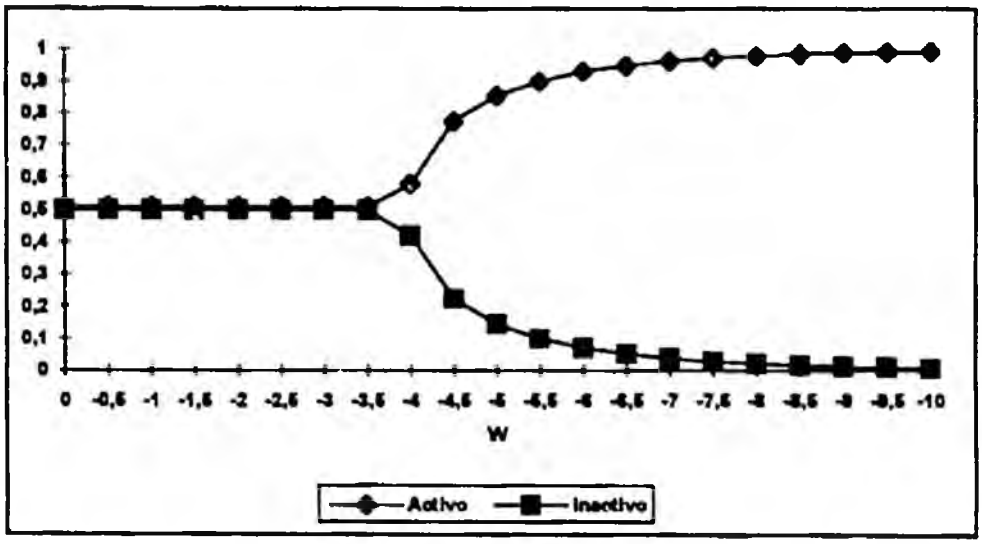

Figura 5.5: Discriminación de estados activo-inactivo 5/10 en una red CH de 10 nodos.

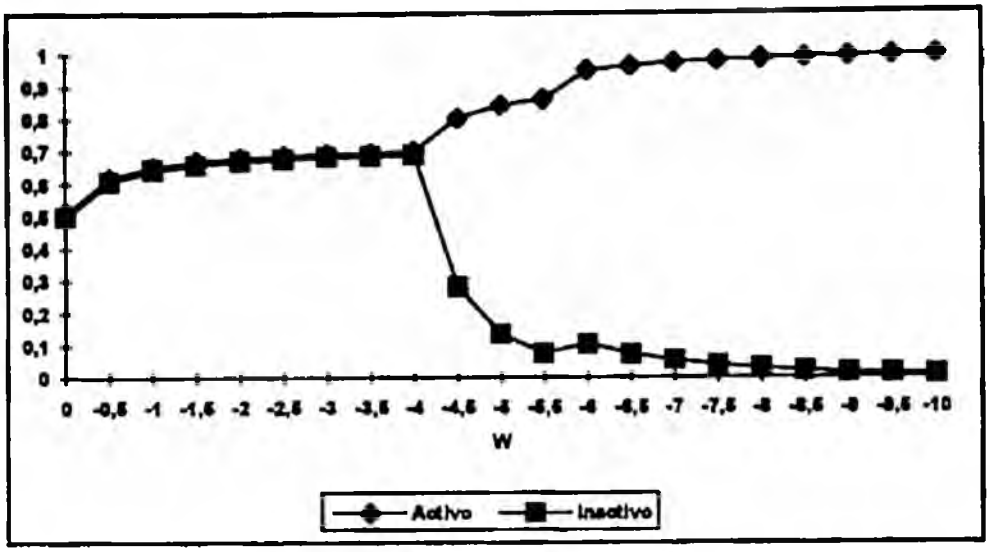

Figura 5.6: Discriminación de estados activo-inactivo 7/10 en una red CH de 10 nodos.

La siguiente proposición permite decidir sobre este valor crítico de W en redes con función de umbral sigmoidal logística

$$f(\mu)=\frac{1}{1+e^{-\mu}}$$

cuando $K \neq N/2$.

Proposición. Dada una red neuronal simétrica de N nodos que posea una función de umbral sigmoidal logistica, si los pesos de las conexiones son negativos y constantes, $w_{ij} = W$, y la tendencia es igualmente constante para todos los nodos, $I_i = I$, verificándose que

$$I=\frac{-W(2K-1)}{2}$$

entonces el valor de W a partir del que se produce la discriminación en K nodos activos de la red ($K \neq N/2$) viene dado por la expresión:

$$W=-\frac{2N}{N-2K}\,\mathrm{Ln}\left(\frac{N-K}{K}\right)$$

Demostración:

El valor de W buscado hace que la red permanezca estabilizada en el valor K/N para todos los nodos, es decir, verifica la siguiente ecuación:

$$\forall j: \quad a_j = \frac{1}{1+e^{-\left(\sum_{i=1}^{N} Wa_i + I\right)}} = \frac{K}{N}$$

por tanto, como

$$\sum_{i=1}^{N} Wa_i = \frac{WK(N-1)}{N} \qquad I = \frac{-W(2K-1)}{2}$$

se tiene que:

$$\frac{1}{1+e^{-\left(\frac{WK(N-1)}{N} \; \frac{W(2K-1)}{2}\right)}} = \frac{K}{N}$$

$$\frac{1}{1+e^{-W\left(\frac{N-2K}{2N}\right)}} = \frac{K}{N}$$

$$e^{-W\left(\frac{N-2K}{2N}\right)} = \left(\frac{N-K}{K}\right)$$

$$-W\left(\frac{N-2K}{2N}\right) = \mathrm{Ln}\left(\frac{N-K}{K}\right)$$

y, finalmente

$$W = -\frac{2N}{N-2K}\,\mathrm{Ln}\left(\frac{N-K}{K}\right)$$

como se quería demostrar.

Observación. Para K = N/2, el valor de W puede obtenerse simplemente calculando el límite cuando K tiende a N/2 de la expresión anterior:

$$\lim_{K\to\frac{N}{2}} -\frac{2N}{N-2K}\,\mathrm{Ln}\left(\frac{N-K}{K}\right) = \lim_{K\to\frac{N}{2}} -\frac{2N}{N-2K}\,\mathrm{Ln}\left(1+\frac{N-2K}{K}\right) =$$

$$\lim_{K\to\frac{N}{2}} -\left(\frac{2N}{N-2K}\right)\left(\frac{N-2K}{K}\right) = \lim_{K\to\frac{N}{2}} -\frac{2N}{K} = \frac{-2N}{\frac{N}{2}} = -4$$

Las expresiones obtenidas corresponden al valor crítico de W para una red simétrica con función de umbral sigmoidal logística. No obstante, las pruebas realizadas muestran que para que se garantice siempre en la práctica la discriminación de K nodos activos en una red de este tipo, es necesario que el valor de W sea inferior al valor crítico en al menos N unidades, es decir:

$$W = -\frac{2N}{N-2K} Ln\left(\frac{N-K}{K}\right) - N$$

Así mismo, es posible obtener expresiones análogas de forma equivalente para cualquier otro modelo neuronal que posea una función de umbral sigmoidal diferente.

El estudio que se ha realizado sobre la relación existente en una red simétrica entre los pesos de las conexiones y la tendencia de cada nodo, con respecto al problema de la discriminación de K nodos activos en la estabilización de la red, lleva a las siguientes conclusiones:

- El equilibrio necesario para conseguir que un número K de nodos de la red quede activo sobre el total N de nodos en la estabilización de la red, requiere que los pesos de las conexiones sean inhibidores (negativos), y la tendencia de cada nodo positiva, a fin de contrarrestar el efecto inhibidor de los primeros.
- Suponiendo que los pesos son todos iguales a un valor negativo cualquiera W y que la tendencia I es común para todos los nodos de la red, la relación que debe existir entre W e I para conseguir la discriminación de K nodos activos de N viene dada por la expresión:

$$I = \frac{-W(2K-1)}{2}$$

- Adicionalmente, en las redes de tipo continuo es necesario proporcionar un valor máximo de inhibición W, a fin de que la red se estabilice en el estado correcto.

Con respecto a la determinación de la función de energía de la red en el proceso de implementación del problema de optimización combinatoria, de lo anterior se deduce que ésta deberá poseer un término encargado de garantizar la discriminación correcta de nodos activos/inactivos en el estado final del procesamiento de la red. Este término, denominado término de *inhibición global*, se define como

$$-\frac{1}{2} W \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j$$

donde W es un valor real negativo que depende de las características del problema. La magnitud de la inhibición global se determina de forma exacta en el momento de la derivación de los pesos de la red, una vez que se hayan identificado los restantes términos componentes de la función de energía. Así mismo, se establece como componente de *tendencia* de la función de energía para cada nodo, el valor:

$$-\sum_{i=1}^{N} I_i a_i = \frac{W(2K-1)}{2} \sum_{i=1}^{N} a_i$$

Restricciones

Las restricciones que deben implementarse en la red pueden dividirse en tres clases:

- *Restricción estructural de la red.* Esta restricción se debe a la forma de representar el problema en la arquitectura de la red, es decir, a la forma de identificar los nodos con las variables del problema. Esta representación se realiza mediante una matriz de nodos en la que cada fila representa una variable discreta del problema, y cada columna un posible valor del dominio. Esto implica que, para que el estado final de la red corresponda a una solución coherente, es necesario que ninguna fila de la matriz posea más de un nodo activo. Lo contrario significaría que a una misma variable del problema se le asignan varios valores diferentes, lo cual es imposible.
- *Restricciones entre las variables del problema.* Corresponden a las restricciones r_i del problema original, que determinan los valores de compatibilidad entre subconjuntos específicos de las variables del problema. Cada restricción r_i es un conjunto de tuplas de valores compatibles que las variables relacionadas pueden tomar simultáneamente en una solución válida.
- *Restricciones especiales sobre el dominio de las variables.* Generalmente, en un problema de optimización combinatoria se asume que todas las variables poseen el mismo dominio D. En el caso de que exista alguna variable x_i del problema que no pueda tomar todos los valores posibles del dominio común D, se dice que el problema presenta una restricción de dominio sobre la variable x_i.

Debido al carácter mutuamente inhibidor de los nodos, en las redes neuronales de optimización las restricciones deben entenderse como conjuntos de nodos incompatibles entre sí, es decir, conjuntos de nodos que no pueden aparecer simultáneamente activos en el estado final de la red para que la solución obtenida sea válida. Dado que los nodos se comportan como elementos que compiten entre sí, inhibiéndose constantemente, una forma de conseguir la no activación simultánea de nodos incompatibles es hacer más negativa la inhibición existente entre ellos, es decir, aumentar en valor absoluto el peso de las conexiones que unen nodos no compatibles, con respecto a la inhibición global W común a todas las conexiones de la red.

Por otra parte, las restricciones de dominio son restricciones unarias que afectan a un solo nodo de la red, el que representa el valor restringido de la variable, por lo que en este caso no puede hablarse de nodos incompatibles entre sí. No obstante, la implementación de las restricciones de dominio en una red neuronal es sencilla: basta con eliminar la tendencia I del nodo afectado por la restricción, con lo que la inhibición global W que recibe constantemente este nodo por sus conexiones de entrada producirá siempre la desactivación del mismo en la estabilización de la red.

Con respecto a las restricciones de incompatibilidad, las dos proposiciones siguientes indican cómo debe ser la inhibición entre nodos incompatibles con respecto a la inhibición global W, a fin de conseguir la no activación simultánea de este tipo de nodos en la estabilización de la red.

Proposición. Sea una red neuronal simétrica de N nodos que posee una función de umbral con valores de saturación 0-1, una inhibición global W, y una tendencia I, verificándose que

$$I = \frac{-W(2K-1)}{2}$$

donde K es el número total de nodos que deben quedar activos en la estabilización de la red. Si dos nodos i y j de la red son incompatibles, para conseguir que sólo uno de ellos quede activo al final del procesamiento, los pesos w_{ij} y w_{ji} de las conexiones simétricas que los unen deben verificar la siguiente relación:

$$w_{ij} = w_{ji} < \tfrac{3}{2}W$$

Demostración:

Por reducción al absurdo, supóngase que la relación anterior no se verifica, es decir

$$w_{ij} = w_{ji} > \tfrac{3}{2}W$$

y supóngase que ya existen K nodos activos, con estado igual o próximo a 1, entre los que están los dos nodos incompatibles. De acuerdo con esto, un nuevo cálculo del estado a_j de uno de los dos nodos activos afectados por la restricción debería producir su desactivación, es decir, la entrada neta del nodo j debe verificar:

$$\mu_j = \sum_{i=1}^{N} w_{ij} a_i + I < 0$$

y como el sumatorio de los estados de los nodos distintos de j ponderados por los pesos respectivos es igual o próximo al valor

$$v+W(K-2)$$

donde v es el peso de la conexión que une los nodos incompatibles, se tiene que:

$$\mu_j = \sum_{i=1}^{N} w_{ij}a_i + I \cong v+W(K-2)-\frac{W(2K-1)}{2} < 0$$

y de aquí:

$$v < \frac{W(2K-1)-2W(K-2)}{2} = \tfrac{3}{2}W$$

lo que contradice la hipótesis:

$$v = w_{ij} = w_{ji} > \tfrac{3}{2}W$$

Luego, se verifica la relación

$$w_{ij} = w_{ji} < \tfrac{3}{2}W$$

como se quería demostrar.

La siguiente proposición puede verse como una generalización de la anterior, y se refiere a la incompatibilidad parcial entre un subjunto de M nodos de la red, de los que deben quedar exactamente P ($0 < P \leq K$) activos al finalizar el procesamiento.

Proposición. Sea una red neuronal simétrica de N nodos que posee una función de umbral con valores de saturación 0-1, una inhibición global W, y una tendencia I, verificándose que

$$I = \frac{-W(2K-1)}{2}$$

donde K es el número total de nodos que deben quedar activos en la estabilización de la red. Si en un subconjunto de M nodos de la red parcialmente incompatibles deben quedar P ($0 < P \leq K$) nodos activos al final del procesamiento, entonces los pesos w_{ij} y w_{ji} de las conexiones simétricas que unen cada par de nodos i y j de este subconjunto, deben verificar la siguiente relación:

$$\frac{W(2P-1)}{2(P-1)} < w_{ij} = w_{ji} < \frac{W(2P+1)}{2P}$$

Demostración:

Por reducción al absurdo, supóngase que la relación anterior no se verifica. Pueden darse dos casos:

a) Sea

$$w_{ij} = w_{ji} > \frac{W(2P+1)}{2P}$$

y supóngase que ya existen K nodos activos, con estado igual o próximo a 1, de los cuales P+1 corresponden al subconjunto de M nodos parcialmente incompatibles. De acuerdo con esto, un nuevo cálculo del estado a_j de un nodo elegido entre los P+1 nodos activos afectados por la restricción debería producir su desactivación, es decir, la entrada neta del nodo j debe verificar:

$$\mu_j = \sum_{i=1}^{N} w_{ij} a_i + I < 0$$

y como el sumatorio de los estados de los nodos distintos de j ponderados por los pesos respectivos es igual o próximo al valor

$$vP + W(K - P - 1)$$

donde v es el peso común de las conexiones que unen los M nodos parcialmente incompatibles, se tiene que:

$$\mu_j = \sum_{i=1}^{N} w_{ij} a_i + I \cong vP + W(K - P - 1) - \frac{W(2K-1)}{2} < 0$$

y de aquí:

$$v < \frac{W(2K-1) - 2W(K-P-1)}{2P} = \frac{W(2P+1)}{2P}$$

lo que contradice la hipótesis:

$$v = w_{ij} = w_{ji} > \frac{W(2P+1)}{2P}$$

luego se verifica que:

$$w_{ij} = w_{ji} < \frac{W(2P+1)}{2P}$$

b) Sea

$$w_{ij} = w_{ji} < \frac{W(2P-1)}{2(P-1)}$$

y supóngase que existen K-1 nodos activos, con estado igual o próximo a 1, de los cuales P-1 corresponden al subconjunto de M nodos parcialmente incompatibles. De acuerdo con esto, un nuevo cálculo del estado a_j de un nodo elegido entre los M-P+1 nodos inactivos afectados por la restricción debería producir su activación, es decir, la entrada neta del nodo j debe verificar:

$$\mu_j = \sum_{i=1}^{N} w_{ij} a_i + I > 0$$

y como el sumatorio de los estados de los nodos distintos de j ponderados por los pesos respectivos es igual o próximo al valor

$$v(P-1)+W(K-P)$$

donde v es el peso común de la conexiones que unen los M nodos parcialmente incompatibles, se tiene que:

$$\mu_j = \sum_{i=1}^{N} w_{ij} a_i + I \cong v(P-1)+W(K-P)-\frac{W(2K-1)}{2} > 0$$

y de aquí:

$$v > \frac{W(2K-1)-2W(K-P)}{2(P-1)} = \frac{W(2P-1)}{2(P-1)}$$

lo que contradice la hipótesis:

$$v = w_{ij} = w_{ji} < \frac{W(2P-1)}{2(P-1)}$$

luego se verifica que:

$$w_{ij} = w_{ji} > \frac{W(2P-1)}{2(P-1)}$$

En conclusión, para que en la estabilización de la red queden K nodos activos, con P de ellos pertenecientes al subconjunto de M nodos afectados por la restricción, es necesario que se verifique la relación:

$$\frac{W(2P-1)}{2(P-1)} < w_{ij} = w_{ji} < \frac{W(2P+1)}{2P}$$

Observación. El valor v de los pesos de las conexiones que unen nodos afectados por la incompatibilidad parcial, debe verificar la relación anterior, o lo que es igual, debe verificar:

$$W + \frac{W}{2(P-1)} < v < W + \frac{W}{2P}$$

y, aplicando la regla de la fracción intermedia:

$$\frac{a}{b} < \frac{a}{e} \Rightarrow \frac{a}{b} < \frac{2a}{b+c} < \frac{a}{c}$$

resulta el valor de v más apropiado para la implementación de la restricción de incompatibilidad parcial del problema:

$$v = W + \frac{W}{2P-1} = W\frac{2P}{2P-1}$$

Así mismo, teniendo en cuenta que la proposición anterior es una generalización de la correspondiente a la restricción entre dos nodos incompatibles, puede particularizarse esta fórmula para P = 1, con lo que se obtiene el valor ideal v del peso de la conexión que une dos nodos mutuamente excluyentes:

$$v = W + \frac{W}{2-1} = 2W$$

es decir, el valor del peso de la conexión que une dos nodos incompatibles debe ser igual al doble de la inhibición global de la red.

Con respecto a la determinación de la función de energía de la red en el proceso de implementación del problema de optimización combinatoria, ésta deberá poseer un término encargado de representar la restricción estructural de la red, así como un término por cada una de las restricciones originales del problema:

- El término correspondiente a la restricción estructural de la red viene determinado por la expresión:

  $$-\frac{1}{2}Wa_ia_j$$

  donde a_i y a_j son dos nodos que representan a la misma variable, es decir, dos nodos de la misma fila de la matriz-red.

- El término correspondiente a una restricción de incompatibilidad entre dos nodos i y j mutuamente excluyentes viene determinado por la expresión:

  $$-\frac{1}{2}Wa_ia_j$$

- Finalmente, el término correspondiente a una restricción de incompatibilidad parcial entre un subconjunto de M nodos de la red viene determinado por la expresión:

$$-\frac{1}{2}\frac{W}{2P-1}a_i a_j$$

donde a_i y a_j son los estados de dos nodos cualesquiera del subconjunto de M nodos afectados por la restricción.

Observación. Estos términos de la función de energía correspondientes a las restricciones se suman al término asociado a la inhibición global, con lo que se completan los valores anteriormente propuestos para los pesos de las conexiones que unen nodos afectados por restricciones.

Objetivos

En general, la función objetivo a minimizar en un problema de optimización combinatoria se expresa, o puede expresarse, como la suma de una serie de términos cada uno de los cuales viene dado en función de una o de dos variables del problema. Es decir, una función objetivo genérica f adopta la forma:

$$f(s) = \sum_{i=1}^{n} g_i(x_i) + \sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} g_{ij}(x_i, x_j)$$

donde g_i son los términos objetivo dependientes de una sola variable x_i, y g_{ij} son los términos objetivo dependientes de dos variables x_i, x_j.

Cada uno de los términos de la función objetivo se implementa mediante un término adicional de la función de energía de la red, de la siguiente manera (en notación matricial de doble subíndice):

- Un término g_i de la función objetivo, dependiente de una sola variable x_i, se implementa añadiendo el siguiente término a la función de energía por cada par de nodos (i,j) y ((i mod n)+1,k) de la red:

$$\frac{1}{2}g_i(j)a_{ij}a_{(i \bmod n)+1,k}$$

donde a_{ij} es el estado del nodo j de la variable-fila i, $a_{(i \bmod n)+1,k}$ es el estado del nodo k de la siguiente variable-fila a i, es decir, la variable-fila (i mod n)+1, y $g_i(j)$ es el objetivo determinado por la variable x_i cuando toma el valor j. Como se

observa, el factor $g_i(j)$ corresponde al valor opuesto a sumar al peso de la conexión que une el nodo (i,j) de la matriz-red, con el nodo ((i mod n)+1,k) de la misma.

- Un término g_{ij} de la función objetivo, dependiente de dos variables x_i y x_j, se implementa añadiendo el siguiente término a la función de energía por cada par de nodos (i,k) y (j,l) de la red:

$$\frac{1}{2} g_{ij}(k,l) a_{ik} a_{jl}$$

donde a_{ik} es el estado del nodo k de la variable-fila i, a_{jl} es el estado del nodo l de la variable-fila j, y $g_{ij}(k,l)$ es el objetivo determinado por las variables x_i y x_j cuando toman respectivamente los valores k y l. Como se observa, el factor $g_{ij}(k,l)$ corresponde al valor opuesto a sumar al peso de la conexión que une el nodo (i,k) de la matriz-red, con el nodo (j,l) de la misma.

Finalmente, como resumen del apartado se proporcionan los siguientes ejemplos, que muestran la manera de realizar en la práctica el proceso de determinación de la función de energía de la red.

Ejemplo 1. Sea una instancia del *problema del viajante comercial*, en el que se trata de minimizar la distancia de recorrido cíclico de n ciudades. Las n variables del problema $X=\{x_1, ..., x_n\}$ son las posiciones del recorrido, y cada variable posee como dominio las n ciudades a visitar. La restricción r del problema viene dada por las permutaciones de las n ciudades, es decir, las soluciones válidas del problema corresponden a asignaciones de valores no repetidos a las n variables. La función objetivo a minimizar viene determinada por la expresión:

$$f(s) = \sum_{i=1}^{n} d\left(x_i, x_{(i \bmod n)+1}\right)$$

donde $d(x_i,x_j)$ es la distancia entre las ciudades x_i y x_j. La función de energía E de la red para este problema toma la expresión (en notación matricial de doble subíndice):

$$E = -\frac{1}{2}W\sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}a_{ik}a_{jl} - \frac{1}{2}W\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq j}}^{n}a_{ij}a_{ik} - \frac{1}{2}W\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq j}}^{n}a_{ji}a_{ki}$$

$$+\frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}d(k,l)a_{ik}a_{jl}\left(\delta(i,(j \bmod n)+1)+\delta((i \bmod n)+1,j)\right)$$

$$+\frac{W(2n-1)}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}$$

donde δ es la función delta de Kroenecker, definida:

$$\delta(i,j)=\begin{cases}1 & \text{si } i=j\\ 0 & \text{si } i\neq j\end{cases}$$

El primer término de la función de energía corresponde a la inhibición global. El segundo término corresponde a la restricción estructural de la red, y se minimiza cuando niguna fila de la matriz-red posee más de un nodo activo. El tercer término corresponde a la restricción del problema, variables con valores distintos, y se minimiza cuando ninguna columna de la matriz-red posee más de un nodo activo. El cuarto término corresponde al objetivo de la optimización, minimizar la distancia total recorrida. Finalmente, el quinto término corresponde a la tendencia global de la red.

Ejemplo 2. Sea una instancia del *problema de la partición de grafos*, en el que se trata de dividir un grafo plano de n vertices en p particiones equilibradas (con n/p vértices cada una) de forma que se minimice el número de conexiones entre vértices de particiones distintas. Las n variables del problema $X=\{x_1, ..., x_n\}$ son los vértices del grafo, y cada variable posee como dominio las p particiones a formar. La restricción r del problema viene dada por las combinaciones equilibradas de valores de las variables, es decir, las soluciones en las que cada valor se repite exactamente en n/p variables. La función objetivo a minimizar viene determinada por la expresión:

$$f(s)=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\gamma\left(x_i,x_j\right)\cdot\phi(i,j)$$

donde γ es una función binaria que devuelve 1 si $x_i \neq x_j$ (los vértices i y j pertenecen a diferentes particiones) y 0 en caso contrario, y ϕ es una función binaria que devuelve 1 si el vértice i está conectado al j, y 0 en caso contrario. La función de energía E de la red para este problema toma la expresión (en notación matricial de doble subíndice):

$$E = -\frac{1}{2}W\sum_{i=1}^{n}\sum_{k=1}^{p}\sum_{j=1}^{n}\sum_{l=1}^{p}a_{ik}a_{jl} - \frac{1}{2}W\sum_{i=1}^{n}\sum_{j=1}^{p}\sum_{\substack{k=1\\k\neq j}}^{p}a_{ij}a_{ik} - \frac{1}{2}\frac{W}{2n/p-1}\sum_{i=1}^{p}\sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq j}}^{n}a_{ji}a_{ki}$$

$$+\frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{p}\sum_{j=1}^{n}\sum_{l=1}^{p}\gamma(k,l)\phi(i,j)a_{ik}a_{jl} + \frac{W(2n-1)}{2}\sum_{i=1}^{n}\sum_{j=1}^{p}a_{ij}$$

El primer término de la función de energía corresponde a la inhibición global. El segundo término corresponde a la restricción estructural de la red, y se minimiza cuando ninguna fila de la matriz-red posee más de un nodo activo. El tercer término corresponde a la restricción del problema, n/p variables con el mismo valor, y se minimiza cuando existen exactamente n/p nodos activos en cada columna de la matriz-red. El cuarto término corresponde al objetivo de la optimización, minimizar el número de conexiones entre diferentes particiones. Finalmente, el quinto término corresponde a la tendencia global de la red.

Ejemplo 3. Sea una instancia del *problema de la asignación cuadrática*, en el que se trata de situar n plantas de proceso en p lugares posibles ($p \geq n$), sabiendo que entre cada dos plantas de proceso se debe transportar una cantidad diferente de material con un coste unitario distinto según el lugar donde se localicen éstas, de forma que se minimice el coste total del transporte de materiales. Las n variables del problema $X=\{x_1, ..., x_n\}$ son las plantas de proceso, y cada variable posee como dominio los p lugares posibles de ubicación. La restricción r del problema viene dada por las asignaciones de valores no repetidos a las n variables. La función objetivo a minimizar viene determinada por la expresión:

$$f(s) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c\left(x_i, x_j\right) \cdot q(i,j)$$

donde $c(x_i,x_j)$ es el coste unitario del transporte del material entre los lugares en que se hallan las plantas i y j, y q(i, j) es la cantidad de material que debe transportarse entre ambas. La función de energía E de la red para este problema toma la expresión (en notación matricial de doble subíndice):

$$E = -\frac{1}{2}W\sum_{i=1}^{n}\sum_{k=1}^{p}\sum_{j=1}^{n}\sum_{l=1}^{p}a_{ik}a_{jl} - \frac{1}{2}W\sum_{i=1}^{n}\sum_{j=1}^{p}\sum_{\substack{k=1\\k\neq j}}^{p}a_{ij}a_{ik} - \frac{1}{2}W\sum_{i=1}^{p}\sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq j}}^{n}a_{ji}a_{ki}$$

$$+\frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{p}\sum_{j=1}^{n}\sum_{l=1}^{p}c(k,l)q(i,j)a_{ik}a_{jl} + \frac{W(2n-1)}{2}\sum_{i=1}^{n}\sum_{j=1}^{p}a_{ij}$$

El primer término de la función de energía corresponde a la inhibición global. El segundo término corresponde a la restricción estructural de la red, y se minimiza cuando

ninguna fila de la matriz-red posee más de un nodo activo. El tercer término corresponde a la restricción del problema, variables con valores distintos, y se minimiza cuando ninguna columna de la matriz-red posee más de un nodo activo. El cuarto término corresponde al objetivo de la optimización, minimizar el coste total del transporte de materiales. Finalmente, el quinto término corresponde a la tendencia global de la red.

5.1.4 Derivación de los pesos de las conexiones.

Una vez determinada la expresión completa de la función de energía de la red, de acuerdo con las características del problema de optimización combinatoria, la derivación de las expresiones de los pesos w_{ij} se realiza de forma inmediata de acuerdo con el criterio que se describe a continuación. Sea w_{ij} el peso de la conexión que une los nodos i y j de la red. Los componentes que forman parte del peso w_{ij} son:

1. La inhibición global W, derivada del primer término de la función de energía.
2. Si los nodos i y j pertenecen a la misma fila de la matriz-red, es decir, si están asociados a la misma variable, el peso w_{ij} se debe incrementar en la magnitud W, derivada del término energético de restricción estructural de la red.
3. Si los nodos i y j son mutuamente excluyentes, el peso se debe incrementar en la magnitud W, derivada del término energético de restricción por incompatibilidad total de los nodos.
4. Si los nodos i y j son parcialmente excluyentes, es decir, pertenecen a un grupo de M nodos de los que deben quedar P activos, el peso se debe incrementar en la magnitud

$$\frac{W}{2P-1}$$

   derivada del término energético de restricción por incompatibilidad parcial de los nodos.
5. Si los nodos i y j corresponden a variables que intervienen en algún término de la función objetivo, el peso se debe decrementar en la magnitud g_{ij} o coste proporcionado por los valores asociados a los nodos i y j, derivada del término energético objetivo correspondiente.

Lógicamente, los casos 2 al 5 son excluyentes entre si. Una vez que se han derivado las expresiones correspondientes a los pesos de las conexiones, es necesario determinar la

magnitud de la inhibición global W de la red, con el fin de asignar el valor numérico apropiado a cada peso. El objetivo de la inhibición global W, en colaboración con la tendencia I, es normalizar la señal de entrada μ_j que recibe cada nodo j de la red, de manera que el efecto inhibidor de los pesos w_{ij}, compensado con el efecto excitador de la tendencia I, permita la estabilización de la red en un estado final con un número exacto K de nodos activos.

En la determinación de la función de energía, los términos definidos para representar las restricciones y la tendencia, dependen todos ellos directamente del valor de inhibición global W. Sin embargo, los términos correspondientes a los objetivos de optimización sólo dependen de las características del problema (distancias, costes, tiempos), y no de la inhibición global W. Por esta razón, si se proporcionase a W un valor negativo arbitrario, las magnitudes de los pesos afectados por los objetivos no tendrían por qué guardar proporción alguna con el valor de W, alterándose completamente el equilibrio necesario entre los pesos w_{ij} de las conexiones y la tendencia I de cada nodo para conducir a la red a un estado final coherente.

La siguiente proposición se refiere a la magnitud que debe tomar la inhibición global W con respecto a los objetivos de optimización, para conseguir la estabilización correcta de la red al finalizar el procesamiento.

Proposición. Sea una red neuronal simétrica de N nodos que posee una función de umbral con valores de saturación 0-1, una inhibición global W, y una tendencia I, verificándose que

$$I = \frac{-W(2K-1)}{2}$$

donde K es el número total de nodos que deben quedar activos en la estabilización de la red. La magnitud que debe tomar la inhibición global W de los pesos de las conexiones con respecto a los objetivos de optimización, para conseguir una estabilización correcta de la red en K nodos activos de N, verifica la siguiente relación:

$$W < 2 \cdot \min\left\{-g_j \middle| j = 1,\ldots,N\right\}$$

donde $-g_j$ es la inhibición negativa, debida a objetivos de optimización, que afectaría al nodo j de la red a través de los pesos w_{ij} de las conexiones que recibe de los restantes nodos, si la red se hallase en un estado correspondiente a una solución factible (con K nodos compatibles activos). Es decir, la inhibición global W debe ser menor que el doble

de la inhibición negativa mínima, debida a objetivos de optimización, que puede recibir cualquier nodo de la red en un estado correspondiente a una solución válida.

Demostración:

Por reducción al absurdo, supóngase que la relación anterior no se verifica, es decir

$$W > 2 \cdot \min\{-g_j | j = 1,\ldots,N\}$$

y supóngase que ya existen K nodos activos, correspondientes a una solución válida, siendo uno de ellos el nodo j que posee la menor inhibición negativa debida a objetivos de optimización. De acuerdo con esto, un nuevo cálculo del estado a_j de este nodo no debería producir su desactivación, es decir, la entrada neta del nodo j debe verificar:

$$\mu_j = \sum_{i=1}^{N} w_{ij} a_i + I > 0$$

y como el sumatorio de los estados de los nodos distintos de j ponderados por los pesos respectivos es igual o próximo al valor

$$W(K-1) + \min\{-g\}$$

donde min{-g} es la inhibición que recibe el nodo j respecto a los objetivos de optimización, se tiene que:

$$\mu_j = \sum_{i=1}^{N} w_{ij} a_i + I \cong W(K-1) + \min\{-g\} - \frac{W(2K-1)}{2} > 0$$

y de aquí:

$$\frac{2W(K-1) - W(2K-1)}{2} + \min\{-g\} > 0$$

$$\frac{2KW - 2W - 2KW + W}{2} + \min\{-g\} > 0$$

$$\frac{-W}{2} + \min\{-g\} > 0$$

$$W < 2 \cdot \min\{-g\}$$

lo que contradice la hipótesis:

$$W > 2 \cdot \min\{-g_j | j = 1,\ldots,N\}$$

Luego, se verifica la relación

$$W < 2 \cdot \min\{-g_j | j = 1,\ldots,N\}$$

como se quería demostrar.

Normalmente, la inhibición global W toma como valor una cota inferior del doble de la inhibición negativa -g, debida a objetivos de optimización, que puede recibir un nodo cualquiera de la red en un estado correspondiente a una solución factible.

Intuitivamente, la regla anterior puede explicarse de la siguiente forma: Es evidente que el valor crítico de inhibición total que puede recibir un nodo activo en una situación correcta de la red es de W(K-1) + 1/2W, ya que con un valor más negativo la tendencia 1 no es suficiente para contrarrestar la inhibición y mantener activo el nodo. Por tanto, ante un estado de la red correspondiente a una solución factible, si la inhibición que recibe un nodo activo debida a objetivos de optimización es superior en valor absoluto a la mitad de la inhibición global, entonces la inhibición total que recibe el nodo es menor que el valor crítico W(K-1) + 1/2W, y por tanto el nodo tendería a desactivarse, produciendo un estado incorrecto de la red.

Ejemplo 1. Sea una instancia del *problema del viajante comercial*, en el que se trata de minimizar la distancia de recorrido cíclico de n ciudades. Las n variables del problema $X=\{x_1, ..., x_n\}$ son las posiciones del recorrido, y cada variable posee como dominio las n ciudades a visitar. Partiendo de la función de energía desarrollada antes para este problema, los pesos de las conexiones de la red adoptan la siguiente expresión (en notación matricial de doble subíndice):

$$w_{ik,jl} = W + W\delta(i,j) + W\delta(k,l) - d(k,l)\left(\delta(i,(j \bmod n)+1) + \delta((i \bmod n)+1,j)\right)$$

donde $i \neq j$ ó $k \neq l$, es decir, $w_{ik,jl}$ no corresponde a una conexión de lazo, y δ es la función delta de Kroenecker. El primer sumando representa la inhibición global, el segundo la restricción estructural de la red, el tercero la restricción del problema (variables con valores distintos), y el cuarto el objetivo de optimización (distancia entre las ciudades k y l). De acuerdo con esta expresión, la inhibición debida a objetivos de optimización, que recibe un nodo activo en un estado de la red correspondiente a una solución factible, es igual al opuesto de la suma de las distancias entre la ciudad que representa el nodo y las ciudades precedente y siguiente en la ruta. Una cota inferior de esta inhibición puede calcularse como el opuesto del doble de la máxima distancia entre dos ciudades cualesquiera a visitar. Por tanto, el valor apropiado de la inhibición global W para este problema es:

$$W = -4 \cdot \max\left\{d(x_i, x_j)\right\}$$

Ejemplo 2. Sea una instancia del *problema de la partición de grafos*, en el que se trata de dividir un grafo plano de n vertices en p particiones equilibradas (con n/p vértices cada una) de forma que se minimice el número de conexiones entre vértices de particiones distintas. Las n variables del problema $X=\{x_1, ..., x_n\}$ son los vértices del grafo, y cada variable posee como dominio las p particiones a formar. Partiendo de la función de energía desarrollada antes para este problema, los pesos de las conexiones de la red adoptan la siguiente expresión (en notación matricial de doble subíndice):

$$w_{ik,jl} = W + W\delta(i,j) + \frac{W}{2n/p-1}\delta(k,l) - \gamma(k,l)\phi(i,j)(1-\delta(i,j))(1-\delta(k,l))$$

donde $i \neq j$ ó $k \neq l$, es decir, $w_{ik,jl}$ no corresponde a una conexión de lazo, y δ es la función delta de Kroenecker. El primer sumando representa la inhibición global, el segundo la restricción estructural de la red, el tercero la restricción del problema (n/p variables con el mismo valor), y el cuarto el objetivo de optimización (vértices i y j conectados y localizados en distintas particiones k y l). De acuerdo con esta expresión, la inhibición debida a objetivos de optimización, que recibe un nodo activo en un estado de la red correspondiente a una solución factible, es igual al opuesto de la suma de las conexiones interpartición existentes entre el vértice que representa el nodo y los restantes vértices del grafo. Una cota inferior de esta inhibición puede calcularse como el opuesto del máximo número de conexiones interpartición que puede poseer un vértice cualquiera del grafo. Por tanto, dado que un vértice puede conectarse como máximo con n - n/p vértices de otras particiones, el valor apropiado de la inhibición global W para este problema es:

$$W = -2(n - n/p)$$

Ejemplo 3. Sea una instancia del *problema de la asignación cuadrática*, en el que se trata de situar n plantas de proceso en p lugares posibles ($p \geq n$), sabiendo que entre cada dos plantas de proceso se debe transportar una cantidad diferente de material con un coste unitario distinto según el lugar donde se localicen éstas, de forma que se minimice el coste total del transporte de materiales. Las n variables del problema $X=\{x_1, ..., x_n\}$ son las plantas de proceso, y cada variable posee como dominio los p lugares posibles de ubicación. Partiendo de la función de energía desarrollada antes para este problema, los pesos de las conexiones de la red adoptan la siguiente expresión (en notación matricial de doble subíndice):

$$w_{ik,jl} = W + W\delta(i,j) + W\delta(k,l) - c(k,l)q(i,j)(1-\delta(i,j))(1-\delta(k,l))$$

donde $i \neq j$ ó $k \neq l$, es decir, $w_{ik,jl}$ no corresponde a una conexión de lazo, y δ es la función delta de Kroenecker. El primer sumando representa la inhibición global, el segundo la restricción estructural de la red, el tercero la restricción del problema (variables con valores distintos), y el cuarto el objetivo de optimización (coste unitario del transporte entre los lugares k y l, por la cantidad a transportar entre las plantas i y j). De acuerdo con esta expresión, la inhibición debida a objetivos de optimización, que recibe un nodo activo en un estado de la red correspondiente a una solución factible, es igual al opuesto de la suma de los costes de transporte de materiales entre la planta que representa el nodo, y las n-1 restantes plantas del problema. Una cota inferior de esta inhibición puede calcularse como el opuesto del máximo coste de transporte entre dos plantas cualesquiera multiplicado por n-1. Por tanto, el valor apropiado de la inhibición global W para este problema es:

$$W = -2(n-1) \cdot \max\left\{c(x_i, x_j)q(i,j)\right\}$$

5.1.5 Indicaciones específicas para la implementación de problemas de optimización con restricciones mediante la red ODE.

La implementación de problemas de optimización combinatoria mediante el optimizador discreto estocástico no difiere sustancialmente del procedimiento descrito en los apartados precedentes. Así, aunque la arquitectura de la red ODE no sea simétrica, el significado de los nodos visibles (principales y auxiliares) de la red y el de los pesos de las conexiones es el mismo que en la red de Hopfield continua, la máquina de Boltzmann y sus variantes, por lo que puede plantearse la implementación del problema siguiendo las mismas normas que se han descrito. Tan sólo existen las siguientes diferencias importantes a considerar:

- La tendencia I de cada nodo, obligatoria en otras redes para contrarrestar el efecto inhibidor de los pesos de las conexiones y garantizar la activación de K nodos de N en la estabilización de la red, no es necesaria en el modelo ODE. Esto se debe a su esquema de procesamiento característico, en el que el número de nodos activos permanece siempre constante.
- El valor de umbral θ del nodo oculto de la red ODE, que marca el incremento de energía máximo que puede aceptarse en una transición de estado de la red, se determina como una cota superior del máximo valor de la función objetivo del problema.

- La inhibición global W solamente posee ahora el significado de término de normalización entre la inhibición debida a las restricciones del problema y la inhibición debida a los objetivos de optimización, ya que no influye en el requisito de la activación de K nodos sobre N al finalizar el procesamiento de la red. Así mismo, la inhibición global W se determinaba en las otras redes como el doble del valor mínimo de la inhibición negativa -g, debida a objetivos de optimización, que recibía un nodo cualquiera de la red en un estado correspondiente a una solución válida. Ahora, en cambio, el valor de W debe ser igual al doble del opuesto del umbral θ del nodo oculto, es decir, $W = -2\theta$. De esta forma, si al generarse una transición, el nuevo estado de la red infringe una restricción del problema, el valor de W hace que el incremento de energía producido sea mayor que el umbral θ, con lo que la transición no se aceptará.
- Por último, las restricciones especiales sobre el dominio de las variables, se implementaban en las redes simétricas eliminando la tendencia positiva I en el nodo j correspondiente al valor restringido de la variable. Dado que la tendencia I no existe en la red ODE, estas restriciones se implementan considerando el nodo j como incompatible con todos los restantes nodos de la red, por lo que deberá sumarse el término W a todos los pesos de las conexiones que entran o salen del nodo j. Esta operación resulta válida en la red ODE, porque, dando un valor inicial nulo al nodo j, éste nunca puede llegar a activarse, y por tanto nunca puede llegar a desactivar a los restantes nodos de la red, como podría ocurrir en las otras redes neuronales de optimización.

5.2 ESPECIFICACIÓN DE LAS INSTANCIAS DE LOS PROBLEMAS A UTILIZAR EN LA EVALUACIÓN DEL MODELO.

El análisis práctico del modelo neuronal ODE se ha realizado evaluando su comportamiento en la resolución de tres problemas clásicos de optimización combinatoria, con respecto a cuatro de los métodos actuales más importantes, dos de tipo heurístico y dos conexionistas: el algoritmo de búsqueda local, el algoritmo del temple simulado, la red de Hopfield continua y la máquina de Boltzmann.

Los problemas utilizados en la comparación son el problema del viajante comercial, el problema de la partición de grafos, y el problema de la asignación cuadrática, los cuales se han definido formalmente en el capítulo 2, y se han empleado como ejemplos en las descripciones realizadas en la tesis hasta el momento. Todas las pruebas con las diferentes técnicas de resolución se han efectuado sobre dos instancias de cada uno de estos problemas. Con el fin de conocer en detalle las características de las pruebas realizadas, se describe a continuación cada una de las instancias para los tres problemas tratados.

5.2.1 Problema del viajante comercial.

El problema del viajante comercial consiste en encontrar el camino más corto de recorrido cíclico de n ciudades, de tal forma que cada ciudad se visite una sola vez. Para realizar las pruebas de evaluación del modelo ODE, se han utilizado dos instancias de la versión euclídea de este problema, las cuales se describen a continuación:

- Problema del viajante sobre un recorrido formado por 15 capitales de provincia españolas, siendo la distancia mínima entre dos ciudades de 71 km, y la distancia máxima de 1118 km. La matriz de distancias entre las 15 ciudades del problema se incluye en el apéndice I de la tesis (tabla 1.11). La ruta óptima que minimiza la distancia total a recorrer es de 3737 km. (figura 5.7):

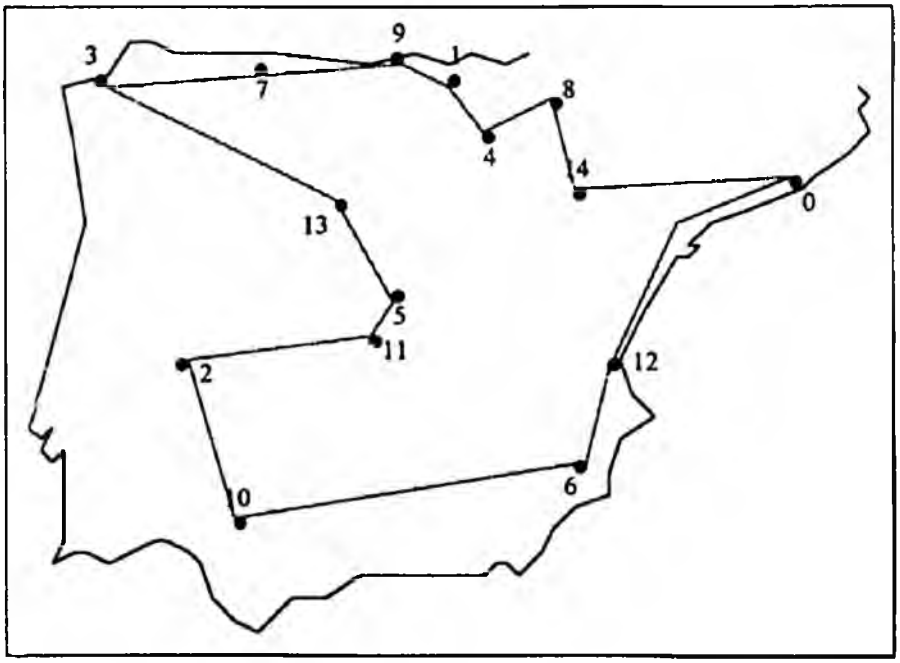

Figura 5.7: Solución óptima para el problema del viajante sobre 15 ciudades: 3737 km.

- Problema del viajante sobre un recorrido formado por 20 capitales de provincia españolas, siendo la distancia mínima entre dos ciudades de 71 km, y la distancia máxima de 1153 km. La matriz de distancias entre las 20 ciudades del problema se incluye en el apéndice I de la tesis (tabla I.12). La ruta óptima que minimiza la distancia total a recorrer es de 4142 km. (figura 5.8):

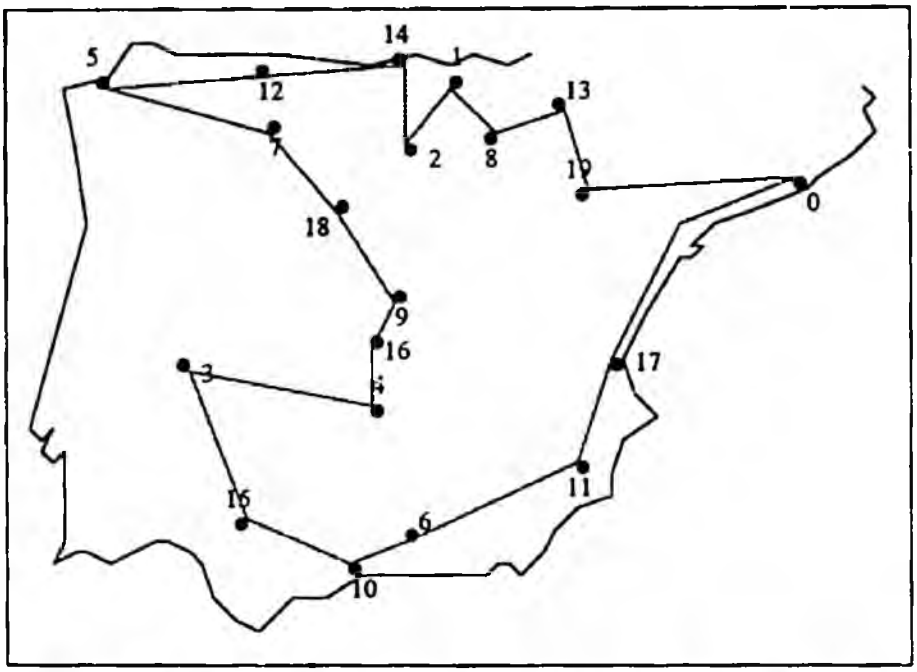

Figura 5.8: Solución óptima para el problema del viajante sobre 20 ciudades: 4142 km.

5.2.2 Problema de la partición de grafos.

El problema de la partición de grafos consiste en dividir un grafo plano de n vértices en p particiones equilibradas, de forma que se minimice el número de conexiones entre vértices de diferentes particiones. Para realizar las pruebas de evaluación del modelo ODE, se han utilizado dos instancias de este problema, las cuales se describen a continuación:

- Problema de la partición de grafos sobre un grafo plano de 50 vértices que debe dividirse en 2 particiones, existiendo entre dos vértices como mínimo 1 conexión y como máximo 5 conexiones. La matriz de conexiones entre los 50 vértices del grafo del problema se incluye en el apéndice II de la tesis (tabla II.11). La solución óptima que minimiza el número de conexiones entre particiones distintas es de 4 conexiones (figura 5.9):

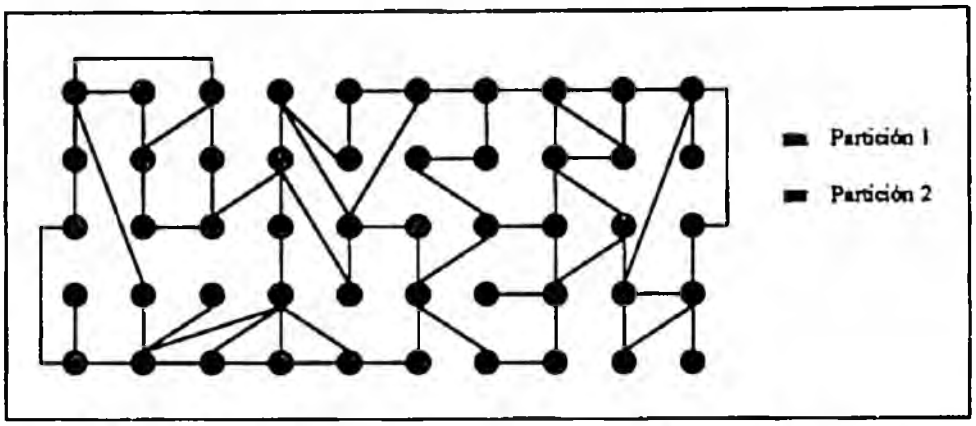

Figura 5.9: Solución óptima para la partición de grafos 50/2: 4 conexiones.

- Problema de la partición de grafos sobre un grafo plano de 50 vértices que debe dividirse en 5 particiones, existiendo entre dos vértices como mínimo 1 conexión y como máximo 5 conexiones. La matriz de conexiones entre los 50 vértices del grafo del problema se incluye en el apéndice II de la tesis (tabla II.11). La solución óptima que minimiza el número de conexiones entre particiones distintas es de 11 conexiones (figura 5.10):

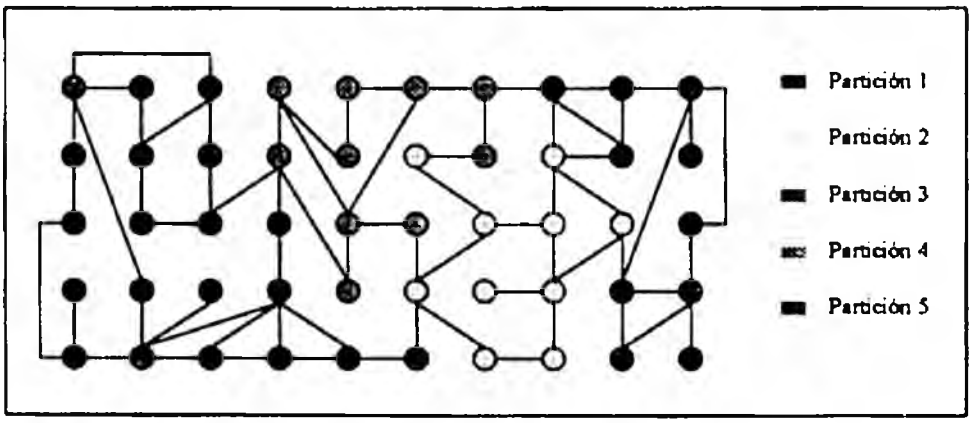

Figura 5.10: Solución óptima para la partición de grafos 50/5: 11 conexiones.

5.2.3 Problema de la asignación cuadrática.

El problema de la asignación cuadrática consiste en situar n plantas de proceso en p lugares posibles ($p \geq n$), sabiendo que entre cada dos plantas de proceso se debe transportar una cantidad diferente de material con un coste unitario distinto según el lugar donde se localicen éstas, de forma que se minimice el coste total del transporte de materiales. Para realizar las pruebas de evaluación del modelo ODE, se han utilizado dos instancias de este problema, las cuales se describen a continuación:

- Problema de la asignación cuadrática en el que se deben ubicar 15 plantas de proceso en 15 ciudades españolas, con cantidades mínima y máxima a transportar entre dos plantas de 1 y 99 respectivamente, y costes mínimo y máximo de transporte entre dos ciudades de 71 y 1118 respectivamente. Las matrices de cantidades y costes del problema se incluyen en el apéndice III de la tesis (tablas III.11 y III.12). La solución óptima que minimiza el coste total del transporte es de 2 311 286 pts (figura 5.11):

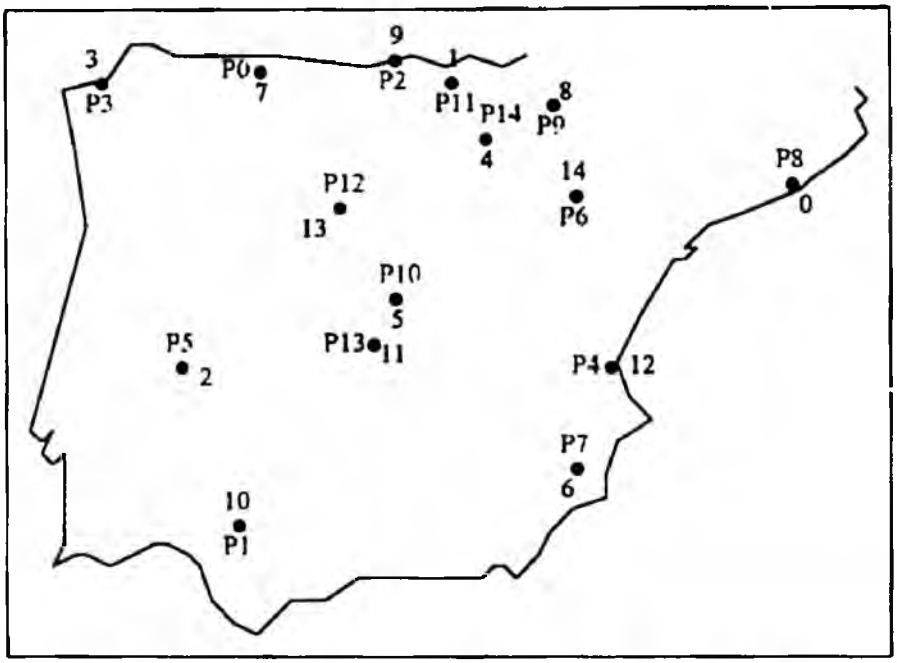

Figura 5.11: Solución óptima para la asignación cuadrática 15: 2 311 286 pts.

- Problema de la asignación cuadrática en el que se deben ubicar 20 plantas de proceso en 20 ciudades españolas, con cantidades mínima y máxima a transportar entre dos plantas de 26 y 999 respectivamente, y costes mínimo y máximo de transporte entre dos ciudades de 71 y 1153 respectivamente. Las matrices de cantidades y costes del problema se incluyen en el apéndice III de la tesis (tablas III.13 y III.14). La solución óptima que minimiza el coste total del transporte es de 44 940 360 pts (figura 5.12):

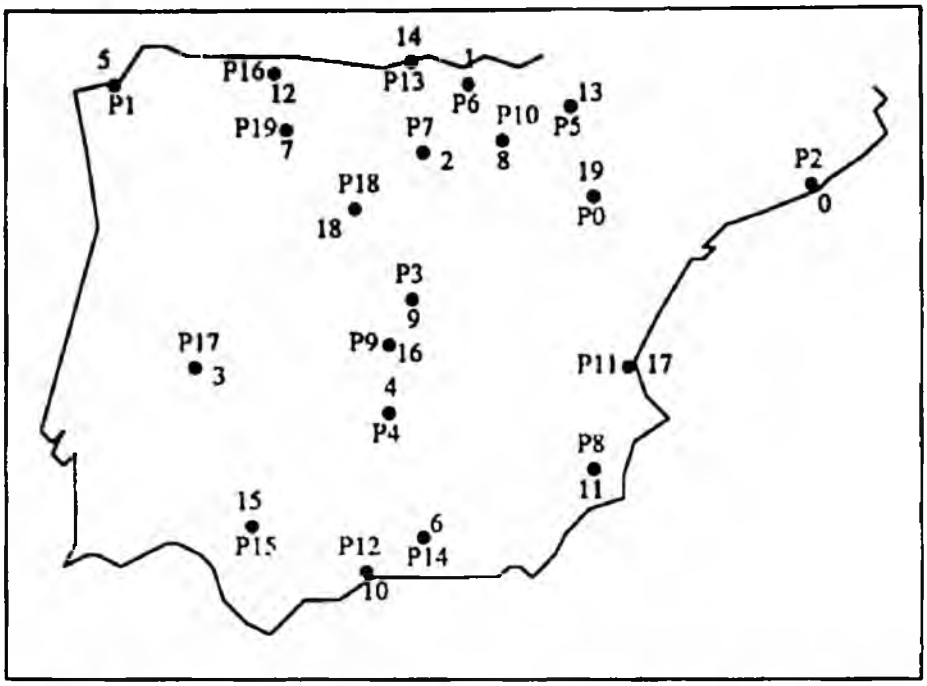

Figura 5.12: Solución óptima para la asignación cuadrática 20: 44 940 360 pts.

5.3 CRITERIOS ESTADÍSTICOS DE EVALUACIÓN.

Para obtener una evaluación correcta de la eficiencia de la red ODE como método de resolución aproximada de problemas de optimización combinatoria, con respecto a las restantes técnicas propuestas, es necesario establecer criterios estadísticos de evaluación de la eficiencia, tanto en la calidad de la solución obtenida como en el tiempo de ejecución empleado.

La calidad de las soluciones obtenidas puede evaluarse utilizando dos índices de referencia importantes [AART89] [PROT90]:

- En primer lugar, puede usarse el índice de optimización con respecto a la mejor solución posible, definido como el cociente entre el coste de la solución media obtenida con cada método y el coste de la solución óptima del problema:

$$I_{opt} = \frac{f(s_m)}{f(s_{opt})}$$

Cuanto más próximo a 1 sea este valor, tanto mejor será el comportamiento medio del algoritmo en cuanto a la calidad de las soluciones obtenidas. Por

tanto, este índice permite evaluar el error medio $\varepsilon = I_{opt} - 1$, cometido por cada uno de los métodos aplicados en la resolución del problema.

- En segundo lugar, puede usarse el índice de variación del coste con respecto a la media, definido para cada método como el cociente entre la desviación típica del coste de las soluciones y el coste de la solución media:

$$I_{var} = \frac{\sigma_c}{f(s_m)}$$

Este índice permite evaluar la dispersión de las soluciones obtenidas con cada uno de los métodos aplicados en la resolución del problema. Cuanto más próximo a 0 sea este valor, tanto más estable será el método con respecto a su nivel de optimización medio, es decir, mayor será la probabilidad de que una sola ejecución del algoritmo proporcione el nivel de optimización esperado que caracteriza al método.

Antes de definir los criterios de evaluación de la eficiencia temporal de las técnicas de resolución, es necesario realizar las siguientes consideraciones:

- Los métodos heurísticos de resolución, basados en la búsqueda local y el temple simulado, poseen complejidad temporal potencial [KERN86], [AART89].
- De la misma forma, los métodos conexionistas de resolución, basados en la red de Hopfield continua, en la máquina de Boltzmann, y en el optimizador discreto estocástico, se han implementado mediante simulaciones secuenciales con una complejidad temporal potencial, que depende fundamentalmente del número de nodos de la red. Sin embargo, hay que indicar que los métodos conexionistas presentarían una eficiencia en tiempo de ejecución muy superior a la obtenida, en el caso de que se realizase una implementación paralela de los mismos.

La eficiencia temporal de los distintos métodos de resolución puede evaluarse utilizando dos índices de referencia importantes [AART89]:

- En primer lugar, dado que no tiene sentido hablar de un índice con respecto al mínimo tiempo de ejecución posible, parece conveniente utilizar uno de los métodos de resolución como referencia, por ejemplo, el más rápido (búsqueda local), y definir un índice temporal con respecto a su tiempo de ejecución medio. Así, este índice se define como el cociente entre el tiempo medio de ejecución de cada método y el tiempo medio de ejecución del método basado en búsqueda local:

$$T_{opt} = \frac{t_m}{t_{bl}}$$

Este índice permite evaluar el tiempo empleado por cada uno de los métodos aplicados en la resolución del problema, en relación con el de búsqueda local, esto es, con el más rápido. Cuanto más próximo a 1 sea este valor, tanto mejor será el comportamiento medio del algoritmo en velocidad de ejecución.

- En segundo lugar, puede utilizarse el índice de variación temporal con respecto a la media, definido como el cociente entre la desviación típica de los tiempos de ejecución y el tiempo de ejecución medio:

$$T_{var} = \frac{\sigma_t}{t_m}$$

Este índice permite evaluar la dispersión de los tiempos obtenidos con cada uno de los métodos aplicados en la resolución del problema. Cuanto más próximo a 0 sea este valor, tanto más estable será el método con respecto a su tiempo de ejecución medio, es decir, mayor será la probabilidad de que una ejecución del algoritmo necesite para su desarrollo el tiempo esperado que caracteriza al método.

5.4 ANÁLISIS DE LAS PRUEBAS REALIZADAS.

Para efectuar las pruebas, se han implementado los diferentes problemas y métodos de comparación en lenguaje C, utilizando una metodología de diseño y programación modular basada en máquinas abstractas. A continuación, se sintetizan y estudian los resultados de las pruebas realizadas (20 por cada instancia de problema y método), las cuales se exponen detalladamente en los apéndices de la tesis.

5.4.1 Pruebas del problema del viajante comercial.

Las tablas 5.1 y 5.2 contienen los resultados correspondientes a niveles de optimización de las pruebas del problema del viajante comercial sobre recorridos de 15 y 20 ciudades respectivamente, realizadas mediante los cinco métodos de comparación: búsqueda local (BL), temple simulado (TS), red de Hopfield continua (CH), máquina de Boltzmann (BM), y optimizador discreto estocástico (ODE). Estos resultados incluyen el coste

medio, el coste mejor y el coste peor, la desviación típica σ_c, el índice de coste medio I_{opt}, y el índice de variación del coste I_{var}:

Viajante 15	C. Medio	C. Mejor	C. Peor	σ_c	I_{opt} (%)	I_{var} (%)
BL	4399.9	3737	4834	330.0	117.74	7.5
TS	3859.4	3737	4314	172.0	103.28	4.46
CH	6766.5	5124	7986	757.5	181.07	11.19
BM	7259.4	5937	8077	587.1	194.26	8.09
ODE	3863.2	3737	4509	224.4	103.38	5.81

Tabla 5.1: Niveles de optimización en el problema del viajante 15.

Viajante 20	C. Medio	C. Mejor	C. Peor	σ_c	I_{opt} (%)	I_{var} (%)
BL	4895.1	4187	5565	393.6	118.18	8.04
TS	4418.5	4142	4881	252.8	106.68	5.72
CH	8180.7	6199	10119	1110.8	197.51	13.58
BM	9319.3	7976	11058	873.0	225	9.37
ODE	4326.6	4142	4811	204.4	104.46	4.72

Tabla 5.2: Niveles de optimización en el problema del viajante 20.

En la figura 5.13 se presenta un diagrama de columnas del índice de coste medio obtenido para el problema del viajante con las diferentes técnicas de resolución. Este índice refleja el nivel medio de optimización alcanzado por cada método con respecto a la solución óptima. Hay que destacar el buen nivel de optimización alcanzado por la red ODE y el temple simulado, con un porcentaje medio de error con respecto al óptimo del 4 % y el 5 % respectivamente. El método basado en búsqueda local proporciona un error del 18 %, mientras que los otros métodos conexionistas basados en las redes CH y BM quedan muy lejos de los niveles de optimización anteriores, con errores que oscilan entre el 81 % y el 98 % para la primera, y el 94 % y el 125 % para la segunda. El gran porcentaje de error cometido por estas redes neuronales confirma el análisis realizado en los capítulos 3 y 4, en el que se exponían las limitaciones intrínsecas de estos modelos para la resolución de problemas de optimización combinatoria.

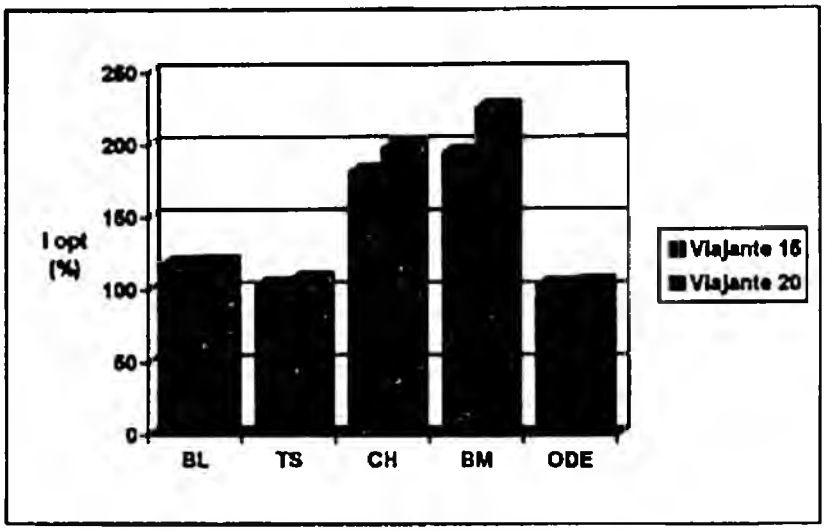

Figura 5.13: Índice de coste medio para el problema del viajante.

En la figura 5.14 se presenta un diagrama de columnas del índice de variación del coste obtenido para el problema del viajante con las diferentes técnicas de resolución. Este índice refleja para cada método la dispersión con respecto a la media de las soluciones obtenidas. Hay que destacar el bajo nivel de variación alcanzado por el temple simulado y la red ODE, con un porcentaje medio del 5 % aproximadamente para ambos casos. El método basado en búsqueda local proporciona una variación del 8 %, mientras que los otros métodos conexionistas basados en las redes CH y BM poseen índices de variación del 12 % y 9 % respectivamente.

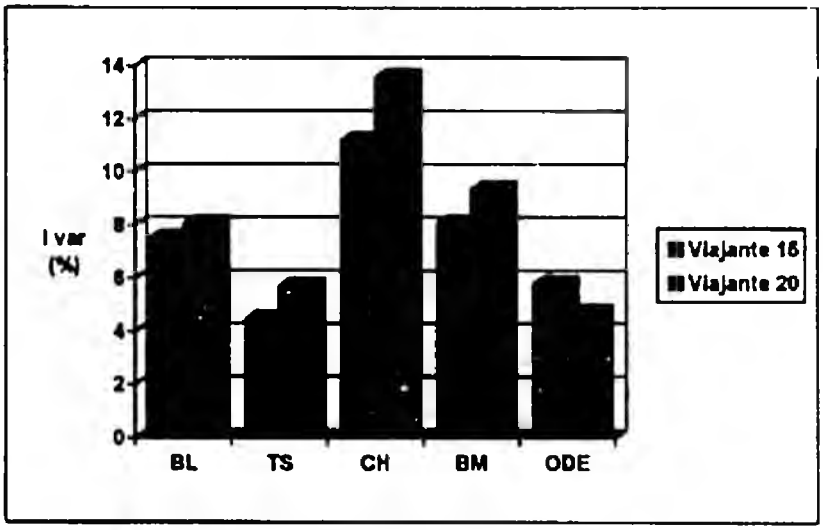

Figura 5.14: Índice de variación del coste para el problema del viajante.

Las tablas 5.3 y 5.4 contienen los resultados correspondientes a tiempos de ejecución de las pruebas del problema del viajante comercial sobre recorridos de 15 y 20 ciudades

respectivamente, realizadas mediante los cinco métodos de comparación: búsqueda local (BL), temple simulado (TS), red de Hopfield continua (CH), máquina de Boltzmann (BM), y optimizador discreto estocástico (ODE). Estos resultados incluyen el tiempo medio, el tiempo mejor y el tiempo peor, la desviación típica σ_t, el índice de tiempo medio T_{opt}, y el índice de variación del tiempo T_{var}:

Viajante 15	T. Medio	T. Mejor	T. Peor	σ_t	T_{opt}	T_{var} (%)
BL	7.2	3	15	3.0	1	41.67
TS	280.6	219	342	39.9	38.97	14.22
CH	1292.9	1023	1619	163.4	179.57	12.64
BM	67.9	64	77	3.0	9.43	4.42
ODE	362.0	274	476	62.7	50.28	17.32

Tabla 5.3: Tiempos de ejecución del problema del viajante 15.

Viajante 20	T. Medio	T. Mejor	T. Peor	σ_t	T_{opt}	T_{var} (%)
BL	42.4	20	59	11.9	1	28.07
TS	1217.7	1031	1516	155	28.72	12.73
CH	6379.1	5133	7867	781.5	150.45	12.25
BM	261.8	246	283	7.5	6.17	2.86
ODE	1326.2	1178	1789	173.8	31.28	13.11

Tabla 5.4: Tiempos de ejecución del problema del viajante 20.

En la figura 5.15 se presenta un diagrama de columnas del índice de tiempo medio de ejecución obtenido para el problema del viajante con las diferentes técnicas de resolución. Este índice refleja el tiempo medio de ejecución empleado por cada método con respecto al de búsqueda local, que sirve como referencia. El algoritmo que resulta más rápido después del de búsqueda local es el correspondiente a la máquina de Boltzmann, con un índice comprendido entre 6.1 y 9.5. A continuación, aparecen muy próximos el algoritmo del temple simulado con un índice comprendido entre 28.7 y 39, y la red ODE con un índice que oscila entre 31.2 y 50.3. Finalmente, el peor comportamiento corresponde a la red de Hopfield continua, que presenta un índice

temporal entre 150.5 y 179.5. No obstante, hay que indicar que los métodos conexionistas de procesamiento síncrono, la red de Hopfield continua y la red ODE, se han implementado mediante simulaciones secuenciales, por lo que el índice presentado no corresponde al que se obtendría en una implementación directa (paralela) de los mismos. Por el contrario, el método basado en la máquina de Boltzmann no puede implementarse directamente en forma paralela, ya que este modelo posee un mecanismo de procesamiento asíncrono, que se caracteriza por la actualización en cada instante del estado de un solo nodo de la red.

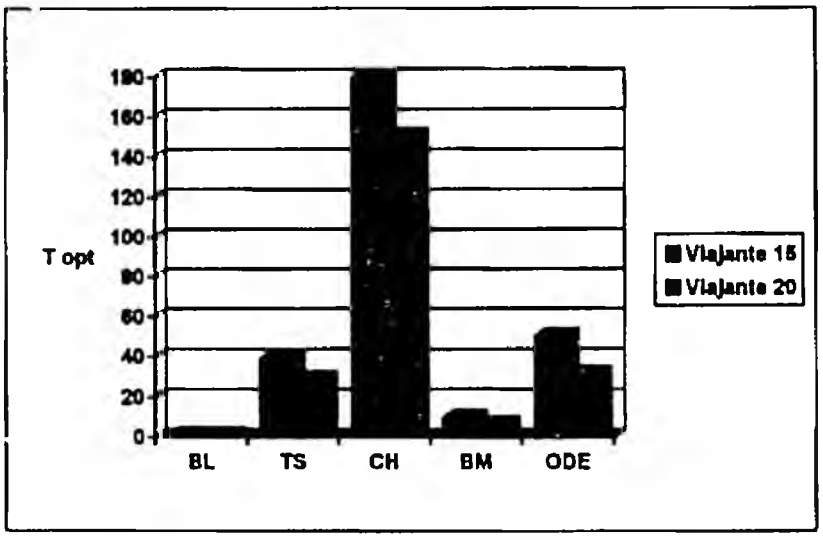

Figura 5.15: Índice de tiempo medio para el problema del viajante.

En la figura 5.16 se presenta un diagrama de columnas del índice de variación temporal obtenido para el problema del viajante con las diferentes técnicas de resolución. Este índice refleja para cada método la dispersión con respecto a la media de los tiempos obtenidos. El menor nivel de variación corresponde a la máquina de Boltzmann, con un porcentaje medio del 3.5 %. Superiores al anterior, pero muy próximos entre sí, son los porcentajes de variación temporal de la red CH (12.5 %), el temple simulado (13.5 %) y la red ODE (15 %). En este caso, el peor comportamiento corresponde al método de búsqueda local, con un porcentaje de variación que oscila entre el 28 % y el 42 %.

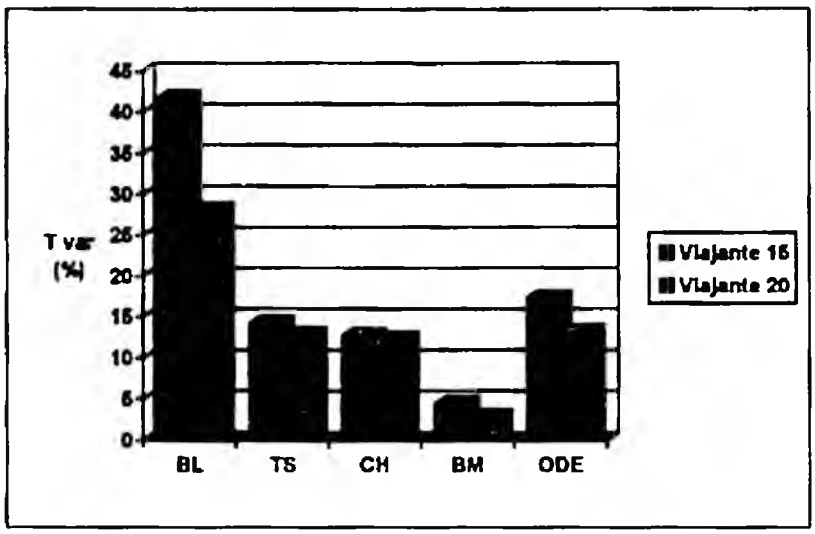

Figura 5.16: Índice de variación temporal para el problema del viajante.

5.4.2 Pruebas del problema de la partición de grafos.

Las tablas 5.5 y 5.6 contienen los resultados correspondientes a niveles de optimización de las pruebas del problema de la partición de grafos sobre un grafo plano de 50 vértices que se divide en 2 y 5 particiones respectivamente, realizadas mediante los cinco métodos de comparación: búsqueda local (BL), temple simulado (TS), red de Hopfield continua (CH), máquina de Boltzmann (BM), y optimizador discreto estocástico (ODE). Estos resultados incluyen el coste medio, el coste mejor y el coste peor, la desviación típica σ_c, el índice de coste medio l_{opt}, y el índice de variación del coste l_{var}:

Grafos 50/2	C. Medio	C. Mejor	C Peor	σ_c	l_{opt} (%)	l_{var} (%)
BL	9	4	14	2.8	225	31.11
TS	5.1	4	8	1.4	127.5	27.45
CH	30.7	22	40	5.3	767.5	17.26
BM	34.1	27	43	4.5	852.5	13.2
ODE	4.8	4	9	1.3	120	27.08

Tabla 5.5: Niveles de optimización en el problema de la partición de grafos 50/2.

Grafos 50/5	C. Medio	C. Mejor	C. Peor	σ_c	I_{opt} (%)	I_{mc} (%)
BL	19.6	13	26	3.4	178.18	17.35
TS	13.8	11	18	1.8	125,45	13.04
CH	52.7	44	61	4.6	479.09	8.73
BM	55	45	62	3.6	500	6.55
ODE	13.5	11	17	1.6	122,73	11.85

Tabla 5.6: Niveles de optimización en el problema de la partición de grafos 50/5.

En la figura 5.17 se presenta un diagrama de columnas del índice de coste medio obtenido para el problema de la partición de grafos con las diferentes técnicas de resolución. Este índice refleja el nivel medio de optimización alcanzado por cada método con respecto a la solución óptima. Hay que destacar que el mejor nivel de optimización corresponde a la red ODE, con un porcentaje medio de error con respecto al óptimo del 21.5 %, quedando muy próximo a éste el del temple simulado con el 26.5 %. El método basado en búsqueda local proporciona un error entre el 78 % y el 125 %, mientras que los otros métodos conexionistas basados en las redes CH y BM quedan muy lejos de los niveles de optimización anteriores, con errores que oscilan entre el 379 % y el 668 % para la primera, y el 400 % y el 753 % para la segunda. También hay que indicar que los porcentajes obtenidos para el índice de coste medio y, por consiguiente, para el error, son bastante elevados debido a los valores concretos que toma la función de coste de este problema.

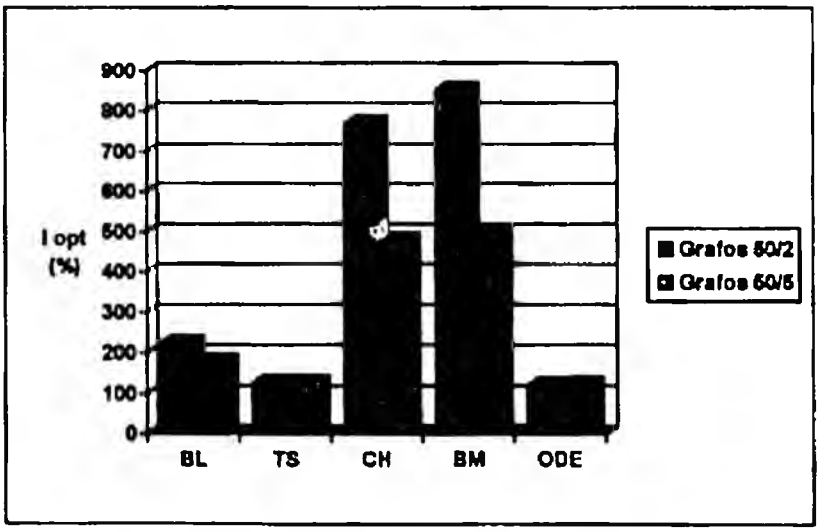

Figura 5.17: Índice de coste medio para el problema de la partición de grafos.

En la figura 5.18 se presenta un diagrama de columnas del índice de variación del coste obtenido para el problema de la partición de grafos con las diferentes técnicas de resolución. Este índice refleja para cada método la dispersión con respecto a la media de las soluciones obtenidas. Hay que destacar el bajo nivel de variación alcanzado por la máquina de Boltzmann, con un porcentaje medio que oscila entre el 6.5 % y el 13.2 %. El método basado en la red de Hopfield continua proporciona una variación comprendida entre el 8.7 % y el 17.3 %, mientras que tanto el temple simulado como la red ODE poseen índices de variación comprendidos entre el 11.8 % y el 27.5 %. Finalmente, el peor comportamiento corresponde al método de búsqueda local, con un porcentaje de variación que oscila entre el 17 % y el 31.2 %.

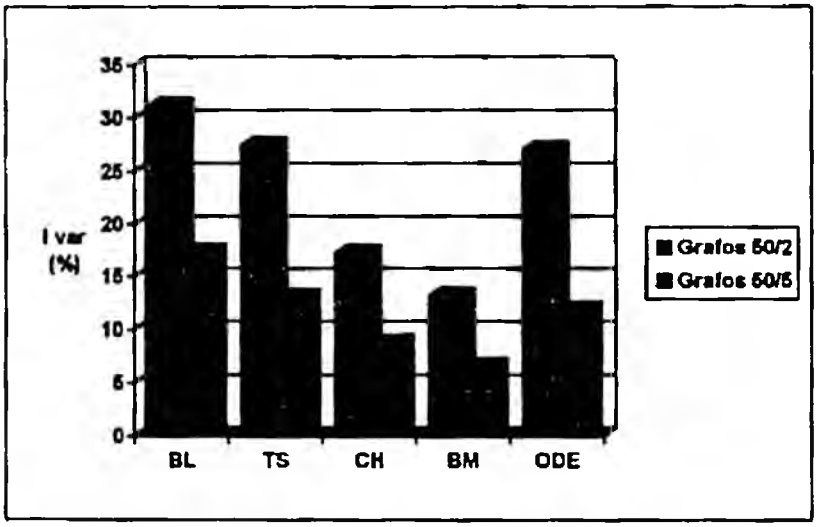

Figura 5.18: Índice de variación del coste para el problema de la partición de grafos.

Las tablas 5.7 y 5.8 contienen los resultados correspondientes a tiempos de ejecución de las pruebas del problema de la partición de grafos sobre un grafo plano de 50 vértices que se divide en 2 y 5 particiones respectivamente, realizadas mediante los cinco métodos de comparación: búsqueda local (BL), temple simulado (TS), red de Hopfield continua (CH), máquina de Boltzmann (BM), y optimizador discreto estocástico (ODE). Estos resultados incluyen el tiempo medio, el tiempo mejor y el tiempo peor, la desviación típica σ_t, el índice de tiempo medio T_{opt}, y el índice de variación del tiempo T_{var}:

Grafos 50/2	T. Medio	T. Mejor	T. Peor	σ_t	T_{opt}	T_{var} (%)
BL	9.2	5	20	4	1	43.48
TS	379	345	417	18.1	41.2	4.78
CH	245.8	73	713	160.3	26,72	65.22
BM	21	19	23	1.1	2.28	5.24
ODE	503.7	495	524	6.8	54.75	1.35

Tabla 5.7: Tiempos de ejecución del problema de la partición de grafos 50/2.

Grafos 50/5	T. Medio	T. Mejor	T. Peor	σ_t	T_{opt}	T_{var} (%)
BL	47.6	23	108	18.2	1	38.24
TS	1134.5	1077	1219	36.9	23.83	3.25
CH	2795.3	761	5712	1432.7	58,72	51.25
BM	97.3	93	102	2.7	2.04	2.77
ODE	3708.6	3608	3784	45.1	77.91	1.22

Tabla 5.8: Tiempos de ejecución del problema de la partición de grafos 50/5.

En la figura 5.19 se presenta un diagrama de columnas del índice de tiempo medio de ejecución obtenido para el problema de la partición de grafos con las diferentes técnicas de resolución. Este índice refleja el tiempo medio de ejecución empleado por cada método con respecto al de búsqueda local, que sirve como referencia. El algoritmo que resulta más rápido después del de búsqueda local es el correspondiente a la máquina de Boltzmann, con un índice de 2.15. A continuación, aparece el algoritmo del temple simulado con un índice comprendido entre 23.8 y 41.2. Finalmente, la red de Hopfield continua presenta un índice temporal entre 26.7 y 58.8, seguido por la red ODE con un índice que oscila entre 54.7 y 78.

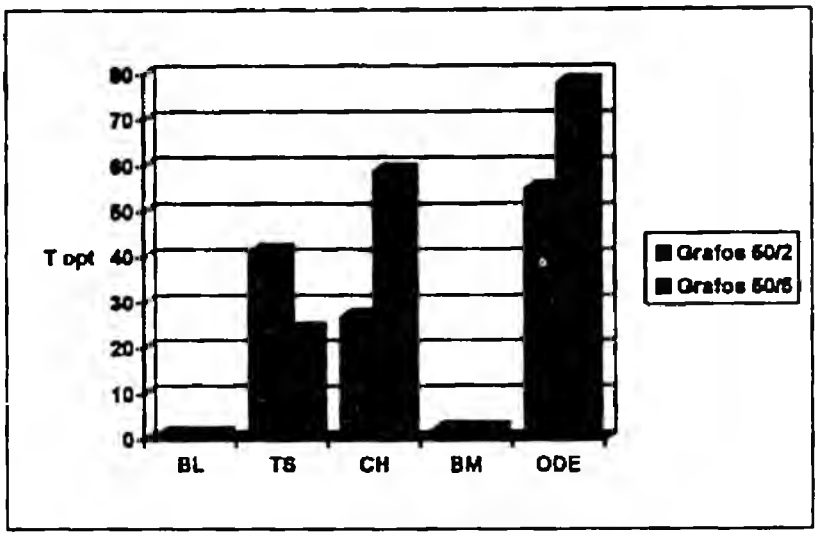

Figura 5.19: Índice de tiempo medio para el problema de la partición de grafos.

En la figura 5.20 se presenta un diagrama de columnas del índice de variación temporal obtenido para el problema de la partición de grafos con las diferentes técnicas de resolución. Este índice refleja para cada método la dispersión con respecto a la media de los tiempos obtenidos. Hay que destacar el bajo nivel de variación alcanzado por la red ODE, con un porcentaje medio del 1.28 %. Así mismo, son también bajos los niveles conseguidos por el temple simulado y la máquina de Boltzmann, los cuales proporcionan una variación aproximada del 4 %. Muy lejos de estos valores se hallan los porcentajes de variación correspondientes a la búsqueda local (41 %) y a la red de Hopfield continua (58 %).

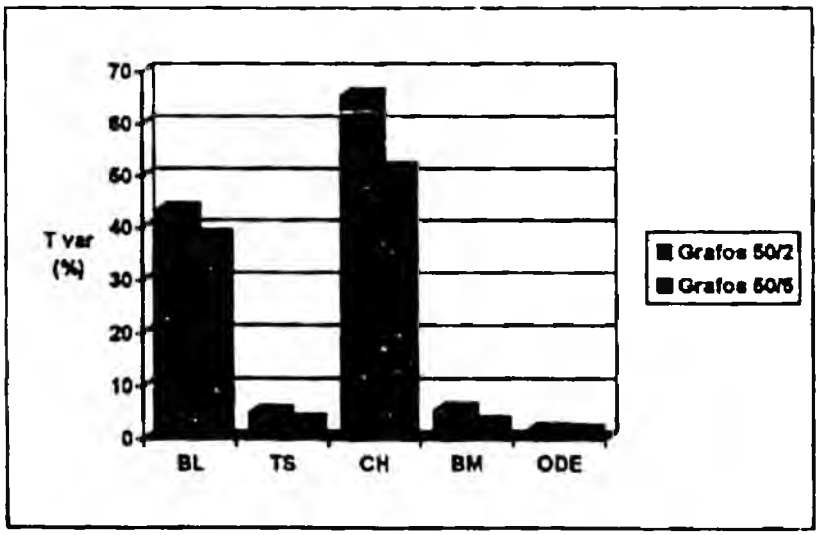

Figura 5.20: Índice de variación temporal para el problema de la partición de grafos.

5.4.3 Pruebas del problema de la asignación cuadrática.

Las tablas 5.9 y 5.10 contienen los resultados correspondientes a niveles de optimización de las pruebas del problema de la asignación cuadrática sobre la ubicación de 15 y 20 plantas de proceso en 15 y 20 lugares posibles respectivamente, realizadas mediante los cinco métodos de comparación: búsqueda local (BL), temple simulado (TS), red de Hopfield continua (CH), máquina de Boltzmann (BM), y optimizador discreto estocástico (ODE). Estos resultados incluyen el coste medio, el coste mejor y el coste peor, la desviación típica σ_c, el índice de coste medio I_{opt}, y el índice de variación del coste I_{var}:

Asign. 15	C. Medio	C. Mejor	C. Peor	σ_c	I_{opt} (%)	I_{var} (%)
BL	2349469.1	2327863	2391011	19141.6	101.65	0.81
TS	2328689.9	2311286	2352633	13587.3	100.75	0.58
CH	2625969.8	2498192	2713249	58188.2	113.62	2.22
BM	2664449.4	2514073	2794526	71726.5	115.28	2.69
ODE	2330089	2311286	2360801	12739.6	100.81	0.55

Tabla 5.9: Niveles de optimización en el problema de la asignación cuadrática 15.

Asign. 20	C. Medio	C. Mejor	C. Peor	σ_c	I_{opt} (%)	I_{var} (%)
BL	45588222.9	44940360	46244623	356827.4	101.44	0.78
TS	45176893	44940360	45729521	234932	100.53	0.52
CH	50982003.8	48774797	53348276	1049135.1	113.44	2.06
BM	52439199.9	49780901	54586071	1210121.4	116.69	2.31
ODE	45143448.6	44940360	45444316	162337.5	100.45	0.36

Tabla 5.10: Niveles de optimización en el problema de la asignación cuadrática 20.

Enla figura 5.21 se presenta un diagrama de columnas del índice de coste medio obtenido para el problema de la asignación cuadrática con las diferentes técnicas de resolución. Este índice refleja el nivel medio de optimización alcanzado por cada método con respecto a la solución óptima. Hay que destacar el buen nivel de optimización alcanzado

por el temple simulado y la red ODE, con un porcentaje medio de error con respecto al óptimo del 0.6 % aproximadamente para ambos casos. El método basado en búsqueda local proporciona un error del 1.5 %, mientras que los otros métodos conexionistas basados en las redes CH y BM quedan muy lejos de los niveles de optimización anteriores, con errores del 13.5 % para la primera, y del 16 % para la segunda.

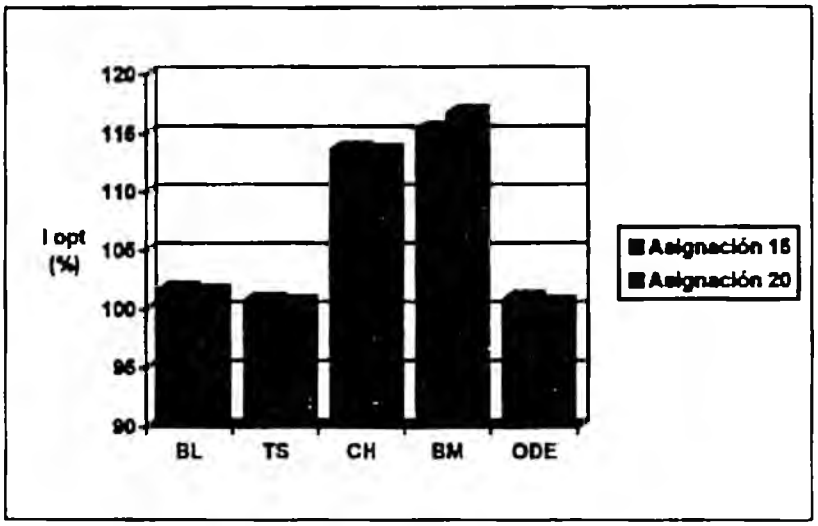

Figura 5.21: Índice de coste medio para el problema de la asignación cuadrática.

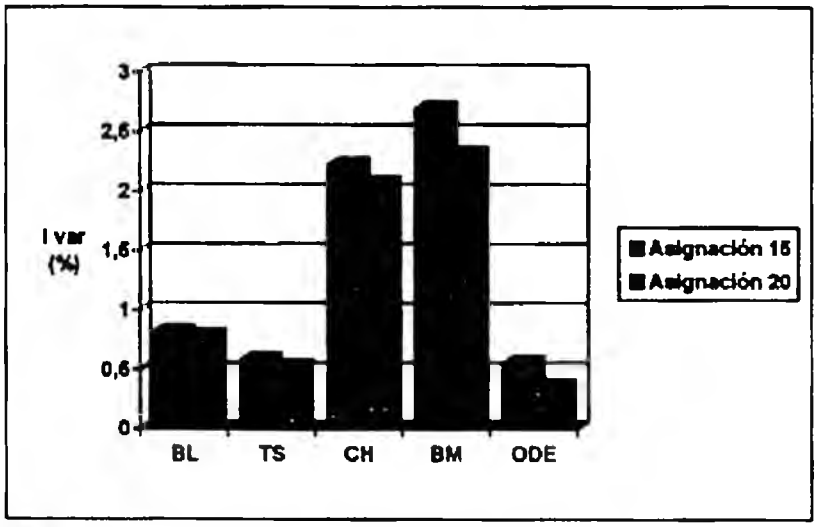

Figura 5.22: Índice de variación del coste para el problema de la asignación cuadrática.

En la figura 5.22 se presenta un diagrama de columnas del índice de variación del coste obtenido para el problema de la asignación cuadrática con las diferentes técnicas de resolución. Este índice refleja para cada método la dispersión con respecto a la media de las soluciones obtenidas. En este caso, hay que destacar el bajo nivel de variación alcanzado por la red ODE, con un porcentaje medio del 0.45 %, seguido inmediatamente

por el temple simulado con una variación del 0.55 %. El método basado en búsqueda local proporciona una variación del 0.8 %, mientras que los otros métodos conexionistas basados en las redes CH y BM quedan muy lejos de los niveles de variación anteriores, con porcentajes aproximados del 2.1 % para la primera, y del 2.45 % para la segunda.

Las tablas 5.11 y 5.12 contienen los resultados correspondientes a tiempos de ejecución de las pruebas del problema de la asignación cuadrática sobre la ubicación de 15 y 20 plantas de proceso en 15 y 20 lugares posibles respectivamente, realizadas mediante los cinco métodos de comparación: búsqueda local (BL), temple simulado (TS), red de Hopfield continua (CH), máquina de Boltzmann (BM), y optimizador discreto estocástico (ODE). Estos resultados incluyen el tiempo medio, el tiempo mejor y el tiempo peor, la desviación típica σ_t, el índice de tiempo medio T_{opt}, y el índice de variación del tiempo T_{var}:

Asign. 15	T. Medio	T. Mejor	T. Peor	σ_t	T_{opt}	T_{var} (%)
BL	7.7	4	13	3	1	38.96
TS	334.8	271	401	36.7	43.48	10.96
CH	1039.1	881	1240	91.5	134.95	8.81
BM	62.8	59	66	1.8	8.16	2.87
ODE	925.3	774	1104	84.8	120.17	9.16

Tabla 5.11: Tiempos de ejecución del problema de la asignación cuadrática 15.

Asign. 20	T. Medio	T. Mejor	T. Peor	σ_t	T_{opt}	T_{var} (%)
BL	43.1	23	67	13.2	1	30.63
TS	1674.8	1472	1878	100	38.86	5.97
CH	5166.7	4053	6224	569.5	119.8	11.02
BM	237.1	223	251	8.3	5.5	3.5
ODE	4254.9	3643	4633	280.2	98.72	6.59

Tabla 5.12: Tiempos de ejecución del problema de la asignación cuadrática 20.

En la figura 5.23 se presenta un diagrama de columnas del índice de tiempo medio de ejecución obtenido para el problema de la asignación cuadrática con las diferentes técnicas de resolución. Este índice refleja el tiempo medio de ejecución empleado por cada método con respecto al de búsqueda local, que sirve como referencia. El algoritmo que resulta más rápido después del de búsqueda local es el correspondiente a la máquina de Boltzmann, con un índice de 6.75. A continuación, aparece el algoritmo del temple simulado con un índice de 41.5, seguido por la red ODE con un índice medio de 110. Finalmente, la red de Hopfield continua presenta un índice temporal de 127.5.

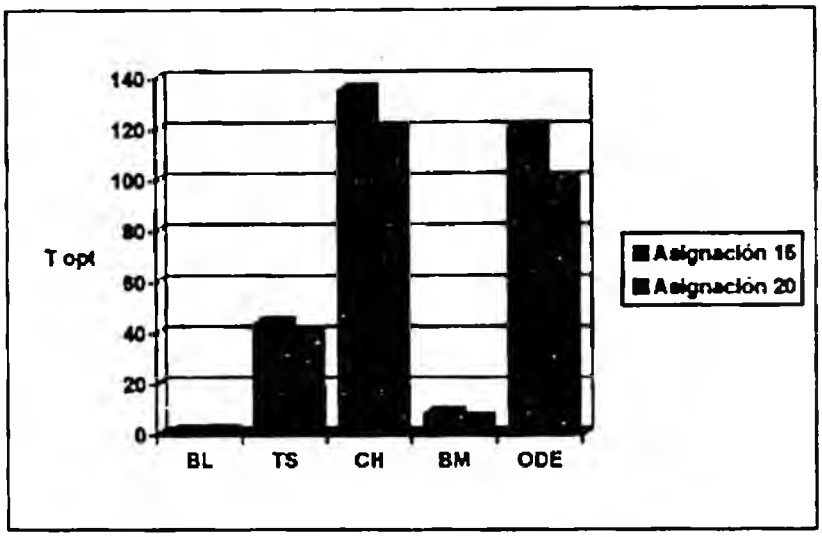

Figura 5.23: Índice de tiempo medio para el problema de la asignación cuadrática.

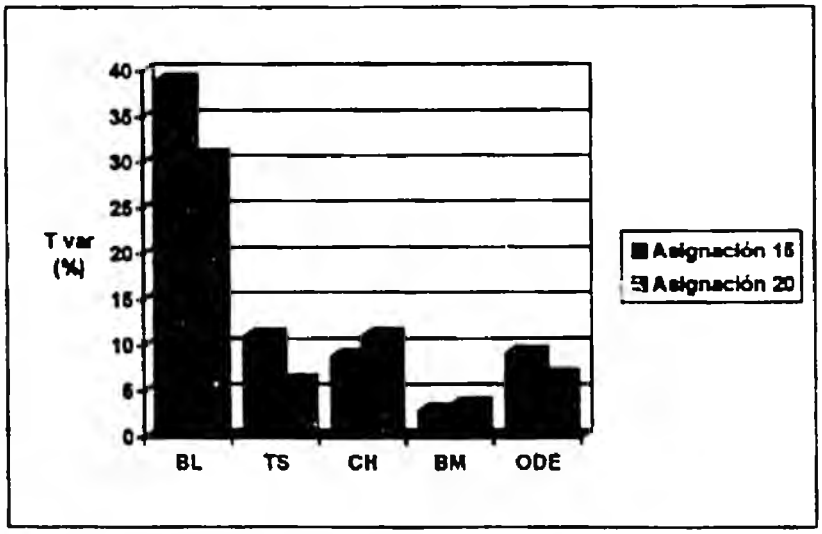

Figura 5.24: Índice de variación temporal para el problema de la asignación cuadrática.

En la figura 5.24 se presenta un diagrama de columnas del índice de variación temporal obtenido para el problema de la asignación cuadrática con las diferentes técnicas de resolución. Este índice refleja para cada método la dispersión con respecto a la media de los tiempos obtenidos. El menor nivel de variación corresponde a la máquina de Boltzmann, con un porcentaje medio del 3.2 %. Superiores al anterior, pero próximos entre sí, son los porcentajes medios de variación temporal de la red ODE (7.8 %), el temple simulado (8.5 %) y la red CH (9.9 %). En este caso, el peor comportamiento corresponde al método de búsqueda local, con un porcentaje de variación que oscila entre el 30.6 % y el 39 %.

5.4.4 Conclusiones.

El estudio comparativo de la eficiencia del optimizador discreto estocástico frente a las distintas técnicas de resolución de problemas de optimización combinatoria, permite obtener las siguientes conclusiones:

- Para todos los problemas tratados, la mejor actuación corresponde a los métodos basados en el algoritmo del temple simulado y en la red neuronal ODE. Puede decirse, por tanto, que el optimizador discreto estocástico es una red neuronal cuya aplicación a problemas de optimización combinatoria es tan satisfactoria como la del temple simulado, el cual se considera el mejor algoritmo general de optimización aproximada conocido.
- El comportamiento del método basado en búsqueda local es ciertamente peor que el de la red ODE, si bien ésta precisa un tiempo medio de ejecución secuencial bastante superior al de la búsqueda local para la resolución del problema. No obstante, esa diferencia podría reducirse, o incluso superarse, si se realizase una implementación masivamente paralela de la red ODE.
- Finalmente, las redes neuronales clásicas de Hopfield y Boltzmann presentan una eficiencia prácticamente nula en cuanto a nivel de optimización alcanzado. Esto se debe a las limitaciones estructurales y dinámicas que presentan estos modelos en su aplicación a problemas de optimización combinatoria, las cuales ya se analizaron en los capítulos precedentes.

A modo de resumen, la figura 5.25 muestra un diagrama comparativo de la eficiencia, en cuanto a la calidad de las soluciones, de los diferentes métodos de resolución con respecto a la red ODE. El diagrama representa el cociente entre el error del coste medio

respecto al óptimo de cada técnica empleada, y el error del coste medio obtenido con el optimizador discreto estocástico ($\varepsilon/\varepsilon_{ODE}$). Puede apreciarse claramente que la red ODE obtiene una eficiencia en optimización superior a la de las restantes técnicas, incluso mejorando el índice de error obtenido por el algoritmo del temple simulado. Los datos correspondientes se recogen en la tabla 5.13.

$\varepsilon/\varepsilon_{ODE}$	Viajante	Grafos	Asignación
BL	4.58	4.76	2.46
TS	1.27	1.24	1.02
CH	22.78	24.5	21.48
BM	27.97	26.98	25.37
ODE	1	1	1

Tabla 5.13: Errores de coste medio con respecto al error de la red ODE ($\varepsilon/\varepsilon_{ODE}$).

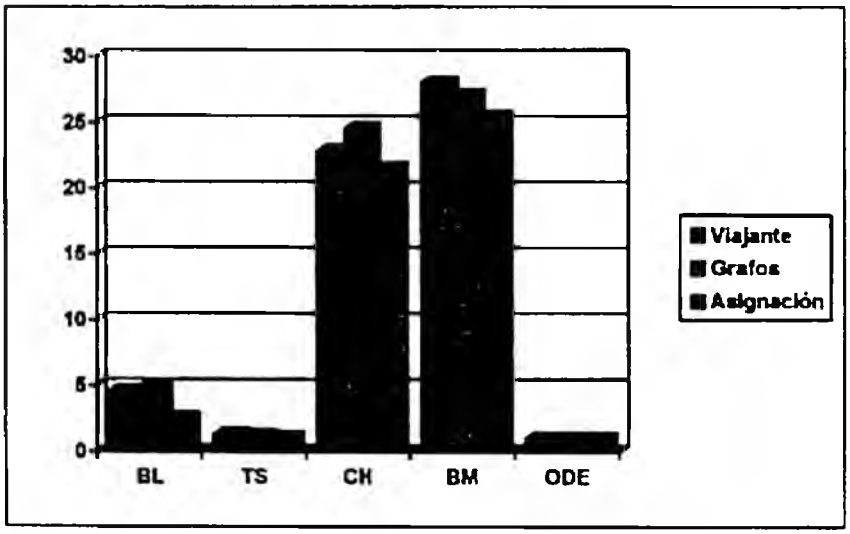

Figura 5.25: Comparativa global de la eficiencia en optimización de la red ODE ($\varepsilon/\varepsilon_{ODE}$).

6

Aplicación a la Programación Predictiva en Sistemas de Producción Discreta

Es en la práctica donde el hombre tiene que demostrar la verdad, es decir, la realidad y el poderio, la terrenalidad de su pensamiento.

Karl Marx

6.1 PROGRAMACIÓN DE PROCESOS DE PRODUCCIÓN. MODELOS DE LA PLANTA INDUSTRIAL.

6.1.1 Aspectos generales de la programación de producción.

Los problemas de asignación de recursos (scheduling) han generado un gran interés en el mundo científico-técnico, desde principios de la segunda guerra mundial hasta nuestros días. Los avances tecnológicos en los años previos a la segunda guerra mundial influyeron de forma especial en el desarrollo de la misma, surgiendo entonces una notable preocupación por optimizar la utilización de los recursos tanto materiales como humanos en determinadas actividades importantes de aquel momento (fabricación de armamentos, transporte de materiales, etc.). Esta preocupación ha continuado hasta hoy, especialmente en la industria y en las empresas de transporte, por la necesidad de una competitividad cada vez mayor, e incluso en la planificación de proyectos, elaboración de turnos de trabajo, etc. El interés en los problemas de asignación de recursos ha producido una gran cantidad de literatura con muchas ramificaciones a través de dos áreas del conocimiento principales: la Investigación Operativa y la Inteligencia Artificial.

Los problemas de programación o asignación de recursos pueden definirse brevemente de la siguiente forma: Se debe realizar una serie de actividades o tareas, y se dispone para ello de un conjunto de recursos materiales y/o humanos. El objetivo es encontrar un programa que asigne los recursos necesarios para realizar cada tarea, de forma que se minimice una función objetivo de coste, teniendo en cuenta las restricciones que puedan existir entre los diferentes elementos del sistema. Entre los problemas clásicos de asignación de recursos pueden citarse la programación de producción, el problema del transporte, la planificación de proyectos, la programación de turnos de trabajo y horarios, etc.

El problema de la programación o planificación de tareas de producción en entornos de fabricación (manufacturing) suscita actualmente un gran interés en el mundo de la investigación en Inteligencia Artificial, como lo prueba el gran número de artículos y proyectos de tesis doctorales dedicados a este problema aparecidos en los últimos años: [FOX83], [HYNY88], [PAPE88], [BURK89], [PROS90], [BERR91], [SADE91]. Este problema admite muchas variantes, las cuales se estudiarán en detalle más adelante. En general, presenta las siguientes características:

- Una factoría o planta industrial tiene como principal objetivo la producción de un conjunto de artículos diferentes. La elaboración de cada artículo se determina mediante un plan de fabricación compuesto por uno o más procesos, los cuales pueden ser secuenciales o producirse en paralelo.
- Así mismo, la factoría dispone de un conjunto de recursos materiales y/o humanos para realizar los procesos de fabricación de los diferentes artículos.
- Existe un conjunto de pedidos u órdenes de producción de los distintos artículos a fabricar, con sus cantidades correspondientes.
- La producción de cada pedido genera tantas operaciones de fabricación como procesos componen el plan de fabricación del artículo correspondiente.
- Existe un número de restricciones, variable según el caso, que deben satisfacerse total o parcialmente para que el programa obtenido sea válido. Así, puede haber restricciones asociadas al plan de fabricación de cada artículo (orden de precedencia en la realización de los procesos, tiempos máximo y mínimo entre procesos consecutivos), restricciones asociadas a los recursos (limitaciones en la operatividad y capacidad de las máquinas, disponibilidad de los operarios), y restricciones asociadas a los pedidos (fechas de lanzamiento y entrega).

- El objetivo de la programación de producción es decidir las asignaciones de recursos a las diversas operaciones de los pedidos de producción con sus correspondientes intervalos de tiempo, preservando las restricciones, optimizando el uso de los recursos, y minimizando costes y tiempos.

El problema de la programación de producción, al igual que la mayoría de los problemas de asignación de recursos, es un problema de tipo NP completo, es decir, un problema cuyo tiempo computacional de resolución crece exponencialmente al aumentar la dimensión del mismo [GARE79]. Debido a ello, aunque se han estudiado y desarrollado a lo largo de los años muchas técnicas de resolución, los problemas de programación de producción de cierto tamaño todavía presentan en la práctica enormes dificultades.

En los apartados siguientes se describen en detalle las características generales de todos los elementos de una planta industrial, así como las restricciones y objetivos más importantes de la programación de producción.

6.1.2 Componentes de una planta industrial.

Artículos

En una planta industrial se fabrican generalmente varios artículos diferentes. Las características que pueden presentar estos artículos son:

- El plan de fabricación de un artículo se compone de uno o más procesos individuales, que deben llevarse a cabo en su totalidad para la elaboración correcta del artículo. Normalmente el plan de fabricación de un artículo es único, es decir, no existen conjuntos alternativos de procesos que permitan la fabricación del mismo.
- El número de procesos de que consta el plan de fabricación de los artículos de la factoría puede ser constante, o bien ser diferente para cada artículo.
- Los procesos necesarios para fabricar un artículo se realizan habitualmente de forma secuencial, si bien en determinados casos algunos de ellos pueden realizarse en paralelo.

Recursos

Para efectuar la fabricación de los artículos, la factoría dispone de una serie de recursos que pueden clasificarse en dos grupos: recursos de fabricación o principales, y recursos auxiliares:

- Los *recursos de fabricación*, también denominados *centros de trabajo* o, más sencillamente, *máquinas*, son los encargados de llevar a cabo activamente los procesos o tareas individuales de que se componen los planes de fabricación de los artículos.
- Los *recursos auxiliares* son aquellos que, si bien son necesarios para la producción, no realizan activamente las procesos de fabricación, y son fundamentalmente de cuatro tipos: recursos humanos (operarios que controlan las máquinas), utensilios y herramientas, materias primas, y suministros energéticos.

Normalmente, en la resolución del problema sólo se tienen en cuenta los recursos de fabricación o máquinas. Las características generales de las máquinas de una factoría son las siguientes:

- Cada máquina de la planta industrial es capaz de realizar al menos un proceso componente del plan de fabricación de un artículo, siendo habitual que una misma máquina pueda realizar varios procesos correspondientes a diversos artículos.
- Así mismo, toda máquina posee una duración y un coste característicos por unidad de artículo procesado, para cada proceso que pueda realizar .
- En una planta pueden existir varios tipos diferentes de máquinas, así como varias máquinas de un mismo tipo. Dos máquinas se consideran *iguales* si pueden realizar los mismos procesos con las mismas características de duración y coste. De la misma forma, dos máquinas se consideran *equivalentes* en cuanto a funcionalidad si pueden realizar los mismos procesos, pero con distintas duraciones y costes. Finalmente, las máquinas se consideran *distintas* si no pueden realizar los mismos procesos. De acuerdo con esto, pueden establecerse entre las máquinas diferentes niveles de calidad, particularizados para cada proceso de fabricación.
- El concepto temporal de la duración de un proceso en una máquina puede descomponerse a su vez en otros dos: el *tiempo de preparación* (set-up time), o

tiempo necesario para acondicionar la máquina antes de realizar el proceso, y el *tiempo de proceso* (processing time) o duración estricta del proceso.

- En el caso más general, dado que una máquina puede realizar varios procesos diferentes, y un mismo proceso puede ser realizado por diferentes máquinas, aparece el concepto de *ruta de fabricación* de un artículo como cada uno de los posibles caminos o secuencias de máquinas que pueden tomarse para llevar a cabo su producción. Este concepto no debe confundirse con el de plan de fabricación, ya que una ruta de fabricación es la secuencia de máquinas capaces de realizar los procesos que constituyen el plan de fabricación de un artículo. Cada ruta de fabricación tendrá asociados una duración mínima y un coste por unidad de artículo, cuyos valores respectivos serán la suma de las duraciones y costes de los procesos componentes en las máquinas que forman la ruta. De lo expuesto anteriormente se deduce que rutas de fabricación formadas por máquinas correspondientes iguales tendrán la misma duración y coste, y rutas de fabricación formadas por máquinas correspondientes equivalentes, pero de distintos tipos, tendrán diferentes duraciones y costes asociados.
- En un sistema realista de programación, toda máquina debe someterse a revisiones periódicas de puesta a punto (actividades de mantenimiento) durante las que no se encuentra disponible para llevar a cabo los procesos de producción.
- Una última característica relevante asociada a los recursos de fabricación de una factoría es la dimensión temporal en que éstos son operativos. Así, puede haber plantas que operan en tiempo continuo (24 horas al día, todos los días del año), o bien pueden existir calendarios de trabajo con días de fiesta y vacaciones, e incluso turnos de trabajo variables a lo largo del año. Así mismo, estos calendarios pueden estar particularizados para cada recurso de la planta industrial.

Pedidos u órdenes

El planteamiento del problema de la programación de producción se fundamenta en la existencia de un conjunto de pedidos u órdenes de fabricación, cada uno correspondiente a una cierta cantidad de un artículo determinado, los cuales deben producirse en la planta o factoría. Sus características generales son:

- Todo pedido hace referencia a una cantidad definida de un único artículo.

- Todo pedido lleva asociados dos instantes de tiempo importantes: la *fecha de lanzamiento* (release date), o momento lo antes posible en que puede comenzar la fabricación del pedido, y la *fecha de entrega* (due date), o momento lo más tarde posible en que puede terminar su fabricación. La fecha de lanzamiento se asocia al instante en que se cumplen todos los requisitos necesarios para comenzar la fabricación del pedido. La fecha de entrega corresponde al instante en que el pedido debe entregarse al cliente o pasar a almacén.
- La producción de un pedido consiste en la realización de tantas operaciones como procesos forman parte del plan de fabricación del artículo correspondiente. Por tanto, las operaciones de los pedidos pueden verse como las instancias o particularizaciones de los diferentes procesos de fabricación de los artículos.
- Las operaciones de fabricación de un pedido se realizan mediante los recursos disponibles. La asignación de una máquina para la realización de una operación conlleva fijar un instante de comienzo y un instante de fin para dicha operación. Durante este intervalo de tiempo, el recurso asignado no estará disponible para la realización de otras operaciones.

6.1.3 Restricciones de la programación.

El concepto de restricción en el problema de la programación de producción hace referencia a cada uno de los requisitos que deben cumplir los distintos componentes de la planta industrial para que los programas de producción obtenidos en la resolución del problema sean factibles y, por tanto, válidos. Las restricciones del problema de la programación de producción pueden clasificarse en tres tipos: restricciones asociadas a los artículos, restricciones asociadas a los recursos, y restricciones asociadas a los pedidos.

Restricciones asociadas a los artículos

Las restricciones asociadas a los artículos se refieren fundamentalmente a las características de los planes de fabricación de los mismos. Las principales restricciones asociadas a los artículos que aparecen en el problema de la programación de producción son las siguientes:

- *Restricciones de precedencia entre los procesos de fabricación de un artículo.* En la mayoría de los sistemas de producción, la fabricación de cada artículo está

formada por una secuencia invariable de procesos, de tal forma que cada uno de ellos debe realizarse inmediatamente después del anterior e inmediatamente antes que el posterior, de acuerdo con el plan de fabricación del artículo.

- *Restricciones de tiempo máximo y mínimo entre procesos consecutivos de fabricación de un artículo.* En determinados casos, tales como procesos sensibles al tiempo (procesos químicos, metalúrgicos, etc.), puede ser necesario establecer este tipo de restricciones que limitan el intervalo temporal de espera entre la realización de procesos consecutivos de un artículo.

Las restricciones sobre artículos que se han indicado son sólo las más características e importantes, si bien en una factoría concreta pueden existir otras restricciones adicionales que también deberían tenerse en cuenta.

Restricciones asociadas a los recursos

Las restricciones asociadas a los recursos (máquinas) se refieren fundamentalmente a los límites de funcionalidad de los mismos. Las principales restricciones de este tipo que aparecen en el problema de la programación de producción son:

- *Restricciones de operatividad de las máquinas.* Una máquina no puede realizar cualquier proceso de fabricación, sino sólo aquellos que permita su funcionalidad específica.
- *Restricciones de capacidad de las máquinas.* Una máquina sólo puede llevar a cabo un número limitado de operaciones de fabricación en un intervalo de tiempo determinado, dado que tiene un rendimiento máximo que no puede sobrepasarse. Esta limitación se debe a la invariabilidad de la característica de duración que posee toda máquina con respecto a cada uno de los procesos específicos que puede realizar.
- *Restricciones de disponibilidad de las máquinas.* Una máquina sólo puede realizar una operación de fabricación en cada instante. Así mismo, una máquina averiada o en mantenimiento no puede realizar operaciones de fabricación durante el tiempo de reparación o puesta a punto.
- *Restricciones de interdependencia entre las máquinas.* Puede suceder que dos máquinas diferentes no puedan estar ambas en funcionamiento en el mismo instante, debido a alguna característica específica de la planta.

Las restricciones sobre recursos que se han indicado son algunas de las más importantes, si bien en una factoría concreta pueden existir otras restricciones adicionales que también deberían tenerse en cuenta.

Restricciones asociadas a los pedidos

Las principales restricciones de este tipo que aparecen en el problema de la programación de producción son las *restricciones referentes al intervalo de fabricación de los pedidos*. Cada pedido posee una fecha de lanzamiento y una fecha de entrega, las cuales no pueden ser sobrepasadas en la programación de las operaciones de fabricación del pedido.

6.1.4 Objetivos de la programación.

La resolución del problema de la programación de producción conlleva la asignación de recursos a las operaciones que constituyen los pedidos, de tal forma que se cumplan todas las restricciones impuestas. Sin embargo, una solución que simplemente satisfaga las restricciones no es suficiente, sino que se debe conseguir la solución que además optimice los objetivos del problema. En programación de producción los principales objetivos a lograr son los siguientes:

- *Reducción de los costes de producción.* La producción de un pedido genera unos costes que dependen en primer lugar de la ruta de fabricación empleada, así como de otros gastos adicionales, generalmente constantes e independientes del propio proceso de producción (gastos de personal, administrativos, materias primas).
- *Reducción del tiempo de fabricación.* Reducir el tiempo total de fabricación de cada pedido es importante a fin de recuperar cuanto antes las inversiones realizadas en materias primas y valor añadido, costes que no se recuperan hasta la entrega del pedido.
- *Optimización de la utilización de los recursos.* Optimizar el rendimiento de las máquinas de la planta es un objetivo fundamental de la organización, el cual se puede conseguir distribuyendo la carga de trabajo lo más posible.

Además de estos objetivos principales, pueden considerarse otros objetivos que formarían parte de un sistema más completo y realista. Estos objetivos adicionales pueden ser:

- *Cumplimiento de fechas de entrega.* Un retraso en la fecha de entrega de un pedido afecta a la satisfacción del cliente, e incluso puede provocar la pérdida de futuros pedidos del mismo, por lo que su cumplimiento es un objetivo muy importante para la empresa. La fecha de entrega es una restricción que puede considerarse de forma opcional como un objetivo de la producción, especialmente en sistemas en que la carga de trabajo sea suficiente como para llegar a saturar la capacidad total de la planta. En estos casos, si la fecha de entrega de los pedidos se impone como restricción, es posible que no exista ninguna solución válida para el problema. Por ello, suele ser conveniente plantear la fecha de entrega como una componente de la función objetivo, cuyo coste es nulo cuando los pedidos se fabrican dentro del límite definido por la misma, y crece al aumentar el retraso.
- *Preferencias en el proceso de fabricación.* Es posible considerar ciertos criterios, de carácter más o menos subjetivo, acerca de la mayor idoneidad de ciertas máquinas respecto a otras del mismo tipo, o sobre la prioridad de unos pedidos sobre otros. Estas preferencias, fruto de la experiencia del personal encargado de la programación de producción, pueden estimarse aproximadamente, y formar parte de la función objetivo a optimizar.

6.1.5 Modelos de planta industrial.

Dentro del problema de la programación de producción existen dos modelos principales de planta o factoría, la planta de fabricación continua (flow-shop), y la planta de fabricación discreta (job-shop), que dan origen a los dos tipos fundamentales de problemas de programación de producción: la programación de producción continua (flow-shop scheduling), y la programación de producción discreta (job-shop scheduling).

Planta de fabricación continua

Una planta de fabricación continua se caracteriza porque todas las máquinas o centros de trabajo se disponen en forma de cadena, de manera que las operaciones de los pedidos se procesan secuencialmente una detrás de otra en forma de flujo continuo. Este tipo de

fabricación es muy común en procesos industriales que requieren una continuidad en el desarrollo de las operaciones que componen el pedido, es decir, procesos muy restringidos en cuanto a tiempos de espera mínimo y máximo entre operaciones (p.e., procesos químicos, metalúrgicos).

El problema de la programación de producción continua es, por tanto, un problema de secuenciamiento de pedidos, que puede definirse de la siguiente forma: Sea una factoría destinada a producir un conjunto de A artículos diferentes. El plan de fabricación de todos los artículos se compone de un mismo número de procesos o tareas que deben realizarse en un orden establecido. Para realizar la producción, la planta industrial dispone de un conjunto de M máquinas o centros de trabajo dispuestos en serie, los cuales realizan secuencialmente los procesos de fabricación de los artículos. Se desea producir un conjunto de N pedidos u órdenes, cada uno de una cierta cantidad de un artículo determinado. El objetivo de la programación es *encontrar el orden o secuencia de producción de los pedidos que, preservando las restricciones del sistema, minimice los costes y tiempos de fabricación.*

Planta de fabricación discreta

Una planta de fabricación discreta se caracteriza porque las máquinas o centros de trabajo son independientes, esto es, no forman parte de una cadena física de producción, y porque pueden existir varias máquinas capaces de realizar los mismos procesos de tal manera que los pedidos pueden producirse en paralelo. Este tipo de fabricación es más general que la fabricación continua y es propia de la producción de bienes de equipo.

El problema general de la programación de producción discreta es, por tanto, un problema prototipo de asignación de recursos, que puede definirse de la siguiente forma: Sea una factoría destinada a producir un conjunto de A artículos diferentes. El plan de fabricación de cada artículo se compone de varios procesos o tareas que deben realizarse en un orden establecido. La planta industrial dispone para realizar la producción de un conjunto de M máquinas o centros de trabajo independientes, en los que se realizan las operaciones de que se componen los pedidos. El objetivo de la programación es *encontrar la asignación óptima de recursos con sus correspondientes intervalos de tiempo a las diversas operaciones de los pedidos de producción, preservándose las diferentes restricciones impuestas en el problema.*

6.1.6 Programación predictiva y programación reactiva.

Los sistemas de programación de producción, especialmente en programación discreta, pueden poseer carácter predictivo o reactivo:

- Un sistema es *predictivo* si no tiene en cuenta los sucesos imprevisibles que pueden producirse en tiempo real sobre la planta, tales como averías de máquinas, falta imprevista de materias primas, o llegada de nuevos pedidos que deben producirse de forma muy urgente. En general, los sistemas de programación predictiva realizan el programa de producción siempre sobre el conjunto total de los pedidos a fabricar, y tomando como base un intervalo de tiempo predeterminado, por ejemplo, diario, semanal, o mensual.
- Un sistema es *reactivo* si tiene en cuenta averías de máquinas, falta de materiales, pedidos urgentes, u otros sucesos imprevisibles que pueden producirse en la planta, efectuándose cuando sea necesario una reprogramación parcial de los pedidos afectados por el suceso. Lógicamente, en este tipo de sistemas, cada vez que se debe elaborar el programa correspondiente a un nuevo conjunto de pedidos, se tiene en cuenta la carga de trabajo actual de la factoría.

La distinción entre programación predictiva y programación reactiva es importante porque muchos sistemas sólo generan planes de producción predictivos, sin tener en cuenta sucesos imprevisibles. Por el contrario, otros sistemas tienen en cuenta sucesos en tiempo real, y crean programas de producción más realistas. Estos dos tipos de sistemas son apropiados para diferentes aplicaciones, por lo que es importante conocer si un sistema de programación de producción es predictivo o reactivo.

6.2 MÉTODOS HEURÍSTICOS ESPECÍFICOS DE RESOLUCIÓN DE LA PROGRAMACIÓN DE PRODUCCIÓN DISCRETA.

6.2.1 Introducción.

Como se ha indicado anteriormente, el problema de la programación de producción consiste en la asignación de recursos en el tiempo para la realización de tareas dirigidas a la fabricación de artículos en una planta de producción. Los problemas de este tipo se

caracterizan por la existencia de un conjunto de restricciones de diversa clase (orden de precedencia en la realización de los procesos de fabricación de los artículos, disponibilidad y capacidad de los recursos, fechas de lanzamiento y entrega de los pedidos), y generalmente poseen un espacio de búsqueda que crece exponencialmente con relación a su dimensión, es decir, son problemas NP-completos.

El propósito de este apartado es revisar las diferentes técnicas de resolución del problema de la programación de producción basadas en la Inteligencia Artificial simbólica, y estudiar sus ventajas e inconvenientes. Existen diversas revisiones de este tipo sobre sistemas de programación de producción inteligentes, siendo las principales las de Tate [TATE85], Shaw y Whinston [SHAW86], Steffen [STEF86], y Atabakhsh [ATAB91].

6.2.2 Satisfacción de restricciones.

Generalidades

La satisfacción de restricciones tiene por objeto el estudio de problemas combinatorios caracterizados por la existencia de un conjunto de variables x_1, ..., x_n de dominios discretos y finitos, entre las que se establece un conjunto de restricciones o tuplas de valores de compatibilidad. El objetivo del problema es encontrar una asignación de valores de las variables que satisfaga todas las restricciones impuestas. Por tanto, estos problemas se caracterizan por la existencia de un gran, aunque finito, conjunto de combinaciones de valores de las variables (soluciones) entre las que se debe encontrar aquella o aquellas que satisfagan las restricciones del problema. Estos problemas son objeto de estudio principalmente en las áreas de Matemática Discreta, y de Computación e Inteligencia Artificial.

Al igual que en la optimización combinatoria, la mayoría de los problemas de satisfacción de restricciones sólo pueden resolverse mediante algoritmos de complejidad superpotencial con respecto a la dimensión del problema, es decir, se trata de problemas *NP-completos* (nondeterministic polynomial time complete problems), caracterizados porque, en la práctica, su resolución exacta es imposible en tiempos de ejecución razonables.

A continuación se proporciona una definición formal del problema de satisfacción de restricciones, y se mencionan los diferentes tipos de algoritmos de resolución más importantes. Una descripción detallada de estos algoritmos puede encontrarse en el artículo de Pedro Meseguer *Constraint Satisfaction Problems: An Overview*, [MESE89].

Definición formal

Definición. Un problema de satisfacción de restricciones, PSR, también llamado problema del etiquetado consistente, es una terna (X, S, R), donde:

- X es el conjunto de variables del problema, $X = \{x_1, ..., x_n\}$, de dominios discretos y finitos $D_1, ..., D_n$.
- S es el espacio de soluciones del problema, formado por todas las combinaciones de valores de las variables, es decir, $S = D_1 \times ... \times D_n$.
- R es el conjunto de restricciones, $R = \{r_1, ..., r_m\}$, cada una de las cuales es un grafo del producto cartesiano de los dominios de las variables relacionadas por la restricción, es decir, cada restricción r_j es un conjunto de tuplas que determinan los valores de compatibilidad entre un subconjunto determinado de las variables del problema.

El objetivo del problema consiste en encontrar una asignación simultánea de valores de las variables, $s \in S$, que satisfaga todas las restricciones impuestas.

En un problema de satisfacción de restricciones se denomina *grado* de una variable a la cardinalidad de su dominio, y *aridad* de una restricción al número de variables que relaciona. El grado del problema es el mayor grado de sus variables, y la aridad del problema es la mayor aridad de sus restricciones. A fin de simplificar el análisis del PSR, y sin pérdida de generalidad [NUDE83], pueden realizarse las siguientes hipótesis:

- Se eligen los dominios de las variables de forma que todos posean la misma cardinalidad.
- En la formulación del problema sólo se utilizan restricciones binarias. Una restricción binaria es aquella que relaciona únicamente dos variables, x_i, x_j, y se denota por los subíndices de las variables relacionadas, r_{ij}.

Ejemplo: Problema de las reinas no amenazadoras. El problema de las reinas no amenazadoras (Non-threatening Queens Problem, NQP) consiste en encontrar la disposición de n reinas en un tablero de ajedrez de n^2 cuadros, de tal forma que ninguna reina pueda comer a las restantes. Formalmente, el problema puede describirse mediante la terna (X, S, R):

- Conjunto de variables del problema $X = \{x_1, ..., x_n\}$, donde cada x_i representa la reina i a colocar en el tablero. El dominio finito D para cada una de las n variables es el conjunto de los n^2 cuadros de que consta el tablero de ajedrez.
- Espacio de soluciones $S = D^n$.
- Conjunto de restricciones binarias del problema $R = \{r_{ij} \mid i, j = 1, ..., n \wedge i \neq j\}$, donde cada r_{ij} representa el conjunto de pares de valores compatibles para las variables i y j, es decir, los pares de posiciones del tablero para los que las reinas i y j no pueden comerse mutuamente.

Un PSR puede verse como un grafo de restricciones, en el que los nodos representan las variables del problema y los arcos representan las restricciones binarias (figura 6.1). Por cada arco que parta de un nodo i y converja en un nodo j, correspondiente a la restricción r_{ij}, existe un arco simétrico desde el nodo j al nodo i, correspondiente a la restricción r_{ji}, tal que $r_{ij}(x_i,x_j) = r_{ji}(x_j,x_i)$. Cuando dos nodos no se hallan conectados, existe entre ellos la llamada restricción universal, que consiste en que todas las posibles combinaciones de valores entre las dos variables son válidas. Así, en el grafo de la figura 6.1, las restricciones r_{14} y r_{34} son universales, dado que no aparecen representadas..

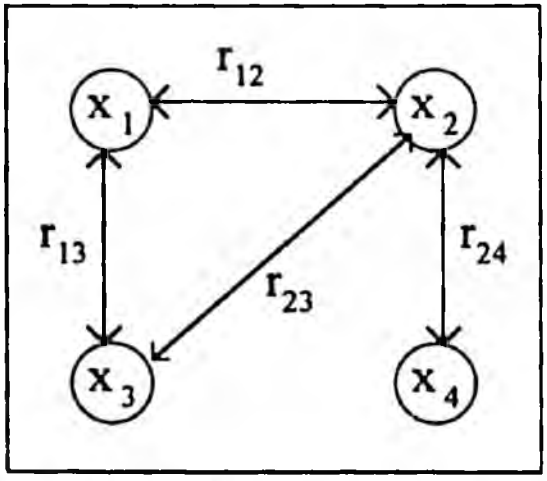

Figura 6.1: Grafo de restricciones binarias.

Existen diferentes planteamientos del PSR, los cuales implican la utilización de diferentes criterios en la resolución. Estos planteamientos consisten en variar el objetivo final del problema, de forma que se precise hallar una solución cualquiera, todas las soluciones, el número de soluciones, o si existe alguna solución. Generalmente, encontrar el número o la existencia de soluciones implica la generación de las mismas. Por esta razón sólo se considerarán en adelante los dos primeros planteamientos del problema.

Técnicas de resolución

La resolución de problemas de satisfacción de restricciones NP-completos se basa en la construcción de algoritmos apropiados sobre varias posibilidades. El algoritmo de resolución más sencillo realiza una búsqueda ciega sistemática sobre el espacio de búsqueda, de tal forma que cuando se llega a un punto de la misma sin posible continuación, se vuelve atrás reconsiderando la última decisión tomada por el proceso de búsqueda.

El algoritmo de búsqueda ciega sistemática consiste fundamentalmente en realizar asignaciones de valores a las variables y comprobar a continuación si las restricciones se satisfacen. Este algoritmo realiza una búsqueda exhaustiva por el espacio de estados del problema, y es capaz de encontrar todas las soluciones del mismo.

En cada paso de la búsqueda las variables ya asignadas se denominan *variables pasadas*, las variables sin asignar *variables futuras*, y la variable cuya asignación se realiza en ese paso *variable actual*. El algoritmo puede describirse de la siguiente forma (se precisa una estructura de datos de tipo pila).

Algoritmo. Satisfacción de restricciones mediante búsqueda ciega sistemática.

1. *Inicialización.* Se debe asignar a cada variable un valor indefinido. La pila está vacía.
2. *Seleccionar variable.* Se debe seleccionar como variable actual una de las variables aún no asignadas, introduciéndola a continuación en la pila. Si ya se han asignado todas las variables, ir al paso 5.
3. *Seleccionar valor.* Se debe seleccionar un valor no usado como nuevo valor de la variable actual, marcándolo como usado. Si ya se han usado todos los valores, ir al paso 6.

4. *Test.* Se debe comprobar si se satisfacen todas las restricciones que relacionan a la variable actual con las variables pasadas. Si las restricciones se cumplen ir al paso 2, en caso contrario ir al paso 3.

5. *Solución*: la asignación actual de valores de las variables es una solución del problema. Si sólo se necesita una solución, terminar, en caso contrario ir al paso 3.

6. *Retroceso (backtracking).* Se debe sacar la variable actual de la pila, asignándole un valor indefinido. La nueva variable actual será la que ocupe la cabecera de la pila. Si la pila está vacía, terminar, y en caso contrario, ir al paso 3.

En este algoritmo se establece un orden prefijado en la selección de variables (paso 2), en la selección de valores para cada variable (paso 3), y en la selección de variables pasadas (paso 4). Así mismo, en el retroceso se impone una ordenación cronológica en la selección de variables, debido al uso de una pila como estructura de almacenamiento de las variables pasadas. Se ha elegido este tipo de ordenación prefijada por razones de simplicidad, aunque no es esencial para el correcto funcionamiento del algoritmo.

La complejidad temporal del algoritmo es exponencial, y la complejidad espacial es logarítmica. Este algoritmo encuentra todas las posibles soluciones al problema debido a que realiza una búsqueda exhaustiva en el espacio de estados, pero es ineficiente en el sentido de que realiza más operaciones de las necesarias para encontrar una solución. Las causas de esta ineficiencia pueden resumirse en dos:

- *Perspectiva local.* En cada paso de la búsqueda el algoritmo centra su atención exclusivamente en la variable actual, y no posee una perspectiva global de la relación entre la variable actual con las restantes variables por asignar (variables futuras). Sería interesante, por ejemplo, comprobar que cada nuevo valor a asignar a la variable actual no fuera totalmente incompatible con los valores de alguna de las variables futuras, lo cual evitaría un retroceso posterior.
- *Memoria limitada.* Este algoritmo no recuerda las acciones realizadas previamente, por lo que en el retroceso se vuelven a verificar continuamente incompatibilidades ya detectadas con anterioridad sobre los valores de las variables.

Una vez que se han mostrado las razones de la ineficiencia del algoritmo de búsqueda ciega sistemática, el objetivo que se plantea es desarrollar algoritmos más eficientes capaces de resolver el PSR. Esto puede lograrse construyendo algoritmos que realicen

sólo las operaciones estrictamente necesarias para encontrar las soluciones del problema. No obstante, las mejoras en la eficiencia tienden a eliminar la simplicidad inicial del algoritmo, e incrementan su complejidad espacial.

Existen tres aproximaciones generales para mejorar la eficiencia, las cuales no dependen de la estructura particular del problema tratado:

- Diseñar algoritmos que reduzcan el espacio de búsqueda de forma previa al comienzo del proceso, suprimiendo las regiones que no contengan soluciones. Los algoritmos que emplean este método se llaman algoritmos que mejoran la consistencia, dado que todos ellos se basan en conseguir un determinado nivel de consistencia local. Los principales algoritmos de este tipo son: la propagación de restricciones, los algoritmos de consistencia local, los algoritmos de búsqueda sin retroceso (backtracking), y los algoritmos de búsqueda con retroceso limitado.
- Construir algoritmos que reduzcan el espacio de búsqueda durante el desarrollo de la misma. Los algoritmos que emplean este método se denominan algoritmos híbridos, porque en ellos el proceso de búsqueda y la reducción del espacio de búsqueda se producen a la vez. Los principales algoritmos de este tipo son: los algoritmos de anticipación (look-ahead), y los algoritmos de retrospección (look-back).
- Desarrollar heurísticas generales que ayuden a alcanzar una solución en la mayoría de los casos. En esta aproximación el proceso de búsqueda está dirigido por heurísticas que contienen información general sobre la estructura del espacio de búsqueda, pero que no dependen de las características particulares del problema a resolver. Las principales heurísticas de este tipo son: las heurísticas básicas que especifican el orden en que deben realizarse determinadas tareas (asignación de variables, asignación de valores, comprobación de variables pasadas), las heurísticas basadas en la teoría, y las heurísticas basadas en redes.

Una forma adicional de resolver el PSR consiste en emplear heurísticas basadas en la estructura particular del problema tratado, en cuyo caso el algoritmo deja de ser general y pasa a ser un algoritmo dependiente del problema. En general, los algoritmos más eficientes suelen combinar todos los métodos mencionados.

6.2.3 Representación de la programación de producción como un problema de satisfacción de restricciones.

Programación de producción y satisfacción de restricciones

En los últimos años se ha argumentado que la Inteligencia Artificial es una ciencia apropiada para la resolución del problema de programación de producción ([GRAN86], [PHEL86], [FOX90a], [FOX90b]), especialmente porque la resolución de este problema es una actividad que requiere gran cantidad de conocimiento y una búsqueda intensiva, y la IA proporciona mecanismos para capturar y representar el conocimiento de forma apropiada, así como técnicas de búsqueda eficientes.

Dentro de la Inteligencia Artificial simbólica, el campo de la satisfacción de restricciones en particular ha sido el más utilizado para realizar la aproximación al problema de programación de producción ([FOX83], [SMIT86], [PAPE88], [BURK89], [ELLE89], [FOX90b], [PROS89], [ATAB91], [SADE91]).

Sin embargo, la programación de producción es fundamentalmente un problema de optimización combinatoria, en el que debe optimizarse un conjunto de objetivos, preservando una serie de restricciones. Por ello, dado que en un problema de satisfacción de restricciones no hay objetivos a optimizar, debe aplicarse una transformación de dichos objetivos en restricciones para que el problema pueda representarse como un PSR. Esta transformación es inmediata, simplemente suponiendo unos valores límite del objetivo dentro de los cuales se considera aceptable la solución. Por ejemplo, el objetivo de minimización de costes puede considerarse una restricción tal que si la solución obtenida conlleva un coste total inferior a $c + \varepsilon$, donde c es el coste mínimo y ε es una cantidad arbitraria, entonces dicha solución se considera válida.

Una vez realizada la transformación de objetivos en restricciones, una forma inmediata de aplicar el problema de programación de producción sobre el PSR consiste en representar las operaciones de los pedidos como variables. El dominio de cada variable/operación es el producto cartesiano del conjunto de posibles recursos que pueden realizar la operación por el conjunto de posibles instantes de comienzo de la operación. El problema consiste en asignar a cada variable un par recurso-tiempo, de manera que se satisfagan todas las restricciones.

Representación del conocimiento

La complejidad de la información acerca de la planta industrial cuyos procesos se van a programar, así como sobre los objetivos de la programación y las restricciones que afectan al problema, requiere una forma apropiada de representación. La técnica de representación del conocimiento debe ser modular, flexible, eficiente y fácil de entender y manejar. Las principales técnicas usadas actualmente en sistemas de programación de producción basados en restricciones son las habituales en programas de IA simbólica, tales como redes semánticas, marcos (frames), lógica de predicados, y sistemas de reglas.

Técnicas heurísticas de resolución

Además de las técnicas de representación del conocimiento, los sistemas de programación de producción utilizan diferentes metodologías en la generación de los programas de producción, basándose la mayoría de ellas en las diferentes técnicas de satisfacción de restricciones existentes [MESE89]. Además, los sistemas basados en restricciones suelen adoptar una característica fundamental que se explica a continuación: la *relajación de restricciones*.

En la búsqueda dirigida por restricciones, el proceso de asignación de recursos a las operaciones de producción puede verse detenido ante una situación de bloqueo, manifestada como una falta de capacidad en el recurso necesario para realizar una operación determinada, llamada *operación crítica o problemática*, en el intervalo de tiempo durante el que debe realizarse la misma. Cuando sucede esto, el recurso se convierte en lo que se conoce como *cuello de botella*. Una situación de bloqueo como ésta puede deberse a que la búsqueda se halla en un camino no viable, o a que, simplemente, la falta de capacidad es real (no hay capacidad suficiente en la planta para realizar todo el trabajo previsto) por lo que no existe ninguna solución que verifique todas las restricciones impuestas inicialmente. Dadas las características heurísticas de la búsqueda dirigida por restricciones, no es posible discriminar cuál de las dos razones mencionadas provoca realmente el cuello de botella. Por ello, la mayoría de los sistemas optan en primer lugar por realizar un retroceso (backtracking) limitado, y, si éste fracasa en la resolución del conflicto, realizan un proceso de relajación o ampliación de los límites de ciertas restricciones, de forma que pueda encontrarse al menos una asignación de recursos que satisfaga las restricciones relajadas.

Sin embargo, no todas las restricciones son relajables. En realidad, en una representación basada en restricciones del problema existen dos tipos de restricciones: *restricciones inviolables* (hard constraints) y *restricciones relajables* (soft constraints). Las restricciones relajables son las correspondientes a los objetivos del problema original de optimización, esto es, minimización de costes de producción, reducción del tiempo de proceso, optimización del uso de las máquinas, cumplimiento de fechas de entrega, y preferencias en el proceso de fabricación. El resto de las restricciones sobre artículos (precedencias en la realización de las procesos de fabricación), y sobre recursos (operatividad, dispombilidad, capacidad e interdependencia de las máquinas) no pueden relajarse.

6.2.4 Sistemas de programación basados en restricciones.

Los sistemas de programación de producción existentes en la actualidad utilizan diferentes técnicas de representación del conocimiento, y diferentes metodologías y criterios de programación basada en restricciones para la resolución del problema tratado. En este apartado se describen brevemente algunos de los más importantes sistemas de programación de producción basados en restricciones, con objeto de introducir las características fundamentales de los mismos, así como de proporcionar una perspectiva general del desarrollo histórico de este área.

ISIS

El sistema ISIS [FOX83] fue desarrollado en la Universidad Carnegie Mellon, y constituye uno de los primeros sistemas de programación de producción inteligentes basados en la técnica de satisfacción de restricciones. Este sistema proporciona una metodología para representar y utilizar la amplia variedad de restricciones presentes en el dominio de la programación de producción discreta. El conocimiento proporcionado por las restricciones se usa para reducir el espacio de soluciones, generar y discriminar la información entre varios niveles de búsqueda, y diagnosticar las soluciones no satisfactorias propuestas.

ISIS posee un módulo de relajación de restricciones, que se utiliza cuando surge algún conflicto en la asignación de los recursos. Posee también un módulo de reprogramación, que se usa cuando alguna asignación ya realizada se invalida por una acción inesperada

(por ejemplo, la avería de una máquina). En este caso, sólo se reprograman las órdenes afectadas.

OPIS

El sistema OPIS [OW86], sucesor del ISIS y aplicado igualmente a la programación de producción discreta, posee una arquitectura jerárquica basada en el modelo de pizarra con un gestor de alto nivel que supervisa la coordinación de las fuentes de conocimiento necesarias para implementar el comportamiento oportunistico y generar el plan de producción. En este sistema se propone una estrategia para aplicar simultáneamente la perspectiva orientada a los pedidos y la orientada a los recursos en la descomposición del problema de la programación de producción.

CSS

El sistema CSS [OW88], también desarrollado en la Universidad Carnegie Mellon, es un sistema distribuido de programación de producción discreta diseñado para grandes plantas de fabricación. El sistema CSS adopta la estrategia del OPIS de emplear múltiples perspectivas del problema. Sin embargo, mientras OPIS utiliza las diferentes perspectivas como aproximaciones alternativas para realizar las decisiones específicas de programación, CSS combina ambas perspectivas en cada toma de decisión. En CCS, la coordinación de las fuentes de conocimiento no precisa de un gestor de alto nivel, sino que ésta se consigue mediante un proceso de negociación entre las propias fuentes de conocimiento.

SOJA y SONIA

El sistema SOJA [PAPE85] está diseñado para construir planes de producción diarios. Con este fin, utiliza un motor de inferencia de encadenamiento oportunistico, que gestiona una base de conocimientos formada por reglas que contienen predicados y funciones LISP. Su sucesor SONIA [COLL88] es un sistema de programación de producción discreta diseñado para detectar y reaccionar a las inconsistencias entre el plan generado y la situación real en la planta de fabricación. Por tanto, SONIA integra módulos de programación predictiva y reactiva. La estructura de control del sistema se basa en el modelo de pizarra y utiliza también razonamiento oportunistico.

OPAL

El sistema OPAL [BENS88], diseñado igualmente para programación de producción discreta, utiliza reglas de producción y heurísticas para determinar prioridades entre las operaciones a programar. Así mismo, el sistema usa una estrategia oportunística para programar en primer lugar el recurso con mayor carga de trabajo. Cuando no puede encontrarse una solución factible el sistema realiza retroceso, pero no incluye un módulo de relajación de restricciones.

PATRIARCH

El sistema PATRIARCH [LAWR86], un proyecto conjunto entre la Universidad Carnegie Mellon e IBM, utiliza un enfoque jerárquico para el tratamiento de la complejidad de los entornos de fabricación. El objetivo del sistema es proporcionar un sistema de soporte de producción en tiempo real para programar los procesos de fabricación en una planta industrial real. Incorpora una combinación de algoritmos heurísticos de programación y sistemas de reglas de producción.

INTESIMPRO (INTEgración de SIMulación y PROducción)

El sistema INTESIMPRO [LAZA92] [LAZA93] es un planificador reactivo para sistemas de producción discreta, diseñado para una amplia variedad de entornos de fabricación basados en órdenes o pedidos. El sistema emplea una estrategia de programación incremental sobre una perspectiva orientada a las órdenes, con módulos de retroceso y relajación de restricciones, los cuales entran en acción cuando se producen situaciones de conflicto. Así mismo, el sistema es capaz de reaccionar a sucesos no previsibles, tales como órdenes muy urgentes, avería de máquinas y falta imprevista de materiales.

6.3 NEUROPROG: UN PROGRAMADOR PREDICTIVO PARA SISTEMAS DE PRODUCCIÓN DISCRETA.

Como ejemplo de aplicación práctica de la técnica de resolución de problemas combinatorios basada en el optimizador discreto estocástico, se presenta a continuación un programador predictivo para sistemas de producción discreta: NEUROPROG.

NEUROPROG es un modelo basado en la red ODE, que se ha desarrollado siguiendo las pautas descritas en el capítulo anterior para la definición de un sistema conexionista aplicado a la resolución de problemas de optimización combinatoria. Este sistema permite resolver una instancia simplificada del problema de la programación de producción discreta.

6.3.1 Modelo de planta industrial implementado.

A continuación se describen las características del modelo de planta industrial asociado a la instancia del problema de la programación de producción discreta que puede resolverse mediante NEUROPROG.

Sea una planta de fabricación con m máquinas destinadas a la producción de a artículos. El plan de fabricación de cada artículo consta de un solo proceso. Cada máquina i puede realizar un subconjunto del total de procesos de fabricación, con distintos valores de costes $\{c_{ij}\}$ y duraciones $\{d_{ij}\}$ unitarios asociados a cada proceso j. Los tiempos de puesta a punto y de transporte entre máquinas se consideran despreciables. Existen n pedidos u órdenes a programar, de manera que cada pedido i especifica la producción de q_i unidades de un mismo artículo. Las restricciones del problema se reducen a las propias de disponibilidad y capacidad de las máquinas. Se desea encontrar la asignación de recursos (máquinas) a los pedidos de forma que se minimice el coste y el tiempo de fabricación, y se optimice la utilización de las máquinas.

Formalmente, el problema puede describirse mediante la cuádrupla (X, S, f, R):

- Conjunto de variables del problema $X = \{x_1, ..., x_n\}$, donde cada x_i representa el pedido i a programar. El dominio finito D para cada una de las n variables es el conjunto de las m máquinas de la planta industrial.
- Espacio de soluciones $S = D^n$

- Función objetivo a minimizar, f: S → R, determinada por la expresión:

$$f(s) = F_1 \sum_{i=1}^{n} c(x_i, \alpha(i)) q(i) + F_2 \sum_{i=1}^{n} d(x_i, \alpha(i)) q(i) + F_3 \sum_{i=1}^{n} \sum_{j=i+1}^{n} \delta(x_i, x_j)$$

donde $\alpha(i)$ es el artículo a fabricar en el pedido i, $c(x_i, \alpha(i))$ es el coste unitario de fabricación del artículo $\alpha(i)$ en la máquina x_i, $d(x_i, \alpha(i))$ es la duración unitaria de fabricación del artículo $\alpha(i)$ en la máquina x_i, $q(i)$ es la cantidad de artículos a fabricar en el pedido i, δ es la función delta de Kroenecker, definida:

$$\delta(i,j) = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

y F_i son factores de normalización asociados a cada término de la función objetivo, los cuales vienen determinados por las características de la planta industrial. Por ejemplo, si desea dar la misma importancia a los tres términos de la función objetivo, estos factores deben ser:

$$F_1 = \frac{1}{\max\{c_{ij}\}} \qquad F_2 = \frac{1}{\max\{d_{ij}\}} \qquad F_3 = \frac{2 \cdot \max\{q_i\}}{n-1}$$

- Conjunto de restricciones del problema R = {r}, donde r determina todos los programas válidos, es decir, aquellos en los que a cada pedido se le asigna una máquina capaz de efectuar el proceso de fabricación correspondiente.

6.3.2 Representación del problema mediante el modelo ODE.

La implementación del problema de la programación de producción discreta mediante la red ODE implica determinar la arquitectura apropiada que identifique los nodos de la red con las variables y soluciones del problema, transformar la función objetivo y las restricciones del problema en una función de energía apropiada para este modelo neuronal, y finalmente obtener los pesos de las conexiones a partir de dicha función de energía.

La representación del problema como una matriz de nodos, en la que las filas representan las variables, y las columnas los valores posibles del dominio, es la forma más intuitiva y clara de implementar un problema de optimización combinatoria en una red neuronal recurrente. Por ello, esta forma de representación simétrica del problema es la que se utiliza en la descripción que se ofrece a continuación. La representación matricial del problema puede transformarse inmediatamente en la apropiada para el modelo ODE,

simplemente considerando que cada fila de la matriz-red constituye un vector de nodos principales-auxiliares de la capa visible de la red ODE, y que los pesos de las conexiones entre los nodos de la matriz se corresponden con los pesos de las conexiones de segundo orden que entran en el nodo oculto de la red ODE.

Las tareas de implementación del problema de la programación de producción discreta se detallan a continuación:

- La arquitectura apropiada para representar la instancia especificada del problema es la de una red de N = n x m nodos dispuestos en forma de matriz, en la que las n filas representan las variables del problema, es decir, los pedidos a programar, y las m columnas los valores posibles de las variables, esto es, las máquinas a asignar a los pedidos (figura 6.2).

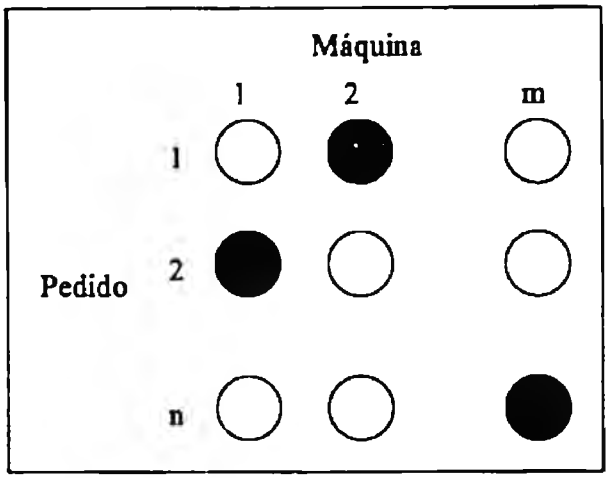

Figura 6.2: Arquitectura de red para el problema de la programación de producción.

- La función de energía E de la red para este problema toma la expresión (en notación matricial de doble subíndice):

$$E = -\frac{1}{2}W\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{m}a_{ik}a_{jl}(1-\phi(\alpha(i),k)\phi(\alpha(j),l))$$

$$+\frac{F_1}{2}\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{m}c(k,\alpha(i))q(i)a_{ik}a_{jl}\delta((i \bmod n)+1,j)$$

$$+\frac{F_2}{2}\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{m}d(k,\alpha(i))q(i)a_{ik}a_{jl}\delta((i \bmod n)+1,j)$$

$$+\frac{F_3}{2}\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{m}a_{ik}a_{jl}\delta(k,l)$$

El primer término corresponde a la restricción de dominio del problema (incapacidad de las máquinas), en el que $\phi(\alpha(i),j)$ es una función binaria que devuelve 1 si la máquina j es capaz de realizar el proceso de fabricación del artículo $\alpha(i)$, y 0 en caso contrario. Los términos segundo, tercero, y cuarto corresponden a los objetivos de la optimización, minimizar costes y tiempos y optimizar el uso de las máquinas.

- Finalmente, los pesos de las conexiones de la red adoptan la siguiente expresión (en notación matricial de doble subíndice):

$$w_{ik,jl} = W(1-\phi(\alpha(i),k)\phi(\alpha(j),l))$$
$$-F_1 c(k,\alpha(i))q(k)\delta((i \bmod n)+1,j)$$
$$-F_2 d(k,\alpha(i))q(k)\delta((i \bmod n)+1,j)$$
$$-F_3\delta(k,l)$$

donde $i \neq j$ ó $k \neq l$, es decir, $w_{ik,jl}$ no corresponde a una conexión de lazo, y δ es la función delta de Kroenecker. El primer sumando representa la restricción de dominio del problema, el segundo el objetivo de minimización de costes, el tercero el objetivo de minimización de tiempos, y el cuarto el objetivo de optimización del uso de las máquinas. El valor de umbral θ del nodo oculto de la red ODE se determina como una cota superior del máximo coste u objetivo del problema:

$$\theta = nF_1\max\{c_{ij}\}\max\{q_i\} + nF_2\max\{d_{ij}\}\max\{q_i\} + \frac{n(n-1)}{2}F_3$$

Finalmente, el valor apropiado de la inhibición global W para este problema es:

$$W = -2\theta = -2nF_1\max\{c_{ij}\}\max\{q_i\} - 2nF_2\max\{d_{ij}\}\max\{q_i\} - n(n-1)F_3$$

La red ODE construida de esta manera permite resolver de forma aproximada cualquier instancia del problema de la programación de producción discreta que se refiera a la planta industrial especificada.

6.3.3 Características generales del software desarrollado.

El sistema NEUROPROG se ha desarrollado íntegramente en lenguaje C, existiendo una versión para computadores PC con un interfaz gráfico de usuario basado en el uso de ventanas y menús desplegables. Para su realización se ha empleado una metodología

específica de diseño y programación modular, denominada programación basada en máquinas abstractas, que permite simular las características de la programación orientada a objetos en un lenguaje imperativo y modular como el ANSI C. Se ha elegido para este desarrollo el lenguaje C, en lugar de utilizar un lenguaje orientado a objetos como C++, por sus mejores características de eficiencia y portabilidad.

La figura 6.3 muestra el esquema entidad-relación extendido correspondiente al sistema NEUROPROG. Así mismo, la figura 6.4 muestra el aspecto del interfaz de usuario correspondiente a la versión de NEUROPROG en entorno PC.

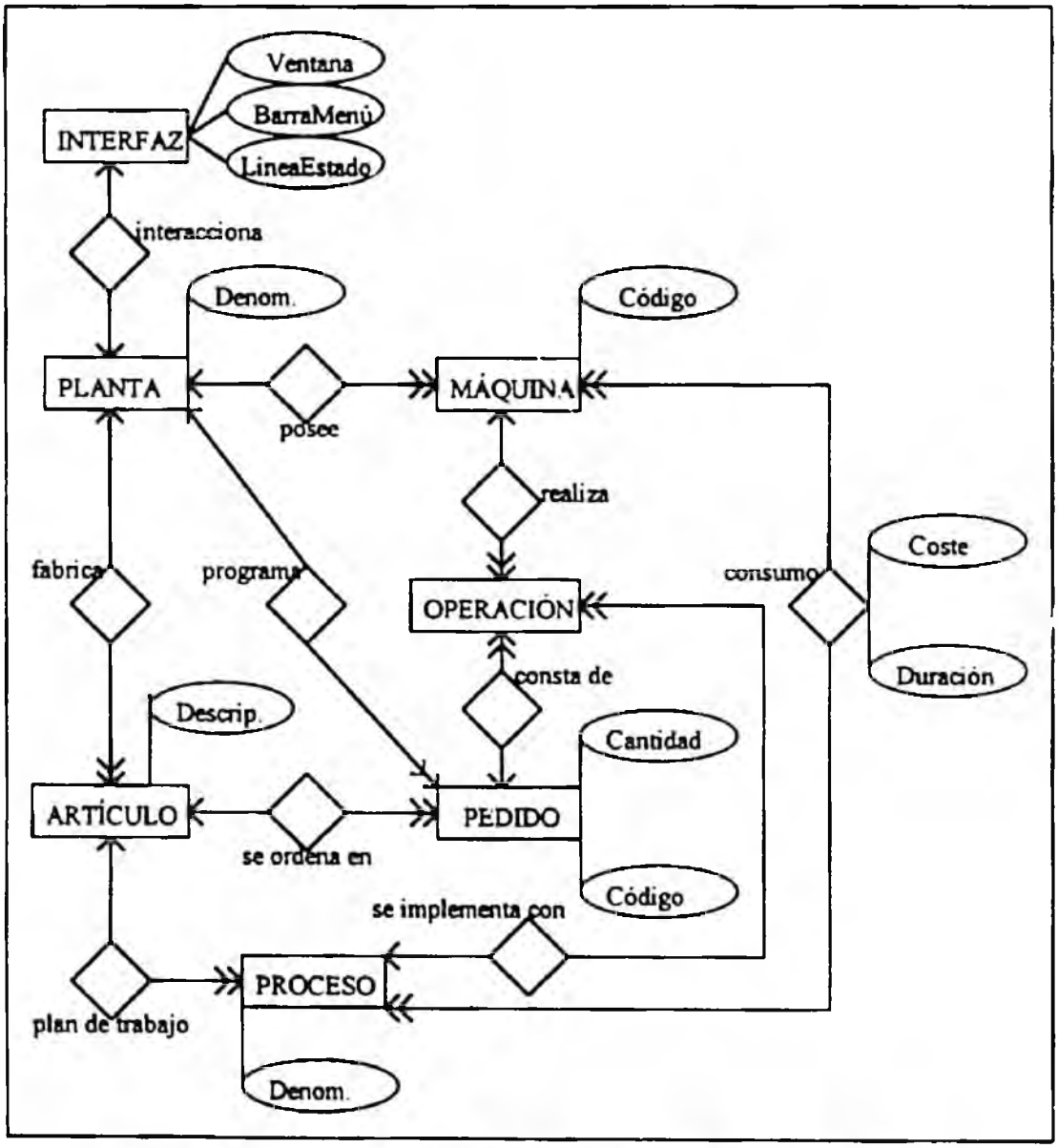

Figura 6.3: Diagrama entidad-relación del sistema NEUROPROG.

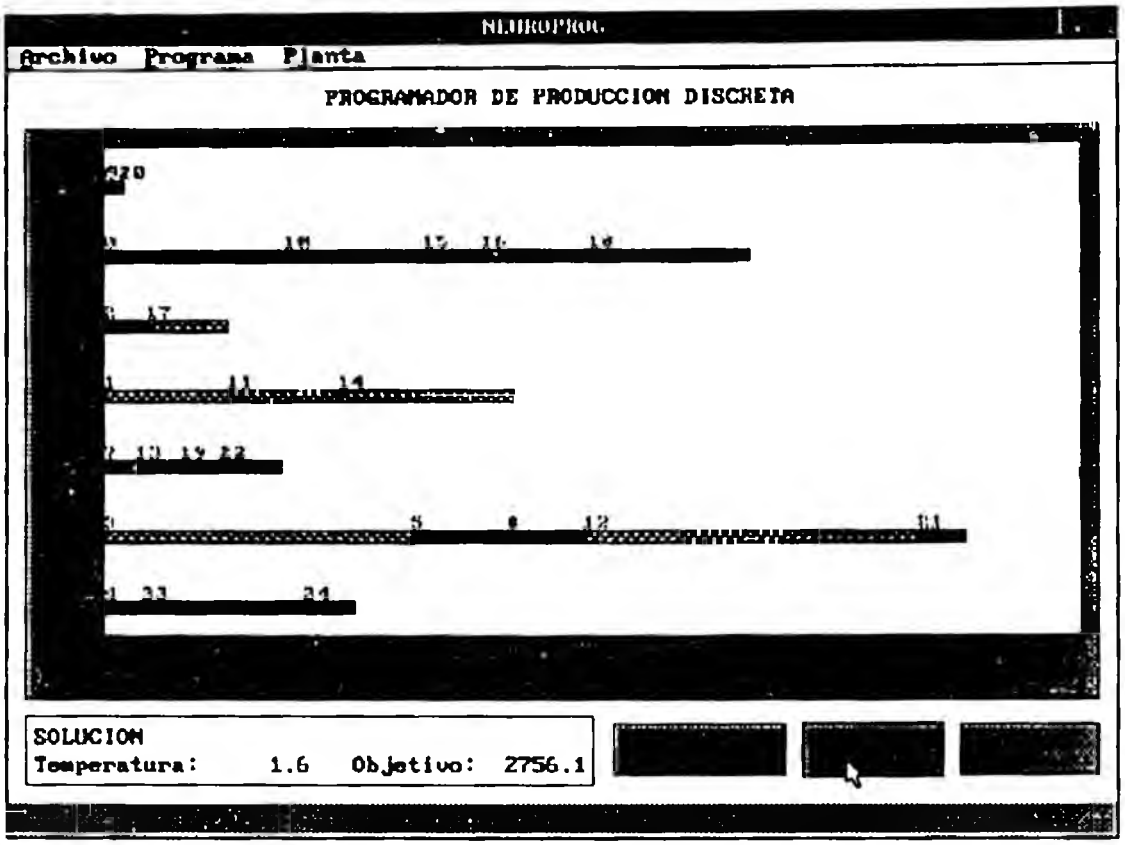

Figura 6.4: Aspecto del interfaz de usuario del sistema NEUROPROG para PC.

6.4 EVALUACIÓN DE NEUROPROG.

6.4.1 Especificación de la instancia del problema a utilizar en la evaluación.

Para realizar las pruebas de evaluación de NEUROPROG, se ha utilizado una instancia del problema de la programación de producción discreta sobre una planta industrial formada por 7 máquinas destinadas a la fabricación de 3 artículos diferentes, con una carga de trabajo de 40 pedidos. Los costes mínimo y máximo asociados a una máquina-proceso son 2 y 91, las duraciones mínima y máxima correspondientes son 1 y 10, y las cantidades mínima y máxima de artículos en un pedido son 1 y 249 respectivamente. Las matrices de costes y duraciones de máquinas-procesos de la planta, así como la lista de los 40 pedidos a programar se incluyen en el apéndice IV de la tesis (tablas IV.6, IV.7, y

IV.8). La solución óptima que minimiza la función objetivo del problema es 4915.9 (figura 6.5):

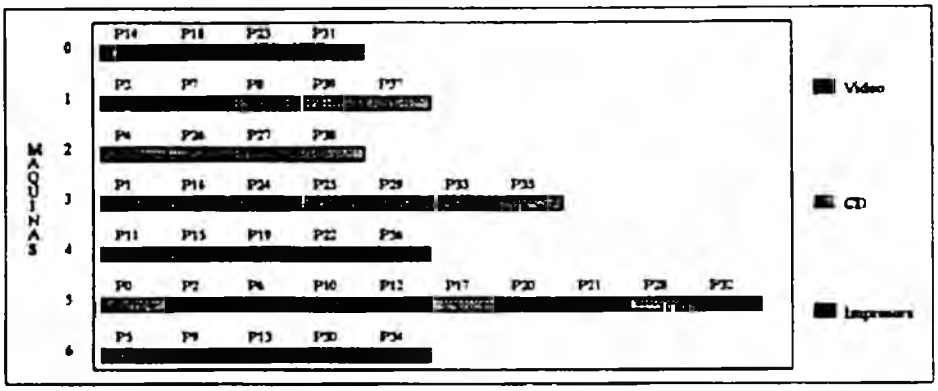

Figura 6.5: Solución óptima para la programación de producción discreta 40: 4915.9.

6.4.2 Análisis de las pruebas realizadas.

Para analizar el comportamiento del sistema NEUROPROG, se ha implementado el problema de la programación de producción discreta con las mismas técnicas utilizadas en el análisis del modelo ODE, esto es, búsqueda local, temple simulado, red de Hopfield continua y máquina de Boltzmann. A continuación, se sintetizan y estudian los resultados de las pruebas realizadas (20 por cada método), las cuales se exponen detalladamente en el apéndice IV de la tesis.

Las tabla 6.1 contienen los resultados correspondientes a niveles de optimización de las pruebas del problema de la programación de producción discreta sobre una carga de trabajo de 40 pedidos, realizadas mediante los cinco métodos de comparación: búsqueda local (BL), temple simulado (TS), red de Hopfield continua (CH), máquina de Boltzmann (BM), y optimizador discreto estocástico (ODE). Estos resultados incluyen el coste medio, el coste mejor y el coste peor, la desviación típica σ_c, el índice de coste medio I_{opt}, y el índice de variación del coste I_{var}:

Prod. 40	C. Medio	C. Mejor	C. Peor	σ_c	I_{opt} (%)	I_{var} (%)
BL	4949.1	4929.5	4974.7	11.8	100.68	0.24
TS	4922	4915.9	4929.4	4.2	100.12	0.09
CH	6110.8	5637.9	6687	300.5	124.31	4.92
BM	6578.5	6089.1	7228.9	320.7	133.82	4.87
ODE	4923.1	4917.2	4932.2	3.8	100.15	0.08

Tabla 6.1: Niveles de optimización en el problema de la programación de producción 40.

En la figura 6.6 se presenta un diagrama de columnas del índice de coste medio obtenido para el problema de la programación de producción con las diferentes técnicas de resolución. Este índice refleja el nivel medio de optimización alcanzado por cada método con respecto a la solución óptima. Hay que destacar el buen nivel de optimización alcanzado por el temple simulado y la red ODE, con un porcentaje medio de error con respecto al óptimo del 0.12 % y el 0.15 % respectivamente. El método basado en búsqueda local proporciona un error del 0.68 %, mientras que los otros métodos conexionistas basados en las redes CH y BM quedan muy lejos de los niveles de optimización anteriores, con errores del 24.3 % para la primera, y del 33.8 % para la segunda.

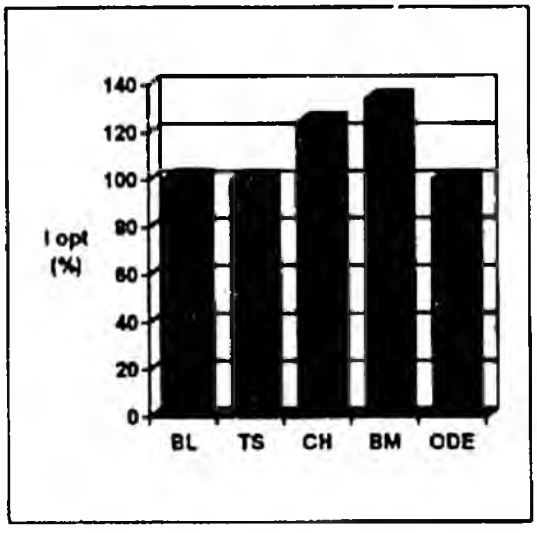

Figura 6.6: Índice de coste medio para el problema de programación de producción 40.

En la figura 6.7 se presenta un diagrama de columnas del índice de variación del coste obtenido para el problema de la programación de producción con las diferentes técnicas de resolución. Este índice refleja para cada método la dispersión con respecto a la media de las soluciones obtenidas. Hay que destacar el bajo nivel de variación alcanzado por la red ODE y el temple simulado, con un porcentaje del 0.08 % y 0.09 % respectivamente. El método basado en búsqueda local proporciona una variación del 0.24 %, mientras que los otros métodos conexionistas basados en las redes CH y BM poseen índices de variación del 4.92 % y 4.87 % respectivamente.

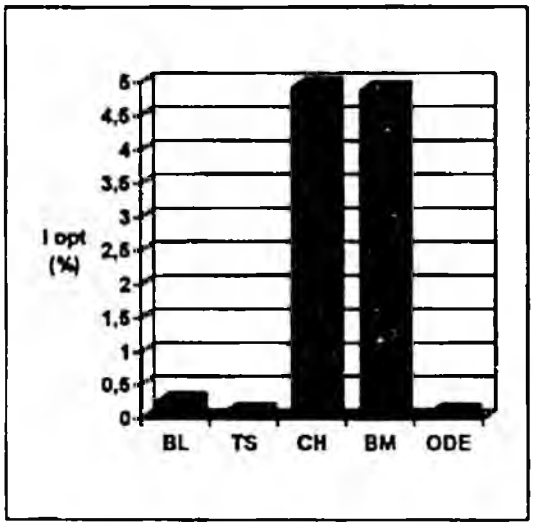

Figura 6.7: Índice de variación del coste para el problema de la programación de producción 40.

La tabla 6.2 contiene los resultados correspondientes a tiempos de ejecución de las pruebas del problema de la programación de producción discreta sobre una carga de trabajo de 40 pedidos, realizadas mediante los cinco métodos de comparación: búsqueda local (BL), temple simulado (TS), red de Hopfield continua (CH), máquina de Boltzmann (BM), y optimizador discreto estocástico (ODE). Estos resultados incluyen el tiempo medio, el tiempo mejor y el tiempo peor, la desviación típica σ_t, el índice de tiempo medio T_{opt}, y el índice de variación del tiempo T_{var}:

Prod. 40	T. Medio	T. Mejor	T. Peor	σ_t	T_{opt}	T_{var} (%)
BL	5.9	4	7	0.8	1	13.56
TS	38.2	33	44	2.8	6.47	7.33
CH	1658.8	1400	1953	167.2	281.15	10.08
BM	147.4	141	154	3.3	24.98	2.24
ODE	6.6	6	8	0.7	1.12	10.61

Tabla 6.2: Tiempos de ejecución del problema de la programación de producción 40.

En la figura 6.8 se presenta un diagrama de columnas del índice de tiempo medio de ejecución obtenido para el problema de la programación de producción con las diferentes técnicas de resolución. Este índice refleja el tiempo medio de ejecución empleado por cada método con respecto al de búsqueda local, que sirve como referencia. El algoritmo que resulta más rápido después del de búsqueda local es el correspondiente a la red ODE, con un índice de 1.12. A continuación, el algoritmo del temple simulado con un índice de 6.47, y la máquina de Boltzmann con un índice de 24.98. Finalmente, el peor comportamiento corresponde a la red de Hopfield continua, que presenta un índice temporal de 281.15.

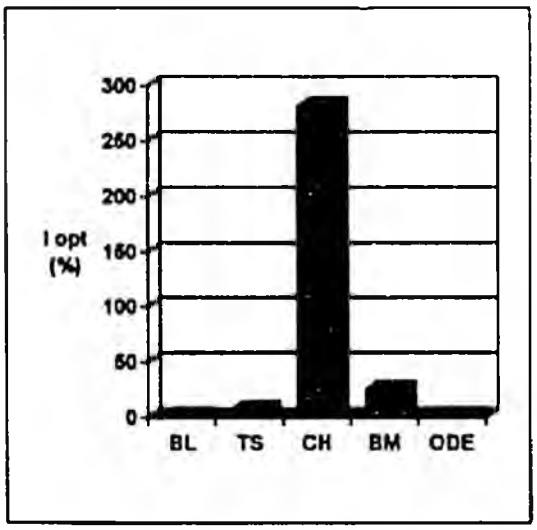

Figura 6.8: Índice de tiempo medio para el problema de programación de producción 40.

En la figura 6.9 se presenta un diagrama de columnas del índice de variación temporal obtenido para el problema de la programación de producción con las diferentes técnicas

de resolución. Este índice refleja para cada método la dispersión con respecto a la media de los tiempos obtenidos. El menor nivel de variación corresponde a la máquina de Boltzmann, con un porcentaje medio del 2.24 %. Superiores al anterior, pero próximos entre sí, son los porcentajes de variación temporal del temple simulado (7.33 %), la red CH (10.08 %), y la red ODE (10.61 %). En este caso, el peor comportamiento corresponde al método de búsqueda local, con un porcentaje de variación del 13.56 %.

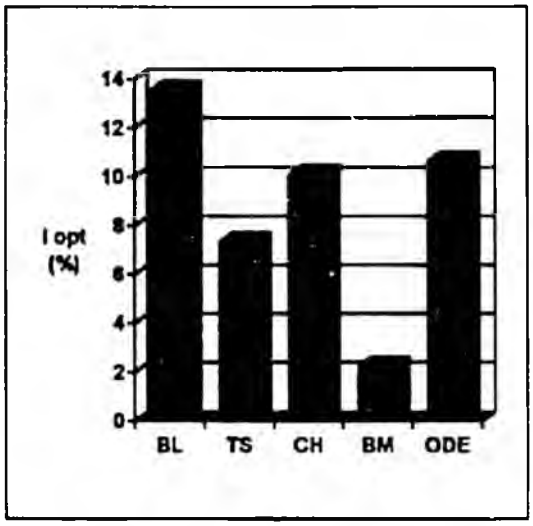

Figura 6.9: Índice de variación temporal para el problema de la programación de producción 40.

El estudio comparativo de la eficiencia del optimizador discreto estocástico en el problema de la programación de producción discreta, frente a las distintas técnicas de resolución de problemas de optimización combinatoria, produce iguales resultados a los ya observados: la mejor actuación corresponde a los métodos basados en el algoritmo del temple simulado y en la red neuronal ODE, incluso con un notable mejor rendimiento temporal de esta última, debido principalmente a las características del problema (poco restringido), y al uso de una estructura de soluciones 1-próximas en la resolución.

A modo de resumen, la figura 6.10 muestra un diagrama comparativo de la eficiencia, en cuanto a la calidad de las soluciones en el problema de la programación de producción discreta, de los diferentes métodos de resolución con respecto a la red ODE. El diagrama representa el cociente entre el error del coste medio respecto al óptimo de cada técnica empleada, y el error del coste medio obtenido con el optimizador discreto estocástico ($\varepsilon/\varepsilon_{ODE}$). Los datos correspondientes se recogen en la tabla 6.3.

$\varepsilon/\varepsilon_{ODE}$	**Producción**
BL	4.61
TS	0.85
CH	165.96
BM	230.92
ODE	1

Tabla 6.3: Errores de coste medio con respecto al error de la red ODE ($\varepsilon/\varepsilon_{ODE}$).

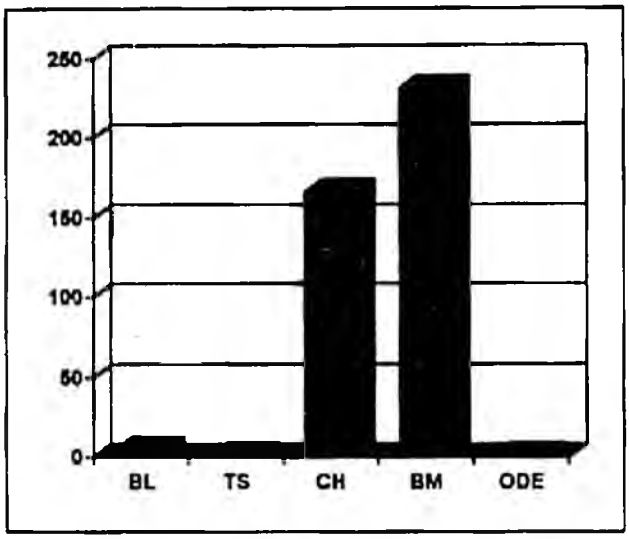

Figura 6.10: Comparativa global de la eficiencia en optimización de la red ODE ($\varepsilon/\varepsilon_{ODE}$).

7

Conclusiones y Líneas Futuras

Es indigno del hombre no seguir buscando
un saber al que es posible aspirar.

Aristóteles

7.1 CONCLUSIONES SOBRE EL TRABAJO REALIZADO.

El objetivo principal que motivó esta tesis doctoral fue el desarrollo de nuevas técnicas de optimización combinatoria basadas en modelos neuronales, que fueran capaces de resolver problemas prácticos de dimensión real, obteniendo soluciones de alta calidad en tiempos de ejecución factibles.

Con este fin, se ha realizado un trabajo de investigación que ha obtenido los siguientes resultados:

- Se ha desarrollado *un nuevo modelo conexionista, el optimizador discreto estocástico, ODE, orientado hacia la resolución aproximada de problemas de optimización combinatoria.* El optimizador discreto estocástico presenta una arquitectura y un modo de procesamiento recurrentes que intentan combinar las ventajas de los modelos conexionistas clásicos utilizados en la resolución de problemas de optimización combinatoria, superando sus limitaciones individuales. Por una parte, la red ODE utiliza un criterio estocástico de transición de estados, similar al de la máquina de Boltzmann [HINT86], que permite superar el comportamiento local de las técnicas deterministas de optimización. Por otra parte, el procesamiento recurrente específico del modelo ODE evita la limitación intrínseca del procesamiento de la máquina de Boltzmann, caracterizado por la actualización de un solo nodo cada vez, con lo que la búsqueda se realiza sobre un espacio de soluciones mayor, y permite alcanzar un alto nivel de calidad en la optimización.

- Se ha definido *un método general de representación de problemas de optimización combinatoria sobre redes neuronales recurrentes*, que permite determinar las características de la red (arquitectura, función de energía y pesos de las conexiones) que mejor representa el problema tratado. Éste era uno de los mayores inconvenientes de la implementación de problemas de optimización combinatoria sobre redes recurrentes, y, generalmente, se realizaba de forma empírica para cada instancia de un determinado problema. Tras un estudio del comportamiento dinámico de los diferentes modelos conexionistas aplicados a problemas de optimización, se ha determinado un procedimiento de diseño para la determinación exacta de las características de la red, ante cualquier instancia de un problema de optimización combinatoria.

- Se ha realizado *un análisis comparativo de la eficiencia de los diferentes métodos de optimización combinatoria actuales con respecto al modelo ODE.* En la evaluación del modelo propuesto se han utilizado como técnicas de comparación los dos algoritmos heurísticos generales de optimización más importantes, la búsqueda local y el temple simulado, y los dos modelos conexionistas clásicos utilizados en este tipo de problemas, la red de Hopfield continua y la máquina de Boltzmann. Cada uno de los cinco métodos se ha evaluado estadísticamente mediante la ejecución de tres problemas clásicos, el problema del viajante sobre un recorrido de 15/20 ciudades, el problema de la partición de grafos sobre un grafo plano de 50 vértices que debe dividirse en 2/5 particiones, y el problema de la asignación cuadrática sobre un total de 15/20 plantas de producción a ubicar en 15/20 localizaciones distintas.

- Finalmente, se ha desarrollado *una aplicación práctica sobre una importante área de gran interés en la industria: NEUROPROG, un programador predictivo para sistemas de producción discreta.* La programación de tareas en producción es un problema de asignación de recursos que ha generado un gran interés en el mundo empresarial e industrial por diferentes razones, entre las que destaca por una parte la necesidad de mejorar la competitividad, y por otra el desarrollo de computadores cada vez más potentes que permiten mejorar las prestaciones de los actuales sistemas de programación de producción. En este trabajo se ha intentado demostrar la adecuación de la red neuronal ODE para la resolución de problemas de asignación de recursos, construyendo un sistema de programación de producción discreta, basado en el optimizador discreto estocástico, que implementa un modelo simplificado de planta industrial.

El trabajo presentado en esta memoria de tesis ha mostrado diversos elementos originales en el campo de las redes neuronales artificiales aplicadas a la resolución aproximada de problemas de optimización combinatoria. Las principales conclusiones que pueden obtenerse del mismo son las siguientes:

- *Las redes neuronales artificiales constituyen una técnica básica dentro del campo de la inteligencia Artificial, que resulta apropiada para la resolución de problemas de optimización combinatoria.* Los primeros modelos conexionistas aparecidos en los años ochenta que fueron aplicados a la optimización combinatoria, la red de Hopfield continua y la máquina de Boltzmann, así como sus variantes, la máquina de Cauchy, la máquina de Gauss y la red de temple determinista, han resultado poco útiles en la práctica, debido a las limitaciones intrínsecas a su modo de procesamiento local sobre una arquitectura simétrica. Por ello, tras el gran auge en la investigación sobre redes neuronales y optimización habido durante la segunda mitad de los ochenta y comienzos de la década actual, el número de estudios sobre este área ha disminuido considerablemente en los últimos tres años, llegándose incluso a cuestionar la aplicabilidad de las redes neuronales a este tipo de problemas [GEE95]. Sin embargo, la limitación de los modelos conexionistas clásicos de optimización combinatoria, en cierta forma análoga a la limitación de las primeras redes neuronales adaptativas con respecto al aprendizaje de problemas no lineales [MINS69], puede superarse utilizando una arquitectura diferente no simétrica. Así, el optimizador discreto estocástico es una red neuronal recurrente con conexiones de segundo orden, capaz de implementar una búsqueda por un espacio de estados similar al de cualquier algoritmo heurístico convencional, igualando los mejores niveles de optimización posibles con este tipo de técnicas.

- *La implementación en una red neuronal artificial de problemas combinatorios, tanto de optimización como de satisfacción de restricciones, puede realizarse con total exactitud mediante un procedimiento de diseño*, es decir, sin necesidad de un cálculo aproximado de parámetros vía ensayo-error. Este procedimiento consta de tres fases o tareas principales: la identificación de la arquitectura de la red con las variables y soluciones del problema, la transformación de la función objetivo y las restrieciones del problema en una función de energía apropiada para el modelo neuronal utilizado, y la derivación de los pesos de las conexiones a partir de dicha función de energía. Así mismo, con el fin de clarificar la descripción del proceso, cada una de estas tareas se ha concretado en los ejemplos del problema del viajante comercial, el problema de la partición de

grafos, y el problema de la asignación cuadrática. La definición de este procedimiento de diseño, descrito detalladamente en el capítulo 5 de esta memoria, es uno de los objetivos iniciales del proyecto de tesis que se ha logrado con mejores resultados.

- En el análisis y evaluación de la eficiencia de las distintas técnicas de resolución de problemas de optimización combinatoria, *la mejor actuación corresponde a los métodos basados en el algoritmo del temple simulado y en la red neuronal ODE*. Puede decirse, por tanto, que el optimizador discreto estocástico es una red neuronal cuya aplicación a problemas de optimización combinatoria es tan satisfactoria como la del temple simulado, el cual se considera el mejor algoritmo general de optimización aproximada conocido.
- En general, el método basado en la red ODE precisa para la resolución del problema un tiempo medio de ejecución secuencial superior al de los métodos heurísticos. No obstante, *el tiempo de ejecución puede reducirse notablemente, si se realiza una implementación masivamente paralela de la red ODE*, la cual es directamente posible debido al carácter síncrono de su modo de procesamiento.
- Finalmente, el buen comportamiento presentado por el sistema NEUROPROG en la implementación del problema de la programación de producción discreta, tanto en la calidad de las soluciones obtenidas como en el tiempo medio de resolución, ha demostrado la adecuación de la red neuronal ODE para la resolución de problemas de asignación de recursos de interés práctico.

7.2 LÍNEAS FUTURAS DE INVESTIGACIÓN.

Las líneas futuras de investigación que pueden desarrollarse como extensión del trabajo realizado resultan en su mayor parte evidentes a partir de la lectura de esta memoria. Algunas de ellas son el fundamento de nuevos proyectos de investigación que acaban de comenzar.

Las líneas de investigación básica se centran en los siguientes aspectos:

- *Determinar y analizar heurísticas generales* que permitan reducir el tiempo de ejecución secuencial del método de optimización basado en la red ODE. Estas

heurísticas se refieren fundamentalmente a la elección de los parámetros del programa de templado de la red: temperatura inicial, función de decremento de la temperatura y número de transiciones de estado para cada valor de la temperatura.

- *Estudiar las propiedades de optimización de la red ODE, utilizando diferentes distribuciones de probabilidad* asociadas al criterio estocástico de transición de estados. Diferentes investigadores (Szu, Takefuji) han propuesto criterios de transición de estados que utilizan distribuciones de probabilidad diferentes a la distribución de Boltzmann, tales como la distribución de Cauchy [SZU87] [TAKE89], argumentando que ésta permite un mejor comportamiento en tiempo de ejecución del sistema debido a que precisa un programa de templado más rápido que el de la primera. Por tanto, puede ser interesante ampliar el estudio realizado sobre la red ODE, evaluando comparativamente la eficiencia de la misma utilizando otras distribuciones de probabilidad (distribución de Cauchy, distribución de Gauss, etc.).

Por otra parte, las líneas de investigación aplicada a desarrollar son:

- *Analizar el comportamiento de la red ODE en la implementación de problemas de satisfacción de restricciones*, en relación con los métodos heurísticos clásicos, así como con otros métodos conexionistas. Una vez definido el procedimiento de diseño que permite implementar de forma correcta problemas combinatorios con restricciones, es posible intentar con éxito la resolución mediante redes neuronales recurrentes de otro de los problemas característicos de la matemática discreta: la satisfacción de restricciones. La aplicación de los modelos conexionistas a este tipo de problemas es similar a la de los problemas de optimización combinatoria, es decir, partiendo de un patrón de entrada que representa una solución aleatoria, y a través del proceso de minimización de energía que caracteriza a estos modelos, la red se estabiliza en un estado correspondiente a la solución del problema que satisface todas las restricciones impuestas. La ventaja de la aproximación conexionista es que, fijando los pesos de las conexiones de acuerdo con los criterios del procedimiento de representación propuesto en la tesis, y sin necesidad de un criterio estocástico de transición de estados (no se trata de un problema de optimización), el procesamiento de la red conduce siempre a un estado estable correspondiente a una solución que satisface las restricciones del problema, si es que tal solución existe.

- *Desarrollar un sistema de programación de producción discreta, evolucionado a partir de NEUROPROG, en el que se incorporen todas las características de una planta industrial real.* Es decir, un sistema que, además de los elementos, restricciones y objetivos contemplados en NEUROPROG, incorpore planes de fabricación de los artículos formados por varios procesos, fechas de lanzamiento y entrega para cada pedido, preferencias entre máquinas y entre pedidos, programación de operaciones de mantenimiento para las máquinas, tratamiento de imprevistos (programación reactiva), etc.

- Finalmente, *estudiar la utilidad de la red ODE en otros problemas de asignación de recursos*, como la planificación de turnos de trabajo, el problema del transporte, etc.

APÉNDICE I

Pruebas Problema del Viajante

Este apéndice contiene los resultados completos de las pruebas del problema del viajante, desarrolladas con el fin de realizar el análisis práctico de la eficiencia del modelo neuronal ODE, respecto a los métodos actuales de optimización combinatoria más importantes: búsqueda local, temple simulado, red de Hopfield continua y máquina de Boltzmann.

Para efectuar las pruebas, se han realizado en lenguaje C las diferentes implementaciones del problema del viajante con cada uno de los métodos de comparación, utilizando en su desarrollo una metodología de diseño y programación modular basada en máquinas abstractas. Estas pruebas se han realizado sobre dos instancias del problema, correspondientes a rutas de 15 y 20 ciudades respectivamente, ejecutando 20 veces cada instancia con cada método de resolución.

Los resultados se presentan agrupados en cinco apartados, correspondientes a cada una de las técnicas de resolución empleadas. Así mismo, al final del apéndice se proporcionan las matrices de distancias correspondientes a las dos instancias del problema del viajante que se han utilizado en las pruebas.

I.1 PRUEBAS PVC BÚSQUEDA LOCAL.

La tabla I.1 presenta los resultados de las pruebas, realizadas mediante el algoritmo de búsqueda local, del problema del viajante sobre una ruta de 15 ciudades españolas. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras I.1, I.2 y I.3):

I	COSTE	TIEMPO	SOLUCIÓN
1	4478	5	6 12 0 14 5 11 10 2 13 4 8 1 9 7 3
2	4490	7	6 12 0 14 8 4 11 10 2 3 7 9 1 13 5
3	4600	7	2 10 6 12 0 14 9 7 3 13 11 5 4 8 1
4	4362	6	11 10 2 13 4 1 9 7 3 8 14 0 12 6 5
5	4353	5	14 0 8 4 1 9 7 3 2 10 6 12 11 5 13
6	4676	6	2 10 6 12 0 14 7 3 13 11 5 4 8 1 9
7	3930	15	0 12 6 5 11 10 2 13 3 7 9 1 4 8 14
8	4678	7	7 3 2 10 11 14 4 9 1 8 0 12 6 5 13
9	4642	6	2 10 6 12 0 14 9 7 1 8 4 5 11 13 3
10	4255	9	12 6 10 2 13 3 7 9 1 5 11 4 8 14 0
11	4079	4	8 1 9 7 3 13 14 0 12 6 10 2 11 5 4
12	4079	9	13 14 0 12 6 10 2 11 5 4 8 1 9 7 3
13	4757	5	14 9 7 3 2 10 11 13 1 8 4 5 6 12 0
14	4834	3	14 4 7 3 2 10 11 13 9 1 8 0 12 6 5
15	4547	6	9 7 3 5 11 6 12 0 14 10 2 13 4 8 1
16	4452	5	10 2 9 1 8 4 7 3 13 11 5 14 0 12 6
17	4673	10	8 1 9 13 2 10 11 5 14 0 12 6 3 7 4
18	4639	5	5 6 12 0 8 1 9 7 3 2 10 11 13 4 14
19	3737	13	6 12 0 14 8 4 1 9 7 3 13 5 11 2 10
20	3737	10	12 6 10 2 11 5 13 3 7 9 1 4 8 14 0

Tabla I.1: Pruebas problema del viajante 15 con búsqueda local.

COSTE		TIEMPO	
Medio:	4399.9 km	Medio:	7.2 sg
Mejor:	3737 km	Mejor:	3 sg
Peor:	4834 km	Peor:	15 sg
Desviación:	330.0 km	Desviación:	3.0 sg

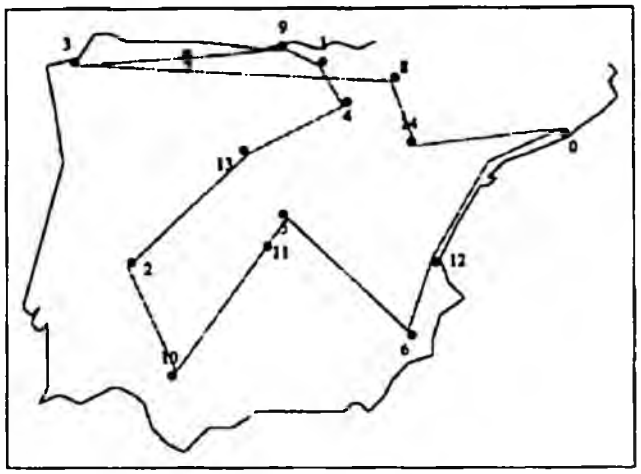

Figura I.1: Solución media al problema del viajante 15 con búsqueda local: 4362 km.

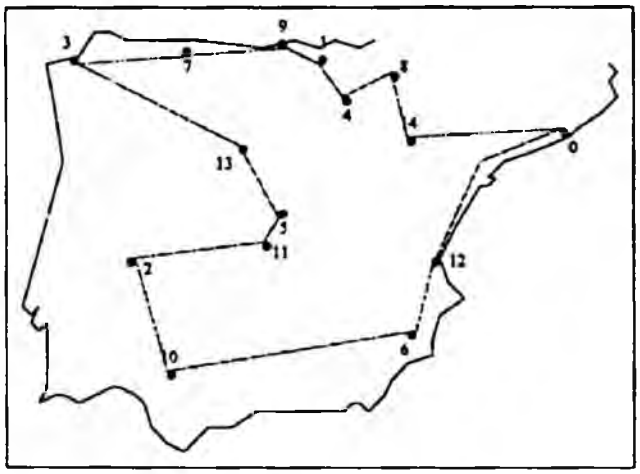

Figura I.2: Solución mejor al problema del viajante 15 con búsqueda local: 3737 km.

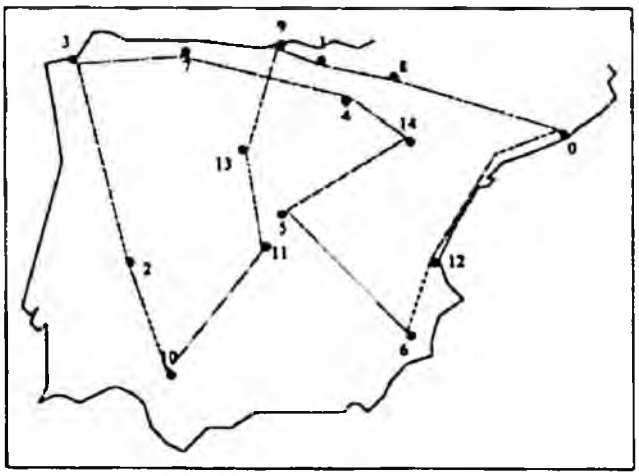

Figura I.3: Solución peor al problema del viajante 15 con búsqueda local: 4834 km.

La siguiente tabla presenta los resultados de las pruebas, realizadas mediante el algoritmo de búsqueda local, del problema del viajante sobre una ruta de 20 ciudades españolas. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras I.4, I.5 y I.6):

I	COSTE	TIEMPO	SOLUCIÓN
1	5098	20	11 7 5 12 14 1 17 0 19 13 8 2 18 9 16 4 3 15 10 6
2	4680	44	2 8 19 0 17 11 16 4 6 10 15 3 9 13 1 14 12 5 7 18
3	4187	46	14 1 8 13 19 0 17 11 6 10 15 3 4 16 9 2 18 7 5 12
4	5206	34	3 4 16 18 1 14 12 5 7 17 0 19 13 8 2 9 11 6 10 15
5	5506	34	0 13 1 14 12 5 7 9 16 4 10 6 19 8 2 18 3 15 11 17
6	5216	42	6 10 15 3 18 14 1 2 16 4 11 17 0 19 13 8 12 5 7 9
7	4957	55	17 0 13 1 14 12 5 7 2 8 19 11 6 10 15 3 18 9 16 4
8	5227	51	17 0 13 1 14 12 5 7 18 3 15 4 16 9 2 8 19 10 6 11
9	4540	36	2 8 13 19 0 17 11 9 16 4 6 10 15 3 18 1 14 12 5 7
10	4308	56	19 13 8 18 7 5 12 14 1 2 9 16 4 3 15 10 6 11 17 0
11	4759	56	6 10 15 3 16 17 0 19 13 8 7 5 12 14 1 2 18 9 4 11
12	4511	50	4 15 10 6 11 17 0 19 13 8 7 5 12 14 1 2 18 3 9 16
13	5258	56	12 7 19 17 0 13 8 1 14 2 18 9 16 3 15 10 6 11 4 5
14	5027	40	0 19 13 1 14 12 5 7 6 10 15 3 18 2 8 9 16 4 11 17
15	4609	29	1 14 12 5 7 3 15 10 6 11 17 4 16 9 18 2 8 19 0 13
16	4405	59	4 3 15 10 6 11 17 0 19 2 18 7 5 12 14 1 13 8 9 16
17	5004	51	6 11 0 19 13 8 1 14 12 5 7 2 17 4 16 9 18 3 15 10
18	5565	25	9 4 16 18 7 12 5 11 6 10 15 3 2 8 19 17 0 13 1 14
19	5140	38	15 11 17 0 19 13 8 18 7 5 12 14 1 2 10 6 4 16 9 3
20	4698	26	2 3 15 10 6 11 17 0 19 13 1 14 12 5 7 18 4 16 9 8

Tabla I.2: Pruebas problema del viajante 20 con búsqueda local.

COSTE		TIEMPO	
Medio:	4895.1 km	Medio:	42.4 sg
Mejor:	4187 km	Mejor:	20 sg
Peor:	5565 km	Peor:	59 sg
Desviación:	393.6 km	Desviación:	11.9 sg

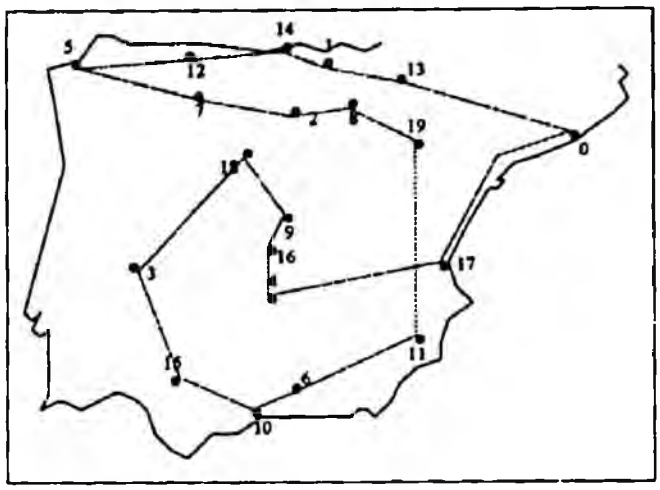

Figura I.4: Solución media al problema del viajante 20 con búsqueda local: 4957 km.

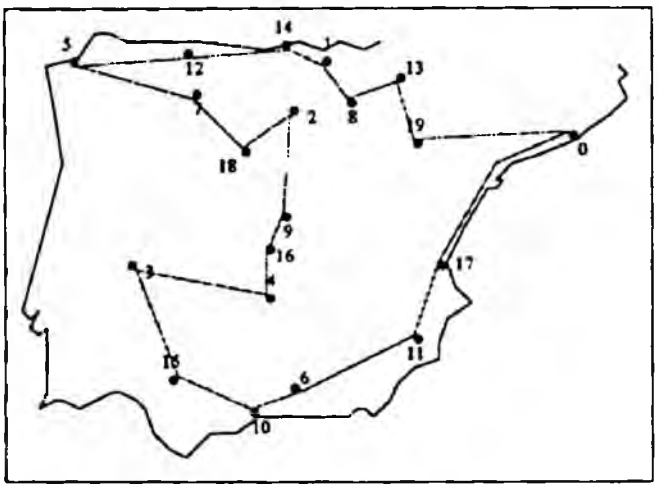

Figura I.5: Solución mejor al problema del viajante 20 con búsqueda local: 4187 km.

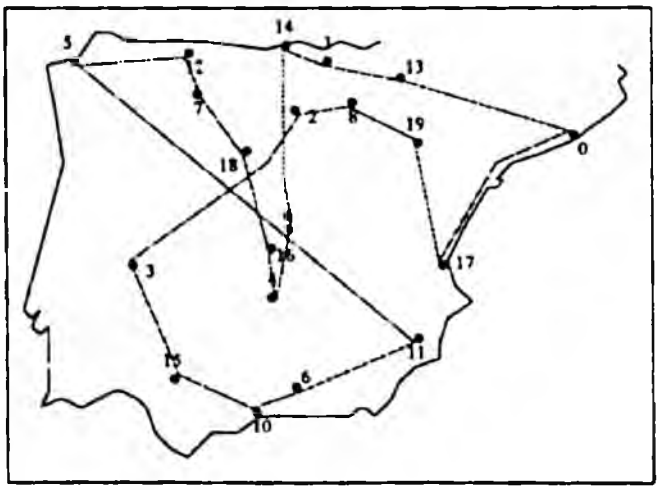

Figura I.6: Solución peor al problema del viajante 20 con búsqueda local: 5565 km.

I.2 PRUEBAS PVC TEMPLE SIMULADO.

La tabla I.3 presenta los resultados de las pruebas, realizadas mediante el algoritmo del temple simulado, del problema del viajante sobre una ruta de 15 ciudades españolas. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras I.7, I.8 y I.9):

I	COSTE	TIEMPO	SOLUCIÓN
1	3930	284	4 8 14 0 12 6 5 11 10 2 13 3 7 9 1
2	3930	303	10 2 13 3 7 9 1 4 8 14 0 12 6 5 11
3	3737	234	9 1 4 8 14 0 12 6 10 2 11 5 13 3 7
4	3737	219	14 8 4 1 9 7 3 13 5 11 2 10 6 12 0
5	3737	342	14 0 12 6 10 2 11 5 13 3 7 9 1 4 8
6	3737	238	5 13 3 7 9 1 4 8 14 0 12 6 10 2 11
7	3930	329	13 2 10 11 5 6 12 0 14 8 4 1 9 7 3
8	3930	327	14 0 12 6 5 11 10 2 13 3 7 9 1 4 8
9	3737	234	7 9 1 4 8 14 0 12 6 10 2 11 5 13 3
10	4253	232	4 0 14 5 11 12 6 10 2 13 3 7 9 1 8
11	3930	291	13 2 10 11 5 6 12 0 14 8 4 1 9 7 3
12	4314	262	4 13 5 11 12 6 10 2 3 7 9 1 8 0 14
13	3737	338	6 10 2 11 5 13 3 7 9 1 4 8 14 0 12
14	3737	310	10 6 12 0 14 8 4 1 9 7 3 13 5 11 2
15	3934	233	4 14 0 12 6 5 11 10 2 13 3 7 9 1 8
16	3737	307	12 6 10 2 11 5 13 3 7 9 1 4 8 14 0
17	3737	267	9 1 4 8 14 0 12 6 10 2 11 5 13 3 7
18	3737	308	12 6 10 2 11 5 13 3 7 9 1 4 8 14 0
19	3930	295	6 5 11 10 2 13 3 7 9 1 4 8 14 0 12
20	3737	259	8 14 0 12 6 10 2 11 5 13 3 7 9 1 4

Tabla I.3: Pruebas problema del viajante 15 con temple simulado.

COSTE		TIEMPO	
Medio:	3859.4 km	Medio:	280.6 sg
Mejor:	3737 km	Mejor:	219 sg
Peor:	4314 km	Peor:	342 sg
Desviación:	172.0 km	Desviación:	39.9 sg

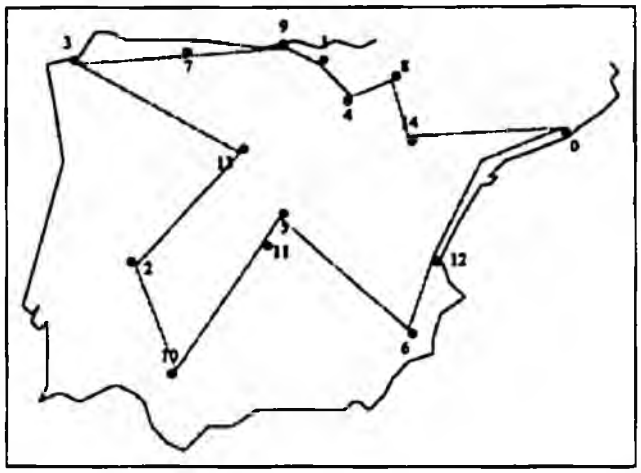

Figura I.7: Solución media al problema del viajante 15 con temple simulado: 3930 km.

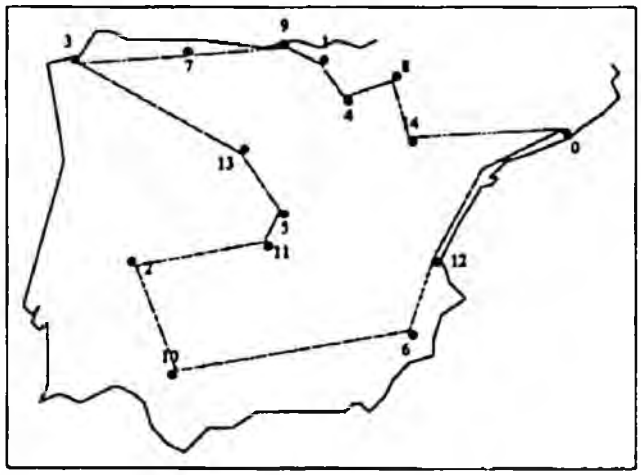

Figura I.8: Solución mejor al problema del viajante 15 con temple simulado: 3737 km.

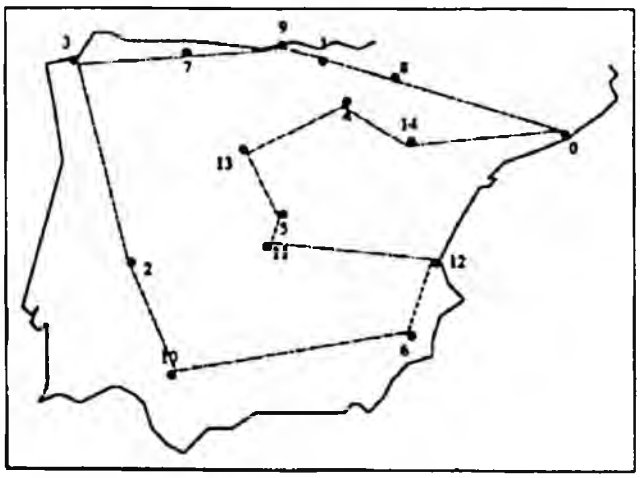

Figura I.9: Solución peor al problema del viajante 15 con temple simulado: 4314 km.

La siguiente tabla presenta los resultados de las pruebas, realizadas mediante el algoritmo del temple simulado, del problema del viajante sobre una ruta de 20 ciudades españolas. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras I.10, I.11 y I.12):

I	COSTE	TIEMPO	SOLUCIÓN
1	4142	1197	10 15 3 4 16 9 18 7 5 12 14 1 2 8 13 19 0 17 11 6
2	4670	1354	16 9 8 13 19 0 17 11 6 10 15 3 2 1 14 12 5 7 18 4
3	4548	1146	10 6 11 17 0 19 9 4 16 18 2 8 13 1 14 12 5 7 3 15
4	4142	1042	4 3 15 10 6 11 17 0 19 13 8 2 1 14 12 5 7 18 9 16
5	4351	1126	7 18 3 15 10 6 4 16 9 11 17 0 19 13 8 2 1 14 12 5
6	4673	1317	3 9 16 4 11 17 0 19 13 8 1 14 12 5 7 18 2 6 10 15
7	4187	1485	11 6 10 15 3 4 16 9 2 18 7 5 12 14 1 8 13 19 0 17
8	4142	1133	9 18 7 5 12 14 1 2 8 13 19 0 17 11 6 10 15 3 4 16
9	4331	1113	18 2 1 14 12 5 7 8 13 19 0 17 11 6 10 15 3 4 16 9
10	4666	1187	13 1 14 12 5 7 18 4 3 15 10 6 11 17 0 19 16 9 2 8
11	4881	1449	7 12 5 9 11 17 0 19 13 8 1 14 2 18 3 15 10 6 4 16
12	4734	1138	13 1 14 12 5 7 3 15 10 6 4 16 9 18 2 11 17 0 19 8
13	4600	1156	16 4 6 10 15 3 9 2 1 14 12 5 7 18 8 13 19 0 17 11
14	4142	1135	7 5 12 14 1 2 8 13 19 0 17 11 6 10 15 3 4 16 9 18
15	4142	1081	2 1 14 12 5 7 18 9 16 4 3 15 10 6 11 17 0 19 13 8
16	4653	1076	10 6 4 16 9 11 17 0 19 7 5 12 14 1 13 8 2 18 3 15
17	4440	1031	8 1 14 12 5 7 13 19 0 17 11 6 10 15 3 4 16 9 18 2
18	4593	1516	16 5 12 7 18 3 15 10 6 11 17 0 19 13 8 1 14 2 9 4
19	4191	1462	6 11 17 0 19 8 13 1 14 12 5 7 18 2 9 16 4 3 15 10
20	4142	1209	19 0 17 11 6 10 15 3 4 16 9 18 7 5 12 14 1 2 8 13

Tabla 1.4: Pruebas problema del viajante 20 con temple simulado.

COSTE		TIEMPO	
Medio:	4418.5 km	Medio:	1217.7 sg
Mejor:	4142 km	Mejor:	1031 sg
Peor:	4881 km	Peor:	1516 sg
Desviación:	252.8 km	Desviación:	155.0 sg

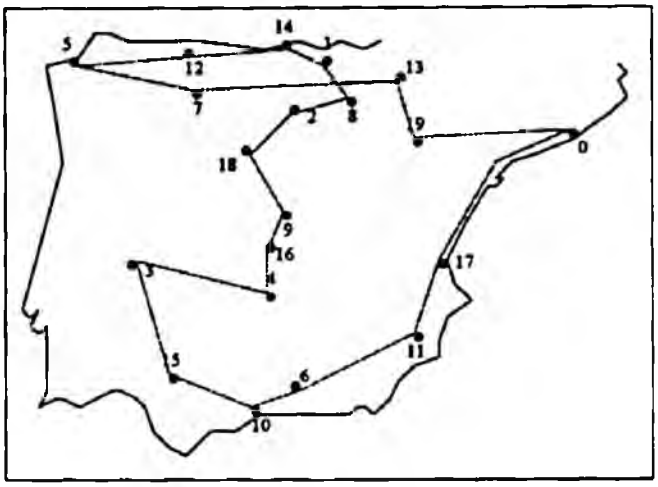

Figura I.10: Solución media al problema del viajante 20 con temple simulado: 4440 km.

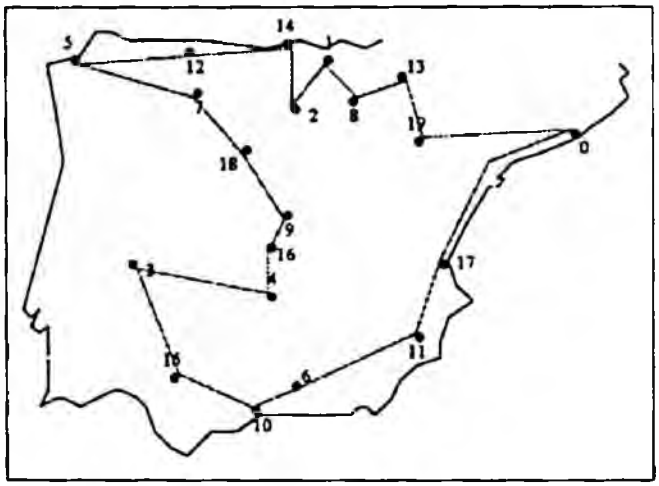

Figura I.11: Solución mejor al problema del viajante 20 con temple simulado: 4142 km.

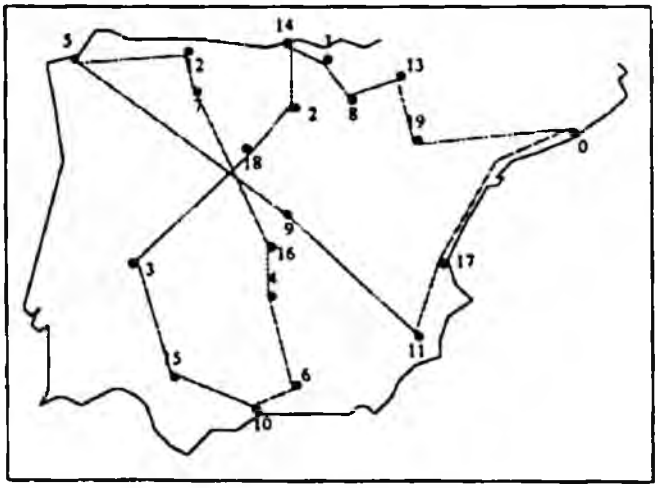

Figura I.12: Solución peor al problema del viajante 20 con temple simulado: 4881 km.

I.3 PRUEBAS PVC RED DE HOPFIELD CONTINUA.

La tabla I.5 presenta los resultados de las pruebas, realizadas mediante la red de Hopfield continua, del problema del viajante sobre una ruta de 15 ciudades españolas. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras I.13, I.14 y I.15):

I	COSTE	TIEMPO	SOLUCIÓN
1	6940	1251	11 4 7 8 0 6 5 2 10 12 13 1 14 3 9
2	7573	1257	12 6 14 4 0 3 8 11 1 2 10 13 9 5 7
3	7887	1381	8 7 9 13 2 14 3 0 12 1 5 6 10 4 11
4	5692	1564	4 14 0 5 2 10 11 8 12 6 7 3 9 1 13
5	5878	1153	8 4 1 9 12 13 6 5 11 2 10 7 3 0 14
6	6684	1152	6 14 4 13 11 8 1 0 12 9 10 2 5 7 3
7	7477	1387	3 8 5 1 11 7 2 12 0 6 10 9 13 4 14
8	5124	1085	5 3 7 13 1 9 4 8 12 6 14 0 10 2 11
9	6484	1383	8 1 4 14 12 0 5 13 9 11 2 10 3 6 7
10	7556	1135	1 0 2 4 11 9 8 7 10 6 12 13 3 5 14
11	6977	1023	11 13 1 7 3 2 10 8 9 12 4 5 14 6 0
12	7986	1493	7 8 12 1 0 2 4 11 6 14 10 5 3 13 9
13	6957	1276	4 11 6 12 3 10 5 0 2 13 1 8 14 9 7
14	6468	1300	10 11 7 14 9 4 8 1 0 6 12 3 13 2 5
15	7034	1079	8 9 10 12 3 7 2 6 5 13 11 1 0 4 14
16	7381	1186	12 5 11 3 9 14 2 7 1 8 0 6 4 10 13
17	6283	1619	4 8 5 9 14 13 6 12 0 3 7 11 10 2 1
18	6289	1391	3 11 5 0 4 7 9 14 1 8 13 12 6 2 10
19	6613	1372	13 9 1 6 4 0 8 14 2 11 12 5 10 3 7
20	6046	1371	10 3 13 4 9 7 1 14 8 0 11 6 5 12 2

Tabla I.5: Pruebas problema del viajante 15 con la red de Hopfield continua.

COSTE		TIEMPO	
Medio:	6766.5 km	Medio:	1292.9 sg
Mejor:	5124 km	Mejor:	1023 sg
Peor:	7986 km	Peor:	1619 sg
Desviación:	757.5 km	Desviación:	163.4 sg

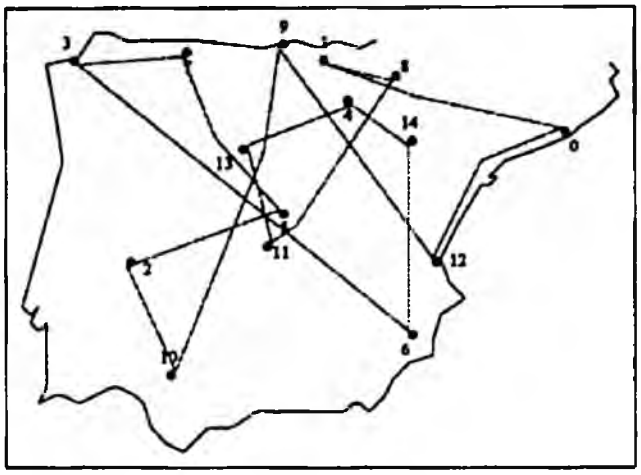

Figura I.13: Solución media al problema del viajante 15 con la red CH: 6684 km.

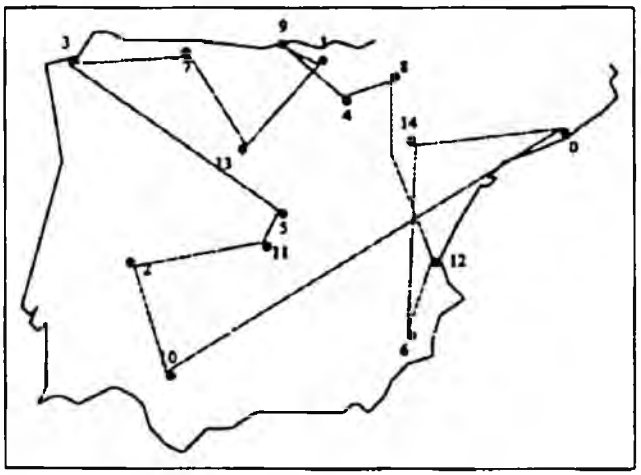

Figura I.14: Solución mejor al problema del viajante 15 con la red CH: 5124 km.

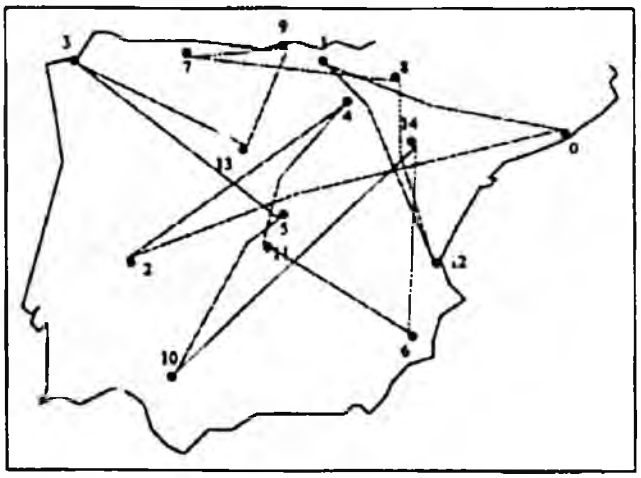

Figura I.15: Solución peor al problema del viajante 15 con la red CH: 7986 km.

La siguiente tabla presenta los resultados de las pruebas, realizadas mediante la red de Hopfield continua, del problema del viajante sobre una ruta de 20 ciudades españolas. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras I.16, I.17 y I.18):

I	COSTE	TIEMPO	SOLUCIÓN
1	7507	7202	0 6 11 4 16 19 2 15 10 3 7 18 1 14 13 5 12 8 9 17
2	6951	6567	1 2 8 5 7 12 13 0 3 15 6 10 19 16 9 11 17 4 18 14
3	8073	6000	17 11 0 9 19 2 12 1 18 8 16 4 15 10 5 14 13 7 3 6
4	8518	6389	6 15 4 9 14 11 3 0 2 13 8 5 12 1 16 19 7 18 17 10
5	9386	5470	14 3 6 18 5 17 7 0 19 13 8 2 12 11 16 15 4 9 10 1
6	10000	7212	14 5 18 6 8 3 16 0 11 10 9 4 13 15 19 17 2 7 1 12
7	7170	5504	6 15 17 3 9 16 7 5 12 14 1 19 0 10 4 2 8 13 18 11
8	9310	5658	13 10 11 4 15 9 3 19 16 0 6 5 7 12 14 17 1 18 8 2
9	8079	6296	15 11 9 8 12 18 3 17 7 5 13 14 2 1 19 16 4 0 6 10
10	10119	6893	19 2 8 7 3 1 13 11 15 9 16 12 18 0 17 14 10 4 5 6
11	6199	6715	5 9 16 18 14 8 17 11 6 10 15 3 4 0 2 13 19 1 12 7
12	7466	6991	12 1 14 8 3 15 6 11 10 16 5 0 13 19 18 2 4 17 9 7
13	8044	5369	7 2 17 0 9 12 3 14 18 4 15 11 6 10 16 5 19 8 1 13
14	9592	6996	7 17 10 13 18 0 3 12 2 9 5 1 19 8 14 16 4 6 11 15
15	7822	7254	16 11 10 4 15 3 5 6 17 1 14 19 12 7 0 8 13 2 18 9
16	8574	5133	15 4 19 1 7 2 17 11 16 0 12 5 18 10 6 3 9 14 8 13
17	8174	6099	1 2 8 19 13 5 12 7 18 11 16 6 0 9 17 14 10 15 3 4
18	8850	5328	2 1 3 9 18 7 12 6 19 13 14 8 16 11 17 5 4 10 15 0
19	6961	7867	6 11 10 17 0 14 8 1 12 5 7 3 16 4 13 2 19 18 9 15
20	6818	6638	17 11 6 10 15 0 8 14 13 19 16 2 1 12 7 5 9 3 4 18

Tabla I.6: Pruebas problema del viajante 20 con la red de Hopfield continua.

COSTE		TIEMPO	
Medio:	8180.7 km	Medio:	6379.1 sg
Mejor:	6199 km	Mejor:	5133 sg
Peor:	10119 km	Peor:	7867 sg
Desviación:	1110.8 km	Desviación:	781.5 sg

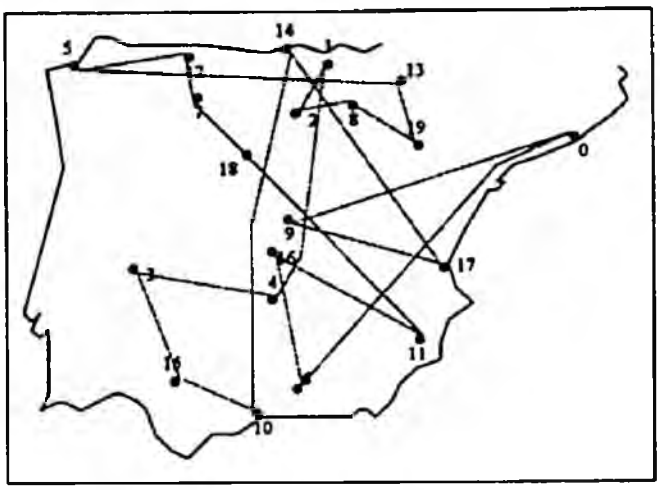

Figura I.16: Solución media al problema del viajante 20 con la red CH: 8174 km.

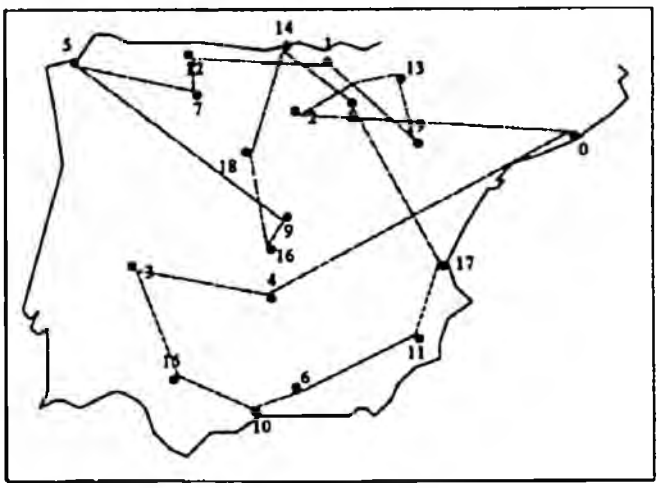

Figura I.17: Solución mejor al problema del viajante 20 con la red CH: 6199 km.

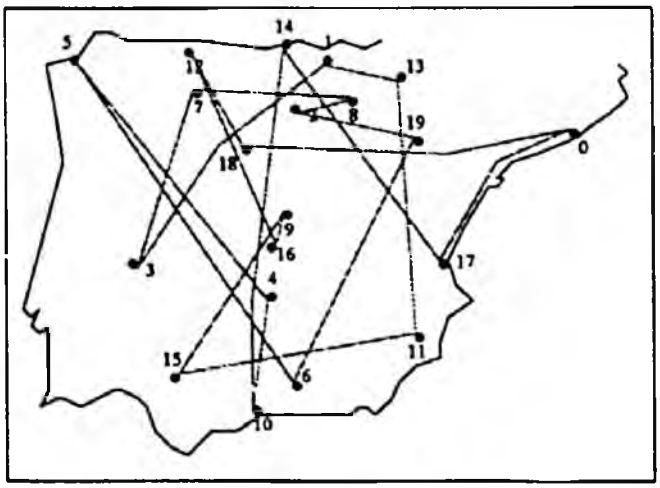

Figura I.18: Solución peor al problema del viajante 20 con la red CH: 10119 km.

I.4 PRUEBAS PVC MÁQUINA DE BOLTZMANN.

La tabla I.7 presenta los resultados de las pruebas, realizadas mediante la máquina de Boltzmann, del problema del viajante sobre una ruta de 15 ciudades españolas. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras I.19, I.20 y I.21):

I	COSTE	TIEMPO	SOLUCIÓN
1	7489	64	13 6 5 8 0 1 2 14 12 10 9 3 7 4 11
2	7776	77	11 4 14 7 8 12 10 0 3 13 5 1 9 2 6
3	7697	68	10 0 11 4 3 13 14 2 5 9 8 7 1 6 12
4	6861	69	10 12 8 1 14 0 3 5 7 13 2 9 4 11 6
5	7751	70	8 14 3 12 9 1 6 10 13 4 0 5 11 7 2
6	6953	66	4 3 2 10 11 9 0 1 8 7 5 6 14 12 13
7	7084	68	14 12 1 0 4 11 5 2 10 13 6 3 8 7 9
8	7767	69	6 7 3 2 1 5 9 0 13 11 4 14 12 8 10
9	7302	67	2 4 9 7 5 14 8 10 13 0 6 12 11 3 1
10	7421	68	12 5 10 9 14 1 8 4 3 7 0 11 13 6 2
11	5937	68	2 10 11 4 1 7 3 5 12 6 8 9 13 14 0
12	7628	72	5 9 1 13 4 7 12 14 0 11 10 8 3 6 2
13	6705	68	11 13 5 12 0 8 14 1 6 9 4 7 10 2 3
14	8077	69	6 13 14 8 7 12 1 3 5 2 9 11 4 0 10
15	6351	64	11 10 6 9 7 2 3 0 12 1 8 4 14 13 5
16	7034	67	10 4 14 11 13 8 1 5 0 12 3 9 7 6 2
17	6634	69	10 13 1 5 2 6 0 12 4 8 14 9 7 11 3
18	6861	64	3 4 7 13 1 9 11 2 6 10 14 8 5 12 0
19	7785	65	1 7 10 12 13 9 6 11 2 14 4 8 3 0 5
20	8075	66	11 7 1 12 4 14 6 5 13 3 10 0 9 2 8

Tabla I.7: Pruebas problema del viajante 15 con la máquina de Boltzmann.

COSTE		TIEMPO	
Medio:	7259.4 km	Medio:	67.9 sg
Mejor:	5937 km	Mejor:	64 sg
Peor:	8077 km	Peor:	77 sg
Desviación:	587.1 km	Desviación:	3.0 sg

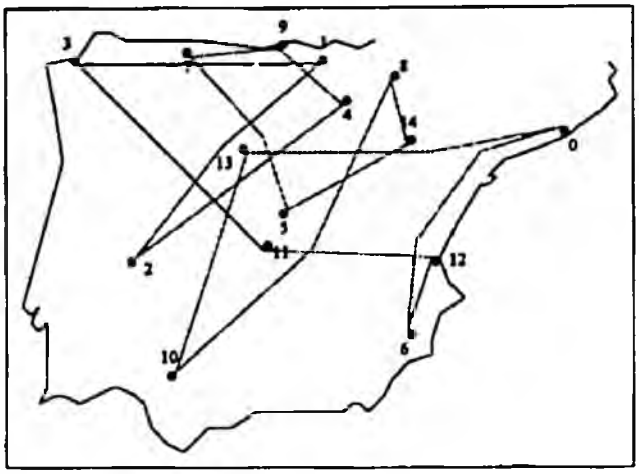

Figura I.19: Solución media al problema del viajante 15 con la red BM: 7302 km.

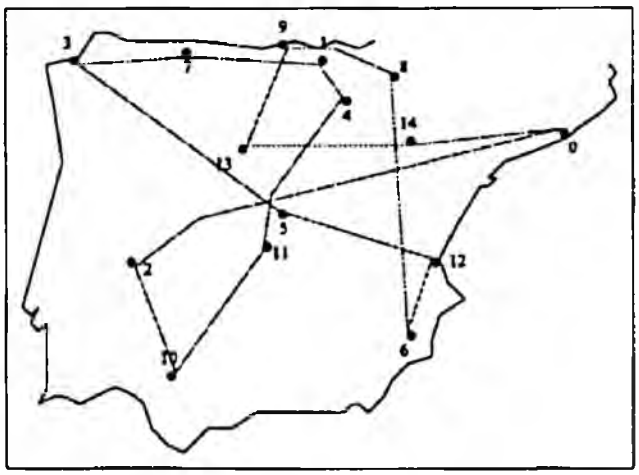

Figura I.20: Solución mejor al problema del viajante 15 con la red BM: 5937 km.

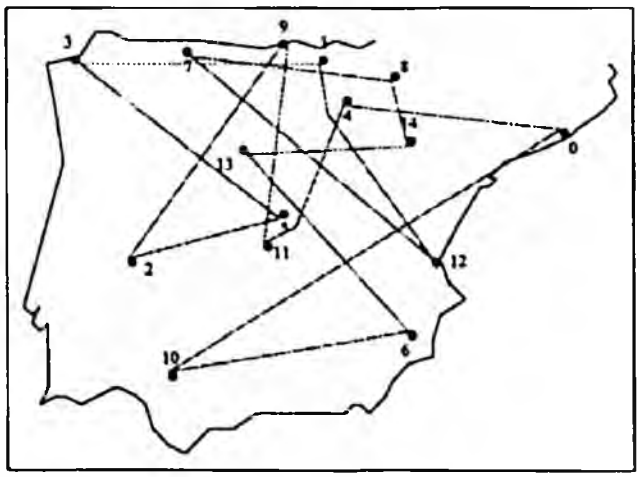

Figura I.21: Solución peor al problema del viajante 15 con la red BM: 8077 km.

La siguiente tabla presenta los resultados de las pruebas, realizadas mediante la máquina de Boltzmann, del problema del viajante sobre una ruta de 20 ciudades españolas. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras I.22, I.23 y I.24):

I	COSTE	TIEMPO	SOLUCIÓN
1	8046	246	15 10 11 4 5 7 13 14 12 16 2 0 19 18 3 6 17 1 8 9
2	9743	266	19 2 11 14 13 9 12 15 10 16 7 18 6 3 17 0 1 5 4 8
3	10403	261	18 15 5 13 14 7 2 16 0 19 1 3 6 17 8 11 9 4 12 10
4	8828	265	11 7 9 16 1 13 17 5 3 12 2 10 15 6 4 8 18 14 0 19
5	8945	283	3 7 8 2 18 5 6 17 0 1 14 9 16 11 19 10 4 15 12 13
6	10391	262	8 10 1 9 4 3 18 0 17 12 19 5 14 2 13 11 15 16 6 7
7	9404	266	14 17 9 19 11 4 15 0 1 18 3 5 7 16 8 6 10 2 12 13
8	8724	253	7 16 15 10 18 2 4 6 13 14 12 3 9 1 17 11 0 19 5 8
9	9786	260	14 18 17 3 1 10 9 0 13 12 5 15 11 16 19 8 6 4 2 7
10	8864	260	14 7 5 8 13 18 2 12 0 9 16 6 4 10 1 11 3 15 17 19
11	9036	267	8 1 14 16 17 6 0 2 19 4 13 18 3 10 9 15 11 7 5 12
12	10529	258	2 3 12 11 7 17 1 13 19 14 5 8 9 18 10 16 0 6 4 15
13	10075	253	2 15 10 19 6 3 16 12 11 8 14 0 13 7 9 17 4 1 5 18
14	7976	260	19 17 16 4 14 18 12 8 5 3 7 13 0 1 15 10 6 11 9 2
15	8617	262	8 3 10 15 11 17 4 13 7 2 18 6 9 16 12 1 19 14 5 0
16	11058	264	10 0 11 13 18 12 7 14 17 1 6 8 5 4 2 9 3 16 15 19
17	9864	270	1 10 3 14 12 9 19 7 18 16 4 0 5 15 8 13 2 6 11 17
18	8518	265	3 12 7 1 14 9 11 0 17 15 10 16 5 2 18 19 13 4 8 6
19	8434	257	7 5 8 10 15 3 11 17 6 4 12 19 9 18 0 16 1 13 2 14
20	9144	257	12 14 16 6 10 8 2 5 15 17 4 11 9 13 19 1 3 7 0 18

Tabla 1.8: Pruebas problema del viajante 20 con la máquina de Boltzmann.

COSTE		TIEMPO	
Medio:	9319.3 km	Medio:	261.8 sg
Mejor:	7976 km	Mejor:	246 sg
Peor:	11058 km	Peor:	283 sg
Desviación:	873.0 km	Desviación:	7.5 sg

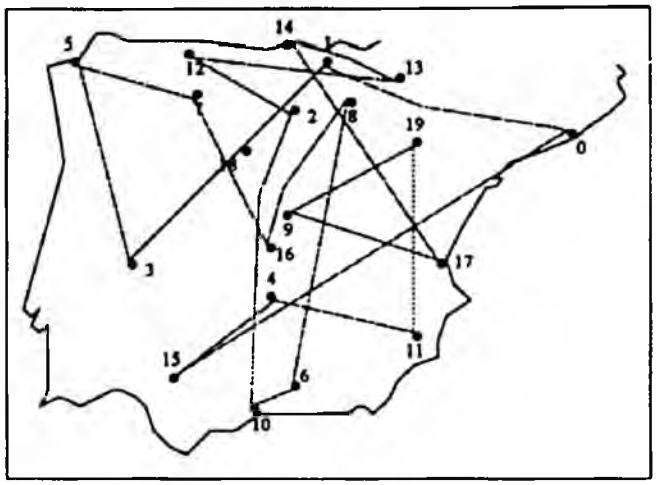

Figura I.22: Solución media al problema del viajante 20 con la red BM: 9404 km.

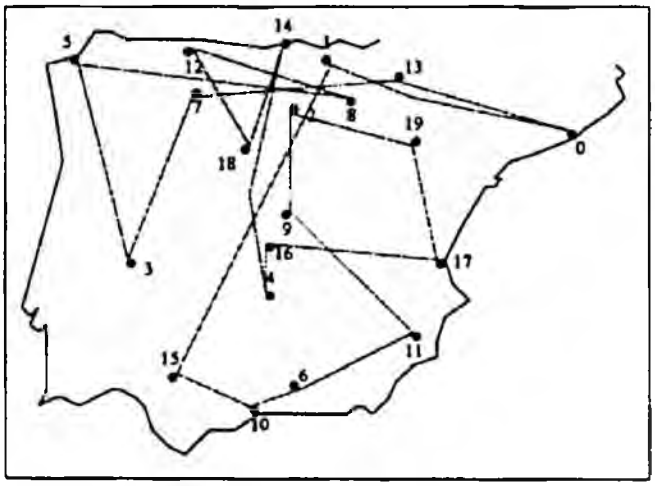

Figura I.23: Solución mejor al problema del viajante 20 con la red BM: 7976 km.

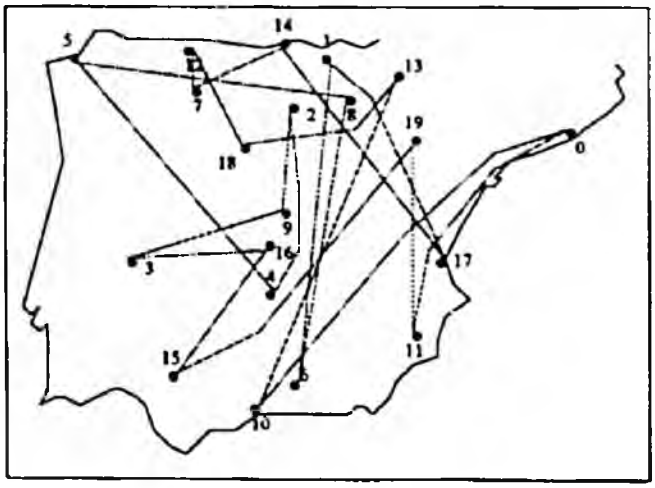

Figura I.24: Solución peor al problema del viajante 20 con la red BM: 11058 km.

I.5 PRUEBAS PVC OPTIMIZADOR DISCRETO ESTOCÁSTICO.

La tabla I.9 presenta los resultados de las pruebas, realizadas mediante el optimizador discreto estocástico, del problema del viajante sobre una ruta de 15 ciudades españolas. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras I.25, I.26 y I.27):

I	COSTE	TIEMPO	SOLUCIÓN
1	4509	353	0 14 8 4 5 11 10 2 13 3 7 9 1 6 12
2	3930	368	7 9 1 4 8 14 0 12 6 5 11 10 2 13 3
3	3741	366	11 5 13 3 7 9 1 8 4 14 0 12 6 10 2
4	4169	283	2 13 3 7 9 1 8 4 5 11 14 0 12 6 10
5	4204	476	7 3 2 10 6 12 0 14 8 4 1 9 5 11 13
6	3741	314	12 0 14 4 8 1 9 7 3 13 5 11 2 10 6
7	3737	438	6 10 2 11 5 13 3 7 9 1 4 8 14 0 12
8	3737	302	0 14 8 4 1 9 7 3 13 5 11 2 10 6 12
9	3737	353	2 11 5 13 3 7 9 1 4 8 14 0 12 6 10
10	3737	413	5 13 3 7 9 1 4 8 14 0 12 6 10 2 11
11	3741	353	14 4 8 1 9 7 3 13 5 11 2 10 6 12 0
12	3741	295	4 8 1 9 7 3 13 5 11 2 10 6 12 0 14
13	3737	296	14 8 4 1 9 7 3 13 5 11 2 10 6 12 0
14	4184	303	5 13 4 8 1 9 7 3 14 0 12 6 10 2 11
15	3930	394	13 2 10 11 5 6 12 0 14 8 4 1 9 7 3
16	3737	458	14 0 12 6 10 2 11 5 13 3 7 9 1 4 8
17	3737	461	9 7 3 13 5 11 2 10 6 12 0 14 8 4 1
18	3737	274	12 0 14 8 4 1 9 7 3 13 5 11 2 10 6
19	3737	350	10 6 12 0 14 8 4 1 9 7 3 13 5 11 2
20	3741	389	8 4 14 0 12 6 10 2 11 5 13 3 7 9 1

Tabla I.9: Pruebas problema del viajante 15 con el optimizador discreto estocástico.

COSTE		TIEMPO	
Medio:	3863.2 km	Medio:	362.0 sg
Mejor:	3737 km	Mejor:	274 sg
Peor:	4509 km	Peor:	476 sg
Desviación:	224.4 km	Desviación:	62.7 sg

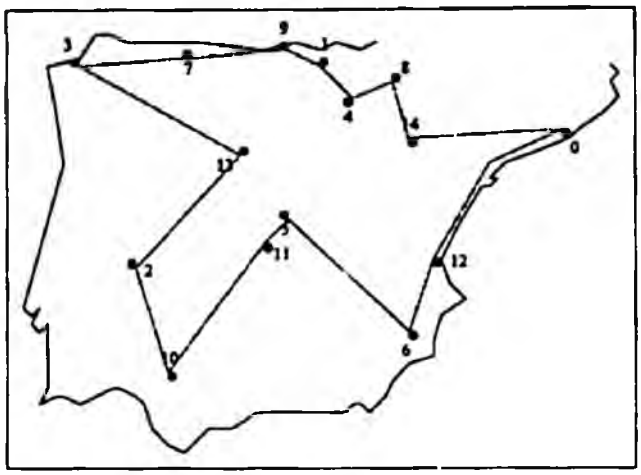

Figura I.25: Solución media al problema del viajante 15 con la red ODE: 3930 km.

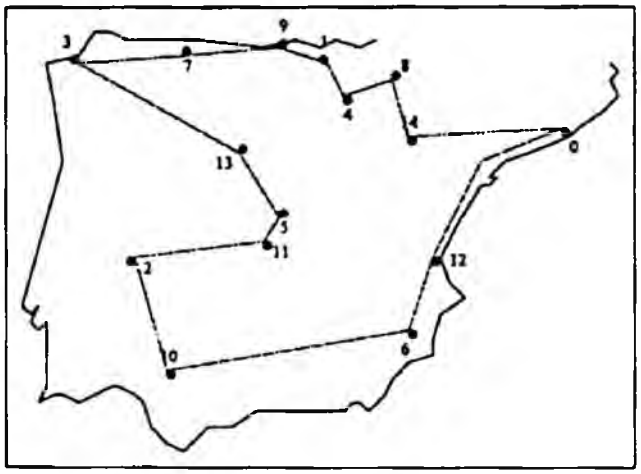

Figura I.26: Solución mejor al problema del viajante 15 con la red ODE: 3737 km.

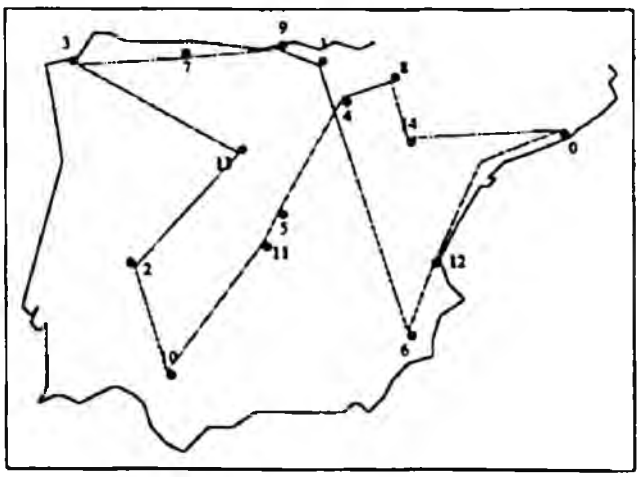

Figura I.27: Solución peor al problema del viajante 15 con la red ODE: 4569 km.

La siguiente tabla presenta los resultados de las pruebas, realizadas mediante el optimizador discreto estocástico, del problema del viajante sobre una ruta de 20 ciudades españolas. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras I.28, I.29 y I.30):

I	COSTE	TIEMPO	SOLUCIÓN
1	4142	1319	6 10 15 3 4 16 9 18 7 5 12 14 1 2 8 13 19 0 17 11
2	4142	1235	6 11 17 0 19 13 8 2 1 14 12 5 7 18 9 16 4 3 15 10
3	4443	1737	4 6 10 15 3 5 12 7 18 2 14 1 8 13 19 0 17 11 9 16
4	4142	1300	12 14 1 2 8 13 19 0 17 11 6 10 15 3 4 16 9 18 7 5
5	4540	1229	2 7 5 12 14 1 18 3 15 10 6 4 16 9 11 17 0 19 13 8
6	4540	1190	1 2 18 3 15 10 6 4 16 9 11 17 0 19 13 8 7 5 12 14
7	4142	1215	9 16 4 3 15 10 6 11 17 0 19 13 8 2 1 14 12 5 7 18
8	4142	1362	7 5 12 14 1 2 8 13 19 0 17 11 6 10 15 3 4 16 9 18
9	4434	1222	16 11 17 0 19 13 8 2 1 14 12 5 7 18 9 3 15 10 6 4
10	4534	1789	6 10 15 3 5 12 7 18 16 4 9 2 14 1 8 13 19 0 17 11
11	4418	1178	11 4 16 9 17 0 19 13 8 2 1 14 12 5 7 18 3 15 10 6
12	4191	1505	11 17 0 19 8 13 1 14 12 5 7 18 2 9 16 4 3 15 10 6
13	4331	1245	19 0 17 11 6 10 15 3 4 16 9 18 2 1 14 12 5 7 8 13
14	4142	1249	5 7 18 9 16 4 3 15 10 6 11 17 0 19 13 8 2 1 14 12
15	4142	1292	19 13 8 2 1 14 12 5 7 18 9 16 4 3 15 10 6 11 17 0
16	4472	1235	16 3 15 10 6 4 9 18 7 5 12 14 1 2 8 13 19 0 17 11
17	4811	1498	3 5 12 7 18 9 16 11 17 0 19 13 8 1 14 2 4 6 10 15
18	4142	1200	1 2 8 13 19 0 17 11 6 10 15 3 4 16 9 18 7 5 12 14
19	4142	1264	9 18 7 5 12 14 1 2 8 13 19 0 17 11 6 10 15 3 4 16
20	4540	1260	12 5 7 8 13 19 0 17 11 9 16 4 6 10 15 3 18 2 1 14

Tabla 1.10: Pruebas problema del viajante 20 con el optimizador discreto estocástico.

COSTE		TIEMPO	
Medio:	4326.6 km	Medio:	1326.2 sg
Mejor:	4142 km	Mejor:	1178 sg
Peor:	4811 km	Peor:	1789 sg
Desviación:	204.4 km	Desviación:	173.8 sg

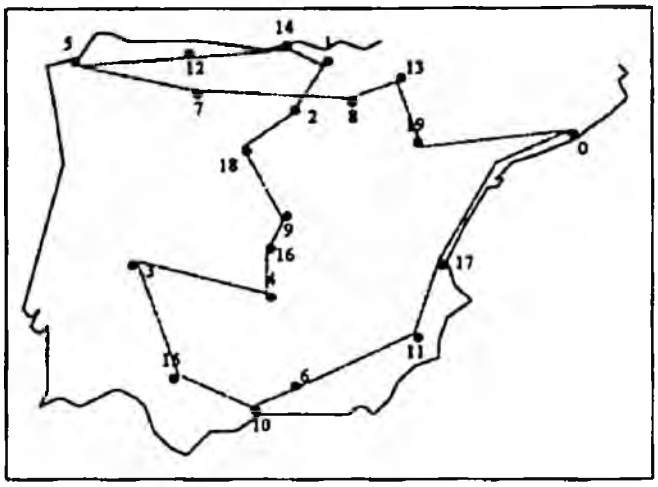

Figura I.28: Solución media al problema del viajante 20 con la red ODE: 4331 km.

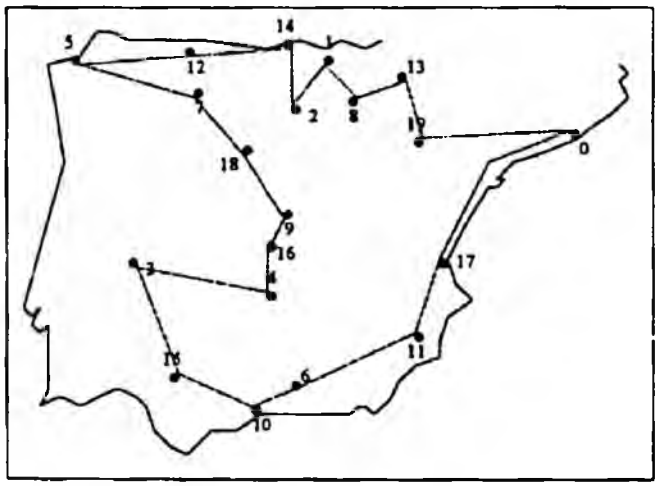

Figura I.29: Solución mejor al problema del viajante 20 con la red ODE: 4142 km.

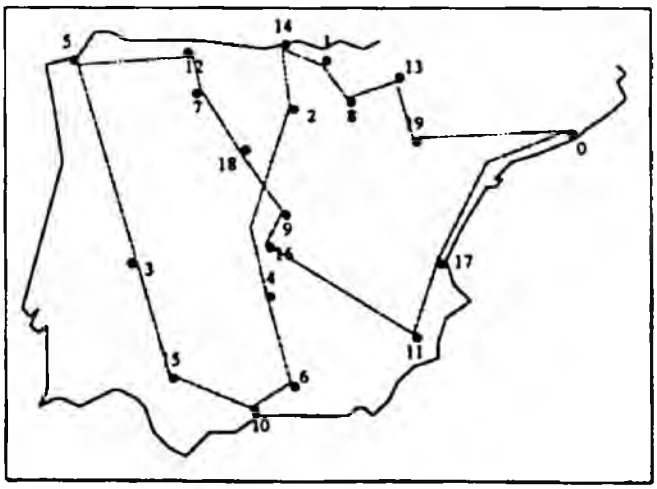

Figura I.30: Solución peor al problema del viajante 20 con la red ODE: 4811 km.

I.6 DATOS UTILIZADOS EN LAS PRUEBAS.

A continuación se ofrecen las matrices de distancias, utilizadas en las pruebas del problema del viajante comercial:

	Bar.	Bil.	Cac.	Coru.	Log.	Mad.	Mur.	Ovi.	Pam.	Sant.	Sev.	Tol.	Val.	Vall	Zar.
Bar.	0	620	918	1118	468	621	590	902	437	693	1046	692	349	663	296
Bil.	620	0	605	644	152	395	796	304	159	108	933	466	633	280	324
Cac.	918	605	0	683	595	297	654	525	650	573	264	264	636	325	622
Coru.	1118	644	683	0	650	609	1010	340	738	547	947	675	961	455	833
Log.	468	152	595	650	0	336	694	391	88	225	874	407	481	237	172
Mad.	621	395	297	609	336	0	401	451	407	393	538	71	352	193	325
Mur.	590	796	654	1010	694	401	0	852	714	794	534	390	241	594	539
Ovi.	902	304	525	340	391	451	852	0	463	207	789	510	803	252	604
Pam.	437	159	650	738	88	407	714	463	0	267	945	478	501	325	175
Sant.	693	108	573	547	225	393	794	207	267	0	837	464	673	248	397
Sev.	1046	933	264	947	874	538	534	789	945	837	0	458	697	589	863
Tol.	692	466	264	675	407	71	390	510	478	464	458	0	372	258	396
Val.	349	633	636	961	481	352	241	803	501	673	697	372	0	545	326
Vall.	663	280	325	455	237	193	594	252	325	248	589	258	545	0	367
Zar.	296	324	622	833	172	325	539	604	175	397	863	396	326	367	0

Tabla I.11: Matriz de distancias entre 15 ciudades españolas.

	Bar.	Bil.	Bur.	Cac.	Ciu.	Coru.	Ora.	Leo.	Log.	Mad.	Mal.	Mur.	Ovi.	Pam.	Sant.	Sev.	Tol.	Val.	Vall.	Zar.
Bar.	0	620	583	918	811	1118	868	784	468	621	997	590	902	437	693	1046	692	349	663	296
Bil.	620	0	158	605	585	644	829	359	152	395	939	796	304	159	108	933	466	633	280	324
Bur.	583	158	0	447	427	535	671	201	115	237	781	638	322	203	156	775	308	517	122	287
Cac.	918	605	447	0	324	683	485	407	595	297	506	654	525	650	573	264	264	636	325	622
Ciu.	811	585	427	324	0	799	278	511	526	190	388	357	641	597	583	339	119	398	377	515
Coru.	1118	644	535	683	799	0	1043	334	650	609	1153	1010	340	738	547	947	675	961	455	833
Ora.	868	829	671	485	278	1043	0	761	770	434	129	278	885	841	827	256	397	519	627	759
Leo.	784	359	201	407	511	334	761	0	316	333	877	734	118	404	293	671	392	685	134	488
Log.	468	152	115	595	526	650	770	316	0	336	880	694	391	88	225	874	407	481	237	172
Mad.	621	395	237	297	190	609	434	333	336	0	544	401	451	407	393	538	71	352	193	325
Mal.	997	939	781	506	388	1153	129	877	880	544	0	407	995	951	937	219	507	648	737	869
Mur.	590	796	638	654	357	1010	278	734	694	401	407	0	852	714	794	534	390	241	594	539
Ovi.	902	304	322	525	641	340	885	118	391	451	995	852	0	463	207	789	510	803	252	604
Pam.	437	159	203	650	597	738	841	404	88	407	951	714	463	0	267	945	478	501	325	175
Sant.	693	108	156	573	583	547	827	293	225	393	937	794	207	267	0	837	464	673	248	397
Sev.	1046	933	775	264	339	947	256	671	874	538	219	534	789	945	837	0	458	697	589	863
Tol.	692	466	308	264	119	675	397	392	407	71	507	390	510	478	464	458	0	372	258	396
Val.	349	633	517	636	398	961	519	685	481	352	648	241	803	501	673	697	372	0	545	326
Vall.	663	280	122	325	377	455	627	134	237	193	737	594	252	325	248	589	258	545	0	367
Zar.	296	324	287	622	515	833	759	488	172	325	869	539	604	175	397	863	396	326	367	0

Tabla I.12: Matriz de distancias entre 20 ciudades españolas.

APÉNDICE II

Pruebas Problema de la Partición de Grafos

Este apéndice contiene los resultados completos de las pruebas del problema de la partición de grafos, desarrolladas con el fin de realizar el análisis práctico de la eficiencia del modelo neuronal ODE, respecto a los métodos actuales de optimización combinatoria más importantes: búsqueda local, temple simulado, red de Hopfield continua y máquina de Boltzmann.

Para efectuar las pruebas, se han realizado en lenguaje C las diferentes implementaciones del problema de la partición de grafos con cada uno de los métodos de comparación, utilizando en su desarrollo una metodología de diseño y programación modular basada en máquinas abstractas. Estas pruebas se han realizado sobre dos instancias del problema, correspondientes a la separación de un grafo de 50 vértices en 2 y 5 particiones respectivamente, ejecutando 20 veces cada instancia con cada método de resolución.

Los resultados se presentan agrupados en cinco apartados, correspondientes a cada una de las técnicas de resolución empleadas. Así mismo, al final del apéndice se proporciona la matriz de conexiones correspondiente al grafo que se ha utilizado en las pruebas.

II.1 PRUEBAS PPG BÚSQUEDA LOCAL.

La tabla II.1 presenta los resultados de las pruebas, realizadas mediante el algoritmo de búsqueda local, del problema de la partición de grafos sobre un grafo de 50 vértices dividido en 2 particiones. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras II.1, II.2 y II.3):

I	COSTE	TIEMPO	SOLUCIÓN
1	10	5	1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 1 1 1 0 1 1 0 0 0 1 1 1 0 0 1 0 0 0 0 1 1 1 0 0
2	8	10	0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1
3	8	5	1 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0
4	4	8	1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0
5	4	20	0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1
6	12	5	0 0 1 0 0 0 0 1 1 1 0 1 1 0 0 0 0 1 1 1 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1
7	10	8	1 1 1 0 0 0 0 1 1 0 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 1 0 0 0 0 1 1 0 0
8	9	9	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
9	10	8	1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
10	10	5	1 0 1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 0 0 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0
11	9	9	0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 0 1 1
12	4	8	1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0
13	14	9	0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 0
14	10	14	0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1
15	6	17	0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 0 0 0 0 1 0 0 1 1 1 0 0 0 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1
16	10	11	1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 1 0 1 1
17	9	9	1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 1 0 1 1 1 1 1 0 0 0 0 0 0 0 1 1
18	13	10	1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
19	10	5	0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1
20	10	9	0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1

Tabla II.1: Pruebas partición de grafos 50/2 con búsqueda local.

COSTE		**TIEMPO**	
Medio:	9.0 conexiones	Medio:	9.2 sg
Mejor:	4 conexiones	Mejor:	5 sg
Peor:	14 conexiones	Peor:	20 sg
Desviación:	2.8 conexiones	Desviación:	4.0 sg

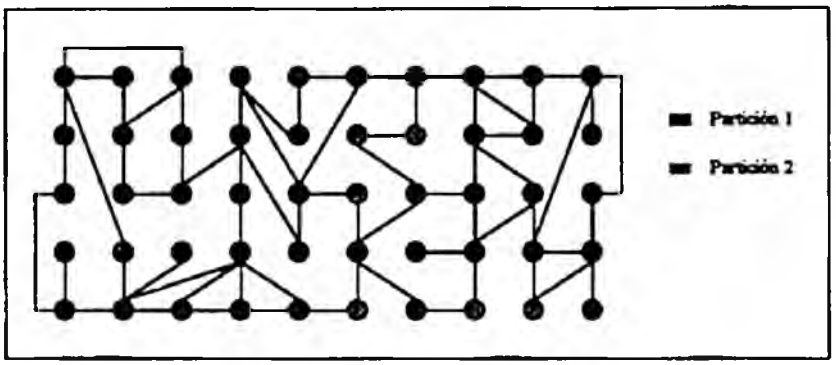

Figura II.1: Solución media a la partición de grafos 50/2 con búsqueda local: 9 conex.

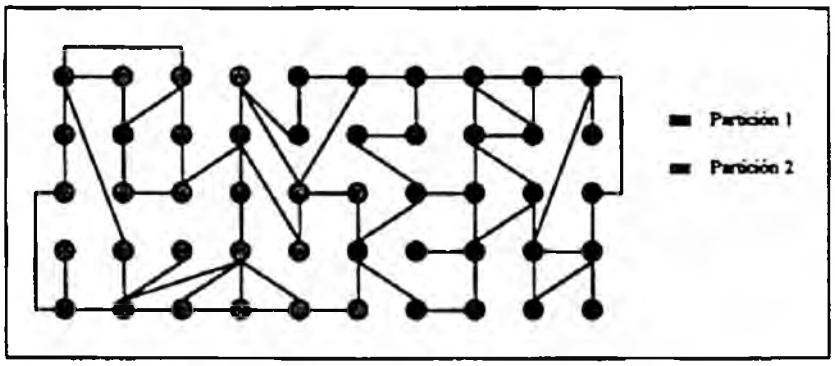

Figura II.2: Solución mejor a la partición de grafos 50/2 con búsqueda local: 4 conex.

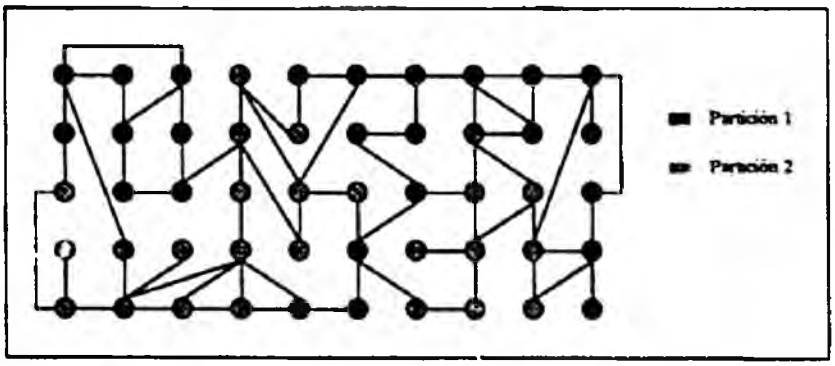

Figura II.3: Solución peor a la partición de grafos 50/2 con búsqueda local: 14 conex.

La tabla siguiente presenta los resultados de las pruebas, realizadas mediante el algoritmo de búsqueda local, del problema de la partición de grafos sobre un grafo de 50 vértices dividido en 5 particiones. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras II.4, II.5 y II.6):

I	COSTE	TIEMPO	SOLUCIÓN
1	23	27	44132220010214302001021 42 20001444342331144333323111
2	18	55	22233311102224311112211 43 33300422433000044444400011
3	22	39	33300330022311003002211 13 34001244434221124444422111
4	19	41	33301441121320144112122 00 44332133004332210000444222
5	26	40	30422111141040244114100 01 14034233202003333222244333
6	20	46	20011133322001143332200 41 44332220414332120004444111
7	20	62	44031110014003111001400 23 31041422233443342222234 32
8	25	34	44444001131443411113122 30 01103433202330043222222 00
9	22	57	22244443302223411330233 34 11302400131000240111144 02
10	20	42	04041401131404100113144 24 40033122240223312222033 33
11	19	23	42400011114243011111224 30 04441233300442223333300 22
12	19	31	22003041124000344112400 10 34312424103332244111333 22
13	19	53	11144032201134433220133 40 02220110440223311444002 33
14	18	49	22211000032221100004211 40 01333224411333324444413 34
15	16	63	11140000001114400300211 44 43331222244133322222443 33
16	14	38	33300044410330044441033 00 34411222203111122222234 11
17	18	48	23200002242323000224433 30 01124421301114441333111 44
18	17	59	44421111124442121112444 22 32002133323000043333300 00
19	24	37	33312200003341100002244 12 20043232411444322441111 33
20	13	108	00022223340002232334000 12 23334404121334441111111 44

Tabla II.2: Pruebas partición de grafos 50/5 con búsqueda local.

COSTE		**TIEMPO**	
Medio:	19.6 conexiones	Medio:	47.6 sg
Mejor:	13 conexiones	Mejor:	23 sg
Peor:	26 conexiones	Peor:	108 sg
Desviación:	3.4 conexiones	Desviación:	18.2 sg

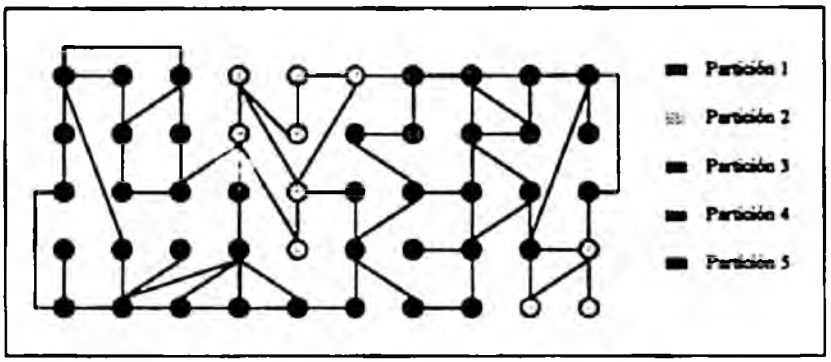

Figura II.4: Solución media a la partición de grafos 50/5 con búsqueda local: 20 conex.

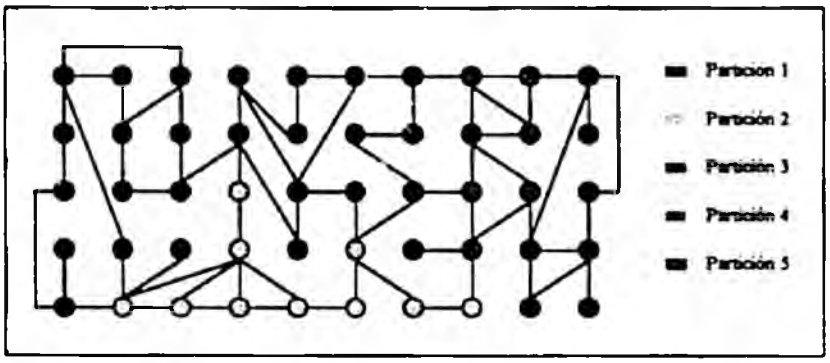

Figura II.5: Solución mejor a la partición de grafos 50/5 con búsqueda local: 13 conex.

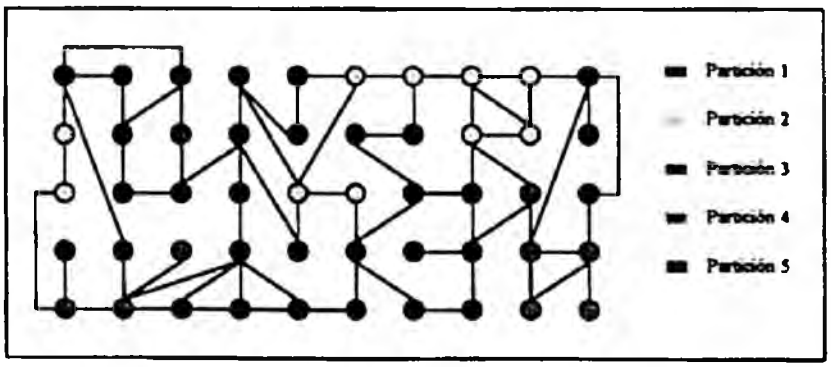

Figura II.6: Solución peor a la partición de grafos 50/5 con búsqueda local: 26 conex.

II.2 PRUEBAS PPG TEMPLE SIMULADO.

La tabla II.3 presenta los resultados de las pruebas, realizadas mediante el algoritmo del temple simulado, del problema de la partición de grafos sobre un grafo de 50 vértices dividido en 2 particiones. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras II.7, II.8 y II.9):

I	COSTE	TIEMPO	SOLUCIÓN
1	6	379	00011111110000111111001101 11110000011110000000001100
2	5	413	11100000011110000000111100 00001111100001111111100011
3	4	373	00001111111000011111100000 01111000001111100000011111
4	4	374	11111000001111100000111111 00000111110000011111100000
5	8	417	00000111100000011110000011 11110000111110000111111100
6	4	376	11111000001111100000111110 00000111110000011111110000
7	6	383	00011110000001111100000011 11000101111000011111111100
8	4	372	11111100001111100000111111 00000110110000011111100000
9	4	360	11111100001111100000111111 00000110110000011111100000
10	6	380	11100001111111000011101100 00111010000111100000000111
11	4	363	00001111110000011111100000 01111000001111100000111111
12	4	383	00000011110003001111100000 11111000001111100001111111
13	8	374	10100000011011000001101110 00001111111000111111100011
14	4	370	11111100001111100000111111 00000011110000011111100000
15	4	345	11110000001111000000111111 10000111110000011111110000
16	4	390	11110000001111000000111111 10000111110000011111110000
17	6	388	11100001111111000011111100 00111010000111100000000011
18	5	368	00011111111000011111100001 11111000000011110000000011
19	4	361	11111100001111100000111111 00000111110000011110000000
20	7	410	00011110010000111101000001 11111000001111100000011111

Tabla II.3: Pruebas partición de grafos 50/2 con temple simulado.

COSTE		**TIEMPO**	
Medio:	5.1 conexiones	Medio:	379.0 sg
Mejor:	4 conexiones	Mejor:	345 sg
Peor:	8 conexiones	Peor:	417 sg
Desviación:	1.4 conexiones	Desviación:	18.1 sg

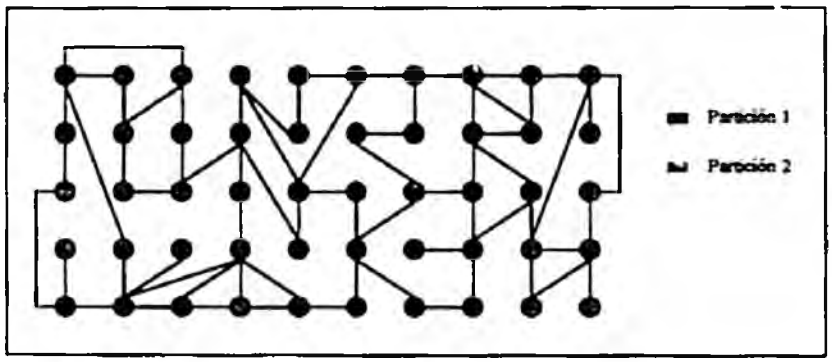

Figura II.7: Solución media a la partición de grafos 50/2 con temple simulado: 5 conex.

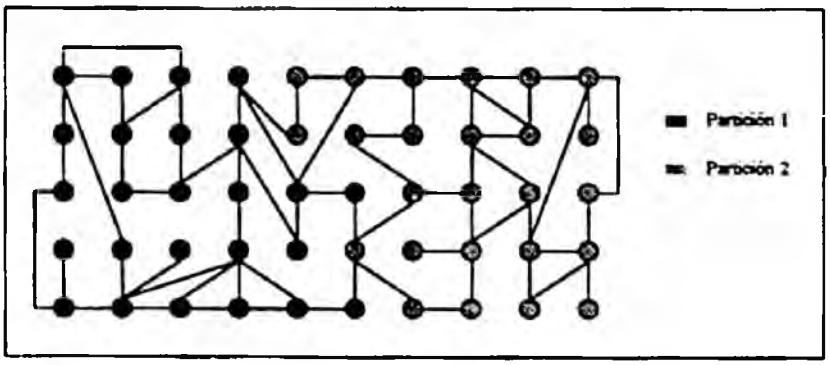

Figura II.8: Solución mejor a la partición de grafos 50/2 con temple simulado: 4 conex.

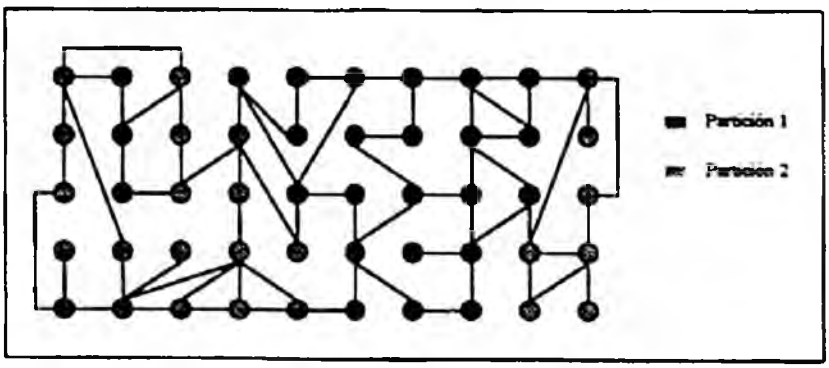

Figura II.9: Solución peor a la partición de grafos 50/2 con temple simulado: 8 conex.

La tabla siguiente presenta los resultados de las pruebas, realizadas mediante el algoritmo del temple simulado, del problema de la partición de grafos sobre un grafo de 50 vértices dividido en 5 particiones. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras II.10, II.11 y II.12):

I	COSTE	TIEMPO	SOLUCIÓN
1	17	1188	4221100223422110022342210 0013334441011333444400133
2	15	1158	3300333222300001322430040 1111234440111223444441122
3	14	1120	4443332200443332222043313 1222004413102004411111100
4	14	1113	0003334444003332244400333 2224411113222401111112200
5	11	1175	0004444222000443432200014 4333210114333221111113322
6	12	1127	2224440003222440000322214 4003312114403331111114333
7	12	1128	2221111333222114133322211 4444300001444330000004432
8	18	1110	3300442333300042233320040 0221123440211112444421111
9	14	1077	0334444332033441133203344 4112222000411122200001122
10	16	1137	0002223113002223311304222 4441300412444330011114433
11	14	1219	3331000002333102000233341 1202243441111224444441122
12	13	1126	2223111114222311111422203 3334402003344440000003344
13	12	1098	0002224443000224444300012 2443310112233331111112334
14	14	1130	2224400004222143000422210 0333412110333441111133344
15	12	1094	4443333222444334312204403 3111204003111220000011122
16	15	1164	0003331444003331104423323 1100420223110442222211144
17	14	1117	3332223004433223300443342 2000411112200441111112044
18	13	1081	2221111334222114433422201 0433412001033441000003344
19	14	1155	3332200224333220022433312 0004413112044441111110044
20	12	1172	4441111000444113300244411 3330024221333002222223301

Tabla II.4: Pruebas partición de grafos 50/5 con temple simulado.

COSTE		**TIEMPO**	
Medio:	13.8 conexiones	Medio:	1134.5 sg
Mejor:	11 conexiones	Mejor:	1077 sg
Peor:	18 conexiones	Peor:	1219 sg
Desviación:	1.8 conexiones	Desviación:	36.9 sg

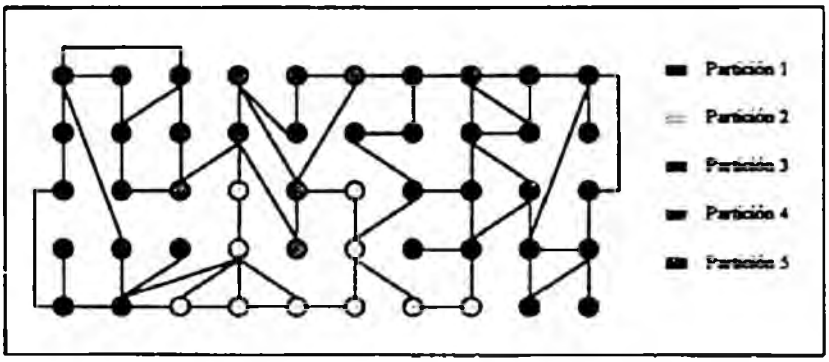

Figura II.10: Solución media a la partición de grafos 50/5 con temple sim.: 14 conex.

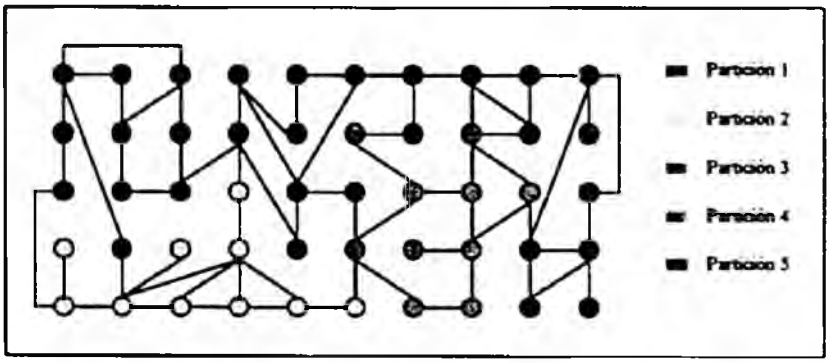

Figura II.11: Solución mejor a la partición de grafos 50/5 con temple sim.: 11 conex.

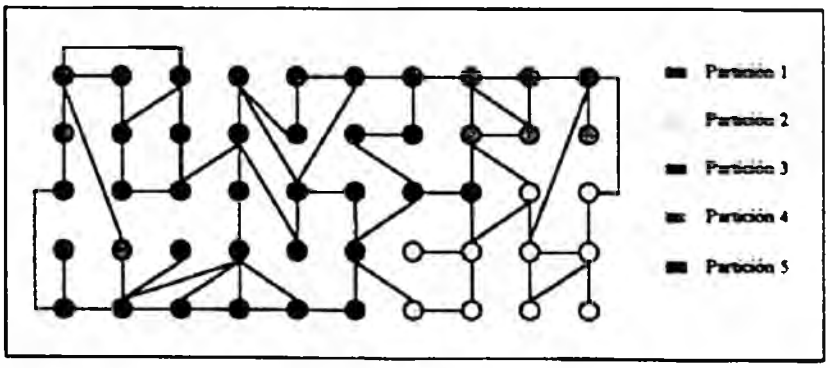

Figura II.12: Solución peor a la partición de grafos 50/5 con temple sim.: 18 conex.

II.3 PRUEBAS PPG RED DE HOPFIELD CONTINUA.

La tabla II.5 presenta los resultados de las pruebas, realizadas mediante la red de Hopfield continua, del problema de la partición de grafos sobre un grafo de 50 vértices dividido en 2 particiones. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras II.13, II.14 y II.15):

I	COSTE	TIEMPO	SOLUCIÓN
1	40	73	11010111010000000110001 10 10100001010110111001101 11
2	22	413	01100001101100000010110 01 10001101111111001111001 100
3	36	244	11110111011001010001001 01 01110010000011011110100001
4	31	119	11001111111011111010100001 0 01100011000010011111000001
5	23	713	00000000001010111000101 00 11101101001101110111011111
6	25	252	11100001101100000110101 10 10011100100100110011011 01
7	24	536	00101100000001110000111111 0 01101101100010111011001 01
8	37	186	01111100000011011111010100 01110000010100101101101 01
9	25	254	10100011001110000011111101 10001100101000011011111001
10	29	212	00011100010100110000110 01 11011111101010101000100111
11	34	160	00010111111001100110110001 00110100111011001010000 11
12	35	142	11110000110010101010011 00 10111011000010011101000 11
13	28	299	00000011000011101111011 00 01111111001100001111011 000
14	33	79	01000011100111111101111111 01011100000001010000100001
15	27	342	01110001000101100110111111 00010011101001101100000 11
16	32	78	10000111111100011100010011 01001101101001001001010 11
17	39	114	00000011111100100010101 01 01011000111101011010110 01
18	31	258	11100110000100101100010 11 00100011011111110100011001
19	31	253	10100010111100001011101110 00101101000001100111010 11
20	32	188	01101010001110001100111 10 10100001101001111000101 01

Tabla II.5: Pruebas partición de grafos 50/2 con la red de Hopfield continua.

COSTE		**TIEMPO**	
Medio:	30.7 conexiones	Medio:	245.8 sg
Mejor:	22 conexiones	Mejor:	73 sg
Peor:	40 conexiones	Peor:	713 sg
Desviación:	5.3 conexiones	Desviación:	160.3 sg

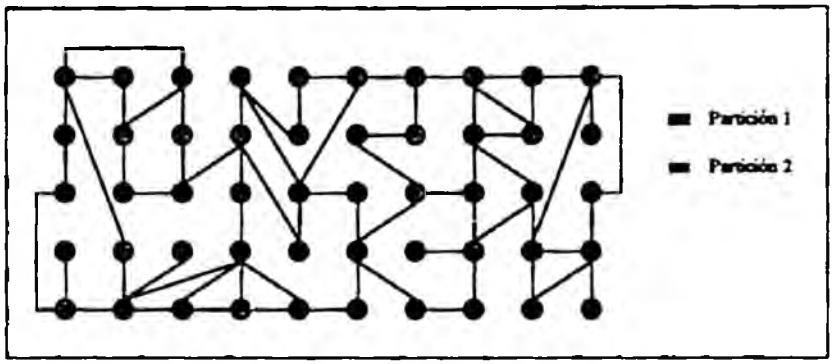

Figura II.13: Solución media a la partición de grafos 50/2 con la red CH: 31 conex.

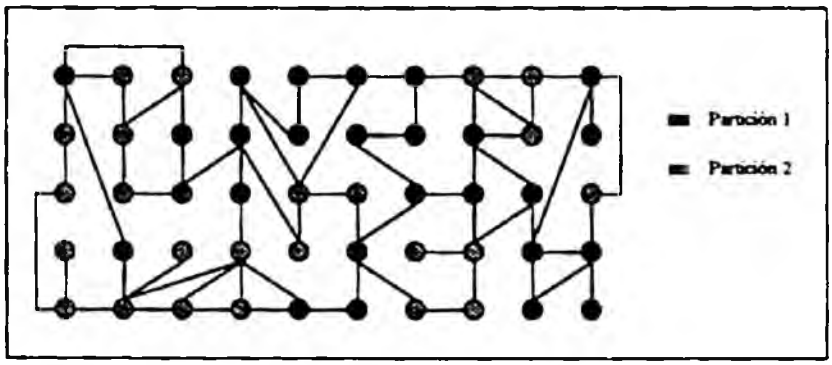

Figura II.14: Solución mejor a la partición de grafos 50/2 con la red CH: 22 conex.

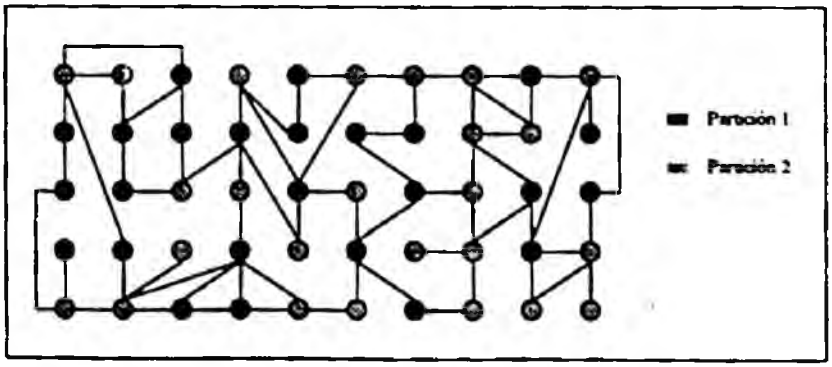

Figura II.15: Solución peor a la partición de grafos 50/2 con la red CH: 40 conex.

La tabla siguiente presenta los resultados de las pruebas, realizadas mediante la red de Hopfield continua, del problema de la partición de grafos sobre un grafo de 50 vértices dividido en 5 particiones. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras II.16, II.17 y II.18):

I	COSTE	TIEMPO	SOLUCIÓN
1	52	2277	1 0 3 1 0 2 4 2 3 4 2 0 1 4 0 3 2 0 2 4 0 4 1 3 1 4 4 2 1 0 2 3 4 3 0 4 1 1 3 2 2 3 2 0 4 1 0 1 3 3
2	54	4153	2 0 1 0 4 1 0 2 4 3 0 1 0 0 4 2 2 3 4 1 2 4 4 1 3 1 3 4 0 1 0 0 3 3 3 4 2 4 0 2 3 3 4 2 1 1 2 1 2 3
3	54	5146	1 2 3 0 1 0 3 1 3 1 2 0 3 3 4 4 4 0 0 3 0 4 3 2 2 2 3 1 2 0 4 0 2 0 2 3 4 4 1 1 1 3 0 1 4 4 1 4 2 2
4	55	2152	4 4 2 4 3 4 0 1 2 0 1 2 1 3 1 1 0 4 2 0 0 2 3 1 3 2 2 0 3 3 1 3 3 4 0 4 0 3 4 3 1 1 0 4 0 1 2 2 2 4
5	61	1796	1 4 0 4 2 2 0 3 2 1 3 3 3 3 1 2 3 4 0 4 2 2 4 4 0 1 3 1 4 3 0 2 1 3 1 1 0 0 1 4 1 0 3 4 0 0 4 2 2 2
6	48	3067	1 1 3 4 0 3 3 0 2 0 1 3 0 2 4 1 3 3 3 0 1 4 2 2 4 4 4 4 3 2 0 4 4 0 0 4 0 1 1 0 3 2 1 2 2 3 2 2 1 1
7	53	3889	4 2 2 4 4 2 2 0 3 3 4 1 0 3 2 1 1 4 3 3 2 2 1 0 2 0 4 4 1 3 3 0 0 0 3 1 2 0 1 0 0 1 3 1 3 4 4 4 1 2
8	60	761	0 3 1 2 0 2 1 4 3 2 3 3 4 3 4 3 4 2 4 4 2 0 1 2 4 4 1 1 1 0 0 0 0 1 2 2 3 2 4 3 3 3 1 0 1 4 1 0 0 2
9	50	3676	0 0 3 3 2 4 3 3 0 3 4 3 2 2 3 4 1 3 0 2 0 4 2 1 4 1 0 4 1 1 4 0 1 1 2 2 1 1 0 2 4 3 3 2 1 4 2 0 4 0
10	46	5712	0 4 4 4 3 3 0 2 0 1 0 2 4 3 4 2 1 2 0 1 2 0 1 3 2 2 3 4 2 3 4 1 0 3 2 1 0 1 3 0 4 1 3 3 4 4 1 1 2 0
11	51	3838	0 1 3 2 1 2 2 0 0 4 0 4 4 4 1 3 2 1 1 0 4 3 0 2 3 3 3 1 1 3 0 4 2 4 1 4 3 3 4 2 4 1 0 0 1 2 0 3 2 2
12	54	967	2 1 2 0 3 0 4 3 4 0 0 3 0 2 2 4 0 1 3 0 2 4 1 1 0 0 2 2 4 3 1 1 4 1 2 3 1 3 4 3 1 4 2 3 3 4 2 4 1 0
13	49	3455	2 4 4 3 3 3 3 3 0 0 0 4 4 2 1 0 1 3 2 1 1 2 0 1 2 4 4 4 0 3 4 2 0 1 0 2 1 3 3 3 4 2 1 1 2 1 2 0 0 4
14	47	4138	0 0 3 2 2 1 2 0 2 1 3 2 0 2 0 4 4 3 0 4 2 0 3 2 1 4 2 1 3 0 1 1 3 1 1 4 3 0 3 3 1 1 4 4 4 2 4 0 3 4
15	60	1563	1 3 0 1 1 3 0 1 4 3 3 3 3 1 4 0 4 4 0 2 0 0 3 4 2 1 0 2 0 0 2 4 4 1 3 1 2 1 2 4 2 0 2 2 1 4 2 4 3 3
16	55	1082	2 4 4 1 2 2 3 0 1 4 0 2 3 1 0 0 3 0 4 0 4 0 1 4 0 3 3 3 2 1 4 2 2 4 1 2 1 3 4 0 3 1 2 3 1 2 4 1 3 0
17	44	2762	2 1 2 0 2 0 2 3 3 3 4 0 4 2 0 0 0 1 3 3 4 1 1 2 0 3 4 2 0 2 4 1 1 1 1 0 0 2 4 4 4 3 3 3 1 1 2 3 4 4
18	50	925	0 3 4 4 3 4 0 2 2 4 3 3 0 3 1 1 1 4 2 4 3 1 3 3 4 0 0 0 4 2 2 0 4 0 1 2 4 3 0 2 2 1 2 1 1 3 1 0 1 2
19	56	2056	4 4 4 3 2 1 0 1 0 3 0 0 1 4 1 3 1 1 2 3 2 2 4 4 0 1 4 3 1 2 3 4 0 4 1 2 3 2 2 4 3 1 0 3 0 2 2 3 0 0
20	54	2491	2 3 4 4 3 3 3 2 2 4 4 0 1 0 0 2 1 2 1 4 4 0 0 0 2 1 3 4 3 2 1 0 0 1 3 2 1 2 2 4 3 3 0 4 1 3 1 1 0 4

Tabla II.6: Pruebas partición de grafos 50/5 con la red de Hopfield continua.

COSTE		TIEMPO	
Medio:	52.7 conexiones	Medio:	2795.3 sg
Mejor:	44 conexiones	Mejor:	761 sg
Peor:	61 conexiones	Peor:	5712 sg
Desviación:	4.6 conexiones	Desviación:	1432.7 sg

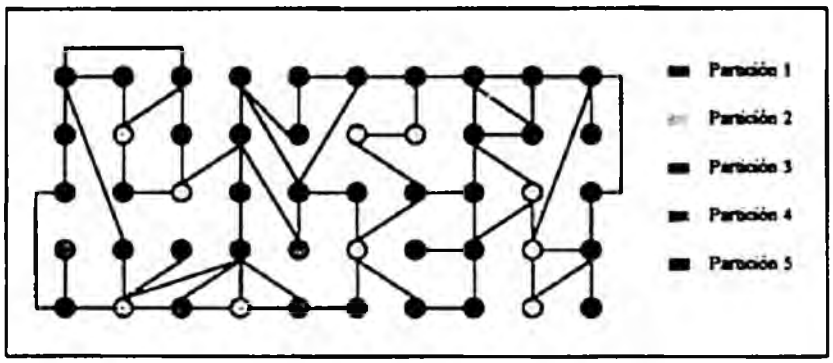

Figura II.16: Solución media a la partición de grafos 50/5 con la red CH: 53 conex.

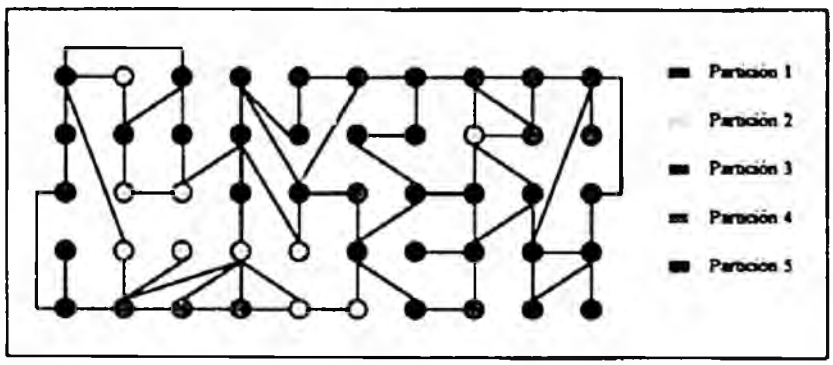

Figura II.17: Solución mejor a la partición de grafos 50/5 con la red CH: 44 conex.

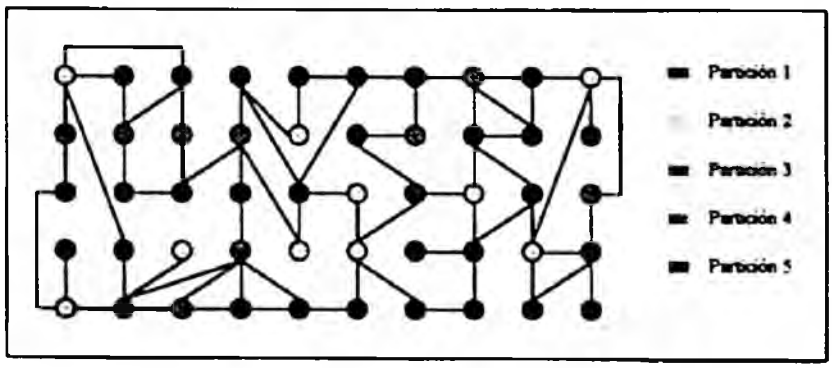

Figura II.18: Solución peor a la partición de grafos 50/5 con la red CH: 61 conex.

II.4 PRUEBAS PPG MÁQUINA DE BOLTZMANN.

La tabla II.7 presenta los resultados de las pruebas, realizadas mediante la máquina de Boltzmann, del problema de la partición de grafos sobre un grafo de 50 vértices dividido en 2 particiones. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras II.19, II.20 y II.21):

I	COSTE	TIEMPO	SOLUCIÓN
1	28	21	11111110101110000000010001 01010011000111001111000101
2	27	21	11101110011100000101111110 00001110000101101110000010
3	32	20	01101001111100101100010011 10011000111001100001011100
4	29	22	01000110010111001111101100 10011001001100100010011111
5	30	22	10011000111000000011011111 10111001111001100000100011
6	30	23	10111100010000111011000011 10001011010110111100010000
7	36	21	11101100000100101100100000 11111001010101111101100100
8	40	21	10110011100000010100010001 01001011010111011111111000
9	37	20	01110111001001100001000000 01100110111111011100010011
10	40	21	01111110100000111101111011 00001100101010100001010001
11	31	22	00001111100101101100000100 01101111001010110110101011
12	35	19	01001000111101010010111100 10000110001111110111001010
13	33	21	01111011110100100110011111 00010010011100010001100010
14	33	21	01000011100100010100101100 01010110111110011111001001 1
15	37	22	10010101001101001011111010 00001101011001100011101110
16	40	23	11101010111011000000011001 11110000101010100100011110
17	34	20	00101110010101001100100 11 10110100101011001110000101
18	36	20	00000011100101111000101111 01111100111101010000100 10
19	43	20	00001011101110010001001 01 00001111011111110110110000
20	31	20	00001100100011101100011110 01101110110111101000001 10

Tabla II.7: Pruebas partición de grafos 50/2 con la máquina de Boltzmann.

COSTE		TIEMPO	
Medio:	34.1 conexiones	Medio:	21.0 sg
Mejor:	27 conexiones	Mejor:	19 sg
Peor:	43 conexiones	Peor:	23 sg
Desviación:	4.5 conexiones	Desviación:	1.1 sg

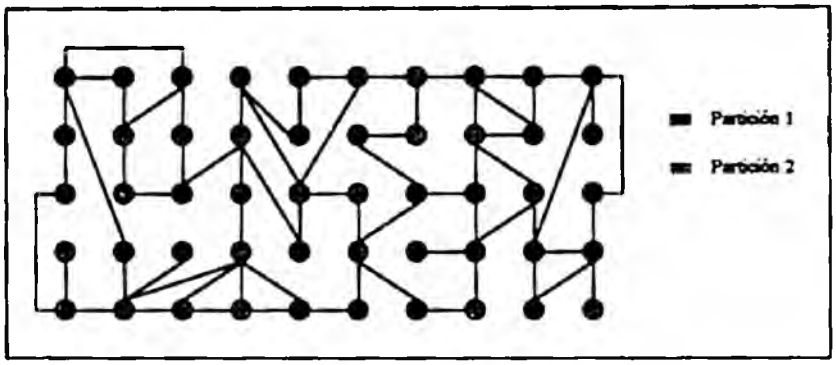

Figura II.19: Solución media a la partición de grafos 50/2 con la red BM: 34 conex.

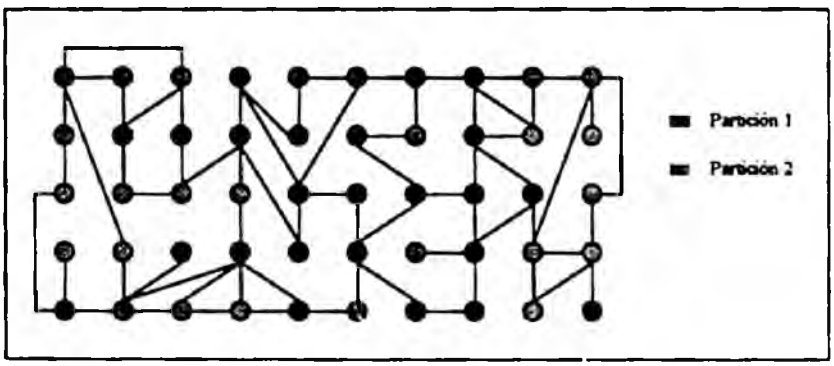

Figura II.20: Solución mejor a la partición de grafos 50/2 con la red BM: 27 conex.

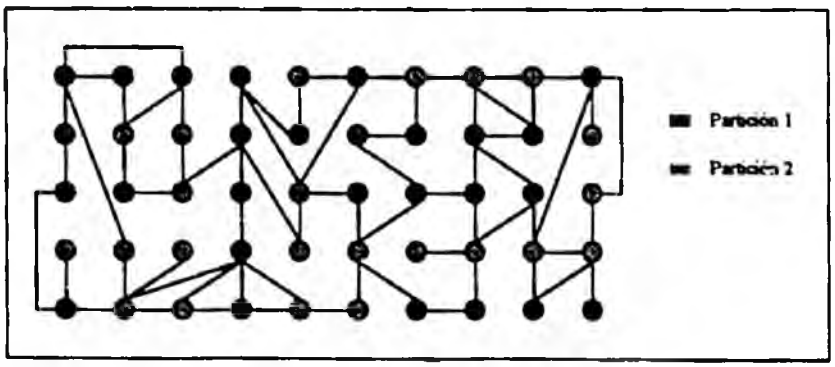

Figura II.21: Solución peor a la partición de grafos 50/2 con la red BM: 43 conex.

La tabla siguiente presenta los resultados de las pruebas, realizadas mediante la máquina de Boltzmann, del problema de la partición de grafos sobre un grafo de 50 vértices dividido en 5 particiones. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras II.22, II.23 y II.24):

I	COSTE	TIEMPO	SOLUCIÓN
1	55	94	43343230341304311103422 22 01441124432232002001104 10
2	57	97	14424140023004111440220 33 42001120221333412333324 01
3	52	102	21101131242244144414330 01 01233320202430302004233 41
4	54	95	04003303123112411142034 40 21412320021343242403032 41
5	58	98	41444101221232304003402 24 13040341132233312302100 41
6	45	100	14441211120413244002244 33 33302310311340210030402 22
7	54	96	30422200141401314013320 33 03021332214404221240441 31
8	62	93	24310210431223202204414 12 01003433233021314034441 01
9	53	99	13141304134202222211342 24 10432434000110233430301 40
10	57	101	41033322013132231310422 02 11341400411202004043443 42
11	50	97	40400210203204014033321 23 41103314341022344421123 21
12	56	98	14040244331323100142210 12 12034243400403313112432 20
13	54	96	34440223102140014420142 21 11231003422323304403313 10
14	54	93	30332443024430231114200 11 04213112024402123303024 14
15	57	101	14231314030012020413233 03 12112444303403242401122 40
16	59	94	02012412433330420214420 13 40342043210011214332130 14
17	52	96	32134324003224042430304 22 33301034104021011421412 11
18	56	98	40431324412121204041041 32 20210332323130314340100 24
19	58	98	01002104131243104413002 03 21322412413432014320344 23
20	57	99	02240302402143100323414 13 24232310030430414142131 21

Tabla II.8: Pruebas partición de grafos 50/5 con la máquina de Boltzmann.

COSTE		**TIEMPO**	
Medio:	55.0 conexiones	Medio:	97.3 sg
Mejor:	45 conexiones	Mejor:	93 sg
Peor:	62 conexiones	Peor:	102 sg
Desviación:	3.6 conexiones	Desviación:	2.7 sg

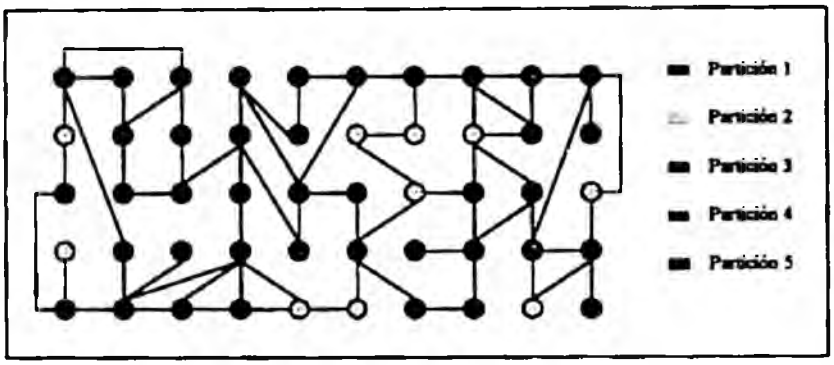

Figura II.23: Solución media a la partición de grafos 50/5 con la red BM: 55 conex.

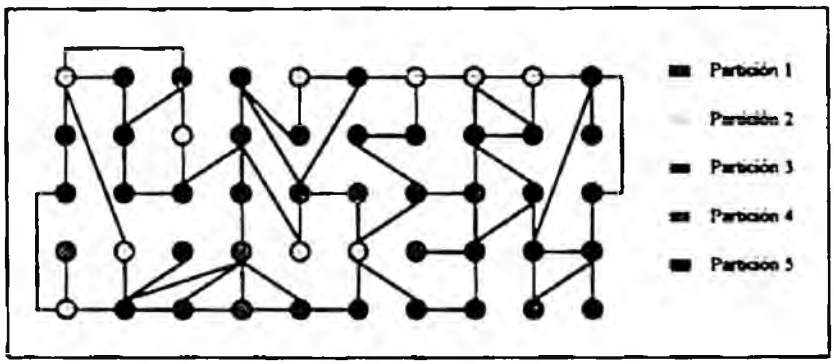

Figura II.23: Solución mejor a la partición de grafos 50/5 con la red BM: 45 conex.

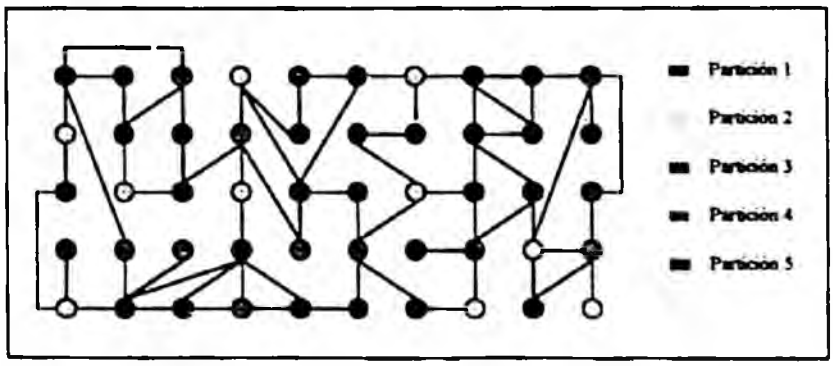

Figura II.24: Solución peor a la partición de grafos 50/5 con la red BM: 62 conex.

II.5 PRUEBAS PPG OPTIMIZADOR DISCRETO ESTOCÁSTICO.

La tabla II.9 presenta los resultados de las pruebas, realizadas mediante la red ODE, del problema de la partición de grafos sobre un grafo de 50 vértices dividido en 2 particiones. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras II.25, II.26 y II.27):

I	COSTE	TIEMPO	SOLUCIÓN
1	4	506	11111100001111100000011111 00000011110000011111100000
2	7	506	01111110000111111100001111 11000000111000000011110000
3	4	503	00000111110000011111100000 11111000000111110000011111
4	9	499	11100011111111001111101100 01111010000110000000000100
5	4	504	11111100001111100000011111 00000110110000011111100000
6	6	498	00011110000000111100000011 11000101111000011111111100
7	4	495	00000111110000011111100001 11111000001111100000011111
8	5	497	11100000001111000000111100 00001111100001111111100011
9	4	503	00001111111000001111100000 11111000001111100000011111
10	4	500	11110000001111100000111111 10000111110000011111100000
11	6	508	00011111111000110011100001 10111000010111100000000111
12	4	507	11110000001111100000111111 00000111110000011111110000
13	4	500	11110000001111100000011111 10000111110000011111110000
14	5	524	11100000001111000000111110 10000111101000011111111100
15	4	496	11111000001111100000111111 00000111110000011111100000
16	4	506	11111100001111100000111111 00000011110000011111100000
17	4	514	00001111111000011111100000 01111000001111100000011111
18	4	500	11111100001111100000111111 00000111110000011110000000
19	5	500	11100000001111000000111110 10000111111000011111111000
20	4	508	00000111110000011111100001 11111000001111100000011111

Tabla II.9: Pruebas partición de grafos 50/2 con el optimizador discreto estocástico.

COSTE		**TIEMPO**	
Medio:	4.8 conexiones	Medio:	503.7 sg
Mejor:	4 conexiones	Mejor:	495 sg
Peor:	9 conexiones	Peor:	524 sg
Desviación:	1.3 conexiones	Desviación:	6.8 sg

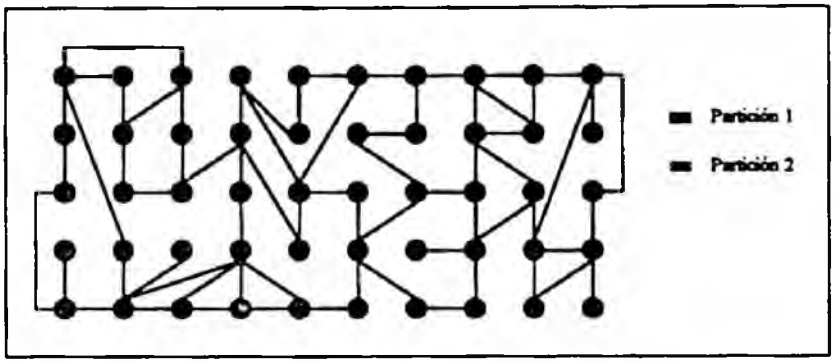

Figura II.25: Solución media a la partición de grafos 50/2 con la red ODE: 5 conex.

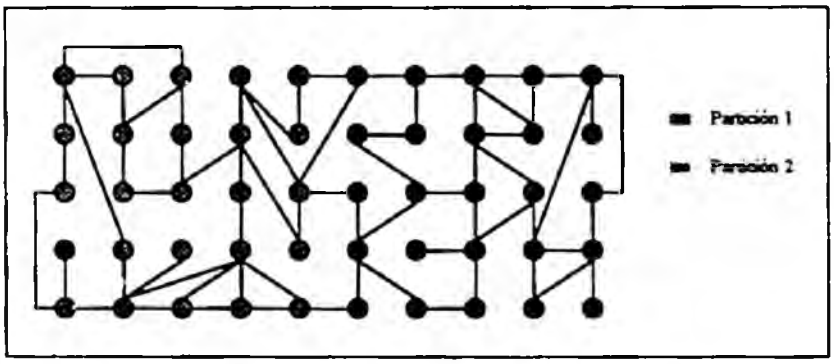

Figura II.26: Solución mejor a la partición de grafos 50/2 con la red ODE: 4 conex.

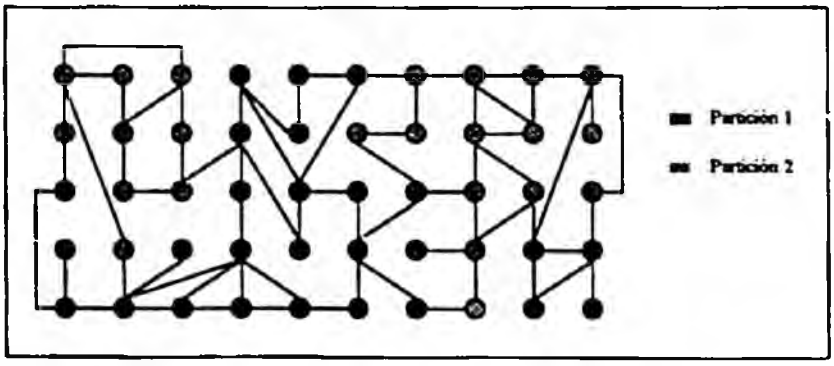

Figura II.27: Solución peor a la partición de grafos 50/2 con la red ODE: 9 conex.

La tabla siguiente presenta los resultados de las pruebas, realizadas mediante la red ODE, del problema de la partición de grafos sobre un grafo de 50 vértices dividido en 5 particiones. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras II.28, II.29 y II.30):

I	COSTE	TIEMPO	SOLUCIÓN
1	12	3720	2220000444222002234412200 033341111033344111113334 4
2	13	3697	1110000332111002033211100 443322144043322244444332 2
3	16	3684	3220011113322201111342222 011034344200033444444 0033
4	11	3709	2223333000222331310022243 311104244311100444444 1100
5	17	3784	2223333444221331144321103 311441200302244100000024 1
6	13	3772	4443333112444330311224433 001122440301122200000 1122
7	13	3722	2220003331222003333122200 033114444031111444444 2110
8	14	3720	1112222000111223340011222 333401444233300144444 3300
9	12	3684	0002222111000224211130032 244413033244411333334444 10
10	14	3757	2224440330222440033012444 433301211433300111111 2200
11	15	3658	1111333330111133333041142 222004044220200444442 2200
12	16	3608	1110000114211003021421100 332242443032244243333 3244
13	12	3644	1112222333111222233111142 000034444200033444440 0033
14	12	3658	3334444222333440022333344 400021111400022111111 0022
15	12	3760	1114444333111442433101144 222230100422233000000 2233
16	14	3682	2221112000122112200312211 444403333144400333334 4400
17	14	3748	3331411224333141122433311 122240340102244000000 0244
18	12	3720	3330000444333002244313300 222441311022244111111 2240
19	15	3721	2220111110222011111032204 444403233044400333333 4402
20	13	3723	1112233442111123344211103 334420000334422000003 4422

Tabla II.10: Pruebas partición de grafos 50/5 con el optimizador discreto estocástico.

COSTE		**TIEMPO**	
Medio:	13.5 conexiones	Medio:	3708.6.x sg
Mejor:	11 conexiones	Mejor:	3608 sg
Peor:	17 conexiones	Peor:	3784 sg
Desviación:	1.6 conexiones	Desviación:	45.1 sg

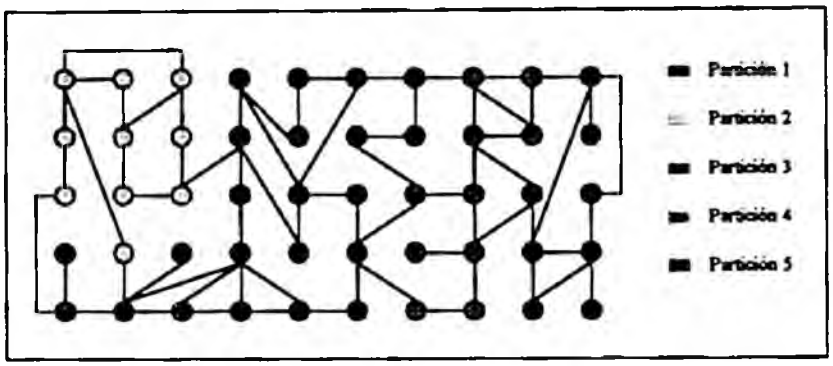

Figura II.28: Solución media a la partición de grafos 50/5 con la red ODE: 13 conex.

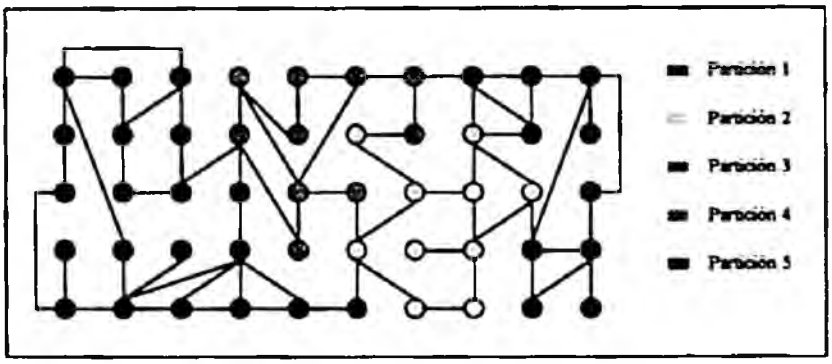

Figura II.29: Solución mejor a la partición de grafos 50/5 con la red ODE: 11 conex.

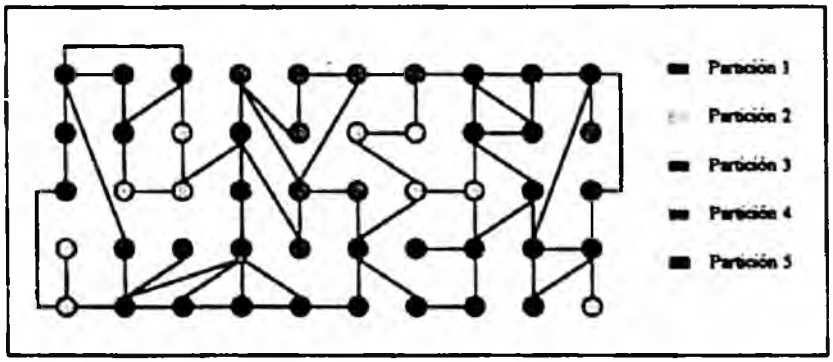

Figura II.30: Solución peor a la partición de grafos 50/5 con la red ODE: 17 conex.

II.6 DATOS UTILIZADOS EN LAS PRUEBAS.

A continuación se ofrece la matriz de conexiones del grafo plano de 50 vértices, utilizado en las pruebas del problema de la partición de grafos:

```
01100000001000000000000000000001000000000000000000
10000000001000000000000000000000000000000000000000
10000000001100000000000000000000000000000000000000
00000000000011000000001000000000000000000000000000
00001000000001000000000000000000000000000000000000
00010100000000000000001000000000000000000000000000
00001010000000001000000000000000000000000000000000
00000101000000001100000000000000000000000000000000
00000010100000000100000000000000000000000000000000
00000001010000000010000000001000000001000000000000
10000000000000000001000000000000000000000000000000
01100000000000000001000000000000000000000000000000
00100000000000000001000000000000000000000000000000
00010000000000000011000000001000000000000000000000
00011000000000000000000000000000000000000000000000
00000000000000001000000001000000000000000000000000
00000010000000001000000000000000000000000000000000
00000001000000001000000001100000000000000000000000
00000011000000001000000000000000000000000000000000
00000000100000000000000000000000000000000000000000
00000000001000000000000000000000000000000000100000
00000000001000000000100000000000000000000000000000
00000000000110000000100000000000000000000000000000
00000000000010000000000000000100000000000000000000
00010100000000000000001000000001000000000000000000
00000000000000000000001000000001000000000000000000
00000000000000001000000001000000100000000000000000
00000000000000001000000001000000010000000000000000
00000000000000001000000000000000011000000000000000
00000000100000000000000000000000000100000000000000
00000000000000000000000000000000000100000000000000
10000000000000000000000000000000000001000000000000
00000000000000000000000000000000000001000000000000
00000000000000001000000000000000000011110000000000
00000000010000000000000000000000000000111000000000
00000000000000000000000000000000000000011000000000
00000000000000000000000000000000000000010000000000
00000000000000000000000000000000000011000000010000
00000000010000000000000000000000000010000000010000
00000000000000000000000000000000000010000000010000
00000000000000000000000000000000000011100000010100
00000000000000000000000000000000000010000000010100
00000000000000000000000000000000000010000000010100
00000000000000000000000000000000000010000000010100
00000000000000000000000000000000000010000000010000
00000000000000000000000000000000000010000000000100
00000000000000000000000000000000000010000000001000
00000000000000000000000000000000000011000000000000
00000000000000000000000000000000000010000000000000
```

Tabla II.11: Matriz de conexiones del grafo de 50 vértices.

APÉNDICE III

Pruebas Problema de la Asignación Cuadrática

Este apéndice contiene los resultados completos de las pruebas del problema de la asignación cuadrática, desarrolladas con el fin de realizar el análisis práctico de la eficiencia del modelo neuronal ODE, respecto a los métodos actuales de optimización combinatoria más importantes: búsqueda local, temple simulado, red de Hopfield continua y máquina de Boltzmann.

Para efectuar las pruebas, se han realizado en lenguaje C las diferentes implementaciones del problema de la asignación cuadrática con cada uno de los métodos de comparación, utilizando en su desarrollo una metodología de diseño y programación modular basada en máquinas abstractas. Estas pruebas se han realizado sobre dos instancias del problema, correspondientes a la ubicación de 15 y 20 plantas de proceso respectivamente, ejecutando 20 veces cada instancia con cada método de resolución.

Los resultados se presentan agrupados en cinco apartados, correspondientes a cada una de las técnicas de resolución empleadas. Así mismo, al final del apéndice se proporcionan las matrices de cantidades y costes correspondientes a las dos instancias del problema de la asignación cuadrática que se han utilizado en las pruebas.

III.1 PRUEBAS PAC BÚSQUEDA LOCAL.

La tabla III.1 presenta los resultados de las pruebas, realizadas mediante el algoritmo de búsqueda local, del problema de la asignación cuadrática de 15 plantas de proceso sobre 15 lugares diferentes. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras III.1, III.2 y III.3):

I	COSTE	TIEMPO	SOLUCIÓN
1	2385000	4	12 8 3 6 11 0 7 10 2 9 13 14 1 4 5
2	2338337	6	2 6 9 3 4 10 14 0 8 1 13 5 7 12 11
3	2339972	7	7 0 9 3 11 6 14 10 2 8 5 4 1 12 13
4	2338763	12	6 0 12 10 7 14 1 9 3 4 13 11 2 8 5
5	2335091	10	7 12 2 3 11 0 8 6 10 4 5 9 1 14 13
6	2362288	7	9 10 2 1 4 6 5 0 3 11 14 13 8 12 7
7	2335938	10	2 7 10 6 14 3 8 12 0 4 5 13 11 9 1
8	2341697	11	6 12 10 2 9 0 1 7 3 13 4 11 8 14 5
9	2377300	5	11 3 2 14 12 7 0 6 10 1 5 4 8 9 13
10	2367063	5	11 7 6 12 2 9 8 3 10 14 13 4 0 1 5
11	2352881	5	9 3 1 8 12 7 6 0 10 11 14 13 4 5 2
12	2363664	6	8 3 14 12 6 9 2 10 0 13 5 4 11 7 1
13	2344916	4	7 10 9 3 14 2 0 6 1 8 5 13 11 12 4
14	2329090	8	8 6 7 9 2 0 11 10 3 13 5 4 14 12 1
15	2327863	13	10 12 2 7 1 6 8 0 3 13 4 11 9 14 5
16	2391011	4	1 0 12 5 2 8 10 3 7 6 13 4 11 14 9
17	2338102	6	13 0 7 10 11 12 8 6 3 9 5 4 2 14 1
18	2356605	7	2 3 0 12 11 7 8 10 6 14 13 4 1 9 5
19	2335938	13	2 7 10 6 14 3 8 12 0 4 5 13 11 9 1
20	2327863	10	10 12 2 7 1 6 8 0 3 13 4 11 9 14 5

Tabla III.1: Pruebas problema de la asignación cuadrática 15 con búsqueda local.

COSTE		TIEMPO	
Medio:	2 349 469.1 pts	Medio:	7.7 sg
Mejor:	2 327 863 pts	Mejor:	4 sg
Peor:	2 391 011 pts	Peor:	13 sg
Desviación:	19 141.6 pts	Desviación:	3.0 sg

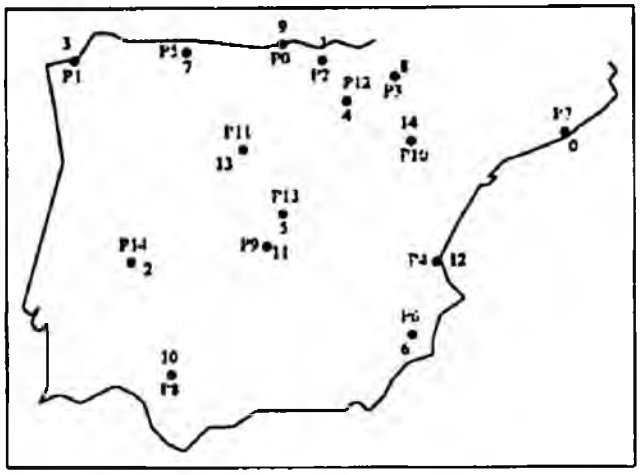

Figura III.1: Solución media a la asignación cuadrática 15 con búsqueda local: 2 352 881.

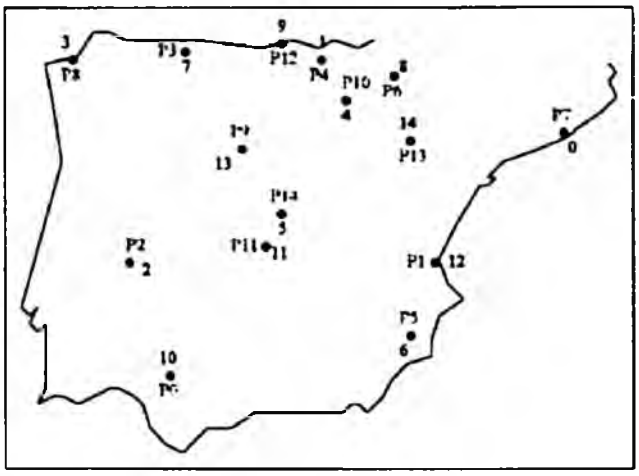

Figura III.2: Solución mejor a la asignación cuadrática 15 con búsqueda local: 2 327 863.

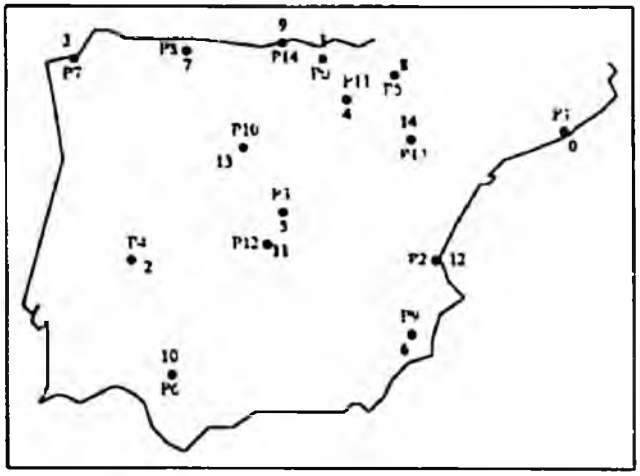

Figura III.3: Solución peor a la asignación cuadrática 15 con búsqueda local: 2 391 011.

La tabla siguiente presenta los resultados de las pruebas, realizadas mediante el algoritmo de búsqueda local, del problema de la asignación cuadrática de 20 plantas de proceso sobre 20 lugares diferentes. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras III.4, III.5 y III.6):

I	COSTE	TIEMPO	SOLUCIÓN
1	45565173	35	14 5 12 9 16 1 19 8 6 4 13 7 10 18 15 11 0 3 2 17
2	45358421	26	9 0 5 18 16 12 1 8 3 4 7 17 10 19 6 15 13 11 2 14
3	45030718	40	13 3 19 2 18 0 17 4 8 7 11 1 5 16 12 15 6 14 9 10
4	45358853	65	8 11 0 9 4 13 1 7 19 16 14 17 10 18 6 15 5 3 2 12
5	45871816	43	7 5 6 16 18 11 19 8 4 2 17 12 15 1 3 10 13 14 9 0
6	45473491	50	12 11 5 18 9 14 13 19 3 16 1 7 15 8 6 10 0 4 2 17
7	45444025	23	8 15 1 2 18 13 11 4 14 16 19 12 5 6 7 0 10 3 9 17
8	45923689	51	8 10 0 2 18 1 14 16 13 9 12 19 17 4 11 5 15 6 7 3
9	45501991	54	1 5 12 9 2 11 17 4 19 8 6 14 0 18 13 10 3 7 16 15
10	45250025	47	19 5 0 18 2 17 6 4 13 7 11 8 1 16 14 3 10 12 9 15
11	46040612	35	14 15 5 2 16 12 13 17 1 4 9 7 10 8 6 11 19 3 18 0
12	46009654	29	7 14 5 9 4 13 17 8 15 16 19 12 10 18 3 6 0 1 2 11
13	46050649	31	13 5 0 9 2 1 12 7 19 16 14 8 11 3 17 10 15 4 18 6
14	46244623	45	16 10 11 18 8 5 12 3 19 9 2 4 0 15 17 13 14 6 7 1
15	45278368	60	4 0 11 9 3 14 1 2 15 16 7 17 10 8 6 5 13 19 18 12
16	45813204	27	3 1 12 9 18 10 6 4 15 2 11 7 5 8 14 0 19 13 16 17
17	44940360	58	19 5 0 9 4 13 1 2 11 16 8 17 10 14 6 15 12 3 18 7
18	45658105	41	6 13 11 16 9 15 3 2 10 4 7 17 0 1 19 5 14 8 18 12
19	45597843	34	2 12 5 9 19 3 10 6 13 8 4 14 0 18 1 17 15 7 16 11
20	45352837	67	17 5 11 9 3 0 13 8 4 7 19 16 15 2 6 10 14 12 18 1

Tabla III.2: Pruebas problema de la asignación cuadrática 20 con búsqueda local.

COSTE		TIEMPO	
Medio:	45 588 222.9 pts	Medio:	43.1 sg
Mejor:	44 940 360 pts	Mejor:	23 sg
Peor:	46 244 623 pts	Peor:	67 sg
Desviación:	356 827.4 pts	Desviación:	13.2 sg

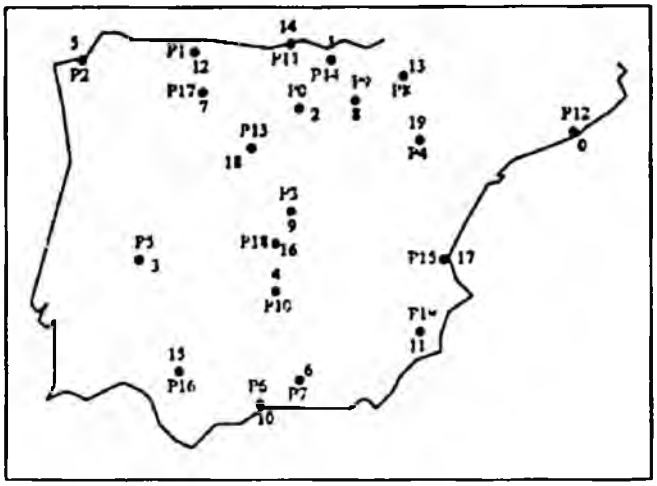

Figura III.4: Solución media a la asignación cuadrática 20 con b. local: 45 597 843.

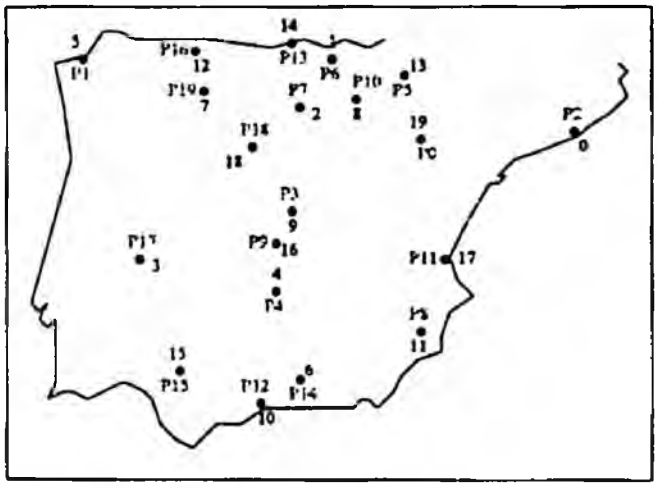

Figura III.5: Solución mejor a la asignación cuadrática 20 con b. local: 44 940 360.

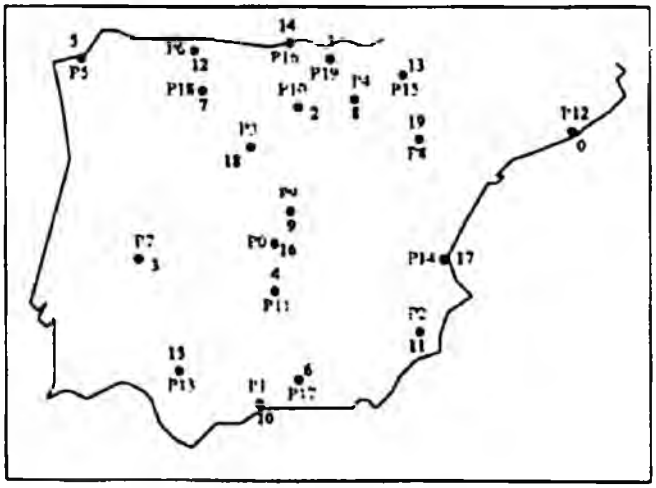

Figura III.6: Solución peor a la asignación cuadrática 20 con b. local: 46 244 623.

III.2 PRUEBAS PAC TEMPLE SIMULADO.

La tabla III.3 presenta los resultados de las pruebas, realizadas mediante el algoritmo del temple simulado, del problema de la asignación cuadrática de 15 plantas de proceso sobre 15 lugares diferentes. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras III.7, III.8 y III.9):

I	COSTE	TIEMPO	SOLUCIÓN
1	2326012	340	0 6 14 8 2 12 7 10 3 9 5 4 1 11 13
2	2345674	288	7 0 1 3 11 10 14 6 9 8 5 13 2 12 4
3	2323637	335	8 6 7 9 2 0 11 10 3 5 13 4 1 12 14
4	2330303	306	0 6 14 8 7 12 2 10 3 5 9 4 1 11 13
5	2352633	297	12 7 6 0 2 1 13 3 10 5 4 14 8 9 11
6	2311286	392	7 10 9 3 12 2 14 6 0 8 5 1 13 11 4
7	2352432	304	8 12 14 3 2 0 11 6 10 9 13 4 7 5 1
8	2311498	318	7 10 9 3 12 2 14 6 0 4 5 1 13 11 8
9	2330303	310	0 6 14 8 7 12 2 10 3 5 9 4 1 11 13
10	2335340	340	6 0 11 10 7 12 1 9 3 8 13 5 2 14 4
11	2331865	354	9 0 7 13 2 6 14 10 3 8 11 4 5 12 1
12	2311286	380	7 10 9 3 12 2 14 6 0 8 5 1 13 11 4
13	2349631	326	10 3 12 6 1 2 9 7 0 14 4 11 8 13 5
14	2311286	385	7 10 9 3 12 2 14 6 0 8 5 1 13 11 4
15	2331566	380	2 6 7 3 8 10 14 0 1 5 4 13 9 12 11
16	2326012	315	0 6 14 8 2 12 7 10 3 9 5 4 1 11 13
17	2327808	326	9 3 1 2 6 7 12 10 0 14 5 4 11 13 8
18	2311286	401	7 10 9 3 12 2 14 6 0 8 5 1 13 11 4
19	2330303	328	0 6 14 8 7 12 2 10 3 5 9 4 1 11 13
20	2323637	271	8 6 7 9 2 0 11 10 3 5 13 4 1 12 14

Tabla III.3: Pruebas problema de la asignación cuadrática 15 con temple simulado.

COSTE		TIEMPO	
Medio:	2 328 689.9 pts	Medio:	334.8 sg
Mejor:	2 311 286 pts	Mejor:	271 sg
Peor:	2 352 633 pts	Peor:	401 sg
Desviación:	13 587.3 pts	Desviación:	36.7 sg

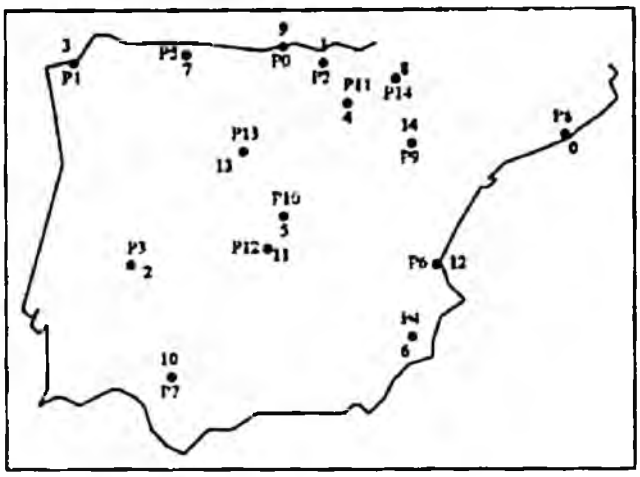

Figura III.7: Solución media a la asignación cuadrática 15 con temple sim.: 2 327 808.

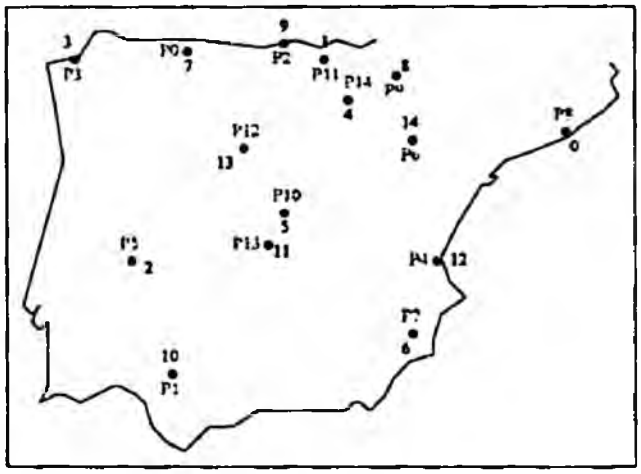

Figura III.8: Solución mejor a la asignación cuadrática 15 con temple sim.: 2 311 286.

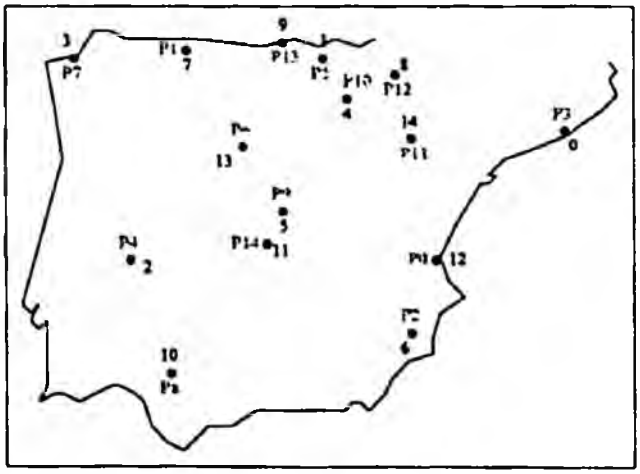

Figura III.9: Solución peor a la asignación cuadrática 15 con temple sim.: 2 352 633.

La tabla siguiente presenta los resultados de las pruebas, realizadas mediante el algoritmo del temple simulado, del problema de la asignación cuadrática de 20 plantas de proceso sobre 20 lugares diferentes. Así mismo, se ofrecen los gráficos de las soluciones media, mejor y peor obtenidas con este método (figuras III.10, III.11 y III.12):

I	COSTE	TIEMPO	SOLUCIÓN
1	44973544	1644	4 0 15 18 9 14 1 8 3 16 7 11 10 19 6 5 13 17 2 12
2	44940360	1708	19 5 0 9 4 13 1 2 11 16 8 17 10 14 6 15 12 3 18 7
3	45729521	1478	4 13 11 16 9 10 15 2 6 8 3 17 0 14 19 5 12 1 18 7
4	44973544	1699	4 0 15 18 9 14 1 8 3 16 7 11 10 19 6 5 13 17 2 12
5	45622601	1641	1 10 5 2 19 12 7 16 13 8 9 14 0 3 17 11 15 4 18 6
6	45175308	1757	13 15 1 2 18 0 17 11 8 9 19 14 12 4 7 5 10 3 16 6
7	44940360	1762	19 5 0 9 4 13 1 2 11 16 8 17 10 14 6 15 12 3 18 7
8	45313029	1878	13 17 0 18 2 11 10 4 8 1 16 19 12 3 14 5 15 7 9 6
9	45083583	1654	14 0 12 18 2 3 4 11 7 8 16 1 5 17 13 15 6 19 9 10
10	45193406	1602	8 17 0 18 2 3 6 4 7 1 16 19 12 11 14 5 10 13 9 15
11	45003206	1690	19 5 0 2 18 17 4 16 8 7 11 13 1 3 14 10 15 12 9 6
12	45372981	1632	1 5 12 18 2 7 15 16 8 19 4 14 0 3 13 11 10 17 9 6
13	45030718	1755	13 3 19 2 18 0 17 4 8 7 11 1 5 16 12 15 6 14 9 10
14	45455877	1714	11 19 0 9 16 13 1 8 6 4 14 17 10 18 15 5 7 3 2 12
15	45170710	1802	9 0 1i 18 4 5 12 8 3 16 7 17 10 13 6 15 1 19 2 14
16	45083583	1568	14 0 12 18 2 3 4 11 7 8 16 1 5 17 13 15 6 19 9 10
17	45164530	1631	19 3 0 18 2 17 6 4 8 14 11 13 12 16 7 5 10 1 9 15
18	44940360	1748	19 5 0 9 4 13 1 2 11 16 8 17 10 14 6 15 12 3 18 7
19	44973544	1660	4 0 15 18 9 14 1 8 3 16 7 11 10 19 6 5 13 17 2 12
20	45397095	1472	17 7 0 9 4 19 13 1 11 16 8 6 10 18 15 5 14 3 2 12

Tabla III.4: Pruebas problema de la asignación cuadrática 20 con temple simulado.

COSTE		TIEMPO	
Medio:	45 176 893.0 pts	Medio:	1674.8 sg
Mejor:	44 940 360 pts	Mejor:	1472 sg
Peor:	45 729 521 pts	Peor:	1878 sg
Desviación:	234 932.0 pts	Desviación:	100.0 sg

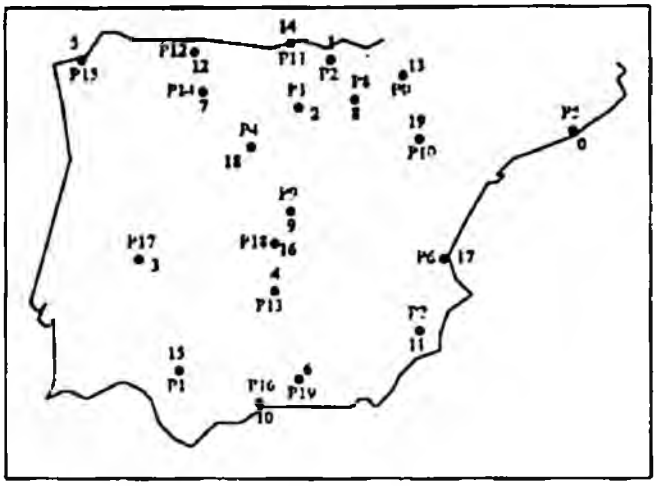

Figura III.10: Solución media a la asignación cuadrática 20 con temple sim.: 45 175 308.

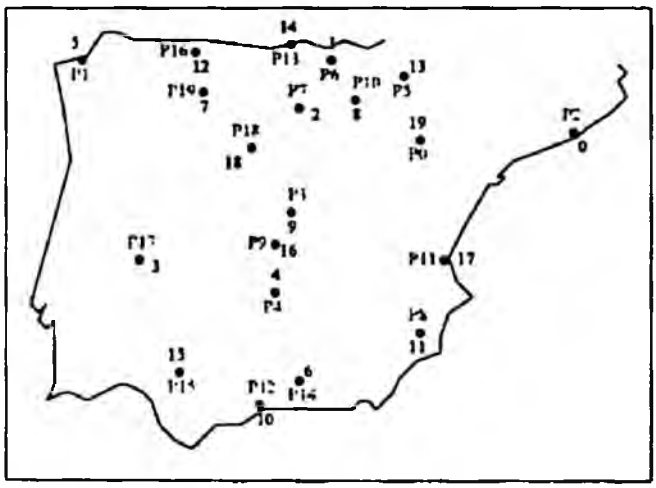

Figura III.11: Solución mejor a la asignación cuadrática 20 con temple sim.: 44 940 360.

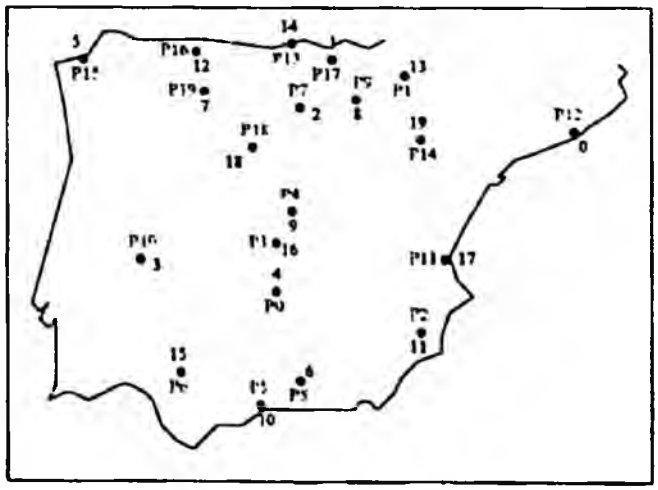

Figura III.12: Solución peor a la asignación cuadrática 20 con temple sim.: 45 729 521.

III.3 PRUEBAS PAC RED DE HOPFIELD CONTINUA.

La tabla III.5 presenta los resultados de las pruebas, realizadas mediante la red de Hopfield continua, del problema de la asignación cuadrática de 15 plantas de proceso sobre 15 lugares diferentes. Así mismo, se ofrecen los gráficos de las soluciones media, mejor y peor obtenidas con este método (figuras III.13, III.14 y III.15):

I	COSTE	TIEMPO	SOLUCIÓN
1	2566488	1115	7 0 10 1 14 6 3 2 13 11 4 9 5 12 8
2	2708594	914	0 2 1 3 4 8 7 11 9 14 13 5 10 12 6
3	2670276	1059	3 13 12 14 8 2 9 11 0 1 4 10 6 7 5
4	2713249	992	13 11 10 8 9 4 6 3 2 1 14 7 5 12 0
5	2646929	959	6 2 1 10 7 8 11 13 0 12 9 5 14 4 3
6	2661311	1062	14 11 0 13 8 3 2 12 7 6 5 4 10 1 9
7	2666372	962	6 12 10 5 0 14 13 2 11 8 1 9 3 4 7
8	2499365	1050	14 7 5 0 10 3 2 11 6 9 12 4 13 8 1
9	2636164	1175	0 12 10 2 11 5 8 7 6 3 9 4 14 13 1
10	2592451	1067	0 10 1 6 12 13 5 11 4 2 7 8 9 3 14
11	2595144	1015	14 11 10 9 0 2 1 3 7 5 4 12 6 13 8
12	2602064	881	14 12 4 6 7 13 5 2 10 1 8 11 3 9 0
13	2648201	1122	6 13 14 7 5 3 2 0 1 9 8 4 10 12 11
14	2633195	1091	14 3 13 0 8 1 12 9 7 4 10 11 2 5 6
15	2623654	1240	4 9 2 6 10 8 7 13 12 3 14 0 11 1 5
16	2636989	1143	6 14 12 11 10 13 2 7 0 3 1 8 4 9 5
17	2620332	945	3 10 13 0 1 11 14 7 12 6 2 8 5 4 9
18	2695378	960	8 14 11 10 6 3 4 1 7 12 2 5 9 13 0
19	2498192	1010	0 14 2 10 8 4 7 1 3 6 5 13 11 9 12
20	2605048	1019	12 13 9 5 2 6 10 3 8 4 14 11 0 1 7

Tabla III.5: Pruebas problema de la asignación cuadrática 15 con la red CH.

COSTE		TIEMPO	
Medio:	2 625 969.8 pts	Medio:	1039.1 sg
Mejor:	2 498 192 pts	Mejor:	881 sg
Peor:	2 713 249 pts	Peor:	1240 sg
Desviación:	58 188.2 pts	Desviación:	91.5 sg

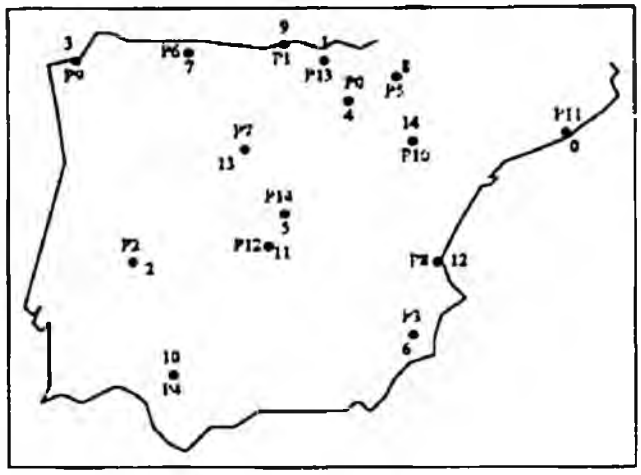

Figura III.13: Solución media a la asignación cuadrática 15 con la red CH: 2 623 654.

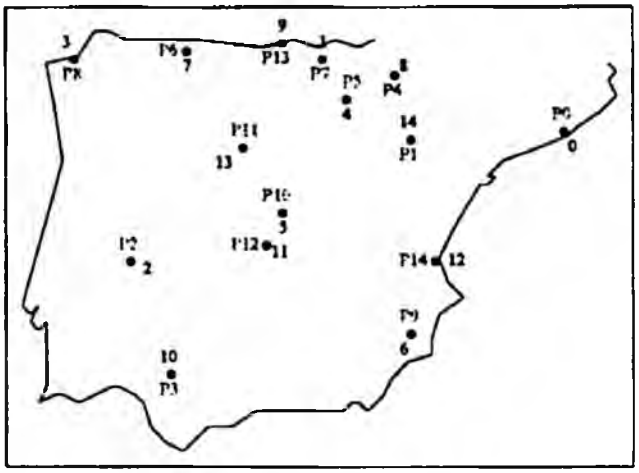

Figura III.14: Solución mejor a la asignación cuadrática 15 con la red CH: 2 498 192.

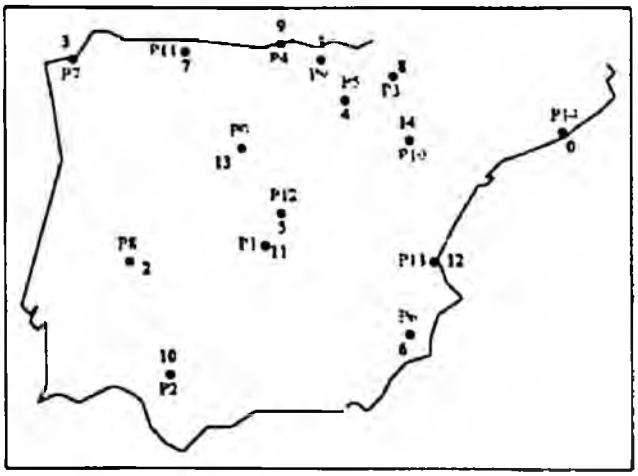

Figura III.15: Solución peor a la asignación cuadrática 15 con la red CH: 2 713 249.

La tabla siguiente presenta los resultados de las pruebas, realizadas mediante la red de Hopfield continua, del problema de la asignación cuadrática de 20 plantas de proceso sobre 20 lugares diferentes. Así mismo, se ofrecen los gráficos de las soluciones media, mejor y peor obtenidas con este método (figuras III.16, III.17 y III.18):

I	COSTE	TIEMPO	SOLUCIÓN
1	53348276	4968	4 18 5 3 2 0 17 15 13 6 10 11 19 1 9 16 14 8 7 12
2	50575850	5198	3 5 13 9 14 15 6 11 19 4 1 12 0 8 17 10 16 2 18 7
3	50741736	5291	12 16 0 13 5 3 8 9 15 19 2 1 11 14 10 6 7 17 18 4
4	50384595	5897	19 10 17 8 4 16 18 1 2 13 9 12 5 7 11 0 6 14 3 15
5	51975236	4053	16 10 15 3 19 17 11 2 6 18 9 5 14 1 13 4 12 8 0 7
6	51604518	5581	14 9 0 13 10 6 18 2 17 19 15 4 11 5 16 8 7 3 1 12
7	52709371	5088	12 9 15 16 4 0 10 18 5 6 14 11 1 2 13 17 7 19 3 8
8	49875875	5491	8 17 6 3 9 2 13 18 19 7 14 4 15 16 10 12 0 1 11 5
9	51856708	4372	5 10 11 18 2 3 14 8 15 6 1 4 7 16 13 9 17 19 12 0
10	50429528	5378	16 10 0 8 5 14 9 1 7 4 18 6 12 3 17 13 15 11 2 19
11	50181187	5206	8 18 15 13 9 10 5 7 1 14 4 3 16 12 6 11 17 19 2 0
12	51001517	5036	3 17 11 7 2 4 12 13 9 10 18 6 0 8 1 16 5 19 15 14
13	50783058	5509	4 7 14 8 12 1 19 13 15 9 16 5 0 17 18 6 11 10 2 3
14	50703618	4791	1 15 6 9 4 8 7 16 13 2 14 17 19 12 18 0 3 11 5 10
15	51102175	4632	19 6 3 16 7 0 14 8 2 17 9 11 5 12 1 18 13 10 15 4
16	51846674	4171	10 5 13 14 4 6 11 9 16 17 8 19 7 1 0 15 2 18 3 12
17	48774797	5999	13 18 10 2 8 0 11 7 1 16 17 14 12 6 5 3 15 9 19 4
18	51325242	6224	7 10 9 8 14 1 13 19 4 17 11 2 12 16 18 5 15 6 0 3
19	50646089	5257	1 5 17 13 18 9 11 7 4 14 15 0 19 16 8 6 10 3 2 12
20	49774026	5191	11 4 3 2 13 12 6 16 19 0 18 9 14 10 8 1 17 5 7 15

Tabla III.6: Pruebas problema de la asignación cuadrática 20 con la red CH.

COSTE		TIEMPO	
Medio:	50 982 003.8 pts	Medio:	5166.7 sg
Mejor:	48 774 797 pts	Mejor:	4053 sg
Peor:	53 348 276 pts	Peor:	6224 sg
Desviación:	1 049 135.1 pts	Desviación:	569.5 sg

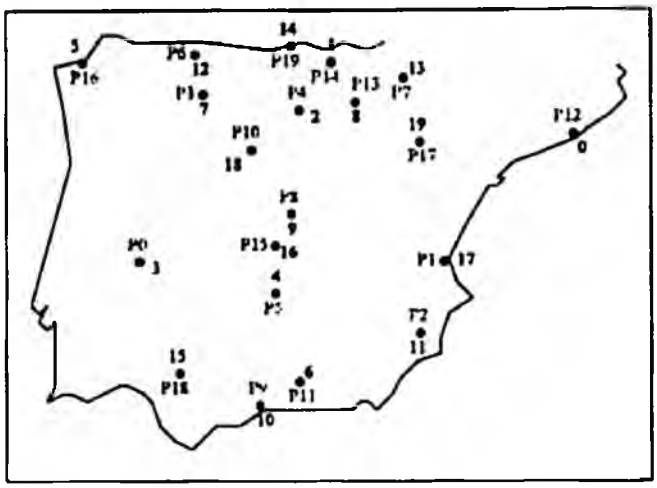

Figura III.16: Solución media a la asignación cuadrática 20 con la red CH: 51 001 517.

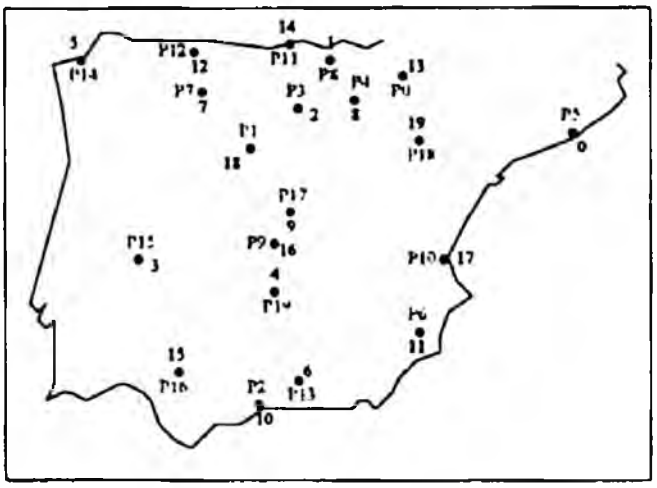

Figura III.17: Solución mejor a la asignación cuadrática 20 con la red CH: 48 774 797.

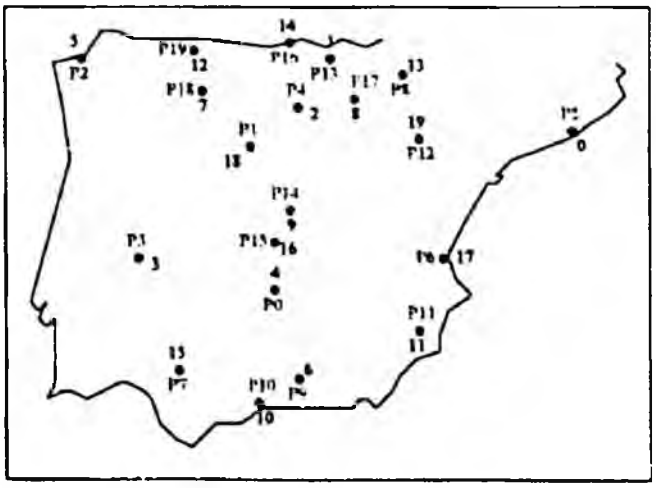

Figura III.18: Solución peor a la asignación cuadrática 20 con la red CH: 53 348 276.

III.4 PRUEBAS PAC MÁQUINA DE BOLTZMANN.

La tabla III.7 presenta los resultados de las pruebas, realizadas mediante la máquina de Boltzmann, del problema de la asignación cuadrática de 15 plantas de proceso sobre 15 lugares diferentes. Así mismo, se ofrecen los gráficos de las soluciones media, mejor y peor obtenidas con este método (figuras III.19, III.20 y III.21):

I	COSTE	TIEMPO	SOLUCIÓN
1	2700506	63	11 2 14 0 12 7 5 4 9 13 8 3 1 6 10
2	2786950	63	1 11 12 4 2 6 13 7 8 10 9 3 5 14 0
3	2691189	60	12 6 9 3 4 14 8 13 11 5 2 10 7 1 0
4	2579018	65	12 11 3 13 1 6 5 8 9 2 14 7 10 0 4
5	2695630	61	5 12 9 8 14 7 2 13 3 11 0 4 1 6 10
6	2734369	59	5 14 8 1 3 12 11 9 2 4 10 13 0 7 6
7	2638459	66	8 5 12 0 2 1 13 7 6 11 10 9 4 3 14
8	2649888	64	10 9 2 5 6 14 13 0 7 3 8 11 1 12 4
9	2614085	61	4 13 10 12 1 7 5 8 2 6 0 11 9 3 14
10	2632578	62	10 7 3 12 1 5 13 9 6 0 11 4 8 14 2
11	2659719	64	7 6 2 8 3 14 4 10 1 13 0 9 11 12 5
12	2514073	65	2 11 3 6 8 12 1 10 0 7 5 14 4 13 9
13	2794526	63	5 2 6 8 12 4 9 11 13 3 10 0 1 7 14
14	2665660	63	5 12 1 8 10 14 3 4 6 7 0 9 11 13 2
15	2636176	62	9 11 8 5 7 3 4 0 10 6 13 14 12 1 2
16	2553115	63	11 12 9 14 5 2 3 6 10 4 7 8 1 13 0
17	2744382	63	14 4 2 0 11 7 8 10 1 5 3 6 12 13 9
18	2713533	60	6 7 3 5 10 8 13 12 11 0 14 1 9 2 4
19	2660180	64	5 10 4 13 11 8 7 6 3 12 2 9 0 1 14
20	2624951	64	9 2 4 1 5 0 12 3 10 13 6 14 8 7 11

Tabla III.7: Pruebas problema de la asignación cuadrática 15 con la red BM.

COSTE		TIEMPO	
Medio:	2 664 449.4 pts	Medio:	62.8 sg
Mejor:	2 514 073 pts	Mejor:	59 sg
Peor:	2 794 526 pts	Peor:	66 sg
Desviación:	71 726.5 pts	Desviación:	1.8 sg

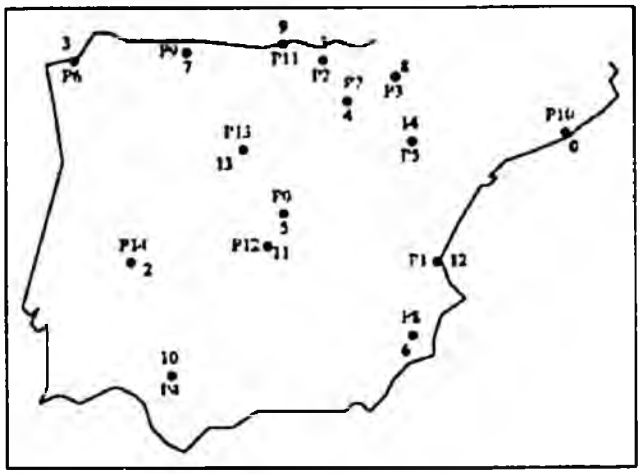

Figura III.19: Solución media a la asignación cuadrática 15 con la red BM: 2 665 660.

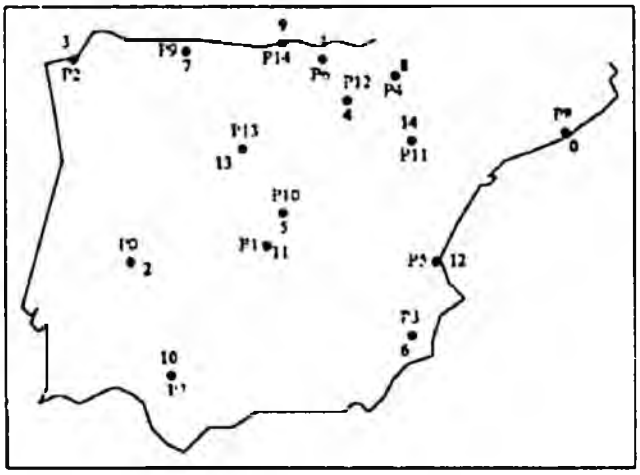

Figura III.20: Solución mejor a la asignación cuadrática 15 con la red BM: 2 514 073.

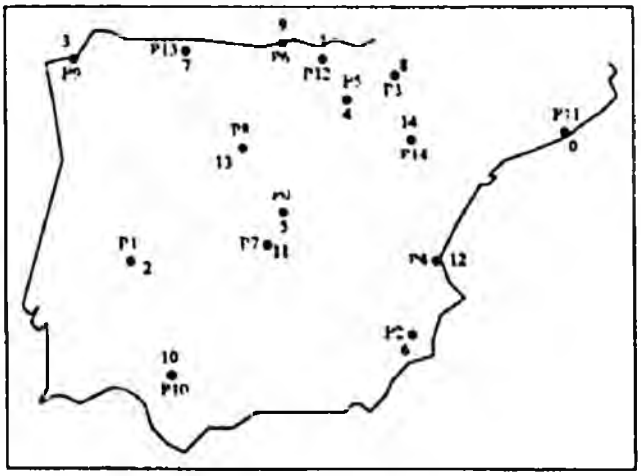

Figura III.21: Solución peor a la asignación cuadrática 15 con la red BM: 2 794 526.

La tabla siguiente presenta los resultados de las pruebas, realizadas mediante la máquina de Boltzmann, del problema de la asignación cuadrática de 20 plantas de proceso sobre 20 lugares diferentes. Así mismo, se ofrecen los gráficos de las soluciones media, mejor y peor obtenidas con este método (figuras III.22, III.23 y III.24):

I	COSTE	TIEMPO	SOLUCIÓN
1	52344607	240	16 0 14 17 15 13 9 2 18 3 10 7 11 12 6 8 5 1 4 19
2	50733491	223	16 10 6 1 2 15 19 4 18 17 11 9 12 0 14 3 8 13 7 5
3	53200723	240	8 18 15 13 6 2 7 17 5 16 12 4 14 9 3 1 19 10 0 11
4	52901293	235	12 18 4 0 3 9 2 14 5 6 17 16 11 13 7 15 1 19 8 10
5	54586071	245	18 16 2 10 19 1 15 14 0 7 3 9 4 6 12 11 13 8 5 17
6	53868838	241	13 8 5 10 12 11 16 1 17 14 15 19 0 6 4 9 7 18 3 2
7	52374552	250	1 9 13 17 3 12 11 6 0 7 15 4 2 19 14 16 5 8 18 10
8	53802360	223	11 16 15 17 0 4 19 3 10 5 6 8 18 13 14 1 2 9 7 12
9	49780901	242	10 7 6 8 4 5 3 14 12 18 15 9 0 16 11 17 1 2 19 13
10	53590745	244	11 0 1 5 9 15 19 12 6 18 3 7 2 13 10 17 16 4 8 14
11	51131459	231	15 14 3 9 2 8 7 16 4 0 12 5 19 13 1 6 11 17 18 10
12	52295027	240	11 4 6 14 15 12 3 17 10 9 18 1 16 0 2 13 7 8 19 5
13	50670807	231	9 11 13 16 8 3 15 4 10 1 19 6 14 12 17 18 5 7 2 0
14	52173071	243	5 8 12 0 3 6 4 17 13 19 16 9 14 7 18 1 15 2 10 11
15	52721696	226	0 10 6 7 5 17 4 11 2 15 14 18 12 8 13 9 1 19 16 3
16	52700509	239	13 2 4 19 12 6 8 17 0 10 16 14 1 18 3 11 9 5 7 15
17	52360078	229	18 1 10 13 8 16 4 11 2 5 3 19 6 15 12 7 9 17 0 14
18	53624066	229	17 5 1 3 13 4 16 0 18 7 6 11 8 15 10 9 2 19 14 12
19	52322921	239	12 10 19 7 11 6 17 3 1 13 0 9 14 8 16 18 2 15 4 5
20	51600783	251	9 16 6 1 2 19 15 8 5 13 11 10 14 3 7 17 4 18 12 0

Tabla III.8: Pruebas problema de la asignación cuadrática 20 con la red BM.

COSTE		TIEMPO	
Medio:	52 439 199.9 pts	Medio:	237.1 sg
Mejor:	49 780 901 pts	Mejor:	223 sg
Peor:	54 586 071 pts	Peor:	251 sg
Desviación:	1 210 121.4 pts	Desviación:	8.3 sg

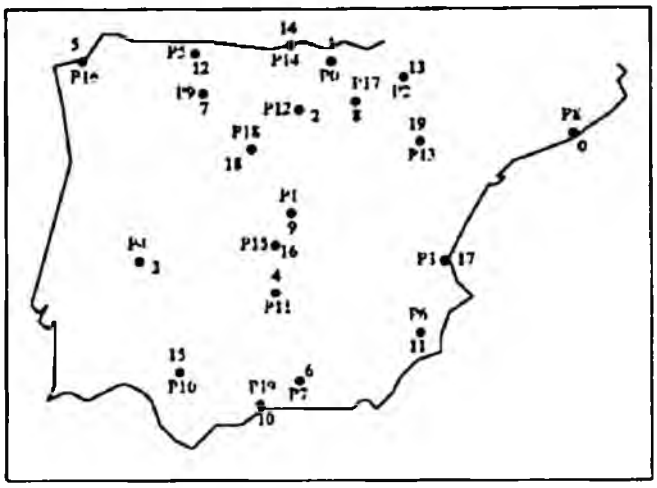

Figura III.22: Solución media a la asignación cuadrática 20 con la red BM: 52 374 552.

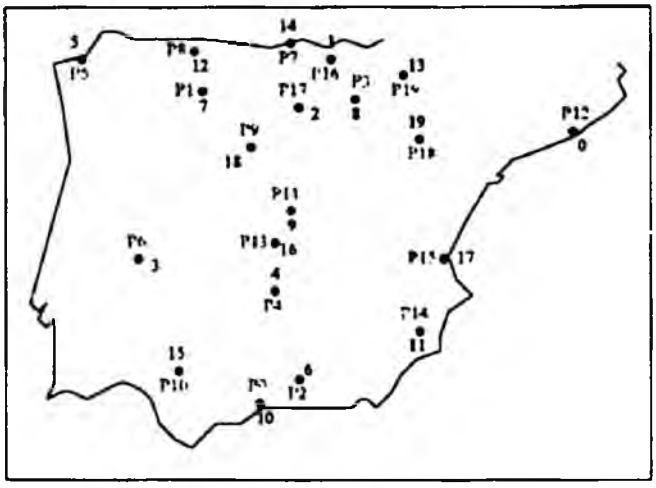

Figura III.23: Solución mejor a la asignación cuadrática 20 con la red BM: 49 780 901.

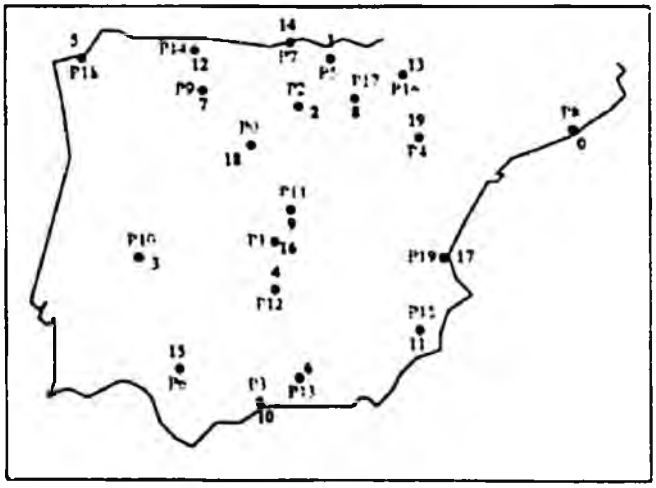

Figura III.24: Solución peor a la asignación cuadrática 20 con la red BM: 54 586 071.

III.5 PRUEBAS PAC OPTIMIZADOR DISCRETO ESTOCÁSTICO.

La tabla III.9 presenta los resultados de las pruebas, realizadas mediante la red ODE, del problema de la asignación cuadrática de 15 plantas de proceso sobre 15 lugares diferentes. Así mismo, se ofrecen los gráficos de las soluciones media, mejor y peor obtenidas con este método (figuras III.25, III.26 y III.27):

I	COSTE	TIEMPO	SOLUCIÓN
1	2333337	958	6 0 10 2 9 12 1 7 3 8 13 5 11 14 4
2	2338382	951	7 8 2 3 12 1 6 0 10 11 4 13 9 14 5
3	2341138	963	2 6 8 1 9 10 0 3 7 14 13 5 4 12 11
4	2331639	915	0 6 7 9 2 12 11 10 3 8 13 4 1 5 14
5	2337392	840	12 10 0 14 7 6 9 2 3 1 5 4 8 11 13
6	2311286	876	7 10 9 3 12 2 14 6 0 8 5 1 13 11 4
7	2332318	926	2 3 10 6 14 7 8 0 12 13 4 11 1 9 5
8	2331159	898	13 0 9 3 2 6 14 10 7 8 11 4 5 12 1
9	2360801	941	8 10 6 1 7 2 5 3 0 12 13 4 9 11 14
10	2311286	1104	7 10 9 3 12 2 14 6 0 8 5 1 13 11 4
11	2328222	802	11 6 10 2 1 12 8 0 3 7 4 5 9 14 13
12	2325504	774	2 6 9 3 14 10 8 12 0 1 5 13 7 11 4
13	2331566	910	2 6 7 3 8 10 14 0 1 5 4 13 9 12 11
14	2349631	776	10 3 12 6 1 2 9 7 0 14 4 11 8 13 5
15	2311286	1024	7 10 9 3 12 2 14 6 0 8 5 1 13 11 4
16	2332407	920	9 3 12 0 11 7 2 10 6 8 5 4 14 13 1
17	2311286	1051	7 10 9 3 12 2 14 6 0 8 5 1 13 11 4
18	2323637	938	8 6 7 9 2 0 11 10 3 5 13 4 1 12 14
19	2331639	952	0 6 7 9 2 12 11 10 3 8 13 4 1 5 14
20	2327863	986	10 12 2 7 1 6 8 0 3 13 4 11 9 14 5

Tabla III.9: Pruebas problema de la asignación cuadrática 15 con la red ODE.

COSTE		TIEMPO	
Medio:	2 330 089.0 pts	Medio:	925.3 sg
Mejor:	2 311 286 pts	Mejor:	1104 sg
Peor:	2 360 801 pts	Peor:	774 sg
Desviación:	12 739.6 pts	Desviación:	84.8 sg

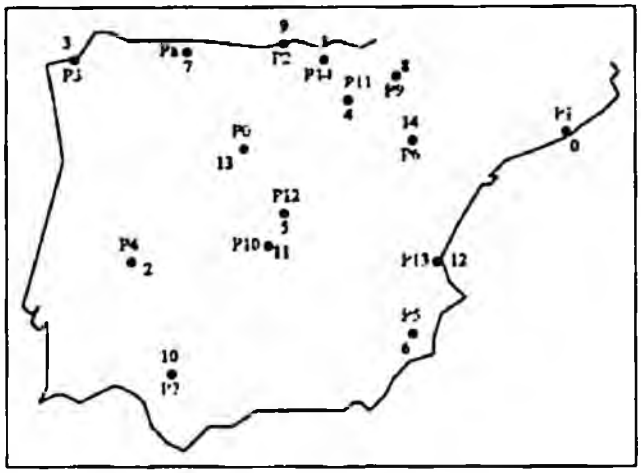

Figura III.25: Solución media a la asignación cuadrática 15 con la red ODE: 2 331 159.

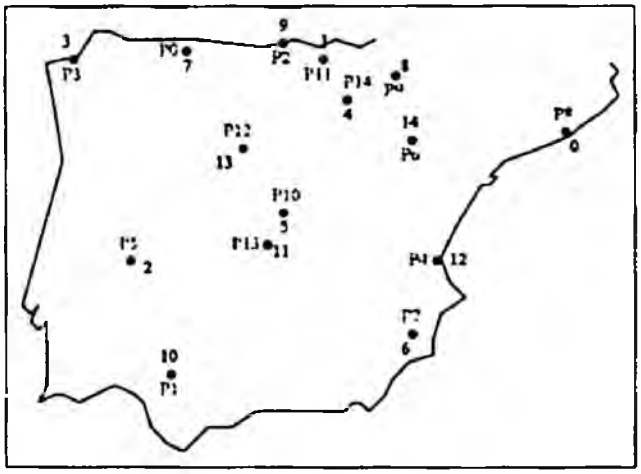

Figura III.26: Solución mejor a la asignación cuadrática 15 con la red ODE: 2 311 286.

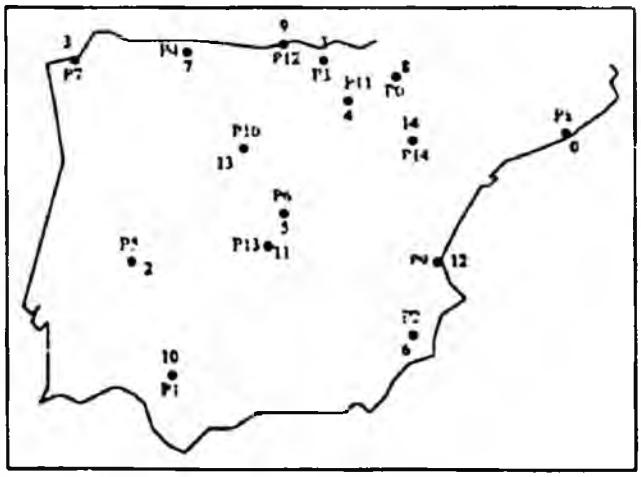

Figura III.27: Solución peor a la asignación cuadrática 15 con la red ODE: 2 360 801.

La tabla siguiente presenta los resultados de las pruebas, realizadas mediante la red ODE, del problema de la asignación cuadrática de 20 plantas de proceso sobre 20 lugares diferentes. Así mismo, se ofrecen los gráficos de las soluciones media, mejor y peor obtenidas con este método (figuras III.28, III.29 y III.30):

I	COSTE	TIEMPO	SOLUCIÓN
1	45175217	4385	13 12 19 9 18 0 17 4 8 2 11 1 5 3 14 10 15 7 16 6
2	44973544	3902	4 0 15 18 9 14 1 8 3 16 7 11 10 19 6 5 13 17 2 12
3	44940360	3986	19 5 0 9 4 13 1 2 11 16 8 17 10 14 6 15 12 3 18 7
4	45003206	4418	19 5 0 2 18 17 4 16 8 7 11 13 1 3 14 10 15 12 9 6
5	45206871	4442	1 0 14 18 7 3 6 16 12 2 4 13 5 17 8 15 11 19 9 10
6	45139248	4311	2 0 12 18 9 3 4 17 7 8 6 1 5 19 14 15 11 13 16 10
7	45061087	4446	1 0 12 18 2 16 6 11 7 8 4 14 5 17 13 3 10 19 9 15
8	45314706	3924	1 5 17 9 2 4 3 18 19 14 6 8 0 7 13 11 15 12 16 10
9	44940360	4213	19 5 0 9 4 13 1 2 11 16 8 17 10 14 6 15 12 3 18 7
10	45170710	4633	9 0 11 18 4 5 12 8 3 16 7 17 10 13 6 15 1 19 2 14
11	45003206	4334	19 5 0 2 18 17 4 16 8 7 11 13 1 3 14 10 15 12 9 6
12	45444316	3967	17 1 10 9 18 0 19 13 11 16 7 6 15 8 4 5 14 3 2 12
13	45030718	4580	13 3 19 2 18 0 17 4 8 7 11 1 5 16 12 15 6 14 9 10
14	45190509	4408	19 1 0 9 18 17 10 4 8 2 11 13 12 3 14 5 15 7 16 6
15	45432024	3964	2 0 13 9 7 10 6 4 14 18 3 8 12 17 1 5 11 19 16 15
16	45108306	4098	14 0 12 18 9 3 4 17 7 2 6 1 5 19 8 15 11 13 16 10
17	45030718	4602	13 3 19 2 18 0 17 4 8 7 11 1 5 16 12 15 6 14 9 10
18	45003206	4570	19 5 0 2 18 17 4 16 8 7 11 13 1 3 14 10 15 12 9 6
19	45359993	3643	18 5 12 9 16 14 13 8 3 4 1 7 10 19 15 6 0 11 2 17
20	45340667	4271	7 1 12 18 19 5 15 16 2 8 4 14 0 3 13 11 10 17 9 6

Tabla III.10: Pruebas problema de la asignación cuadrática 20 con la red ODE.

COSTE		TIEMPO	
Medio:	45 143 448.6 pts	Medio:	4254.9 sg
Mejor:	44 940 360 pts	Mejor:	3643 sg
Peor:	45 444 316 pts	Peor:	4633 sg
Desviación:	162 337.5 pts	Desviación:	280.2 sg

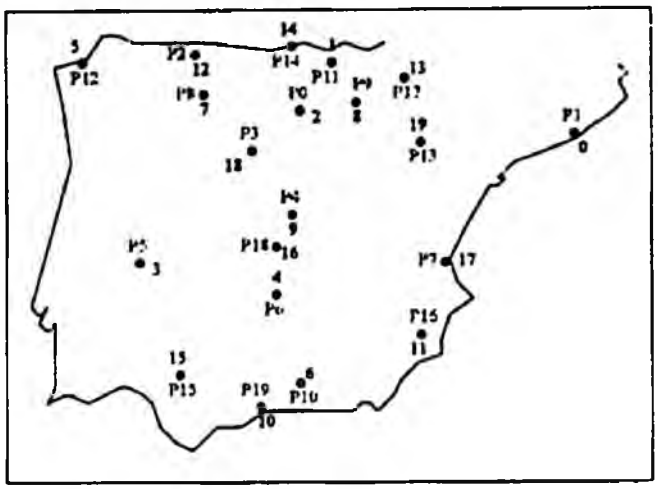

Figura III.28: Solución media a la asignación cuadrática 20 con la red ODE: 45 139 248.

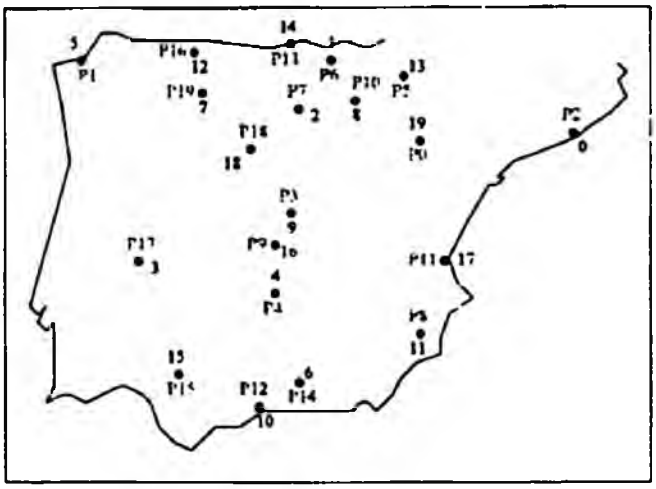

Figura III.29: Solución mejor a la asignación cuadrática 20 con la red ODE: 44 940 360.

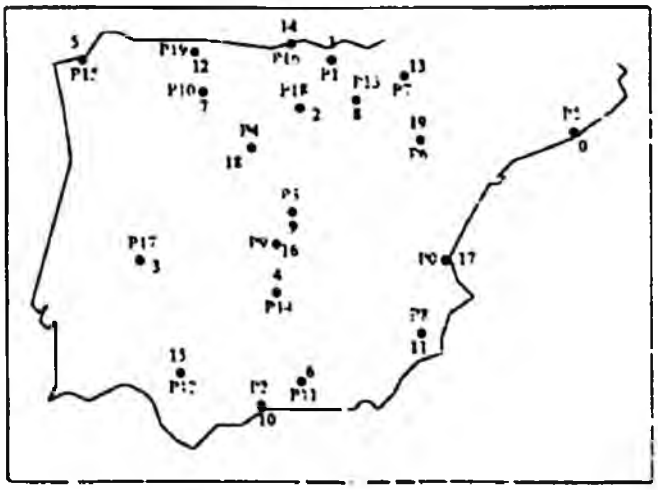

Figura III.30: Solución peor a la asignación cuadrática 20 con la red ODE: 45 444 316.

III.6 DATOS UTILIZADOS EN LAS PRUEBAS.

A continuación se ofrecen las matrices de cantidades y costes, utilizadas en las pruebas del problema de la asignación cuadrática:

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14
P0	0	33	67	60	45	80	10	34	23	11	57	99	40	2	89
P1	33	0	56	30	9	75	27	66	3	60	27	44	12	98	45
P2	67	56	0	55	34	20	34	25	50	73	64	64	36	25	22
P3	60	30	55	0	50	9	10	40	38	47	47	75	61	55	33
P4	45	9	34	50	0	36	94	91	88	25	74	7	81	37	72
P5	80	75	20	9	36	0	1	51	7	38	38	71	52	93	25
P6	10	27	34	10	94	1	0	52	14	94	34	90	41	94	39
P7	34	66	25	40	91	51	52	0	63	7	89	10	3	52	4
P8	23	3	50	38	88	7	14	63	0	67	45	78	1	25	75
P9	11	60	73	47	25	38	94	7	67	0	37	64	73	48	97
P10	57	27	64	47	74	38	34	89	45	37	0	58	97	89	63
P11	99	44	64	75	7	71	90	10	78	64	58	0	72	58	96
P12	40	12	36	61	81	52	41	3	1	73	97	72	0	45	26
P13	2	98	25	55	37	93	94	52	25	48	89	58	45	0	67
P14	89	45	22	33	72	25	39	4	75	97	63	96	26	67	0

Tabla III.11: Matriz de cantidades entre 15 plantas de proceso.

	Bar.	Bil	Cac.	Coru	Log	Mad	Mur	Ovi	Pam	Sant	Sev	Tol	Val	Vall	Zar
Bar.	0	620	918	1118	468	621	590	902	437	693	1046	692	349	663	296
Bil	620	0	605	644	152	395	796	304	159	108	933	466	633	280	324
Cac	918	605	0	683	595	297	654	525	650	573	264	264	636	325	622
Coru	1118	644	683	0	650	609	1010	340	738	547	947	675	961	455	833
Log	468	152	595	650	0	336	694	391	88	225	874	407	481	237	172
Mad	621	395	297	609	336	0	401	451	407	393	538	71	352	193	325
Mur	590	796	654	1010	694	401	0	852	714	794	534	390	241	594	539
Ovi	902	304	525	340	391	451	852	0	463	207	789	510	803	252	604
Pam	437	159	650	738	88	407	714	463	0	267	915	478	501	325	175
Sant.	693	108	573	547	225	393	794	207	267	0	837	464	673	248	397
Sev	1046	933	264	947	874	538	534	789	945	837	0	458	697	589	863
Tol	692	466	264	675	407	71	390	510	478	464	458	0	372	258	396
Val	349	633	636	961	481	352	241	803	501	673	697	372	0	545	326
Vall	663	280	325	455	237	193	594	252	325	248	589	258	545	0	367
Zar	296	324	622	833	172	325	539	604	175	397	863	396	326	367	0

Tabla III.12: Matriz de costes entre 15 ciudades españolas.

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19
P0	0	51	657	770	598	765	501	754	918	927	714	989	140	643	769	40	236	427	170	156
P1	51	0	404	369	658	435	35	942	105	569	301	813	42	588	876	127	430	501	583	474
P2	657	404	0	884	242	521	72	163	507	215	791	841	464	829	443	336	261	197	468	140
P3	770	369	884	0	714	997	877	302	946	994	543	809	878	753	597	993	826	457	910	646
P4	598	658	242	714	0	71	496	512	921	993	938	788	726	334	758	616	585	617	921	271
P5	765	435	521	997	71	0	721	601	658	61	807	446	55	164	288	81	27	417	952	628
P6	501	35	72	877	496	721	0	906	889	642	924	75	60	946	112	45	895	43	675	439
P7	754	942	163	302	512	601	906	0	122	643	992	420	799	960	169	204	734	493	590	868
P8	918	105	507	946	921	658	889	122	0	853	97	862	620	26	711	272	81	105	429	734
P9	927	569	215	994	993	61	642	643	853	0	247	298	510	456	883	340	182	999	904	371
P10	714	301	791	543	938	807	924	992	97	247	0	416	166	81	435	889	630	539	747	784
P11	989	813	841	809	788	446	75	420	862	298	416	0	421	578	689	57	170	858	768	63
P12	140	42	464	878	726	55	60	799	620	510	166	421	0	124	958	733	121	471	553	29
P13	643	588	829	753	334	164	946	960	26	456	81	578	124	0	934	157	944	778	875	688
P14	769	876	443	597	758	288	112	169	711	883	435	689	958	934	0	682	116	492	91	228
P15	40	127	336	993	616	81	45	204	272	340	889	57	733	157	682	0	470	206	642	970
P16	236	430	261	826	585	27	895	734	81	182	630	170	121	944	116	470	0	786	600	828
P17	427	501	197	457	617	417	43	493	105	999	539	858	471	778	492	206	786	0	401	317
P18	170	583	468	910	921	952	675	590	429	904	747	768	553	875	91	642	600	401	0	477
P19	156	474	140	646	271	628	439	868	734	371	784	63	29	688	228	970	828	317	477	0

Tabla III.13: Matriz de cantidades entre 20 plantas de proceso.

	Bar.	Bil.	Bur.	Cac.	Ciu.	Coru.	Gra.	Leo.	Log.	Mad.	Mal.	Mur.	Ovi.	Pam.	Sant.	Sev.	Tol.	Val.	Vall.	Zar.
Bar.	0	620	583	918	811	1118	868	784	468	621	997	590	902	437	693	1046	692	349	663	296
Bil.	620	0	158	605	585	644	829	359	152	395	939	796	304	159	108	933	466	633	280	324
Bur.	583	158	0	447	427	535	671	201	115	237	781	638	322	203	156	775	308	517	122	287
Cac.	918	605	447	0	324	683	485	407	595	297	506	654	525	650	573	264	264	636	325	622
Ciu.	811	585	427	324	0	799	278	511	526	190	388	357	641	597	583	339	119	398	377	515
Coru.	1118	644	535	683	799	0	1043	334	650	609	1153	1010	340	738	547	947	675	961	455	833
Gra.	868	829	671	485	278	1043	0	761	770	434	129	278	885	841	827	256	397	519	627	759
Leo.	784	359	201	407	511	334	761	0	316	333	877	734	118	404	293	671	392	685	134	488
Log.	468	152	115	595	526	650	770	316	0	336	880	694	391	88	225	874	407	481	237	172
Mad.	621	395	237	297	190	609	434	333	336	0	544	401	451	407	393	538	71	352	193	325
Mal.	997	939	781	506	388	1153	129	877	880	544	0	407	995	951	937	219	507	648	737	869
Mur.	590	796	638	654	357	1010	278	734	694	401	407	0	852	714	794	534	390	241	594	539
Ovi.	902	304	322	525	641	340	885	118	391	451	995	852	0	463	207	789	510	803	252	604
Pam.	437	159	203	650	597	738	841	404	88	407	951	714	463	0	267	945	478	501	325	175
Sant.	693	108	156	573	583	547	827	293	225	393	937	794	207	267	0	837	464	673	248	397
Sev.	1046	933	775	264	339	947	256	671	874	538	219	534	789	945	837	0	458	697	589	863
Tol.	692	466	308	264	119	675	397	392	407	71	507	390	510	478	464	458	0	372	258	396
Val.	349	633	517	636	398	961	519	685	481	352	648	241	803	501	673	697	372	0	545	326
Vall.	663	280	122	325	377	455	627	134	237	193	737	594	252	325	248	589	258	545	0	367
Zar.	296	324	287	622	515	833	759	488	172	325	869	539	604	175	397	863	396	326	367	0

Tabla III.14: Matriz de costes entre 20 ciudades españolas.

APÉNDICE IV

Pruebas Problema de la Programación de Producción Discreta

Este apéndice contiene los resultados completos de las pruebas del problema de la programación de producción discreta, desarrolladas con el fin de mostrar la adecuación del modelo neuronal ODE para implementar problemas prácticos de asignación de recursos. Estas pruebas sirven como referencia del comportamiento de la red ODE en la resolución de este tipo de problemas, respecto a los métodos actuales de optimización combinatoria más importantes: búsqueda local, temple simulado, red de Hopfield continua y máquina de Boltzmann.

Para efectuar las pruebas, se han realizado en lenguaje C las diferentes implementaciones del problema de la programación de producción discreta con cada uno de los métodos de comparación, utilizando en su desarrollo una metodología de diseño y programación modular basada en máquinas abstractas. Estas pruebas se han realizado sobre una instancia del problema, correspondiente a la programación de 40 pedidos de 3 tipos de artículos en una planta industrial con 7 máquinas.

Los resultados se presentan agrupados en cinco apartados, correspondientes a cada una de las técnicas de resolución empleadas. Así mismo, al final del apéndice se proporcionan las matrices de costes y duraciones, así como la lista de pedidos, correspondientes a la instancia del problema de la programación de producción discreta que se ha utilizado en las pruebas.

IV.1 PRUEBAS PPPD BÚSQUEDA LOCAL.

La tabla IV.1 presenta los resultados de las pruebas, realizadas mediante el algoritmo de búsqueda local, del problema de la programación de producción discreta. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras IV.1, IV.2 y IV.3):

I	COSTE	TIEMPO	SOLUCIÓN
1	4937.6	7	5 1 5 1 2 6 5 4 3 6 6 0 5 6 0 4 3 5 0 4 5 5 4 2 3 3 2 2 5 3 6 0 5 3 5 3 4 1 1 1
2	4965.2	6	5 1 5 1 4 2 5 4 2 2 6 0 5 6 0 4 3 5 0 4 5 5 4 6 3 3 2 3 5 3 6 0 5 1 6 5 1 3 1 3
3	4962.6	6	5 3 5 1 2 6 5 4 2 6 6 0 5 6 0 4 3 5 0 4 5 5 4 6 5 3 2 4 3 3 2 0 5 1 5 3 1 1 1 3
4	4957.4	6	5 1 5 1 4 2 5 4 2 0 6 0 5 6 0 4 3 5 0 1 5 5 1 6 3 3 4 4 5 3 6 2 5 3 6 3 1 1 2 3
5	4948.8	6	5 2 5 1 2 6 5 4 3 6 5 0 5 6 0 4 3 5 0 4 5 5 4 2 3 3 2 3 5 3 6 0 5 1 6 3 1 1 1 4
6	4942.7	6	5 1 5 1 2 6 5 4 3 6 6 0 5 6 0 4 3 5 0 4 5 5 4 6 3 3 2 2 5 3 2 0 5 3 5 3 4 1 1 1
7	4947.4	4	5 1 5 1 4 6 5 4 1 6 6 0 5 6 0 4 3 5 0 4 5 5 4 2 3 3 2 1 5 3 2 0 5 3 6 3 1 1 2 3
8	4940.5	7	5 2 5 1 2 6 5 4 2 6 6 0 5 6 4 4 3 5 0 4 5 5 4 0 3 3 2 3 5 3 6 0 5 3 5 3 1 1 1 1
9	4935.5	7	5 1 5 1 2 6 5 4 1 6 5 0 5 6 0 4 3 5 0 4 5 5 4 6 3 3 2 3 5 3 2 0 5 3 6 3 4 1 2 1
10	4929.5	6	5 1 5 1 2 6 5 4 3 6 5 0 5 6 0 4 3 5 0 4 5 5 4 6 3 3 2 4 5 3 2 0 5 3 6 3 1 1 2 1
11	4946.4	5	5 1 5 1 2 6 5 4 3 6 6 0 5 6 0 4 3 5 0 4 5 5 4 6 3 3 2 1 5 3 2 0 5 1 5 3 1 2 4 3
12	4951.1	6	5 1 5 1 4 0 5 4 3 2 6 4 5 6 0 4 3 5 0 4 5 5 1 6 3 3 2 3 5 3 6 0 5 3 6 5 1 2 1 2
13	4948.7	5	5 2 5 1 2 6 5 4 3 6 6 0 5 6 0 4 3 5 0 4 5 5 4 2 3 3 2 4 3 5 6 0 5 3 5 3 1 1 1 1
14	4967.0	6	5 3 5 1 2 6 5 4 2 2 5 0 5 6 0 4 3 5 0 4 5 5 4 6 3 3 1 4 3 3 6 0 5 3 6 5 1 2 1 1
15	4954.8	6	5 1 5 1 2 2 5 4 1 6 6 0 5 6 0 4 3 5 0 4 5 5 1 6 3 3 2 3 5 3 2 0 5 3 6 3 4 1 4 1
16	4945.1	6	5 3 5 1 2 6 5 4 1 6 6 0 5 6 0 4 3 5 0 4 5 5 4 0 3 3 2 4 3 3 6 2 5 3 5 5 1 1 2 1
17	4948.1	5	5 3 5 1 2 6 5 4 2 6 6 0 5 6 0 4 3 5 0 4 5 5 4 0 3 3 2 4 5 3 6 2 5 1 5 3 1 3 1 1
18	4931.8	6	5 1 5 1 4 6 5 4 1 6 5 0 5 6 0 4 3 5 0 4 5 5 4 2 3 3 2 3 5 3 6 0 5 3 6 3 1 1 2 2
19	4974.7	5	5 3 5 1 2 6 5 4 2 2 6 0 5 6 0 4 3 5 0 4 5 5 4 0 3 3 2 4 5 3 6 6 5 3 5 1 1 1 1 3
20	4947.8	7	5 2 5 1 4 6 5 4 3 6 6 0 5 6 0 4 3 5 0 4 5 5 1 2 3 3 2 1 5 3 6 2 5 3 5 3 1 1 4 1

Tabla IV.1: Pruebas problema de la prog. prod. discreta 40 con búsqueda local.

COSTE		**TIEMPO**	
Medio:	4 949.1	Medio:	5.9 sg
Mejor:	4 929.5	Mejor:	4 sg
Peor:	4 974.7	Peor:	7 sg
Desviación:	11.8	Desviación:	0.8 sg

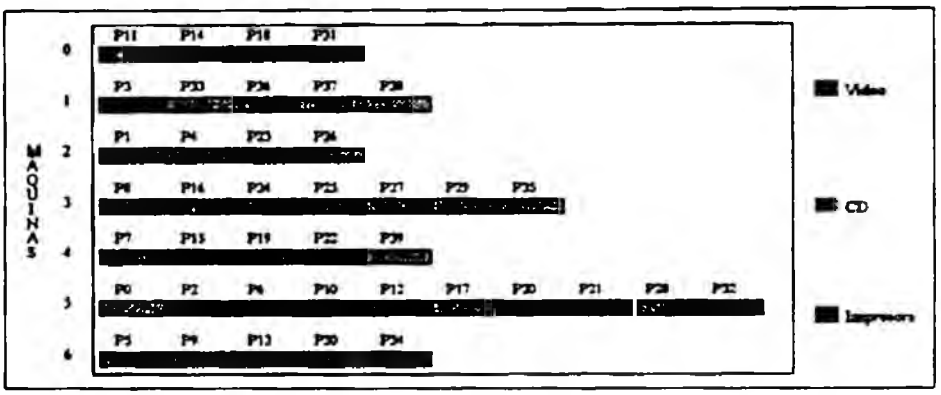

Figura IV.1: Solución media a la prog. prod. discreta 40 con búsqueda local: 4 948.8.

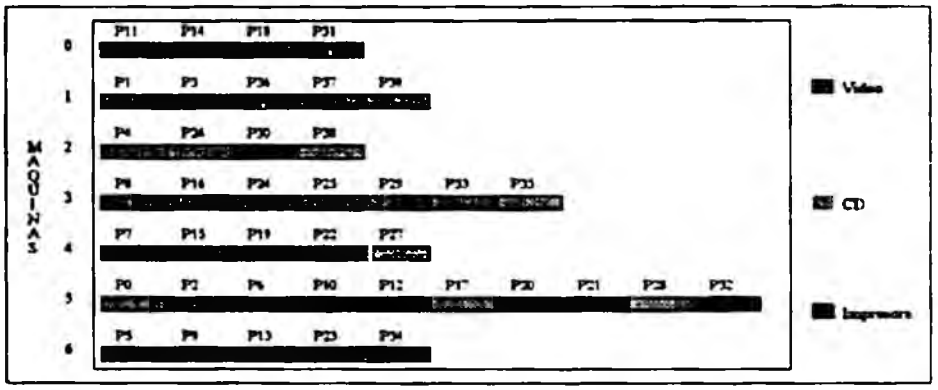

Figura IV.2: Solución mejor a la prog. prod. discreta 40 con búsqueda local: 4 929.5.

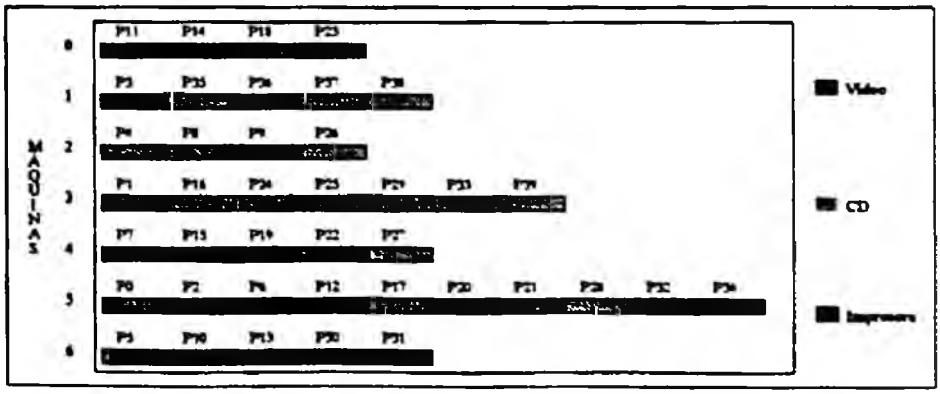

Figura IV.3: Solución peor a la prog. prod. discreta 40 con búsqueda local: 4 974.7.

IV.2 PRUEBAS PPPD TEMPLE SIMULADO.

La tabla IV.2 presenta los resultados de las pruebas, realizadas mediante el algoritmo de temple simulado, del problema de la programación de producción discreta. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras IV.4, IV.5 y IV.6):

I	COSTE	TIEMPO	SOLUCIÓN
1	4921.4	34	5 1 5 1 4 6 5 1 1 6 5 4 5 6 0 4 3 3 0 4 5 5 4 0 5 3 2 2 5 3 6 0 5 3 6 3 1 2 2 3
2	4927.3	36	5 3 5 1 2 6 5 1 2 6 6 0 5 6 0 4 3 3 4 4 5 5 4 0 3 3 2 2 5 5 6 0 5 3 5 3 1 1 4 1
3	4920.3	36	5 3 5 1 2 6 5 1 3 6 6 4 5 6 0 4 3 5 0 4 5 5 1 0 5 3 2 2 5 3 6 0 5 3 6 3 4 1 2 1
4	4922.0	40	5 1 5 1 2 6 5 1 3 6 6 4 5 6 0 4 3 3 0 4 5 5 4 0 5 3 2 2 5 3 6 0 5 3 5 3 4 1 2 1
5	4922.6	44	3 1 5 1 2 6 5 4 1 6 5 4 5 6 0 4 3 3 0 4 5 5 4 0 5 3 2 2 5 5 6 0 5 3 6 3 1 3 2 1
6	4919.8	40	5 3 5 1 2 6 5 1 1 0 5 0 5 6 0 4 3 5 4 4 5 5 4 6 3 3 2 2 5 3 6 0 5 3 6 3 1 1 2 4
7	4917.8	40	5 1 5 1 2 0 5 1 1 6 5 4 5 6 0 4 3 5 0 4 5 5 4 6 3 3 2 2 5 3 6 0 5 3 6 3 4 1 2 3
8	4919.6	35	5 2 5 1 2 0 5 1 3 6 5 4 5 6 0 4 3 5 0 4 5 5 4 6 3 3 2 1 5 3 6 0 5 3 6 3 4 1 2 1
9	4921.7	36	3 1 5 1 2 0 5 1 1 6 5 4 5 6 0 4 3 5 0 4 5 5 4 6 5 3 2 2 5 3 6 0 5 3 6 3 4 1 2 3
10	4920.2	35	5 2 5 1 2 0 5 1 3 6 6 4 5 6 0 4 3 5 4 4 5 5 4 0 3 3 2 1 5 5 6 0 5 3 6 3 1 3 2 1
11	4922.5	40	5 1 5 1 2 6 5 1 2 6 5 0 5 6 4 4 3 3 0 4 5 5 4 0 3 3 2 4 5 5 6 0 5 3 6 3 1 3 2 1
12	4918.1	41	5 3 5 1 2 6 5 1 1 6 5 4 5 6 0 4 3 3 0 4 5 5 4 0 3 3 2 2 5 5 6 0 5 3 6 3 4 1 2 1
13	4915.9	41	5 3 5 1 2 6 5 1 1 6 5 4 5 6 0 4 3 5 0 4 5 5 4 0 3 3 2 2 5 3 6 0 5 3 6 3 4 1 2 1
14	4917.0	39	5 1 5 1 2 0 5 1 1 6 5 4 5 6 0 4 3 5 0 4 5 5 4 6 3 3 2 2 5 3 6 0 5 3 6 3 1 3 2 4
15	4928.9	39	3 1 5 1 2 6 5 1 3 6 6 0 5 6 0 4 3 3 4 4 5 5 4 0 5 3 2 4 5 5 6 0 5 3 5 3 1 1 2 2
16	4917.8	33	5 1 5 1 2 6 5 1 3 0 6 4 5 6 4 4 3 5 0 4 5 5 4 0 3 3 2 2 5 5 6 0 5 3 6 3 1 3 2 1
17	4929.3	40	5 1 5 1 2 0 5 1 2 6 6 4 5 6 0 4 3 3 0 4 5 5 1 6 3 3 2 2 5 5 6 0 5 3 5 3 1 3 4 4
18	4922.3	38	5 3 5 1 4 6 5 1 1 6 5 4 5 6 0 4 3 3 0 4 5 5 4 0 5 3 2 1 5 3 6 0 5 3 6 3 1 2 2 2
19	4929.4	39	5 3 5 1 2 0 5 1 3 6 6 4 5 6 4 4 1 5 0 4 5 5 4 0 3 3 2 2 5 3 6 0 5 3 6 5 1 3 2 1
20	4926.9	38	5 1 5 1 2 6 5 1 1 6 6 4 5 6 0 4 3 5 0 4 5 5 4 0 3 3 2 3 5 3 6 0 5 3 5 3 1 2 2 4

Tabla IV.2: Pruebas problema de la prog. prod. discreta 40 con temple simulado.

COSTE		TIEMPO	
Medio:	4 922.0	Medio:	38.2 sg
Mejor:	4 915.9	Mejor:	33 sg
Peor:	4 929.4	Peor:	44 sg
Desviación:	4.2	Desviación:	2.8 sg

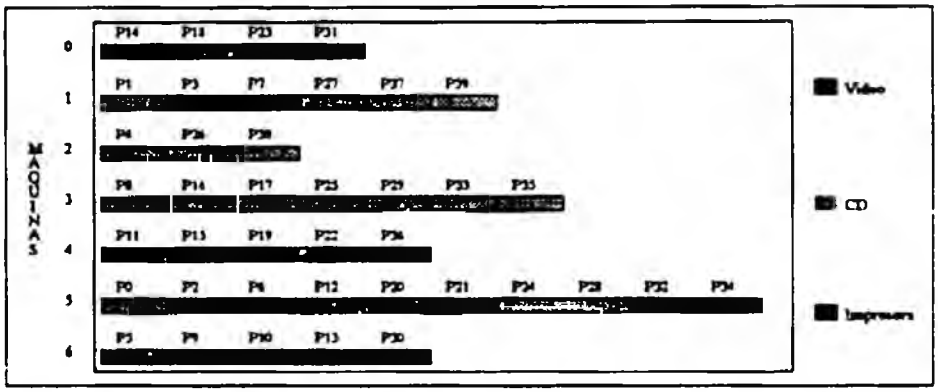

Figura IV.4: Solución media a la prog. prod. discreta 40 con temple simulado: 4 922.0.

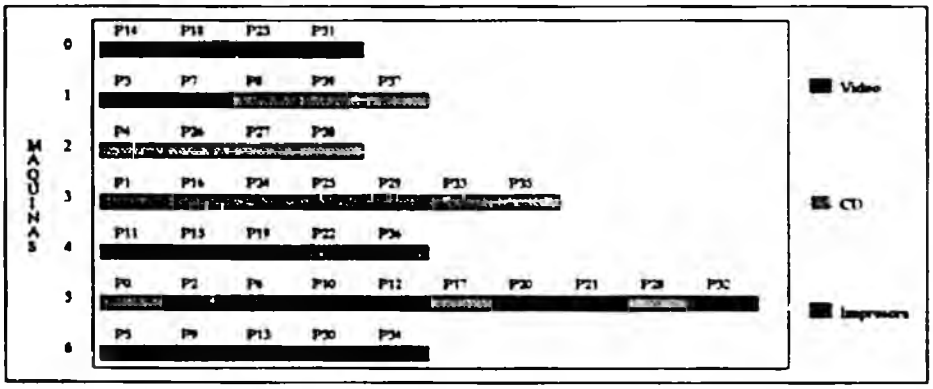

Figura IV.5: Solución mejor a la prog. prod. discreta 40 con temple simulado: 4 915.9.

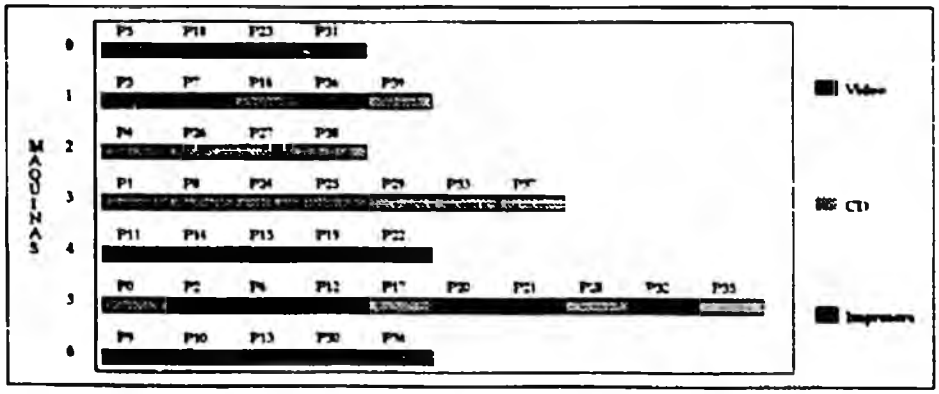

Figura IV.6: Solución peor a la prog. prod. discreta 40 con temple simulado: 4 929.4.

IV.3 PRUEBAS PPPD RED DE HOPFIELD CONTINUA.

La tabla IV.3 presenta los resultados de las pruebas, realizadas mediante la red de Hopfield continua, del problema de la programación de producción discreta. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras IV.7, IV.8 y IV.9):

I	COSTE	TIEMPO	SOLUCIÓN
1	5882.8	1785	5 5 3 1 1 6 5 0 3 5 5 6 5 2 0 1 2 5 1 5 0 6 4 0 4 5 2 4 1 4 2 4 5 3 6 3 3 6 3 5
2	6687.0	1498	1 4 5 1 3 6 3 6 6 2 0 1 4 4 0 3 6 3 6 1 3 3 6 0 4 5 4 4 5 5 6 0 5 4 0 6 0 2 5 2
3	6344.5	1540	1 5 6 4 3 2 2 1 5 6 6 1 2 3 5 3 1 5 0 3 0 2 5 5 6 2 2 6 1 3 4 0 5 3 0 5 5 5 4 3
4	6142.8	1570	5 3 0 4 2 4 5 4 2 6 4 4 4 3 6 0 5 3 3 1 6 5 1 5 4 4 5 1 1 3 0 4 5 1 5 5 4 4 6 3
5	5923.2	1758	5 5 5 5 2 6 2 0 4 6 0 4 3 5 4 0 5 3 4 0 5 5 0 3 3 1 4 2 2 3 6 0 5 1 6 5 5 3 2 4
6	6217.9	1476	2 4 5 1 2 5 2 4 3 3 4 1 5 3 6 0 1 3 3 6 0 4 4 5 4 4 3 1 5 6 6 2 5 1 3 1 4 6 2 6
7	5994.2	1643	1 5 6 1 5 0 0 3 5 3 6 1 2 6 0 3 6 1 1 1 5 5 4 0 3 5 2 4 3 2 2 6 6 5 2 5 4 3 6 1
8	5637.9	1693	3 5 2 1 1 2 6 1 1 5 5 0 5 6 5 3 4 5 4 4 6 5 4 3 2 5 2 5 5 3 0 0 6 3 4 5 4 2 1 3
9	6343 1	1422	4 1 5 5 5 2 6 0 1 4 0 3 6 4 6 3 1 4 0 4 2 2 0 2 3 4 5 1 3 5 6 5 2 5 3 3 4 2 5 1
10	5892.4	1935	5 2 5 1 2 0 6 1 5 2 6 1 5 3 4 1 5 4 0 3 6 0 4 4 5 1 2 4 6 6 0 3 2 4 6 3 1 1 4 5
11	5934.3	1953	3 5 5 4 5 5 6 1 2 6 2 0 0 2 4 0 4 6 6 4 6 2 1 5 5 1 2 4 1 3 3 6 6 3 5 5 3 3 5 1
12	6664 8	1462	5 4 0 1 4 6 6 5 4 4 0 6 0 0 1 4 4 1 5 0 2 6 3 0 6 6 3 3 3 3 6 2 4 5 6 1 5 6 5
13	5838.1	1400	6 3 6 0 3 5 5 4 4 4 5 4 2 6 1 0 1 5 1 1 2 3 4 6 5 5 1 3 3 5 6 5 0 3 2 2 1 2 4 5
14	6632 7	1648	1 1 4 3 5 6 5 4 5 2 4 1 3 2 5 0 4 3 6 6 2 5 4 4 5 1 6 6 3 6 6 5 2 1 5 4 0 5 2 3
15	6126 5	1670	3 3 3 4 4 0 4 0 5 2 0 4 0 4 1 6 1 3 6 1 6 6 5 2 3 2 1 1 1 5 5 2 6 3 6 5 4 1 4 5
16	5981 1	1896	5 2 0 1 5 3 0 5 5 0 3 1 5 5 6 4 4 5 4 1 0 6 4 0 5 5 3 5 2 3 6 6 0 4 2 4 1 5 2 3
17	6137 7	1587	5 4 5 5 1 6 5 4 2 5 6 3 3 3 1 6 2 6 0 4 5 0 1 5 6 3 2 3 3 3 6 5 6 3 5 2 1 4 5 5
18	6138 8	1678	4 1 0 4 4 0 3 1 3 2 5 6 5 6 3 3 5 5 3 6 3 2 5 0 1 5 1 1 3 2 2 4 5 1 0 3 1 4 2 3
19	5691 9	1725	1 5 6 4 3 4 5 1 1 6 0 4 6 2 0 4 1 4 0 4 5 5 4 2 3 3 2 1 2 3 5 5 6 3 0 2 1 6 5 1
20	6004 0	1837	5 6 2 4 1 5 5 1 3 0 6 4 2 0 1 0 3 1 0 3 6 0 4 3 4 6 6 6 4 5 6 4 5 5 6 5 0 5 1 2

Tabla IV.3: Pruebas problema de la prog. prod. discreta 40 con la red CH.

COSTE		TIEMPO	
Medio:	6 110.8	Medio:	1658.8 sg
Mejor:	5 637.9	Mejor:	1400 sg
Peor:	6 687.0	Peor:	1953 sg
Desviación:	300.5	Desviación:	167.2 sg

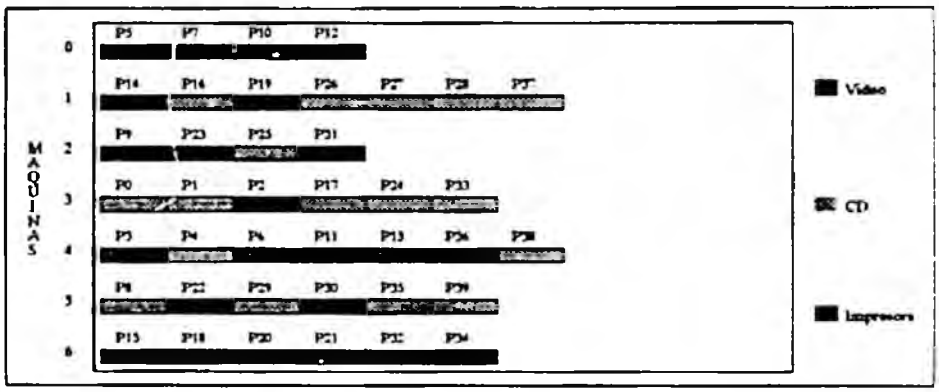

Figura IV.7: Solución media a la prog. prod. discreta 40 con la red CH: 6 126.5.

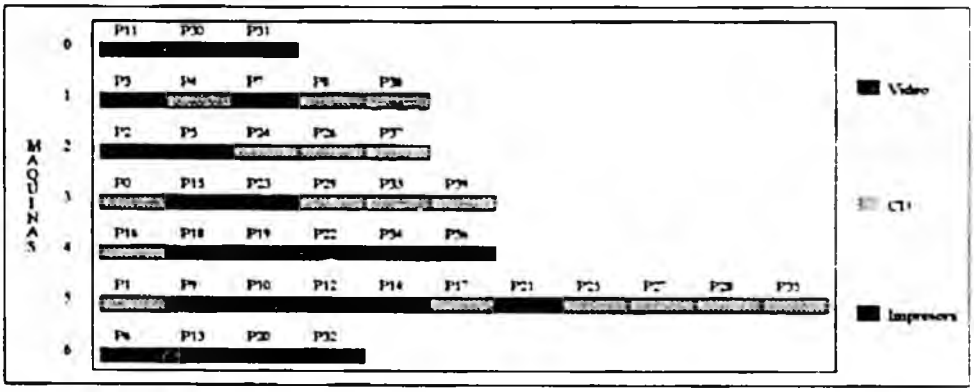

Figura IV.8: Solución mejor a la prog. prod. discreta 40 con la red CH: 5 637.9.

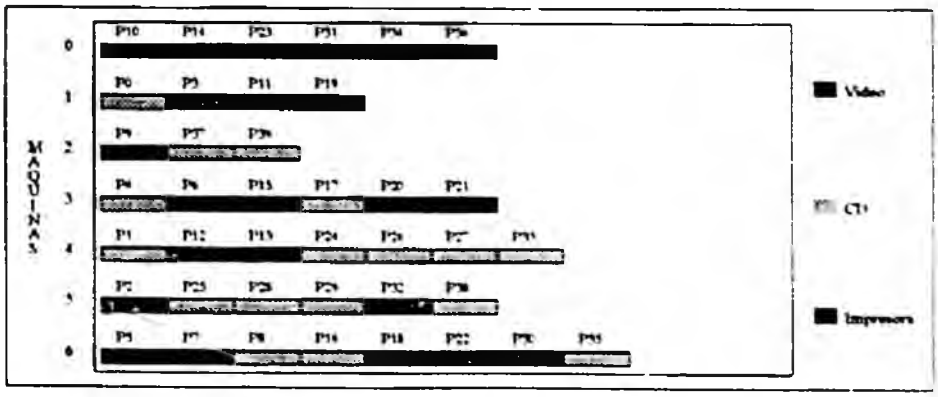

Figura IV.9: Solución peor a la prog. prod. discreta 40 con la red CH: 6 687.0.

IV.4 PRUEBAS PPPD MÁQUINA DE BOLTZMANN.

La tabla IV.4 presenta los resultados de las pruebas, realizadas mediante la máquina de Boltzmann, del problema de la programación de producción discreta. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras IV.10, IV.11 y IV.12):

I	COSTE	TIEMPO	SOLUCIÓN
1	6306.5	149	5 6 3 1 2 5 5 1 3 3 6 0 2 5 5 4 3 6 1 1 3 6 3 5 1 1 1 4 1 2 5 4 2 4 2 5 4 4 5 3
2	6401.8	147	1 5 5 0 6 3 3 1 5 4 0 6 2 6 6 3 5 5 4 0 0 3 3 2 2 6 6 3 5 1 0 4 5 3 3 1 0 6 2 2
3	6751.5	147	6 4 0 4 3 2 0 5 5 3 6 3 2 3 3 1 2 1 0 6 5 4 1 2 4 1 1 3 6 6 4 5 5 6 6 4 1 6 3 5
4	6602.1	143	5 3 0 0 4 6 3 4 3 3 6 4 4 6 6 5 3 1 5 6 6 4 1 3 3 2 2 4 5 6 2 3 3 3 2 5 1 4 2 1
5	6556.7	150	1 6 5 4 5 5 4 4 6 5 0 6 0 6 3 6 4 3 0 3 4 4 4 5 5 1 6 5 5 2 3 2 5 2 0 6 4 3 5 4
6	6616.5	148	3 2 0 3 4 0 5 5 2 6 6 1 5 6 6 1 4 6 6 4 0 0 0 4 1 5 6 4 2 2 0 5 3 5 0 1 0 1 3 4
7	6730.4	147	2 6 4 0 2 0 0 4 5 3 3 1 0 4 4 5 1 1 4 0 5 0 5 2 1 2 4 3 3 2 4 3 6 4 6 6 4 4 6 5
8	6316.8	147	3 5 3 0 6 5 0 1 3 6 2 3 5 6 4 5 4 2 5 1 6 2 4 4 3 4 1 6 3 2 6 2 2 3 0 4 0 6 5 2
9	6966.3	149	1 1 5 6 5 5 2 3 4 0 4 0 0 6 4 5 1 2 4 5 4 5 3 4 6 1 6 6 5 5 2 6 5 1 3 1 5 1 4 1
10	7055.5	151	2 3 5 5 5 0 5 0 2 0 2 5 3 2 3 0 4 1 6 5 0 6 5 2 2 4 1 4 2 5 2 3 3 6 0 4 6 2 6 2
11	7228.9	148	6 4 0 6 6 6 2 3 4 4 4 4 6 6 1 4 2 6 4 4 6 4 1 3 5 1 1 2 5 2 6 5 6 6 0 1 5 3 2 3
12	6089.1	150	2 3 6 6 1 3 6 4 4 5 6 3 6 5 3 4 1 5 4 0 3 2 0 0 5 4 2 6 1 1 5 0 5 5 3 1 4 5 1 1
13	6427.2	141	2 5 5 5 4 6 6 0 1 2 6 0 2 5 0 4 5 3 4 0 0 6 1 4 4 5 5 4 1 1 5 0 3 6 4 3 3 4 4 1
14	6089.1	149	3 3 5 1 2 0 5 5 6 2 0 0 5 5 5 4 3 5 1 4 6 0 6 4 5 4 4 3 1 6 2 3 5 3 3 1 3 3 2 2
15	6821.2	144	1 4 3 5 2 0 5 5 3 3 5 6 2 5 6 6 3 1 4 4 3 0 1 6 4 1 2 5 4 1 3 3 3 3 3 5 6 4 2 1
16	6326.2	154	6 5 5 1 3 5 5 5 1 0 2 3 3 6 4 4 6 3 0 5 6 5 5 4 1 5 4 2 2 1 5 6 2 2 5 6 4 1 3 4
17	6549.8	146	4 1 6 3 4 4 5 0 4 3 0 5 3 6 5 4 1 5 3 3 5 5 0 6 1 2 2 2 6 1 0 0 5 6 2 6 0 1 4 4
18	6415.0	145	5 2 5 1 1 5 0 3 3 5 6 6 0 0 5 5 1 4 0 1 3 2 3 2 1 1 1 1 4 1 5 2 5 3 2 3 0 2 4 4
19	6300.5	151	1 2 0 5 4 3 3 0 4 2 0 3 5 5 6 1 1 3 3 0 5 6 3 3 5 2 4 6 6 5 2 4 2 5 5 5 1 1 2 6
20	7019.6	142	3 2 3 3 5 2 5 5 2 6 6 3 4 0 5 3 1 6 0 5 6 5 5 2 5 1 2 4 2 4 3 3 4 4 0 3 4 6 5 1

Tabla IV.4: Pruebas problema de la prog. prod. discreta 40 con la red BM.

COSTE		TIEMPO	
Medio:	6 578.5	Medio:	147.4 sg
Mejor:	6 089.1	Mejor:	141 sg
Peor:	7 228.9	Peor:	154 sg
Desviación:	320.7	Desviación:	3.3 sg

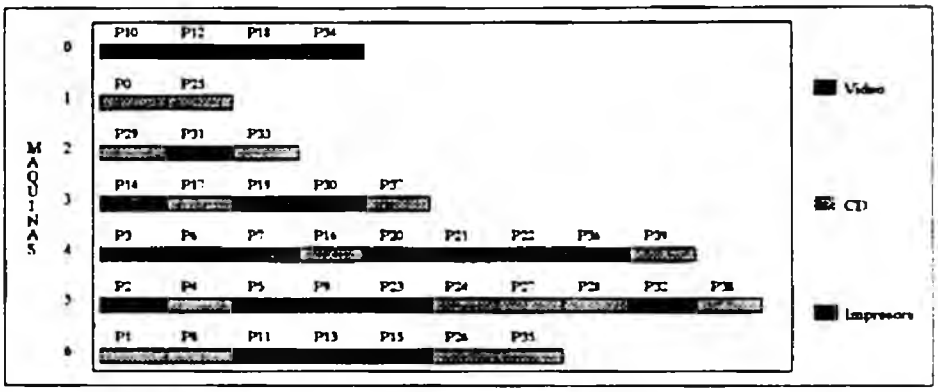

Figura IV.10: Solución media a la prog. prod. discreta 40 con la red BM: 6 556.7.

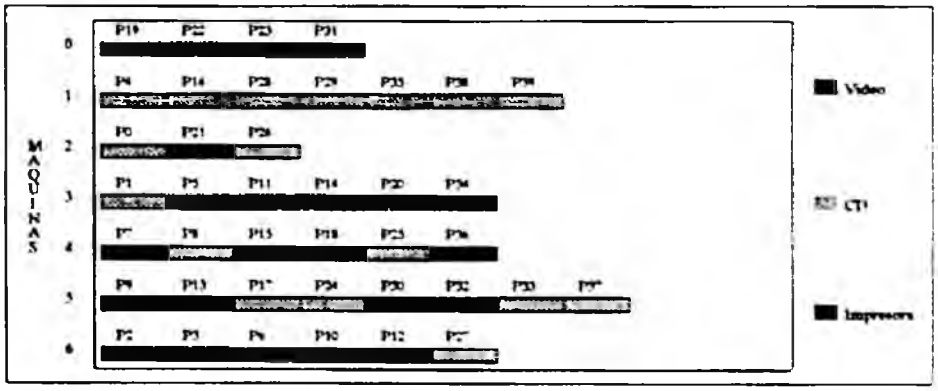

Figura IV.11: Solución mejor a la prog. prod. discreta 40 con la red BM: 6 089.1.

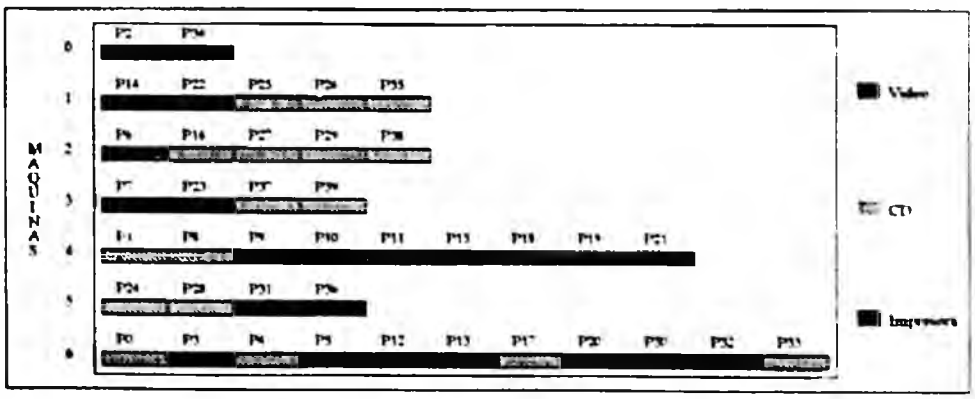

Figura IV.12: Solución peor a la prog. prod. discreta 40 con la red BM: 7 228.9

IV.5 PRUEBAS PPPD OPTIMIZADOR ESTOCÁSTICO.

La tabla IV.5 presenta los resultados de las pruebas, realizadas mediante el optimizador discreto estocástico, del problema de la programación de producción discreta. Así mismo, se ofrece la representación gráfica de las soluciones media, mejor y peor obtenidas con este método (figuras IV.13, IV.14 y IV.15):

I	COSTE	TIEMPO	SOLUCIÓN
1	4926.6	7	5 1 5 1 2 6 5 4 1 0 5 4 5 6 0 4 3 3 0 4 5 5 4 6 3 5 2 2 5 3 6 0 5 3 6 3 1 3 2 1
2	4920.9	6	5 1 5 1 2 0 5 1 1 6 6 4 5 6 0 4 3 5 0 4 5 5 4 6 3 3 2 2 5 3 6 0 5 3 5 3 1 3 2 4
3	4924.9	6	5 1 5 1 2 0 5 4 2 6 5 0 5 6 0 4 3 3 4 4 5 5 4 6 5 3 2 1 5 3 6 0 5 3 6 3 1 3 2 1
4	4918.4	7	5 1 5 1 2 0 5 1 1 6 6 4 5 6 0 4 3 5 4 4 5 5 4 0 5 3 2 2 5 3 6 0 5 3 6 3 1 3 2 3
5	4923.3	8	5 1 5 1 2 6 5 4 3 0 6 4 5 6 0 4 3 3 0 4 5 5 4 6 5 3 2 2 5 3 6 0 5 3 5 3 1 1 2 1
6	4922.1	7	5 1 5 1 2 6 5 1 1 6 6 4 5 6 0 4 3 5 0 4 5 5 4 0 3 3 2 4 5 3 6 0 5 3 5 3 1 2 2 3
7	4920.2	6	3 1 5 1 2 6 5 1 3 6 5 4 5 6 0 4 3 5 0 4 5 5 4 0 3 3 2 4 5 5 6 0 5 3 6 3 1 1 2 2
8	4928.3	7	5 2 5 1 2 0 5 1 1 6 5 4 5 6 0 4 3 3 0 4 5 5 4 6 3 3 2 1 5 3 6 0 5 3 6 5 4 3 2 1
9	4932.2	6	3 1 5 1 2 6 5 1 1 0 6 0 5 6 4 4 3 5 4 4 5 5 4 0 5 3 2 3 5 5 6 0 5 3 6 3 1 2 2 3
10	4924.9	6	5 3 5 1 1 0 5 1 3 6 6 4 5 6 0 4 3 5 0 4 5 5 4 6 5 3 2 2 5 3 6 0 5 3 6 3 1 1 2 2
11	4917.2	7	5 3 5 1 2 6 5 1 1 0 5 4 5 6 0 4 3 5 0 4 5 5 4 6 3 3 2 2 5 3 6 0 5 3 6 3 4 1 2 1
12	4918.6	6	5 3 5 1 2 6 5 1 1 6 5 4 5 6 0 4 3 3 0 4 5 5 4 0 3 2 2 4 5 5 6 0 5 3 6 3 1 1 2 2
13	4924.7	7	5 1 5 1 2 6 5 4 3 6 6 0 5 6 4 4 3 3 0 4 5 5 4 0 5 3 2 2 5 3 6 0 5 3 5 3 1 1 2 1
14	4924.0	6	5 3 5 1 2 0 5 1 2 6 6 4 5 6 0 4 3 3 0 4 5 5 4 6 5 3 2 4 5 3 6 0 5 3 5 3 1 1 2 1
15	4921.7	6	5 2 5 1 2 6 5 1 3 0 6 4 5 6 0 4 3 5 0 4 5 5 4 6 5 3 2 1 5 3 6 0 5 3 6 3 1 3 2 1
16	4925.9	7	5 3 5 1 2 6 5 1 1 0 6 4 5 6 0 4 3 3 0 4 5 5 4 2 5 3 2 4 5 5 6 0 5 3 6 3 1 3 2 1
17	4926.5	6	5 3 5 1 2 2 5 1 1 0 6 4 5 6 0 4 3 3 0 4 5 5 4 6 5 3 2 4 5 5 6 0 5 3 6 3 1 3 2 1
18	4921.7	8	3 1 5 1 2 0 5 1 1 6 5 4 5 6 0 4 3 5 0 4 5 5 4 6 5 3 2 2 5 3 6 0 5 3 6 3 4 1 2 3
19	4917.2	7	5 1 5 1 2 6 5 1 3 0 5 4 5 6 0 4 3 5 0 4 5 5 4 6 3 3 2 2 5 3 6 0 5 3 6 3 1 1 2 4
20	4923.3	6	5 3 5 1 2 6 5 1 1 0 6 4 5 6 0 4 3 3 0 4 5 5 4 6 3 3 2 2 5 5 6 0 5 3 5 3 4 1 2 1

Tabla IV.5: Pruebas problema de la prog. prod. discreta 40 con la red ODE.

COSTE		**TIEMPO**	
Medio:	4 923.1	Medio:	6.6 sg
Mejor:	4 917.2	Mejor:	6 sg
Peor:	4 932.2	Peor:	8 sg
Desviación:	3.8	Desviación:	0.7 sg

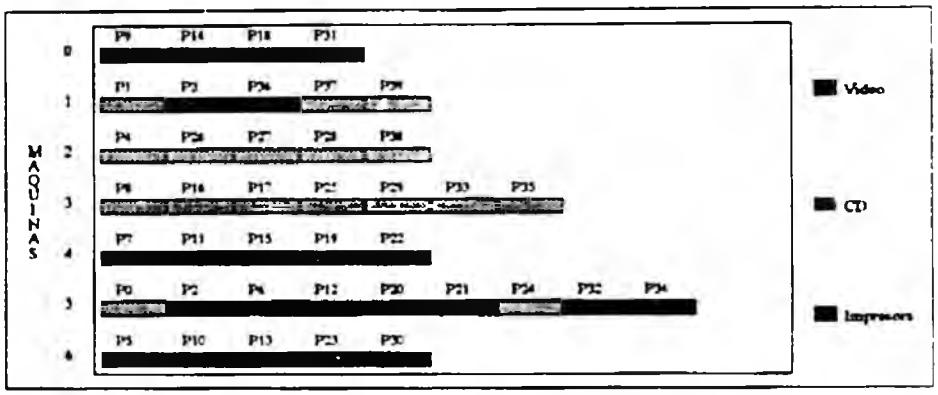

Figura IV.13: Solución media a la prog. prod. discreta 40 con la red ODE: 4 923.3.

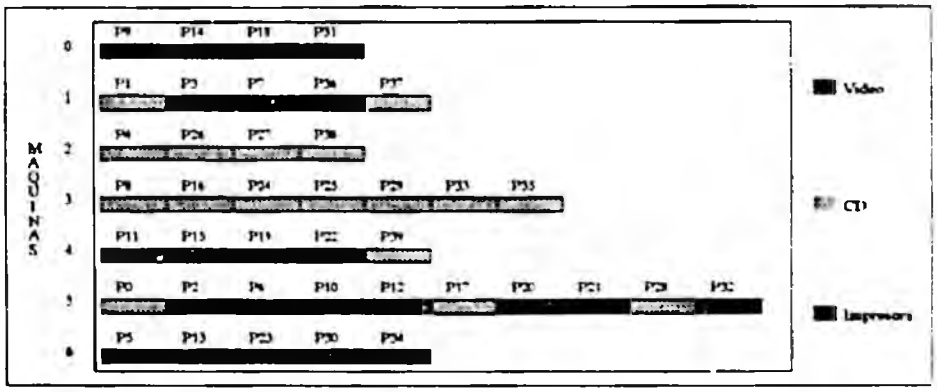

Figura IV.14: Solución mejor a la prog. prod. discreta 40 con la red ODE: 4 917.2.

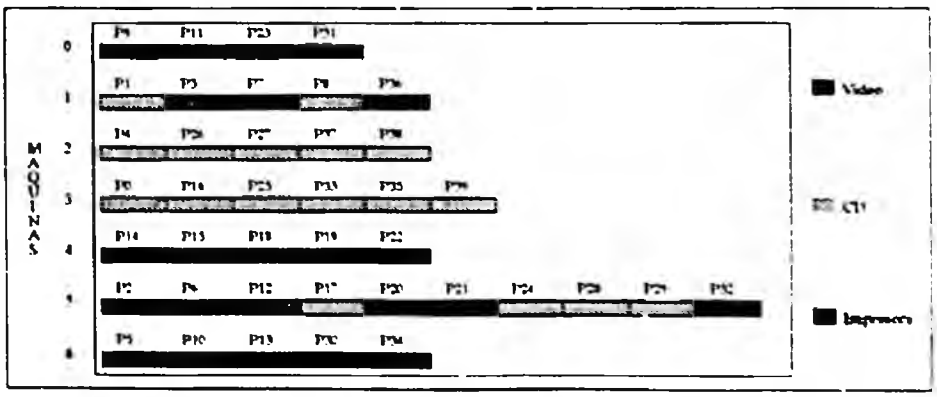

Figura IV.15: Solución peor a la prog. prod. discreta 40 con la red ODE: 4 932.2.

IV.6 DATOS UTILIZADOS EN LAS PRUEBAS.

A continuación se ofrecen las matrices de costes y duraciones, así como la lista de pedidos, utilizados en las pruebas del problema de la programación de producción discreta:

	Art. 0	Art. 1	Art. 2
Máq. 0	89	0	91
Máq. 1	0	66	41
Máq. 2	51	71	0
Máq. 3	45	19	77
Máq. 4	91	23	56
Máq. 5	12	2	80
Máq. 6	24	50	80

Tabla IV.6: Matriz 7x3 de costes de los procesos de fabricación.

	Art. 0	Art. 1	Art. 2
Máq. 0	1	0	1
Máq. 1	0	3	3
Máq. 2	5	4	0
Máq. 3	6	6	9
Máq. 4	4	9	2
Máq. 5	2	6	4
Máq. 6	5	9	10

Tabla IV.7: Matriz 7x3 de duraciones de los procesos de fabricación.

Pedido	Art.	Cant.	Pedido	Art.	Cant.	Pedido	Art.	Cant.	Pedido	Art	Cant
0	1	216	10	0	138	20	0	177	30	0	85
1	1	104	11	2	42	21	0	156	31	0	3
2	0	247	12	0	249	22	2	139	32	0	235
3	2	231	13	0	94	23	0	68	33	1	139
4	1	50	14	2	33	24	1	195	34	0	129
5	0	70	15	2	105	25	1	159	35	1	163
6	0	158	16	1	133	26	1	5	36	2	192
7	2	193	17	1	207	27	1	84	37	1	102
8	1	104	18	2	33	28	1	247	38	1	35
9	0	72	19	2	87	29	1	195	39	1	98

Tabla IV.8: Lista de 40 pedidos a programar.

BIBLIOGRAFÍA

[AART89] Aarts, E.H.L. & Korst, J.: *Simulated Annealing and Boltmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Wiley, Chichester, 1989.

[ABE89] Abe, S.: "Theories on the Hopfield neural networks". *Proceedings of the 1989 International Joint Conference on Neural Networks*, vol. 1: 557-564. Washington DC, 1989.

[ABRA92] Abramson, D. & Dang, H.: "School timetables: a case study in simulated annealing", en *Applied Simulated Annealing*, 103-124. Springer-Verlag. 1992.

[AIYE90] Aiyer, S.V.B., Niranjan, M. & Fallside, F.: "A theoretical investigation into the performance of the Hopfield model". *IEEE Transactions on Neural Networks*, vol. 1, No. 2 (1990):204-215.

[AKIY89] Akiyama, Y., Yamashita, A., Kajiura, M. & Aiso, H.: "Combinatorial optimization with Gaussian Machines", *Proceedings of the 1989 International Joint Conference on Neural Networks*, vol. 1: 533-540. Washington DC, 1989.

[AMAR72] Amari, S.I.: "Learning patterns and pattern sequences by self-organizing nets of threshold elements", *IEEE Transactions on Computers*, 21 (1972):1197-1206.

[AMAR74] Amari, S.I.: "A method of statistical neurodynamics". *Kybernetik*, 14 (1974):201-215.

[ATAB91] Atabakhsh, H.: "A Survey of Constraint Based Scheduling Systems using an Artificial Intelligence Approach". Artificial Intelligence in Engineering. Vol. 6, No. 2 (1991):58-73.

[AZEN92] Azencott, R.: *Simulated Annealing: Parallelization Techniques*. Wiley, 1992.

[BENS88] Bensana, E., Bel, G. & Dubois, D.: "OPAL: A Multiknowledge-Based System for Industrial Job-Shop Scheduling". International Journal of Production Research, Vol. 26, No. 5 (1988)

[BERR91] Berry, P. M.: A Predictive Model for Satisfying Conflicting Objectives in Scheduling Problems. Ph. D. Thesis, Department of Computer Science, University of Strathclyde, Glasgow, 1991.

[BOUT89] Van den Bout, D.E. & Miller, T.K.: "Graph partitioning using annealed neural networks", *Proceedings of the 1989 International Joint Conference on Neural Networks*, vol. I: 521-528. Washington DC, 1989.

[BURK89] Burke, P.: Scheduling in Dynamic Environments. Ph. D. Thesis, Department of Computer Science, University of Strathclyde, Glasgow, 1989.

[CERN85] Cerny, V.: "Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm", *Journal of Optimization Theory and Applications*, 45(1985):41-51.

[CHEN90] Chen, H. & Lee, S.J.: "Optimization search using neural networks", *Proceedings of the 1990 International Joint Conference on Neural Networks*, vol. II: 503-506. Washington DC, 1990.

[COHE83] Cohen, M. & Grossberg, S.: "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks", *IEEE Transactions on Systems, Man, and Cybernetics*, 13 (1983):815-825.

[COLL87] Collins, N.E., Eglese, R.W. & Golden, B.L.: *Simulated Annealing - An Annotated Bibliography*, Cambridge University Press, Cambridge, 1987.

[COLL88] Collinot, A., Le Pape, C. & Pinoteau, G.: "SONIA: A Knowledge-Based Scheduling System", Artificial Intelligence in Engineering, Vol. 3, No. 2 (1988).

[CULI90] Culioli, J.C., Protopopescu, V., Briton JR, C.L. & Ericson, M.N.: "Neural networks models for linear programming", *Proceedings of the 1990 International Joint Conference on Neural Networks*, vol. I: 293-296. Washington DC, 1990.

[DAOU95] Daouas, T., Ghedira, K., & Muller, J.P.: "Distributed flow-shop scheduling problem: global versus local optimization", *Proceedings of the First International Conference on Multi-Agent Systems*, 443. San Francisco, CA, 1995.

[ELLE89] Elleby, P., Fargher, H. E., Addis, T.R.: "A Constraint-Based Scheduling System for VLI Wafer Fabrication", en J. Browne (ed.) Knowledge-Based Production Management Systems. Elsevier Science Publishers, 1989.

[FOO88] Foo, Y. P. S., & Takefuji, Y.: "Stochastic neural networks for solving Job-Shop Scheduling", *Proceedings of the 1988 International Joint Conference on Neural Networks*, vol. II: 275-290. San Diego, CA, 1988.

[FOX83] Fox, M. S.: "Constraint Directed Search: A Case Study of Job-shop Scheduling", Ph. D. Thesis Carnegie Mellon University. Pittsburgh, PA,1983.

[FOX90a] Fox, M. S.: "AI and Expert Systems Myths, Legends, and Facts". IEEE Expert, Feb. 1990.

[FOX90b] Fox, M. S. & Sadeh, N.: "Why is Scheduling Difficult? A CSP Perspective". Proceedings of ECAI-90, 754-767.

[FREE95] Freeman, J. A. & Skapura, D. M.: *Redes Neuronales. Algoritmos, Aplicaciones y Técnicas de Programación.* Addison-Wesley/Díaz de Santos, Madrid, 1995.

[GARE79] Garey, M.R. & Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[GEE95] Gee, A.: "Are neural networks of any use for optimization?", *ICANN'95. International Conference on Artificial Neural Networks*, Tutorial n° 4. Paris, 1995.

[GEMA84] Geman, S. & Geman, D.: "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images", *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 6 (1984):721-741.

[GRAN86] Grant, T. J.: "Lessons for OR from AI: A Scheduling Case Study" Journal of the Operational Research Society, 37/1 (1986):41-57.

[GUID90] Guida, M. & Basaglia, G.: "Integrating Operational Research and Artificial Intelligence in a distributed approach to dynamic scheduling: the B.I.S. project", *Esprit'90 Conference Proceedings*, 544-558. Kluwer Academic Publishers, 1990.

[HAJE88] Hajek, B.: "Cooling schedules for optimal annealing", *Mathematics of Operations Research*, 13 (1988):311-329.

[HEBB49] Hebb, D. O.: *The Organization of Behavior.* Wiley, New York, 1949.

[HECH88] Hecht-Nielsen, R.: "Applications of counterpropagation networks", *Neural Networks, 1* (1988):131-140.

[HEDG88] Hedge, S.U., Sweet, J.L. & Levy, W.B.: "Determination of parameters in a Hopfield/Tank computational network", *Proceedings of the 1988 International Joint Conference on Neural Networks*, vol. II: 291-298 San Diego, CA, 1988.

[HINT86] Hinton, G.E. & Sejnowski, T.J.: "Learning and relearning in Boltzmann machines", en *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. I:282-317. Bradford Books, Cambridge MA, 1986.

[HOPF82] Hopfield, J.J.: "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences of the USA*, 79 (1982):2554-2558.

[HOPF84] Hopfield, J.J.: "Neurons with graded response have collective computational properties like those of two-state neurons", *Proceedings of the National Academy of Sciences of the USA*, 81 (1984):3088-3092.

[HOPF85] Hopfield, J.J. & Tank, D.: "Neural computation of decisions in optimization problems", *Biological Cybernetics*, 52 (1985):141-152.

[HYNY88] Hynynen, J.: A framework for Coordination in Distributed Production Management. Ph. D. Thesis, Laboratory of Information Processing, Computer Science Department, Helsinki University of Technology, 1988.

[JEON89] Jeong, H. & Park, H.: "Lower bounds of annealing schedule for Boltzmann and Cauchy machines", *Proceedings of the 1989 International Joint Conference on Neural Networks*, vol. I: 581-586. Washington DC, 1989.

[JOHN85] Johnson, D.S., Papadimitriou, C.H. & Yannakakis, M.: "How easy is local search?", *Proceedings of the Annual Symposium on Foundations of Computer Science*, pp. 39-42. Los Angeles, 1985.

[KAHN89] Kahng A.B.: "Traveling salesman heuristics and embedding dimension in the Hopfield model", *Proceedings of the 1989 International Joint Conference on Neural Networks*, vol. I: 513-520. Washington DC, 1989.

[KERN86] Kern, W.: "A probabilistic analysis of the switching algorithm for the Euclidean TSP", *Technical Report*, 86.28. Universität zu Köln, Köln, 1986.

[KIRK83] Kirkpatrick, S., Gelatt, C.D. & Vecchi, M.P.: "Optimization by simulating annealing", *Science*, 220 (1983):671-680.

[KOSK88] Kosko, B.: "Bidirectional associative memories", *IEEE Transactions on Systems, Man, and Cybernetics*, 18 (1988):42-60.

[LAAR87] Laarhoven, P.J.M. Van & Aarts, E.H.L.: *Simulated Annealing: Theory and Applications*, Reidel, Dordrecht, 1987.

[LAWL85] Lawler, L.E., Lenstra, J.K., Rinnooy Kan, A.H.G. & Shmoys, D.B. (eds.): *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, New York, 1985.

[LAWR86] Lawrence, S. R. & Morton, T. E.: "PATRIARCH: Hierarchical Production Scheduling", Symposium on Real Time Optimization in Automated Manufacturing Facilities (1986).

[LAZA92] Lázaro, J. M., Maseda, J. M., Díaz, F., Escalada, G. & Sturesson, H.: "Reactive Scheduler for Discrete Manufacturing", Workshop Notes on Scheduling of Production Processes, 10th European Conference on Artificial Intelligence (1992).

[LAZA93] Lázaro, J.M., Maseda, J.M., Díaz, F., Escalada, G. & Sturesson, H.: "Reactive Scheduler for Discrete Manufacturing", en J. Dorn & K. A. Froeschl (eds.): *Scheduling of Production Processes*, Ellis Horwood, 1993.

[LEE90] Lee, B. W. & Sheu, B. J.: "Combinatorial optimization using competitive-Hopfield neural network", *Proceedings of the 1990 International Joint Conference on Neural Networks*, vol. II: 627-630. Washington DC. 1990.

[LEE91] Lee, B. W. & Sheu, B. J.: "Modified Hopfield neural network for retrieving the optimal solution", *IEEE Transactions on Neural Networks*, vol. II, n° 1 (1991):137-142.

[LIN65] Lin, S.: "Computer solutions of the traveling salesman problem", *Bell System Technical Journal*, 44 (1965):2245-2269.

[LIN93] Lin, F. T., Kao, C. Y. & Hsu, C. C.: "Applying the genetic approach to simulated annealing in solving some NP-hard problems", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, n° 6 (1993):1752-1767.

[MESE89] Meseguer, P.: "Constraint Satisfaction Problems: An Overview", AICOM, vol. 2 No. 1 (1989):3-17.

[METR53] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. & Teller, E.: "Equation of state calculations by fast computing machines", *Journal of Chemical Physics*, 21 (1953):1087-1092.

[MINS69] Minsky, M. & Papert, S.: *Perceptrons*. MIT Press, Cambridge, MA, 1969.

[MITR86] Mitra, D., Romeo, F. & Sangiovanni-Vincentelli, A.L.: "Convergence and finite-time behavior of simulated annealing", *Advances in Applied Probability*, 18 (1986):747-771.

[NUDE83] Nudel, B.: "Consistent-Labeling Problems and their Algorithms: Expected-Complexities and Theory-Based Heuristics", Artificial Intelligence, 21 (1983):135-178.

[ONOD93] Onoda, J. & Hanawa, Y.: "Actuator placement optimization by genetic and improved simulated annealing algorithms", *AIAA Journal*, vol 31, n° 6 (1993):1167-1169.

[OW86] Ow, P. S. & Smith, S. F.: "Towards an Opportunistic Scheduling System", Proceedings of the 9th Annual Hawaii International Conference on System Sciences, 1986.

[OW88] Ow, P. S., Smith, S. F. & Howie, R. A.: "A Cooperative Scheduling System", en M. D. Oliff (ed.) Expert Systems and Intelligent Manufacturing. Elsevier Science Publishing Co., 1988.

[PAPE85] Le Pape, C.: "SOJA: A daily Workshop Scheduling System", Proceedings of the 5th Technical Conference of the British Computer Society, 1985.

[PAPE88] Le Pape, C.: Des Systemes d'Ordonnancement Flexibles et Opportunistes. Ph. D. Thesis, Universite de Paris-Sud, Centre d'Orsay, 1988.

[PELL94] Pellerin, D. & Hérault, J.: "Scheduling with neural networks: application to timetable construction", *Neurocomputing*, 6 (1994):419-442.

[PHEL86] Phelps, R. I.: "Artificial Intelligence - An Overview of similarities with OR". Journal of the Operational Research Society, 37/1 (1986):13-20.

[PROS89] Prosser, P.: "A Reactive Scheduling Agent", Proceedings of the IJCAI, 1989.

[PROS90] Prosser, P.: Distributed Asyncronous Scheduling. Ph. D. Thesis, Department of Computer Science, University of Strathclyde, Glasgow, 1990.

[PROT90] Protzel, P. W.: "Comparative performance measure for neural networks solving optimization problems", *Proceedings of the 1990 International Joint Conference on Neural Networks*, vol. II: 523-526. Washington, DC, 1990.

[ROFK92] Rofkar, J.D. & Takefuji, Y.: "A parallel algorithm for solving unfriendly beehive problems", *Neurocomputing*, 4 (1992): 167-179.

[RUME92] Rumelhart, D. E., McClelland, J. L. & the PDP research group: *Introducción al Procesamiento Paralelo Distribuido*. Alianza Editorial, Colección Alianza Psicología, 37, Madrid, 1992.

[SADE91] Sadeh, N.: Look-ahead Techniques for Micro-Opportunistic Job-Shop Scheduling. Ph. D. Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1991.

[SCHA92] Schaller, H. N.: "A collection of constraint design rules for neural optimization networks", *Proceedings of the 1992 International Conference on Artificial Neural Networks*, vol. II: 1039-1042. Brighton, 1992.

[SHAW86] Shaw, M. J. P. & Whinston, A. B.: "Applications of Artificial Intelligence to Planning and Scheduling in Flexible Manufacturing", Flexible Manufacturing Systems: Methods and Studies. A. Kusiak (ed.), 1986.

[SMIT86] Smith, S., Fox, M.S. & Ow, P. S.: "Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-based Factory Scheduling Systems". AI Magazine, Vol. 7 No. 4 (1986):45-61.

[STEF86] Steffen, M. S.: "A Survey of Artificial Intelligence Based Scheduling Systems", Fall Industrial Engineering Conference Proceedings, Boston, 1986.

[SZU87] Szu, H. & Hartley, R.: "Fast simulated annealing", *Physical Letters A*, 122 (1987):157-162.

[TAGL87] Tagliarini,G. A. & Page, E. W.: "Solving constraint satisfaction problems with neural networks", *Proceedings of the 1987 International Joint Conference on Neural Networks*, vol. III: 741-747. SanDiego, CA, 1987

[TAGL89] Tagliarini, G. A. & Page, E. W.: "Learning in systematically designed networks", *Proceedings of the 1989 International Joint Conference on Neural Networks*, vol. I: 497-502. Washington DC, 1989.

[TAKE89] Takefuji, Y. & Szu, H.: "Design of parallel distributed Cauchy Machines", *Proceedings of the 1989 International Joint Conference on Neural Networks*, vol. I: 529-532. Washington DC, 1989.

[TATE85] Tate, A. A.: "A Review of Knowledge Based Planning Techniques", Proceedings of the Fifth Technical Conference of the British Computer Society, 1985.

[TODA83] Toda, M., Kubo, R. & Saitô, N.: *Statistical Physics*. Springer-Verlag, Berlin, 1983.

[WILH94] Wilhelm, U.: "Finding improved simulated annealing schedules with genetic programming", *Proceedings of the 1994 IEEE Word Congress on Computational Intelligence*, vol. I:391-395. Orlando, Florida, 1994.

[WILS88] Wilson, G. & Pawley, G.: "On the stability of the traveling salesman problem algorithm of Hopfield and Tank", *Biological Cybernetics*, 58 (1988):63-70.

[YU90] Yu, T. L.: "Time-table scheduling using neural network algorithms", *Proceedings of the 1990 International Joint Conference on Neural Networks*, vol. I: 279-284. San Diego, CA, 1990.

[SHAW86] Shaw, M. J. P. & Whinston, A. B.: "Applications of Artificial Intelligence to Planning and Scheduling in Flexible Manufacturing", Flexible Manufacturing Systems: Methods and Studies. A. Kusiak (ed.), 1986.

[SMIT86] Smith, S., Fox, M.S. & Ow, P. S.: "Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-based Factory Scheduling Systems". AI Magazine, Vol. 7 No. 4 (1986):45-61.

[STEF86] Steffen, M. S.: "A Survey of Artificial Intelligence Based Scheduling Systems", Fall Industrial Engineering Conference Proceedings, Boston, 1986.

[SZU87] Szu, H. & Hartley, R.: "Fast simulated annealing", *Physical Letters A*, 122 (1987):157-162.

[TAGL87] Tagliarini,G. A. & Page, E. W.: "Solving constraint satisfaction problems with neural networks", *Proceedings of the 1987 International Joint Conference on Neural Networks*, vol. III: 741-747. SanDiego, CA, 1987.

[TAGL89] Tagliarini, G. A. & Page, E. W.: "Learning in systematically designed networks", *Proceedings of the 1989 International Joint Conference on Neural Networks*, vol. I: 497-502. Washington DC, 1989.

[TAKE89] Takefuji, Y. & Szu, H.: "Design of parallel distributed Cauchy Machines", *Proceedings of the 1989 International Joint Conference on Neural Networks*, vol. I: 529-532. Washington DC, 1989.

[TATE85] Tate, A. A.: "A Review of Knowledge Based Planning Techniques", Proceedings of the Fifth Technical Conference of the British Computer Society, 1985.

[TODA83] Toda, M., Kubo, R. & Saitô, N.: *Statistical Physics*. Springer-Verlag, Berlin, 1983.

[WILH94] Wilhelm, U.: "Finding improved simulated annealing schedules with genetic programming", *Proceedings of the 1994 IEEE Word Congress on Computational Intelligence*, vol. I:391-395. Orlando, Florida, 1994.

[WILS88] Wilson, G. & Pawley, G.: "On the stability of the traveling salesman problem algorithm of Hopfield and Tank", *Biological Cybernetics*, 58 (1988):63-70.

[YU90] Yu, T. L.: "Time-table scheduling using neural network algorithms", *Proceedings of the 1990 International Joint Conference on Neural Networks*, vol. I: 279-284. San Diego, CA, 1990.

[ZHOU91] Zhou, D. N., Cherkassky, V., Baldwin,T.R. & Olson, D. E.: "A neural network approach to Job-Shop Scheduling", *IEEE Transactions on Neural Networks*, vol. II, nº 1 (1991):175-179.

[ZWIE91] Zwietering, P. J., van Kraaij, M. J. A. L., Aarts, E. H. L. & Wessels, J.: "Neural networks and production planning", *Proceedings of the 4th International Conference on Neural Networks and their Applications. NEURO-NIMES'91*, Nimes, 1991.