Fabriciu Alarcão Veiga Benini

Rede Neural Recorrente com Perturbação Simultânea Aplicada no Problema do Caixeiro Viajante

Dissertação de mestrado apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, sendo parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Ivan Nunes da Silva

São Carlos
Dezembro 2008

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

Benini, Fabriciu Alarcão Veiga
B467r Rede neural recorrente com perturbação simultânea aplicada no problema do caixeiro viajante / Fabriciu Alarcão Veiga Benini ; orientador Ivan Nunes da Silva. -- São Carlos, 2008.

Dissertação (Mestrado-Programa de Pós-Graduação em Engenharia Elétrica e Área de Concentração em Sistemas Dinâmicos) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2008.

1. Redes neurais recorrentes. 2. Rede recorrente de Wang. 3. Perturbação simultânea com aproximação estocástica. 4. Regra de treinamento. 5. Problema do caixeiro viajante. 6. Problema combinatorial. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **FABRICIU ALARCÃO VEIGA BENINI**

Dissertação defendida e julgada em 15/12/2008 perante a Comissão Julgadora:

APROVADO

Prof. Associado **IVAN NUNES DA SILVA (Orientador)**
(Escola de Engenharia de São Carlos/USP)

Aprovado

Prof. Associado **ZHAO LIANG**
(Instituto de Ciências Matemáticas e de Computação)

APROVADO

Dr. **ROGÉRIO ANDRADE FLAUZINO**
(Pós-Doutorando/FAPESP)

Prof. Associado **GERALDO ROBERTO MARTINS DA COSTA**
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica e
Presidente da Comissão de Pós-Graduação

"Quanto a vós, sede fortes, não vos
acovardeis, pois vosso labor
terá sua recompensa."
II Crônicas 15,7

Dedicatória

À Luciane, minha esposa, com amor, admiração e gratidão por sua compreensão, renúncia, carinho, presença e incansável apoio ao longo do período de minha jornada em busca de conhecimento que se materializou por meio desse trabalho.

Agradecimentos

À minha mãe que sempre me incentivou, ao meu pai pela força e garantia de tranqüilidade nos momentos difíceis.

Ao Prof. Ivan Nunes da Silva que acreditou em mim, me apoiou, cedeu sua preciosa experiência para me orientar, pelo investimento de seu tempo em nossas conversas semanais sem deixar que eu desviasse o foco da pesquisa.

Aos outros professores da USP de São Carlos que me acompanharam nas diferentes etapas, sobretudo o Prof. Eduardo do Valle Simões quem me acompanhou no começo do mestrado me proporcionando o impulso inicial no campo de inteligência artificial.

Aos colegas de pós-graduação que compartilharam comigo suas experiências contribuindo para o meu amadurecimento na pesquisa científica.

À CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, pelo auxílio financeiro concedido em parte de minha jornada.

Resumo

BENINI, F. A. V. (2008). Rede Neural Recorrente com Perturbação Simultânea Aplicada no Problema do Caixeiro Viajante. Dissertação de Mestrado – Escola de Engenharia de São Carlos, Universidade de São Paulo, 2008.

O presente trabalho propõe resolver o clássico problema combinatorial conhecido como Problema do Caixeiro Viajante. Foi usado no sistema de otimização de busca do menor caminho uma rede neural recorrente. A topologia de estrutura de ligação das realimentações da rede adotada aqui é conhecida por Rede Recorrente de Wang. Como regra de treinamento de seus pesos sinápticos foi adotada a técnica de Perturbação Simultânea com Aproximação Estocástica. Foi elaborado ainda uma minuciosa revisão bibliográfica sobre todos os temas abordados com detalhes sobre a otimização multivariável com perturbação simultânea. Comparar-se-á também os resultados obtidos aqui com outras diferentes técnicas aplicadas no Problema do Caixeiro Viajante visando propósitos de validação.

Palavras chave: Rede neural recorrente, rede recorrente de Wang, perturbação simultânea com aproximação estocástica, regra de treinamento, problema do caixeiro viajante, problema combinatorial.

Abstract

BENINI, F. A. V. (2008). Recurrent Neural Network with Simultaneous Perturbation applied to Traveling Salesman Problem. Dissertation (Master's Degree) – Escola de Engenharia de São Carlos, Universidade de São Paulo, 2008.

This work proposes to solve the classic combinatorial optimization problem known as Traveling Salesman Problem. A recurrent neural network was used in the system of optimization to search the shorter path. The structural topology linking the feedbacks of the network adopted here is known by Wang recurrent network. As learning rule to find the appropriate values of the weights was used the Simultaneous Perturbation with Stochastic Approximation. A detailed bibliographical revision on multivariable optimization with simultaneous perturbation is also described. Comparative results with other different techniques applied to the Traveling Salesman are still presented for validation purposes.

Keywords: Recurrent neural network, Wang recurrent network, simultaneous perturbation stochastic approximation, learning rule, traveling salesman problem, combinatorial problem.

Lista de Siglas e Abreviaturas

AG	Algoritmo Genético
CF	Colônia de Formigas
MAOE	Mapas Auto-Organizáveis Expandido
MAOK	Mapas Auto-Organizáveis de Kohonen
PCV	Problema do Caixeiro Viajante
RHC	Rede de Hopfield Contínua
RN	Rede Neural
RNA	Rede Neural Artificial
RNH	Rede Neural de Hopfield
RNR	Rede Neural Recorrente
RRW	Rede Recorrente de Wang
VLT	Vencedor Leva Tudo

Lista de Figuras

Lista de Tabelas

Sumário

CAPÍTULO 1

Introdução

1.1 Motivação e Relevância do Trabalho

Os problemas de cobrimento de nós têm aplicações práticas no dia a dia da sociedade. Um veículo de coleta de lixo, por exemplo, precisa traçar um itinerário para coletar os tambores de lixo, sem repetir a passagem pelo ponto já visitado, sendo que depois deve levar os resíduos para o depósito. Quanto menor o caminho traçado, menos desperdício de recursos e tempo haverá para se executar a tarefa. O mesmo vale para o carteiro entregar as correspondências, pois ele parte da central entregando as correspondências e, no final, retorna para o mesmo ponto. Em problemas de produção, onde o seqüenciamento de *n* tarefas em uma única máquina visa minimizar o tempo total de execução das mesmas, assim como na linha de montagem de componentes eletrônicos, busca-se encontrar o roteiro de mínima distância para um equipamento cuja tarefa é soldar todos os componentes de uma placa eletrônica. O menor percurso total do equipamento para percorrer todos os pontos da placa está diretamente associado à produtividade da linha (SOUZA, 1993).

Por outro lado, problemas combinatórios têm ampla aplicação no campo computacional. São usados, por exemplo, em problemas de agrupamento (*clustering*) para classificação de dados e na recuperação a partir de análises efetuadas em bases de dados. Um algoritmo de agrupamento procura grupos naturais inerentes aos dados, usando medidas de distâncias entre si com similaridades entre eles individualmente (FURTADO, 1998).

Algoritmos de criptografia também usam largamente sistemas combinatórios para codificar e decodificar dados.

O Problema do Caixeiro Viajante (PCV) é caracterizado como sendo de otimização combinatória e sua dificuldade está no número elevado de soluções existentes, pois o mesmo pertence à categoria conhecida como NP-difícil (*NP-hard*), que significa que possui ordem de complexidade exponencial (GOTO e KAWAMURA, 2008). Assumindo que a distância de uma cidade i a outra j seja simétrica, isto é, que $d_{ij} = d_{ji}$, o número total de soluções possíveis é $(n - 1)! / 2$. Desta forma, a resolução do PCV por enumeração completa é inviável para valores elevados de n. Em outras palavras, o esforço computacional para a sua resolução cresce exponencialmente com o tamanho do problema. Pela sua definição singela e seu grau de abrangência em aplicações práticas o PCV tornou-se um dos problemas mais estudados em otimização combinatória motivando centenas de trabalhos publicados (CUNHA, BONASSER e ABRAHÃO, 2002).

O estudo do PCV desperta atenção pela possibilidade de aplicação em diferentes áreas do conhecimento, tais como pesquisa operacional, matemática, física, biologia e, no caso deste trabalho, inteligência computacional com foco em redes neurais artificiais (RNA). Consequentemente, o mesmo tem sido usado como estudo de caso para avaliação de novos algoritmos e estratégias de solução que envolve busca e otimização. Pela simplicidade que o PCV apresenta torna-se fácil entender e aplicar algoritmos para sua solução, que associado às inúmeras aplicações práticas, faz desse problema um dos mais aplicados em testes de técnicas de inteligência computacional atualmente.

1.2 Proposta e Justificativa da Dissertação

A presente dissertação visa investigar o desempenho da técnica conhecida como perturbação simultânea (PS) como regra de treinamento para ajustar os pesos de rede neural recorrente (RNR), de modo que a solução do PCV seja atingida satisfatoriamente. Para isso, a proposta será comparada com outras arquiteturas de RNs convencionais e/ou híbridas.

O PCV é um problema de otimização combinatorial classificado como NP-difícil. A única forma de se determinar com 100% de acerto a menor distância percorrida é por meio da enumeração de todas as combinações possíveis de caminhos, passando por todos os nós a fim de formar um ciclo Hamiltoniano, sendo que só então se escolhe o menor deles. A grande barreira em se adotar a força bruta é que, à medida que a quantidade de cidades a serem resolvidas cresce, o custo computacional aumenta exponencialmente. Para se selecionar o menor caminho percorrido para um número elevado de cidades, passa-se a ser inviável aplicar este algoritmo de força bruta.

Com o aumento do poder de processamento dos computadores, o número de cidades validadas por meio de algoritmos que usam a força bruta, onde todas as possíveis combinações são calculadas vêm aumentando. A última solução, usando esses algoritmos de força bruta, que se tem notícia foi realizada por Applegate et al (2007), onde foi encontrada a menor rota para uma composição de 85.900 nós.

A característica combinatorial que o PCV apresenta abre perspectivas para que um algoritmo eficiente que consiga resolver o maior número de cidades, com resultados mais próximos do real possível, tenha aplicações importantes para os mais variados campos de aplicação dessa natureza.

Entre os algoritmos mais comuns aplicados a esse tipo de problema destacam-se os que empregam técnicas de heurísticas. Eles são capazes de resolver problemas combinatoriais como o PCV sem necessitar testar todas as possibilidades em um tempo bastante reduzido se

comparado com RNAs. Os algoritmos que adotam técnicas heurísticas convencionais são aplicados em problemas combinatoriais desde o primórdio da computação.

Foi no final da década de 80 que RNs começaram a ser testadas para esse tipo de problema, principalmente usando a topologia da rede de Hopfield (SIQUEIRA, 2005). Naquele tempo os resultados ficavam bem aquém do desempenho obtido com os algoritmos heurísticos convencionais.

Com o amadurecimento do uso de RNs, aplicado em problemas combinatoriais, no que diz respeito à adaptabilidade das redes dentro do contexto, passou-se a intensificar a utilização de RNA, como em Salcedo-Sanz et al (2008), em que usou um banco de rede de Hopfield binário para resolver classes de restrições de problemas de natureza combinatorial, funcionando como um solucionador de restrições para otimizar a busca.

Cada vez mais a otimização com multivariáveis estocásticas vem envolvendo a maioria das funções de análises, com configurações híbridas em regras de treinamento para RNAs. Nos problemas de otimização é necessário usar algoritmos iterativos para encontrar um resultado quando uma solução analítica dificilmente é disponível, existindo ainda muitos pontos de convergência que não o global, mas sim local. Spall (1992) desenvolveu um método de aproximação estocástica com perturbação simultânea para problemas de otimização multivariável, o qual foi baseado em estudos anteriores, onde o conceito de perturbação simultânea foi largamente testado em RNR, em especial com a topologia de Hopfield. Maeda, Hirano e Kanata (1995) demonstraram que o aprendizado de uma rede pode ser conduzido com mais eficiência por meio da regra de perturbação simultânea e concluíram que seria eficiente para a aplicação em redes recorrentes. Maeda e Wakamura (2005) usaram perturbação simultânea em redes recorrentes e apontaram como principais vantagens dessa técnica a simplicidade e eficiência, reduzindo assim o custo computacional.

1.3 Organização da Dissertação

O presente trabalho apresenta a integração de várias áreas de pesquisa. Essa integração mostra o quê cada uma das partes tem de importante para se chegar a um resultado interessante para o meio científico. Por meio de comparações com outros trabalhos envolvendo RNAs aplicado ao PCV tenta-se estabelecer o nível da pesquisa frente ao estado da arte que existe atualmente. Em função dessas particularidades, este trabalho está dividido da seguinte forma:

O Capítulo 2 trata dos aspectos de diversos sistemas inteligentes aplicados no PCV, onde inicialmente se faz uma breve introdução sobre esse problema e, em seguida, o PCV é definido. Diversas topologias que utilizam RNAs para solucionar o PCV são também expostas, assim como se comenta sobre as topologias híbridas aplicadas no PCV.

No Capítulo 3 é apresentado o conceito de rede recorrente, juntamente com o conceito da perturbação simultânea, bem como a forma como os dois conceitos foram integrados. Antes, será feita uma explicação mais detalhada de cada um dos dois separadamente. Essa junção será explicada em cima de um mapeamento do PCV.

No Capítulo 4 serão expostos os resultados dos mais diversos trabalhos de RNA aplicadas ao PCV, destacando as diferenças entre os resultados de cada abordagem com aqueles obtidos com a proposta desta dissertação.

Finalmente, no Capítulo 5, apresentam-se as conclusões e os trabalhos futuros.

Capítulo 2

Aspectos de Sistemas Inteligentes Aplicados no Problema do Caixeiro Viajante

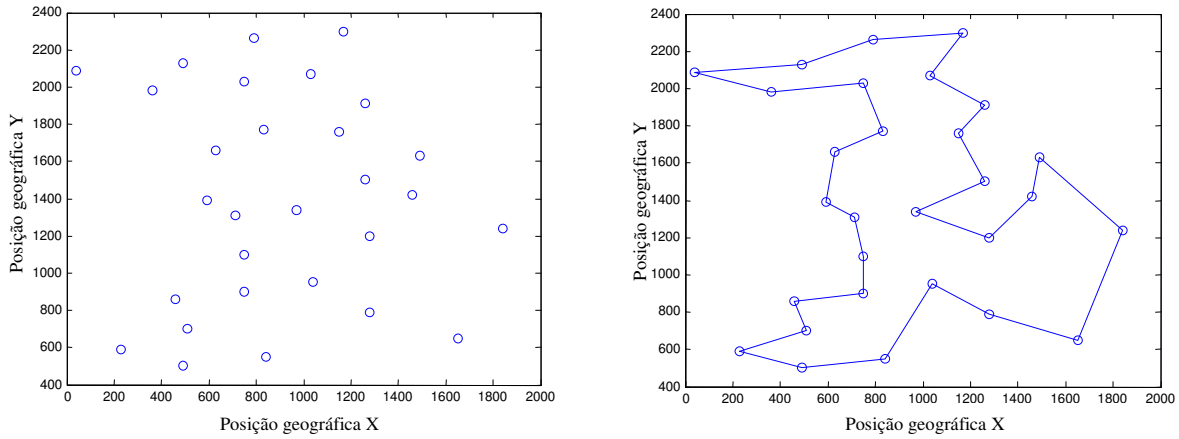
2.1 Introdução

O PCV foi proposto inicialmente por Hassler Whitney, em 1934, em um seminário na Universidade de Princeton. Naquela época os algoritmos computacionais estavam apenas começando (FLOOD, 1956). Foi no final da década de 1940 que se têm notícias sobre a relação do PCV usando computação em uma programação linear por J. B. Robinson, 1949 (APPLEGATE et al., 2007). Houve um problema antecessor do PCV, denominado como Problema de Transportação, numa analogia ao transporte por meio do ônibus escolar, onde não se retorna ao ponto inicial de partida (DANTZIG, FULKERSON e JOHNSON, 1954). Foi Hopfield e Tank (1985) que aplicaram pela primeira vez redes neurais para solucionar o PCV no artigo intitulado '*"Neural" Computation of Decisions in Optimization Problems*.'

2.2 Aspectos do Problema do Caixeiro Viajante

Flood (1956) explica que o PCV deve encontrar o conjunto de permutação $P = \{1\ i_2\ i_3 \ldots\ i_n\}$ de inteiros de 1 até n que minimize a quantidade $a_{1,i_2} + a_{i_2,i_3} + a_{i_3,\,i\,4} + \ldots + a_{i_n,1}$, onde $a_{\alpha,\beta}$ são um conjunto de números reais que representam as distâncias. Mais precisamente, deve-se

existir $(n - 1)!$ possibilidades a serem levadas em conta. Se considerarmos que a distância de um ponto ao outro é a mesma nas duas direções, o problema passa então a ter $(n - 1)! / 2$ combinações diferentes. O problema consiste em construir um método eficiente capaz de minimizar o valor da permutação. Em termo prático, consiste em fazer um caixeiro viajante visitar uma cidade uma única vez, passando por todas elas, pré-definidas em uma matriz de custo simétrica contendo as distâncias entre cada cidade, e depois voltar para o ponto inicial. A Figura 2.1(a) ilustra uma distribuição de nós extraída da biblioteca TSPLIB, denominada bayg29, para ser aplicado ao PCV. Na Figura 2.1(b) tem-se o melhor percurso para essa instância.



(a) (b)

Figura 2.1 – (a) Distribuição de vinte e nove "cidades" para o caixeiro viajante visitar uma única vez , (b) melhor percurso da rota percorrida pelo caixeiro viajante

Matematicamente, o PCV pode ser formulado como:

Minimizar $$z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}\,, \qquad (2.1)$$

Sujeito a $$\sum_{i=1}^{n} x_{ij} = 1\,, \qquad j = 1, 2, ..., n, \qquad (2.2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, 2, ..., n, \qquad (2.3)$$

$$x_{ij} \in \{0,1\} \qquad i, j = 1, 2, ..., n \qquad (2.4)$$

$\tilde{x}$ forma um ciclo Hamiltoniano (2.5)

onde c_{ij} e x_{ij} são, respectivamente, o custo e a variável de decisão associados à atribuição dos elementos *i* para a posição *j*. O vetor $\tilde{x}$ tem todo o percurso da rota ao longo de todos os nós, mais a posição inicial de partida, formando a solução do PCV.

A função objetivo (2.1) representa o custo total a ser minimizado. O conjunto de restrições em (2.2) garante que cada posição de *j* é ocupada somente por uma cidade, enquanto que no conjunto de restrições em (2.3) garante o mesmo para as cidades posicionadas em *i*. O conjunto de restrições de (2.4) representa a integridade das variáveis de restrições zero-um x_{ij}, e podem ser substituídas pela restrição do tipo que se segue (AHUJA, MANGNANTI e ORLIN, 1993):

$$x_{ij} \geq 0, \qquad i, j = 1, 2, ..., n. \tag{2.6}$$

A restrição (2.5) assegura que, no final da rota, cada cidade será visitada uma única vez e que não serão formadas sub-rotas.

Hung e Wang (2003) descreveram o problema em termos de matriz. De (2.1) a (2.5) segue:

Minimizar $\boldsymbol{z = c^T x}$, (2.7)

Sujeito a $\boldsymbol{Ax = b}$, (2.8)

$x_{ij} \geq 0, \qquad i, j = 1, 2, ..., n,$ (2.9)

$\tilde{x}$ forma um ciclo Hamiltoniano (2.10)

Em (2.7) o termo $c^T = (c_{11}, c_{12}, ..., c_{1n}, c_{21}, c_{22}, ..., c_{2n}, ..., c_{n1}, c_{n2}, ..., c_{nn})$ é o vetor de coeficientes de custo, $x = (x_{11}, x_{12}, ..., x_{1n}, x_{21}, x_{22}, ..., x_{2n}, ..., x_{n1}, x_{n2}, ..., x_{nn})^T$ é o vetor de decisão, *b* é um vetor de tamanho 2*n* cujos elementos valem 1, e a matriz *A* tem a seguinte forma geral:

$$A = \begin{pmatrix} I & I & I & \dots & I \\ B_1 & B_2 & B_3 & \dots & B_n \end{pmatrix} \in \Re^{2n \mathrm{X} n^2} \qquad (2.11)$$

onde I é uma matriz identidade $n \times n$ e B_i, para $i = 1, 2, \ldots, n$, são n x n matrizes de zeros exceto a i-ésima linha que possui 1 em todas as posições.

2.3 Aspectos de Redes Neurais Artificiais

Partindo do princípio de que todas as capacidades cognitivas que um ser humano é dotado derivam das interações de cada neurônio com seus vizinhos no sistema nervoso, pode-se ter uma pista do potencial que o estudo das RNs pode oferecer. Pode-se encontrar aplicações nos mais diversos campos da ciência visando solucionar os mais variados problemas. Por meio de um modelamento matemático adequado das interações neurais é possível conseguir aplicações em sistemas autônomos com capacidade adaptativa como em:

- Sistemas de controle de aeronaves em situações críticas;
- Classificação de sistemas estelares;
- Identificação de padrões como influências em campanhas de marketing;
- Análise de imagens de satélite para áreas de desmatamento e focos de incêndio;
- Reconhecimento e síntese de voz.

Nos dias de hoje sistemas inteligentes são utilizados em elevadores, onde não se percebe a partida ou chegada; em eletrodomésticos, para aumentar o desempenho e reduzir o desperdício, etc. Atualmente uma RNA é capaz de resolver muitos problemas complexos de engenharia, baseando-se em amostras para o aprendizado. Sua capacidade de generalização a partir de amostras de treinamento a transforma em uma solução tangível para sistemas cuja dinâmica de funcionamento não seja totalmente conhecido, ou haja muitas variáveis difíceis de prever. Com a descoberta crescente de novas outras técnicas de inteligência

computacional, tais como a lógica fuzzy, algoritmo genético, colônia de formigas, entre as mais conhecidas, forma-se uma combinação de ferramentas de modelagem e computação que podem ser também denominadas como técnicas de *Soft Computing*, cujo objetivo de destaque desse conjunto de técnicas está em alcançar resultados aceitáveis com robustez, baixo custo e altas taxas de eficiência.

Os primeiros artigos publicados sobre RNs têm mais de 50 anos e começou com bastante entusiasmo, sendo que vários cientistas aderiram à moda na época. Porém, no ano de 1969, a neuro-computação sofreu um duro impacto com a publicação do livro *"Perceptrons – An Introducion to Computation Geometry*", dos autores Minsky e Papert (1969). Neste livro, Minsky e Papert fazem críticas e mostram a limitação de RNs com um único nível, como o *Perceptron* e o ADALINE, em aprender o padrão de uma função lógica simples como o XOR (OU-Exclusivo). A partir da publicação deste livro, houve então um período em que as pesquisas na área de RNA praticamente ficaram paralisadas até ser retomada na década de 80 com o impulso das publicações de Hopfield.

2.3.1 Histórico Resumido

O primeiro modelo matemático, que tenta reproduzir a resposta de um neurônio biológico aplicado na computação, que se tem notícia foi publicado em um artigo por McCulloch e Pitts (1943). Esse trabalho foi seguido por vários pesquisadores. Entre o lançamento do primeiro modelo de neurônio computacional e a publicação do livro que lançou dúvidas sobre seu potencial, intitulado *"Perceptrons – An Introduction to Computation Geometry"*, dos autores Minsky e Papert (1969), houve vários pesquisadores que desenvolveram o trabalho de modelamento matemático baseado no neurônio biológico. Muitas topologias surgiram baseadas nos modelos desenvolvidos e, consequentemente, algoritmos de aprendizados novos. Entre as publicações que se seguiram as mais relevantes

são a de Frank Rosenblatt (1959) que desenvolveu o primeiro neuro-computador, chamado de *Mark I Perceptron*, idealizando o modelo básico do *Perceptron*. Esse modelo chamou a atenção pela capacidade de reconhecer padrões. A rede ADALINE (*ADAptive LINear Element*) e posteriormente MADALINE (*Múltipla ADALINE*), desenvolvida por Widrow e Hoff (1960), baseava-se no algoritmo de aprendizado chamado "Regra Delta", também conhecida como algoritmo de aprendizado LMS (*Least Mean Square*).

Após esse período de novidades, com a publicação do livro de Minsky e Papert (1969), onde eles fazem duras críticas demonstrando o problema do ou-exclusivo, onde nenhuma rede de camada simples seria capaz de aprender esse padrão reproduzindo o mesmo resultado dessa porta lógica, houve um período em que as pesquisas na área de redes neurais artificiais praticamente ficaram paralisadas.

No passar dos anos, à medida que os computadores foram evoluindo, o interesse por RNA foi aos poucos retomando, voltando com força na década de 80 com algoritmos de aprendizado mais eficientes e com a publicação do artigo sobre o trabalho do físico John Hopfield, intitulado *"Neural Network and Physical Systems with Emergent Collective Computational Abilities"* (1982).

A quebra do estigma causado pelo problema do ou-exclusivo veio, ainda na década de 80, com a publicação do livro intitulado *"Parallel Distributed Processing",* dos autores Rumelhart, Hinton e Williams (1986). Neste livro, os autores desenvolveram um algoritmo que permite ajustar os pesos em uma rede com mais de uma camada, solucionando assim o antigo problema de aprendizado dos padrões da função lógica ou-exclusivo. O algoritmo, chamado de *"Backpropagation"*, reascendeu ainda mais as pesquisas em redes neurais artificiais.

Mais recentemente, pode-se destacar importantes trabalhos na área de RNA, tais como em robótica (FROLOV et al., 2002), em controle de processos (CHEN e HUANG, 2004), em

reconhecimentos de padrões (DAI e WANG, 2007), e em processamento de sinais (KING et al., 2002).

2.3.2 Modelo do Neurônio Biológico e Modelo do Neurônio Artificial

O neurônio biológico constitui praticamente todo o sistema nervoso, unidos entre si por conexões sinápticas. Estes são células que podem ser divididas em três partes: o corpo da célula, onde se encontra seu núcleo; os dendritos, responsáveis por receber os impulsos nervosos provenientes de outros neurônios; e o axônio, por onde passam os impulsos nervosos para os dendritos dos neurônios seguintes.

Os impulsos nervosos acabam trazendo a informação celular que resulta da interação de outros impulsos, vindos por meio de outros axônios, e conduzidos até o corpo celular onde a mesma é processada gerando assim novos impulsos. O corpo do neurônio mede alguns milésimos de milímetros e os dendritos apresentam poucos milímetros de comprimento. A Figura 2.2 ilustra uma célula de neurônio com suas partes descritas anteriormente. O axônio é a parte mais longa da célula. A região onde axônio e dendrito se encontram formando uma interface é conhecida como sinapse. São pelas sinapses que os neurônios se unem, formando as redes neurais.

As sinapses funcionam como válvulas, sendo capazes de controlar a transmissão de impulsos, isto é, o fluxo da informação entre os neurônios da rede neural. O efeito das sinapses é variável, e é esta variação que dá ao neurônio a capacidade de adaptação.

Dentro do corpo celular do neurônio há reações químicas causadas pelos impulsos que na prática compara todos os impulsos nervosos provenientes das conexões sinápticas, tantas quantas existirem, e as retransmitem para as próximas células neurais. As comparações entre todos os impulsos são submetidos a um limiar; se um certo porcentual desses sinais em um intervalo curto de tempo é suficientemente alto, a célula “dispara”, produzindo um impulso

que é transmitido para as células seguintes. Este sistema simples é responsável pela maioria das funções realizadas pelo cérebro humano, em que aplicado à média de 10^{11} neurônios interligados por 10^{14} sinapses resulta nas capacidades cognitivas que todos conhecemos. Essa capacidade de solucionar funções complexas surge com a operação em paralelo de todos estes neurônios e sinapses.

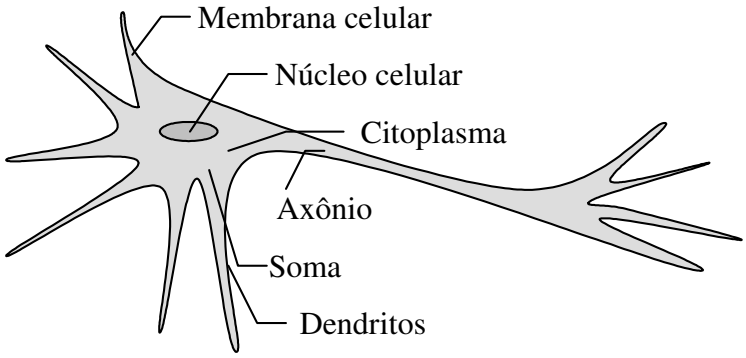


Figura 2.2 – Célula de neurônio com as três partes que a divide: Corpo celular, dendritos e axônio.

O neurônio artificial é fruto da publicação do primeiro artigo sobre um modelo de neurônio computacional proposto por McCulloch e Pitts em 1943, em que se sabia muito pouco sobre os detalhes da célula neural e por isso o modelo inicial era bem simplificado. Mesmo assim, até nos dias de hoje, o mesmo modelo é utilizado na maioria das arquiteturas neurais. O resumo da descrição matemática que foi descrita naquele artigo é a seguinte: supondo uma função com n entradas, ou seja, $x_1, x_2, ..., x_n$, que representam os dendritos. Sua saída sendo y, representando o axônio. Para simular o comportamento das sinapses, definem-se os pesos $w_1, w_2, ...,w_n$, cujos valores são alterados de acordo com o algoritmo de treinamento e podem ser positivos ou negativos. A Equação (2.12) representa o efeito de uma sinapse particular i no neurônio dado pela multiplicação de x_i por w_i:

$$y = g(\sum_{i=1}^{n} w_i x_i + b) \tag{2.12}$$

onde:

- n é o número de entradas do neurônio;

- w_i é o peso associado com a i-ésima entrada;
- b é o limiar associado ao neurônio;
- x_i é a i-ésima entrada do neurônio;
- $g(.)$ é a função de ativação do neurônio;
- y é a saída do neurônio.

Fazendo uma analogia com uma célula neural, pode-se chamar as variáveis x_i como se fossem os terminais de entrada, ou seja, a função que um dendrito exerce. Cada célula possui apenas uma saída, que no caso aqui seria a variável y. Os pesos w_i são usados pelos terminais de entrada do neurônio. Os pesos sinápticos influenciam a entrada resultando na função das sinapses. A Figura 2.3 ilustra o diagrama de distribuição das entradas, saídas e os demais elementos envolvidos nas interações que resulta na resposta final de uma célula neural modelada matematicamente.

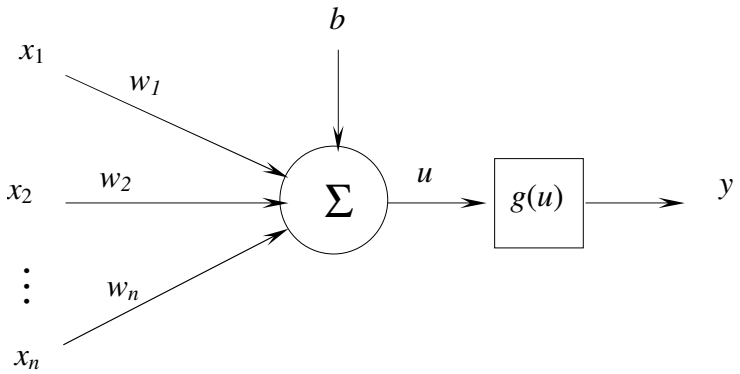


Figura 2.3 – Diagrama de distribuição das entradas, saída e demais elementos envolvidos na resposta de um neurônio mediante excitações.

Conforme ilustrado na Figura 2.3, em conjunto com a Equação (2.12), os sinais são apresentados às entradas por meio das variáveis x. Cada sinal é multiplicado por um peso que representa o seu nível de relevância na saída da unidade; em seguida, é feita uma soma ponderada dos sinais, resultando assim em um nível de atividade; caso este nível de atividade exceda certo limite, a unidade produzirá uma saída. Esse limiar é especificado pela função de

ativação que na Figura 2.3 é ilustrada pela função *g(u)*. A função de ativação pode assumir os seguintes tipos de valores:

- Binários (0 ou 1);
- Bipolares (-1 ou 1);
- Reais.

Durante a etapa de treinamento das redes neurais artificiais, ao final de cada treinamento, podem-se alterar os principais parâmetros da rede de modo a buscar uma configuração de rede eficiente. Um dos parâmetros que pode ser alterado é a função de ativação. As principais funções de ativação utilizadas são:

i) Função Degrau (binária)

A função degrau é definida por:

$$g(u) = \begin{cases} 1, & \text{se } u \geq 0 \\ 0, & \text{se } u < 0 \end{cases} \qquad (2.13)$$

ou seja, se o nível de atividade interna do neurônio for um valor positivo ou igual à zero, a saída assumirá o valor um; caso contrário, assumirá o valor zero. A Figura 2.4 mostra a representação gráfica desta função.

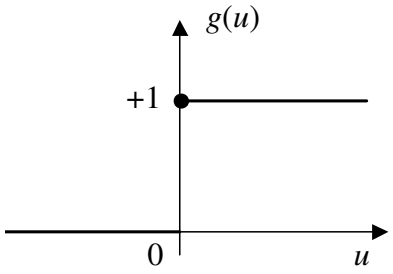


Figura 2.4 - Função degrau (binária).

ii) Função Degrau (bipolar)

Esta função é definida por:

$$g(u) = \begin{cases} +1, & \text{se } u \geq 0 \\ -1, & \text{se } u < 0 \end{cases} \qquad (2.14)$$

Assim como a função de degrau binária, se o nível de atividade interna do neurônio for um valor positivo ou igual à zero, a saída assumirá o valor um; porém, neste caso, se o nível de atividade interna do neurônio for um valor negativo, a saída do neurônio assumirá o valor -1. A Figura 2.5 mostra a representação gráfica desta função.

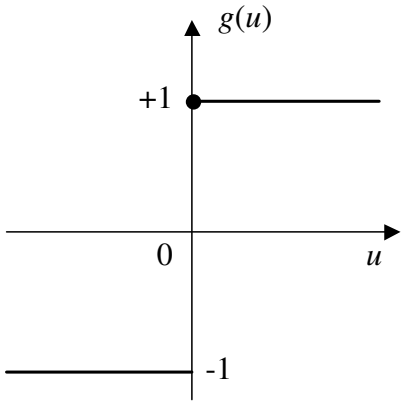


Figura 2.5 - Função degrau (bipolar).

iii) Função Rampa

A função rampa é definida por:

$$g(u) = \begin{cases} +1, & \text{se } u \geq a \\ u, & \text{se } -a < u < a \\ -1, & \text{se } u \leq -a \end{cases} \qquad (2.15)$$

Nesta função, os valores máximo e mínimo da saída do neurônio são +1 e –1. Os valores de saída poderão variar de acordo com a função $g(u)$ no intervalo definido em $[-a, a]$, sendo $a \neq 0$. Na Figura 2.6 tem-se uma representação gráfica desta função.

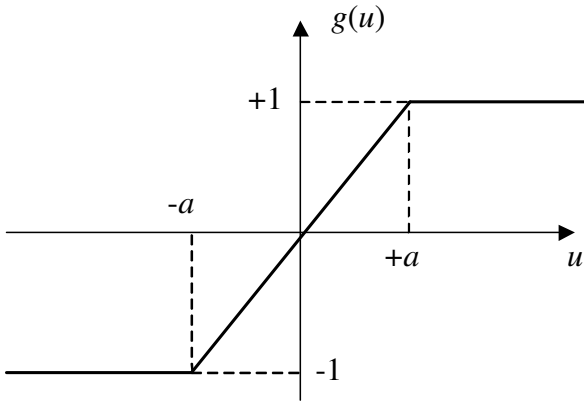


Figura 2.6 - Função rampa.

iv) Função Sigmóide

A definição da função sigmóide é dada por:

$$g(u) = \frac{1}{1 + e^{(-\beta u)}} \tag{2.16}$$

Esta é uma das respostas mais próximas do real que se obtém em um neurônio biológico. O parâmetro β desta função define a suavidade ou grau de inclinação da curva da função sigmóide. A saída no neurônio assumirá valores entre 0 e 1. A Figura 2.7 mostra a representação gráfica desta função.

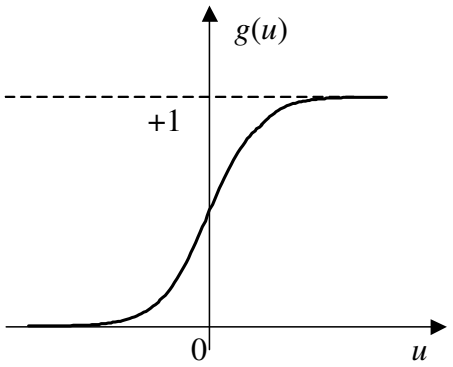


Figura 2.7 - Função sigmóide.

v) Função Tangente Hiperbólica

A função tangente hiperbólica é defina por:

$$g(u) = \tanh(u) = \frac{1 - e^{-u}}{1 + e^{-u}} \tag{2.17}$$

Usando a função tangente hiperbólica, a saída do neurônio assumirá valores positivos e negativos, oscilando no domínio de –1 a 1. Graficamente, esta representação pode ser ilustrada por meio da Figura 2.8.

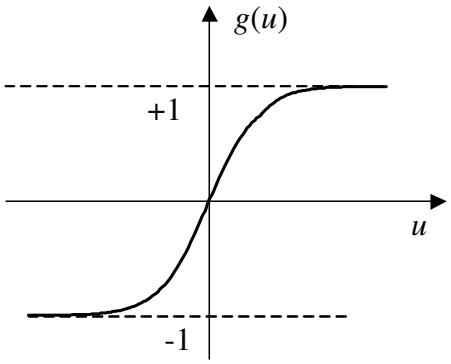


Figura 2.8 - Função tangente hiperbólica.

2.3.3 Topologia de Redes Neurais Artificiais

Existem diversos tipos de topologias de redes neurais. As topologias estão diretamente ligadas à configuração das interligações entre os neurônios. Portanto, o tipo do problema a se resolver, incluindo aí a forma de como os pesos dos neurônios será ajustado, está diretamente relacionado à topologia adotada. O algoritmo de treinamento da rede dependerá, assim, da topologia.

Uma rede neural multicamada pode ser dividida em três partes:

- **Camada de entrada** – responsável em receber as informações externas à RNA, sendo o local onde as amostras são apresentadas;
- **Camadas intermediárias ou ocultas** – nessas camadas ocorrem a extração e/ou classificação das características dos padrões das amostras apresentadas na camada de entrada. Todo o processamento é feito paralelamente. A maioria do processamento ocorre nessa camada;
- **Camada de saída** – todo o processamento resultante nas camadas anteriores resulta nessa camada, onde a resposta é apresentada ao exterior da RNA.

A Figura 2.9 apresenta uma topologia genérica de RNA com destaque às três partes das camadas descritas anteriormente.

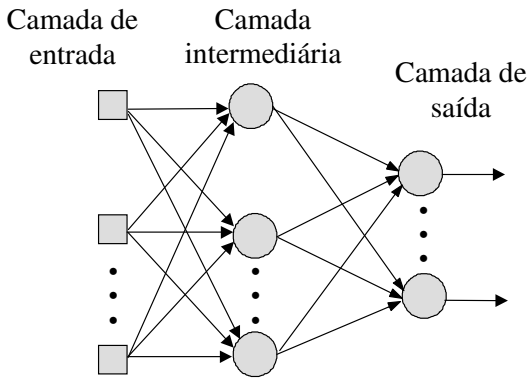


Figura 2.9 – Topologia genérica de RNA e as três partes de camadas possíveis que ela apresenta.

Entre tantos tipos de topologias de RNA, também conhecidas por arquiteturas, pode-se citar três principais:

- Redes *FeedForward* (camada única);
- Redes *FeedForward* (multicamadas);
- Redes Recorrentes.

As redes *FeedForward* com camada única é, ela própria, tanto a camada de entrada quanto a camada de saída, pois possui uma única camada de neurônios, conforme ilustra a Figura 2.10. Esta é a configuração usada pelo *Perceptron* e *ADALINE*. Estas redes são aplicadas em reconhecimento de padrões e memórias associativas.

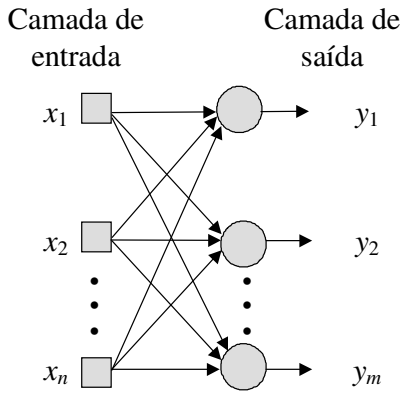


Figura 2.10 – Configuração de rede *Feedfoward* com camada única contendo *n* entradas e *m* saídas.

A rede *FeedFoward* multicamada é a topologia clássica onde as três partes estão presentes, a camada de entrada, a camada intermediária ou camada escondida e a camada de saída, conforme ilustra a Figura 2.11. As etapas do processamento dos sinais seguem pela camada de entrada, depois a camada intermediária classifica os sinais que, em seguida, são apresentados pela camada de saída. Geralmente, esta topologia é aplicada em reconhecimento de padrões, e como aproximador universal de funções, uma vez que se pode aproximar funções não-lineares. As principais redes que usam esta topologia são o *Perceptron* multicamadas e a Função Base Radial.

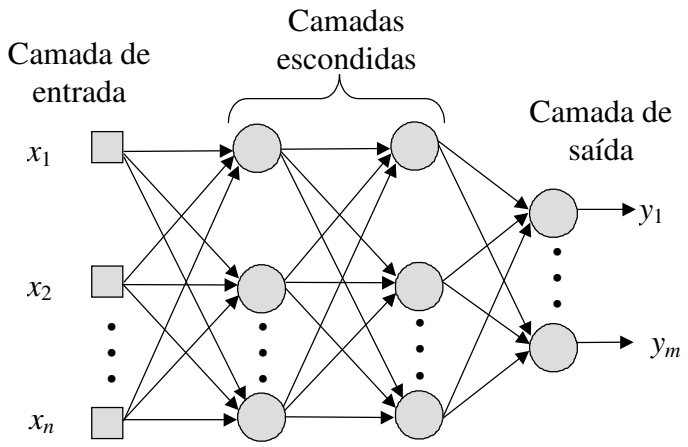


Figura 2.11 – Topologia de rede *FeedFoward* multicamada com as três partes de camada que esta constitui: camada de entrada, camada intermediária ou escondida e camada de saída.

As redes recorrentes possuem realimentação entre os neurônios de camadas diferentes, conforme ilustra Figura a 2.12. Estas redes possuem capacidade de memória, reproduzindo seu comportamento com relação aos padrões anteriormente apresentados e, desta forma, possibilitando processar seqüências de informações. São aplicadas em sistemas combinatórios, sistemas dinâmicos, séries temporais, sistemas de predição, reconhecimento de padrões e controle. Os modelos mais adotados neste tipo de topologia são as redes *Perceptron* com realimentação e redes de Hopfield.

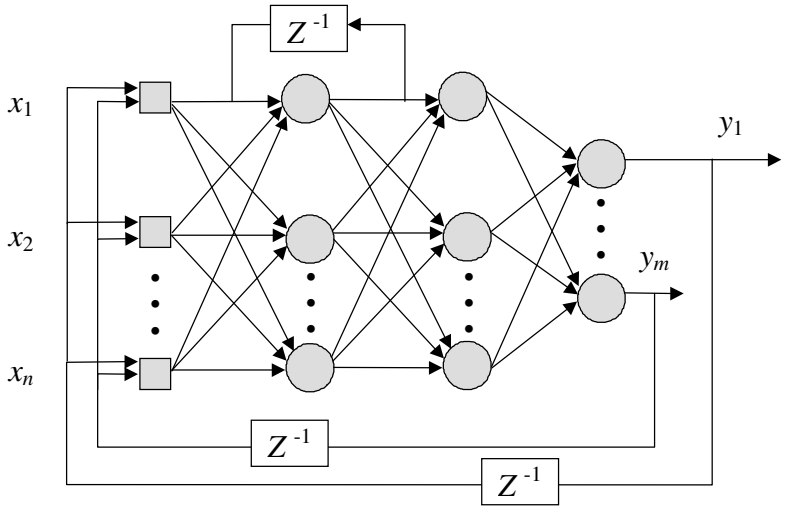


Figura 2.12 – Topologia de rede recorrente com as realimentações entre diferentes camadas da rede.

2.3.4 Treinamento de Redes Neurais Artificiais

Um dos fatores que tornam RNAs factíveis é a capacidade de generalização, ou seja, após o treinamento da rede, baseado em amostras, elas conseguem encontrar soluções para amostras não apresentadas no treinamento. Essa generalização é devida à capacidade que a RNA tem de aprender.

Os dois principais processos de aprendizagem são: Aprendizado Supervisionado e Aprendizado Não Supervisionado.

2.3.4.1 Aprendizado Supervisionado

O processo de aprendizado supervisionado ocorre com a apresentação de amostras, cuja resposta é conhecida, na camada de entrada e, depois, comparada entre a resposta da rede e os resultados esperados. Durante esta fase a rede extrai as características do processo ou sistema que lhe está sendo apresentado. O aprendizado vai ocorrendo à medida que os pesos são ajustados, sendo essa a forma que uma RNA usa para armazenar conhecimento. A etapa de treinamento só termina quando a RNA consegue generalizar soluções, dentro de uma margem satisfatória de erro, para o problema apresentado. O algoritmo de aprendizagem é

determinante para o tipo de problema a se resolver, pois os parâmetros da rede são alterados de acordo com um conjunto de regras definidas por ele. Atualmente, existem vários desses algoritmos usados no treinamento, e o que muda de um para o outro é o modo em que os ajustes dos pesos sinápticos são feitos.

As épocas são os pontos divisores entre uma iteração de ajuste dos pesos sinápticos seguida da comparação com a resposta desejada, considerando para tanto toda a base de dados de comparações. Numa mesma época, todas as amostras disponíveis são apresentadas para treinamento.

Dentre as amostras disponíveis cujas respostas são conhecidas, costuma-se separar um porcentual para o treinamento, sendo que o restante serve para a validação do sistema, pois às vezes, a generalização pode ser deficiente e as amostras reservadas para esse fim podem detectar essa falha.

No Aprendizado Supervisionado as amostras são apresentadas às entradas juntamente com o resultado pretendido. Um algoritmo externo à RNA é responsável em ir ajustando os pesos sinápticos, de modo que a resposta da rede vai se aproximando com o resultado pretendido até que tais saídas atinjam uma tolerância mínima de desvio. A generalização é validada, com as amostras remanescentes separadas, considerando um porcentual da quantidade total de amostras disponíveis, de acordo com os parâmetros estabelecidos.

2.3.4.1 Aprendizado Não Supervisionado

A inexistência de um agente supervisor, ou algoritmo externo para ajustar os pesos, é a marca determinante para o sistema de Aprendizado Não Supervisionado. Nesse sistema os pesos vão se agrupando de acordo com as características dos subconjuntos dos estímulos apresentados às regiões das camadas de entrada. Assim, os pesos vão sendo ajustados gradualmente, sendo que a resposta da rede varia de acordo com a natureza dos padrões

apresentados. Portanto, quanto mais "evidente" forem as particularidades marcantes da população de entrada, mais "fácil" será a aprendizagem da rede, e vice-versa.

As redes que adotam o neurônio do tipo *Perceptron* também podem ser recorrentes, que é o caso dessa dissertação. Nessa configuração de rede as camadas de entrada recebem os estímulos que repassam às camadas intermediárias, onde as informações são extraídas e preparadas para que a camada de saída faça a codificação apropriada para o ambiente externo. Algumas dessas camadas, ou todas elas, podem ser direcionadas para as camadas anteriores, principalmente a camada de entrada. A regra de treinamento usada nessa dissertação, conhecida por Perturbação Simultânea, apresenta um tópico específico que trata da teoria da mesma, mais especificamente no Capítulo 3, na Seção 3.2 (Aspectos da Técnica de Perturbação Simultânea).

2.4 Topologias de Redes Neurais Aplicadas no Problema do Caixeiro Viajante

O PCV é um problema combinatorial de ordem exponencial que possui vários mínimos locais de convergência. A primeira abordagem sobre o PCV usando RNA foi feita por Hopfield e Tank (1985). Ela foi baseada na minimização da função de energia ou função de Lyapunov.

Embora qualquer disposição dos nós tenha padrões característicos, que diferencie entre as distâncias percorridas mais curtas, existem topologias de RNA mais adequadas para identificar esses padrões, com destaque para redes recorrentes, redes neurais de Hopfield (RNH) e mapas auto-organizáveis de Kohonen (MAOK). Entretanto, até mesmo uma rede do tipo *Perceptron* com uma camada pode ser também empregada nesse problema (BENINI e SILVA, 2008).

A principal característica de RNH é sua tendência de estabilizar em um ponto de mínimo local de uma função de energia bem definida, sendo este um dos motivos dela ser bastante aplicada em problemas de otimização. Essa característica é também bem apropriada ao PCV. Uma ressalva para a adoção dessa rede é que a mesma costuma falhar na convergência de uma solução válida, pois muitas vezes o mínimo local em que ela convergiu fica aquém da solução ideal (WILSON e PAWLEY, 1988).

Xu e Tsai (1991) foram os primeiros a introduzir adaptabilidade nas redes de Hopfield. Limitando a quantidade de mínimos locais da rede, de acordo com a quantidade e posição das cidades apresentadas, as chances de convergência para o ponto de mínimo global do problema são, portanto, aumentadas. No entanto, essa técnica era eficaz apenas para uma quantidade limitada de número de cidades.

Talaván e Yáñez (2006) fizeram uma abordagem para o problema quadrático da mochila (PALMEIRA, 1999). Este problema pode ser visto como uma generalização de problemas combinatoriais bem conhecidos, inclusive o PCV. Eles usaram uma rede de Hopfield contínua (RHC), baseada na função de Lyapunov, em que a energia decresce à medida que o mínimo local é atingido. Para isso foi proposta uma nova função de energia de modo que qualquer programação com restrição linear 0-1 com função objetivo quadrática (COELHO e MARIANI, 2006) possa ser resolvida. Sua técnica foi inspirada no problema quadrático da mochila, incluindo casos particulares conhecidos do PCV e problema de atribuição quadrática (COELHO e TAVARES NETO, 2004). A função de energia proposta, aplicada em RHC, aliada aos procedimentos de ajuste dos parâmetros e a um conjunto de condições analíticas, é generalizada para os problemas de natureza combinatorial, garantindo a possibilidade de atingir pontos estáveis em mínimos locais dos problemas mapeados. Os resultados teóricos foram testados em cima do PCV com até 1002 cidades, utilizando para

tanto a base de dados TSPLIB de Reinelt (1991). Eles concluem que típicas dificuldades de problemas de otimização combinatorial podem ser eficientemente resolvidas com a RHC.

Antes, porém, Talaván e Yáñez (2002) já haviam usado RHC para o PCV, observando-se que uma das maiores desvantagens era a dificuldade de se obter soluções tangíveis e as arbitrariedades envolvidas nas configurações baseadas na tentativa e erro dos ajustes nos parâmetros do modelo. Eles conseguiram evitar esse inconveniente a partir da introdução de um conjunto de condições analíticas, garantindo então que qualquer ponto de equilíbrio da RNH caracteriza uma rota para o PCV, podendo assim ser resolvido qualquer conjunto de cidades com o ajuste destes parâmetros.

Foi a partir do mesmo ponto de vista, em relação às falhas de uma RNH de apresentar resultados falhos para o PCV, que Tan, Tang e Ge (2005) desenvolveram seu trabalho. Eles associaram esse comportamento a problemas de configuração dos parâmetros da rede. Para contornar tal problema foram estabelecidos alguns critérios de estabilização que garante a convergência para soluções válidas a fim de suprimir as falhas. Dessa forma, um aprimoramento da formulação do mapeamento do PCV para RNH foi criado. Com o método sistemático desenvolvido, a convergência para soluções válidas, suprimindo estabilizações espúrias e analisando a estabilidade dinâmica no hipercubo unitário, um método teórico para selecionar os parâmetros de penalidade é obtido. Com as configurações dos parâmetros propostos, a qualidade das soluções foi melhorada em comparação com os resultados de Talaván e Yáñez (2002).

Wang, Tang e Cao (2002) propuseram, com resultados de 100% de otimização para o PCV com poucos nós, uma função de energia para RNH. Eles utilizaram um método de aprendizagem que ajusta o balanço entre os termos de restrição e os termos de custo da função de energia de tal forma que o mínimo local da rede seja atingido, fazendo-se então com que a RNH continue atualizando o gradiente em direção descendente da energia.

Siqueira, Scheer e Steiner (2007) usaram redes recorrentes com o princípio "Vencedor Leva Tudo" (VLT) para solucionar o PCV. A vantagem dessa técnica, segundo os autores, é a facilidade de implementação de seu algoritmo e a baixa complexidade computacional, conseguindo bons resultados e a possibilidade de resolver problemas simétricos e assimétricos. A técnica VLT, usada em conjunto com a arquitetura RNR, proposta nesse trabalho, acelera a convergência para o mínimo local, corrigindo eventuais problemas, devido a múltiplas soluções, que às vezes se encontram próximas uma das outras. A violação das restrições do PCV são medidas até, após certo número de iterações, o valor mudar muito pouco. Isto evidencia que as restrições do problema estão sendo satisfeitas. É nesse momento que VLT é aplicado.

Por outro lado, o algoritmo usando MAOK é um algoritmo de aprendizagem não supervisionado. O processo de aprendizagem está relacionado com a topologia dos dados de entrada. A principal característica de se usar MAOK, para o PCV, é que a rota final é determinada através dos neurônios, que possuem pesos mais perto do par de coordenadas relacionadas, a cada cidade do problema. Devido a essa analogia, muitos trabalhos usando essa arquitetura em problemas combinatoriais foram realizados.

Leung, Jin e Xu (2004) se inspiraram em MAOK para desenvolver uma RNA para o PCV Euclidiano, denominada Mapas Auto-Organizáveis Expandidos (MAOE). Em cada iteração de aprendizagem os neurônios próximos da cidade de entrada são ressaltados; neste meio tempo, eles são "empurrados" em direção ao fecho convexo (GONZALES e WOODS, 2000) das cidades cooperativamente. Ao mesmo tempo, ele preserva as propriedades do fecho convexo, adquirindo mais nós entre os vizinhos do mesmo, que vai se destacando nas iterações. Esta estratégia de funcionamento resulta em soluções próximas do ideal. Eles fizeram um minucioso estudo teórico e empírico dos resultados, usando *benchmark* com diferentes quantidades de nós (entre 50 e 2400 cidades), concluindo então por meio dos

experimentos a superioridade que o MAOE apresentou em relação a diversos tipos de MAOK.

Zhang et al (2006) usaram MAOK com foco em um método eficiente de inicialização, investigando algumas questões:

1) Ele converge para uma solução tangível?
2) Existe uma melhor regra de adaptação de parâmetros para o algoritmo?
3) Qual é o efeito de diferentes seqüenciamentos de nós ao longo de um caminho inicial?
4) Como será organizado o índice dos neurônios vencedores e a ordem inicial das cidades?

Apesar de vários tipos de algoritmos para MAOK terem sido abordados para resolver o PCV na literatura, o propósito nesse trabalho foi estudar um método eficiente de inicialização e de definição na regra de adaptação dos parâmetros, visando alcançar a melhor qualidade de resultados com a mais rápida convergência possível. Por isso, recomenda-se que a ordem das cidades seja apresentada aleatoriamente na inicialização, a fim de aumentar a robustez, pois a convergência para uma solução está diretamente relacionada a isso. Os resultados obtidos por eles foram animadores. Para um conjunto de 12 diferentes exemplos resolvidos do PCV, o desvio médio apresentado em relação ao resultado ótimo foi de 2,4372% contra 3,69% (FREDERICO, ADIAO e JOSE, 2003), utilizando para tanto a mesma base de dados de comparação encontrado na literatura a respeito de MAOK.

2.5 Topologias Híbridas Aplicadas no Problema do Caixeiro Viajante

Cada vez mais cresce o uso de outros conceitos de inteligência computacional, aliados a RNA, para melhorar o desempenho de convergência para localizar pontos mínimos para o

PCV. Entre os mais comuns pode-se citar algoritmo genético (AG), colônia de formigas (CF) e lógica *fuzzy*.

Teoh et al (2008) propuseram uma abordagem de RNR de limiar linear. As coordenadas das cidades, que compõem o PCV, são mapeadas em uma RNR com camada única, inserindo as restrições do problema diretamente dentro da dinâmica da rede. Ela difere da clássica RNH no método de atualização dos pesos, bem como da função de ativação. Aliado ao AG, para ajustar os parâmetros da RNR proposta, essa combinação assegura uma convergência estável da RNR para diferentes problemas. Os autores concluem que os resultados obtidos nessa combinação, com atualização dos pesos assíncrona, foi superior ao clássico método de solução com RNH, para alguns casos de exemplos do PCV. As melhoras nesses resultados foram creditadas ao método de limiar linear dos neurônios, que evitam níveis saturados, sendo esta uma característica de muitas funções de ativação.

Sivagaminathan e Ramakrishnan (2007) adicionaram em uma RNA o conceito de CF para identificar subconjuntos de variáveis, os quais sejam correlacionados para aplicação em classificação e modelamento. Eles conseguiram associar o seu trabalho, cujo domínio era para diagnósticos médicos, ao PCV na parte pertinente à CF. Foram comparadas as três categorias do espaço de procura da solução ótima (DORIGO e GAMBARDELLA, 1997) ao trabalho deles, ou seja:

i) O melhor caminho – associada às melhores características repetidas no melhor subconjunto;
ii) O caminho testado – as características testadas na iteração anterior;
iii) Os caminhos não usados – as características que nunca estiveram no melhor subconjunto.

Créput e Koukam (2008) fizeram um estudo em cima da hibridização de MAOK com um algoritmo evolutivo para resolver o PCV. A dinâmica evolucionária consiste em intercalar o MAOK, executando com um operador de mapeamento, uma adaptação evolutiva e um operador selecionador. O mapeamento no MAOK é baseado nas cidades com as coordenadas mais próximas com movimentos simples apresentados no plano. Os testes foram feitos em PCV padrão e demonstraram-se melhores em relação à qualidade das soluções e tempo de computação do que outras RNA abordadas anteriormente na literatura. A base de dados usada para os testes foi extraída de 91 publicações que usavam exemplos de PCV com até 85900 cidades.

CAPÍTULO 3

Rede Recorrente com Perturbação Simultânea Aplicada no Problema do Caixeiro Viajante

3.1 Introdução

O conceito de rede neural vem se aprimorando nos últimos tempos. Atualmente, existem técnicas robustas de treinamento e aplicação da mesma, sendo que uma exceção para essa regra são as redes neurais recorrentes (RNR). Isso se dá, principalmente, devido às diferentes possibilidades de combinações de topologias na realimentação para camadas anteriores. Por isso, a criação de uma teoria para aplicação genérica é praticamente inviável, principalmente no que diz respeito ao treinamento dos pesos sinápticos em função de complicações na determinação de critérios de convergência (MAEDA e WAKAMURA, 2005). Por causa desses problemas é usual criar uma rede recorrente para aplicações específicas e, a partir daí, desenvolver toda a teoria para a aplicação, tendo como exemplo mais conhecido a famosa RNH aplicada principalmente em reconhecimento de imagens.

A topologia de RNR adotada nesse trabalho foi proposta por Wang (1992), conhecida como Rede Recorrente de Wang (RRW), que foi criada originalmente para resolver o Problema da Designação e, posteriormente, foi adaptada, pelo mesmo autor, para resolver o Problema do Menor Caminho (WANG, 1996), sendo todos estes problemas combinatoriais. No entanto, Siqueira, Scheer e Steiner (2007) foram quem usaram a RRW para o PCV

associado ao princípio VLT. Sua topologia de realimentação se encaixa perfeitamente à matriz de custo das distâncias entre os nós, pois cada linha em combinação com cada coluna é usada no processamento da RN, conforme ilustra a Figura 3.1. Isso permite que todas as distâncias em relação ao nó central, do cruzamento entre linha e coluna, sejam levadas em consideração e comparadas.

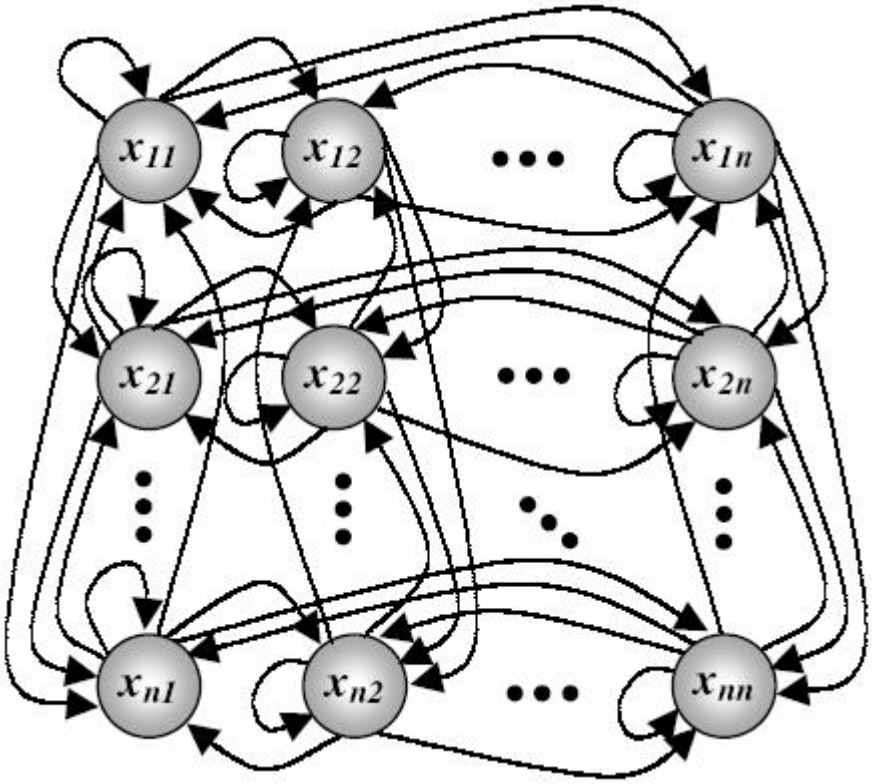


Figura 3.1 – Representação da estrutura da RRW (SIQUEIRA, 2005).

Um dos grandes problemas na aplicação da RRW é o ajuste dos parâmetros, tornando trabalhosa sua aplicação, pois esses parâmetros são determinados empiricamente. Devido a essas ressalvas, e também por questão de foco, adotou-se nesta dissertação somente a estrutura de realimentação da RRW, sem aplicar então o conceito de convergência e ajuste dos pesos sinápticos, tarefa esta que ficou a cargo da teoria que envolve o conceito de Perturbação Simultânea.

O termo "Perturbação Simultânea" (PS) surgiu pela primeira vez com Spall (1987). A idéia básica por trás do método proposto é uma extensão aprimorada do método da

aproximação estocástica de Kiefer-Wolfowitz (SPALL, 1992). Ele provou que esse método, por meio de um algoritmo, converge para um mínimo local de uma função de regressão com probabilidade igual a 1. A fim de garantir a convergência teórica, seu algoritmo requer condições estritas na perturbação, um coeficiente de ganho e uma forma de função de regressão. A teoria, juntamente com os resultados, demonstrou que o algoritmo pode ser significativamente mais eficiente do que o algoritmo padrão baseado em diferenças finitas para problemas matriciais de dimensões elevadas.

Embora o próprio Spall e Cristion (1992) tenham sido os primeiros a aplicarem o método de PS para treinamento de uma rede neural, foi Maeda que mais explorou tal método em treinamento de redes neurais, com destaque para redes recorrentes (MAEDA, HIRANO e KANATA, 1993). Devido às propriedades do algoritmo de estar mais bem adaptado ao funcionamento em paralelo, ele foi largamente explorado para ser implementado em hardware, sendo que com isso sofreu algumas simplificações para funcionar dentro das limitações intrínsecas pertinentes a esse tipo de tecnologia, principalmente para o FPGA.

3.2 Aspectos da Técnica de Perturbação Simultânea

O PCV pode ser considerado um problema de otimização multivariável. Essa característica vai de encontro com o método PS por ser o mesmo um processo de procura iterativa, pois não é possível uma análise matemática analítica.

Adicionalmente, se os pesos sinápticos de uma RNA são alterados simultaneamente, seguindo um gradiente de minimização do erro, o desempenho da convergência do mesmo é potencializado. A otimização multivariável motivou o interesse em algoritmos sem usar a informação do gradiente como direcionamento de convergência, como é o caso do treinamento em RNR, por exemplo. Antes, esses algoritmos são baseados em aproximações

para a formação do gradiente a partir de medidas feitas na função de perda. A função de perda é um termo adotado por Spall (1998) e representa a medição indireta do gradiente de uma função em relação aos parâmetros que estão sendo otimizados para zero.

O diferencial dessa técnica é minimizar a função de perda $L(\theta)$, sendo $\theta \in R^p$, $p \geq 1$. O algoritmo baseado em PS trabalha com iterações a partir de valores iniciais de θ, cujos valores iniciais são baseados em conceitos estatísticos, dependendo da natureza do problema a ser aplicado. O processo de iteração depende da aproximação para o gradiente $g(\theta) \equiv \partial L/\partial \theta$. A técnica possui uma vantagem, em relação a outros métodos de aproximação estocástica, no número de iterações para a convergência quando a função de perda possui ruído, em (3.1) indicado pela letra R; dessa forma, os resultados em cima da função de perda com ruído $y(\theta)$ estão disponíveis para quaisquer valores de θ, ou seja:

$$y(\theta) = L(\theta) + R. \qquad (3.1)$$

Em alguns casos o exato valor da função de perda pode ser obtido, o qual corresponde à ausência de ruído. A direção do gradiente $g(\theta)$, com ou sem ruído, não está disponível para medida. Nos casos onde mais de um ponto satisfaz $g(\theta) = 0$, pode significar que a convergência é para um mínimo local, pois geralmente não há garantia de convergência para um ponto máximo ou ponto de sela de $L(\theta)$.

De modo geral, algoritmos estocásticos com determinação de convergência, sem a necessidade de medir a direção do gradiente, apresentam propriedades de convergência similares àqueles baseados na medida da direção do gradiente, enquanto requerem somente medidas da função de perda. Os algoritmos baseados na direção do gradiente contam diretamente com medidas do gradiente da função de perda em relação aos parâmetros que estão sendo otimizados. Essas medidas tipicamente resultam em uma estimativa do gradiente.

A principal vantagem dos algoritmos livres do gradiente é que eles não requerem conhecimentos detalhados da relação funcional entre os parâmetros sendo otimizados e a

função de perda que está sendo minimizada, o que não é possível nos algoritmos baseados na direção do gradiente. Existe uma grande economia computacional para se calcular a função de perda em relação aos requerimentos para se calcular o gradiente. Há casos em que a observação da direção do gradiente é usada com considerável vantagem, incluindo análises de perturbações infinitesimais para otimizações, como é o caso do treinamento de redes neurais com *backpropagation*.

A forma recursiva de ajuste de θ é obtida segundo a Equação (3.2)

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \, \hat{g}_k(\hat{\theta}_k) \tag{3.2}$$

onde $\hat{g}_k(\hat{\theta}_k)$ é a estimativa do gradiente $g(\theta)$ na iteração $\hat{\theta}_k$, conforme comentado anteriormente, da medida da função de perda. Sob condições apropriadas, a iteração da Equação (3.2) convergirá para a solução global θ^* em algum sentido estocástico.

A parte essencial de (3.2) é a aproximação do gradiente $\hat{g}_k(\hat{\theta}_k)$. Todos os elementos de $\hat{\theta}_k$ recebem uma perturbação randômica ao mesmo tempo, sendo que daí origina o nome de Perturbação Simultânea. Com isso se obtém duas medidas de $y(\cdot)$, em que cada componente de $\hat{g}_k(\hat{\theta}_k)$ é formado da média envolvendo a diferença de duas medidas correspondentes e o componente individual no vetor de perturbação. A Equação (3.3) ilustra como se obtém a estimativa do gradiente:

$$\hat{g}_k(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k \cdot \Delta_k) - y(\hat{\theta}_k - c_k \cdot \Delta_k)}{2 \cdot c_k \cdot \Delta_k} \tag{3.3}$$

$$\Delta_k = \begin{pmatrix} \Delta_{k1} \\ \Delta_{k2} \\ . \\ . \\ . \\ \Delta_{kp} \end{pmatrix} \qquad (3.4)$$

onde os vetores que constituem a expressão (3.4) são gerados aleatoriamente e independentemente, em torno de zero. Eles constituem o vetor da perturbação simultânea e o mais indicado pelo autor e seguido nesse trabalho é a distribuição de Bernoulli ±1 com probabilidade de ½ para cada resultado que pode ser 1 ou -1.

Vale observar que o número de medidas da função de perda $y(\cdot)$ necessita de apenas duas amostras, independentemente do tamanho de p, desde que o numerador seja igual para todos os componentes p, característica que agiliza o processo computacional em relação a outros algoritmos de aproximação estocástica que exige medidas da função de perda em cada iteração de número p.

A economia de medidas da função de perda por iteração revela o potencial da técnica de PS para potencializar o processo de convergência do total de número de estimação para convergência de θ^*, principalmente quando p for grande. Este potencial é tangível somente se o número de iterações requerido para a convergência para θ^* não incrementar de forma a cancelar a economia das medidas pela aproximação do gradiente em cada iteração. Embora o resultado da convergência, com os requisitos computacionais reduzidos por cada iteração, seja uma característica importante dessa técnica, o resultado teórico mais importante e que mais justifica o uso da técnica da PS são as conclusões quanto à eficiência assintótica que segue do resultado normalmente assintótico.

3.2.1 Condição de Convergência

Para haver convergência nas iterações ($\hat{\theta}_k \to \theta^*$) é necessário impor condições nas seqüências dos ganhos a_k e c_k, na distribuição de Δ_k, e na relação estatística de Δ_k para com as medidas de $y(\cdot)$. Essas condições garantem a convergência de $\hat{\theta}_k$ para o ponto mínimo θ^*.

As condições dadas por Spall são demonstradas pela Equação (3.5), ou seja:

$$k^{\beta/2}(\hat{\theta}_k - \theta^*) \xrightarrow{dist.} N(\mu, \Sigma), \text{ quando } k \to \infty \tag{3.5}$$

onde $\xrightarrow{dist.}$ denota convergência na distribuição, $\beta > 0$, μ e Σ são vetores de média e matriz de covariância. Aqui μ depende do Hessiano e da terceira derivação de $L(\theta)$ em relação a θ^* e Σ depende da matriz Hessiana a θ^*.

A distribuição assintótica que resulta em (3.5) permite que se determine o ganho ótimo da taxa de decaimento que fornece o valor máximo de $\beta/2$. Este valor máximo é $\beta/2 = 1/3$. Portanto, a taxa de possibilidade mais rápida em que o erro $\hat{\theta}_k - \theta^*$ vai para zero é $k^{-1/3}$.

A aplicação do resultado normalmente assintótico da expressão (3.5) foi usada para estabelecer a eficiência da técnica de PS. Esta eficiência, portanto, depende da forma de $L(\theta)$, dos valores para a_k e c_k, da distribuição de Δ_{ki} e da medida do termo de ruído $\{\varepsilon_k^{\pm}\}$. As Equações (3.6) e (3.7) demonstram como se pode obter a_k e c_k, sendo a, c, A, α e γ constantes positivas.

$$a_k = \frac{a}{(A + k + 1)^{\alpha}} \tag{3.6}$$

$$c_k = \frac{c}{(k + 1)^{\gamma}} \tag{3.7}$$

Em relação aos métodos determinísticos padrão, a otimização estocástica abre um vasto campo para solucionar problemas cuja solução não seja elementar, em especial os

problemas de natureza combinatorial. Algoritmos empregando a técnica PS permitem o tratamento efetivo de problemas das mais diversas áreas, dentre elas o treinamento de redes neurais.

As principais características que PS apresentam são as seguintes:

i) Relativa facilidade de implementação e baixa necessidade de maiores detalhes sobre o gradiente da função de perda;

ii) Robustez na leitura da função de perda com ruído;

iii) Evidência empírica na habilidade para encontrar o mínimo local (global) mesmo havendo múltiplos locais, tanto localmente quanto globalmente.

3.3 Proposição de Rede Recorrente com Perturbação Simultânea

Redes recorrentes possuem larga aplicação para manipular informações de sistemas dinâmicos, bem como armazenamento de padrões, tendo como destaque a RNH que é uma arquitetura particular de redes recorrentes.

Tais redes recorrentes apresentam dificuldades em sua utilização devido a uma falta de teoria genérica visando um treinamento efetivo de seus pesos. Diversos autores aplicaram os conceitos de PS como regra para treinamento dos pesos sinápticos devido à sua simplicidade. PS consegue estimar o gradiente de uma função usando somente duas medidas da sua própria função. A facilidade dessa técnica aplicada no treinamento de RNR está no fato de somente o valor do erro final ser requerido para estimar o gradiente da função do erro com respeito aos pesos, sendo que essa simplicidade coloca tal técnica como uma das mais promissoras no treinamento de RNR.

Na maioria das aplicações usando RNR geralmente é possível conhecer a saída ideal por meio do conjunto de amostras conhecidas. Usando essa informação é então possível

avaliar o desempenho da rede como uma realimentação para realizar a otimização do ajuste dos pesos sinápticos da rede. Com um estado estável conhecido, é possível portanto obter a quantidade a ser ajustada de todos os pesos da rede sem as dificuldades conhecidas que envolvem o treinamento de uma RNR.

MAEDA e WAKAMURA (2005) descrevem uma regra para treinamento de RNR com a teoria de PS conforme (3.8) e (3.9).

$$\Delta w_t^i = \frac{J(w_t + c_k s_t) - J(w_t)}{c_k s_t^i} \tag{3.8}$$

$$w_{t+1}^i = \begin{cases} w_{\max}, & if\ (w_t^i - a_k \Delta w_t^i) > w_{\max} \\ -w_{\max}, & if\ (w_t^i - a_k \Delta w_t^i,) < -w_{\max} \\ w_t^i - a_k \Delta w_t^i, & demais \end{cases} \tag{3.9}$$

onde w_t e w_t^i denota o vetor de pesos da rede e seu *i*-ésimo elemento na *t*-ésima iteração, respectivamente, a_k é uma constante positiva e c_k é a magnitude da perturbação. O termo Δw^i representa o *i*-ésimo elemento do vetor modificado e w_{max} é o valor máximo dos pesos. Os parâmetros s_t e s_t^i denotam o vetor de sinal e seu *i*-ésimo elemento que podem assumir o valor 1 ou -1, respectivamente. O sinal de s_t^i é determinado randomicamente. E mais, o sinal de s_t^i é independente do sinal do *j*-ésimo elemento s_t^j do vetor sinal, conforme (3.10):

$$E(s_t^i) = 0,\ E(s_{t_1}^i s_{t_2}^j) = \delta_{ij}\delta_{t_1 t_2} \tag{3.10}$$

onde *E* denota a expectativa, δ é o delta de Kronecker, *J(w)* denota um erro ou uma função de avaliação, por exemplo, definido pelas saídas do neurônio em um estado estável e um padrão para ser inserido. Na Seção 3.3.1 será explicado com mais detalhes como funciona a função *J(w)*.

A utilidade de w_{max} estabelece limites para evitar disparidades dos valores absolutos entre os pesos sinápticos. Na maioria das vezes ele não é necessário, mas devido à natureza errática dos valores dos pesos ele pode ser muito prático.

Quando (3.8) é expandido em relação à w_t, obtém-se w_{s1} conforme (3.11).

$$\Delta w_t^i = s_t^i s_t^T \frac{\partial J(w_t)}{\partial w} + \frac{c s_t^i}{2} s_t^T \frac{\partial^2 J(w_{s_1})}{\partial w^2} s_t \tag{3.11}$$

A partir de (3.11) uma expectativa é esperada. Por meio da expressão (3.10) é possível extrair o vetor de sinal s_t, onde resulta (3.12).

$$E(\Delta w_t^i) = \frac{\partial J(w_t)}{\partial w_t^i} \tag{3.12}$$

ou seja, Δw_t^i se aproxima de $\frac{\partial J(w_t)}{\partial w_t^i}$. Desde que a expressão (3.8) seja um valor estimado do coeficiente de primeira derivação da função de erro (função de perda), a regra de aprendizado é um tipo de método de gradiente estocástico.

Em sua forma básica, a técnica de PS pode ser usada com diferentes distribuições. Conforme SPALL (1998) descreveu, a mais apropriada é a distribuição de Bernoulli, pois simplifica e reduz a complexidade da implementação.

3.3.1 Roteiro para Treinamento de RNR

A maior vantagem em relação à aplicação de PS no treinamento de RNR apresentada por essa regra de aprendizagem é o fato de requerer somente dois valores da função de erro. Na elaboração de uma estratégia de treinamento de uma RNR, o primeiro passo consiste em executar a mesma com os pesos sinápticos iniciados, juntamente com o estado inicial da rede, até um estado estável ser atingido. A seguir, calcula-se o valor do erro que pode ser definido, por exemplo, como (3.13):

$$J(w) = \sum_i (o^i - d^i)^2 \tag{3.13}$$

onde o^i e d^i denota um estado estável da RNR e sua correspondente saída ideal, respectivamente. Este erro é a diferença entre o padrão final ideal e o presente padrão final da rede. Em alguns casos, as mudanças no estado da rede podem influenciar no desempenho da mesma para otimização; neste caso, pode-se considerar a expressão (3.14):

$$J(w) = \sum_t \sum_i (o_t^i - d_t^i)^2 \qquad (3.14)$$

onde o_t^i e d_t^i denota o estado do neurônio e o correspondente estado desejado na t-ésima iteração, respectivamente. Se o valor do erro for pequeno, significa que a trajetória da rede está próxima de uma trajetória ideal.

Após determinado os erros nas saídas da RNR, adiciona-se perturbação a todos os pesos da rede simultaneamente. A perturbação deve ter magnitude igual a $+c_k$ ou $-c_k$, determinados randomicamente. Após a perturbação simultânea nos pesos, a rede é então executada novamente e o erro é calculado conforme (3.13) ou (3.14), dependendo do contexto, obtendo-se assim $J(w+c_k.s)$.

Todo o processo é repetido até a resposta desejada ser atingida. Vale ressaltar que somente 2 valores de erros da RNR são usados para atualizar todos os pesos na rede; portanto, o procedimento é muito simples e fácil de ser implementado.

3.4 Mapeamento do Problema do Caixeiro Viajante na Rede Recorrente com Perturbação Simultânea

Conforme visto anteriormente, em todas as aplicações que se tem PS como regra de treinamento de RNR, envolve-se obrigatoriamente a necessidade de haver um conjunto de amostras de respostas conhecidas, caso este que não é possível no PCV, pois devido à sua natureza combinatorial de ordem exponencial, dependendo da quantidade de nós envolvendo a rede, torna a procura por um mínimo local (global) inviável computacionalmente. Por isso, foram necessárias algumas adaptações na estratégia de treinamento da RNR adotada aqui. Segundo a função objetivo (2.1) que representa o custo total a ser minimizado, implica-se em dizer que ela deve ter o menor valor possível, por isso, segundo a expressão (3.13), a saída ideal d^i passa a valer sempre zero.

Em relação à estrutura de realimentação da RNR, adotou-se a topologia empregada na RRW, sem levar em consideração a regra de treinamento juntamente com as restrições para determinação de um caminho viável. Os pesos foram inicializados com valores aleatórios entre -1 e 1. A RRW é executada inicialmente com os valores de entrada iguais aos componentes da matriz de custo, com valores normalizados entre 0 e 1.

Após a convergência da rede, aplica-se a função de ativação sigmóide e determina-se o maior valor da primeira coluna da matriz de custo, sendo que essa coluna se refere ao nó número 1 de partida. O maior valor de cada coluna implica no próximo nó. Ao mesmo tempo, todos os valores pertencentes à linha e à coluna do maior valor são zerados antes de analisar a próxima coluna da matriz de custo.

O processo de encontrar o maior valor da coluna e zerar toda a linha e coluna pertencente a esse valor é repetido até a última coluna da matriz de custo, de modo que só restarão os valores diferentes de zero, pertinentes ao itinerário do Caixeiro Viajante. A distância é então calculada como uma das duas medidas necessárias para se executar o ajuste dos pesos sinápticos.

Em seguida, são adicionadas perturbações aos pesos sinápticos, conforme expressão (3.15) e a RRW é executada novamente seguindo os mesmos passos, conforme descrito anteriormente, até o cálculo da nova distância encontrada com as perturbações adicionadas, que é a segunda medida necessária para o ajuste definitivo dos pesos sinápticos, $J(w_t + c_k s_t)$, ou seja:

$$w_p = w + c_k \cdot s_t^i \qquad (3.15)$$

onde:

w_p é o novo peso com perturbação;

w é o peso original;

c_k é a magnitude da perturbação em função da iteração número *k*;

s_t^i é determinado randomicamente com valores iguais a 1 ou -1.

De posse das duas medidas das distâncias percorridas, as expressões (3.8) e (3.9) são então aplicadas, realizando-se assim um novo ajuste nos pesos sinápticos e uma nova rodada de iterações com esses novos pesos começa novamente. O fim do laço de execução de iterações e ajustes é pré-determinado assumindo-se inicialmente um valor máximo para *k*.

3.4.1 Exemplo para Determinação do Caminho Após Convergência da RRW

Após a primeira rodada de iteração de a RRW convergir a um estado estável e a função de ativação ser aplicada a cada uma das *n* x *n* saídas, assumindo para esse exemplo um valor de *n* igual a seis cidades, foi obtido o seguinte resultado apresentado pela Figura 3.2.

$$\begin{pmatrix} 0{,}6855 & 0{,}2649 & 0{,}5497 & 0{,}1975 & 0{,}8756 & 0{,}5421 \\ 0{,}6345 & 0{,}3473 & 0{,}4044 & 0{,}4003 & 0{,}6695 & 0{,}8406 \\ 0{,}0875 & 0{,}1824 & 0{,}4522 & 0{,}6470 & 0{,}4955 & 0{,}3347 \\ 0{,}7228* & 0{,}4425 & 0{,}2478 & 0{,}3737 & 0{,}8643 & 0{,}7474 \\ 0{,}6949 & 0{,}6942 & 0{,}6437 & 0{,}8352 & 0{,}3255 & 0{,}7827 \\ 0{,}2260 & 0{,}6915 & 0{,}6909 & 0{,}4439 & 0{,}4855 & 0{,}2063 \end{pmatrix}$$

Figura 3.2 – Resultado inicial da RRW, após convergir, para o PCV com seis nós (n = 6).

Começando pela primeira coluna, que significa o primeiro nó de partida, selecionou-se o maior valor, que nesse caso é a quarta linha marcada com um asterisco, sendo que todos os demais integrantes pertencentes à linha e coluna desse resultado é zerado, ou seja, a primeira coluna e a quarta linha, conforme ilustrado na Figura 3.3. Esse resultado significa que a partir do primeiro nó o próximo será o nó número 4.

$$\begin{pmatrix} 0 & 0{,}2649 & 0{,}5497 & 0{,}1975 & 0{,}8756 & 0{,}5421 \\ 0 & 0{,}3473 & 0{,}4044 & 0{,}4003 & 0{,}6695 & 0{,}8406 \\ 0 & 0{,}1824 & 0{,}4522 & 0{,}6470 & 0{,}4955 & 0{,}3347 \\ 0{,}7228 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}6942 & 0{,}6437 & 0{,}8352 & 0{,}3255 & 0{,}7827 \\ 0 & 0{,}6915 & 0{,}6909 & 0{,}4439 & 0{,}4855 & 0{,}2063 \end{pmatrix}$$

Figura 3.3 – Determinação do próximo nó, nesse caso o nó quatro, através do maior valor resultante na primeira coluna com todos os demais integrantes da linha e coluna zerados.

Em seguida, a próxima coluna, subseqüente à primeira, é escolhida para se determinar o maior valor, e novamente todos os demais resultantes pertencentes à coluna e linha do maior valor encontrado são zerados; aqui resultou da coluna 2 e da linha 5, conforme ilustrado na Figura 3.4. Nesse exemplo o resultado indica que partindo do nó 2 deve-se ir para o nó 5.

$$\begin{pmatrix} 0 & 0 & 0{,}5497 & 0{,}1975 & 0{,}8756 & 0{,}5421 \\ 0 & 0 & 0{,}4044 & 0{,}4003 & 0{,}6695 & 0{,}8406 \\ 0 & 0 & 0{,}4522 & 0{,}6470 & 0{,}4955 & 0{,}3347 \\ 0{,}7228 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}6942 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0{,}6909 & 0{,}4439 & 0{,}4855 & 0{,}2063 \end{pmatrix}$$

Figura 3.4 – Determinação da seqüência de nós no segundo passo da sistemática de seleção de nós.

O processo é repetido para todas as colunas seqüencialmente até a derradeira, conforme seqüência de figuras exposto na Figura 3.5, de modo que ao final da procura os valores diferentes de zero que restarem em cada coluna indicarão o caminho a ser adotado. A Figura 3.5(a) resulta o terceiro passo do itinerário cujo trecho de percurso indicado é do nó 3 para o 6. A Figura 3.5(b) assinala do nó 4 para o nó 3 e finalmente na Figura 3.5(c) tem-se condições de se conhecer os dois últimos trechos do percurso, ou seja do nó 5 para o nó 1, indicado na quinta coluna da matriz de saída da rede, e finalmente, na sexta coluna, tem-se indicado saindo do nó 6 para o 2 fechando o percurso por todos os nós sem repeti-los. Ainda na Figura 3.5(c) os resultados vencedores da sistemática para determinação do caminho a ser adotado foram marcados com um círculo.

$$\begin{pmatrix} 0 & 0 & 0 & 0{,}1975 & 0{,}8756 & 0{,}5421 \\ 0 & 0 & 0 & 0{,}4003 & 0{,}6695 & 0{,}8406 \\ 0 & 0 & 0 & 0{,}6470 & 0{,}4955 & 0{,}3347 \\ 0{,}7228 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}6942 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0{,}6909 & 0 & 0 & 0 \end{pmatrix}$$

(a)

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0{,}8756 & 0{,}5421 \\ 0 & 0 & 0 & 0 & 0{,}6695 & 0{,}8406 \\ 0 & 0 & 0 & 0{,}6470 & 0 & 0 \\ 0{,}7228 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}6942 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0{,}6909 & 0 & 0 & 0 \end{pmatrix}$$

(b)

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0{,}8756 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0{,}8406 \\ 0 & 0 & 0 & 0{,}6470 & 0 & 0 \\ 0{,}7228 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}6942 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0{,}6909 & 0 & 0 & 0 \end{pmatrix}$$

(c)

Figura 3.5 – Seqüência do processo de procura do caminho até a última coluna, onde o caminho a ser adotado será os valores diferentes de zero. (a) Indica do nó 3 para o nó 6; (b) Resulta do nó 4 para o nó 3; (c) Resulta os últimos dois percursos, do nó 5 para o nó 1 e do nó 6 para o nó 2, os valores circulados indica o percurso completo 1 – 4 – 3 – 6 – 2 – 5 – 1.

A distância é então calculada, como primeira medida, para ser usada posteriormente na determinação dos pesos w_{k+1}, logo após a próxima iteração com a perturbação adicionada aos pesos. Com a adição de perturbações aos pesos, a RRW é executada novamente até convergir a um estado estável, seguindo a mesma sistemática de seleção do percurso conforme descrito e ilustrado anteriormente nas figuras. A distância do novo caminho resultante dessa procura é então calculada como a segunda medida para ser usada nas expressões (3.8) e (3.9). Em seguida, todo o processo é repetido novamente, com os novos pesos ajustados, até o número de rodada k ser incrementado para o valor máximo pré-determinado.

CAPÍTULO 4

Resultados de Aplicação da Metodologia Proposta

4.1 Introdução

Preliminarmente foi executada uma série de treinamentos adotando uma das instâncias disponíveis na biblioteca TSPLIB, conhecida por ulysses22; com número de nós igual a 22, posições fixas e menor caminho conhecido. Conforme ilustrado na Figura 4.1, a menor distância é igual a 7.013. A partir dessa referência foi realizado um estudo sistemático a fim de estabelecer as relações entre os três coeficientes *a*, *c* e *A* da PS, assumindo um número de iterações de no mínimo 45.000, o qual é suficiente para a rede encontrar um mínimo local. Foi verificada também a possibilidade de se sair de um mínimo local para outro até uma eventual solução ótima global. Embora Spall afirme várias vezes em seus artigos que a técnica de PS consegue sair de um mínimo local para o global, isso não foi constatado nos experimentos realizados aqui.

Latitude	Longitude
38,24	20,42
39,57	26,15
40,56	25,32
36,26	23,12
33,48	10,54
37,56	12,19
38,42	13,11
37,52	20,44
41,23	9,10
41,17	13,05
36,08	-5,21
38,47	15,13
38,15	15,35
37,51	15,17
35,49	14,32
39,36	19,56
38,09	24,36
36,09	23,00
40,44	13,57
40,33	14,15
40,37	14,23
37,57	22,56

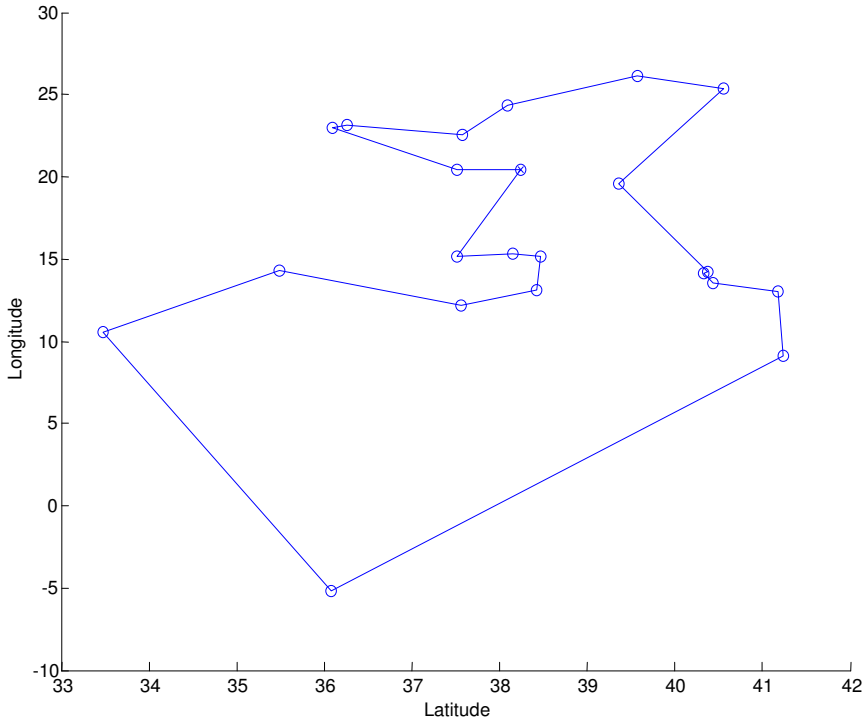


Figura 4.1 – Posição dos 22 nós e melhor caminho para ulysses22 da biblioteca TSPLIB.

Após uma série de ensaios rápidos a fim de estabelecer uma faixa de valores factíveis para os coeficientes *a*, *c* e *A* fazerem a RRW convergir para um mínimo local, foram encontrados respectivamente 0,09; 0,5 e 0,1 * max(*k*), onde *k* é o número da iteração de uma rodada de treinamento qualquer. Num primeiro momento, dos três valores anteriores, dois foram mantidos fixos e *A* foi variando de 0,01 até 1,00 com incremento de 0,01 entre cada rodada de iterações. O melhor resultado encontrado para *A* foi igual a 0,13, com menor distância percorrida igual a 15.536, conforme ilustrado na Figura 4.2(a). A Figura 4.2(b) ilustra o gráfico das iterações para *A* valendo 0,13 e suas respectivas distâncias encontradas por iteração onde é possível observar a convergência para um mínimo local. Na Figura 4.2(c) têm-se os menores caminhos para cada rodada de iteração para a faixa de valores incrementados de *A*.

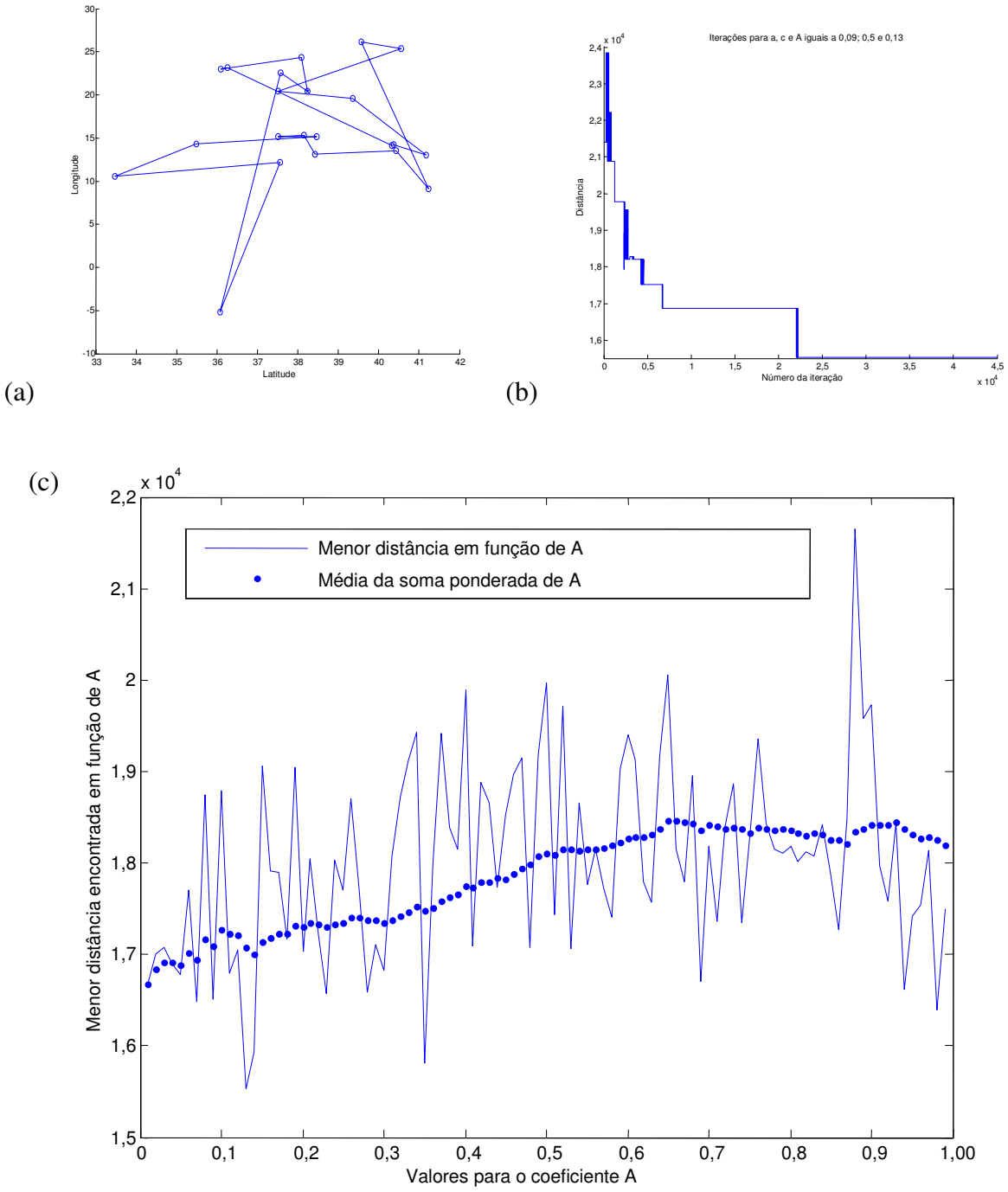
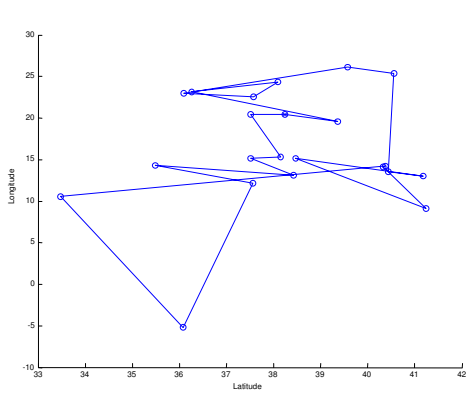


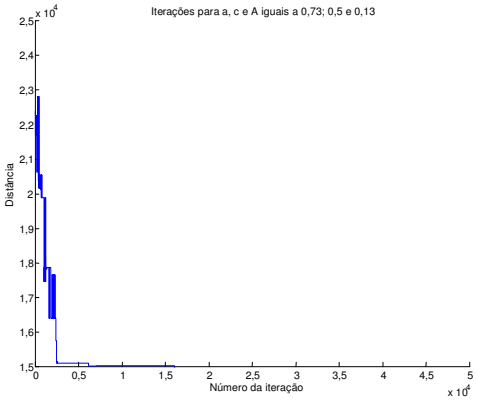
Figura 4.2 – Resultados para *a*, *c* e *A* iguais a respectivamente 0,09; 0,5 e 0,13. (a) Resultado do menor percurso encontrado. (b) Gráfico de convergência para o mínimo local em função das iterações. (c) Menor caminho encontrado nas rodadas de iterações em função dos incrementos de *A* (traço contínuo) e média ponderada para os 35 últimos valores do eixo vertical (pontilhado).

Em seguida, com o novo valor de *A* fixada em 0,13, o coeficiente *a* foi variado de 0,01 a 1,00 com incremento entre cada rodada de iterações de 0,01. O melhor resultado encontrado para *a* foi igual a 0,73, com menor distância percorrida igual a 14.649, conforme ilustrado na

Figura 4.3(a). A Figura 4.3(b) ilustra o gráfico das iterações e suas respectivas distâncias encontradas por iteração para *a* valendo 0,73; onde é possível observar a convergência para um mínimo local. Na Figura 4.3(c) têm-se os menores caminhos para cada rodada de iteração para a faixa de valores incrementados de *a*.



(a)



(b)

(c)

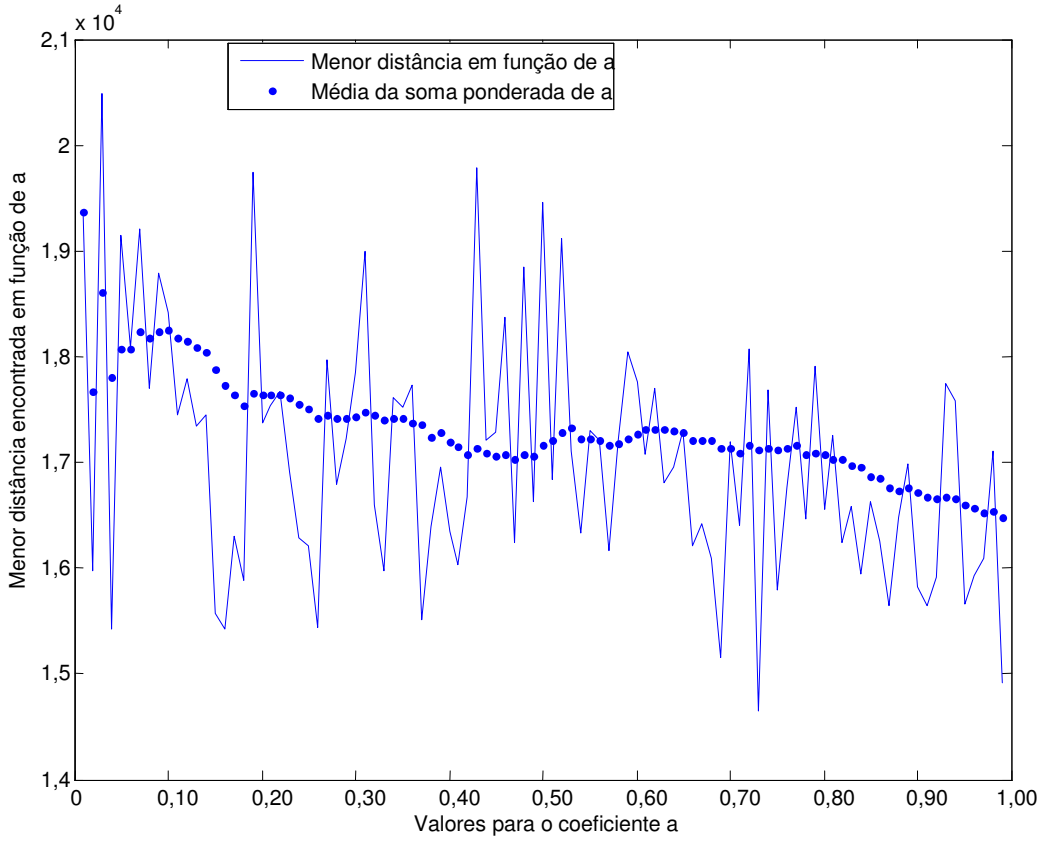
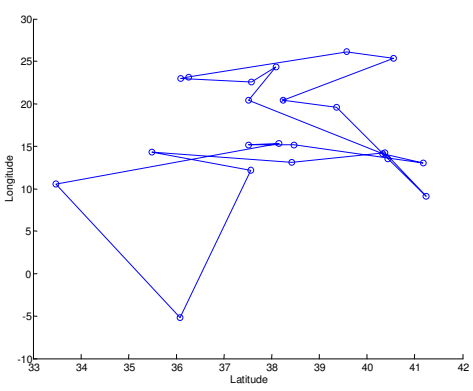


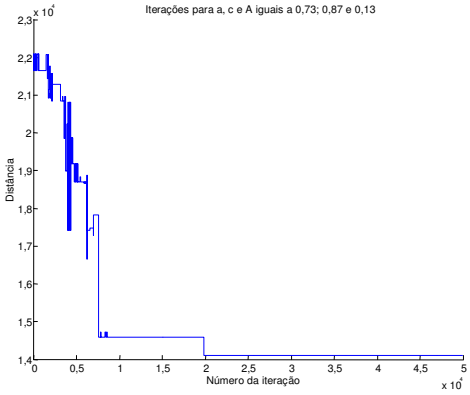
Figura 4.3 – Resultados para *a*, *c* e *A* iguais a respectivamente 0,73; 0,5 e 0,13. (a) Resultado do menor percurso encontrado. (b) Gráfico de convergência para o mínimo local em função das iterações. (c) Menor caminho encontrado nas rodadas de iterações em função dos incrementos de *a* (traço contínuo) e média ponderada para os 35 últimos valores do eixo vertical (pontilhado).

Finalmente, foram fixados os valores para *A* e *a* respectivamente em 0,13 e 0,73, que vêm a serem os melhores valores obtidos nas duas rodadas de iterações anteriores. Para o coeficiente *c*, inicialmente, foi variado de 0,01 até 1,00 com o mesmo valor de incremento das

duas vezes anteriores, ou seja, de 0,01. Mas, devido a evidências no gráfico sobre os melhores resultados em função de cada rodada de iterações, decidiu-se então estender os incrementos de *c* para até 2,19. Foi encontrado o menor caminho obtido até agora, para *c* valendo 0,87, de 14.111, ilustrado na Figura 4.4(a). O gráfico relativo à convergência para um mínimo local em função das iterações pode ser visto na Figura 4.4(b) e na Figura 4.4(c) encontra-se os menores caminhos para cada rodada de iteração dos incrementos de *c*.



(a)



(b)

(c)

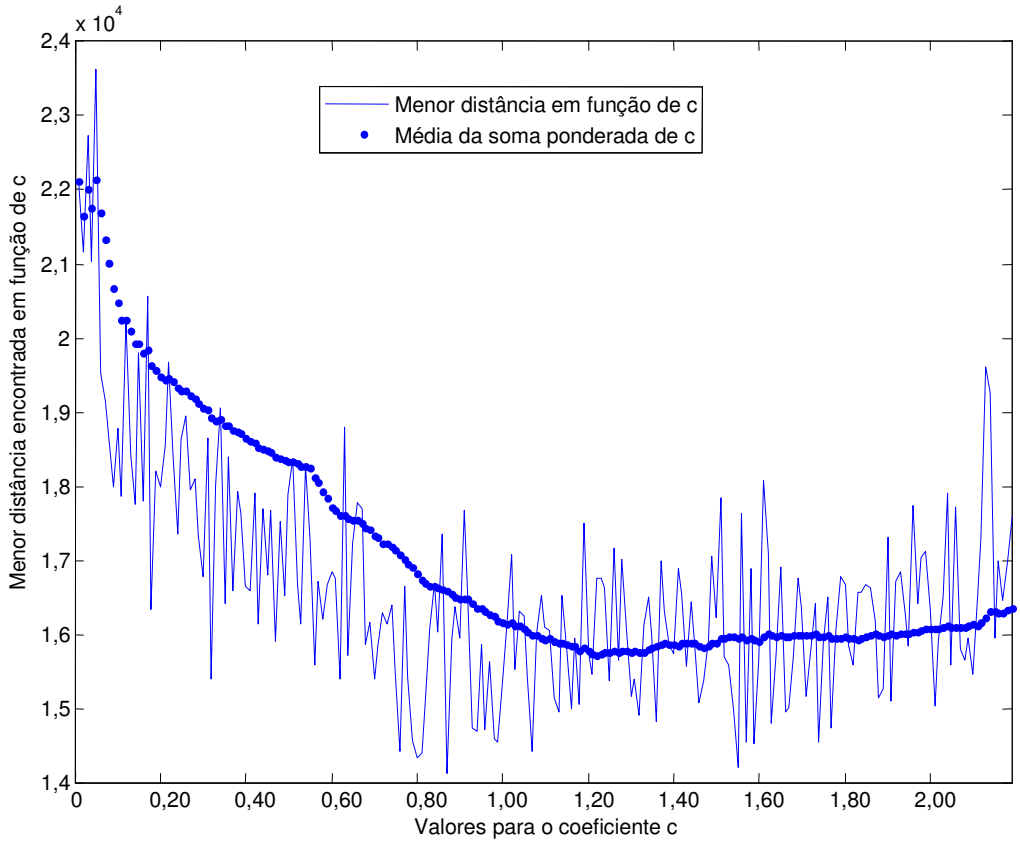


Figura 4.4 – Resultados para *a*, *c* e *A* iguais a respectivamente 0,73; 0,87 e 0,13. (a) Resultado do menor percurso encontrado. (b) Gráfico de convergência para o mínimo local em função das iterações. (c) Menor caminho encontrado nas rodadas de iterações em função dos incrementos de *c* (traço contínuo) e média ponderada para os 35 últimos valores do eixo vertical (pontilhado).

Dessa forma, com os valores dos coeficientes *A*, *a* e *c* ajustados para respectivamente 0,13; 0,73 e 0,87 foi possível obter resultados para outras bibliotecas com número de nós próximos a 22. Para valores mais elevados, tornam-se necessários novos ajustes.

Nas seções seguintes serão apresentados alguns resultados dos trabalhos apresentados e comentados no Capítulo 2, Seção 2.4, para serem comparados com os resultados obtidos com a técnica de treinamento da RRW com PS.

4.2 Caso Experimental I

No trabalho de Wang, Tang e Cao (2002) foram usados apenas duas bibliotecas do TSPLIB para demonstração da técnica proposta, ulysses22 e eil51, cujo número de cidades para roteamento são respectivamente iguais a 22 e 51. Nos demais experimentos, eles geraram um conjunto de 100 instâncias, cada um com 10 cidades em posições aleatórias. Eles afirmam que, em todos os exemplos, obtiveram 100% de soluções ótimas, inclusive para ulysses22 e eil51, que possuem menor distância igual a 7.013 e 426, respectivamente. Na Tabela 4.1 são exibidos os resultados obtidos em ulysses22 para diferentes valores de *A*, *a* e *c*. Os resultados foram obtidos rodando 10 vezes cada combinação de coeficientes com 50.000 iterações. Na Figura 4.5 é ilustrado as respectivas médias da Tabela 4.1 com o desvio padrão delimitando seus extremos verticalmente.

Tabela 4.1 – Resultados para diferentes coeficientes rodando ulysses22 com 22 cidades.

	***A* = 0,13** ***a* = 0,73** ***c* = 1,56**	***A* = 0,13** ***a* = 0,73** ***c* = 0,87**	***A* = 0,001** ***a* = 0,1** ***c* = 0,50**	***A* = 0,005** ***a* = 0,09** ***c* = 0,50**	***A* = 0,10** ***a* = 0,09** ***c* = 0,50**	***A* = 0,012** ***a* = 0,09** ***c* = 0,50**	***A* = 0,5** ***a* = 0,0009** ***c* = 0,20**
Mínimo	14.844	15.087	15625	16.334	15.922	15.816	20.502
Média	15.617	16.284	17.033	17.247	17.607	17.897	21.216
Máximo	16.529	17.898	19.024	18.816	19.204	19.281	23.267
Desvio padrão	614,42	832,74	1.275,4	852,72	1.214	1.069,3	883,64

Na Tabela 4.2 são exibidos os resultados obtidos de eil51 para algumas combinações dos coeficientes *A*, *a* e *c*. Devido à característica da topologia da RRW, conforme o número de cidades *n* aumenta, o número de pesos sinápticos necessários para a rede aumenta na proporção $2.n^3-n^2$ e, conseqüentemente, o tempo para concluir as iterações. Por isso, os

valores exibidos são resultantes de uma rodada de 50.000 iterações cada linha. O tempo de processamento a partir de problemas com mais de 50 cidades passa a ser crítico.

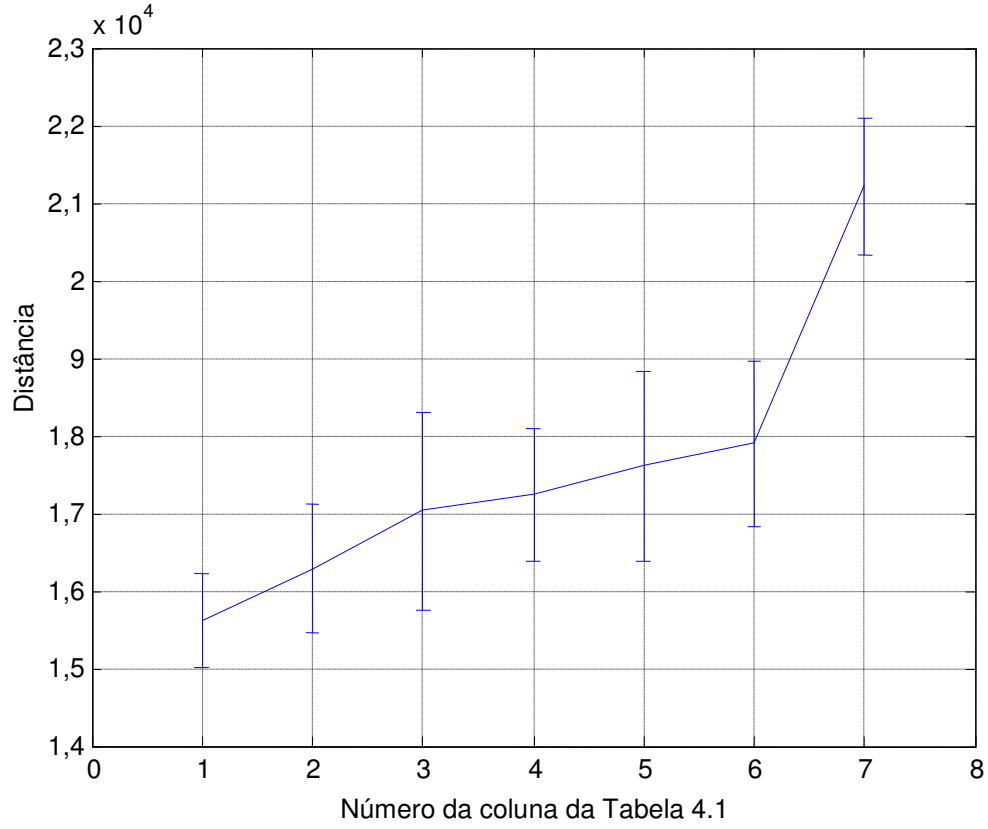


Figura 4.5 – Gráfico com as respectivas delimitações do desvio padrão

Tabela 4.2 – Resultados de eil51 a partir de 50.000 iterações.

	Coeficientes	**Menor caminho encontrado**
1	*A = 0,01; a = 0,10; c = 1,00*	876
2	*A = 0,01; a = 0,20; c = 1,00*	879
3	*A = 0,13; a = 0,73; c = 0,87*	920
4	*A = 0,01; a = 0,1667; c = 1,00*	941
5	*A = 0; a = 300; c = 100*	1065

Vale observar, através da Figura 4.6, o resultado da convergência em função das iterações referentes à linha 5 da Tabela 4.2. A função que *A* tem sobre a estabilidade das iterações, embora o menor caminho obtido não tenha ficado muito acima das demais linhas, a RRW não conseguiu convergir para um mínimo local como ocorre normalmente.

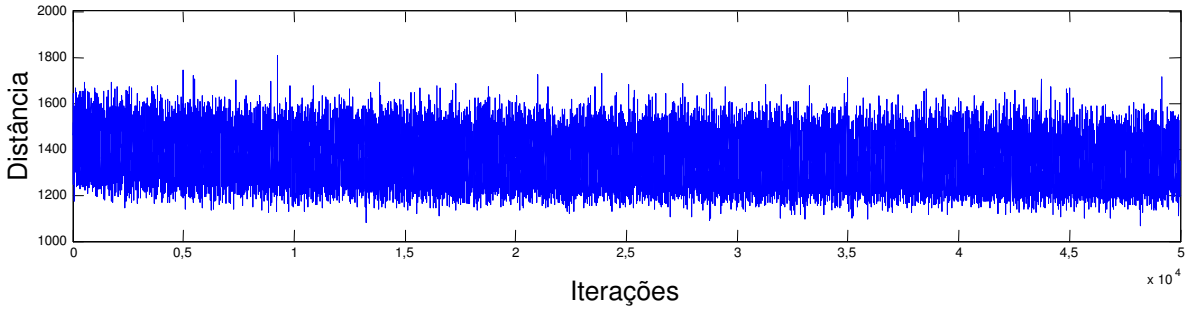


Figura 4.6 – Gráfico das iterações de busca de um mínimo local para *A* igual a zero de eil51.

4.3 Caso Experimental II

A topologia de RNR no qual esse trabalho se inspirou foi a adotada em Siqueira, Scheer e Steiner (2007). Eles usaram algumas das instâncias da biblioteca TSPLIB pertencentes aos grupos com menores números de cidade em relação aos disponíveis, sendo que isso se deve à demanda por pesos sinápticos à medida que o número de cidades cresce, já comentada na Seção 4.2. Os resultados expostos na Tabela 4.3, para efeito de comparação, estão representados em porcentagem do desvio em relação ao caminho ótimo, descrevendo-se o número de cidades de cada instância, o caminho ótimo, os resultados apresentados no artigo de Siqueira, Scheer e Steiner (2007), denominado RRW + VLT, e os resultados obtidos por essa dissertação indicados por RRW + PS, seguido pela coluna com os valores dos coeficientes usados na simulação.

Tabela 4.3 – Comparação dos resultados obtidos entre RRW + VLT e RRW + PS com os respectivos coeficientes para os menores caminhos obtidos.

TSPLIB	***n***	**Melhor Caminho**	**RRW + VLT**		**RRW + PS**	**Coeficientes**
			Max.	Mín.		
eil51	51	426	1,16%	1,16%	103,7%	*A = 0,01; a = 0,10; c = 1,00*
st70	70	675	4,04%	2,71%	203,7%	*A = 0,01; a = 0,1667; c = 1,00*
eil76	76	538	2,49%	1,03%	143,3%	*A = 0,01; a = 0,1667; c = 1,00*
rd100	100	7.910	7,17%	6,83%	243,6%	*A = 0,5; a = 0,1667; c = 1,00*
eil101	101	629	7,95%	3,02%	274,9%	*A = 0,01; a = 0,1667; c = 1,00*
bier127	127	118.282	5,08%	4,22%	201,6%	*A = 0,02; a = 0,1667; c = 1,00*
rat195	195	2.323	8,82%	5,55%	303,9%	*A = 0,02; a = 0,1667; c = 1,00*
kroa200	200	29.368	12,25%	8,95%	719,8%	*A = 0,5; a = 0,10; c = 1,00*

Os coeficientes ajustados para ulysses22, descrito no começo desse capítulo, já não serviram para as instâncias da Tabela 4.3, pois apresentaram resultados muito deficientes, com exceção de eil51, que mesmo assim não foi o melhor resultado. Serão apresentados apenas os melhores resultados das poucas combinações de coeficientes executados devido ao elevado tempo de processamento.

Na Figura 4.7 ilustra o gráfico de convergência para um mínimo local em função das iterações obtidas de eil51.

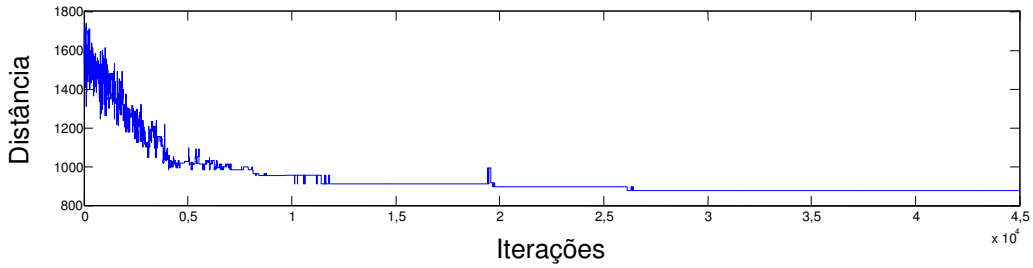


Figura 4.7 – Convergência para um mínimo local em função das iterações para eil51.

Com mais de 70 cidades, além de serem necessárias mais iterações para um mínimo local ser estabilizado, os coeficientes começam a serem inadequados, conforme indica a Figura 4.8 referente a st70.

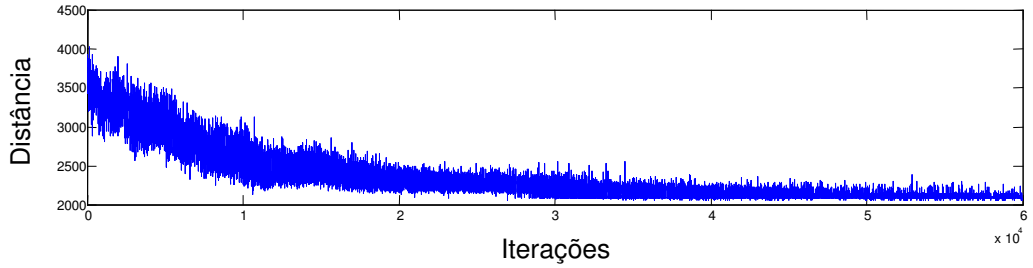


Figura 4.8 – Tendência de convergência para um mínimo local em função das iterações para st70.

Em relação ao número de cidades, para os mesmos valores dos coeficientes de st70, o resultado ilustrado na Figura 4.9 demonstra ter aparência e desempenho semelhante para eil76.

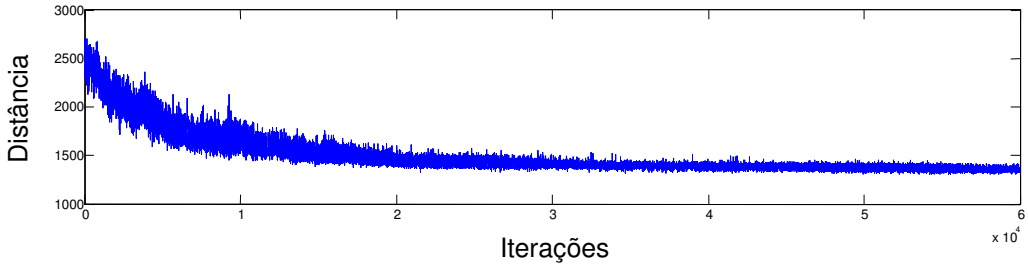


Figura 4.9 – Tendência de convergência para um mínimo local em função das iterações para eil76.

Para rd100 foi necessário realizar um ajuste em *A*. Foi possível com isso fazer a rede estabilizar em um mínimo local sem se alterar os demais parâmetros, mesmo assim o desempenho ficou bem aquém do que o de costume, como pode ser visto na Figura 4.10.

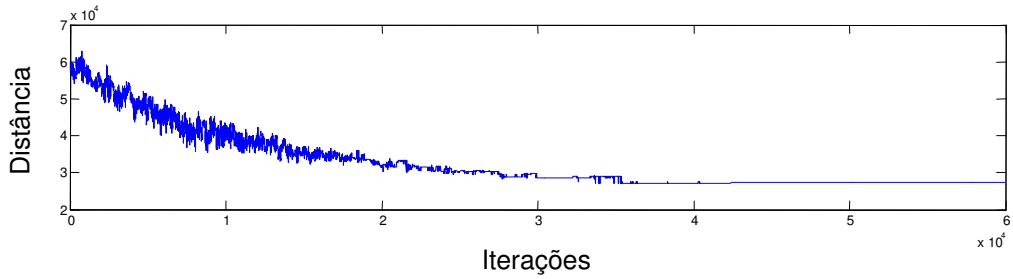


Figura 4.10 – Convergência para um mínimo local em função das iterações para rd100.

Tentou-se manter *A* igual a 0,01 em eil101, ao contrário de rd100, e como pode ser visto na Figura 4.11 a convergência foi lenta, quase imperceptível.

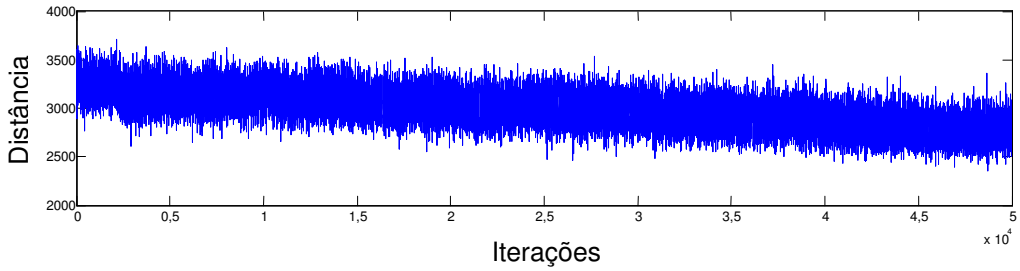


Figura 4.11 – Tendência de convergência para um mínimo local em função das iterações para eil101.

Em bier127 foi possível recuperar o desempenho obtido em st70 e eil76, mesmo ampliando o número de iterações para 90.000, com um pequeno ajuste de *A*, conforme ilustra Figura 4.12.

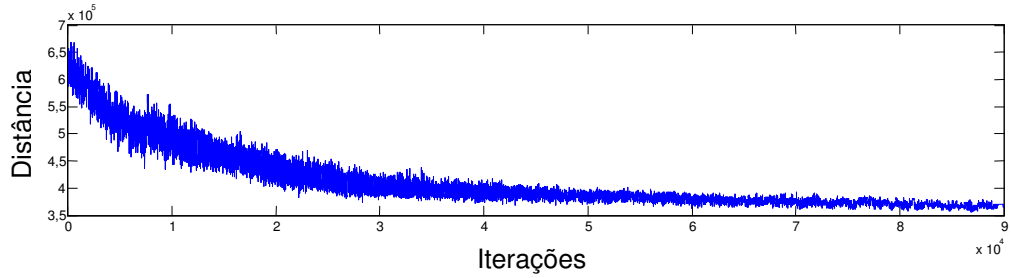


Figura 4.12 – Tendência de convergência para um mínimo local em função das iterações para bier127.

Para rat195 a convergência para um local mínimo foi mantida, mesmo que momentaneamente se variando nas primeiras 1.500 iterações, e depois se estabilizando, para os mesmos valores dos coeficientes de bier127, conforme ilustra Figura 4.13.

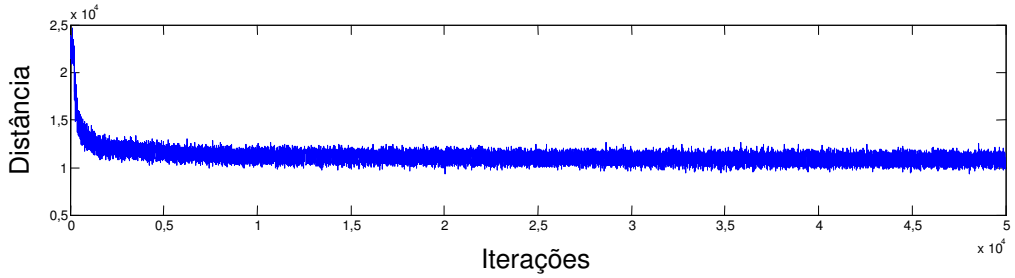


Figura 4.13 – Tendência de convergência para um mínimo local em função das iterações para rat195.

Finalmente, na Figura 4.14, tem-se o mais demorado de todas as simulações executadas aqui, mas ainda foi possível fazer com que a rede demonstre tendência de convergência para kroa200. É possível constatar que à medida que o número de cidades aumenta o número de iterações necessárias para se encontrar um mínimo local também se eleva. Infelizmente, não foi possível determinar o quanto vem a ser esse aumento.

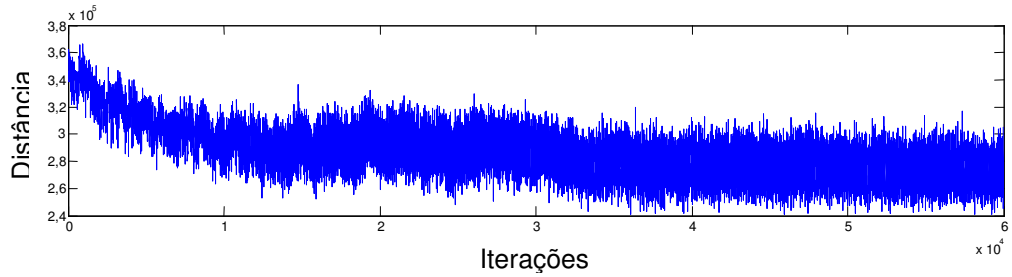


Figura 4.14 – Tendência de convergência para um mínimo local em função das iterações para kroa200.

4.4 Caso Experimental III

Apesar do trabalho de Talaván e Yáñez (2006) ter sido sobre o problema quadrático da mochila, foi publicado a aplicação de sua técnica sobre algumas instâncias da biblioteca TSPLIB, onde pode ser visto na Tabela 4.4 indicado na coluna RHC. Descreve-se nesta tabela a média entre os valores apresentados no artigo e o número de execuções por instância para formação do resultado. Juntamente são expostos os resultados obtidos por esta dissertação nomeado por RRW + PS, o número de execuções, seguido pelos valores de A, a e c.

Os resultados são expressos segundo a expressão (4.1).

$$\varsigma = \frac{y}{C} \tag{4.1}$$

onde y é a distância do percurso encontrado e C é o melhor caminho da instância.

Tabela 4.4 - Comparação dos resultados obtidos entre RHC e RRW + PS com os respectivos coeficientes para os menores caminhos obtidos.

TSPLIB	***n***	**Melhor Caminho**	**Número de execuções**	**ς (RHC)**	**Número de execuções**	**ς (RRW + PS)**	**Coeficientes**
gr24	24	1.272	100	1,37	10	1,65	*A = 0,13; a = 0,73; c = 0,87*
fri26	26	937	100	1,48	10	1,79	*A = 0,13; a = 0,73; c = 0,87*
bayg29	29	1.610	100	1,48	10	1,82	*A = 0,13; a = 0,73; c = 0,87*
Bays29	29	2.020	100	1,44	10	1,66	*A = 0,13; a = 0,73; c = 0,87*
eil51	51	426	100	1,51	1	2,05	*A = 0,01; a = 0,10; c = 1,00*
eil76	76	538	50	1,61	1	2,43	*A = 0,01; a = 0,1667; c = 1,00*
rd100	100	7.910	50	1,98	1	3,43	*A = 0,5; a = 0,1667; c = 1,00*
eil101	101	629	50	1,81	1	3,75	*A = 0,01; a = 0,1667; c = 1,00*

Os valores ajustados para A, a e c, descrito na Seção 4.1, funcionaram bem para instâncias de até 29 cidades, conforme pode ser observado na Figura 4.15(a)-(d), onde a convergência se estabiliza em um mínimo local após certo número de iterações. Os gráficos de convergência das quatro últimas instâncias da Tabela 4.4 foram descritos na Seção 4.3.

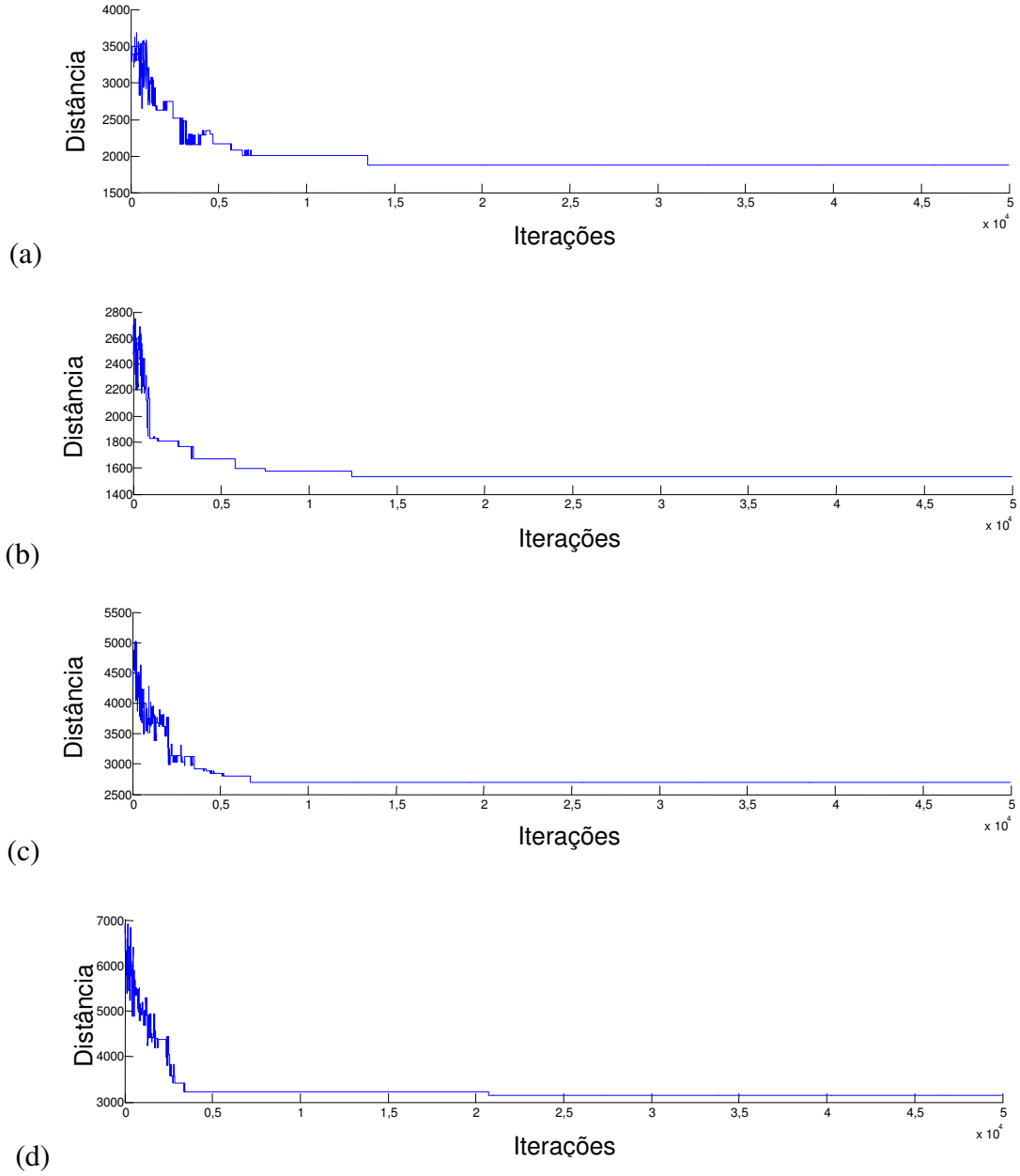


Figura 4.15 – Convergência para um mínimo local em função das iterações para o melhor resultado entre as 10 execuções de cada instância. (a) Convergência para gr24. (b) Convergência para fri26. (c) Convergência para bayg29. (d) Convergência para bays29.

4.5 Caso Experimental IV

Créput e Koukam (2008) apresentaram uma série de resultados com diferentes tipos de combinações dos parâmetros envolvidos em sua técnica (Memetic SOM), onde era analisada a qualidade do resultado juntamente com o tempo de execução. Cada instância apresentada por eles foi rodada 10 vezes e a porcentagem do desvio do resultado médio em relação ao

melhor caminho foi exposta. A Tabela 4.5 ilustra os melhores resultados apresentados por eles sem levar em conta as separações por parâmetros.

Tabela 4.5 - Comparação dos resultados obtidos entre memetic SOM e RRW + PS com os respectivos coeficientes para os menores caminhos obtidos.

TSPLIB	*n*	Melhor Caminho	Memetic SOM	RRW + PS	Coeficientes
eil51	51	426	1,64%	103,7%	*A = 0,01; a = 0,10; c = 1,00*
berlin52	52	7.542	0%	118,5%	*A = 0,01; a = 0,10; c = 1,00*
st70	70	675	0,59%	203,7%	*A = 0,01; a = 0,1667; c = 1,00*
eil76	76	538	1,86%	143,3%	*A = 0,01; a = 0,1667; c = 1,00*
rd100	100	7.910	0,43%	243,6%	*A = 0,5; a = 0,1667; c = 1,00*
eil101	101	629	2,07%	274,9%	*A = 0,01; a = 0,1667; c = 1,00*
bier127	127	118.282	1,25%	201,6%	*A = 0,02; a = 0,1667; c = 1,00*
rat195	195	2.323	4,69%	303,9%	*A = 0,02; a = 0,1667; c = 1,00*
kroa200	200	29.368	0,70%	719,8%	*A = 0,5; a = 0,10; c = 1,00*

Das instâncias apresentadas aqui resta ilustrar o gráfico de convergência de berlin52, Figura 4.16, assim como eil51 que possui número de cidades similar. O gráfico convergiu para um mínimo local, alcançando um resultado semelhante para os mesmos valores dos coeficientes.

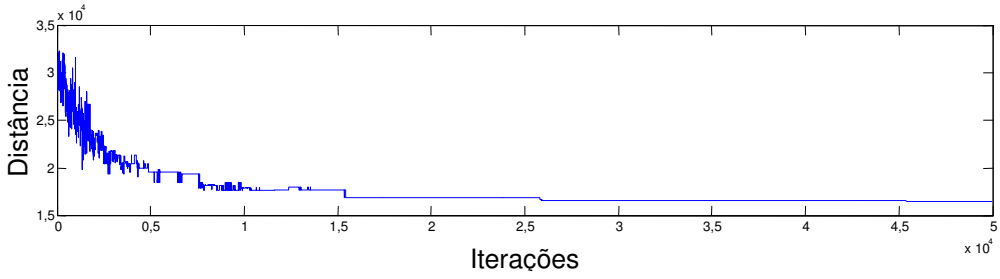


Figura 4.16 – Convergência para um mínimo local em função das iterações para berlin52.

CAPÍTULO 5

Conclusões Gerais e Trabalhos Futuros

5.1 Conclusões Gerais

A técnica de PS demonstrou-se capaz de ajustar os pesos sinápticos de forma a conduzir a RRW a atingir um mínimo local em diferentes instâncias da biblioteca TSPLIB. Foi identificado, porém, uma forte dependência na qualidade dos resultados em função dos coeficientes *A*, *a* e *c*. Além disso, a topologia de rede recorrente adotada aumenta a complexidade, com implicações diretas no processamento, à medida que o número de cidades *n* aumenta na proporção de $2.n^3$-n^2. Isto acaba influenciando na dificuldade do ajuste desses coeficientes, pois há um desvio muito grande dos valores adequados à medida que *n* aumenta, com a agravante de o custo de processamento aumentar na mesma proporção, o que torna praticamente inviável um ajuste baseado na tentativa e erro.

Foi possível constatar o papel de *A* na estabilidade, conforme mostra a Figura 4.6, pois quando seu valor é igual a zero o sistema oscila muito e não sinaliza uma tendência de convergência. Mesmo na comparação entre as Figuras 4.10 e 4.11, é possível verificar que mantendo os coeficientes iguais e apenas reduzindo o valor de *A* o gráfico passa de uma condição de estabilidade em um mínimo local para grandes oscilações, mesmo demonstrando tendência de convergência. Também foi identificada sua influência em relação aos resultados, pois à medida que seu valor se eleva o resultado piora, conforme valor obtido de rd100 na

Tabela 4.3, em que mesmo havendo estabilidade em um mínimo local o desempenho esteve pior que as demais instâncias que se estabilizaram em um mínimo local.

Os melhores resultados foram obtidos das instâncias com menor número de cidade justamente pela facilidade em se determinar os valores de *A*, *a* e *c* empiricamente e pelo estudo sistemático realizado em ulysses22 no Capítulo 4 (Seção 4.1).

5.2 Trabalhos Futuros

Para futuros trabalhos, a fim de aperfeiçoar os resultados e possibilitar que instâncias com um número maior de cidades sejam testadas com RNR e PS, sugere-se que a topologia da rede seja alterada a fim de que haja uma maior simplificação no número de pesos sinápticos e, conseqüentemente, no tempo de processamento neural. Dessa forma, seria possível implementar um sistema de auto-regulação dos coeficientes *A*, *a* e *c*. Se uma simplificação da RNR for realizada, torna-se possível desenvolver um sistema de ajuste dos coeficientes através da medida da derivada dos resultados do gradiente passados sobre um filtro como realimentação para os ajustes.

Ao final do processo de procura de um mínimo local realizado pela rede, um sistema heurístico de procura como 2-opt (SIQUEIRA, SCHEER e STEINER, 2007) poderia ser usado para otimizar o resultado.

Por fim, na sistemática da determinação da rota, após a RRW ter convergido conforme descrito no Capítulo 3, Seção 3.4.1, a lógica de escolha para o próximo nó pode ser alterada, ao invés de sempre começar pela primeira coluna equivalente da matriz de custo, poderia começar pela coluna onde se encontrar o maior valor da saída.

Referências Bibliográficas

AHUJA, R.K., MANGNANTI, T.L. e ORLIN, J.B. (1993). *Network Flows*. Prentice-Hall, New Jersey.

APPLEGATE, D. L., BIXBY, R. E., CHVÁTAL, V. et al (2007). *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, Princeton, New Jersey, USA.

BAZARAA, M.S., JARVIS, J.J., SHERALI, H.D. (1990). *Linear Programming and Network Flows*. Wiley, New York.

BENINI, F. A. V., SILVA, I. N. (2008). *Uma abordagem usando Colônia de Formigas e inteligência artificial para o Problema do Caixeiro Viajante.* 7th Brazilian Conference on Dynamics, Control and Applications. 07-09 maio 2008.

CHEN, J., HUANG, T. C. (2004). *Applying neural networks to on-line updated PID controllers for nonlinear process control.* Journal of Process Control, v. 14, issue 2, pp. 211 – 230.

COELHO, L. S., MARIANI, V. C. (2006). *Evolução diferencial híbrida com programação quadrática aplicada ao problema de despacho econômico de energia elétrica.* Revista Controle & Automação, v. 17, n. 4, pp. 409-423.

COELHO, L. S., TAVARES NETO, R. F. (2004). *Colônia de Formigas: Uma Abordagem Promissora para Aplicações de Atribuição Quadrática e Projeto de Layout.* XXIV ENEGEP, Florianópolis/SC, novembro.

CUNHA, C. B., BONASSER, U. O. e ABRAHÃO, F. T. M. (2002). *Experimentos Computacionais com Heurísticas de Melhorias para o Problema do Caixeiro Viajante.* Anais do XVI Congresso de Pesquisa e Ensino em Transportes, ANPET, Natal/RN, v. 2, p. 105-117.

DAI, W., WANG, P. (2007). *Application of Pattern Recognition and Artificial Neural Network to Load Forecasting in Electric Power System.* Third International Conference on Natural Computation 2007, v. 1, pp. 381 – 385.

DANTZIG, G., FULKERSON, R. e JOHNSON, S. (1954). *Solution of a Large-Scale Traveling-Salesman Problem.* Journal of the Operations Research Society of America 2, pp. 393-410.

DORIGO, M., & GAMBARDELLA, L. M. (1997). *Ant colony system: A cooperative learning approach to the traveling salesman problem.* IEEE Transaction on Evolutionary Computation, l(1), pp. 53 – 66.

FLOOD, M. M. (1956). *The Traveling-Salesman Problem*. Operations Research, vol. 4, n. 1, pp. 61-75, fevereiro.

FREDERICO, C.V., ADIAO, D.D.N., JOSE, A.F. (2003). *An efficient approach to the traveling salesman problem using self-organizing maps*. International Journal of Neural Systems 13(2), pp. 59–66

FROLOV, A. A., DUFOSSE, M., BENSMAIL, S., OUEZDOU, F. B. (2002). *Biologically inspired neural network approach to manipulator movement control.*

FURTADO, J. C. (1998). *Algoritmo Genético Construtivo na Otimização de Problemas Combinatoriais de Agrupamentos*. Tese de Doutorado em Computação Aplicada no INPE.

GONZALES, R.C., WOODS, R.E. (2000). *Processamento de Imagens Digitais*. São Paulo: Edgard Blücher Ltda, pp. 509.

GOTO, A. e KAWAMURA, M. (2008). *Solution Method Using Correlated Noise for TSP.* Lec. Notes in Comp. Science, v. 4984/2008, pp. 733-741.

HOPFIELD, J. J. (1982). *Neural networks and physical systems with emergent collective computational abilities*. Proc. Natl. Acad. Sci. USA, v. 79, pp. 2554-2558, abril.

HOPFIELD, J.J. e TANK, D.W. (1985). *"Neural" computation of decisions in optimization problems*. Biological Cybernetics, v. 52, pp. 141–152.

HUNG, D.L. e WANG, J. (2003). *Digital hardware realization of a recurrent neural network for solving the assignment problem*. Neurocomputing 51, pp. 447–461.

KING, D., FLANAGAN, C., LYONS, W. B., LEWIS, E. (2002). *An optical fibre water sensor utilizing signal processing techniques and artificial neural network pattern recognition*. Proceedings of IEEE Sensors. v. 2, pp. 1374 – 1378.

LEUNG, K. S., JIN, H. D. e XU, Z. B. (2004). *An expanding self-organizing neural network for the traveling salesman problem.* Neurocomputing, v. 62, pp. 267 – 292.

MAEDA, Y., HIRANO, H. e KANATA, Y. (1993). An *Análogo Network Circuit with a Learning Rule via Simultaneous Perturbation.* Proceedings of 1993 International Joint Conference on Neural Networks, pp. 853-856

MAEDA, Y., HIRANO, H. e KANATA, Y. (1995). *A Learning Rule of Neural Networks via Simultaneous Perturbation and Its Hardware Implementation*. Pergamon, Neural Networks, vol. 8, n. 2, pp. 251-259.

MAEDA, Y. e WAKAMURA, M. (2005). *Simultaneous Perturbation Learning Rule for Recurrent Neural Networks and Its FPGA Implementation*. IEEE Trans. On Neural Networks, vol. 16, n. 6, pp. 1664-1672, novembro.

MA, M., ZHANG, L. B. (2007). *Optimizing a Fuzzy Neural Network with a hierarchical Genetic Algorithm.* Proceedings of the Sixth International Conf. on Machine Learn. And Cyb., pp. 2812-2815, Hong Kong, 19-22 agosto.

MCCULLOCH, W. S. e PITTS, W. (1943). *A Logical Calculus of the Ideas Immanent in Nervous Activity.* Bulletin of Mathematical Biophysics, v. 5, pp 115-133.

MINSKY, M., PAPERT, S. (1969). *Perceptrons, An Introduction to Computational Geometry*. MIT Press, Cambridge, Mass.

PALMEIRA, M. M. (1999). *Um algoritmo relax-and-cut para o problema quadrático da mochila 0-1.* Pontifícia Universidade Católica do Rio de Janeiro, Mestrado, 13 de julho de 1999.

REINELT, G. (1991). *TSPLIB—A traveling salesman problem library*. ORSA Journal on Computing, 3, 376–384.

ROSENBLATT, F. (1959). *Two theorems of statistical separability in the perceptron*. In M. Selfridge (Ed.), Mechanisation of thought processes: Proceedings of a symposium held at the National Physical Laboratory. London: HM Stationary Office.

RUMELHART, D.E., HINTON, G.E., WILLIAMS, R.J. (1986). *Learning internal representation by back- propagating errors*. In: Rumelhart, D.E., McCleland, J.L., the PDP Research Group (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press, MA.

SALCEDO-SANZ, S., ORTIZ-GARCÍA, E. G., PÉREZ-BELLIDO, A. M. et al (2008). *Using a bank of binary Hopfield networks as constraints solver in hybrid algorithms*. Neurocomputing, vol. 71, pp. 1061-1068.

SIQUEIRA, P. H. (2005). *Uma Nova Abordagem na Resolução do Problema do Caixeiro Viajante.* Tese de doutorado apresentado na Universidade Federal do Paraná, Curitiba 2005.

SIQUEIRA, P. H., SCHEER, S. e STEINER M. T. A. (2006). *A New Neural Network Aproach to the Traveling Salesman Problem.* Lecture Notes in Computer Science – Springer.

SIVAGAMINATHAN, R. K. e RAMAKRISHNAN, S. (2007). *A hybrid approach for feature subset selection using neural networks and ant colony optimization.* Expert Systems with Applications, v. 33, pp. 49 – 60.

SOUZA, P. S. (1993). *Asynchronous organizations for multi-algorithms problems*. Tese de Doutorado na University Carnegie Mellow, Department of Electrical and Computer Engineering, Pittsburgh.

SPALL, J. C. e CRISTION, J. A. (1992). *Stochastic approximation for neural network weight estimation in the control of uncertain nonlinear systems.* Neural Networks, 1992. IJCNN., International Joint Conference on, v. 3, pp. 930-935, 7 a 11 de junho.

SPALL, J. C. (1987). *A stochastic approximation technique for generating maximum likelihood parameter estimates*. In Proceedings of the American Control Conference, pp. 1161-1167.

SPALL, J. C. (1992). *Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation.* IEEE Trans. On Automatic Control, v. 37, n. 3, pp. 332-341, março.

SPALL, J. C. (1998). *An overview of the simultaneous perturbation method for efficient optimization*. Johns Hopkins APL technical digst, vol. 19, n. 4, pp. 482-492.

TALAVÁN, P. M., YÁÑEZ, J. (2002). *Parameter setting of the Hopfield network applied to TSP*. Neural Networks, v. 15, pp. 363-373.

TALAVÁN, P. M., YÁÑEZ, J. (2006). *The generalized quadratic knapsack problem. A neuronal network approach.* Neural Networks, v. 19, pp. 416-428.

TAN, K. C., TANG, H., GE, S. S. (2005). *On Parameter Settings of Hopfield Networks Applied to Traveling Salesman Problems.* IEEE Transactions on Circuits and Systems, v. 52, n. 5, pp. 994-1002, maio.

TEOH, E. J., TAN, K. C., TANG, H. J. et al. (2008). *An asynchronous recurrent linear threshold network approach to solving the traveling salesman problem.* Neurocomputing, v. 71, pp 1359 – 1372.

XU, X. e TSAI, W. T. (1991). *Effective Neural Algorithms for the Traveling Salesman Problem*. Pergamon Press plc, Neural Networks, vol. 4, pp. 193-205.

WANG, J. (1992), Analog Neural Network for Solving the Assignment Problem. Electronic Letters, v. 28, n. 11, p. 1047-1050, maio 1992.

WANG, J., (1996). *A Recurrent Neural Network for Solving the Shortest Path Problem.* IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, vol. 43, n. 6, pp. 482-486, june 1996.

WANG, R. L., TANG, Z. e CAO, Q. P. (2002). *A learning method in Hopfield neural network for combinatorial optimization problem.* Neurocomputing. Vol. 48, pp. 1021-1024.

WIDROW, B., HOFF, M. E. (1960). *Adaptive switching circuits.* In 1960 WESCON Convention Record, New York.

WILSON, G.V., PAWLEY, G.S. (1988). *On the stability of the traveling salesman problem algorithm of Hopfield and Tank.* Biological Cybernetics. Vol. 58, no. 1, pp. 63-70.

ZHANG, W. D., BAI, Y. PING, HU, H. P. (2006). *The incorporation of an efficient initialization method and parameter adaptation using self-organizing maps to solve the TSP.* Applied Mathematics and Computation, v. 172, pp. 603-623.