Pós-Graduação em Ciência da Computação

"Otimização Global em Redes Neurais Artificiais"

Por

Cleber Zanchettin

Tese de Doutorado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

Recife, 02/2008

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Cleber Zanchettin

"Otimização Global em Redes Neurais Artificiais"

Tese apresentada à Pós-Graduação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Doutor em Ciência da Computação.

Orientadora: Dra. Teresa Bernarda Ludermir
Co-orientador: Dr. Tsang Ing Ren

Recife, Fevereiro/2008

Zanchettin, Cleber
Otimização global em redes neurais artificiais / Cleber Zanchettin. – Recife: O Autor, 2008.
xvi, 146 p.: il., fig., tab.

Tese (doutorado) – Universidade Federal de Pernambuco. CIn. Ciência da Computação, 2008.

Inclui bibliografia e apêndice.

1. Redes Neurais (Ciência da Computação) I. Título.

006.32 CDD (22.ed.) MEI2008-036

Dedicatória

Para minha família.

Agradecimentos

Agradeço a todos que auxiliaram, direta ou indiretamente, no desenvolvimento desse trabalho, tornando-o uma realidade ao contrário de uma pretensão.

Especialmente a professora Teresa Bernarda Ludermir, por ter acreditado e confiado em mim desde o momento em que nos conhecemos.

A minha namorada Ana Bárbara Guedes Alcoforado Farias pelo carinho, compreensão e apoio incondicional.

Aos meus irmãos Joel da Silva e Tarcisio Pinto Câmara pelas infindáveis discussões acadêmicas, pessoais e culinárias.

Aos amigos Adélia, Afonso, Alexandrino, Ana, Angélica, Átila, Bruno, Byron, Cynthia, Eduardo, Elias, George, Jennifer, Juliano, Luciana, Oscar, Paulo, Rodrigo, Sineco, Simith e Valmir pelo companheirismo e confiança.

Todos vocês fazem parte desta conquista.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

ANFIS	*Adaptative Neuro Fuzzy Inference System*
ANOVA	*Analysis of Variance*
BP	*Backpropagation*
CO_2	Dióxido de Carbono
ECOS	*Evolving Conectionist Systems*
EFuNN	*Evolving Fuzzy Neural Network*
GA	*Genetic Algorithms*
GaTSa	*Tabu Search, Simulated Annealing e Genetic Algorithms*
GL	*Generalization Lost*
ITO	Substrato Víteo Condutor
k-NN	*K – Nearest Neighbors*
MLP	*Multi-layer Perceptron*
MO	Multiobjetivo
MP	Média Ponderada
SA	*Simulated Annealing*
TS	*Tabu Search*
TSa	*Tabu Search e Simulated Annealing*
UCI	*University of California*
WD	*Weight Decay*

Resumo

Zanchettin, C. (2008) **Otimização Global em Redes Neurais Artificiais**, *Tese (Doutorado)* – Centro de Informática, Universidade Federal de Pernambuco, Recife, 2008, 146p.

Esta tese apresenta um método de otimização global e local, baseado na integração das heurísticas das técnicas *Simulated Annealing*, *Tabu Search*, Algoritmos Genéticos e *Backpropagation*.

O desempenho deste método é investigado na otimização simultânea da topologia e dos valores dos pesos das conexões entre as unidades de processamento de redes neurais artificiais *Multi-layer Perceptron*, a fim de gerar topologias com poucas conexões e alto desempenho para qualquer conjunto de dados. A heurística proposta realiza a busca de forma construtiva e baseada na poda das conexões entre as unidades de processamento da rede. Assim, são geradas redes com arquitetura variável e que podem ser ajustadas para cada problema de forma automática.

Experimentos demonstram que o método pode também ser utilizado para a seleção de atributos relevantes. Durante a otimização da arquitetura da rede, unidades de processamento de entrada podem ser eliminadas de acordo com sua relevância para o desempenho do modelo. Desta forma, é obtida uma seleção de atributos inerente ao processo de otimização das redes neurais artificiais.

Os principais parâmetros de configuração do método tiveram sua influência estimada através da técnica de planejamento fatorial de experimentos. Com base no planejamento fatorial de experimentos, foi possível verificar a influência, interação e a inter-relação entre os parâmetros de configuração do modelo. Estas análises são importantes para identificar a influência de cada parâmetro e possivelmente diminuir a quantidade de parâmetros ajustáveis no projeto deste método.

Nesta tese são realizados experimentos com cinco diferentes bases de dados de classificação e duas bases de dados de previsão. A técnica proposta apresentou resultados estatisticamente relevantes em comparação com outras técnicas de otimização global e local.

Palavras-chave – Otimização global; redes neurais artificiais; seleção de atributos relevantes; planejamento de experimentos.

Abstract

Zanchettin, C. (2008) **Global Optimization in Artificial Neural Networks**, *Thesis (PhD)* – Centro de Informática, Universidade Federal de Pernambuco, Recife, 2008, 146p.

In this thesis, a global and local optimization method is presented. This method is based on the integration of the heuristic Simulated Annealing, Tabu Search, Genetic Algorithms and Backpropagation.

The performance of the method is investigated in the simultaneous optimization of Multi-layer Perceptron artificial neural network architecture and weights. The aim is to achieve topologies with few connections and high performance in any dataset. The proposed heuristics perform the search in a constructive way and based on the pruning of irrelevant connections among the network processing units. Thus, the method yields networks with flexible architecture and can be adjusted automatically.

Experiments demonstrated that the method can also be used for relevant feature selection. During the topology network optimization, input processing units may be eliminated in accordance with the performance of the method. Thus, a feature selection is obtained that is inherent to the artificial neural network optimization process.

The main configuration parameters of the method have their influence estimated by factorial experimental design, through which it is possible to determine the influence, interaction and inter-relationship among the configuration parameters of the method. These analyses are important in identifying the influence of each parameter and possibly reduce the number of adjustable parameters in the project of the method.

Experiments are performed in this thesis, with five different classification and two prediction datasets. The proposed technique presented statistically relevant results in comparison with other global and local optimization techniques.

Keywords – *Global optimization; artificial neural networks; relevant feature selection; experimental design.*

Capítulo 1

Introdução

Otimização é o processo de encontrar a melhor solução em um conjunto de possíveis soluções para um problema. Um problema de otimização possui uma função objetivo – a qual permite o cálculo da aptidão de cada solução para o problema – e um conjunto de restrições a serem satisfeitas, ambas relacionadas às variáveis de decisão investigadas (Papalambros e Wilde, 2000). Os algoritmos evolucionários (Bäck, et al, 1997), *Simulated Annealing* - SA (Kirkpatrick, 1983) e *Tabu Search* -TS (Glover e Laguna, 1997) são, em geral, algoritmos iterativos para resolução de problemas de otimização combinatória. O termo algoritmo evolucionário é utilizado para se referir a qualquer algoritmo probabilístico, cujo desenvolvimento seja inspirado por mecanismos evolucionários biológicos. Um dos algoritmos mais conhecidos desta categoria são os Algoritmos Genéticos - AG (Goldberg, 1989).

Estes três algoritmos são os mais populares, de uma classe de algoritmos de otimização, conhecidos como algoritmos iterativos gerais. A heurística de busca dos três algoritmos possui várias similaridades (Sait e Youssef, 1999): *(i)* são algoritmos de aproximação, ou seja, não garantem encontrar uma solução ótima; *(ii)* são cegos, pois não sabem quando encontraram uma solução ótima. Também não sabem quando interromper a busca a esta solução; *(iii)* possuem a capacidade de ocasionalmente aceitar movimentos que se afastem de uma solução ótima; *(iv)* são gerais, pois podem facilmente implementar uma diversidade de problemas, bastando para esta tarefa, adaptar a representação da solução, função de custo e o mecanismo para gerar novas soluções sobre o espaço de busca; e *(v)* em certas condições, podem assintoticamente convergir para uma solução ótima.

A proposta desta tese é construir uma nova técnica de otimização baseada na integração das principais potencialidades destas três heurísticas. A idéia surgiu a partir da constatação de suas similaridades e potenciais características complementares. A abordagem proposta é avaliada na

otimização simultânea do número de conexões e no valor das conexões entre unidades de processamento de uma rede neural artificial *Multi-layer Perceptron* - MLP (Rumelhart e Weigend, 1990).

A rede *Multi-layer Perceptron* treinada com o algoritmo *Backpropagation* – BP (Rumelhart et al, 1986) é um dos modelos conexionistas mais utilizados na literatura (Haykin, 2001). Para obtenção de sucesso em sua utilização, um fator muito importante é a definição da topologia da rede. Um número insuficiente de conexões pode tornar a rede incapaz de resolver o problema investigado, pela insuficiência de parâmetros ajustáveis (*undefitting*), enquanto que o excesso pode provocar sobre-treinamento (*overfitting*) nos dados de treinamento.

Em geral, o treinamento das redes neurais artificiais é realizado através de repetidas tentativas com diferentes topologias de rede, até serem obtidos resultados satisfatórios para o problema. Além de consumir tempo e exigir experiência no treinamento de redes neurais artificiais, este processo pode obter redes com conexões e unidades de processamento desnecessárias. Sem considerar o fato de que quanto maior a topologia da rede, mais complexo é o ajuste do valor destas conexões. Assim, a otimização simultânea de arquiteturas e pesos de redes neurais artificiais é uma abordagem interessante para a geração de redes eficientes com topologias pequenas. A versatilidade do método proposto permite ainda sua utilização para seleção dos atributos relevantes das bases de dados investigadas, permitindo a redução da dimensionalidade do problema e a complexidade das redes neurais otimizadas.

Devido às características do problema, que possui conhecidamente grande quantidade de mínimos locais, para obtenção de bons resultados neste contexto, faz-se necessário utilizar uma abordagem que possa encontrar boas soluções em torno de mínimos locais. Também é conhecido que técnicas de otimização global, são relativamente ineficientes para o ajuste fino em buscas locais. Assim, foi adicionada ao método proposto à heurística da técnica de busca local *Backpropagation*, a fim de garantir um ajuste mais preciso das soluções geradas, obtendo arquiteturas de rede com maior poder de generalização.

Segundo Yamazaki (2004), um dos problemas na otimização da arquitetura de redes neurais artificiais, quando os pesos das conexões não são codificados nas soluções, é que a avaliação das redes passa a apresentar ruído, pois uma rede treinada com um conjunto completo de pesos é utilizada para avaliar o custo de uma solução, que não contém nenhuma informação sobre os valores dos pesos entre as unidades de processamento. As diferenças nos parâmetros de treinamento e nas inicializações dos pesos podem gerar resultados distintos para a mesma topologia. Desta forma, para evitar este problema, foi optado por otimizar simultaneamente a arquitetura da rede e os pesos das conexões entre as unidades processamento. Segundo esta abordagem, cada ponto do espaço de busca codifica tanto a estrutura topológica da rede como os

valores das conexões entre as unidades de processamento, tornando a avaliação do custo mais precisa.

Outra característica interessante no desenvolvimento de sistemas automáticos de otimização, para modelos baseados em aprendizagem de máquina, é verificar qual dos parâmetros de configuração são relevantes para o desempenho médio do modelo otimizado, visto a grande quantidade de parâmetros ajustáveis destas técnicas. Desta forma, o método de otimização pode focar a busca nos parâmetros estatisticamente relevantes, diminuindo a complexidade e o tempo de otimização. O método de *planejamento fatorial de experimentos* (Montgomery, 1997), é uma técnica estatística com esta capacidade e neste trabalho foi utilizada com este fim na análise dos parâmetros de configuração de dois modelos neuro-difusos e da técnica proposta.

1.1 Objetivo

O objetivo deste trabalho é a proposição, implementação e a análise de um método de otimização global e local, baseado na integração das heurísticas de busca *Simulated Annealing*, *Tabu Search*, Algoritmos Genéticos e *Backpropagation* (GaTSa). A efetividade deste método é investigada na otimização simultânea da topologia e dos valores dos pesos das conexões entre as unidades de processamento de redes neurais artificiais *Multi-layer Perceptron*, a fim de gerar topologias com poucas conexões e alto desempenho para qualquer conjunto de dados. A heurística proposta realiza a busca de forma construtiva e baseada na poda das conexões entre as unidades de processamento da rede. Assim, devem ser geradas redes com arquitetura variável e que podem ser ajustadas para cada problema de forma automática.

1.2 Contribuições desta Tese

Em especial, este trabalho propõe-se a trazer os seguintes avanços e contribuições:

(i) Proposta de um método de otimização que combine as potencialidades das principais técnicas de otimização global e local;

(ii) Otimização de redes neurais artificiais *Multi-layer Perceptron* de forma construtiva e baseada na poda das conexões da rede. Objetiva-se evitar que o projetista da rede necessite definir uma arquitetura máxima para a topologia da rede neural artificial;

(iii) Verificar se a abordagem pode ser utilizada para seleção de atributos relevantes. Durante a otimização da arquitetura da rede, unidades de processamento de entrada

podem ser eliminadas de acordo com sua relevância para o desempenho do modelo. Objetiva-se verificar se estes atributos, representados pelas unidades de processamento eliminadas, são mesmo irrelevantes na base de dados, desta forma, seria obtida uma seleção de atributos inerente;

(iv) Identificar o comportamento do método de forma sistemática e controlada, utilizando técnicas estatísticas de experimentação científica;

(v) Verificar o desempenho da técnica para diferentes tipos de problema – classificação e previsão;

(vi) Identificar os parâmetros mais relevantes no desempenho dos modelos baseados em aprendizagem, verificando suas influências e possivelmente diminuindo a quantidade de parâmetros ajustáveis no projeto destes modelos;

(vii) Comparação entre a técnica e diferentes métodos de otimização.

Partes destes tópicos geraram as seguintes publicações científicas:

Zanchettin, C. e Ludermir, T.B. (2007) "Global Optimization Methods for Designing and Training Feedforward Artificial Neural Networks", **Dynamics of Continuous, Discrete & Impulsive Systems, Series A: Mathematical Analysis – A Supplement, Advances in Neural Networks**, Watam Press, Vol. 14(S1) pp. 328–337.

Zanchettin, C., Minku, F.L. e Ludermir, T.B. (2007) "Design of Experiments in Neuro-Fuzzy Systems" (Submetido - artigo convidado pelo jornal), **Journal of Intelligent & Fuzzy Systems**, IOS Press.

Ludermir, T.B., Yamazaki, A. e Zanchettin, C. (2006) "An Optimization Methodology for Neural Network Weights and Architectures", **IEEE Transactions on Neural Networks**, IEEE Computer Society, vol. 17, n. 5, pp. 1452–1459.

Zanchettin, C. e Ludermir, T.B. (2008). "Feature Subset Selection in a Methodology for Training and Improving Artificial Neural Network Weights and Connections" (Artigo submetido), **Proceedings of the International Joint Conference on Neural Networks (IJCNN 2008)**, IEEE Computer Society, Hong Kong – China.

Zanchettin, C. e Ludermir, T.B. (2007) “Comparison of the Effectiveness of Different Cost Functions in Global Optimization Techniques”, **Proceedings of the International Joint Conference on Neural Networks (IJCNN 2007)**, IEEE Computer Society, Orlando – EUA, vol. 1, pp. 2701–2706.

Zanchettin, C. e Ludermir, T.B. (2006) “A Methodology to Training and Improve Artificial Neural Networks` Weights and Connections”, **Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006)**, IEEE Computer Society, Vancouver – Canada, vol. 1, pp. 10723–10730.

Zanchettin, C. e Ludermir, T.B. (2006) “The Influence of Different Cost Functions in Global Optimization Techniques”, **Proceedings of the Brazilian Symposium on Neural Networks (SBRN 2006)**. IEEE Computer Society, Ribeirão Preto – Brasil, vol. 1, pp. 17–23.

Zanchettin, C, Minku, F.L. e Ludermir, T.B. (2005) “Design of Experiments in Neuro-Fuzzy Systems”, **Proceedings of the Hybrid Intelligent Systems (HIS2005)**. IEEE Computer Society, Rio de Janeiro – Brasil, vol. 1, pp. 218–226.

Zanchettin, C. e Ludermir, T.B. (2005) “Hybrid Technique for Artificial Neural Network Architecture and Weight Optimization” **Proceedings of the 16th European Conference on Machine Learning (ECML) and the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD),** Springer-Verlag Lecture Notes in Artificial Intelligence. Porto - Portugal, vol. 3721, pp. 709–716.

Zanchettin, C. e Ludermir, T.B. (2005) “Técnica Híbrida de Otimização para Pesos e Arquiteturas de Redes Neurais Artificiais”, **Anais do Encontro Nacional de Inteligência Artificial (ENIA 2005)**. São Leopoldo - Brasil, vol. 1, pp. 47–52.

1.3 Organização Geral do Documento

A proposta de tese está organizada em dez capítulos de acordo com o fluxograma apresentado na Figura 1.1.

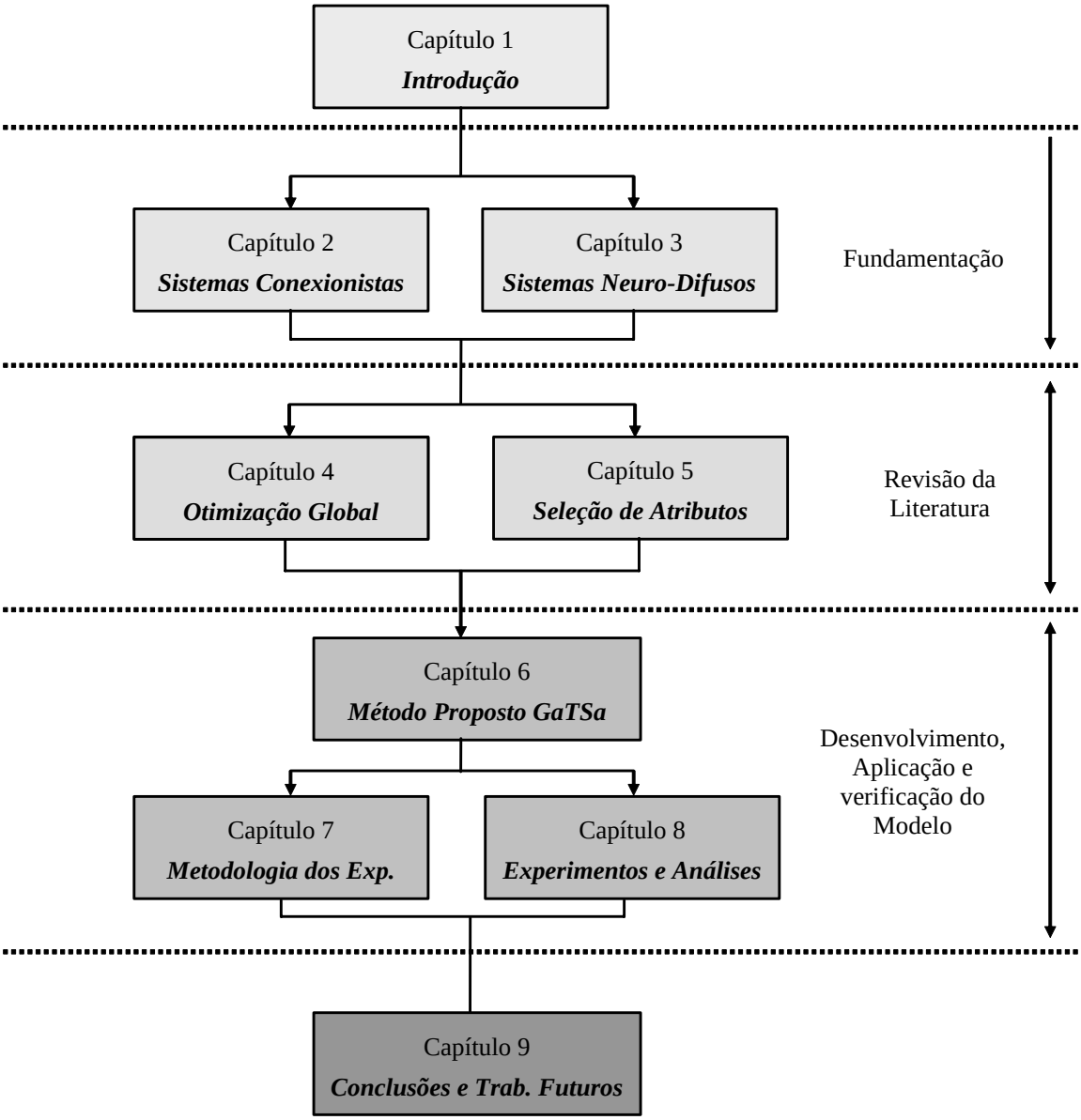

Figura 1.1 – Estrutura de desenvolvimento da tese.

CAPÍTULO 2: Sistemas Conexionistas.

Este capítulo apresenta conceitos básicos sobre sistemas conexionistas, principalmente sobre o foco do trabalho, que são as redes neurais artificiais. São apresentados detalhes de projeto, implementação e treinamento desta técnica, a fim de familiarizar o leitor com o objetivo e as potencialidades do trabalho.

CAPÍTULO 3: Sistemas Neuro-difusos.

Neste capítulo são apresentados dois sistemas neuro-difusos. Estes sistemas são utilizados nas simulações que objetivam verificar a potencialidade do planejamento fatorial de experimentos na

definição e controle sobre os parâmetros de configuração das técnicas de aprendizagem. Os sistemas abordados foram *Adaptative Neuro Fuzzy Inference System* - ANFIS (Jang, 1993) e *Evolving Fuzzy Neural Network – EFuNN (*Kasabov, 1998b).

CAPÍTULO 4: Otimização Global.

O capítulo apresenta uma revisão sobre as principais técnicas de otimização global apresentadas na literatura, destacando detalhes de implementação e as principais características que motivaram sua utilização na construção da técnica proposta. O capítulo é dividido pelas técnicas: *(i)* Algoritmos Genéticos; *(ii) Simulated Annealing*; e *(iii) Tabu Search*. Neste capítulo também é apresentada uma revisão bibliográfica com vários trabalhos relacionados à otimização de redes neurais artificiais. São descritos vários trabalhos que integram de forma total ou parcial as heurísticas das técnicas de busca Algoritmos Genéticos, *Simulated Annealing* e *Tabu Search* ou que propõe otimizar a arquitetura ou os pesos de redes neurais artificiais.

CAPÍTULO 5: Seleção de Atributos Relevantes.

Neste capítulo é apresentada uma revisão sobre técnicas clássicas de seleção de atributos relevantes, destacando detalhes de implementação. O capítulo é dividido pelas técnicas: *(i) Hill-Climbing* (Vafaie e De Jong, 1993); *(ii) Random Bit Climber* (Davis, 1991); *(iii) Best-First* (Aha e Banker, 1995); *(iv) Beam Search* (Aha e Banker, 1995); e *(v) Las Vegas* (Liu e Setiono, 1996).

CAPÍTULO 6: Método Proposto GaTSa.

O capítulo apresenta o método proposto que integra às heurísticas das técnicas de busca *Simulated Annealing*, *Tabu Search*, Algoritmos Genéticos e *Backpropagation*. São descritas suas principais características de funcionamento, detalhes de implementação e potencialidades.

CAPÍTULO 7: Metodologia dos Experimentos.

Neste capítulo são descritas as bases de dados utilizadas nas simulações realizadas durante o desenvolvimento da tese. Também são expostos detalhes sobre a realização dos experimentos, como a metodologia de treinamento, pré-processamento dos dados, seleção de atributos relevantes, métodos utilizados na comparação das abordagens e o planejamento fatorial de experimentos aplicado sobre os modelos.

CAPÍTULO 8: Experimentos e Análises.

Neste capítulo são apresentados os experimentos realizados a fim de verificar as contribuições propostas na tese. Primeiramente verificou-se, estatisticamente, a relevância e influência de cada

parâmetro de configuração de dois sistemas neuro-difusos, através da aplicação do planejamento fatorial de experimentos. Posteriormente pode ser verificada a evolução do trabalho a partir dos diferentes experimentos realizados com o método proposto. As simulações abordam vários aspectos relevantes à seleção de atributos, otimização da topologia e valores dos pesos de redes neurais artificiais e planejamento fatorial de experimentos em diferentes bases de dados.

CAPÍTULO 9: Conclusões e Trabalhos Futuros.

Este capítulo traz um resumo do que foi exposto ao longo deste documento, bem como diversas sugestões para continuação da pesquisa com métodos de otimização global.

Um apêndice acompanha este documento. Este apêndice contém a teoria do Planejamento Fatorial de Experimentos, utilizado na análise da relevância estatística de cada parâmetro de configuração dos modelos investigados.

Capítulo 2

Sistemas Conexionistas

As redes neurais artificiais, ou sistemas conexionistas constituem uma representação computacional de um sistema de processamento de informação com desempenho e características semelhantes às redes neurais biológicas. Estes sistemas conexionistas correspondem a um modelo matemático inspirado na estrutura neural de organismos inteligentes, que adquirem conhecimento através da experiência. A base desta estrutura é a interconexão de pequenas unidades de processamento que compartilham e processam informações de forma paralela (Haykin, 2001).

2.1 Definição

Uma rede neural artificial é constituída por elementos computacionais muito simples, designados por unidades de processamento, os quais processam sinais ponderados pelos pesos associados às conexões que interligam estas unidades de processamento. As unidades de processamento estão estruturadas em camadas, e geralmente as camadas somente estão conectadas a camadas adjacentes. Porém, podem existir conexões interligando camadas não adjacentes. Desta forma, os sinais são propagados a partir das unidades de entrada para as unidades de processamento de saída. Sistemas conexionistas com estas características são conhecidos como redes neurais artificiais *feedforward*. Na Figura 2.1 é apresentado um exemplo da topologia de uma rede neural artificial *feedforward* e de suas unidades de processamento.

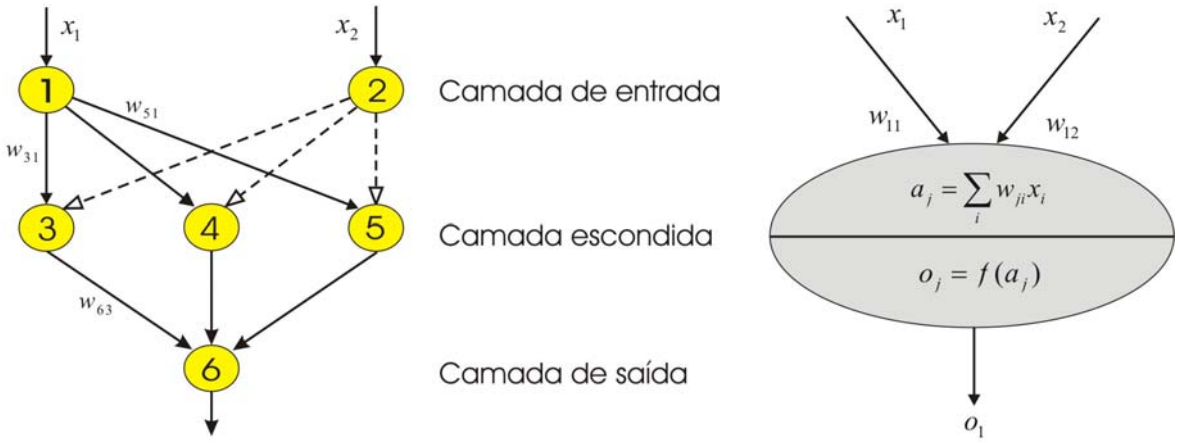

Figura 2.1 – Exemplo da topologia de uma rede neural artificial *feedforward* (esquerda) e de uma unidade de processamento (direita).

Cada unidade de processamento realiza operações elementares em suas entradas, de modo que o resultado da ativação desta unidade de processamento possa ser utilizado como entrada por outras unidades de processamento. As conexões funcionam como elementos de comunicação entre as unidades de processamento e possuem pesos que são utilizados para ponderar os sinais que transmitem, sendo que esta ponderação pode alterar ou não o sinal transportado. Uma conexão que vincule a saída de uma unidade de processamento *i* à entrada de uma unidade de processamento *j* é designada por w_{ji}. Considerando o_i como a saída da unidade de processamento *i*, o sinal de entrada da unidade de processamento *j* é designado por $w_{ji}o_i$. Os pesos w_{ji} são números reais que ponderam o sinal que transmitem, podendo atenuá-lo ou amplificá-lo, dependendo do seu valor.

Uma unidade de processamento pode estar conectada a uma ou várias outras unidades de processamento. Os sinais recebidos por uma unidade de processamento são avaliados e constituem a ativação a_j desta unidade, onde $a_j = \sum_j w_{ji}o_i$. Antes da ativação da unidade de processamento ser propagada para outras unidades de processamento, ao sinal de ativação é aplicada uma função linear ou não-linear $f(.)$, normalmente a saída de uma rede neural artificial não varia linearmente com sua ativação, assim $o_j = f(a_j)$. A função $f(.)$ é, em geral, uma função simétrica, monótona que tende para um valor finito quando o módulo de seu argumento cresce (Braga et al, 2000). A literatura apresenta uma variedade de funções que procuram preservar estas características com maior ou menor fidelidade (Braga et al, 2000; Haykin, 2001). Dentre estas abordagens podem ser destacadas as abordagens mais comuns: *(i)* função linear; *(ii)* função tangente hiperbólica; e *(iii)* a função sigmóide, descritas pelas Equações (2.1), (2.2) e (2.3), respectivamente.

$$f(a_j) = a_j \tag{2.1}$$

No caso de uma função de ativação linear, somente o comportamento monotônico é preservado, desprezando-se a saturação.

$$f(a_j) = \frac{1 - \exp(-a_j)}{1 + \exp(-a_j)} \tag{2.2}$$

$$f(a_j) = (1 + \exp(-a_j))^{-1} \tag{2.3}$$

Tanto a tangente hiperbólica quanto a função sigmóide preservam as características de monotonicidade e saturação.

Considerando o exposto, uma unidade de processamento *j* somente realiza o processamento elementar $o_j = f(\sum_i w_{ji} o_i)$. Além disto, um componente de entrada adicional *b*, conhecido como *bias*, pode ser inserido no modelo com o intuito de aumentar ou diminuir (se positivo ou negativo) o valor de entrada da função de ativação (Haykin, 2001). A adição deste termo à entrada pode fazer com que este caia dentro ou fora da região de ativação ou saturação da função de ativação. Desta forma a saída da unidade de processamento pode ser reescrita segundo a Equação (2.4).

$$o_j = f(b + \sum_i w_{ji} o_i) \tag{2.4}$$

Apesar de serem formadas por unidades de processamento que realizam operações elementares, as redes neurais artificiais têm obtido desempenhos promissores em diversas aplicações práticas. O poder desta técnica esta na agregação destas unidades de processamento em arquiteturas que podem realizar operações muito mais complexas. Existem diversos modelos conexionistas na literatura, inspirados em estruturas biológicas e baseados em unidades de processamento, porém, nesta tese somente serão abordadas redes neurais artificiais multicamadas. O formato de uma rede multicamada é semelhante ao de um grafo direcionado, de forma que cada nó representa uma unidade de processamento (similar ao exemplo da Figura 2.1).

2.2 Rede Neural *Multi-layer Perceptron*

Dentre os vários modelos de redes neurais artificiais existentes, as redes *Multi-layer Perceptron* (Rumelhart e Weigend, 1990) são as redes mais difundidas, parte por sua facilidade de

implementação, parte por sua simplicidade (Haykin, 2001). A arquitetura de uma rede neural artificial compreende principalmente o número de camadas escondidas e seus respectivos números de unidades de processamento. A topologia da rede compreende as camadas de entrada, escondida, saída e do número de unidades de processamento associadas a cada uma delas. Estes parâmetros são definidos de acordo com o problema tratado.

Em uma rede neural artificial *feedforward*, por exemplo, as saídas da última camada de processamento constituem uma função complexa dos sinais de entrada da rede neural e de todos os seus pesos. Assim, a saída da rede pode ser definida como $O = g(W, X)$, onde X representa os sinais de entrada, O as saídas da última camada da rede neural artificial e W os pesos da rede. Desta forma, o exemplo da Figura 2.1 pode ser matematicamente representado pela Equação (2.5).

$$o_1 = f_6(w_{63} f_3(w_{31}x_1 + w_{32}x_2) + w_{64} f_4(w_{41}x_1 + w_{42}x_2) + w_{65} f_5(w_{51}x_1 + w_{52}x_2)) \quad \textbf{(2.5)}$$

2.3 Treinamento

A propriedade mais importante das redes neurais artificiais é sua habilidade de aprender a partir de estímulos do ambiente e com isso melhorar seu desempenho. O processo de aprendizagem é realizado através de um processo iterativo de ajustes aplicado aos pesos das conexões entre as unidades de processamento. O treinamento de uma rede neural artificial é realizado por um conjunto de regras bem definidas, conhecido como algoritmo de aprendizagem. Existem vários tipos de algoritmos de aprendizado específicos para determinados modelos de redes neurais artificiais, estes algoritmos diferem entre si principalmente pelo modo como os pesos são modificados.

Considere *Y(t)* como sendo a resposta no domínio do tempo de um sistema dinâmico que pode ser aproximado pela Equação (2.6).

$$Y(t) = f(W,X(t)) \quad \textbf{(2.6)}$$

em que *f(.)* é uma função de mapeamento aproximada entre os parâmetros de entrada e a saída desejada, *X(t)* é um vetor conhecido que representa as entradas do sistema e *W* é um vetor de coeficientes constantes e, a priori, desconhecido. No caso das redes neurais artificiais o vetor *W* representa todas as unidades de processamento da rede e *f(.)* o algoritmo de treinamento da rede.

A dimensão *M* do vetor *W* depende do número de entradas, do número de camadas intermediárias e o do número de unidades de processamento em cada camada. O processo de cálculo do vetor *W* é conhecido na literatura como treinamento ou aprendizado. Este processo é

realizado utilizando informações sobre o problema para modelar o conhecimento da rede neural artificial. Estas informações são estruturadas na forma de entradas (descritores) e suas respectivas saídas para o sistema neural. Quando se dispõe tanto das entradas quanto das saídas desejadas o processo de treinamento é conhecido como *supervisionado*.

O cálculo do vetor de pesos ou matriz de pesos é realizado com o objetivo de obter o menor erro quadrático médio possível sobre os dados do problema. O erro médio quadrático fornece uma idéia do estado atual do sistema em relação ao estado desejado, sendo definido pela Equação (2.7).

$$E = \frac{1}{2N}\sum_{i=1}^{N}(d_i - y_i)^2 \qquad \textbf{(2.7)}$$

onde *N* é a dimensão do conjunto de dados de treinamento, d_i corresponde ao valor desejado da rede para a *i-ésima* amostra do conjunto de treinamento e y_i é valor de saída da rede neural artificial.

A modificação do processamento ou da estrutura de conhecimento de uma rede neural envolve a alteração do seu padrão de interconexão. Em princípio, isto pode ser feito de três maneiras:

(i) Inserção de novas conexões;

(ii) Remoção de conexões existentes na rede; ou

(iii) Modificação dos pesos das conexões já existentes.

Mesmo em uma topologia de rede fixa é possível simular *(i)* e *(ii)* através de *(iii)*, porém com limite superior. Tomando-se uma conexão com peso zero, e modificando-a para um valor positivo ou negativo, equivale a desenvolver ou inserir esta conexão na rede. Da mesma forma, alterar o peso de uma conexão para zero significa desconectá-la da estrutura da rede. Portanto, as regras de aprendizado alteram os pesos das conexões das redes através da experiência.

Sendo assim, a obtenção do vetor de pesos *W* corresponde à solução de um problema de otimização sem restrições num espaço *M-dimensional*.

2.4 Método do Gradiente Descendente (*Backpropagation*)

O método do gradiente descendente (Rumelhart et al, 1986) é um dos métodos mais utilizados no treinamento de redes neurais artificiais. O método baseia-se no calculo, a cada iteração *l*, do gradiente do erro *E* e utiliza esta informação para ajustar o valor do peso *w* da rede, como descrito na Equação (2.8).

$$w_{ji}(l+1) = w_{ji}(l) + \frac{\partial E_{(l)}}{\partial w_{ji}} \tag{2.8}$$

onde,

$$\frac{\partial E}{\partial w_{ji}} = -\eta \rho_j o_i \tag{2.9}$$

sendo,

$$\rho_j = \begin{cases} f_j'(o_j)(d_j - o_j) & \text{para uma unidade de processamento de saída} \\ f_j'(o_j)\sum_k \delta_k w_{jk} & \text{para uma unidade de processamento escondida} \end{cases} \tag{2.10}$$

onde d_j é a saída desejada da rede, η determina o tamanho do passo e o_j a saída obtida.

Durante o treinamento do vetor de pesos, o algoritmo efetua um passo *forward* de forma que um exemplo *X* é submetido à entrada, às ativações e saídas de cada unidade de processamento da rede. Em cada camada, estes valores são calculados e propagados. Após a propagação da saída da rede, o erro δ_j na camada de saída é calculado. Em seguida um passo *backward* é realizado para cada camada. O erro para a unidade de processamento *j* é calculado a partir da ativação e dos erros das *k* unidades de processamento topologicamente seguintes, e os pesos atualizados.

Os novos pesos são calculados utilizando informações dos pesos anteriores, mais uma correção, $w_{ji}(l+1) = w_{ji}(l) + \Delta w_{ji}$. Sendo a correção $\Delta w_{ji} = \eta \delta_j o_i$ designada por *passo*, sendo $0 < \eta < 1$ um parâmetro que determina o tamanho do passo. É interessante verificar que a correção dos pesos associados a uma determinada unidade de processamento utiliza apenas informação local, disponível na própria unidade de processamento ou nas unidades de processamento diretamente conectadas a ela.

2.5 O problema do Mínimo Local

Os métodos baseados em gradiente necessitam de um ponto de partida no espaço de busca para que a busca seja iniciada. Porém, a maioria destes algoritmos somente converge para o ponto de mínimo mais próximo do ponto arbitrado inicialmente (Haykin, 2001). Esta estratégia faz com que exista a possibilidade do método convergir para um mínimo local ao invés de um mínimo global.

Como a escolha inicial de um bom ponto de partida não pode ser realizada de maneira isolada, existem algumas alternativas que objetivam diminuir a ocorrência de mínimos locais. As mais utilizadas são:

(i) Diminuição da taxa de aprendizado η: se a taxa de alteração dos pesos é diminuída progressivamente, o algoritmo do gradiente decrescente pode chegar a uma solução melhor adaptada ao problema (Beale e Jackson, 1990).

(ii) Adição de unidades de processamento nas camadas intermediárias: é possível considerar que aumentando a capacidade de processamento da rede esta possa modelar melhor o problema, diminuindo a ocorrência de mínimos locais.

(iii) Introdução de um termo de momento: termo introduzido para que se possa aumentar/diminuir a taxa de aprendizado sem que ocorram oscilações indesejadas, como descrito na Equação (2.11).

$$w_{ji}(l+1) = w_{ji}(l) - \eta \frac{\partial E_{(l)}(w)}{\partial w_{ji}(l)} + \mu \Delta w_{ji}(l) \qquad \textbf{(2.11)}$$

onde o termo μ, onde $0 < \mu < 1$, é uma constante positiva menor que *1*. A escolha do valor do termo de momento é feito de forma empírica.

Outro problema enfrentado pelos algoritmos baseados em retro-propagação do erro é a quebra de simetria, ou seja, se os pesos começam todos com o mesmo valor e se a solução só pode ser alcançada com pesos diferentes, o sistema poderá não aprender. Como o erro é retro-propagado em proporção ao valor dos pesos das conexões, se todas os pesos forem iguais, os erros serão os mesmos para todas as unidades de processamento (Rumelhart e Weigend, 1990).

2.6 O problema do Sobre-Ajuste

O sobre-ajuste ou *overfitting* ocorre quando para a amostra de treinamento ajustamos um polinômio de grau superior ao que realmente representa o problema em questão. No caso das redes neurais artificiais, se pode dizer que acontece quando a rede memoriza os padrões de treinamento e perde assim a capacidade de generalizar. Desta forma, quando existe sobre-ajuste a rede perde a capacidade de conseguir prever corretamente as saídas relativas a pontos que não se encontrem no conjunto de treinamento. Este problema normalmente ocorre quando a rede possui pesos em excesso, relativamente, aos que seriam necessários para aprender o problema em questão e o treinamento é realizado durante muitas iterações do algoritmo. Nestas condições, a rede começa a memorizar a informação do conjunto de treinamento (inclusive eventual ruído), perdendo assim a capacidade de generalizar.

Alguns trabalhos da literatura sugerem que se existirem, pelo menos, *30* vezes mais exemplos de treinamento que pesos na rede, o treinamento não estará suscetível a sofrer sobre-ajuste (Moody, 1992) e (Weigend, 1994). Quando não é possível obter bases de dados muito grandes, uma das soluções para este problema é separar da amostra de treinamento alguns exemplos que serão utilizados para validar o desempenho do sistema durante seu treinamento. Este conjunto de dados não será utilizado para o cálculo do vetor de pesos *W* e poderá fornecer indícios da generalização do sistema. Assim, quando o erro nos pontos de validação começar a crescer, significa que o sobre-ajuste está ocorrendo e o treinamento deve ser interrompido (Smith, 1996).

Outro problema relacionado ao número de unidades de processamento da rede, porém inversamente proporcional ao *overfitting* é o *underfitting.* O *underfitting* representa a falta de conexões e unidades de processamento que pode tornar a rede incapaz de resolver o problema investigado, devido à quantidade insuficiente de parâmetros ajustáveis (Bartlett, 1997).

Capítulo 3

Sistemas Neuro-difusos

Os modelos neuro-difusos correspondem a uma classe de modelos híbridos conhecidos como Sistemas Híbridos Inteligentes (Abraham e Nath, 2000; Goonatilake e Khebbal, 1995; Medsker e Bailey, 1992). Dentre estes modelos, tem se destacado um modelo que integra as características das redes neurais artificiais e dos sistemas difusos. Este modelo se constitui da combinação dos conceitos de lógica difusa e redes neurais artificiais (Abraham e Nath, 2000). Estas arquiteturas agregam as propriedades de aproximação universal e aprendizagem das redes neurais com a facilidade de manipulação de informação lingüística e dedução dos sistemas difusos, resultando em um modelo eficiente e robusto, com capacidade para tratar as incertezas existentes na informação. Estes sistemas sobrepõem métodos de controle clássico em vários aspectos, tal como simplicidade algorítmica, robustez e a capacidade de trabalhar com imprecisão e incerteza.

3.1 Adaptative Neuro Fuzzy Inference System

O método *Adaptative Neuro Fuzzy Inference System* – ANFIS (Jang, 1993) é possivelmente o primeiro sistema neuro-difuso integrado (Abraham e Nath, 2000a). O modelo propõe um tipo de modelagem difusa que utiliza uma estrutura de redes neurais artificiais em conjunto com estimativas de mínimos quadrados para a implementação da lógica difusa. O modelo implementa regras difusas *Se-Então* do tipo *Takagi-Sugeno* (Takagi e Sugeno, 1985) e pertence a uma classe de sistemas de extração de regras que utilizam estratégia decomposicional. Após a extração, as regras são agregadas de forma a constituírem uma descrição do comportamento global do modelo.

O sistema possui estrutura similar a uma rede neural artificial, com unidades de processamento e conexões, sendo que algumas destas unidades de processamento são adaptativas, ou seja, seus parâmetros internos são alterados de acordo com uma regra de minimização de erro prescrita. O

ANFIS utiliza um algoritmo de aprendizagem híbrido para estimar os parâmetros das regras *Takagi-Sugeno*. O modelo ANFIS ajusta, através do algoritmo *backpropagation*, os parâmetros (formato e localização) das funções de pertinência, as quais devem ser de um tipo único (todas triangulares, ou todas Gaussianas, etc), dependendo da escolha do projetista. O método dos quadrados mínimos recursivo é utilizado para localização dos coeficientes das funções lineares, que formam a parte conseqüente das regras difusas.

Na Figura 3.1 é apresentada a arquitetura do sistema ANFIS para um problema com dois atributos e uma classe. A seguir serão descritas as principais características de cada uma das camadas da arquitetura do modelo.

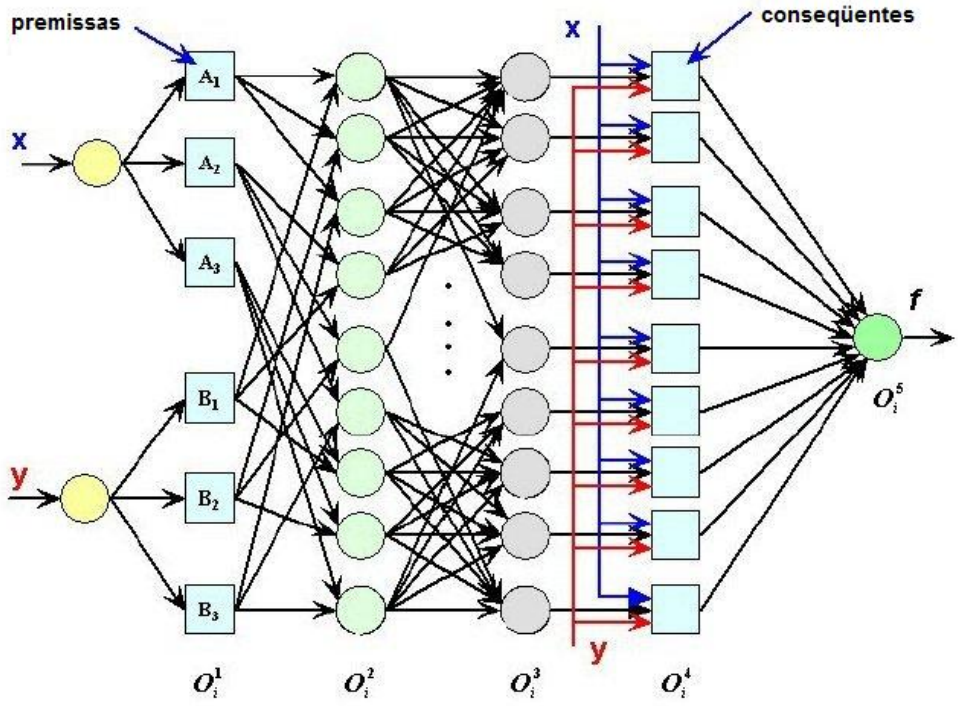

Figura 3.1 - Arquitetura ANFIS (Abraham e Nath, 2000).

Camada 1

As unidades de processamento desta camada são constituídas pelos conjuntos nebulosos associados às variáveis de entrada x_i, onde $i = 1,2,...,N$. Cada unidade de processamento possui uma função de pertinência Φ.

$$o_i^1 = \Phi_{A_i}(X), \quad i = 1,2 \tag{3.1}$$

ou,

$$o_i^1 = \Phi_{B_{i-2}}(Y), \quad i = 3,4,... \tag{3.2}$$

onde A_i e B_i são os valores lingüísticos (alto, baixo, etc.) associados a função de pertinência da unidade de processamento. o_i^l é a função de pertinência de A_i e especifica o grau pelo qual um dado *X* pertence ao conjunto A_i. Diferentes funções de pertinência podem ser utilizadas, como por exemplo, as funções gaussiana ou sino, definidas pelas Equações (3.3) e (3.4).

$$G(X, \mathrm{K}, \phi) = e^{-\frac{1}{2}\left(\frac{X-\mathrm{K}}{\phi}\right)} \tag{3.3}$$

onde *c* representa o centro da função de pertinência e σ a largura da função de pertinência.

$$S(X, \Gamma, \mathrm{Z}, \mathrm{K}) = \frac{1}{1+\left[\left(\frac{X-\mathrm{K}}{\Gamma}\right)^2\right]^{\mathrm{Z}}} \tag{3.4}$$

onde Z e K definem a largura da função de pertinência e Γ o centro. Os parâmetros desta camada se referem aos parâmetros das premissas das regras.

Camada 2

As unidades de processamento desta camada realizam a operação lógica definida pelo conectivo das regras. Cada unidade de processamento de saída representa a abrangência da regra.

$$o_i^2 = w_i = \Phi_{A_i}(X) \times \psi_{B_i}(Y), \quad i = 1,2 \tag{3.5}$$

Em geral, qualquer operador *T-norm* que realiza um *AND* difuso pode ser utilizado como unidade de processamento desta camada.

Camada 3

Cada *i-ésima* unidade de processamento nesta camada calcula a razão da intensidade de disparo da *i-ésima* regra, pela soma das intensidades de disparo de todas as regras.

$$o_i^3 = \overline{w_i} = \frac{w_i}{w_1 + w_2}, \quad i = 1,2 \tag{3.6}$$

As saídas desta camada são denominadas intensidades de disparo normalizadas.

Camada 4

Esta camada é constituída por unidades de processamento adaptativas cujos parâmetros correspondem aos conseqüentes de cada regra.

$$o_i^4 = \overline{w_i f_i} = \overline{w_i}(p_i x + q_i y + r_i) \quad \textbf{(3.7)}$$

onde $\overline{w_i}$ corresponde a saída da camada 3, e $\{p_i, q_i, r_i\}$ representam o conjunto de parâmetros. Os parâmetros desta camada se referem aos parâmetros conseqüentes das regras.

Camada 5

Esta camada possui uma única unidade de processamento, responsável por computar a saída global do modelo (*Y*) como o somatório das saídas parciais de cada regra.

$$o_1^5 = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \quad \textbf{(3.8)}$$

O modelo ANFIS utiliza um misto de *backpropagation* (para aprender os parâmetros das premissas) e mínimos quadrados (para determinar os parâmetros conseqüentes).

Um passo no processo de aprendizagem constitui duas ações: *(i)* os parâmetros de entrada são propagados e parâmetros conseqüentes ótimos são estimados por um processo de mínimos quadrados recursivo, enquanto as premissas são fixadas para o ciclo corrente a partir do conjunto de treinamento; *(ii)* os padrões são propagados novamente e nesta iteração de processamento do algoritmo *backpropagation* é utilizado para modificar os parâmetros que representam as premissas da regra, enquanto os parâmetros que representam os conseqüentes permanecem fixos. Após este cálculo o processo é então iterado.

Embora o ANFIS seja considerado uma referência em qualquer aplicação envolvendo aproximação de funções, este sistema possui algumas deficiências que podem prejudicar seu desempenho. O processo de adaptação (aprendizado) do modelo somente está preocupado com a adaptação dos parâmetros em uma estrutura fixa. Para problemas de larga escala, é difícil determinar os valores ótimos das premissas-consequentes, número de regras, etc. A estrutura da rede ANFIS assegura-se que cada termo lingüístico é representado por somente um conjunto difuso. Porém, o processo de aprendizado do ANFIS não provê meios para aplicar limites que restrinjam os tipos de modificações realizadas nas funções de pertinência.

Rizzi et al. (1999) afirmam que o uso do ANFIS é restringido pela quantidade de parâmetros a serem definidos como: o tipo das funções de pertinência, a granularidade dos universos, que, por sua vez, fixa o total de regras difusas e, por fim, o operador de agregação dos antecedentes (A_i).

Todos estes parâmetros dependem da intervenção do projetista e nenhuma informação é fornecida a fim de sugerir qual a melhor escolha para determinada aplicação. Como alternativa, os autores propõem um modelo de aprendizado baseado em agrupamento (*clustering*) para se extrair um conjunto de regras difusas de modelos *Takagi-Sugeno*. A proposta utiliza uma estrutura similar ao modelo ANFIS, sendo que o aprendizado é automático, sem a necessidade de intervenção do projetista na definição de alguns parâmetros críticos. Uma outra desvantagem do ANFIS está relacionada a interpretabilidade do modelo.

O problema dos mínimos locais é discutido por Chen & Liou (1999), resultante do uso do método de otimização baseado no gradiente, esta deficiência pode prejudicar o aprendizado. Os autores apresentam uma proposta para aumentar a velocidade de convergência do ANFIS, mas o grau de intervenção do projetista permanece similar ao do modelo original proposto por Jang (1993).

3.2 Evolving Fuzzy Neural Network

Toda a estrutura da rede *Evolving Fuzzy Neural Network* (Kasabov, 1998b) é construída sobre um conjunto de regras difusas, sendo possível modificar dinamicamente a topologia da rede. Assim, o modelo pode se adaptar a novos dados de treinamento (inclusive inserindo novas classes), de uma maneira *on-line* e aprender incrementalmente a partir dos exemplos de treinamento. Como a estrutura conexionista da rede pode ser interpretada por regras difusas, todo o conhecimento adquirido pela rede pode ser expresso na forma de regras explicativas difusas do tipo *Se-Então*.

Desta forma, uma estrutura *Evolving Fuzzy Neural Network* é um sistema conexionista que aprende a partir de exemplos, manipula o conhecimento a partir de regras difusas e realiza inserção, extração e agregação destas regras na arquitetura da rede. As redes *Evolving Fuzzy Neural Networks* operam em um modo *on-line* e aprendem incrementalmente, através de características locais dos dados.

A arquitetura *Evolving Fuzzy Neural Network* é denominada evolutiva porque sua topologia cresce com a apresentação de novos dados na fase de aprendizado e regularmente diminui através de poda (*prunning*), ou através da agregação de unidades de processamento de regras.

Dinâmica de funcionamento

A rede *Evolving Fuzzy Neural Network* adapta mecanismos de várias técnicas de aprendizagem (Kasabov, 1999). Porém, em seu funcionamento todas as unidades de processamento são criadas dinamicamente, durante a fase de aprendizado da rede (possivelmente em modo *one-pass learning*). Cada uma das unidades de processamento da rede representa uma ou mais funções de pertinência difusa, as quais podem ser modificadas durante o processo de treinamento.

A arquitetura *Evolving Fuzzy Neural Network* possui cinco camadas, como apresentado na Figura 3.2. A camada de entrada recebe o vetor de entrada (*X*, *Y*), que correspondente aos atributos de entrada e a saída desejada de cada exemplo, respectivamente. Cada atributo de entrada é representado por um grupo de unidades de processamento que realizam a quantificação difusa destas variáveis.

Novas unidades de processamento podem ser inseridas dinamicamente na segunda camada durante a adaptação da rede. Isso ocorre quando, para um dado vetor de entrada, uma variável não pode ser representada por nenhuma das unidades existentes. A correspondência entre uma nova variável e as funções de pertinência existentes são medidas pela similaridade entre elas. Por exemplo, uma nova unidade de processamento difuso ou uma nova unidade de entrada podem ser criadas durante a fase de adaptação da rede se a dissimilaridade entre a variável de entrada e os nodos de processamento ultrapassar um limiar pré-estabelecido pelo projetista.

A terceira camada do modelo contém as unidades de processamento correspondentes às regras difusas, estes nodos representam a associação dos padrões de entrada e saída, e são adaptados através de um aprendizado híbrido: supervisionado e não-supervisionado. Cada unidade de processamento ou regra representa um protótipo da associação entre exemplos de entrada e saída, graficamente representado como uma associação entre hiper-esferas dos espaços de entrada e saída difusos do problema.

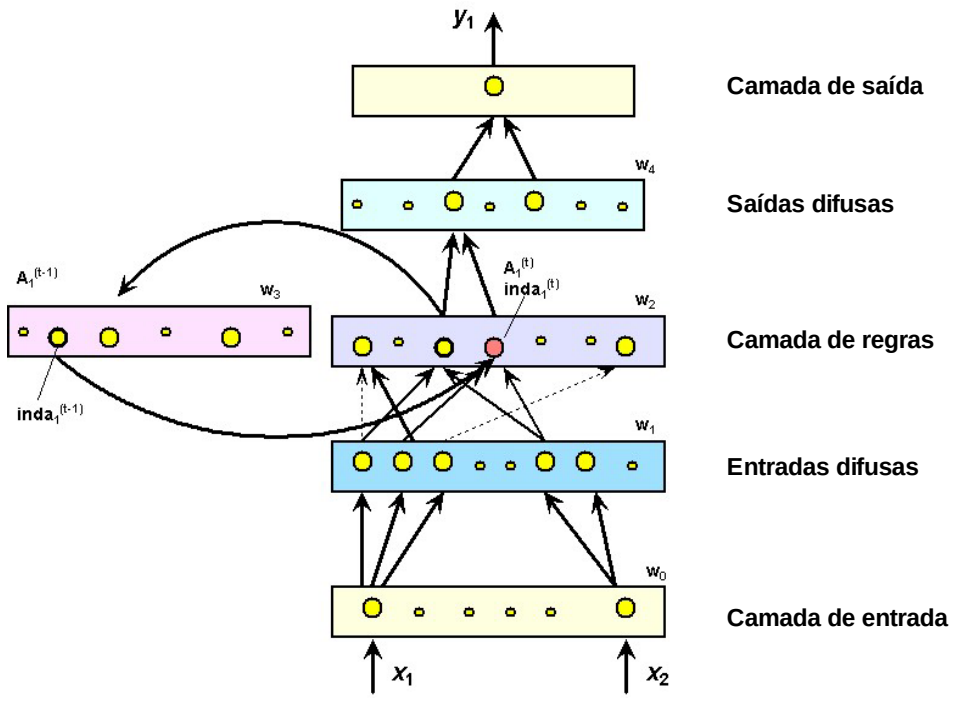

Figura 3.2 - Arquitetura da rede EFuNN - adaptado de Kasabov (2001).

Na Figura 3.3 é apresentado um exemplo de uma unidade de processamento de regras que representa uma associação entre duas hiper-esferas (entrada e saída). Espacialmente, o nodo de regras se move quando um novo par de entradas difusas (X_f, Y_f) é aprendido.

Durante a fase de aprendizado uma camada de memória de curta-duração pode ser utilizada para adaptação da rede, através de conexões de *feedback* provenientes da camada de regras, como é apresentado na Figura 3.2. Como a rede utiliza aprendizado local, nodos de regras podem ser modificados ou excluídos sem prejudicar o aprendizado global do sistema ou o conhecimento previamente adquirido pela rede.

Cada nodo de regras *r* é definido por dois vetores de conexões *W1(r)* e *W2(r)*, em que a conexão *W1(r)* é ajustada com aprendizagem supervisionada, baseado no erro de saída. E a conexão *W2(r)* é ajustada de forma não-supervisionada, baseado na medida de similaridade entre características locais do espaço de entrada e as funções de pertinência existentes. O vetor de conexões *W1(r)* representa as coordenadas do nodo *r* no espaço de entradas difusas, identificando o centro da hiperesfera de entrada, e o vetor de conexões *W2(r)* representa as coordenadas do nodo *r* no espaço de saídas difusas, identificando o centro da hiperesfera de saída.

A quarta camada de unidades de processamento representa a quantificação difusa das variáveis de saída, no espaço de saída do problema. Esta representação é similar à representação difusa das unidades de processamento de entrada. A quinta camada representa os valores reais das variáveis de saída. A saída é um valor real que indica a grau de confiança associado a cada classe.

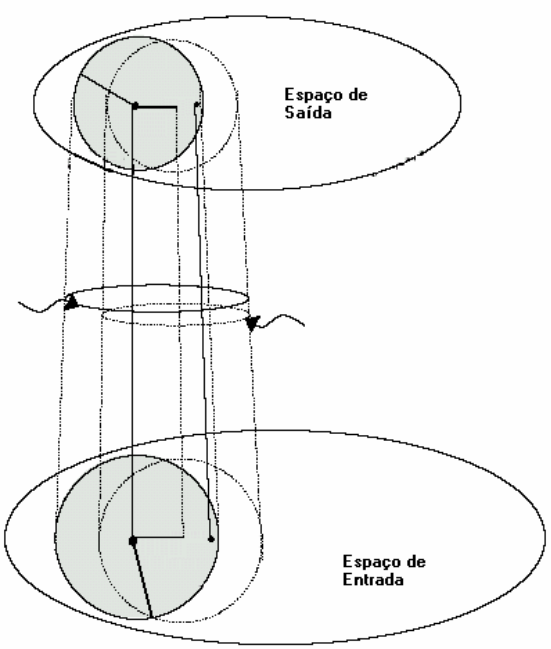

Figura 3.3 - Associação entre hiper-esferas de entrada e saída.

Um determinado exemplo (*X, Y*) é associado a um nodo de regras *r* se *X* estiver dentro do campo receptivo da hiperesfera de entrada do nodo *r* (isso ocorre quando a distância difusa local normalizada entre *W1* e *X* for menor que o raio *R*) e *Y* estiver dentro do campo receptivo da hiperesfera de saída do nodo *r* (isso ocorre quando o erro normalizado de saída *Err* gerado for menor que o erro máximo *E* permitido pelo sistema). A Figura 3.3 ilustra um exemplo difuso associado ao nodo de regra *r*.

O processo de evolução (aprendizagem) da rede pode ser baseado em qualquer uma das seguintes suposições: *(i)* nenhum nodo de regras existe antes do início da fase de aprendizagem e todos são criados durante o processo de evolução da rede; ou *(ii)* existe um conjunto inicial de regras com conhecimento prévio sobre o problema. As conexões entre os exemplos de entrada e saída são definidas e atualizadas a partir deste conjunto de regras, durante a evolução da rede.

O último caso é o mais biologicamente plausível (Kasabov, 1999). Da mesma forma que, por exemplo, uma criança ao nascer possui todos os neurônios e conexões sinápticas em seu cérebro, mas suas estruturas cognitivas e funcionais evoluem de acordo com seu desenvolvimento pessoal, a rede *Evolving Fuzzy Neural Network* adapta seu conhecimento a partir dos nodos de regras já existentes e cria novas conexões ou novos nodos durante o aprendizado. O algoritmo de treinamento apresentado na próxima seção não faz distinção entre estas duas abordagens.

Na camada de regras cada nodo de regras difusas, por exemplo, r_j representa uma associação entre hiper-esferas do espaço de entrada e saída difusas do problema, como mostra a Figura 3.3. Os valores da conexão $W1(r_j)$ representam as coordenadas do centro da esfera no espaço de entrada difuso, e $W2(r_j)$ representa as coordenadas no espaço de saída difuso. O raio da hiper-esfera de entrada da unidade de processamento r_j é definido como $R_j=1-S_j$, onde S_j é o parâmetro de limiar de sensibilidade *sthr*, que define a ativação mínima no nodo de regras r_j obtido por um novo vetor *x* de entrada em um novo exemplo (*X, Y*). O grau desta ativação determina se este exemplo é associado a uma regra existente ou uma nova regra é criada.

O par difuso de entrada-saída (X_f, Y_f) será alocado ao nodo de regras r_j se X_f estiver no campo receptivo de entrada r_j (dentro do raio da hiperesfera de entrada), e se Y_f estiver na hiperesfera receptiva de saída r_j. Isto é garantido através de duas condições: *(i)* se a diferença difusa local entre X_f e $W1(r_j)$ for menor que o raio r_j; e *(ii)* se o erro de saída normalizado, descrito pela Equação (3.9), for menor que o limiar de erro *E*.

$$Err=||Y-Y'||/N_{out} \tag{3.9}$$

onde N_{out} é o número de saídas da rede, *Y* e *Y'* são os vetores da saída desejado e obtido, respectivamente. O parâmetro de limiar de erro *E* ajusta a tolerância de erro do sistema.

Aprendizagem

O algoritmo de treinamento da rede *Evolving Fuzzy Neural Network* é baseado no princípio de que nodos de processamento somente existem se forem realmente necessários. Enquanto cada exemplo de treinamento é apresentado, os valores de ativação das unidades de processamento da camada de regras e o erro sobre as unidades de processamento de saída são examinados. Se a ativação máxima nos nodos de regras estiver abaixo de um limiar estabelecido pelo projetista (limiar de sensibilidade) o nodo de regras é suprimido. Se o erro do nodo de saída estiver acima do valor de limiar também determinado pelo usuário (limiar de erro), um novo nodo de regras é adicionado. E finalmente, se o nodo de saída mais ativado não for o vencedor desejado, uma nova unidade de saída é então adicionada.

Capítulo 4

Otimização Global

A otimização é o processo de encontrar a melhor solução em um conjunto de possíveis soluções para um problema (Panos e Wilde, 2000). O processo de busca normalmente parte de uma solução inicial ou de um conjunto delas, realizando melhoramentos progressivos até chegar a um outro conjunto que contenha uma ou todas as melhores soluções possíveis dentro do espaço de busca. A solução de um problema de otimização pode ser caracterizada como um processo de busca local ou global.

Assim, seja uma função $f : \Re^n \rightarrow \Re$ e considere o problema de minimizar $f(s)$, $s \in S$, onde S representa o conjunto de soluções para o problema. Se $s' \in S$ e se existe uma vizinhança de s' contida em *S*, denotada por $V(s')$, tal que $\forall\ s \in V(s'),\ f(s) \geq f(s'),\ s'$ é chamado de mínimo local de *f(.)*. Da mesma forma, se $s' \in S$ e $f(s) > f(s'),\ \forall s \in V(s'),\ s \neq s'$, então s' é uma solução ótima local de *f(.)*. Assim, um ponto $s' \in S$ é a solução ótima global de *f(.),* ou solução do problema se $\forall\ s \in S,\ f(s) \geq f(s')$.

O processo de busca local objetiva encontrar a melhor solução dentro de um conjunto de soluções em um espaço restrito. Na otimização local, a solução encontrada depende do ponto de início do processo de busca. No processo de busca global o objetivo é encontrará a melhor solução possível, independentemente das condições de início do processo de busca, porém, geralmente, requisita um maior poder computacional (que depende muito do ponto de início da busca). Dependendo do problema, pode ser praticamente impossível encontrar uma solução ótima global, entretanto, uma solução ótima local pode ser bastante eficiente. Em muitos casos, encontrar o ótimo global não é necessário. Encontrar rapidamente uma boa solução (ótimo local) pode ser melhor do que encontrar demoradamente a melhor solução possível.

Quando se tem um universo enumerável de possíveis combinações/permutações de elementos que se pretende minimizar ou maximizar, tem-se uma classe chamada de otimização combinatória. Os algoritmos de otimização combinatória, em geral, se caracterizam pela estratégia de busca empregada, utilização de informações sobre o domínio do problema e complexidade. Os Algoritmos Genéticos (Goldberg, 1989), *Simulated Annealing* (Kirkpatrick, 1983) e *Tabu Search* (Glover e Laguna, 1997) são, em geral, algoritmos iterativos para resolução de problemas de otimização combinatória. Estes três algoritmos são os mais populares, de uma classe de algoritmos de otimização, conhecidos como algoritmos iterativos gerais (Sait, e Youssef, 1999).

4.1 Algoritmos Genéticos

Os Algoritmos Genéticos surgiram da analogia entre otimização e os mecanismos da genética e evolução natural das espécies. Eles são baseados na representação cromossômica das variáveis de otimização, no processo de reprodução e em operadores genéticos como recombinação e mutação (Holland, 1986).

A heurística dos algoritmos genéticos se caracteriza por uma busca paralela no espaço de estados, um contra-ponto às técnicas tradicionais. A busca paralela é alcançada com o uso de um conjunto de possíveis soluções para o problema, conhecido como população. Um indivíduo da população é um vetor de símbolos que, de forma abstrata, representa a solução para o problema. Estes símbolos são chamados genes e cada vetor de genes um cromossomo. Os indivíduos desta população são avaliados por uma medida de adaptabilidade.

A população de cromossomos evolui de uma geração para outra utilizando dois tipos de operadores genéticos: *(i)* mutação e inversão, operador unário que altera a estrutura genética de um cromossomo; e *(ii)* recombinação, operador que gera um novo indivíduo a partir da combinação do material genético de dois cromossomos pais. Os cromossomos pais são escolhidos a partir de técnicas de seleção (Goldberg, 1989).

O Algoritmo 1 apresenta o pseudo-código da técnica Algoritmos Genéticos.

Algoritmo 1 - Pseudo-código da técnica Algoritmos Genéticos

```
1.  Seja i ← 0 tempo inicial
2.  Inicialize a população P_i
3.  Avalie P_i
4.  Para i = 0 até critérios de parada (tempo, aptidão, etc.)
6.        i ← i + 1
7.        Selecionar pares para recombinação P' ← P_i
8.        Recombine P'_i
9.        Mutação P'_i
10.       Avalie P'_i
11.       Atualize P_i ← P'_i
12.  Retorne a melhor solução de P
```

4.2 Simulated Annealing

O algoritmo *Simulated Annealing* é um método de busca que se diferencia dos demais por aceitar movimentos que caracterizam uma degradação em seu desempenho para escapar de mínimos locais (Dowsland, 1993). O processo de busca consiste em uma seqüência de iterações. A cada iteração ocorre a mudança aleatória da solução corrente para criar uma nova solução na vizinhança. A vizinhança é definida com a escolha do mecanismo de geração de novas soluções. Uma vez que a nova solução é criada, a mudança correspondente na função de custo é computada para decidir se a nova solução pode ser aceita como a solução atual. Se o custo da nova solução for menor que o custo da solução atual, a nova solução é aceita, caso contrário, a nova solução pode ser aceita ou não, dependendo do critério *Metropolis* (Metropolis et al, 1953), baseado na probabilidade de Boltzmann. De acordo com este critério, é gerado um número aleatório δ no intervalo [0, 1], a partir de uma distribuição uniforme. Se $\delta \leq e^{\frac{\Delta E}{T}}$ onde ΔE é a variação no custo e *T* é um parâmetro chamado de temperatura, então a nova solução gerada é aceita como solução atual. Caso contrário, a nova solução não é aceita, e o processo continua a partir da solução atual. O Algoritmo 2 apresenta o pseudo-código da técnica *Simulated Annealing*.

Algoritmo 2 - Pseudo-código do algoritmo *Simulated Annealing*

```
1.  Seja s_0 ∈ S solução inicial
2.  T_0 ← (temperatura inicial)
3.  Para i = 0 até I_max − 1
4.        i ← i + 1
5.        Seja s' uma nova solução gerada
6.        Se f(s') < f(s_i)
7.           s_{i+1} ← s'
8.        Senão
9.           s_{i+1} ← s' com probabilidade e^{(f(s')−f(s_k))/T_{i+1}}
10.       Atualize T_i
11.  Retorne s_i
```

onde *S* representa um conjunto de soluções e *f* uma função de custo real. A busca objetiva encontrar o mínimo/máximo global *s*, tal que $f(s) \leq f(s'),\ \forall s' \in S$. A cada iteração são geradas *k* novas soluções a partir da solução atual s_i . O processo termina após $I_{\max}$ iterações retornando a melhor solução s_i corrente.

4.3 Tabu Search

Tabu Search (TS) é um algoritmo de busca iterativa caracterizado pelo uso de uma memória flexível (Glover e Laguna, 1997). Neste método, cada iteração consiste em avaliar uma certa quantidade de novas soluções. A melhor destas soluções (em termos da função de custo) é aceita, mesmo que seu custo seja maior do que o custo da solução atual. Desta forma, o algoritmo escolhe a nova solução que produz a maior melhoria ou a menor deterioração na função de custo, e isto permite que o método escape de mínimos locais. É utilizada uma lista *tabu*, para armazenar uma certa quantidade de soluções recentemente visitadas, as quais são classificadas como proibidas (*tabu*) em iterações posteriores. A lista tabu registra as *n* últimas soluções visitadas. Quando a lista fica cheia, um novo movimento é registrado em substituição ao movimento mais antigo armazenado na lista (Glover, 1986; Glover e Laguna, 1995; Glover e Laguna, 1997).

O método consiste na geração de uma solução x_0 e, em seguida, movimentos aleatórios são gerados na vizinhança desse ponto, com o objetivo de encontrar uma melhor solução para o problema. As soluções geradas são adicionadas à lista *tabu*, que representa a memória do método. O objetivo da lista é manter um histórico, que tem por finalidade impedir a repetição de movimentos recentes com o intuito de evitar a geração de soluções repetidas. A lista *tabu*, por ser limitada, possibilita que, após um certo número de iterações ou um certo tamanho pré-definido, esses caminhos, voltem a serem analisados. Se for delimitado um tamanho para a lista *tabu*, no momento em que a mesma estiver cheia, um certo número de elementos mais antigos será removido para que novos elementos nela sejam incluídos. O tamanho da lista é um fator de grande influência no processo, pois feita à opção por uma lista de tamanho pequeno, o processo ficará reduzido a pequenas regiões de busca. A utilização de uma lista de comprimento maior possibilita que a busca possa ser realizada em grandes áreas (Blum e Roli, 2003).

No caso de se executar a busca na vizinhança de uma solução e for verificado em algum destes movimentos houve uma melhora, o ponto inicial será substituído por este novo ponto encontrado e, a partir deste, o processo continua; ou seja, uma nova série de movimentos aleatórios será realizada na vizinhança desse novo ponto dando continuidade ao processo. O Algoritmo 3 apresenta o pseudo-código da heurística da técnica tabu search.

Algoritmo 3 - Pseudo-código do algoritmo *Tabu Search*

```
1.  Seja s_0 ∈ S solução inicial
2.  Lista ← ∅ (lista tabu)
3.  Atualize s_BSF ← s_0 (melhor solução até o momento)
4.  Atualize Lista ← s_0
5.  Para i = 0 até I_max − 1
6.        i ← i + 1
7.        Seja s' o melhor elemento de um conjunto com k novas soluções
8.           s_{i+1} ← s'
9.           Lista ← Lista - (solução mais antiga) + (solução s')
10.          s ← s'
11.          Se f(s) < f(s_BSF)
12.               s_BSF ← s
13.  Retorne s_BSF
```

neste algoritmo *S* representa um conjunto de soluções, s_{BSF} representa a melhor solução encontrada até o momento e *f* uma função de custo real. A busca objetiva encontrar o mínimo/máximo global *s*, tal que $f(s) \leq f(s'), \ \forall s' \in S$. A cada iteração são geradas *k* novas soluções a partir da solução atual s_i . O processo termina após $I_{\max}$ iterações retornando a melhor solução s_{BSF} (*best so far*) encontrada.

4.4 Trabalhos Relacionados

Nesta seção são apresentados alguns trabalhos relacionados à otimização de redes neurais artificiais utilizando técnicas de otimização global. O objetivo desta seção não é elaborar um resumo da aplicação das técnicas de otimização global existentes, mas sim fornecer uma noção geral de como está sendo desenvolvida a aplicação destes métodos para a otimização de redes neurais artificiais.

Algumas abordagens na literatura já utilizam a integração das heurísticas *Tabu Search*, *Simulated Annealing* e Algoritmos Genéticos para aplicações específicas. Uma forma de integração das três heurísticas foi proposta por Mantawy et al (1999), também por Liu et al (2000) e novamente em Jiang et al (2000). Em Li et al (2002), algoritmos genéticos e simulated annealing foram combinados para otimização de processos em planejamentos de engenharia. Em Du et al (2006) as heurísticas de *Simulated Annealing*, Algoritmos Genéticos e modelos de Markov foram combinados para eliminação de ruídos e segmentação de imagens. Esta combinação também foi proposta por Shieh e Peralta (2005), Hongbo et al (2006) e Yogeswaran et al (2007). Smith et al (2004) propuseram uma versão multiobjetivos de *Simulated Annealing*. A integração das três técnicas também foi proposta por Fox (1992) e em Osman (1993) ela é utilizada para resolver o problema de roteamento de veículos.

No treinamento de redes neurais artificiais as heurísticas *Simulated Annealing*, *Tabu Search* e Algoritmos Genéticos foram utilizadas isoladamente em diversas aplicações. Na maioria das abordagens com o objetivo de minimizar o problema de convergência local do algoritmo *backpropagation*.

Dentre as três abordagens, os algoritmos genéticos são os mais utilizados na literatura no treinamento e otimização de redes neurais artificiais. Em alguns trabalhos são otimizados simultaneamente os parâmetros de configuração e a arquitetura da rede (Harp et al, 1989). Outros autores utilizam os algoritmos genéticos para otimizar simultaneamente os parâmetros e os valores iniciais das conexões entre as unidades de processamento (Köhn, 1996). García-Pedrajas et al (2006) propõem novos operados de recombinação para o treinamento de redes neurais artificiais utilizando algoritmos genéticos.

Lacerda (2003) estudou a aplicação de algoritmos genéticos à otimização de redes neurais do tipo função de base radial. Tsai et al (2006) utilizam um algoritmo híbrido para o ajuste da arquitetura e parâmetros de redes neurais *feedforward*. Palmes et al (2005) utilizam um algoritmo baseado em mutação para treinar os pesos da rede neural artificial *Multi-layer Perceptron*. Gepperth e Roth (2006) utilizaram processos evolucionários multiobjetivo para otimização de arquiteturas *feedforward*.

Trabalhos utilizando *Simulated Annealing* e *Tabu Search*, na otimização de redes neurais artificiais, são mais escassos. Algumas aplicações utilizando *Simulated Annealing* e *backpropagation* foram implementadas para o treinamento de redes *Multi-layer Perceptron* com topologia fixa utilizando duas camadas intermediárias. O problema abordado foi o reconhecimento de respostas de sonar (Porto et al, 1995).

O método de *Simulated Annealing* já foi utilizado com sucesso em diversos problemas de otimização global, como se observa em Corana et al (1987) e Sexton et al (1999). Em Stepniewski e Keane (1997), *Simulated Annealing* e algoritmos genéticos foram utilizados para otimizar arquiteturas de redes *Multi-layer Perceptron*. Experimentos parecidos foram realizados por Sexton et al (1999) nos quais as soluções no espaço de busca foram representadas por vetores de números reais contendo todos os pesos da rede. Os mesmos experimentos foram realizados por Sexton et al (1998) aplicando o algoritmo de tabu search. Metin et al (2000) utilizaram Simulated Annealing para otimizar arquiteturas de redes neurais artificiais aplicadas a sistemas de diagnósticos auxiliados por computador. Em Hamm (2002) Simulated Annealing também é utilizado para otimizar os pesos de redes neurais artificiais.

Karaboga e Kalinli (1997) propuseram um modelo *Tabu Search* baseado em paralelismo para treinar uma rede neural artificial recorrente com o objetivo de identificar sistemas dinâmicos. Cannas et al. (1999) utilizaram *Tabu Search* para otimizar arquiteturas de redes

neurais artificiais para previsão de séries temporais. Martí e El-Fallahi (2004) apresentam uma nova proposta baseada na heurística de tabu search para o ajuste dos pesos de redes neurais *Multi-Layer Perceptron*.

A integração de *Tabu Search*, *Simulated Annealing* e *backpropagation* para otimização da arquitetura e pesos em redes neurais artificiais *Multi-layer Perceptron* foi proposta por Yamazaki et al (2002). O algoritmo proposto por Yamazaki combina características de *Simulated Annealing* e busca *tabu*, além de fazer uso de treinamento híbrido, através da utilização de uma técnica local de treinamento. Neste algoritmo, em cada iteração, é gerado um conjunto de soluções novas a partir da solução atual. Cada uma tem seu custo avaliado, e a melhor solução é escolhida, assim como acontece na busca *tabu*. No entanto, esta solução nem sempre á aceita, diferentemente do que ocorre na busca *tabu*. O critério de aceitação é o mesmo utilizado na técnica de *Simulated Annealing*, porém o método não utiliza lista *tabu*. Durante o processo de otimização, armazena-se apenas a melhor solução encontrada. Esta solução é a solução final retornada pelo método. O pseudo-código do método proposto por Yamazaki é apresentado no Algoritmo 4.

Algoritmo 4 - Pseudo-código do algoritmo Yamazaki

```
1.  s_0 ← solução inicial
2.  T_0 ← temperatura inicial
3.  Atualize s_BSF com a melhor s_0 (melhor solução até o momento)
4.  Para i = 0 até I_max − 1
5.     Se i + 1 não for múltiplo de I_T
6.        T_{i+1} ← T_i
7.     Senão
8.        T_{i+1} ← nova temperatura
9.        Se critério de parada baseado em validação for satisfeito
10.           Interrompa execução da busca global
11.    Gere um conjunto com K soluções novas a partir de s_i
12.    Escolha a melhor solução s' do conjunto
13.    Se f(s') < f(s_i)
14.       s_{i+1} ← s'
15.    Senão
16.       s_{i+1} ← s' com probabilidade e^{(f(s') − f(s_k)) / T_{i+1}}
17.    Atualize s_BSF (se f(s_{i+1} < f(S_BSF)))
18.  Fixe a topologia contida em s_BSF e use o valor das conexões como
valores iniciais para o algoritmo de busca local
```

neste algoritmo *S* representa um conjunto de soluções, s_{BSF} representa a melhor solução encontrada até o momento, *T* representa o valor da variável temperatura, $I_{\max}$ o número de iterações e *f* uma função de custo real. A busca objetiva encontrar o mínimo/máximo global *s*, tal que $f(s) \leq f(s'),\ \forall s' \in S$.

As soluções são mapeadas em vetores que representam a topologia da rede e os pesos relacionados às conexões entre as unidades de processamento. Para geração de novas soluções *bits* de conectividade da solução, que representam a arquitetura da rede corrente são invertidos, de

acordo com uma probabilidade. Esta escolha é realizada da seguinte forma: extrai-se um número aleatório δ a partir de uma distribuição uniforme no intervalo [0, 1]. O valor do novo bit de conectividade c'_i da conexão i é definido pela Equação (4.1).

$$c'_i \begin{cases} \overline{c_i}, & se\ \alpha \leq p \\ c_i, & se\ \alpha > p \end{cases} \quad \textbf{(4.1)}$$

onde $\overline{c_i}$ é o inverso do bit c_i e *p* é a probabilidade de inverter cada bit de conectividade.

Além da inversão dos bits de conectividade, um número aleatório θ retirado de uma distribuição uniforme no intervalo [-1, +1] é adicionado aos pesos das conexões existentes. Desta forma o valor do novo peso w'_i correspondente à conexão *i* é definido pela Equação (4.2).

$$w'_i = w_i + \theta \quad \textbf{(4.2)}$$

Nesta abordagem a topologia máxima da rede deve ser definida pelo projetista da rede neural artificial.

Lins (2005) propôs algumas alterações sobre o modelo de Yamazaki objetivando aumentar o espaço de busca. Assim, durante a modificação do valor dos pesos das conexões entre as unidades de processamento, o intervalo foi estendido na proporção do próprio valor do peso. Ou seja, foi adicionado um número aleatório retirado de uma distribuição uniforme no intervalo [-n, +n] sendo este o valor do peso da respectiva conexão. A este valor pode ainda ser multiplicado por uma determinada taxa ajustável. As outras alterações propostas foram em relação ao esquema de esfriamento e na função de custo. O esquema de esfriamento original foi substituído pelo esquema proposto por Lundy e Mess (1986). O custo de cada solução é medido pela média ponderada entre o erro de classificação do conjunto de treinamento e a porcentagem de conexões utilizadas pela rede.

Em todas estas abordagens, *Simulated Annealing*, *Tabu Search* e Algoritmos Genéticos são normalmente utilizados para ajustar os valores das conexões entre as unidades de processamento em topologias fixas. O objetivo deste trabalho é apresentar uma técnica que combina estas três heurísticas de busca na otimização simultânea da topologia e dos valores das conexões de uma rede *Multi-layer Perceptron*, resultando em arquiteturas com baixa complexidade e alta performance de generalização.

Na Tabela 4.1 é apresentado um resumo das principais diferenças entre as principais técnicas existentes na literatura e o método proposto nesta tese (GaTSA) para otimização da arquitetura e pesos em redes neurais artificiais *Multi-layer Perceptron*.

Tabela 4.1 – Diferenças entre os algoritmos investigados

Características	Fox (1992)	Stepniewski e Keane (1997)	Yamazaki et al (2002)	Lins (2005)	Método proposto GaTSA
Heurísticas	Algoritmos Genéticos, *Simulated Annealing*, *Tabu Search* e *Backpropagation*	Algoritmos Genéticos, *Simulated Annealing*	*Simulated Annealing*, *Tabu Search* e *Backpropagation*	*Simulated Annealing*, *Tabu Search* e *Backpropagation*	Algoritmos Genéticos, *Simulated Annealing*, *Tabu Search* e *Backpropagation*
Inicialização	Solução isolada	Solução isolada	Solução isolada	Solução isolada	População de soluções
Vizinhança	Gera a vizinhança a partir de uma solução	Gera a vizinhança a partir de uma solução	Gera a vizinhança a partir de uma solução	Gera a vizinhança a partir de uma solução	Evolui a população com micro-evoluções genéticas
Evolução	Operadores genéticos	Operadores genéticos	Operadores de inversão de bits	Operadores de inversão de bits modificado em relação a Yamazaki et al (2002)	Operadores genéticos
Avaliação	Encontra a melhor solução da população	Encontra a melhor solução da população	Encontra a melhor solução da vizinhança	Encontra a melhor solução da vizinhança	Encontra a melhor solução da população – foram testadas 5 diferentes funções de avaliação
Visitação	Mantém uma lista das soluções recentemente visitadas	-	Visita soluções aleatoriamente	Visita soluções aleatoriamente	Mantém uma lista das soluções recentemente visitadas
Ordem da Busca	-	Projetista define topologia máxima e algoritmo realiza a poda das conexões	Projetista define topologia máxima e algoritmo realiza a poda das conexões	Projetista define topologia máxima e algoritmo realiza a poda das conexões	Algoritmo inicia com topologia mínima e evolui de forma construtiva e baseada na

					poda de conexões sem relevância
Aceitação	Baseado na função de avaliação dos Algoritmos Genéticos	Aceita soluções baseado no critério *Metrópolis*	Aceita soluções baseado no critério *Metrópolis*	Aceita soluções baseado no critério *Metrópolis*, utilizando esquema de esfriamento proposto por Lundy e Mess (1986)	Aceita soluções baseado no critério *Metrópolis*
Critério de Parada	Número de iterações	Número de iterações	Critério GL_5 do *Proben1* e número de iterações	Critério GL_5 do *Proben1* e número de iterações	Critério GL_5 do *Proben1* e número de iterações
Ajuste Local	Não possui	Não possui	Realiza ajuste local utilizando *Backpropagation*	Realiza ajuste local utilizando *Backpropagation*	Realiza ajuste local utilizando *Backpropagation*

Capítulo 5

Seleção de Atributos

A seleção de atributos tornou-se um importante tópico de pesquisa com o surgimento de bases de dados contendo muitos atributos. Geralmente o desempenho dos algoritmos de aprendizado de máquina é negativamente influenciado pelo aumento da razão entre o número de atributos e o número de instâncias. Desta forma, a identificação dos atributos realmente relevantes para a tarefa de classificação pode influenciar muito no desempenho do classificador.

A tarefa de seleção de atributos pode ser formalizada como segue: Seja $X' \subset X$ um subconjunto de atributos de X, e $f(X')$ os valores associados aos vetores correspondentes a X'. A seleção de atributos consiste em encontrar o subconjunto mínimo de atributos X' tal que $P(C \mid Y = f(X')) \approx P(C \mid Y = f(X))$, onde $P(C \mid Y = f(X'))$ e $P(C \mid Y = f(X))$ são as distribuições de probabilidades das N_{C_i} possíveis classes, dados os valores dos atributos de X' e X, respectivamente. O subconjunto mínimo X' é denominado subconjunto ótimo de atributos (Yu e Liu, 2004).

A seleção de atributos relevantes pode ser realizada de formas distintas. Porém, antes de se decidir "como" encontrar os atributos que melhor descrevem uma base de dados é necessário definir o que pode ser considerado um bom atributo ou descritor.

Existem na literatura muitas definições em torno de como considerar a importância de um atributo para uma base de dados. Podem ser citados os trabalhos de (Gennari et al., 1989), (Almuallim e Dietterich, 1992), (Kira e Rendell, 1992), (John et al., 1994) e (Koller e Sahami, 1996). Em geral, podemos considerar um atributo importante para uma base de dados se quando removido a medida de importância considerada em relação aos demais atributos é deteriorada (Liu e Motoda, 1998).

Algumas métricas podem ser utilizadas para definir a importância de um atributo, dentre as técnicas existentes na literatura, podem ser destacadas (Lee, 2005):

- *Métricas baseadas em Informação:* determinam o ganho de informação a partir de um atributo. Pode ser definido como sendo a incerteza *a priori* e *a posteriori* ao considerar-se determinado atributo da base de dados
- *Métricas baseadas em Distância:* também denominadas de medidas de separabilidade, divergência ou discriminação. Definem a diferenciação entre as classes baseado em algum tipo de distancia, como a distância de *Mahalanobis*, por exemplo.
- *Métricas baseadas em Dependência:* também conhecidas como métricas de correlações ou associação. Pode ser considerada a correlação entre os atributos ou ainda como a dependência entre os atributos, o que pode caracterizar redundância entre os atributos.
- *Métricas baseadas em Consistência:* fortemente dependentes do conjunto de treinamento e baseadas em hipóteses consistentes que possam ser definidas a partir do menor número possível de atributos. Buscam manter a consistência entre os atributos – a inconsistência entre descritores pode ser definida como a existência de atributos pertencentes a classes distintas com valores semelhantes.
- *Métricas baseadas em Precisão:* algoritmos de aprendizado são utilizados para selecionar diferentes subconjuntos de atributos, o subconjunto de atributos que maior precisão proporcionar ao modelo gerado será selecionado como subconjunto ideal de atributos. Vale ressaltar que este tipo de medida de precisão é dependente do algoritmo de aprendizado considerado, pois os subconjuntos de atributos serão importantes em relação à precisão do classificador utilizado.

A tarefa de selecionar atributos pode ainda ser examinada a partir de diferentes perspectivas, dentre as quais podem ser destacadas (Liu e Motoda, 1998):

1. Como encontrar os “melhores” atributos de uma base de dados?
2. Como determinar quais são os melhores atributos e qual o critério deve ser utilizado para avaliação desses atributos?
3. Como os subconjuntos de atributos devem ser gerados para selecionar os melhores atributos, adicionando ou excluindo um atributo do subconjunto existente ou alterando o subconjunto aleatoriamente – busca com geração paralela ou seqüencial?

Existem diferentes formas de responder cada uma destas questões. Em relação à primeira questão, o problema de seleção de atributos pode ser abordado de formas distintas: ordenando-se os atributos segundo uma medida de importância ou utilizando um método de busca para encontrar o melhor subconjunto de atributos. A avaliação de quais são os melhores atributos necessariamente passa pela utilização de algum tipo de métrica, como as destacadas anteriormente. A resposta à terceira questão, também relacionada com a primeira, exige que sejam feitas considerações sobre

como realizar uma busca aos melhores subconjuntos de atributos. Estas considerações serão realizadas nos próximos parágrafos.

Sob o ponto de vista de aprendizado de máquina a seleção de atributos pode ser interessante por vários aspectos (Michie et al, 1994): melhora na precisão do classificador, pois atributos irrelevantes ou redundantes normalmente prejudicam o desempenho do classificador; velocidade na execução do classificador, normalmente pode-se dizer que a complexidade de uma base é reduzida com a diminuição no número de atributos – também é fato que a redução no volume de dados reduz a carga computacional exigida na execução do algoritmo – estas características também podem influenciar no tempo de aprendizado do classificador; a seleção de atributos pode também ajudar na compreensibilidade da base de dados e conseqüente simplificação do conceito induzido; e este processo é especialmente importante em bases de dados muito grandes em que se desconhece a relevância de cada um dos atributos.

A grande maioria dos métodos de seleção de atributos, utilizados em conjunto com aprendizado de máquina, são baseados em busca. Os conceitos e pressupostos destas técnicas são os mesmos de técnicas clássicas de busca e precisam ser corretamente definidos para se obter sucesso nesta tarefa, são eles:

1. Ponto de partida ou direção da busca;
2. Estratégia de busca;
3. Critérios para avaliação das soluções geradas; e
4. Critério de parada da busca.

A geração dos subconjuntos de atributos durante o processo de busca pelos melhores atributos é determinada por dois pontos chave. Primeiro é necessário definir o ponto (ou pontos) inicial da busca que passam a influenciar sua direção. A busca pode iniciar com um conjunto vazio e sucessivamente adicionar características (*Para Frente*), ou iniciar com um conjunto completo e remover sucessivamente as características (*Para Trás*), ou ainda iniciar com ambos e adicionar e remover atributos simultaneamente (*Bidirecional*). A busca também pode ser iniciada com um subconjunto selecionado aleatoriamente (*Aleatória*), para evitar a armadilha de um ótimo local (Santoro, 2005). A Figura 5.1 apresenta um esquema das cinco diferentes estratégias de busca em um espaço de busca definido com três atributos. Nesta figura os círculos brancos representam os atributos não selecionados e os círculos sombreados os atributos selecionados.

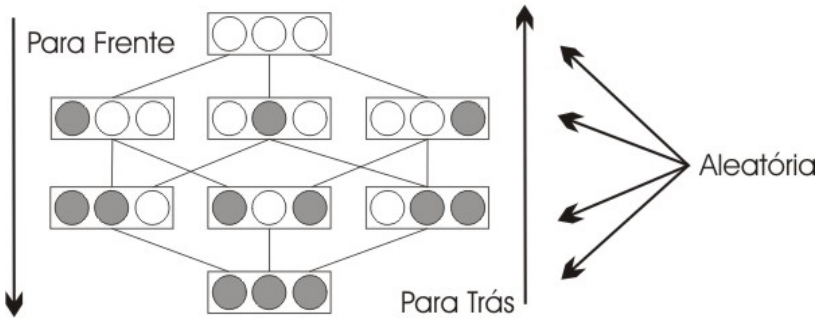

Figura 5.1 – Estratégias para direção de busca Para Frente, Para Trás e Aleatória

O segundo passo é definir a estratégia de busca utilizada. Diferentes estratégias de busca tem sido apresentadas na literatura para esta tarefa, como a busca exaustiva, a busca seqüencial e a busca aleatória (Liu e Yu, 2005). A busca exaustiva garante encontrar um resultado ótimo segundo o critério de avaliação utilizado. Este tipo de busca faz todas as combinações possíveis antes de retornar o subconjunto de atributos selecionado (Liu e Motoda 1998). A busca exaustiva é completa. No entanto, uma busca não precisa ser exaustiva para garantir que seja completa. Diferentes funções heurísticas podem ser utilizadas para reduzir o espaço de busca sem colocar em risco as chances de encontrar um resultado ótimo (Liu e Yu, 2005).

A busca seqüencial, por sua vez, tem apresentado resultados eficientes em problemas de seleção de atributos, mas possui a desvantagem de não considerar a interação entre atributos (Liu e Yu, 2005). A seleção seqüencial para frente inicia a busca pelo melhor subconjunto de atributos com um conjunto vazio de atributos. A seleção seqüencial para trás, diferente da anterior, inicia a busca por um subconjunto de atributos ótimo com uma solução representando todos os atributos, e a cada iteração um atributo é removido da solução atual, até que não se consiga melhorar a qualidade da solução encontrada.

Diferente das estratégias anteriores, a busca aleatória é iniciada com um subconjunto selecionado aleatoriamente e pode realizar o restante da busca de duas formas distintas. Uma delas é seguir a busca seqüencial, inserindo aleatoriedade nas abordagens seqüenciais clássicas. E a outra é gerar o próximo subconjunto de maneira também aleatória. A utilização da aleatoriedade pode ajudar a busca a escapar de mínimos locais, porém o custo computacional pode ser considerável sem a garantia de obter uma solução ótima. Surpreendentemente, para uma grande quantidade de problemas, a utilização de algoritmos randômicos se constitui na forma mais simples e/ou mais rápida de implementação (Arantes et al, 2005).

Em relação ao tipo de avaliação das soluções que pretendem realizar, algoritmos de busca podem implementar três abordagens distintas: *wrapper*, em uma etapa de pré-processamento, o algoritmo de indução é utilizado na seleção do subconjunto de atributos; *embedded*, a seleção do subconjunto de atributos é embutida no próprio algoritmo de aprendizado; e filtro, um subconjunto

de atributos é selecionado numa etapa de pré-processamento, independentemente do classificador a ser utilizado.

5.1 Abordagem Filtro

Na abordagem filtro, a idéia é selecionar atributos filtrando, segundo algum critério, os atributos irrelevantes. Esta filtragem, que corresponde a um pré-processamento, considera características gerais do conjunto de dados para selecionar alguns atributos e excluir outros. Na Figura 5.2 é apresentado um esquema de funcionamento da abordagem filtro.

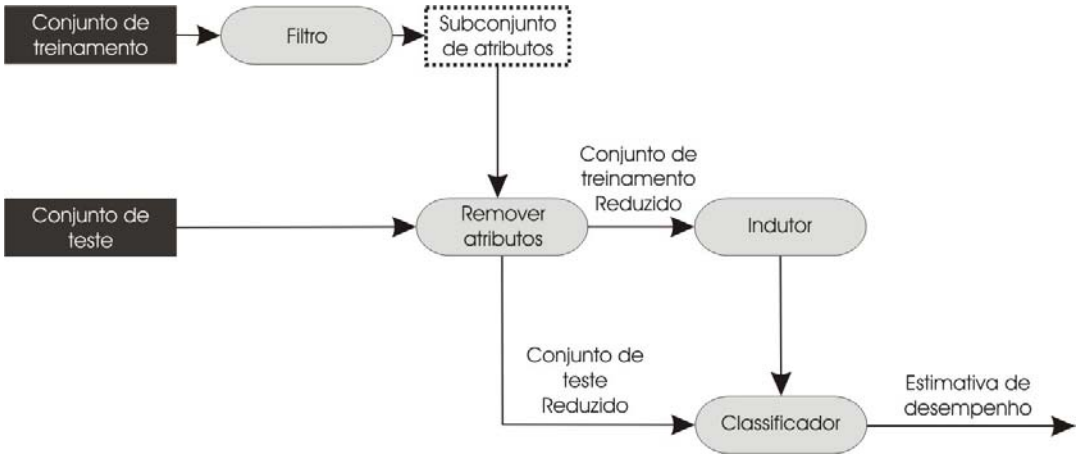

Figura 5.2 - Abordagem filtro

5.2 Abordagem *Embedded*

Nesta abordagem a seleção de atributos é realizada dentro do próprio algoritmo de classificação. Uma das vantagens desta abordagem é fazer melhor uso dos dados disponíveis (desta forma não é necessário dividir os dados de treinamento) e ser mais rápida, por não realizarem múltiplos treinamentos. Na Figura 5.3 é apresentado um esquema de funcionamento da abordagem *embedded*.

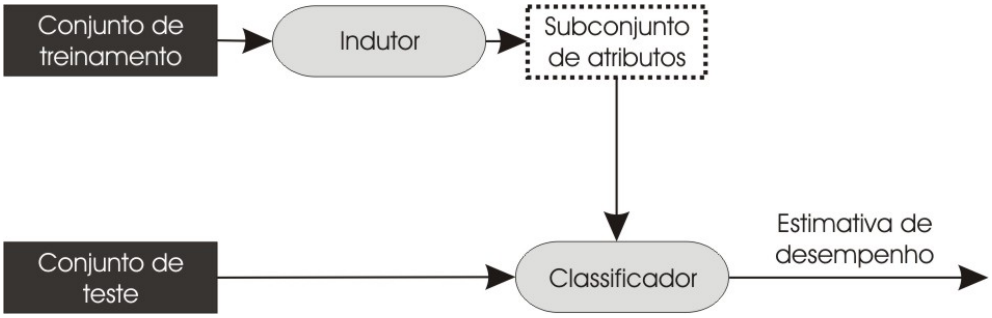

Figura 5.3 - Abordagem *embedded*

5.3 Abordagem Wrapper

Como destacado anteriormente, na abordagem *wrapper*, a seleção dos atributos acontece numa etapa de pré-processamento. Os subconjuntos de atributos são avaliados utilizando-se uma estimativa de desempenho fornecida por algum algoritmo de classificação a partir de uma base de dados de treinamento. O classificador utilizado na avaliação dos subconjuntos pode ser visto como uma "caixa-preta" (o projetista não precisa conhecer o algoritmo de classificação). O critério para avaliar a relevância dos atributos é a precisão do classificador. Na Figura 5.4 é apresentado um esquema do funcionamento da abordagem *wrapper*.

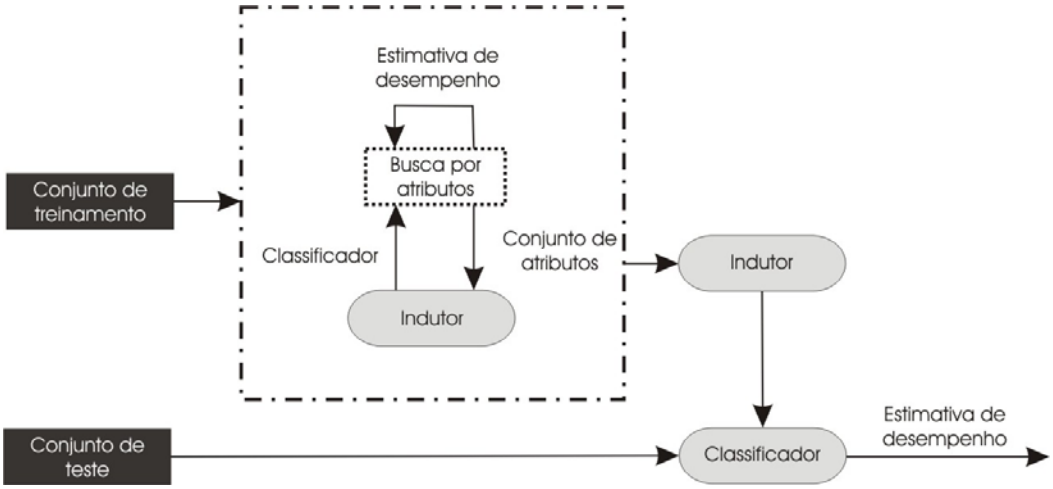

Figura 5.4 - Abordagem *wrapper*

Independente da abordagem de seleção de atributos utilizada, o momento de parar a busca também esta relacionado à satisfação de critérios estabelecidos pelo projetista, dentre eles podem ser destacados (Lee, 2005): parar de remover ou adicionar atributos quando não é encontrada nenhuma evolução na qualidade da solução; evoluir a busca enquanto não houver degradação na qualidade da solução; gerar novas soluções até percorrer todo o espaço de busca; gerar novas soluções até que as classes estejam perfeitamente separadas; e ordenar os atributos segundo um

ranking e determinar um ponto de parada (assim que determinado número de atributos seja selecionado).

Neste trabalho, para avaliar o desempenho do método proposto na seleção de atributos, métodos clássicos de busca serão utilizados em uma abordagem *wrapper* para seleção de atributos em diferentes bases de dados. As próximas seções descrevem brevemente cada uma destas técnicas, nomeadas, *Hill-Climbing* (Vafaie e De Jong, 1993), *Random Bit Climber* (Davis, 1991), *Best-First* (Aha e Banker, 1995), *Beam Search* (Aha e Banker, 1995) e *Las Vegas* (Liu e Setiono, 1996).

Estes métodos foram escolhidos por serem métodos simples e apresentarem características similares às do método proposto. Podem ser destacadas as características aleatórias do método *Las Vegas*, a utilização de listas com as soluções visitadas dos métodos *Best-First* e *Beam Search*, as inversões pontuais nas soluções do método *Random Bit Climber* e as peculiaridades clássicas dos algoritmos baseados em busca presentes no método *Hill-Climbing*.

Hill-Climbing

Uma busca *Hill-Climbing* (Pearl, 1984) é caracterizada pela estratégia de controle baseada numa otimização local, onde o conhecimento é usado de forma "gulosa", procurando obter o melhor benefício possível a cada passo, para escolher o que parece ser o melhor caminho em direção ao nó objetivo. Este tipo de busca utiliza o conhecimento local para construir a solução global. Esta alternativa pode ser interessante quando se dispõem de informações que permitem guiar a busca para próximo do ótimo global.

A estratégia de busca é iniciada com apenas um estado corrente, que é expandido para a geração de novos estados. Os novos estados são gerados com a adição ou remoção de cada um dos atributos da base de dados. Dentre os estados resultantes deste processo, aquele que obteve maior precisão é definido como novo estado corrente. O Algoritmo 5 apresenta o pseudo-código da técnica *Hill-Climibing*.

Algoritmo 5 - Pseudo-código do algoritmo *Hill-Climbing*

```
1.  s_0 ← solução inicial
2.  Atualize s_BSF ← s_0 (melhor solução até o momento)
3.  Enquanto expandir = 1
4.     expandir ← 0
5.     filhos ← expande s_0 com a adição/remoção de atributos
6.     Escolha a melhor solução s' do conjunto filhos
7.     Se f(s') < f(s_BSF)
8.        s_BSF ← s'
9.        expandir ← 1
10.  Retorne s_BSF
```

neste algoritmo *S* representa um conjunto de soluções, s_{BSF} representa a melhor solução encontrada até o momento e *f* uma função de custo real.

Random Bit Climber

O algoritmo *Random Bit Climber* foi introduzido por Davis (Davis, 1991). Neste algoritmo, a partir de uma representação binária da solução, a busca é caracterizada pela inversão dos bits que representam a solução, sendo que a ordem de remoção ou inversão dos bits é definida aleatoriamente. Durante a busca, a melhor solução encontrada é mantida como estado corrente e novas inversões de bits são realizadas sobre este novo estado. O algoritmo *Random Bit Climber* tem a capacidade de percorrer o espaço de busca rapidamente, pois qualquer melhoria encontrada durante a busca é aceita, sem avaliar o espaço de busca local. O Algoritmo 6 apresenta o pseudo-código da técnica *Random Bit Climber*.

Algoritmo 6 - Pseudo-código do algoritmo *Random Bit Climber*

```
1.  s_0 ← solução inicial
2.  Atualize s_BSF ← s_0 (melhor solução até o momento)
3.  Enquanto expandir = 1
4.     ordem ← vetor com a ordem de inversão dos atributos
5.     expandir ← 0
6.     Para i = 1 até o número de atributos do conjunto original
7.        filho ← inverte o bit ordem(i) do subconjunto s_BSF
8.        Se f(filho) < f(s_BSF)
9.           s_BSF ← filho
10.          expandir ← 1
11.  Retorne s_BSF
```

neste algoritmo *S* representa um conjunto de soluções, s_{BSF} representa a melhor solução encontrada até o momento e *f* uma função de custo real.

Best-First

O método de busca *Best-First* (Aha e Bankert, 1995) pode ser definido como um refinamento da técnica de busca *Breadth-first* (Skiena, 1990). A busca *Best-First* (Aha e Bankert, 1995) parte de um nó inicial e mantêm um conjunto de caminhos (soluções) candidatos. Ao contrário da busca *Breadth-First* que sempre escolhe para expansão o caminho candidato mais curto, a busca *Best-First* expande caminhos alternativos, mas que apresentam boa avaliação. Para isso utiliza listas que armazenam as soluções já expandidas (para evitar visitar as soluções novamente) e as soluções já geradas, mas não expandidas ainda. A expansão das soluções candidatas é realizada pela

adição/remoção de atributos – dependendo da estratégia de busca utilizada – a partir da melhor solução presente na lista de soluções não expandidas.

Como o processo de busca pode percorrer todo o espaço de busca, caso nenhuma restrição seja imposta, é comum limitar o processo a um número de expansões seguidas sem melhoras na avaliação da solução. O Algoritmo 7 apresenta o pseudo-código da técnica *Best-First*.

Beam Search

O algoritmo *Beam Search* (Aha e Banker, 1995) é uma otimização da busca *Best-First*. Assim como a busca *Best-First*, ele direciona a busca de forma a expandir somente as soluções mais promissoras, sem visitá-las em sucessivas expansões. No entanto, apenas os *m* nós mais promissores são examinados em cada profundidade, onde *m* (tamanho do *beam*) é um número definido pelo projetista. Com esta limitação, a busca nunca será completa, pois alguns estados nunca serão expandidos. O Algoritmo 7 apresenta o pseudo-código da técnica *Beam Search*. A única diferença entre os algoritmos *Best-First* e *Beam Search* é a utilização de um limite no número de soluções expandidas pelo algoritmo *Beam Search*.

Algoritmo 7 - Pseudo-código dos algoritmos *Best-First* e *Beam Search*

```
1.  s_0 ← solução inicial
2.  open ← s_0
3.  Atualize s_BSF ← s_0 (melhor solução até o momento)
4.  closed ← ∅
5.  expansões ← 0
6.  Enquanto expansões menor que o número de expansões k sem melhoria
7.     s ← melhor conjunto do vetor open
8.     Remover s do vetor open
9.     Adicionar s no vetor closed
10.    Se f(s) < f(s_BSF)
11.        s_BSF ← s
12.        expansões ← 0
13.    expansões ← expansões+1
14.    filhos ← expansão da solução s
15.    Escolha a melhor solução s' do conjunto filhos
16.    Adiciona s' no vetor open
17.    Limita o tamanho do vetor open pelo tamanho do beam - Beam Search
18.  Retorne s_BSF
```

neste algoritmo *S* representa um conjunto de soluções, s_{BSF} representa a melhor solução encontrada até o momento, *open* e *closed* representam vetores de armazenamento temporários e *f* uma função de custo real.

Las Vegas

O método de *Las Vegas* (Brassard e Bratley, 1996) é um algoritmo randômico. Seu funcionamento é extremamente simples, num processo onde são geradas sucessivas soluções

aleatórias, sem qualquer heurística de busca. Durante o processo de busca, cada solução gerada é avaliada e a melhor solução é armazenada. O processo de geração aleatória é realizado até que *k* soluções em seqüência sejam geradas sem melhora no desempenho da solução. O Algoritmo 8 apresenta o pseudo-código da técnica *Las Vegas*.

Algoritmo 8 - Pseudo-código do algoritmo Las Vegas

```
1.  k ← 0
2.  Atualize s_BSF ← ∅ (melhor solução até o momento)
3.  Enquanto k for menor que o número K de iterações
4.     s ← conjunto gerado aleatoriamente
5.     Se f(s) < f(s_BSF) OU..
..        f(s) = f(s_BSF) E número de atributos em s é menor
6.        s_BSF ← s
7.        k ← 0
8.     Senão
9.        k ← k + 1
10.  Retorne s_BSF
```

neste algoritmo *S* representa um conjunto de soluções, s_{BSF} representa a melhor solução encontrada até o momento e *f* uma função de custo real.

Capítulo 6

Método Proposto GaTSa

Como destacado nos capítulos anteriores, os Algoritmos Genéticos (Goldberg, 1989), *Simulated Annealing* (Kirkpatrick, 1983) e *Tabu Search* (Glover e Laguna, 1997) são os mais populares, de uma classe de algoritmos de otimização, conhecidos como algoritmos iterativos gerais. Apesar de possuírem características muito distintas, a heurística de busca dos três algoritmos possui várias similaridades (Sait, e Youssef, 1999). Dentre as principais características de cada método, pode ser destacada a habilidade do método *Simulated Annealing* de escapar de mínimos locais, pela escolha entre aceitar ou não uma nova solução que deteriore o custo da solução atual. O método *Tabu Search*, em contraste, avalia um conjunto de soluções a cada iteração (em vez de uma única solução, como em *Simulated Annealing*). Este fato torna o método *Tabu Search* mais rápido, de modo a necessitar, em geral, menos iterações para convergir. Além da velocidade na convergência, o método *Tabu Search* possui um componente de memória de curto prazo, de forma a evitar visitas à mesma solução mais de uma vez. O processo de busca dos Algoritmos Genéticos, por sua vez, envolve uma seqüência de iterações, na qual um conjunto de soluções passa pelos processos de seleção e reprodução. Este processo, muito mais elaborado que nos algoritmos anteriores pode resultar em soluções com melhor qualidade.

As características destas três técnicas, de certa forma complementares, motivaram a proposta de um método de otimização (GaTSa) que combina as principais potencialidades de *Simulated Annealing*, Algoritmos Genéticos e *Tabu Search*, procurando evitar suas limitações. Este método possui duas fases de otimização: otimização global e local. A otimização global é realizada pela integração das heurísticas de *Simulated Annealing*, *Tabu Search* e Algoritmos Genéticos. A fase de otimização local é responsável pelo ajuste fino da solução proposta ao problema abordado, sendo realizada por um algoritmo de otimização local. A busca é realizada de forma construtiva, de forma que as soluções iniciais possuem a topologia mínima de uma rede neural artificial válida. Durante a

otimização conexões e unidades de processamento são agregadas e removidas da arquitetura da rede objetivando encontrar topologias pequenas e com melhor desempenho.

6.1 Descrição do Algoritmo GaTSa

As heurísticas de busca dos algoritmos *Tabu Search*, *Simulated Annealing* e Algoritmos Genéticos incorporam conhecimento específico sobre o domínio do problema para realizar o processo de otimização. Estes algoritmos também aceitam vários elementos não determinísticos que auxiliam a busca a escapar de mínimos locais. Contam com uma função de custo apropriada para cada problema, o que os torna muito abrangentes em relação aos problemas a que podem ser aplicados. A principal diferença entre eles é como e onde o conhecimento específico do domínio é utilizado. Por exemplo, em *Simulated Annealing* tal conhecimento é incluso na função de custo. Soluções evoluídas de perturbações aleatórias são selecionadas aleatoriamente e são aceitas ou rejeitadas de acordo com uma probabilidade.

No caso dos Algoritmos Genéticos, o conhecimento do domínio do problema é explorado na maioria das fases do algoritmo. O conhecimento está incluso na adaptabilidade das soluções individuais ao problema, na seleção dos indivíduos para reprodução, nos operadores genéticos, tal como na heurística para geração de novas populações. *Tabu Search*, diferente das heurísticas anteriores, possui um componente de memória explícito. A cada iteração, a vizinhança da solução corrente é parcialmente explorada e o movimento é feito em direção à melhor solução da vizinhança que não seja *tabu*. A geração da vizinhança, tal como o tamanho da lista *tabu* e seu conteúdo armazenam conhecimento sobre o domínio do problema. A direção da busca também é influenciada pela estrutura da memória.

A integração proposta utiliza maior quantidade de informação sobre o domínio do problema e utiliza estas informações em praticamente todas as fases da busca. Isto se deve a integração das principais potencialidades das três heurísticas investigadas. Além disto, a técnica propõe dois estágios de busca bem definidos, uma fase de busca global: na qual faz uso da capacidade dos Algoritmos Genéticos em gerar novas soluções, do esquema de resfriamento e função de custo da técnica *Simulated Annealing*, e das características de memória da técnica *Tabu Search*; e uma fase de busca local, na qual faz uso de características como gradiente descendente para um ajuste mais preciso da solução obtida.

O pseudo-código do método proposto é apresentado no Algoritmo 9. O objetivo proposto pode ser assim formalizado: seja *S* um conjunto de soluções e *f* uma função de custo real, o algoritmo procura o mínimo global *s*, tal que $f(s) \leq f(s'), \forall\ s' \in S$.

Algoritmo 9 - Pseudo-código do Algoritmo Proposto

```
1.  P_0 ← população inicial com K soluções s_k e tamanho s_z
2.  Lista ← ∅ (lista tabu)
3.  T_0 ← temperatura inicial
4.  Atualize s_BSF com a melhor s_k de P_0 (melhor solução até o momento)
5.  Atualize Lista ← s_BSF
6.  Para i = 0 até I_max − 1
7.     Se i + 1 não for múltiplo de I_T
8.        T_{i+1} ← T_i
9.     Senão
11.       T_{i+1} ← nova temperatura
12.       Se critério de parada baseado em validação for satisfeito
13.          Interrompa execução da busca global
14.       Aumente o tamanho dos cromossomos de P_i inserindo z nodos
15.       P_i ← P_z
16.    Para j = 0 até g_n
17.       Gere uma nova população P' a partir de P_i
18.       P_i ← P'
19.    Escolha a melhor solução s_k de P_i que não seja solução tabu
20.    Se f(s_k) < f(s_i)
21.       s_{k+1} ← s_k
22.       Lista ← Lista - (solução mais antiga) + (solução s_k)
23.    Senão
24.       s_{k+1} ← s_k com probabilidade e^{(f(s')−f(s_k))/T_{i+1}}
25.       Lista ← Lista - (solução mais antiga) + (solução s_k)
26.    Se f(s_{k+1}) < f(s_BSF)
27.       Atualize s_BSF
28. Fixe a topologia contida em s_BSF e use o valor das conexões como
valores iniciais para o algoritmo de busca local
```

No algoritmo proposto, o esquema de esfriamento atualiza a temperatura T_i da iteração *i* a cada I_T iterações do algoritmo. Esta escolha é baseada no cálculo de uma probabilidade, que depende do aumento no custo e de um parâmetro do algoritmo (temperatura). Entretanto, em muitos casos, o método pode apresentar uma convergência lenta para soluções aceitáveis, dependendo da escolha do esquema de esfriamento (redução da temperatura). Se a temperatura for reduzida de forma muito brusca ao longo das iterações, pode não ser realizada uma exploração satisfatória no espaço de busca, já que, em temperaturas muito baixas, diminui muito a probabilidade de aceitar soluções que deterioram o custo. Por outro lado, se a temperatura for reduzida de forma muito suave, a convergência pode se tornar excessivamente lenta, sendo necessária uma quantidade muito grande de iterações para a convergência.

A cada iteração, uma nova população com *k* soluções é gerada, utilizado uma micro-evolução genética de g_n gerações, a partir da população atual, como ocorre nos Algoritmos Genéticos. Cada solução tem seu custo avaliado, e a melhor solução é escolhida, assim como acontece em *Tabu Search*. A habilidade da técnica *Tabu Search* para escapar de mínimos locais está justamente na geração de um conjunto de novas soluções a cada iteração e na posterior escolha da melhor solução gerada. Entretanto este processo exige o armazenamento das soluções recentemente

visitadas em uma lista, para evitar ciclos na trajetória da busca. A implementação desta lista de soluções proibidas (*tabu*) exige memória e tempo de processamento em cada iteração do algoritmo, pois cada nova solução deve ser comparada com as soluções armazenadas na lista, a fim de verificar se é solução proibida ou não. Diferente do que ocorre em *Tabu Search*, no algoritmo proposto, esta solução nem sempre é aceita. O critério de aceitação é o mesmo utilizado na heurística da técnica *Simulated Annealing* – se a solução escolhida possuir menor custo do que a solução atual, a solução é aceita; caso contrário, pode ser aceita ou não, dependendo do cálculo de uma probabilidade, que é dada pela mesma expressão utilizada no método *Simulated Annealing*. A busca se encerra após $I_{\max}$ iterações ou se o critério de parada baseado no erro de validação for satisfeito, retornando a melhor solução encontrada S_{BSF} (*best so far*).

Durante a busca global o algoritmo proposto pode otimizar também o número de unidades de processamento na camada intermediária. A maioria dos métodos de otimização baseados em poda requer que o projetista defina a topologia máxima da rede neural artificial, para a partir desta estrutura, eliminar conexões até encontrar uma arquitetura de rede com um resultado aceitável. O algoritmo proposto é baseado no princípio que unidades de processamento e conexões somente existem se forem realmente necessárias. Desta forma, as soluções iniciais para o problema representam a topologia mínima de uma rede válida – o número de unidades de processamento na camada de entrada da rede corresponde ao número de atributos, a camada escondida possui somente uma unidade de processamento e a camada de saída possui unidades de processamento correspondendo ao número de classes do problema – e durante a otimização novas unidades de processamento são adicionadas à medida que forem necessárias. As novas unidades de processamento são criadas com todas conexões entre as unidades de processamento adjacentes e eventualmente são eliminadas de acordo com sua influência no desempenho da rede.

6.2 Representação das Soluções

A definição da arquitetura de uma rede neural artificial *Multi-layer Perceptron* (Rumelhart e Weigend, 1990) depende da escolha do número de camadas da rede e da quantidade de nodos de processamento em cada uma destas camadas. Dentre as arquiteturas de rede representadas neste trabalho, todas as topologias de rede possuem somente uma camada intermediária, na qual são consideradas as possíveis conexões entre camadas adjacentes. Desta forma, a topologia máxima da rede neural artificial projetada possui N_1 unidades de processamento de entrada, N_2 unidades de processamento intermediárias e N_3 unidades de processamento de saída.

Os valores para os parâmetros N_1 e N_3 são definidos pelo problema, dependendo do conjunto de dados e do pré-processamento adotado (número de atributos do problema e número de classes). O parâmetro N_2 deve ser definido pelo projetista da rede neural artificial. Desta forma, a quantidade máxima de conexões na rede neural é descrita pela Equação (6.1).

$$N_{\max} = N_1 \times N_2 + N_2 \times N_3 \tag{6.1}$$

Nas simulações realizadas, cada solução *s* é codificada por um vetor, que representa as *C* conexões e os *P* pesos entre as unidades de processamento de uma rede neural artificial *Multi-layer Perceptron*. Neste vetor, a arquitetura da rede neural artificial é especificada por dois parâmetros: *(i)* bit de conectividade, variável de valor booleano que simboliza a existência ou não de uma conexão entre duas unidades de processamento; e *(ii)* o peso desta conexão, um número real que representa o peso da conexão entre duas unidades de processamento. Desta forma,

$$s = (C, P) \tag{6.2}$$

$$C = (c_1, c_2, ..., c_{N_{\max}}), \quad c_i \in \{0,1\}, \quad i = 1,2,..., N_{\max} \tag{6.3}$$

$$P = (p_1, p_2, ..., p_{N_{\max}}), \quad p_i \in \Re, \quad i = 1,2,..., N_{\max} \tag{6.4}$$

sendo que $\Re$ é o conjunto dos números reais.

Na Figura 6.1 é apresentado um exemplo da codificação do vetor que representa a arquitetura da rede neural artificial *Multi-layer Perceptron*. Sendo assim, a conexão *i* é especificada pelo bit de conectividade c_i e pelo valor desta conexão p_i. Nesta representação, se o bit de conectividade c_i assumir o valor *0*, o peso da conexão entre as unidades de processamento é desconsiderado, ou seja, a conexão *i* entre as unidades de processamento associadas é eliminada.

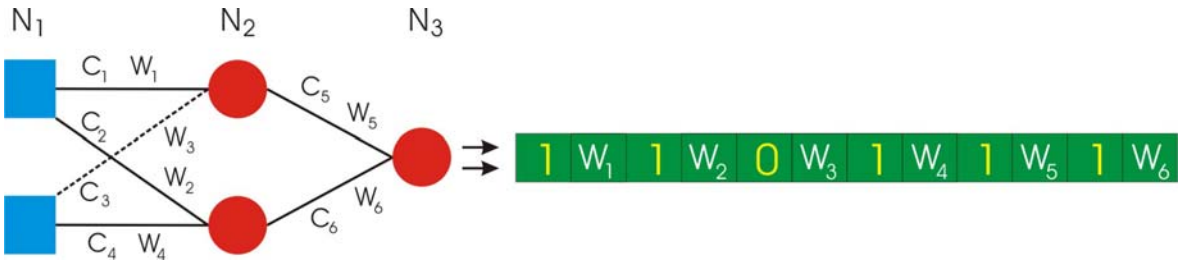

Figura 6.1 - Exemplo de codificação de uma rede neural artificial *Multi-layer Perceptron*.

6.3 Inserção de Novas Unidades na Camada Intermediária

Um processo construtivo é utilizado para adicionar novas unidades de processamento na topologia da rede neural artificial (corresponde a linha 14 do Algoritmo 9). Durante o início do processo de busca a probabilidade de adicionar novas unidades de processamento é alta, porém para realizar uma melhor exploração sobre a superfície de busca, a adição de novas unidades de processamento precisa ser controlada. Este controle é necessário para evitar gerar topologias muito grandes desnecessariamente. Desta forma, novas unidades de processamento são adicionadas à rede de acordo com uma regra probabilística. De acordo com esta regra, a nova probabilidade de adicionar novas unidades é igual à probabilidade atual multiplicada por um fator de redução (ε), que deve ser menor do que *1*. A probabilidade inicial λ_ε e o fator ε são parâmetros de implementação, definidos pelo projetista da rede. Desta forma, a probabilidade λ_i da iteração *i* é definida pela Equação (6.5).

$$\lambda_i = \begin{cases} \varepsilon\lambda_{i-1}, & se\ \ i = k\,I_\lambda, \qquad k = 1,2,...,\dfrac{I_{\max}}{I_\lambda} \\ \lambda_i, & caso\ contrário \end{cases} \tag{6.5}$$

6.4 Mecanismo de Geração de Vizinhança

O operador para geração da vizinhança é utilizado para derivar novas soluções a partir da solução atual (corresponde a linha 17 do Algoritmo 9). A solução inicial é aleatoriamente gerada com N_1 e N_3 sendo variáveis dependentes do problema e $N_2 = \ell$, tal que $\ell = 1,2,...,N_3$. A partir de uma população inicial, uma nova solução s' é gerada através de micro-evoluções genéticas com g_n gerações, a partir da solução atual $s = (C,W)$. Os cromossomos são classificados por *Rank Based Fitness Scaling* (Baker, 1987). A função de *Fitness Scaling* converte os valores de adaptabilidade retornados pela função de avaliação em um intervalo adequado para a função de seleção. A função de seleção utiliza os valores de adaptabilidade ordenados para selecionar os indivíduos pais da próxima geração. A função de seleção atribui a mais alta probabilidade de seleção para indivíduos com os maiores valores ordenados. A função *Rank Based Fitness Scaling* ordena os escores de adaptabilidade baseado na ordenação de cada indivíduo em vez do seu escore. O *rank* de cada indivíduo é sua posição nos escores ordenados.

Os pais da nova geração são escolhidos de maneira probabilística, utilizando *Universal Stochastic Sampling* (Baker, 1987). Neste método os indivíduos são mapeados em segmentos

contíguos de uma linha, de tal modo que cada indivíduo é representado proporcionalmente ao seu índice de aptidão, como no método da Roleta (Baker, 1987). Ponteiros igualmente espaçados são colocados ao longo da linha tantos quantos forem os indivíduos a serem selecionados. Consideremos θ o número de indivíduos a serem selecionados. A distância entre os ponteiros é igual a $1/\theta$ e a posição do primeiro ponteiro é dada por um número gerado aleatoriamente no intervalo [0, $1/\theta$]. Para um exemplo com *6* indivíduos a serem selecionados, a distância entre os ponteiros é de $1/6 = 0.167$. Na Figura (6.2) é apresentado o processo de seleção com um número aleatório *0.1* no intervalo [*0, 0.167*]. Após a seleção, a população constituída para recombinação é composta dos indivíduos: *1, 2, 3, 4, 6, 8*.

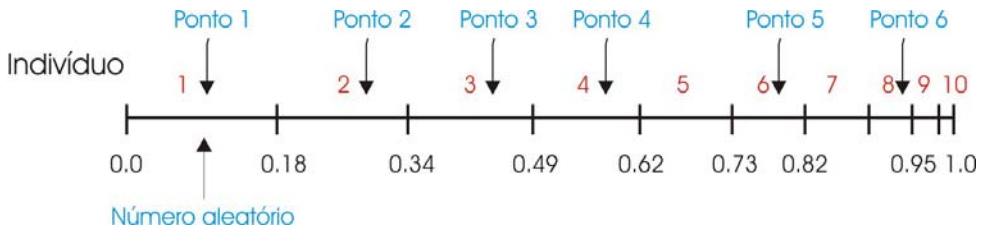

Figura 6.2 - *Universal Stochastic Sampling*

O elitismo não foi utilizado no algoritmo. Para a combinação dos cromossomos pai foi utilizado o operador genético *Uniform Crossover* (Sywerda, 1989). Diferente do *Single* e *Multi-point Crossover* (Sywerda, 1989) que definem pontos onde as informações dos pais serão trocadas, o *Uniform Crossover* generaliza este esquema para fazer de todo o cromossomo um potencial ponto de recombinação. Uma máscara, do mesmo comprimento do cromossomo é criada de forma aleatória e a paridade dos *bits* na máscara indicam os pontos dos cromossomos pai onde as informações serão trocadas.

O operador de mutação utilizado foi *Gaussian Mutation* (Sexto et al, 1999). Este tipo de mutação consiste basicamente na adição de um valor aleatório de uma distribuição gaussiana a cada elemento escolhido para criar a nova população.

A geração de novas soluções depende do tipo de busca empregado. Se durante a otimização o número de unidades de processamento na camada intermediária também estiver sendo otimizado, os cromossomos que representam as soluções possuirão tamanho variável.

O tipo de busca também influencia no número de unidades de processamento na camada intermediária. Caso a otimização seja restrita aos valores dos pesos e conexões da rede (sem considerar o número de unidades na camada intermediária) o projetista deve definir N_2 com o número máximo de unidades permitido. Caso contrário, o valor de N_2 será automaticamente ajustado para a menor topologia de rede possível.

Caso os cromossomos possuíam tamanho variável, a operação de recombinação é realizada combinando as partes dos cromossomos pais de mesmo tamanho, como no exemplo apresentado na Figura 6.3.

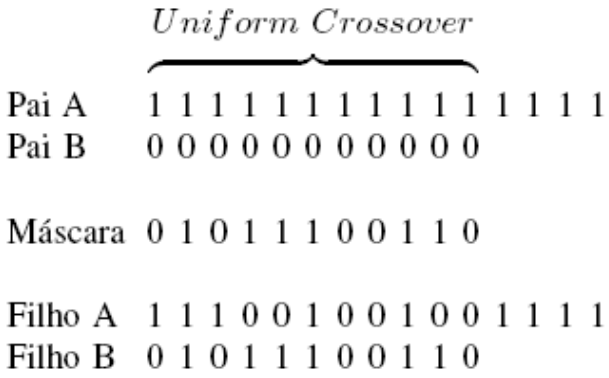

Figura 6.3 - Exemplo da operação de recombinação em cromossomos de tamanho diferente.

6.5 Avaliação do Custo

Diferente dos algoritmos construtivos, que geram uma solução somente ao final do processo, os algoritmos iterativos derivam possíveis soluções a cada iteração. A função de custo é utilizada para avaliar o desempenho entre iterações consecutivas e selecionar uma solução que minimize (ou maximize) uma função objetivo (corresponde a linha 19 do Algoritmo 9).

Considerando N_C classes em uma base de dados qualquer, a classe verdadeira do padrão *x* do conjunto de treinamento P_t é definida segundo a Equação (6.6):

$$\gamma(x) \in \{1,2,...,N_C\}, \quad \forall x \in P_t \tag{6.6}$$

Para os problemas de classificação, a regra o vencedor-leva-tudo (*winner-takes-all*) foi utilizada na saída da rede neural artificial para definir a resposta da rede. Segundo esta regra, a unidade de saída da rede neural artificial que apresentar o maior valor determina a classe do padrão classificado. Assim, o número de unidades de saída da rede neural artificial ($N3$) é igual ao número de classes do problema (N_C).

Sendo $o_k(X)$ o valor de saída da unidade de processamento de saída *k* para o padrão *X*, a classe associada ao padrão *X* é descrita pela Equação (6.7).

$$\phi(X) = \arg\max o_k(X), \quad \forall X \in P_t, \quad k \in \{1,2,...,N_3\} \tag{6.7}$$

Uma vez definida a classe verdadeira e a classe associada ao padrão *X*, o erro da rede neural artificial para o padrão *X* é definido pela Equação (6.8).

$$\varepsilon(X) = \begin{cases} 1, & se \quad \phi(X) \neq \gamma(X) \\ 0, & se \quad \phi(X) = \gamma(X) \end{cases} \tag{6.8}$$

Assumindo as convenções acima adotadas, o *erro de classificação* para o conjunto P_t, que representa a porcentagem de padrões de treinamento classificados de maneira incorreta é descrito pela Equação (6.9).

$$E(P_t) = \frac{100}{\#P_t} \sum_{X \in P_t} \varepsilon(X) \tag{6.9}$$

onde $\#P_t$ é a quantidade de padrões do conjunto P_t.

Sendo $N_{\max}$ o número máximo de conexões presentes na topologia da rede neural artificial, a porcentagem de conexões utilizadas pela rede neural artificial é definida pela Equação (6.10).

$$\psi(C) = \frac{100}{N_{\max}} \sum_{i=1}^{N_{\max}} c_i \tag{6.10}$$

Várias funções de custo $f(s)$ podem ser utilizadas para avaliar a qualidade da solução gerada. Neste trabalho foram investigadas cinco abordagens distintas: *(i)* média aritmética entre a porcentagem de conexões utilizadas pela rede neural artificial e seu erro; *(ii)* média aritmética ponderada entre a percentagem de conexões, a percentagem de unidades escondidas utilizadas pela rede e o erro de classificação obtido pela rede; *(iii)* mecanismo *weight-decay* (Hinton, 1986), inspirado pela implementação *weight decay* do algoritmo de aprendizagem *Backpropagation*; *(iv)* estratégia de otimização multiobjetivo nos operadores genéticos; e *(v)* combinação das estratégias *weight-decay* e otimização multiobjetivo.

Em todas as abordagens o objetivo é minimizar simultaneamente o erro de treinamento e o número de conexões da rede neural artificial. Somente arquiteturas válidas (redes que possuam ao menos uma unidade de processamento na camada de entrada, intermediária e saída) são consideradas.

Método da média aritmética

O método da média aritmética é um dos meios mais intuitivos de converter problemas com múltiplos objetivos em problemas com um único objetivo (Déb, 2001). Para isso, a técnica utiliza a média aritmética entre os objetivos a serem otimizados no problema.

No problema investigado, os dois objetivos a serem otimizados são o número de conexões e o erro do modelo conexionista sobre o conjunto de dados. Desta forma, a função $f_1(s)$ que mede o custo (qualidade ou adaptabilidade) de cada solução é dada pela média aritmética entre a porcentagem de conexões da rede e seu erro sobre o conjunto de dados de treinamento.

Para problemas de classificação, a função $f_1(.)$ para a solução *s* é definida pela Equação (6.11).

$$f_1(s) = \frac{1}{2}(E(P_t) + \psi(C)) \tag{6.11}$$

Para problemas de previsão, a função $f_1(.)$ é dada pela média aritmética entre o erro percentual quadrático (SEP – *squared error percentage*) sobre o conjunto de treinamento e a porcentagem de conexões utilizadas pela rede neural artificial.

$$f_1(s) = \frac{1}{2}(SEP(P_t) + \psi(C)) \tag{6.12}$$

onde o erro SEP é definido pela Equação (6.13).

$$SEP = 100\frac{o_{\max} - o_{\min}}{N_C \# P_t}\sum_{p=1}^{\#P_t}\sum_{i=1}^{N_C}(\phi(X)_{pi} - \gamma(X)_{pi})^2 \tag{6.13}$$

onde $o_{\min}$ e $o_{\max}$ são os valores de mínimo e máximo dos coeficiente de representação do problema (assumindo que são os mesmos para todas as unidades de processamento de saída).

O método da média aritmética ponderada

Este é um tipo de média que considera a relevância proporcional de cada componente, ao invés de considerar os componentes com pesos equivalentes como na média aritmética simples. No problema investigado, os componentes a serem considerados na média aritmética ponderada são: a percentagem de conexões da rede, a percentagem de unidades escondidas e o erro do modelo conexionista sobre o conjunto de dados. Estes valores vão ser ponderados pelas variáveis α, β e

κ, respectivamente. Para problemas de classificação, a função $f_1(.)$ para a solução *s* é definida pela Equação (6.14).

$$f_1(s) = \frac{(E(P_t) * \alpha) + (\psi(C) * \beta) + (pN * \kappa)}{(\alpha + \beta + \kappa)} \quad \textbf{(6.14)}$$

Para problemas de previsão, a função $f_1(.)$ é dada pela média aritmética entre o erro percentual quadrático (SEP – *squared error percentage*) sobre o conjunto de treinamento e a porcentagem de conexões utilizadas pela rede neural artificial.

$$f_1(s) = \frac{(SEP(P_t) * \alpha) + (\psi(C) * \beta) + (pN * \kappa)}{(\alpha + \beta + \kappa)} \quad \textbf{(6.15)}$$

onde pN representa a porcentagem de unidades escondidas utilizadas pela rede, $\alpha = 1$, $\beta = 0.5$ e $\kappa = 0.25$. Estes valores foram escolhidos empiricamente.

O método *weight decay*

A abordagem *weight decay* (Hinton, 1986) foi inicialmente sugerida como uma implementação para melhorar o desempenho do algoritmo de treinamento *Backpropagation*. O algoritmo modifica a função de custo de forma a evitar que os pesos da rede se afastem significativamente de zero, privilegiando o *bias* de redes não sensíveis a ruídos. A técnica de *weight decay* favorece fortemente a ocorrência de muitos pesos de pequena magnitude, enquanto a técnica de *weight elimination* (Weigend et al, 1991) permite a ocorrência de pesos significativamente diferentes de zero. (Hinton, 1986; Kramer, e Sangiovanni-Vincentelli, 1988; Weigend et al, 1991).

O algoritmo *Backpropagation* usa o gradiente descendente para direcionar o processo de busca, cujo objetivo é minimizar a função de erro da rede neural artificial.

$$f(s) = \frac{1}{2}\sum_i (d_i - y_i)^2 \quad \textbf{(6.16)}$$

onde d_i representa a saída desejada e y_i representa a saída obtida da rede neural artificial. Conexões entre unidades de processamento com grandes valores de pesos podem comprometer a capacidade de generalização, pois resultam em variância excessiva na saída da rede neural artificial. O método é implementado com a adição de um termo de *bias B* (penalização) à função de custo original, como descrito pela Equação (6.17).

$$f(s) = f(s_0) + \frac{1}{2}\mu B \tag{6.17}$$

onde μ representa a importância do termo de penalização. Este termo faz com que os pesos tendam a convergir para o menor valor absoluto. Várias abordagens para o termo de penalização já foram propostas. Uma escolha simples para o termo é definido pela Equação (6.18).

$$B = \sum_{ji} w_{ji}^2 \tag{6.18}$$

onde w_{ji} representa o peso da conexão entre a unidade de processamento *j* e a unidade de processamento *i*. Porém, o *bias* da Equação (6.17) afeta igualmente as conexões com grandes e pequenos valores de pesos. Uma alternativa seria afetar com maior intensidade as conexões com pesos maiores:

$$B = \sum_{ji} \frac{w_{ji}^2}{(1 + w_{ji}^2)} \tag{6.19}$$

Uma escolha mais elaborada foi sugerida por Weigend et al., (1991), definida na Equação (6.20).

$$B = \sum_{ji} \frac{\left(\frac{w_{ji}^2}{w_0^2}\right)}{\left(1 + \frac{w_{ji}^2}{w_0^2}\right)} \tag{6.20}$$

onde w_0 é um parâmetro livre e quando seu valor é pequeno, a rede privilegia conexões com pesos grandes. Quando w_0 é grande, a rede privilegia conexões com pesos pequenos.

Neste estudo o mecanismo *weight decay* é utilizado na função de custo do método proposto para eliminar soluções com pesos grandes, as quais podem codificar padrões ruidosos, como definido na Equação (6.21).

$$f_2(s) = \frac{1}{2}\sum_i E(P_t) + \frac{1}{2}\psi(C) + \frac{1}{2}\mu \sum_{ij} \frac{w_{ij}^2}{(1 + w_{ij}^2)} \tag{6.21}$$

O método multiobjetivo

Grande parte dos problemas do mundo real envolve a otimização de múltiplos objetivos. Porém, a maioria dos métodos para resolução desses problemas evita as complexidades que um problema multiobjetivo envolve. Com isso, surgiram muitos métodos para converter problemas multiobjetivos em problemas com um único objetivo.

Dispondo-se de vários métodos de conversão, esqueceu-se que, na verdade, o problema de otimização com um objetivo é derivado de problemas com mais de um objetivo, e que existem diferenças fundamentais entre eles. A principal delas está relacionada à solução do problema. Por tratar de objetivos conflitantes, na otimização multiobjetivo, cada objetivo corresponde a uma solução ótima. Isso faz com que esses problemas apresentem várias soluções ótimas, enquanto que algoritmos que solucionam problemas de otimização com um objetivo normalmente geram apenas uma solução ótima. Porém, sob o ponto de vista prático, é necessário que apenas uma solução seja escolhida. Se existem diferenças suficientes entre soluções ótimas correspondentes a diferentes objetivos, as funções objetivo são ditas conflitantes entre si. Quando isso ocorre, um conjunto de soluções ótimas surge. Essas soluções ótimas são conhecidas como soluções ótimas de Pareto (Coello, 1996).

O problema da otimização multiobjetivo pode ser definido como a busca por um vetor de variáveis de decisão que satisfaça às restrições e otimize uma função cujos elementos representam as funções objetivo. Estas funções formam uma descrição matemática dos critérios de performance que estão geralmente em conflito mútuo. Assim, otimizar significa encontrar as soluções que forneçam o melhor compromisso entre os diversos objetivos para o problema.

O problema geral de otimização multiobjetivo pode ser definido como: encontrar um vetor $s' = [s_1', s_2', ..., s_n']$ que satisfaça as *m* restrições de desigualdade, $d_i(s) \geq 0, \quad i = 1,2,...,m$, as *p* restrições de igualdade, $h_i(s) = 0, \quad i = 1,2,..., p$ e otimize um vetor função, $f(s) = [f_1(s), f_2(s),..., f_k(s)]$.

Desta forma, o problema não tem uma solução única, uma solução global, mas um conjunto de soluções igualmente eficientes, não inferiores entre si, conhecidas como conjunto Pareto ótimo (Coello, 1996). Matematicamente, o conceito de Pareto ótimo pode ser definido como segue: considerando nenhuma perda de generalização, um problema de minimização e duas soluções $s_1, s_2 \in S$. Pode ser afirmado que $s_1 \prec s_2$ $(s_1 \text{ domina } s_2)$ *se*:

$$\forall i \in \{1,2,...,n\} : f_i(s_1) \leq f_i(s_2) \ \wedge \\ \exists j \in \{1,2,...,n\} : f_j(s_1) < f_j(s_2) \quad \textbf{(6.22)}$$

Assim, s' é Pareto ótimo se não existir um vetor de variáveis de decisão $s \in S$ que decresça algum critério sem causar um simultâneo acréscimo em pelo menos um outro critério. Os vetores s correspondentes às soluções incluídas no conjunto pareto ótimo são chamados não-dominados (Deb, 2001).

Neste trabalho, a estratégia multiobjetivo é utilizada na função de avaliação dos operadores genéticos para evoluir a população, considerando os dois objetivos a serem minimizados – o tamanho da rede neural artificial e sua generalização. Como em um problema multiobjetivo a definição da adaptabilidade de cada uma das soluções deve considerar todas as variáveis de decisão, é necessário definir um critério de *ranking* para ordenar as soluções obtidas em cada uma das gerações da evolução genética.

A definição de *ranking* utilizada foi a mesma utilizada em (Coello, 1996). Desta forma, o *rank* de um indivíduo *i* na geração *g* é igual ao número de indivíduos $j \neq i$ que dominam o indivíduo *i* na geração *g*. Assumindo estas restrições, um indivíduo pareto ótimo (não dominado por nenhum outro indivíduo da população) possui sempre *rank 0*.

O melhor indivíduo da população é considerado como sendo o indivíduo que possui o menor *rank*. No caso de mais de um indivíduo possuir o mesmo *rank*, o melhor indivíduo entre eles é a solução que possui a melhor generalização, ou seja, o menor erro obtido pela rede neural artificial sobre o conjunto de treinamento.

A seleção dos indivíduos pais, utilizados na geração de novas soluções, é determinada pela probabilidade destes serem escolhidos pelo método da roleta (Baker, 1987). A probabilidade de cada indivíduo é descrita pela Equação (6.23).

$$\varphi_{i,g} = \frac{\zeta_g - rank_{i,g}}{tam_p - 1} \\ \sum_{j=0} rank_{j,g} \quad \textbf{(6.23)}$$

onde ζ_g é o *rank* do melhor indivíduo da população na geração *g*, $rank_{i,g}$ é a ordem (*rank*) do indivíduo *i* na geração *g*, e tam_p é o tamanho da população.

6.6 Esquema de Esfriamento

Para atualização da variável de temperatura utiliza-se o esquema geométrico de esfriamento (corresponde a linha 24 do Algoritmo 9), segundo o qual a nova temperatura é igual à temperatura atual, multiplicada por um fator de redução (*r*), que deve ser menor do que *1*, porém, próximo de *1* (Dowsland, 1993). A temperatura inicial T_0 e o fator de temperatura *r* são parâmetros de implementação, definidos pelo projetista da rede, assim como I_T (quantidade de iterações entre duas atualizações sucessivas da temperatura) e $I_{\max}$ (máximo número de iterações permitidas). Desta forma, a temperatura T_i da iteração *i* é definida pela Equação (6.24).

$$T_i = \begin{cases} rT_{i-1}, & se \;\; i = k\, I_T, \qquad k = 1,2,...,\dfrac{I_{\max}}{I_T} \\ T_i, & caso\ contrário \end{cases} \tag{6.24}$$

6.7 Critérios de Parada

A parada do treinamento ocorre se for satisfeito o critério GL_5 do *Proben1* (Prehelt, 1994) por duas vezes (para evitar que o treinamento seja interrompido por oscilações iniciais no erro de validação), ou se a quantidade máxima de *I* iterações for alcançada (corresponde a linha 6 do Algoritmo 9).

O critério GL_5 fornece uma idéia sobre a perda da generalização durante o treinamento da rede, sendo muito útil para evitar *overfitting* (Prehelt, 1994). É definido como o aumento do erro de validação em relação ao erro de validação mínimo obtido. Nas Equações (6.25) e (6.26), *E* representa a função de erro quadrático, $E_{tr}(i)$ o erro médio por padrão sobre o conjunto de treinamento, estimado durante a iteração *i*. A variável $E_{va}(i)$ é o erro no conjunto de validação após a iteração *i*, e $E_{opt}(i)$ corresponde ao menor erro de validação obtido até a iteração *i*.

$$E_{opt}(i) = \min_{i' \le i} E_{va}(i') \tag{6.25}$$

A perda de generalização (*GL*) na iteração *i* é definida como o aumento do erro de validação em relação ao menor erro de validação obtido até a iteração *i* (em porcentagem).

$$GL(i) = 100 \times \left(\frac{E_{va}(i)}{E_{opt}(i)} - 1\right) \tag{6.26}$$

Desta forma, é estabelecido um limiar Ω, de modo que o critério GL_{Ω} é caracterizado pela parada do treinamento quando a perda de generalização ultrapassa o valor de Ω. Sendo assim, o critério GL_5 estabelece a parada do treinamento quando $GL_{\Omega} > 5$.

6.8 Seleção de Atributos Relevantes

Durante o processo de otimização dos pesos e conexões da rede o algoritmo está apto a eliminar unidades de entrada da arquitetura da rede *Multi-layer Perceptron* resultante do processo de treinamento. Devido a dinâmica do modelo, as entradas com menor relevância estatística são as primeiras a serem eliminadas da arquitetura da rede, na tentativa do método em diminuir sua topologia. Desta forma o algoritmo proposto também pode ser utilizado para selecionar atributos relevantes, diminuir a dimensionalidade do problema e conseqüentemente a complexidade das redes neurais artificiais geradas. A seleção de atributos realizada pelo método proposto se enquadra em uma junção das definições *wrapper* e *embedded*.

6.9 Algoritmo de Aprendizado Local

As técnicas de otimização global, devido a suas características de busca, são relativamente ineficientes para um ajuste controlado na busca por mínimos (corresponde a linha 28 do Algoritmo 9). Assim, é importante melhorar o desempenho de generalização das redes neurais artificiais treinando as topologias finais com um método de busca local. Esta estratégia corresponde à segunda fase de otimização do método proposto e foi implementado da seguinte forma: sem alterar as topologias geradas pela busca global, as redes finais produzidas são utilizadas como pontos iniciais da busca de um algoritmo local de treinamento. Esta combinação de métodos de otimização global com técnicas de otimização local, freqüentemente referenciada como treinamento híbrido, tem sido utilizada em alguns trabalhos na literatura (Yao, 1999; Yamazaki, 2004).

Em geral, os algoritmos de treinamento mais tradicionais para o treinamento de redes *Multi-layer Perceptron* são baseados em gradiente, sendo que o exemplo mais conhecido é o algoritmo *Backpropagation* (Rumelhart et al, 1986). Neste trabalho, o algoritmo *Backpropagation* foi escolhido pela sua facilidade de implementação, uma vez que o objetivo é mostrar que a combinação cuidadosa de técnicas simples é capaz de gerar bons resultados globais.

Capítulo 7

Metodologia dos Experimentos

Neste capítulo são descritas as bases de dados utilizadas nas simulações e as principais características dos experimentos realizados. Foram utilizados cinco problemas de classificação e dois problemas de previsão.

7.1 Base de dados do Nariz Artificial

A base de dados corresponde à exposição do protótipo de um nariz artificial (Santos, 2000) frente a odores provenientes de três safras distintas de um mesmo vinho tinto comercial (Almadém, Brasil - safra 1995, 1996 e 1997) produzido com uvas do tipo merlot. A partir deste protótipo foram realizados testes de sensibilidade no arranjo de seis sensores baseados em *filmes de polipirrol*[1]. Os sensores foram preparados com diferentes dopantes, por polimerização *in situ*[2], em *substratos de ITO* (substrato vítreo condutor).

A base de dados utilizada foi obtida pela equipe multidisciplinar do Projeto Aroma, formada por membros dos departamentos de Informática, Física e Química Fundamental da Universidade Federal de Pernambuco (Souza, 1999).

A aquisição dos padrões de odores no nariz artificial envolveu três etapas distintas:

(i) Coleta da linha de base (com duração de 10 minutos). Os sensores são expostos a um gás inerte (gás Nitrogênio);

(ii) Coleta dos valores de resistência do arranjo de sensores, frente ao gás analisado (com duração de 40 minutos); e

(iii) Purga ou limpeza dos sensores com gás Nitrogênio (com duração média de 5 minutos).

---

1 Polímero Condutor, material orgânico, do tipo "plástico", geralmente derivado do petróleo, que conduz eletricidade.

2 No local de origem; de forma natural.

Foram realizadas três repetições de aquisição de dados. Em cada uma destas repetições, para cada uma das três safras de vinho, o valor da resistência de cada um dos seis sensores foi registrado de meio em meio segundo, sendo que o processo de aquisição estendeu-se por cinco minutos. Dessa forma, cada sensor obteve *600* valores registrados para cada safra de vinho. O conjunto formado pelos seis valores dos sensores no mesmo instante de tempo foi considerado como uma instância da base de dados. Assim, cada repetição contém *1.800* padrões (*600* padrões de cada safra). Como existem três repetições, tem-se um total de *5.400* padrões.

7.2 Base de dados Diabetes

Esta base de dados possui como título original "Pima Indians Diabetes Database" e é de propriedade do National Institute of Diabetes and Digestive and Kidney Diseases. A base Diabetes contém dados pessoais e resultados de exames médicos de pacientes, utilizados para a classificação da possibilidade de um índio Pima ser diabético. Neste conjunto de dados todos os pacientes são mulheres com idade mínima de *21* anos e pertencentes à linhagem de Índios Pima que vivem próximos a Phoenix, Arizona, USA. A base foi obtida do repositório UCI[3] de bases de dados para aprendizagem de máquina.

A base de dados Diabetes é formada por 768 registros (indivíduos) com 8 atributos contínuos, que correspondem a informações médicas:

(i) Números de vezes que a paciente ficou grávida;
(ii) Concentração de glicose no plasma em teste de tolerância de glicose oral de 2 horas;
(iii) Pressão sanguínea mínima (diastólica) (mm/Hg);
(iv) Medida das dobras na pele do tríceps (mm);
(v) Aplicações de 2 em 2 horas de soro com insulina (um U/ml);
(vi) Índice de massa corpórea (peso em Kg / (altura em m)^2);
(vii) Função de genealogia de diabetes;
(viii) Idade da paciente (anos).

A base de dados registra se os pacientes observados apresentam sinais de diabetes de acordo com critérios definidos pela Organização Mundial de Saúde (World Health Organization), isto é, se 2 horas após aplicação de plasma glicose existem pelo menos 200 mg/dl em qualquer exame, de rotina ou não, correspondendo a duas classes binárias.

3 http://www.ics.uci.edu/~mlearn/MLRepository.html

A análise dos dados foi realizada a partir de uma algoritmo que gerou resultados no intervalo entre 0 e 1. Estes valores contínuos foram transformados em valores binários 0 e 1 com valor de corte = 0.448, sendo:

(i) Valor 0 (zero): indica que o paciente não possui diabetes – 500 registros (65.1%);

(ii) Valor 1 (um): o paciente possui diabetes – 268 registros (34.9%).

7.3 Base de dados Íris de Fisher

O conjunto de dados da Íris foi apresentado por Fisher (1936) ao ilustrar os princípios da análise discriminante. A base de dados possui *150* observações, cada observação contém seis descritores:

(iii) Número que representa o tipo da espécie da flor;

(iv) Nome da espécie (Íris Setosa, Íris Virgínia e Íris Versicolor);

(v) Largura da pétala;

(vi) Comprimento da pétala;

(vii) Largura da sépala;

(viii) Comprimento da sépala.

O conjunto de dados contém *50* instâncias de cada classe, cada classe refere-se a um tipo de Íris (Setosa, Virgínia e Versicolor). Uma classe é linearmente separável das outras duas; e as outras duas não o são uma da outra. A base de dados possui *4* atributos e *150* padrões. Esta base também foi obtida do repositório UCI[4] de bases de dados para aprendizagem de máquina.

7.4 Base de dados Tireóide

A base de dados contém informações relacionadas à disfunção da tireóide. O problema é determinar se um paciente tem uma tireóide funcionando normalmente, com níveis excessivos de hormona tiroideia (hipertireóide) ou níveis insuficientes deste hormônio (hipotireóide). Existem 7200 casos na base de dados, sendo *3.772* são de 1985 e *3.428* de 1986 (Quinlan, 1987). A classe hipertireóide representa *2.3%* (166 casos) dos exemplos, a classe hipotireóide representa *5.1%* (386 casos) das observações, enquanto o grupo de exemplos com níveis normais de hormona tiróideia somam os *92.6%* (6666 casos) dos casos restantes. Esta base de dados é extremamente desbalanceada e considerada um problema difícil para métodos tradicionais de classificação. A

4 http://www.ics.uci.edu/~mlearn/MLRepository.html

base de dados possui *21* atributos com *15* variáveis binárias e *6* contínuas, utilizadas para determinar em qual das três classes o paciente pertence.

Estes foram os dados utilizados por Quinlan no estudo de caso de seu artigo "Simplifying Decision Trees" (Quinlan, 1987). Infelizmente estes dados diferem da versão apresentada originalmente por Quinlan, na qual são descritos todos os atributos da base e não se sabe maiores detalhes sobre estes dados. A única coisa que se pode afirmar, baseado em experimentos na encontrados na literatura, é que esta é uma base de dados difícil de ser treinada com Redes Neurais Artificiais utilizando *Backpropagation*[5]. Esta base também foi obtida no repositório UCI[6] de bases de dados para aprendizagem de máquina.

7.5 Base de Dados Reconhecimento de Vinho

Os dados são resultado da análise química de vinhos provenientes da mesma região da Itália, porém produzidos com diferentes tipos de uvas (Forina et al, 1991). As análises determinaram as quantidades de *13* componentes encontrados em cada um dos três tipos de vinho. Os padrões possuem *13* atributos e três classes.

Os atributos correspondem aos seguintes componentes:

(i) Teor Alcoólico;
(ii) Ácido Málico;
(iii) Cinzas;
(iv) Alcalinidade das Cinzas;
(v) Magnésio;
(vi) Fenóis;
(vii) Flavanóides.
(viii) Fenóis Não-Flavanóides
(ix) Antocianinas;
(x) Intensidade da Cor;
(xi) Tonalidade da Cor;
(xii) Taxa de Diluição;
(xiii) Prolina.

A base de dados foi dividida em dois conjuntos, treinamento e teste. Foi realizado treinamento *off-line* utilizando *133* padrões e os *45* padrões restantes foram utilizados para testar as redes neuro-

5 http://mlearn.ics.uci.edu/databases/thyroid-disease/ann-thyroid.names

6 http://www.ics.uci.edu/~mlearn/MLRepository.html

difusas. Foram realizadas replicações dos experimentos utilizando cinco partições diferentes da base de dados. Esta base também foi obtida do repositório UCI[7] de bases de dados para aprendizagem de máquina.

7.6 Série Caótica *Mackey Glass*

A série *Mackey-Glass* é uma série temporal contínua, unidimensional e apresenta oscilações quase periódicas. O objetivo da série é modelar a dinâmica da produção de células brancas do sangue de pacientes com câncer. A série resulta da integração da equação diferencial caótica com atraso proposta por Mackey e Glass (1977), dada pela Equação (7.1):

$$\frac{dx}{dt} = -bx(t) + a\frac{x(t-\tau)}{1+x(t-\tau)} \qquad \textbf{(7.1)}$$

A série não apresenta um período definido de forma clara, sendo também bastante sensível às condições iniciais e dependendo dos valores dos parâmetros τ, a, b e $x(0)$, e ainda pode apresentar comportamento caótico. Para a construção da série, fixou-se os parâmetros $a = 0.2$, $b = 0.1$, $\tau = 17$, $x(0) = 1.2$ e $x(t) = 0$ para $t < 0$. Foram utilizados valores no intervalo $t \in [0;\ 2000]$, tendo-se selecionado 1000 pares entrada-saída no intervalo $t \in [118;\ 1117]$. A série *Mackey-Glass* é reconhecida como uma referência no estudo da aprendizagem e da capacidade de generalização em diferentes tipos de classificadores.

O treinamento foi realizado utilizando *500* pontos ($t = 118$ a $t = 618$). Destes padrões, *250* pontos foram utilizados para validação ($t = 618$ a $t = 868$), e a rede neural artificial foi testada com outros *250* pontos ($t = 868$ a $t = 1118$). Replicações foram realizadas utilizando os valores da série temporal em diferentes pontos temporais. Foram construídas cinco séries *w* distintas:

w(1)=[x(t - 12) x(t - 8) x(t - 4) x(t), y(t+ 4)]

w(2)=[x(t - 18) x(t - 12) x(t - 6) x(t), y(t + 6)]

w(3)=[x(t - 24) x(t - 16) x(t - 8) x(t), y(t + 8)]

w(4)=[x(t - 30) x(t - 20) x(t - 10) x(t), y(t + 10)]

w(5)=[x(t - 36) x(t - 24) x(t - 12) x(t), y(t + 12)

7 http://www.ics.uci.edu/~mlearn/MLRepository.html

7.7 Série Temporal *Gas Furnace*

A série temporal *Gas Furnace* corresponde ao processo de uma fornalha a gás no qual o fluxo de gás *x(t)* corresponde a entrada da fornalha e a concentração de CO_2 *y(t)* a saída da fornalha. Esta série corresponde à conhecida série Box e Jenkins *Gas Furnace* (Box e Jenkins, 1970). A base de dados original contém 296 pares de dados [*x(t), y(y)*]. Replicações foram realizadas utilizando os valores da série temporal em diferentes pontos temporais. Para os experimentos foram construídas cinco séries *w* distintas:

w(1)=[y(t - 1) y(t - 2) x(t - 1) x(t - 2), y(t)]

w(2)=[y(t - 1) y(t - 3) x(t - 1) x(t - 3), y(t)]

w(3)=[y(t - 2) y(t - 3) x(t - 2) x(t - 3), y(t)]

w(4)=[y(t - 3) y(t - 4) x(t - 3) x(t - 4), y(t)]

w(5)=[y(t - 4) y(t - 5) x(t - 4) x(t - 5), y(t)]

Foi realizado treinamento *off-line* utilizando os primeiros *145* pontos da série. A série possui quatro atributos. As redes neuro-difusas foram testadas com os *145* pontos remanescentes.

7.8 Resumo das Bases de Dados Utilizadas

Na Tabela 7.1 é apresentado um resumo das bases de dados utilizadas nas simulações realizadas nesta tese. Nesta tabela são apresentados: o número de instâncias; o número de atributos binários (b), contínuos (c), nominais (n) e total (t); o número de classes binárias (e) e contínuas (c); e a entropia em *bits* por exemplo de cada base de dados. Neste contexto, a entropia é o número esperado de *bits* necessários para codificar cada classe. A entropia é máxima quando o número de atributos nas classes são equivalentes. A entropia é mínima e igual a *0* quando todos os exemplos pertencem à mesma classe. A entropia para *c* classes pode ser calculada pela Equação (7.2).

$$entropia(\wp) = -\sum_{c} p(c)\log_2 p(c) \qquad \textbf{(7.2)}$$

sendo $\wp$ o número de exemplos e *p* a proporção de exemplos de cada classe.

Tabela 7.1 -Resumo das bases de dados utilizadas.

Base de dados	Instâncias	Atributos				Classes		Entropia
		b	c	n	t	b	c	
Nariz Artificial	5.400	0	6	0	6	3	0	1.5849
Diabetes	768	0	8	0	8	2	0	0.9332
Íris	150	0	4	0	4	3	0	1.5849
Tireóide	7.200	9	6	0	21	3	0	0.4468
Mackey Glass	1.000	0	4	0	4	0	1	2.0000
Gas Furnace	290	0	4	0	4	0	1	2.0000
Reconhecimento de Vinho	178	0	13	0	13	3	0	1.5668

Na escolha das bases de dados a serem utilizadas nos experimentos se buscou bases de dados com características diferentes, mesclando problemas de classificação e previsão. A preocupação com a diversidade nos experimentos visou dar maior suporte aos resultados obtidos. Segundo um interessante estudo teórico de Wolpert e Macready (1997), a performance média de qualquer par de algoritmos iterativos (determinísticos ou não) sobre todos os tipos de problemas é idêntica. Sendo assim, se um algoritmo obtém uma performance muito boa sobre uma classe de problemas, então necessariamente paga por isso com uma degradação de performance no conjunto remanescente de problemas. Partindo deste pressuposto, uma grande quantidade de simulações é necessária para se obter um resultado conclusivo sobre este tipo de algoritmos.

7.9 Metodologia de Treinamento

O algoritmo de treinamento local utilizado é uma versão do método de *backpropagation* com momento, descrita em (Rumelhart et al, 1986). Para cada topologia, foram realizadas 30 (trinta) inicializações de valores diferentes e aleatórios para as conexões entre as camadas de processamento.

A parada do treinamento ocorre se for satisfeito o critério GL_5 do *Proben1* (Prehelt, 1994) por duas vezes (para evitar que o treinamento seja interrompido por oscilações iniciais no erro de validação), se for satisfeito o critério de *progresso de treinamento* do *Proben1* (Prehelt, 1994), com $P_5(t) < 0.1$, ou se a quantidade máxima de $I_T = 5.000$ iterações for alcançada.

O critério de progresso de treinamento do *Proben1* (Prehelt, 1994) fornece uma idéia de quanto o erro médio de treinamento é maior que o mínimo erro de treinamento, durante uma seqüência de iterações consecutivas. Considerando uma seqüência de *k* iterações numeradas de *n + 1* até *n + k*, sendo *n* divisível por *k*, o progresso de treinamento *P* (medido em partes por mil) após esta seqüência de *k* épocas é dado pela Equação (7.3).

$$P_k(t) = 1000.\left(\frac{\sum_{t' \in t-k+1,\ldots,t} E_{tr}(t')}{k. \min_{t' \in t-k+1,\ldots,t} E_{tr}(t')}\right) \quad \textbf{(7.3)}$$

É estabelecido um limiar, de modo que o critério do progresso de treinamento é caracterizado pela parada do treinamento quando o progresso de treinamento estiver menor que este limiar.

Em qualquer problema a ser resolvido através do uso de redes neurais artificiais com treinamento supervisionado, é necessário o uso de dados que correspondam a pares de entrada-saída já conhecidos. Muitas vezes a quantidade disponível de dados, para o treinamento do modelo conexionista, não é muito grande, mesmo assim faz-se necessário dividir a base de dados existente em dois conjuntos bem definidos: um conjunto de treinamento e um conjunto de teste.

O treinamento da rede neural artificial é realizado a partir do conjunto de dados de treinamento. É, entretanto, necessário medir a performance da rede, considerando seu desempenho a pares de entrada-saída não apresentados durante a fase de treinamento. Esta avaliação é feita a partir do conjunto de teste, que mede o grau de generalização apresentado pela rede para o problema abordado.

Obter esta informação sobre o desempenho da rede durante a fase de treinamento pode ser muito útil, pois pode evitar um problema conhecido como *overfitting* ou sobre-ajuste (Haykin, 2001). Isto é feito com a utilização de um terceiro conjunto de pares de entrada-saída, retirado do conjunto de treinamento, conhecido como conjunto de validação. O conjunto de validação serve para que o desempenho da rede seja avaliado durante a fase de treinamento e também pode ser utilizado como um mecanismo de parada precoce do treinamento (Prechelt, 1994).

Nos experimentos realizados nesta tese, a base de padrões foi dividida em três conjuntos de dados. Os quais correspondem aos conjuntos de treinamento, validação e teste. A proporção de padrões contidos em cada um dos conjuntos foi a seguinte: *50%* dos padrões foram dispostos no conjunto de treinamento; *25%* dos padrões correspondem ao conjunto de validação; e os *25%* restantes foram utilizados no conjunto de teste. Esta divisão dos dados é sugerida pelo relatório *Proben1* (Prechelt, 1994), um conhecido *Benchmark* para o treinamento de redes neurais artificiais.

Outra fase importante do pré-processamento dos dados é a normalização. Para uma maior efetividade e rapidez no treinamento de redes neurais artificiais, é aconselhável a transformação de todos os atributos de valores reais para um mesmo intervalo. Dados com atributos em escalas diferentes podem confundir o aprendizado da rede, fazendo que esta tenha maior consideração por valores com grandes magnitudes. As técnicas de normalização evitam este tipo de problema, transpondo os dados em intervalos bem definidos. Nos experimentos realizados os dados foram normalizados para o intervalo [*-1, +1*].

7.10 Experimentos com os sistemas Neuro-difusos

Adaptative Neuro Fuzzy Inference System

Para realização dos experimentos a implementação da rede ANFIS disponível no toolbox de Lógica Difusa do software de simulação Matlab (Matlab 6.0, 2000) foi utilizada como base.

Evolving Fuzzy Neural Networks

Para realização dos experimentos a implementação da rede Evolving Fuzzy Neural Networks parte do código disponibilizado por Kasabov (1999) foi utilizada como base.

7.11 Experimentos com as Técnicas de Otimização

Algoritmos Genéticos

Nos experimentos realizados com Algoritmos Genéticos, cada cromossomo representa a arquitetura e os pesos de uma rede neural artificial *Multi-layer Perceptron*. A população inicial foi definida com tamanho de *10* cromossomos. Os cromossomos são classificados por *Rank Based Fitness Scaling* (Baker, 1987). A escolha dos pais que vão formar a próxima geração é feita de forma probabilística, utilizando *Stochastic Universal Sampling* (Baker, 1987). Nesta escolha o elitismo – conceito em que os melhores cromossomos são preservados na população – também é considerado, com uma probabilidade de *10%*. Para combinação dos cromossomos pai foi utilizado o operador de recombinação *Uniform Crossover* (Sywerda, 1989), com uma probabilidade de *80%*. O operador de mutação utilizado foi a *Mutação Gaussiana* (Sexto net al, 1999), com uma probabilidade de *10%*. Os critérios de parada utilizado foram: *(i)* o critério GL_5 definido no *Proben1* (Prehelt, 1994), (baseado no erro de classificação para o conjunto de validação); e *(ii)* o número máximo de *500* gerações. Os valores de configuração em todos os experimentos foram escolhidos empiricamente.

Simulated Annealing

O algoritmo de *Simulated Annealing* foi originalmente derivado de simulações em termodinâmica e por esta razão o parâmetro *T* é referenciado como temperatura e a maneira pela qual ela é reduzida é conhecido como processo de resfriamento. A estratégia de resfriamento escolhida nos experimentos foi à regra de resfriamento geométrico. De acordo com esta regra, a nova temperatura é igual à temperatura corrente, multiplicada por um fator de redução – menor, porém próximo a um (Harp et al, 1989). O valor inicial escolhido para a temperatura foi *1.00*, e o fator de redução escolhido foi *0.9*. A temperatura decresce a cada iteração em um número máximo de *1.000* iterações.

Tabu Search

Nos experimentos com *Tabu Search*, é utilizada uma vizinhança com *20* soluções. O algoritmo escolhe a melhor solução que não esteja na lista *tabu*. Para comparar duas soluções foi utilizado o critério de proximidade (Sexton, 1998). Uma nova solução é considerada idêntica a uma solução pertencente a lista *tabu* se: *(i)* cada *bit* de conectividade for igual ao *bit* de conectividade correspondente à solução *tabu*; e *(ii)* o valor de cada conexão na nova solução esta entre $\pm D$ do valor da conexão correspondente na lista *tabu*. O parâmetro *D* é um número real, com valor *0.001*. Como o algoritmo gera várias soluções a cada interação, foi permitido um número máximo de *100* iterações.

Método proposto por Yamazaki

Nos experimentos com esta técnica foi considerada uma probabilidade de *20%* nos operadores de geração de novas soluções para inversão dos *bits* que representam as conexões da rede.

Método proposto GaTSa

Nos experimentos realizados, a população inicial foi definida com *10* cromossomos. O elitismo não foi utilizado. Para recombinação dos cromossomos pai foi utilizado o operador genético *Uniform Crossover* (Sywerda, 1989), com uma probabilidade de *20%*. O operador de mutação utilizado foi *Gaussian Mutation* (Sexto net al, 1999), com uma probabilidade de *10%*.

A estratégia de resfriamento escolhida nos experimentos foi à regra de resfriamento geométrico. O valor inicial escolhido para a temperatura foi *1.00*, e o fator de redução escolhido foi *0.9*.

7.12 Experimentos de Seleção de Atributos

Os pré-requisitos para realização da seleção de atributos baseada nas abordagens investigadas são: definir como será feita a busca no espaço de subconjuntos de atributos; definir como avaliar o desempenho do classificador para guiar a busca; e definir o classificador a ser utilizado.

Nos experimentos realizados foram utilizadas estratégias de busca distintas, dependendo das características da técnica de busca. Para os algoritmos de busca *Hill-Climbing*, *Best-first* e *Beam Search*, foram utilizadas as estratégias de busca *Para Frente*, *Para Trás* e *Aleatória*. O método *Random Bit Climber* difere dos anteriores, pois durante o processo de busca, atributos podem ser adicionados e removidos, desta forma, para caracterizar uma busca em várias direções foram utilizados diferentes estados iniciais – solução inicial sem nenhum atributo, com todos os atributos

ou uma solução com atributos aleatoriamente selecionados. O algoritmo de busca do método *Las Vegas*, assim como o algoritmo proposto GaTSa, implementa estratégias de busca próprias.

A fim de reduzir o tempo de necessário para realização dos experimentos, o classificador utilizado para avaliar a qualidade de cada um dos subconjuntos de atributos foi o algoritmo de aprendizagem baseada em instâncias *K-Nearest Neighbour – k-NN* (Mitchell, 1997). Neste método as instâncias são agrupadas conforme a proximidade entre elas. O *k-NN* assume que todas as instâncias correspondem a pontos no espaço *n*-dimensional. A função que mede a distância entre as instâncias é de muito importante para este algoritmo. A função de distância utilizada foi à distância Euclidiana (Mitchell, 1997).

Nas simulações realizadas, os dados disponíveis foram divididos em dois conjuntos distintos, sendo metade das instâncias disponíveis utilizadas para o processo de seleção de atributos e as restantes utilizadas no processo de validação dos subconjuntos encontrados. A qualidade de cada subconjunto foi dada pelo desempenho do classificador utilizado. Para obtenção do desempenho do classificador foi utilizado um processo de validação cruzada. Durante o processo de seleção de atributos foi utilizada validação cruzada com *5 folds* e na avaliação do subconjunto foram utilizados *10 folds*. Esta diferença nos métodos de estimativa de precisão foi necessária para reduzir a carga computacional dos experimentos durante o processo de busca pelo melhor subconjunto. Na Figura 7.1 é apresentado um esquema de como foi obtido o desempenho de cada uma das soluções avaliadas durante o processo de seleção de atributos relevantes.

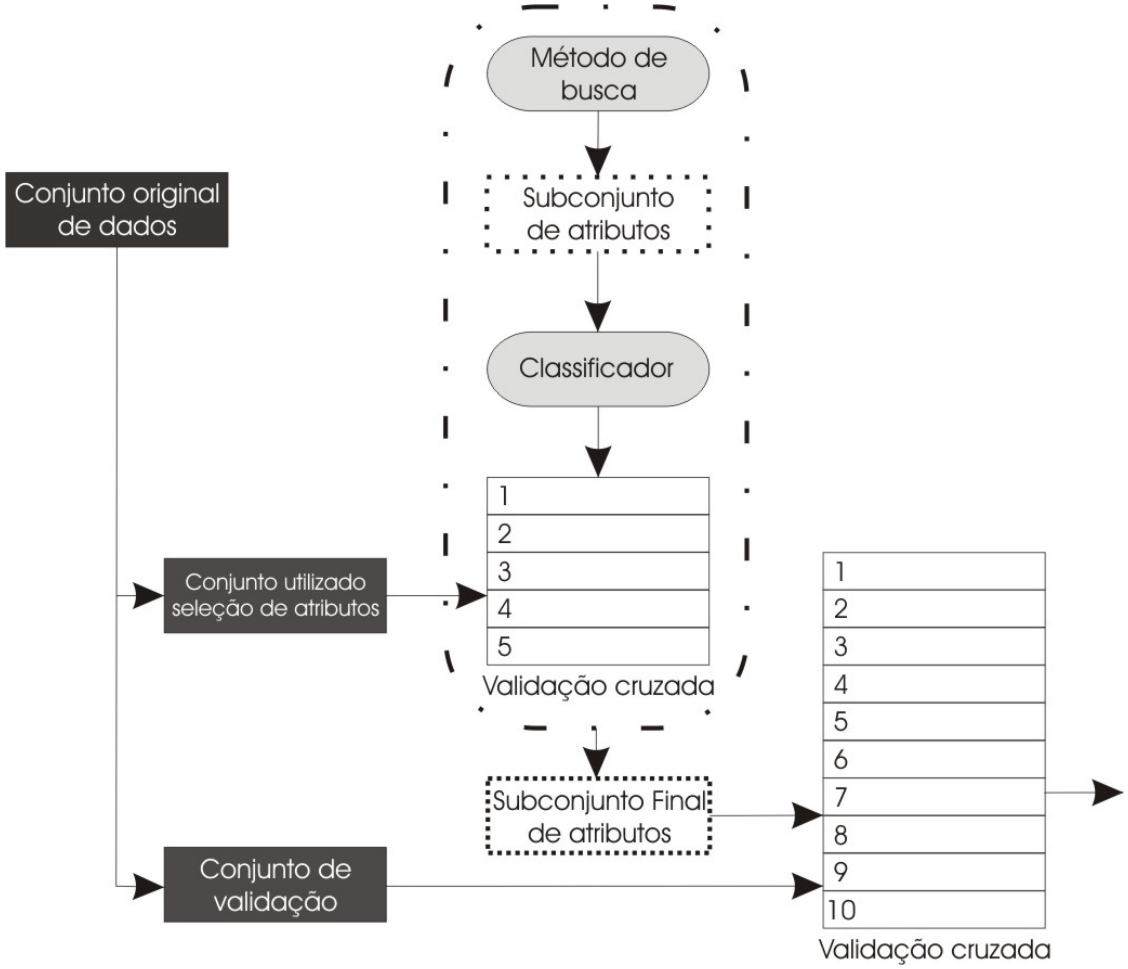

Figura 7.1 - Esquema de avaliação utilizado

7.13 Métodos de Comparação entre as Abordagens

Para verificar a relevância estatística dos resultados apresentados foi utilizado um *Teste de Hipóteses* (Duda et al., 2001; Lehman, 1986, Mitchell 1997). Este procedimento é amplamente utilizado nas áreas em que as variáveis envolvidas estão sujeitas à variabilidade. O teste de hipóteses utilizado na comparação dos métodos investigados é o teste *t* de *Student*. Foram realizados testes pareados dos resultados obtidos nos experimentos.

O teste *t* de *Student* é um teste de hipóteses paramétrico, sendo os seus pressupostos de que as duas variáveis envolvidas são intervalares e que ambas tem distribuição normal. Sendo tais pressupostos verdadeiros, a variável analisada resultante pode ser matematicamente transformada, de modo a formar a distribuição *t* de *Student*; em função da qual se determina se um dado resultado é ou não estatisticamente significativo.

7.14 Planejamento Fatorial de Experimentos

O planejamento fatorial de experimentos (Montgomery, 1997) fornece um método para investigar simultaneamente os efeitos de variáveis múltiplas (fatores) em uma variável de saída (resposta). Este estudo consiste em uma série de verificações, ou testes, nas quais alterações deliberadas são realizadas para inserção de variáveis ou fatores. Os resultados destas alterações são coletados em cada verificação para avaliação do comportamento do método investigado.

Os fatores de um estudo experimental são as variáveis controláveis do processo que podem afetar seu desempenho. Também podem ser introduzidos no experimento fatores não controláveis ao processo, mas que podem ser controlados em laboratório. A inclusão de fatores não controláveis é importante para obter estudos pouco sensíveis às variações de fatores ambientais ou de outros ruídos. O efeito de um fator é definido como a variação na resposta do método, produzida pela mudança do nível desse fator. Denomina-se o nível de cada fator a variação no valor deste fator.

Por exemplo, um experimento com dois fatores A e B, cada um com dois níveis (A-, A+, B-, B+), o efeito principal do fator A é a diferença entre a resposta média no nível alto de A e a resposta média no nível baixo de A. A diferença na resposta entre os níveis de um fator não é mesma em todos os níveis dos outros fatores, existe uma interação entre fatores. Se a interação entre os fatores é significativa o conhecimento da interação AB pode ser mais útil do que o conhecimento do efeito individual de cada fator.

A combinação dos níveis de fatores incluídos no modelo do estudo experimental é conhecida como tratamento. Para a realização do estudo experimental são realizados vários ensaios em cada condição experimental (tratamento). Nestes ensaios, é necessária a utilização de replicações, objetivando avaliar o erro experimental. Replicações aleatorizadas possibilitam verificar o efeito provocado por possíveis fatores que estão agindo no processo, mas que não foram incluídos no estudo.

O planejamento fatorial de experimentos possui como principais vantagens:

(i) Diminuição do número de ensaios;
(ii) Estudo de um número considerável de fatores;
(iii) Detecção das interações entre os fatores;
(iv) Detecção dos níveis ótimos;
(v) Melhoria da precisão dos resultados;
(vi) Otimização dos resultados.

O planejamento fatorial de experimentos foi utilizado no treinamento de sistemas neuro-difusos e da técnica proposta para identificar quais parâmetros mais influenciam em seu desempenho. A partir desta informação o projetista ou o sistema de configuração automática pode

considerar estas variáveis estatisticamente significativas e obter uma melhor generalização do modelo com menor custo. Desta forma, a análise estatística busca identificar a interação e a inter-relação entre as variáveis no projeto deste sistema e identificar os fatores mais relevantes neste processo. Maiores informações sobre planejamento fatorial de experimentos podem ser encontradas no Apêndice A do presente trabalho.

Capítulo 8

Experimentos e Análises

8.1 Planejamento Fatorial de Experimentos

A técnica de Planejamento Fatorial de Experimentos (Montgomery, 1997) foi utilizado para verificar quais fatores possuem maior influência no desempenho dos sistemas neuro-difusos ANFIS (Jang, 1993) e EFuNN (Kasabov, 1998b). A análise da influência de cada um dos parâmetros de configuração de um modelo pode ser muito útil no projeto e treinamento destes modelos. O projetista ou método de treinamento automático pode focar seus esforços no ajuste dos parâmetros mais relevantes e diminuir o custo desta configuração. A expectativa é que uma quantidade pequena de parâmetros possua grande influência média no desempenho dos modelos. Nesta análise são verificados os parâmetros mais influentes e também a interação e inter-relação entre eles em várias bases de dados distintas.

Neste estudo optou-se pela realização de um experimento fatorial com dois níveis (experimento fatorial 2^k), objetivando diminuir a quantidade de experimentos a serem realizados. Os fatores controlados na investigação são apresentados na Tabela 8.1.

Tabela 8.1 - Fatores controlados nos experimentos com ANFIS.

Fatores		Níveis	
		Inferior (-1)	Superior (+1)
A	Número de Funções de Pertinência de Entrada	2	3
B	Tipo de Função de Pertinência de Entrada	Sino	Triangular
C	Tipo de Função de Pertinência de Saída	Linear	Constante
D	Número de Épocas de Treinamento	10	50
E	Taxa de Aprendizagem Inicial	0.01	0.1
F	Método de Otimização das Regras	Híbrido	*Backpropagation*

Os fatores *G = grid partitions* (tipo de particionamento dos dados de entrada), *H = 0* (Erro mínimo de treinamento), *I = 0.9* (Valor de incremento para a taxa de aprendizagem) e *J = 1.1* (valor de decremento para a taxa de aprendizagem) foram mantidos fixos durante os experimentos.

Os fatores controlados nos experimentos com EFuNN são apresentados na Tabela 8.2.

Tabela 8.2 - Fatores controlados nos experimentos com EFuNN.

Fatores		Níveis	
		Inferior (-1)	Superior (+1)
A	Número de Funções de Pertinência	3	5
B	Tipo de Função de Pertinência	Triangular	Sino
C	*Error Threshold*	0.01	0.16
D	*Sensitivity Threshold*	0.90	0.99
E	*m-of-n*	1	3
F	*Maximum Radius*	0.30	0.80

Os fatores de configuração *G = 1/número de exemplos representados pelas unidades de processamento* (taxa de aprendizado para $W1$ e $W2$), *H = nonpruning* (controle de poda das unidades de processamento) e *I = nonaggregation* (controle para agregação das unidades de processamento) foram mantidos fixos com seus valores padrão.

Os ensaios foram realizados em ordem aleatória, sendo considerados seis fatores de controle (definidos nas Tabelas Tabela 8.1 e Tabela 8.2), cada um deles com dois níveis, resultando em 64 combinações. Considerando as replicações, cada uma das combinações dos níveis dos fatores de controle foi realizada cinco vezes, num total de 320 ensaios.

Foram utilizadas duas bases de dados de previsão: *Mackey-Glass* e *Gas Furnace*; e duas bases de classificação: Íris de Fischer e Reconhecimento de Vinhos. Estes experimentos estão publicados em Zanchettin et al (2005) e Zanchettin et al (2007).

Base de dados *Mackey-Glass*

Com base na análise de variância do experimento fatorial (ANOVA) realizado, considerando o nível estatístico de significância $\alpha = 5\%$ na função de distribuição *F, 4* fatores foram identificados com maior influência sobre a resposta da rede neuro-difusa ANFIS. Estas análises são apresentadas na Tabela 8.3. Os fatores mais relevantes são: Tipo da Função de Pertinência de Saída (C), explicando $\approx 31.78\%$ da variância total do sistema; Número de Funções de Pertinência de Entrada (A), que explica $\approx 26.16\%$ da variância; Tipo da Função de Pertinência de Entrada (B), que explica $\approx 0.89\%$ da variância; e Número de Épocas de Treinamento (D), que corresponde a $\approx 0.75\%$ da variância. Também foi identificada interação entre os fatores: Número de Funções de Pertinência de Entrada (A) e Tipo da Função de Pertinência de Saída (C), correspondendo a $\approx 14.15\%$ da variância do sistema; Tipo da Função de Pertinência de Saída (C) e Número de

Épocas de Treinamento (D), com $\approx 0.45\%$ da variância; e Número de Épocas de Treinamento (D) e Taxa de Aprendizagem Inicial (E), correspondendo a $\approx 0.43\%$ da variância total dos dados.

Tabela 8.3 - Resultados da tabela ANOVA (Montgomery e Runger, 2003) para os experimentos ANFIS – Base de dados *Mackey-Glass*.

	Soma Quadrática	Graus de Liberdade	Média Quadrática	F_0	Nível de significância
Fatores					
A	0.0128083	1	0.0128083	282.93	0.000
B	0.0004353	1	0.0004353	9.62	0.002
C	0.0155286	1	0.0155286	343.03	0.000
D	0.0003656	1	0.0003656	8.08	0.005
E	0.0000423	1	0.0000423	0.93	0.335
F	0.0000000	1	0.0000000	0.00	1.000
Interações					
AC	0.0069277	1	0.0069277	153.03	0.000
CD	0.0002206	1	0.0002206	4.87	0.028
DE	0.0002106	1	0.0002106	4.65	0.032

É possível verificar que a função de pertinência de saída e a função de pertinência de entrada possuem grande relevância estatística. A influência de cada fator e a interação entre eles pode ser verificada na Figura 8.1.

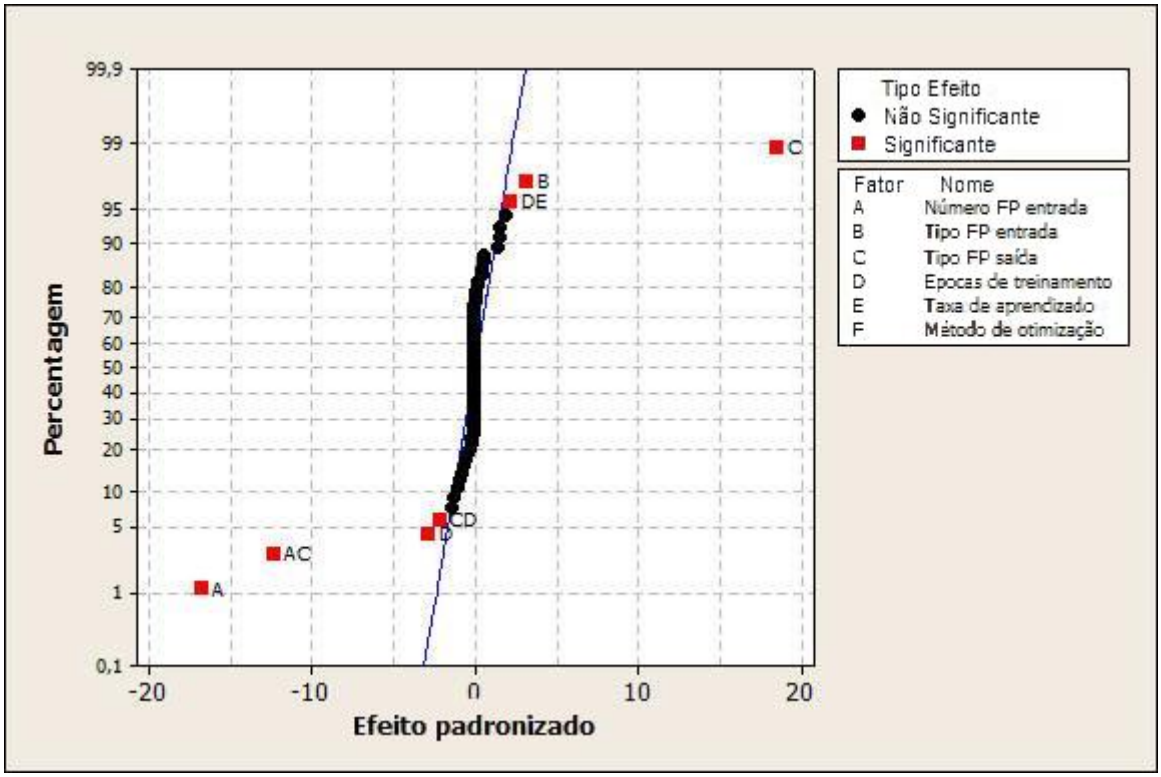

Figura 8.1 - Significância estatística de cada fator - Base de dados *Mackey-Glass*.

Na Figura 8.2 são apresentados os principais efeitos de cada fator sobre a resposta da rede neuro-difusa. Pode ser observado que o Método de Otimização utilizado no treinamento não possui influência na resposta da rede. O valor da Taxa de Aprendizagem Inicial (E) também possui pequena participação na resposta do sistema. Porém, os parâmetros Tipo de Função de Pertinência de Saída (C) e Número de Funções de Pertinência de Entrada (A) apresentaram grande influência sobre a resposta da rede. Uma análise mais profunda revelou que estatisticamente um maior número de épocas de treinamento pode auxiliar na generalização do modelo. Dentre as funções de pertinência de saída, a função de pertinência linear obteve um melhor desempenho do modelo, da mesma forma que um maior número de funções de pertinência de entrada. O tipo das funções de pertinência de entrada teve pouca influência sobre a variabilidade do modelo.

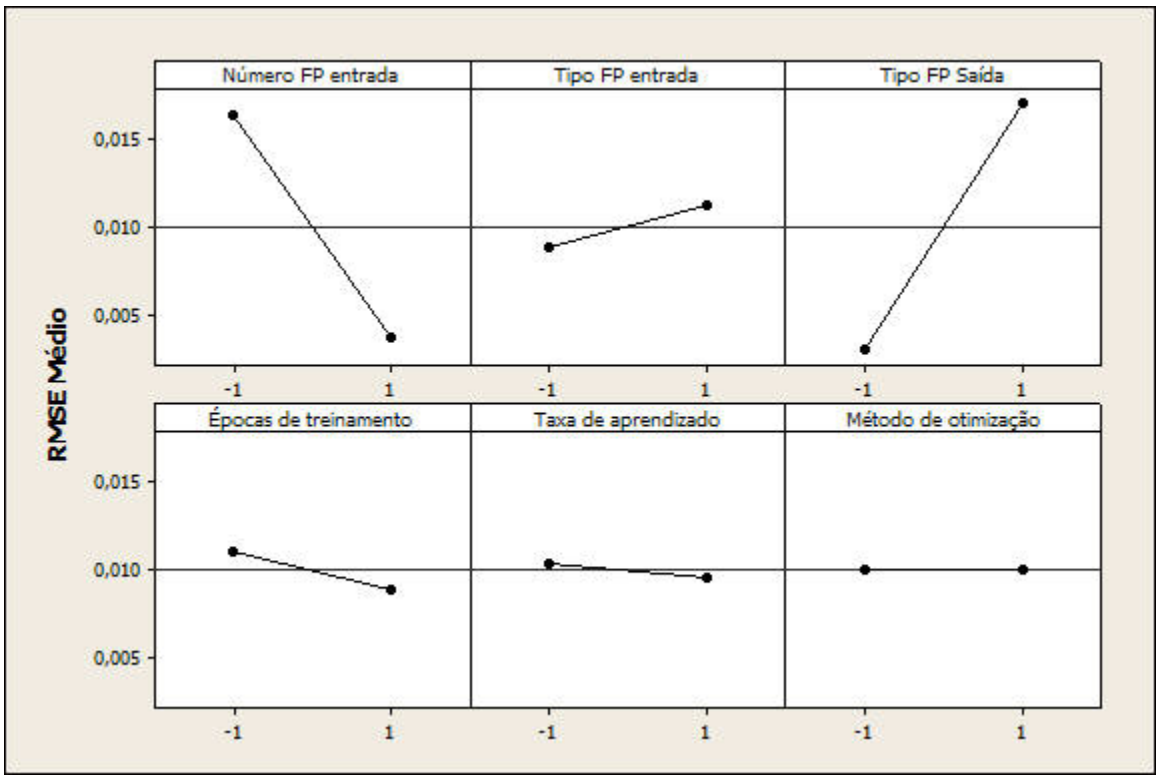

Figura 8.2 - Efeito causado por cada fator - Base de dados *Mackey-Glass*.

Nos experimentos com a rede neuro-difusa EFuNN, 5 fatores foram identificados com maior influência sobre a resposta da rede, são eles: Tipo da Função de Pertinência (B), explicando $\approx 68.84\%$ da variância total do sistema; Número de Funções de Pertinência (A), que explica $\approx 13.11\%$ da variância; *m-of-n*, que explica $\approx 0.92\%$ da variância; *Error Threshold* (C), explica $\approx 0.54\%$ da variância; e *Sensibility Threshold* (D), que corresponde a $\approx 0.53\%$ da variância nos resultados. Também foi identificada interação entre os fatores: Número de Funções de Pertinência (A) e Tipo da Função de Pertinência (B), correspondendo a $\approx 12.92\%$ da variância do sistema;

Tipo da Função de Pertinência (B) e *m-of-n* (E), com $\approx 0.90\%$ da variância; *Sensibility Threshlod* (D) e *Error Threshold* (C), correspondendo a $\approx 0.48\%$; Ocorreram ainda outras interações entre fatores, mas todas com uma variância menor que $\approx 0.50\%$ da variância total do sistema. A tabela ANOVA destes experimentos é apresentada na Tabela 8.4.

Tabela 8.4 - Resultados da tabela ANOVA para os experimentos com EFuNN – Base de dados *Mackey-Glass*.

	Soma Quadrática	Graus de Liberdade	Média Quadrática	F_0	Nível de significância
Fatores					
A	11.1949	1	11.1949	4136.83	0.000
B	58.7819	1	58.7819	21721.61	0.000
C	0.4507	1	0.4507	166.56	0.000
D	0.4572	1	0.4572	168.94	0.000
E	0.7855	1	0.7855	290.27	0.000
F	0.0006	1	0.0006	0.23	0.633
Interações					
AB	11.0328	1	11.0328	4076.93	0.000
AE	0.0466	1	0.0466	17.22	0.000
BC	0.2110	1	0.2110	77.98	0.000
BD	0.2060	1	0.2060	76.12	0.000
BE	0.7650	1	0.7650	282.71	0.000
CD	0.4106	1	0.4106	151.74	0.000
DE	0.0115	1	0.0115	4.24	0.041
ABE	0.0457	1	0.0457	16.87	0.000
BCD	0.1935	1	0.1935	71.50	0.000
CDE	0.0120	1	0.0120	4.45	0.036
ABDE	0.0094	1	0.0094	3.48	0.063

Como pôde ser verificado, no treinamento das redes EFuNN existe interação entre vários fatores (Figura 8.3). A interação mais relevante foi entre os fatores AB, a análise revelou que um número baixo de funções de pertinência associada à função de pertinência triangular pode resultar em modelos com melhor generalização. A interação entre os demais fatores foi estatisticamente relevante, mas com uma capacidade de explicar a variância do sistema muito pequena.

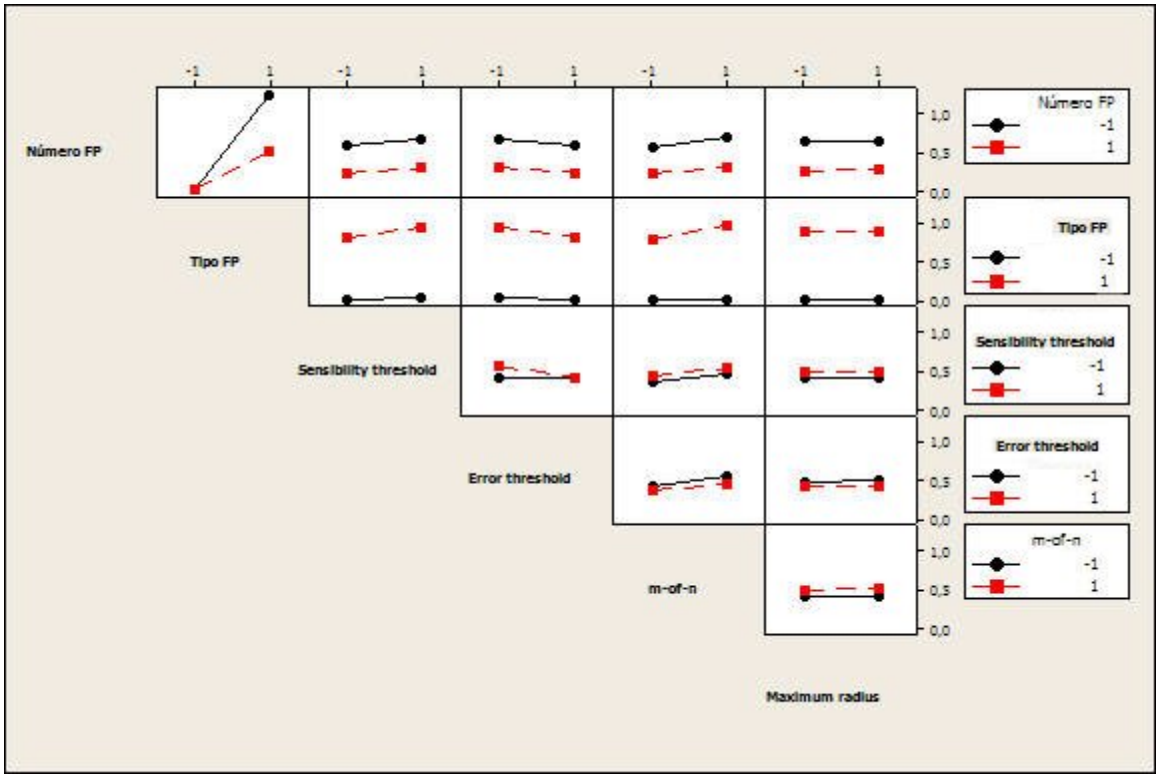

Figura 8.3 - Gráfico de interação entre os fatores da rede EFuNN - Base de dados *Mackey-Glass*.

Base de dados *Gas Furnace*

Nos experimentos com ANFIS, apresentados na Tabela 8.5, dois fatores foram identificados como estatisticamente relevantes no desempenho do sistema, são eles: o Tipo de Função de Pertinência de Saída (C), correspondendo a $\approx 22.83\%$ da variância dos resultados; e o Número de Funções de Pertinência de Entrada (A), correspondendo a $\approx 20.33\%$ da variância total das respostas. Várias interações entre fatores foram identificadas, porém como na maioria dos experimentos realizados, elas são entre os mesmos fatores, são elas: Número de Função de Pertinência de Entrada (A) e Tipo de Função de Pertinência de Saída (C), correspondendo a $\approx 14.65\%$ da variância. As demais interações podem ser visualizadas na Figura 8.4, porém cada uma delas corresponde a menos de 1% da variação do sistema.

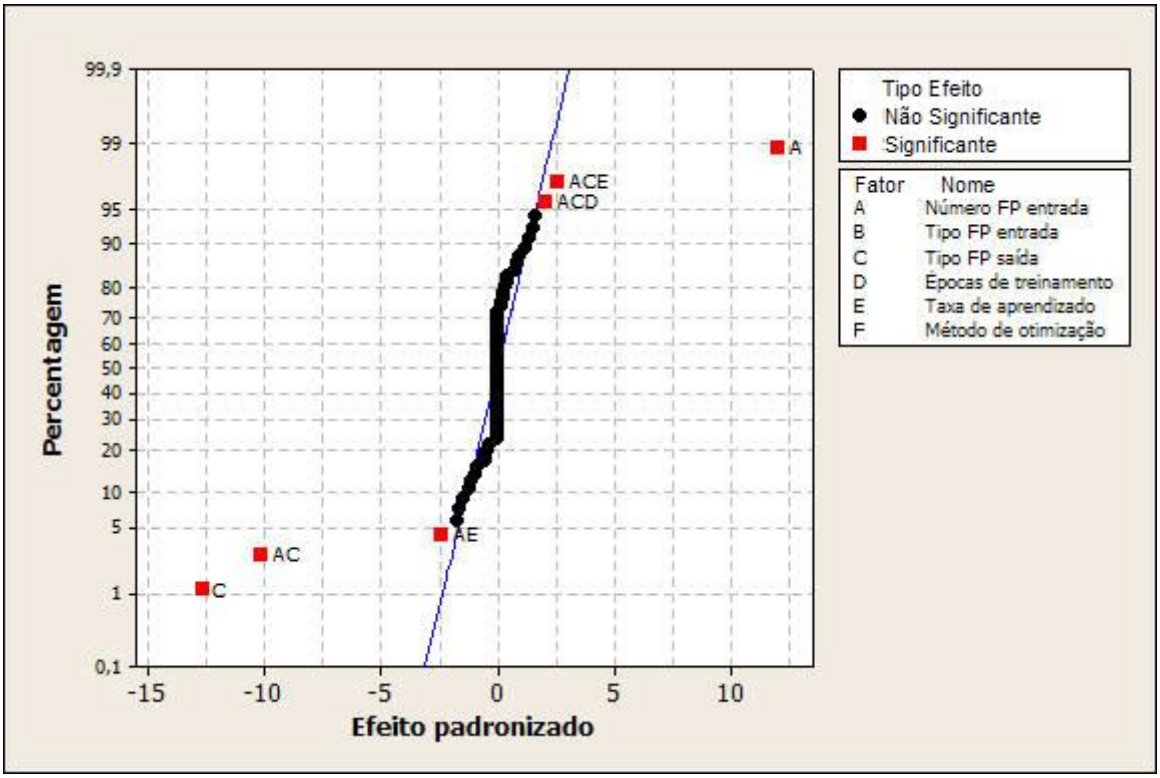

Figura 8.4 - Significância estatística de cada fator - Base de dados *Gas Furnace*.

Tabela 8.5 - Resultados da tabela ANOVA para os experimentos ANFIS – Base de dados *Gas Furnace*.

	Soma Quadrática	Graus de Liberdade	Média Quadrática	F_0	Nível de significância
Fatores					
A	8384.910000	1	8384.910000	143.32	0.000
B	72.2800000	1	72.2800000	1.24	0.267
C	9413.5600000	1	9413.5600000	160.90	0.000
D	49.0800000	1	49.0800000	0.84	0.361
E	133.3000000	1	133.3000000	2.28	0.132
F	0.0000000	1	0.0000000	0.00	1.000
Interações					
AC	6040.9800000	1	6040.9800000	103.26	0.000
AE	337.5300000	1	337.5300000	5.77	0.017
ACD	247.4400000	1	247.4400000	4.23	0.041
ACE	374.2600000	1	374.2600000	6.40	0.012

Nos experimentos com EFuNN, três fatores foram relevantes nas simulações: Tipo de Função de Pertinência (B), correspondendo a $\approx 58.48\%$ da variância; Número de Funções de Pertinência (A), correspondendo a $\approx 0.56\%$ da variação; e *m-of-n* (E) correspondendo a $\approx 4.00\%$ da variação dos resultados. A iteração entre os fatores representa pequena participação na variação total dos resultados. Os resultados das simulações e os efeitos de cada um dos fatores são apresentados na Tabela 8.6 e Figura 8.5.

Tabela 8.6 - Resultados da tabela ANOVA para os experimentos EFuNN – Base de dados *Gas Furnace*.

	Soma Quadrática	Graus de Liberdade	Média Quadrática	F_0	Nível de significância
Fatores					
A	10.8590000	1	10.8590000	5.56	0.019
B	1124.6350000	1	1124.6350000	575.67	0.000
C	0.8980000	1	0.8980000	0.46	0.498
D	0.0010000	1	0.0010000	0.00	0.985
E	77.0180000	1	77.0180000	39.42	0.000
F	0.0000000	1	0.0000000	0.00	1.000
Interações					
AB	11.0060000	1	11.0060000	5.63	0.018
AE	75.3070000	1	75.3070000	38.55	0.000
BE	40.6280000	1	40.6280000	20.80	0.000
ABE	74.9390000	1	74.9390000	38.36	0.000

A partir da Figura 8.5 pode ser verificado que os fatores *Sensibility Threshold* (D), *Error Threshold* (C) e *Maximus Radius* (F) possuem pouca ou quase nenhuma influência nos resultados. Os parâmetros mais influentes são o Tipo de Função de Pertinência (B) e o *m-of-n* (E).

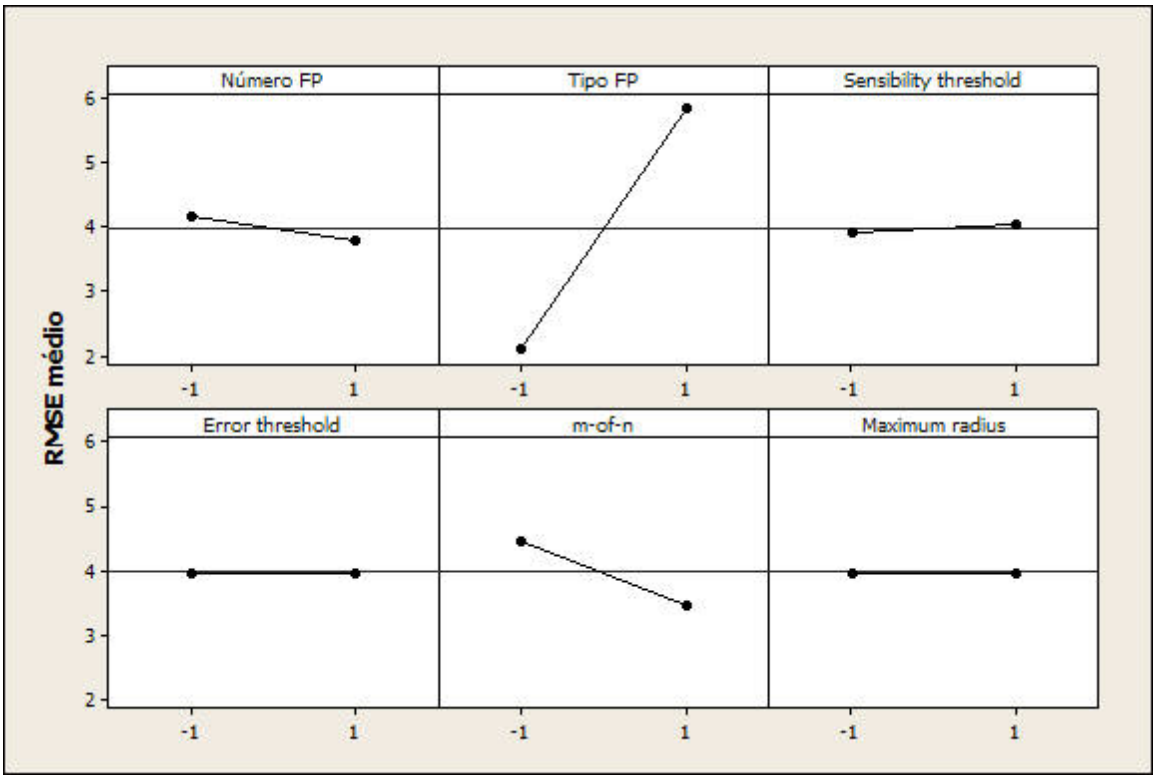

Figura 8.5 - Efeito causado por cada fator - Base de dados *Gas Furnace*.

Base de dados Íris de Fisher

Os fatores mais influentes nos experimentos com a rede ANFIS foram novamente o Número de Funções de Pertinência de Entrada (A) e o Tipo de Funções de Pertinência de Saída (C), correspondendo a $\approx 13.52\%$ e $\approx 8.63\%$ da variação do sistema, respectivamente. Foram identificadas também várias interações entre fatores, porém, somente algumas apresentam grande influência no desempenho do sistema, são elas: Número de Funções de Pertinência de Entrada (A), Tipo de Função de Pertinência de Saída (C), Taxa de Aprendizado Inicial (E) e Número de Épocas de Treinamento (D). Estas interações são apresentadas na Figura 8.6 e Tabela 8.7.

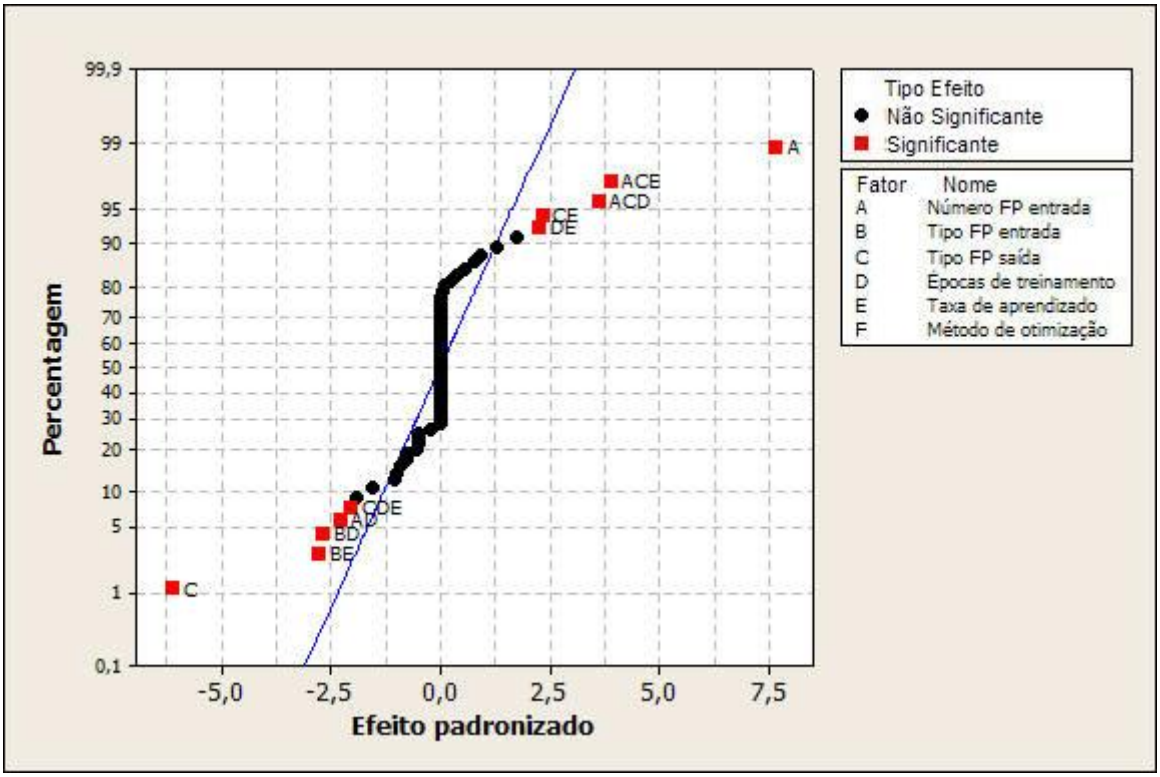

Figura 8.6 - Significância estatística de cada fator - Base de dados Íris de Fisher.

Tabela 8.7 - Resultados da tabela ANOVA para os experimentos ANFIS – Base de dados Íris de Fisher.

	Soma Quadrática	Graus de Liberdade	Média Quadrática	F_0	Nível de significância
Fatores					
A	85.3940000	1	85.3940000	58.53	0.000
B	1.3860000	1	1.3860000	0.95	0.331
C	54.5080000	1	54.5080000	37.36	0.000
D	0.0220000	1	0.0220000	0.02	0.901
E	0.4870000	1	0.4870000	0.33	0.564
F	0.0000000	1	0.0000000	0.00	1.000

Interações					
AD	7.6310000	1	7.6310000	5.23	0.023
BD	10.5720000	1	10.5720000	7.25	0.008
BE	11.2750000	1	11.2750000	7.73	0.006
CE	7.9280000	1	7.9280000	5.43	0.021
DE	7.4830000	1	7.4830000	5.13	0.024
ACD	19.0350000	1	19.0350000	13.05	0.000
ACE	22.0150000	1	22.0150000	15.09	0.000
CDE	6.0710000	1	6.0710000	4.16	0.042

Nos experimentos com a rede EFuNN, apresentados na Figura 8.7, Tabela 8.8 e Figura 8.8, três fatores foram identificados com maior relevância: Tipo de Função de Pertinência (B), correspondendo a $\approx 76.47\%$ da variação; Número de Funções de Pertinência (A), correspondendo a $\approx 10.12\%$ da variação dos resultados; e Taxa de Aprendizado Inicial (E), correspondendo a $\approx 0.39\%$ da variação do sistema.

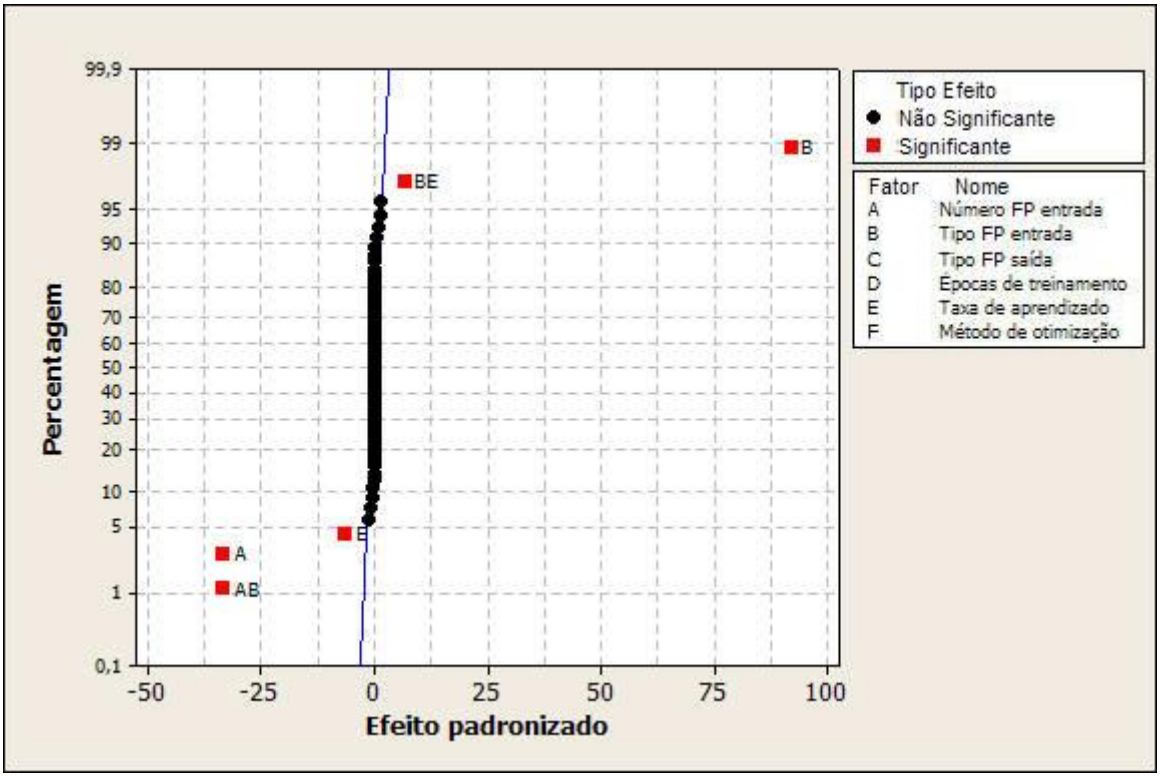

Figura 8.7 - Significância estatística de cada fator - Base de dados Íris de Fisher.

Tabela 8.8 - Resultados da tabela ANOVA para os experimentos EFuNN – Base de dados Íris de Fisher.

	Soma Quadrática	Graus de Liberdade	Média Quadrática	F_0	Nível de significância
Fatores					
A	4.3409400	1	4.3409400	1114.28	0.000
B	32.7930800	1	32.7930800	8417.69	0.000
C	0.0000000	1	0.0000000	0.00	1.000
D	0.0059300	1	0.0059300	1.52	0.218
E	0.1679800	1	0.1679800	43.12	0.000
F	0.0000100	1	0.0000100	0.00	0.968
Interações					
AB	4.3922100	1	4.3922100	1127.44	0.000
BE	0.1654300	1	0.1654300	42.47	0.000

Os fatores com maior interação são o *m-of-n* (E), Número e o Tipo de Funções de Pertinência (B). Porém somente a interação entre o Número (A) e Tipo de Função de Pertinência (B) possuem valores relevantes, correspondendo a $\approx 10.24\%$ da variação dos resultados.

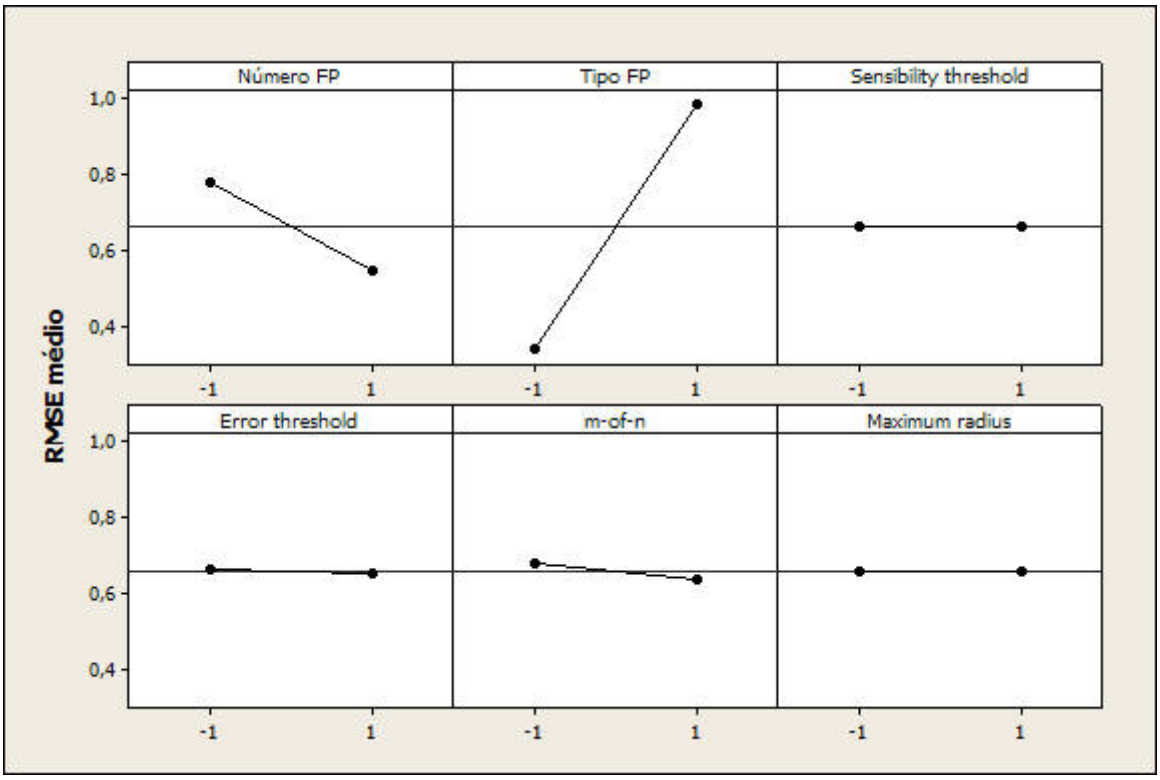

Figura 8.8 - Efeito causado por cada fator - Base de dados Íris de Fisher.

Base de dados Reconhecimento de Vinho

Nas simulações realizadas com ANFIS os fatores mais influentes foram: Número de Funções de Pertinência de Entrada (A), $\approx 13.52\%$ da variação; Tipo da Função de Pertinência de Saída (C),

$\approx 6.33\%$ da variação; e Tipo da Função de Pertinência de Entrada (B), correspondendo a $\approx 3.71\%$ da variação total do sistema. A interação mais representativa entre os fatores foi entre o Número de Funções de Pertinência de Entrada (A), o Tipo das Funções de Pertinência de Saída (C) e Taxa de Aprendizagem Inicial (E), correspondendo a $\approx 5.28\%$ da variação do sistema. Os efeitos em cada um dos fatores é apresentado na Figura 8.9 e na Tabela 8.9.

Tabela 8.9 - Resultados da tabela ANOVA para os experimentos ANFIS – Base de dados Reconhecimento de Vinho.

	Soma Quadrática	Graus de Liberdade	Média Quadrática	F_0	Nível de significância
Fatores					
A	73.2940000	1	73.2940000	47.53	0.000
B	31.0860000	1	31.0860000	20.95	0.001
C	54.0080000	1	54.0080000	39.36	0.001
D	0.0420000	1	0.0420000	0.06	0.906
E	0.3870000	1	0.3870000	0.43	0.664
F	0.0000000	1	0.0000000	0.00	1.000
Interações					
AB	0.7624300	1	0.7624300	3.61	0.009
BE	6.1833900	1	6.1833900	7.46	0.001
ACD	0.8453200	1	0.8453200	3.70	0.008
ACE	9.4341700	1	9.4341700	10.03	0.004

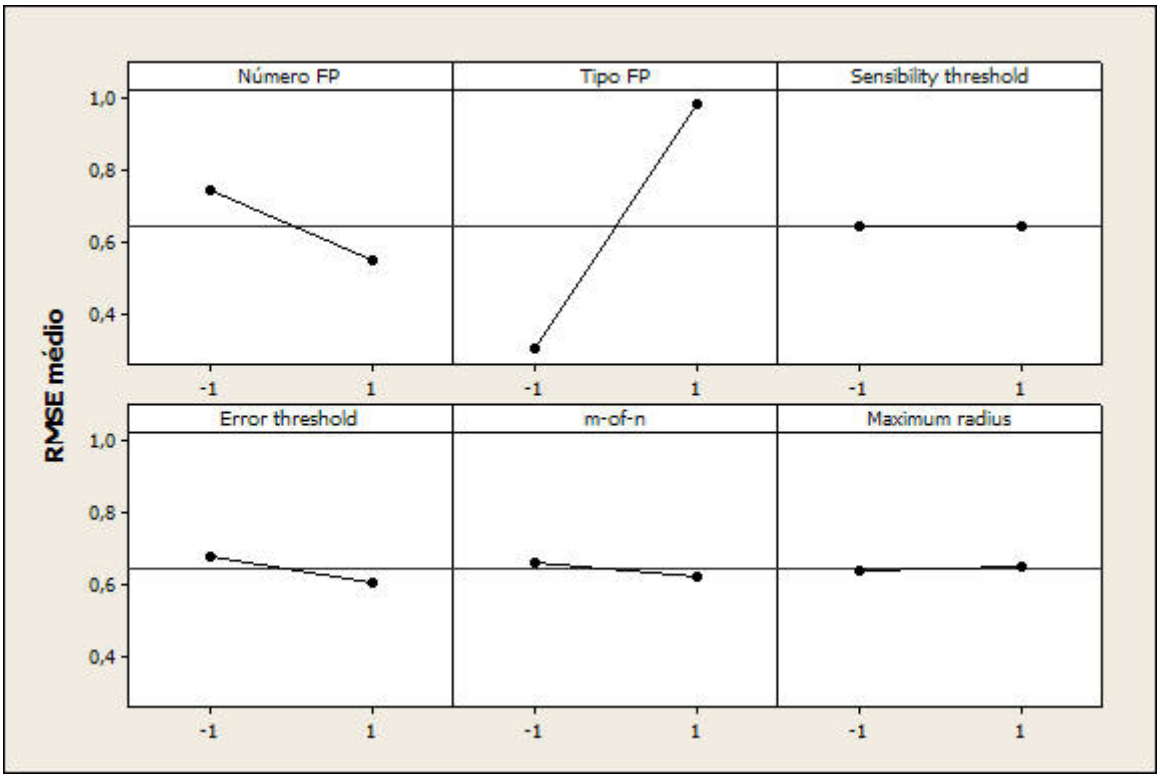

Figura 8.9 - Efeito causado por cada fator – Base de dados Reconhecimento de Vinho.

Nos experimentos com EFuNN, apresentados na Figura 8.10 e na Tabela 8.10, os fatores mais relevantes foram o Tipo de Função de Pertinência (B) - correspondendo a $\approx 66.11\%$ da variação dos resultados e o Número de Funções de Pertinência (A) - correspondendo a $\approx 5.50\%$ da variação. Os fatores, *Sensitivity Threshold* (D) e *m-of-n* (E), apesar de relevantes, representam pequena porcentagem da variação (menos que 1%).

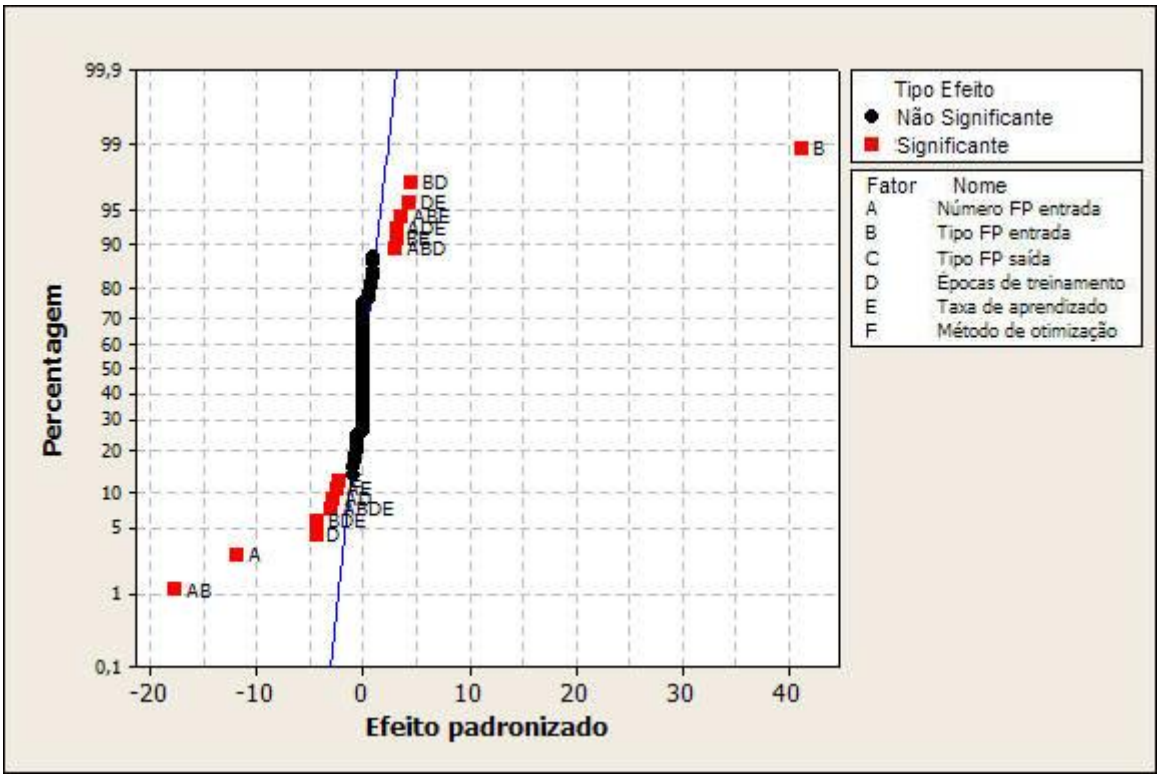

Figura 8.10 - Significância estatística de cada fator – Base de dados Reconhecimento de Vinho.

Tabela 8.10 - Resultados da tabela ANOVA para os experimentos EFuNN – Base de dados Reconhecimento de Vinho.

	Soma Quadrática	Graus de Liberdade	Média Quadrática	F_0	Nível de significância
Fatores					
A	3.0646800	1	3.0646800	141.39	0.000
B	36.7799800	1	36.7799800	1696.91	0.000
C	0.0000000	1	0.0000000	0.00	1.000
D	0.4341700	1	0.4341700	20.03	0.000
E	0.1174600	1	0.1174600	5.42	0.021
F	0.0108600	1	0.0108600	0.50	0.480
Interações					
AB	6.8624300	1	6.8624300	316.61	0.000
AD	0.1833900	1	0.1833900	8.46	0.004
AE	0.1453200	1	0.1453200	6.70	0.010

BD	0.4341700	1	0.4341700	20.03	0.000
BE	0.2027000	1	0.2027000	9.35	0.002
DE	0.4027300	1	0.4027300	18.58	0.000
ABD	0.1833900	1	0.1833900	8.46	0.004
ABE	0.2615000	1	0.2615000	12.06	0.001
ADE	0.2048000	1	0.2048000	9.45	0.002
BDE	0.4027300	1	0.4027300	18.58	0.000
ABDE	0.2048000	1	0.2048000	9.45	0.002

Foram identificadas várias interações entre fatores, como pode ser verificado na Figura 8.11, porém apenas a interação entre os parâmetros Número (A) e Tipo de Funções de Pertinência (B) representa grande variação nos resultados ($\approx 12.33\%$), os demais fatores possuem participação menor que $\approx 1\%$.

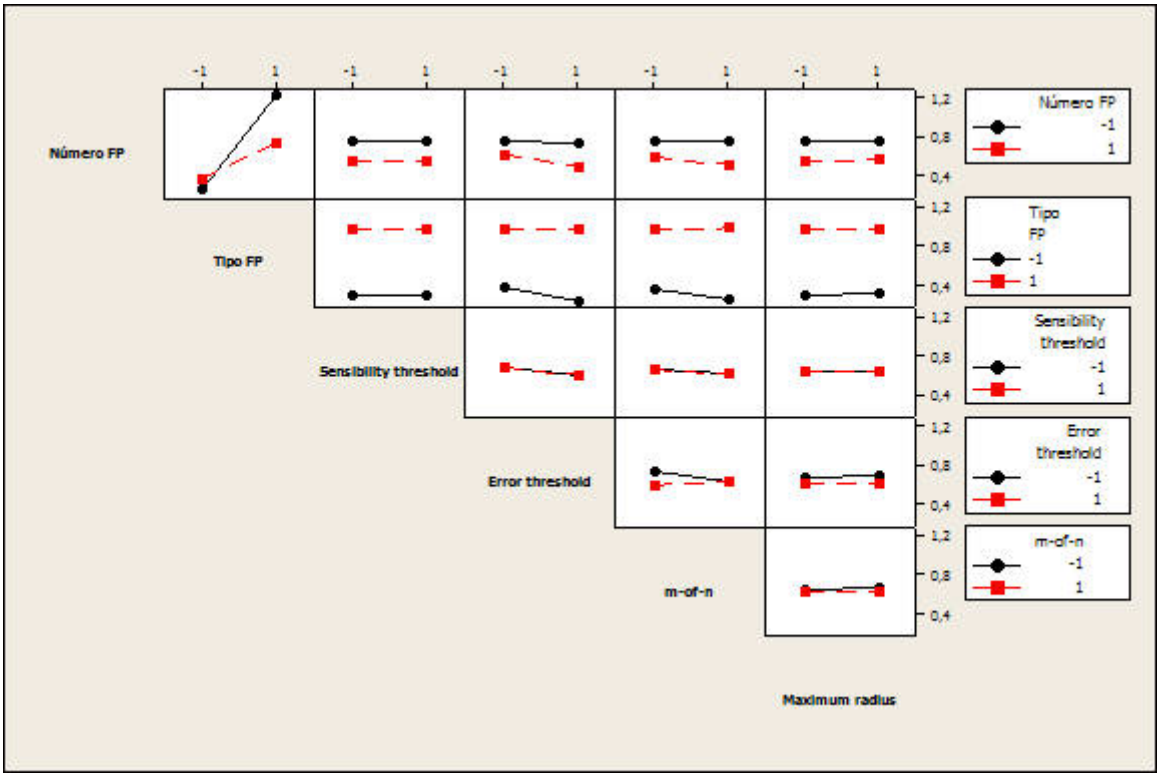

Figura 8.11 - Gráfico de interação entre os fatores – Base de dados Reconhecimento de Vinho.

Após a realização dos experimentos com os dois sistemas em várias bases dados diferentes, a expectativa de que o comportamento dos modelos variava pouco em relação a cada problema foi confirmada. Os fatores mais relevantes no desempenho da rede neuro-difusa ANFIS foram o Número de Função de Pertinência de Entrada (A) e o Tipo de Função de Pertinência de Entrada (B) e Saída (C). A interação mais importante é a existente entre os fatores Número de Funções de Pertinência de Entrada (A) e Tipo da Função de Pertinência de Saída (C). Como era esperado, foi confirmado que um número maior de épocas de treinamento pode ajudar na generalização do

modelo, porém não apresenta uma influência tão grande como se especulava. A Função de Pertinência de Saída linear apresenta melhores efeitos sobre a superfície de erro, porém o Tipo de Função de Pertinência de entrada representa pouco em relação à generalização do modelo.

Com a rede neuro-difusa EFuNN, o mesmo comportamento foi alcançado, alguns parâmetros representam quase que a totalidade da variação em seu desempenho. O fator que mais influenciou no desempenho do modelo foi o Tipo da Função de Pertinência utilizada, seguido de longe pelo Número de Funções de Pertinências. Sendo que a interação destes fatores representou praticamente a totalidade da variação na interação entre os parâmetros investigados. A grande influência do Tipo de Função de Pertinência no desempenho do modelo se deve, possivelmente, porque ele influencia diretamente na representação do problema, mais até que a adição de novas funções de pertinência. Os parâmetros *Maximum Radius*, *Initial Sensibility Threshold* e *Error Threshold* são utilizados para determinar se uma nova unidade de processamento será criada ou não, porém eles obtiveram diferentes influências nos experimentos. Na análise da variância, *Maximum Radius* apresentou pequena influência no desempenho do modelo. Se *Sensibility Threshold* é grande, o *radius* (*radius* = 1 - *Sensibility Threshold*) é muito pequeno e desta forma é difícil violar a área do *Maximum Radius*. O parâmetro *Maximum Radius* provavelmente só exerce influência se valores pequenos são usados em comparação com *Sensibility Threshold*.

8.2 Experimentos com a rede neural *Multi-layer Perceptron*

Os experimentos compreenderam, para cada topologia, *10* simulações com o algoritmo de treinamento. Cada simulação corresponde à média de *30* inicializações aleatórias dos pesos da rede neural artificial. Na Tabela 8.11 são apresentados os resultados do treinamento de uma rede neural artificial *Multi-layer Perceptron* (Rumelhart e Weigend, 1990) totalmente conectada, utilizando algoritmo de treinamento *Backpropagation* com momento (Rumelhart et al, 1986). A taxa de aprendizagem foi fixada em $\alpha = 0.001$ e o termo de momento em $\eta = 0.7$. O treinamento da rede era interrompido pelos critérios de parada GL_5, progresso de treinamento, com $P_5(t) < 0.1$, e número máximo de iterações $I_T = 5.000$. Estes valores foram definidos empiricamente em experimentos prévios. Os parâmetros avaliados foram: *(i)* Erro médio de classificação para o conjunto de teste; e *(ii)* o Erro Percentual Quadrático (SEP). Estes experimentos estão publicados em Zanchettin et al (2005).

Tabela 8.11 - Experimentos com a rede *Multi-layer Perceptron*.

Base de dados / Número de unidades escondidas	Nariz Artificial	Íris	Tireóide	Diabetes	*Mackey-Glass*
	Erro médio de classificação (%)				SEP
02	33.6296	19.0598	10.2000	-	4.2146
03	-	18.2051	-	-	-
04	17.8123	7.9487	9.2704	27.8819	1.4357
05	-	6.8376	-	-	-
06	14.1185	10.6838	-	30.2951	1.8273
07	-	8.9744	-	-	-
08	11.1136	-	13.1519	28.4201	1.9045
10	6.3086	-	7.3800	27.0833	1.5804
12	8.8667	-	7.3804	27.3264	2.3831
14	11.9704	-	7.4824	28.4549	2.7860
16	-	-	10.2537	-	-

A melhor performance da rede *Multi-layer Perceptron* sobre a base de dados do Nariz Artificial foi obtida pela topologia com *10* unidades de processamento na camada intermediária (uma arquitetura de rede com *90* conexões) e um erro médio de classificação de *6.31%*. Sobre a base de dados Íris de Fisher, o melhor resultado foi obtido pela topologia com *5* unidades na camada intermediária (com *32* conexões) e um erro médio de classificação de *6.84%*. Com a base de dados Tireóide o menor erro de classificação (*7.38%*) foi obtido pela topologia com *10* unidades de processamento escondidas (uma arquitetura utilizando *240* conexões). A rede totalmente conectada que apresentou o melhor desempenho sobre a base de dados Diabetes utilizou *10* unidades de processamento escondidas (contendo *72* conexões) e obteve um erro de classificação de *27.08%*. Para o problema de previsão *Mackey-Glass*, a melhor performance da rede *Multi-layer Perceptron* foi obtida pela topologia com *4* unidades de processamento escondidas (uma arquitetura de rede contendo *20* conexões) e um erro percentual quadrático de *1.43*.

8.3 Experimentos com a Metodologia que Integra *Simulated Annealing*, *Tabu Search* e *Backpropagation*

Os experimentos foram realizados para verificar o desempenho da metodologia que integra as técnicas *Simulated Annealing* e *Tabu Search* (TSa), proposta por Yamazaki (2004), em relação às técnicas isoladas. As simulações foram realizadas em cinco bases de dados diferentes. Estes experimentos estão publicados em Ludermir et al (2006).

Na realização das simulações a topologia máxima permitida para as redes neurais artificiais geradas no conjunto de dados do Nariz Artificial contém *6* unidades de processamento de entrada, *10* unidades escondidas e *3* unidades de saída ($N_1 = 6$, $N_2 = 10$, $N_3 = 3$ e o número máximo de conexões da rede ($N_{\max}$) é igual a *90*). Para as simulações com a base de dados Íris de Fisher a

topologia máxima da rede contém $N_1 = 4$, $N_2 = 5$, $N_3 = 3$ e $N_{\max} = 35$. Para a base de dados Tireóide a topologia máxima da rede contém $N_1 = 21$, $N_2 = 10$, $N_3 = 3$ e $N_{\max} = 240$. Nos experimentos com a base de dados Diabetes a topologia máxima possui $N_1 = 8$, $N_2 = 10$, $N_3 = 2$ e $N_{\max} = 100$. Para as simulações com a base de previsão *Mackey-Glass* a topologia máxima da rede possui $N_1 = 4$, $N_2 = 4$, $N_3 = 1$ e $N_{\max} = 20$. Em todas as topologias de rede os valores de N_1 e N_3 são definidos pelo problema, dados pelo número de atributos e classes da base de dados, respectivamente. Os valores para o número de unidades de processamento escondidas foram obtidos empiricamente com os experimentos da seção anterior.

Na Tabela 8.12 são apresentadas as performances médias dos algoritmos de otimização. Nesta tabela são descritos: *(i)* o erro de classificação do conjunto de testes; *(ii)* o número de unidades de processamento nas camadas de entrada e intermediária após a otimização; e *(iii)* o número de conexões da rede ao final da otimização, em relação à topologia máxima permitida.

Tabela 8.12 – Performance média das técnicas de otimização.

Base de dados \ Método		MLP	SA	TS	TSa
Nariz Artificial	Erro Class. (%)	6.3086	3.3689	3.2015	1.4244
	Entradas	6	5.9400	5.9667	5.8800
	Escondidos	10	7.8067	8.0667	7.0567
	Connec. (%)	90	35.3700	36.6333	29.1033
Íris de Fisher	Erro Class. (%)	6.8376	12.6496	12.4786	4.6154
	Entradas	4	2.8500	2.8767	2.7100
	Escondidos	5	2.7567	3.4867	2.6567
	Connec. (%)	32	8.3433	8.3000	7.7633
Tireóide	Erro Class. (%)	7.3800	7.3813	7.3406	7.3322
	Entradas	21	20.7700	20.7700	20.3700
	Escondidos	10	7.2267	7.4667	6.3900
	Connec. (%)	240	83.7300	86.1400	71.5467
Diabetes	Erro Class. (%)	27.0833	27.1562	27.4045	25.8767
	Entradas	8	7.7600	7.7800	7.5633
	Escondidos	10	5.2700	5.3700	4.5300
	Connec. (%)	72	30.3833	30.8167	25.5067

Mackey-Glass	SEP de Teste	1.4357	2.0172	0.8670	0.6847
	Entradas	4	3.6167	3.7967	3.4567
	Escondidos	4	1.9000	2.2700	1.8933
	Connec. (%)	20	9.6300	12.0700	8.5667

Nas simulações com a base de dados Nariz Artificial o erro de classificação obtido pela rede neural artificial foi de *1.42%* (a rede *Multi-layer Perceptron* totalmente conectada obteve um erro de classificação de *6.30%* - Tabela 8.11) utilizando um número médio de conexões de *29.10* (o número máximo permitido era de *90* conexões).

Na base de dados Íris de Fisher o erro de classificação obtido foi de *4.61%* (o erro era de *6.84%* em uma arquitetura totalmente conectada) e o número médio de conexões foi *7.73* (em um máximo de *32* em uma rede totalmente conectada). Na base de dados Tireóide o erro médio de classificação foi de *7.33%* (era de *7.38%* em uma rede totalmente conectada) e o número médio de conexões utilizada foi de *71.54* em *240* conexões possíveis para uma arquitetura totalmente conectada. Na base de dados Diabetes o erro médio de classificação foi de *25.87%* (uma arquitetura totalmente conectada obteve um erro médio de classificação de *27.08%*) utilizando em média *25.50* conexões (a arquitetura totalmente conectada possui *100* conexões).

Nos experimentos com o problema de previsão *Mackey-Glass*, o erro percentual quadrático (SEP) obtido foi de *0.68* (era *1.43* em uma arquitetura totalmente conectada), utilizando em média *8.56* conexões, ao invés das *50* conexões utilizadas em uma arquitetura totalmente conectada.

O teste estatístico *t-student*, com nível de confiança de *5%* (Mitchell, 1997), foi aplicado sobre os experimentos, a fim de confirmar a significância estatística das análises. O desempenho das técnicas *Simulated Annealing* e *Tabu Search* foram estatisticamente similares. A técnica que integra as heurísticas dos algoritmos *Tabu Search, Simulated Annealing* e *Backpropagation* foi estatisticamente superior às técnicas isoladas nas bases de dados Íris de Fisher e Nariz Artificial e similar nas demais bases de dados.

8.4 Experimentos com o Método Proposto que Integra *Simulated Annealing*, *Tabu Search*, Algoritmos Genéticos e *Backpropagation*

Experimentos com Arquitetura Fixa

As simulações com a técnica proposta (GaTSa) foram realizadas sobre a base de dados Nariz Artificial. Em estudos prévios com esta base de dados verificou-se que o melhor desempenho da rede *Multi-layer Perceptron* foi obtido com uma arquitetura com *6* unidades na camada de entrada, *4* unidades na camada intermediária e *3* unidades de processamento na camada de saída (Yamazaki

et al, 2002). Esta topologia foi fixada como arquitetura máxima nos experimentos de otimização realizados. Em todos os algoritmos investigados, os parâmetros de configuração foram mantidos em sua configuração padrão ou ajustados com base em experimentos prévios. Possivelmente os valores utilizados não são os valores ótimos para o problema, porém, o objetivo da simulação é demonstrar as potencialidades das técnicas e não a configuração ideal para o problema. Estes experimentos estão publicados em Zanchettin et al (2005a) e Zanchettin et al (2005b).

Na Tabela 8.13 são apresentados os desempenhos médios de cada uma das técnicas de otimização investigadas. Estes resultados foram obtidos com cada técnica na otimização do número de conexões e dos valores destas conexões entre as unidades de processamento de uma rede neural artificial *Multi-layer Perceptron*. Foram avaliados: *(i)* o Erro Percentual Quadrático (SEP) e o erro de classificação (Class) dos conjuntos de treinamento, validação e teste; *(ii)* número de iterações do algoritmo; *(iii)* número de conexões da rede neural artificial ao final da otimização; e *(iv)* o valor final da variável temperatura. Nesta tabela são apresentados os resultados médios de *10* simulações com cada uma das técnicas. Cada simulação compreendeu a *30* execuções diferentes do algoritmo.

Tabela 8.13 – Performance média das técnicas de otimização.

Técnica	Treinamento		Validação		Teste		Iterações	Conexões
	SEP	Class.	SEP	Class	SEP	Class		
TS	18.7420	5.4435	18.8640	5.8798	18.7460	5.3805	51	11.42
SA	19.6530	6.9144	19.7590	7.4721	19.6450	6.9331	715	11.77
GA	21.6640	15.8760	21.7280	16.5150	21.6550	15.9240	315	16.64
GaTSa	18.6900	3.5789	18.7600	3.8062	18.6860	3.5664	46	8.33
GaTSa + BP	4.7815	3.4051	2.4065	3.2237	2.1413	2.8684	46	8.33
BP	6.2537	8.9589	3.1464	7.5262	2.8367	6.3086	90	36

A técnica que combina as heurísticas de *Tabu Search*, *Simulated Annealing*, Algoritmos Genéticos e *Backpropagation* obteve os melhores resultados. Esta técnica estatisticamente superior mesmo quando não fez uso da busca local, para otimizar o valor das conexões da rede neural artificial. O erro médio de classificação obtido foi de 2.87%, com uma média de *8* conexões, de um total de *36* conexões possíveis para uma rede totalmente conectada. Utilizando uma rede totalmente conectada, a técnica de otimização local *Backpropagation* obteve um erro médio de 6.30%.

Nas Figuras Figura 8.12, Figura 8.13 e Figura 8.14 são apresentados gráficos que comparam o desempenho das técnicas investigadas. A técnica proposta obteve os melhores resultados, considerando o erro de classificação, número final de conexões da rede e a quantidade de iterações necessárias para a otimização da arquitetura.

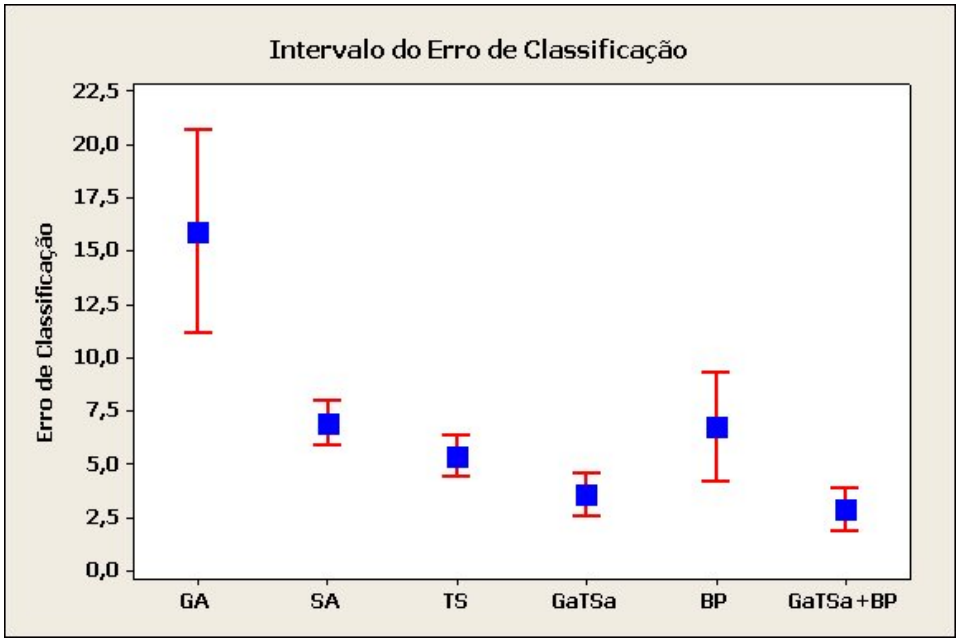

Figura 8.12 - Erro de classificação da rede.

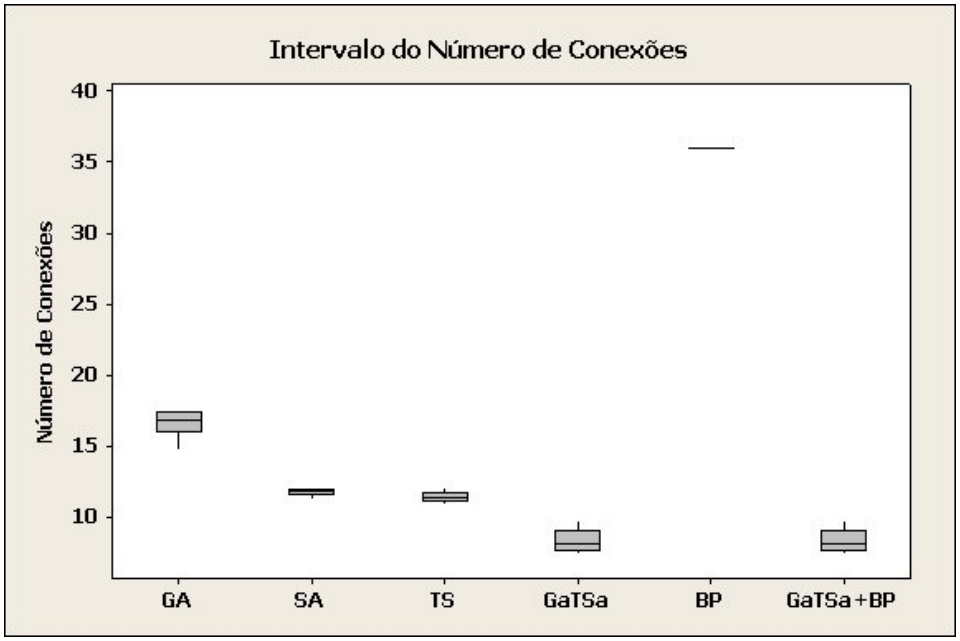

Figura 8.13 - Número de conexões.

As heurísticas de busca dos algoritmos *Tabu Search*, *Simulated Annealing* e Algoritmos Genéticos incorporam conhecimento específico sobre o domínio do problema. Estes algoritmos também suportam vários elementos não determinísticos que auxiliam a busca a escapar de mínimos locais. A integração proposta utiliza uma maior quantidade de informação sobre o domínio do problema e utiliza estas informações em praticamente todas as fases da busca. Isto se deve a integração das principais potencialidades das três heurísticas investigadas. Estas características

podem auxiliar na obtenção de melhores soluções para os problemas investigados, com menor tempo de busca, custo computacional e maior abrangência sobre o espaço de busca.

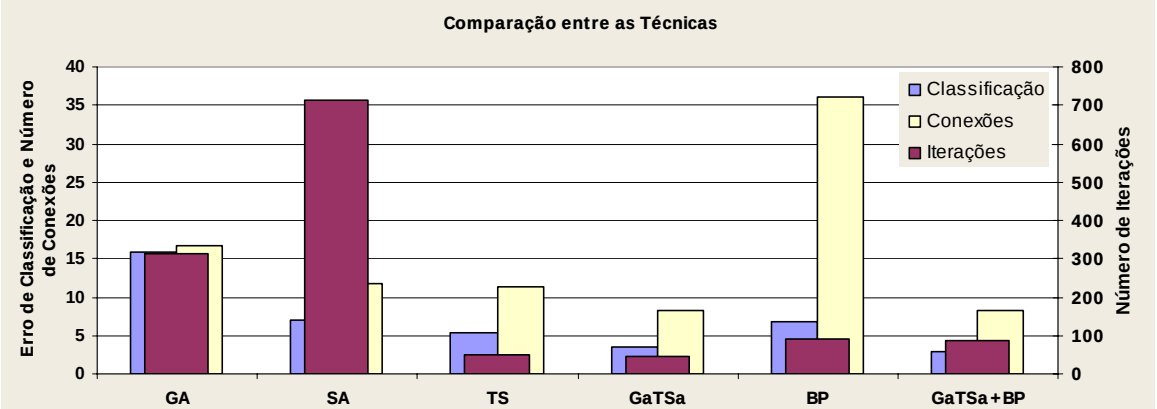

Figura 8.14 - Comparação entre as técnicas investigadas.

Experimentos com Arquitetura Variável

Nestes experimentos, o método de otimização proposto é comparado com as técnicas que o compõe e com outra metodologia de otimização que integra as técnicas *Simulated Annealing*, *Tabu Search* e *Backpropagation* – Tsa (Yamazaki, 2004). Além da integração das heurísticas dos algoritmos *Simulated Annealing*, *Tabu Search*, Algoritmos Genéticos e *Backpropagation*, a busca é realizada de maneira construtiva e baseada em poda. Desta forma, durante a busca pela melhor topologia é otimizado também o número de unidades de processamento na camada intermediária. Estes experimentos estão publicados em Zanchettin e Ludermir (2006a).

Para os experimentos com os algoritmos SA, TS, GA e TSa a topologia máxima da rede neural artificial no conjunto de dados do Nariz Artificial contém *6* unidades de processamento de entrada, *10* unidades escondidas e *3* unidades de saída ($N_1 = 6$, $N_2 = 10$, $N_3 = 3$ e o número máximo de conexões da rede ($N_{\max}$) é igual a *90*). Para as simulações com a base de dados Íris de Fisher a topologia máxima da rede contém $N_1 = 4$, $N_2 = 5$, $N_3 = 3$ e $N_{\max} = 35$. Para a base de dados Tireóide a topologia máxima da rede contém $N_1 = 21$, $N_2 = 10$, $N_3 = 3$ e $N_{\max} = 240$. Nos experimentos com a base de dados Diabetes a topologia máxima possui $N_1 = 8$, $N_2 = 10$, $N_3 = 2$ e $N_{\max} = 100$. E para as simulações de previsão com a base de dados *Mackey-Glass* a topologia máxima da rede possui $N_1 = 4$, $N_2 = 4$, $N_3 = 1$ e $N_{\max} = 20$. Em todas as topologias de rede os valores de N_1 e N_3 são definidos pelo problema, dados pelo número de atributos e classes da base

de dados, respectivamente. Os valores para o número de unidades de processamento escondidas foram obtidos empiricamente com os experimentos da Seção 8.2. Para as simulações com o algoritmo GaTSa, foram utilizados os mesmos valores para as variáveis N_1 e N_3, porém, o valor para N_2 é otimizado de forma construtiva pelo algoritmo juntamente com os pesos e conexões da rede neural artificial.

Na Tabela 8.14 são apresentados os desempenhos médios de cada uma das técnicas de otimização investigadas. Estes resultados foram obtidos com cada técnica na otimização do número de conexões e dos valores destas conexões entre as unidades de processamento de uma rede neural artificial *Multi-layer Perceptron*. Foram avaliados: *(i)* o Erro Percentual Quadrático (SEP) e o erro de classificação (Class) do conjunto de teste; *(ii)* número médio de unidades de processamento de entrada; *(iii)* número médio de unidades de processamento escondidas; e *(iv)* a porcentagem de conexões da rede neural artificial ao final da otimização[8] (em relação ao número máximo de conexões da topologia máxima permitida). Nesta tabela são apresentados os resultados médios obtidos em *10* simulações com cada uma das técnicas. Cada simulação compreendeu a *30* execuções diferentes do algoritmo.

Tabela 8.14 - Performance média das técnicas de otimização.

Base de dados \ Método		SA	TS	GA	TSa	GaTSa[9]
Nariz Artificial	Erro Class. (%)	3.3689	3.2015	13.3884	1.4244	0.7914
	Entradas	5.9400	5.9667	4.9667	5.8800	5.2267
	Escondidos	7.8067	8.0667	1.9333	7.0567	4.4867
	Connec. (%)	39.3000	40.7036	31.3556	32.3370	36.2254
Íris de Fisher	Erro Class. (%)	12.6496	12.4786	3.0341	4.6154	5.2256
	Entradas	2.8500	2.8767	2.4333	2.7100	3.3600
	Escondidos	2.7567	3.4867	1.5667	2.6567	4.1333
	Connec. (%)	26.0728	25.9375	22.9792	24.2603	31.8538
Tireóide	Erro Class. (%)	7.3813	7.3406	7.2850	7.3322	7.1509
	Entradas	20.7700	20.7700	12.8333	20.3700	7.1233
	Escondidos	7.2267	7.4667	2.4000	6.3900	2.0833
	Connec. (%)	34.8875	35.8916	14.3902	29.8111	12.6400

[8] Nos experimentos com GaTSa a média do número de conexões foi computado em relação à topologia máxima que a rede gerou, em vez de ser calculada em relação à topologia máxima da arquitetura fixa (como nos outros modelos). Esta pareceu ser a abordagem mais justa, porém, de certa forma prejudicou o modelo, porque na maioria das vezes este gerou arquiteturas menores que a topologia máxima permitida.

[9] Nestes experimentos a função de custo utilizada foi à média entre o erro de classificação e o número de conexões da rede neural artificial otimizada.

Diabetes	Erro Class. (%)	27.1562	27.4045	25.9948	25.8767	27.0615
	Entradas	7.7600	7.7800	4.9667	7.5633	1.5100
	Escondidos	5.2700	5.3700	1.9333	4.5300	1.2000
	Connec. (%)	30.3833	30.8167	18.7033	25.5067	9.0975
Mackey-Glass	SEP de Teste	2.0172	0.8670	0.6542	0.6847	0.7300
	Entradas	3.6167	3.7967	2.2667	3.4567	1.0533
	Escondidos	1.9000	2.2700	1.3667	1.8933	1.0100
	Connec. (%)	19.2600	24.1400	15.9533	17.1334	11.4923

Em grande parte das simulações a melhor performance na otimização das arquiteturas de redes neurais artificiais foi obtida pelo método proposto. Na base de dados do Nariz Artificial o erro médio de classificação ficou em torno de *0.79%* (em uma rede totalmente conectada este erro era de *6.30%*) e a porcentagem média do número de conexões foi de somente *36%* do número máximo de conexões permitidas. Na Figura 8.15 são apresentados os desempenhos médios de classificação e número de conexões obtidos por cada um dos algoritmos na base de dados Nariz Artificial.

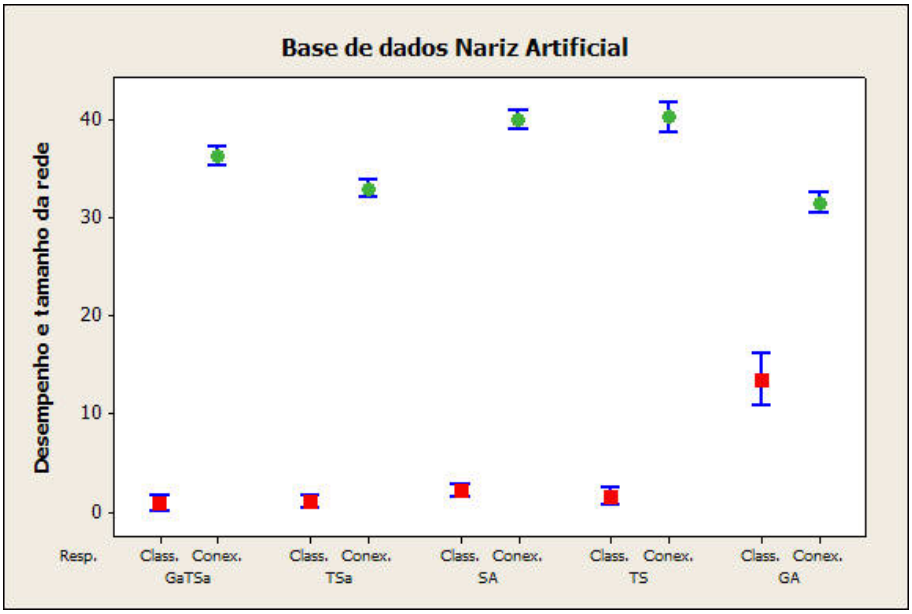

Figura 8.15 - Desempenho dos algoritmos sobre a base de dados Nariz Artificial.

Com a base de dados Íris de Fisher, o melhor erro de classificação foi obtido pelo método Algoritmos Genéticos, um erro de *3.03%* (*6.84%* em uma rede totalmente conectada) utilizando somente *22.98%* das conexões permitidas. Na Figura 8.16 são apresentados os desempenhos médios de classificação e número de conexões obtidos por cada um dos algoritmos sobre esta base de dados.

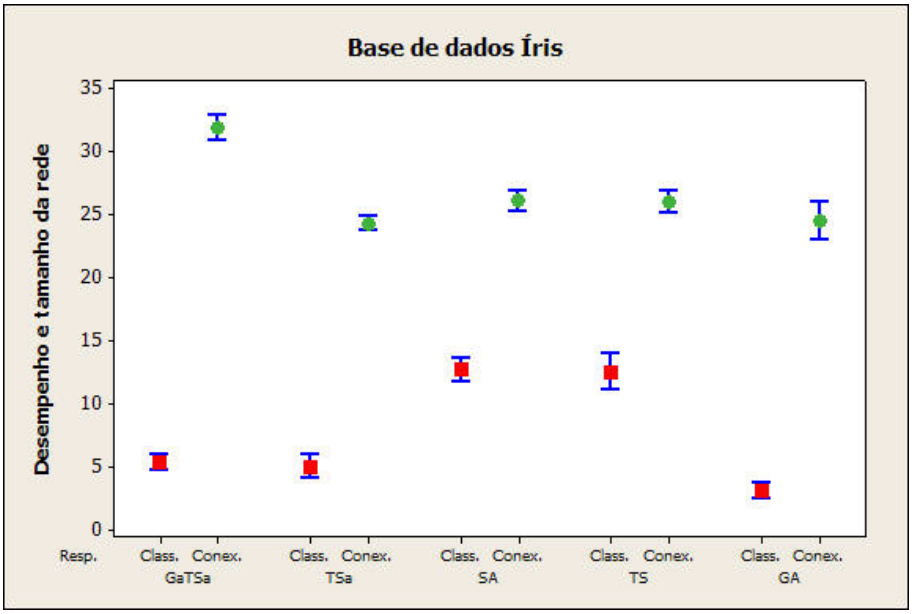

Figura 8.16 - Desempenho dos algoritmos sobre a base de dados Íris de Fisher.

Nas simulações com a base de dados Tireóide o erro médio de classificação foi de *7.15%* (*7.38%* em uma rede *Multi-layer Perceptron* totalmente conectada) e a porcentagem média de conexões foi de somente *12.64%* do total de conexões de uma arquitetura totalmente conectada. Na Figura 8.17 são apresentados os desempenhos médios de classificação e número de conexões obtidos por cada um dos algoritmos sobre a base de dados Tireóide.

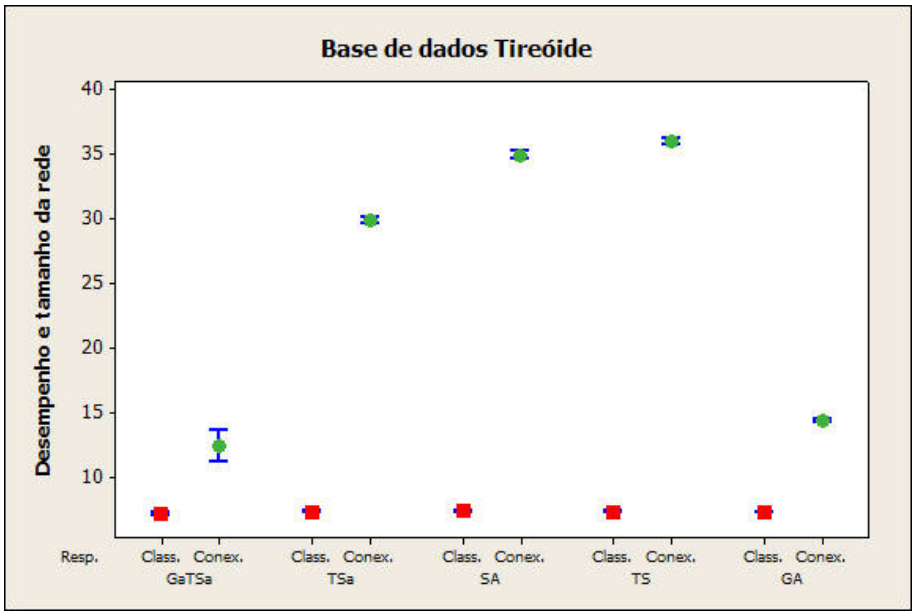

Figura 8.17 - Desempenho dos algoritmos sobre a base de dados Tireóide.

Para a base de dados Diabetes o erro médio de classificação foi de *27.06%* (pouco abaixo do desempenho da arquitetura totalmente conectada com *27.08%*), porém utilizando somente *9.10%* das conexões de uma rede totalmente conectada. Na Figura 8.18 são apresentados os desempenhos médios de classificação e número de conexões obtidos por cada um dos algoritmos sobre a base de dados Diabetes.

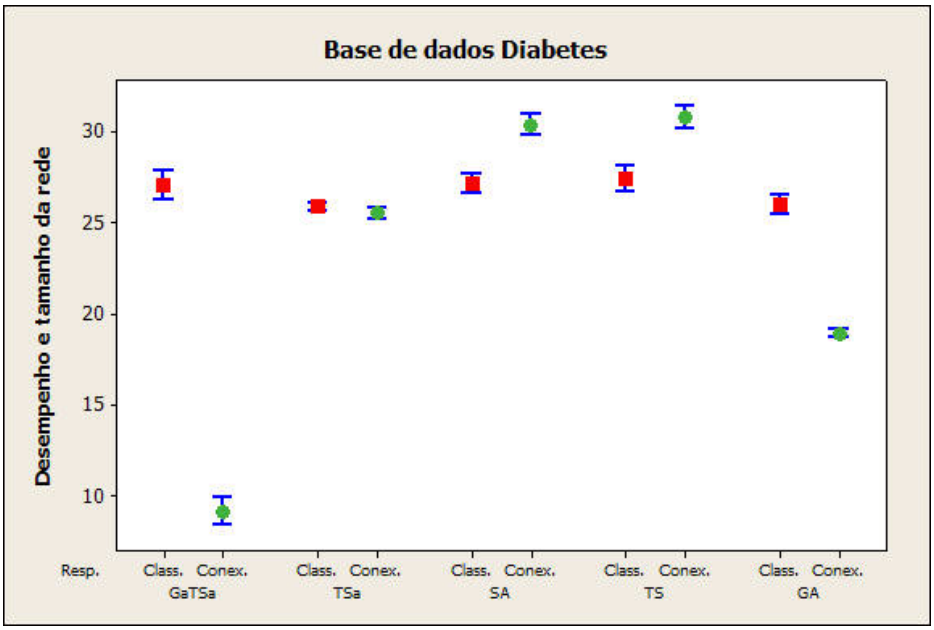

Figura 8.18 - Desempenho dos algoritmos sobre a base de dados Diabetes.

Nas simulações com o problema de previsão *Mackey-Glass*, o erro percentual quadrático obtido foi de *0.73* (*1.43* em uma rede totalmente conectada) utilizando uma porcentagem média de *11.49%* das conexões de uma arquitetura totalmente conectada. Na Figura 8.19 são apresentados os desempenhos médios de classificação e número de conexões obtidos por cada um dos algoritmos sobre a base de dados *Mackey-Glass*.

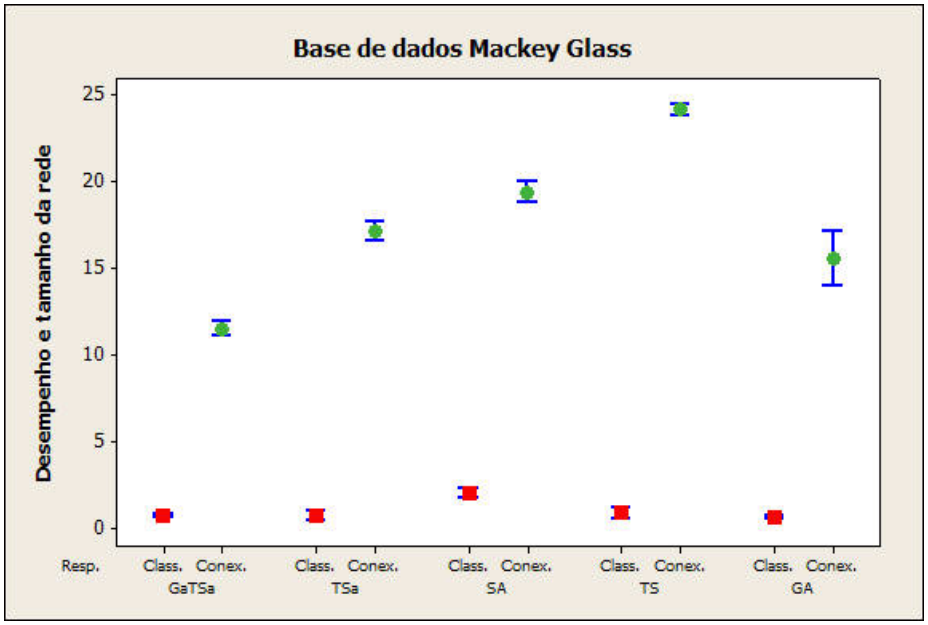

Figura 8.19 - Desempenho dos algoritmos sobre a base de dados *Mackey-Glass*.

Dentre as técnicas de busca isoladas, pode ser destacada a superioridade nos resultados obtidos pela técnica *Tabu Search* sobre *Simulated Annealing* em alguns casos. De certa forma isto se deve ao método de busca empreendido pelas técnicas, a cada iteração *Tabu Search* avalia *20* novas soluções, assim tem maiores chances de encontrar melhores soluções no espaço de busca. O método *Simulated Annealing* avalia somente uma solução a cada iteração, desta forma precisa de mais iterações para encontrar uma boa solução. O número máximo de iterações para o algoritmo *Simulated Annealing* foi de *1.000* iterações, porém esta quantidade pode não ter sido suficiente para alguns problemas com superfícies de busca mais complexas. Em contraste com *Simulated Annealing*, *Tabu Search* utilizou um máximo de *100* iterações nas simulações realizadas.

O método proposto obteve melhor desempenho, na maioria dos experimentos, na otimização do número de conexões da rede neural artificial utilizada. Como pode ser visto a combinação das heurísticas proposta realiza uma melhor exploração na superfície de busca do problema abordado, gerando arquiteturas de rede com um pequeno número de conexões e alta performance de generalização.

O teste estatístico *t-student*, com nível de confiança de *5%* (Mitchell, 1997), foi aplicado sobre os experimentos, a fim de confirmar a significância estatística das análises. O método GaTSa foi estatisticamente superior na otimização do número de entradas da rede neural artificial. Na otimização do número de conexões na camada intermediária todos os métodos tiveram desempenho estatisticamente similar. Se for considerado no teste o erro de classificação obtido pelas redes otimizadas, os métodos obtiveram desempenho estatisticamente similar nas bases de dados Tireóide, Diabetes e *Mackey-Glass*. O método proposto obteve desempenho estatisticamente

superior sobre a base de dados Nariz Artificial e o método Algoritmos Genéticos foi estatisticamente superior na base de dados Íris de Fisher.

Experimentos Verificando a Influência da Função de Custo

Os experimentos desta seção foram realizados para investigar a influência das funções de custo no desempenho do método proposto. As funções de custo são responsáveis por avaliar a qualidade de cada uma das soluções geradas pelo algoritmo e direcionar a busca para um possível ponto de ótimo (global ou local). Foram investigadas cinco funções de custo com características totalmente distintas: *(i)* média entre o erro de classificação e a percentagem de conexões utilizadas pela rede neural artificial (Média); *(ii)* média ponderada entre o erro de classificação, porcentagem de conexões e porcentagem de unidades escondidas utilizadas pela rede neural artificial (MP); *(iii)* um mecanismo inspirado na técnica *weight-decay* (WD); *(iv)* uma estratégia de otimização multiobjetivo (MO); e *(v)* a combinação das estratégias *weight-decay* e otimização multiobjetivo (MO+WD). Estes experimentos estão publicados em Zanchettin e Ludermir (2006) e Zanchettin e Ludermir (2007b).

Os experimentos com GaTSa foram realizados com arquitetura fixa. Esta estratégia foi utilizada para melhor controlar as simulações, visto que a adição de novas unidades de processamento na camada intermediária não é realizada de forma determinística. A topologia máxima permitida para as redes neurais artificiais geradas no conjunto de dados do Nariz Artificial contém 6 unidades de processamento de entrada, *10* unidades escondidas e *3* unidades de saída ($N_1 = 6$, $N_2 = 10$, $N_3 = 3$ e o número máximo de conexões da rede ($N_{\max}$) é igual a *90*). Para as simulações com a base de dados Íris de Fisher a topologia máxima da rede contém $N_1 = 4$, $N_2 = 5$, $N_3 = 3$ e $N_{\max} = 35$. Para a base de dados Tireóide a topologia máxima da rede contém $N_1 = 21$, $N_2 = 10$, $N_3 = 3$ e $N_{\max} = 240$. Nos experimentos com a base de dados Diabetes a topologia máxima possui $N_1 = 8$, $N_2 = 10$, $N_3 = 2$ e $N_{\max} = 100$. Para as simulações com a base de previsão *Mackey-Glass* a topologia máxima da rede possui $N_1 = 4$, $N_2 = 4$, $N_3 = 1$ e $N_{\max} = 20$. Em todas as topologias de rede os valores de N_1 e N_3 são definidos pelo problema, dados pelo número de atributos e classes da base de dados, respectivamente. Os valores para o número de unidades de processamento escondidas foram obtidos empiricamente com os experimentos da Seção 8.2.

Na Tabela 8.15 são apresentados os resultados médios de *10* simulações com cada uma das funções de custo. Cada simulação compreendeu a *30* execuções diferentes do algoritmo.

Nesta tabela são descritos: *(i)* erro percentual quadrático (SEP) e o erro de classificação do conjunto de testes; *(ii)* o número de unidades de processamento nas camadas de entrada e intermediária da rede após a otimização; e *(iii)* o número de conexões da rede ao final da otimização, em relação à topologia máxima permitida.

Tabela 8.15 – Performance média das funções de custo nas simulações.

Base de dados \ Função de Custo		Média	Média Ponderada	*Weight-Decay*	Multiobjetivo	*Weight-Decay* + Multiobjetivo
Nariz Artificial	Class. (%)	11.8595	11.0807	7.5462	7.0407	12.6237
	Entradas	5.9967	6.0000	5.9233	5.9967	5.9867
	Escondidos	9.1533	8.5133	7.5600	8.4833	8.6200
	Connec. (%)	50.2400	53.3834	33.2433	49.6500	42.0467
Íris de Fisher	Class. (%)	6.1197	5.4615	6.9316	4.2735	3.9829
	Entradas	3.5667	3.8367	3.8167	3.8567	3.1467
	Escondidos	3.7967	3.1900	4.4367	3.5500	3.0767
	Connec. (%)	13.3867	16.1367	18.1167	15.8600	9.8000
Tireóide	Class. (%)	7.1024	7.1798	6.8196	6.9270	6.8609
	Entradas	20.6733	20.9300	20.7800	20.9800	20.9767
	Escondidos	7.1833	3.5633	7.8067	8.5233	8.5167
	Connec. (%)	83.8800	99.3200	91.6700	114.6667	115.0933
Diabetes	Class. (%)	28.4583	28.4323	25.7552	28.2639	25.8542
	Entradas	7.7367	7.9267	7.7767	7.9700	7.9667
	Escondidos	5.6100	2.3067	5.2000	4.4133	6.3100
	Connec. (%)	31.8433	38.1833	31.7433	43.2700	42.0367
Mackey-Glass	SEP de Teste	0.6215	0.8070	0.2721	0.5745	0.6293
	Entradas	1.1067	1.4100	1.0533	2.0933	1.7867
	Escondidos	1.0000	1.0000	1.0067	1.0100	1.1800
	Connec. (%)	2.1533	2.5433	2.0833	3.6300	3.3233

Na base de dados Nariz Artificial os melhores resultados de classificação foram obtidos pela abordagem multiobjetivos e a melhor otimização da arquitetura da rede foi obtida pela abordagem *weight-decay*. Na Figura 8.20 são apresentados os resultados obtidos neste experimento. Nesta figura pode ser verificada a presença de alguns *outliers* nos resultados – representados no gráfico por asteriscos.

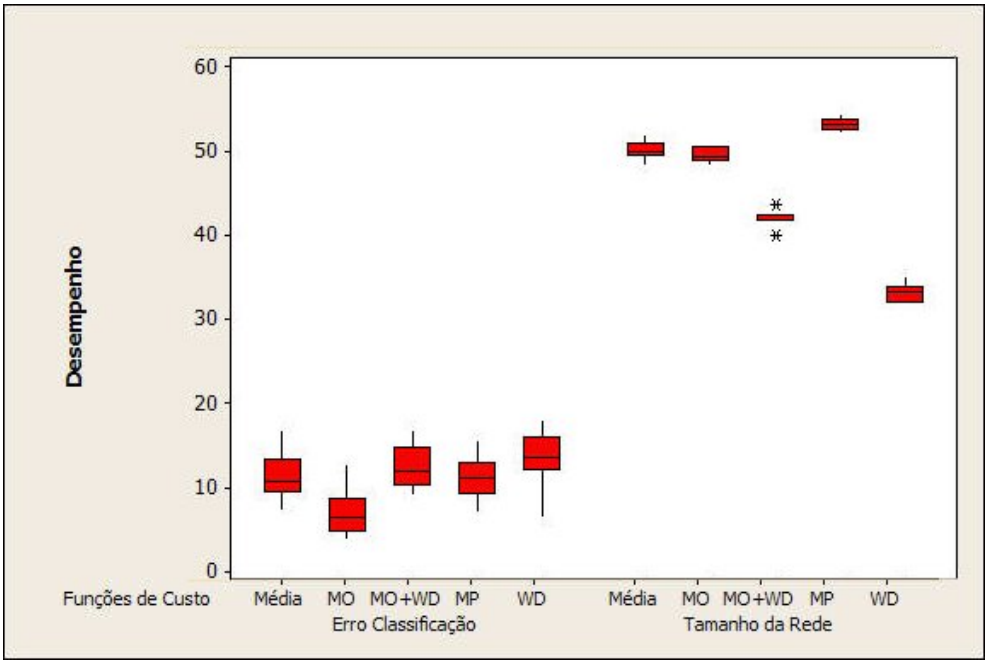

Figura 8.20 - Experimentos sobre a base Nariz Artificial.

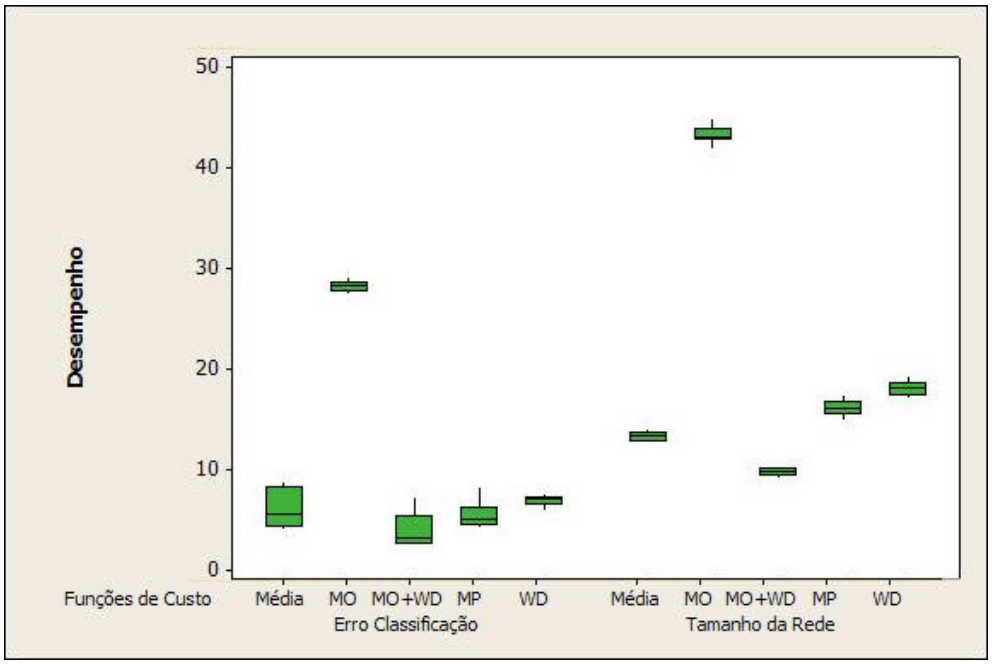

Figura 8.21 - Experimentos sobre a base Íris de Fisher.

Na base de dados Íris de Fisher a combinação de *weight-decay* e otimização multiobjetivos apresentaram a melhor performance média, conforme demonstrado na Figura 8.21.

A função de custo utilizando a abordagem *weight-decay* apresentou a melhor performance nas bases de dados Tireóide, Diabetes e Mackey-Glass. Estes resultados são apresentados na Figura 8.22, Figura 8.23 e Figura 8.24. A melhor performance das redes neurais artificiais otimizadas foi

obtida pelo algoritmo de otimização utilizando a abordagem *weight-decay*. Este resultado demonstra a capacidade do método em restringir as características das redes geradas. Durante o processo de otimização foram favorecidas arquiteturas de rede que produzissem superfícies de erro homogêneas. A função de custo *weight-decay* possui a capacidade de obter topologias com valores de pesos menores (tendendo a zero), pois valores muito grandes podem produzir grande variabilidade na saída da rede e modelarem ruídos.

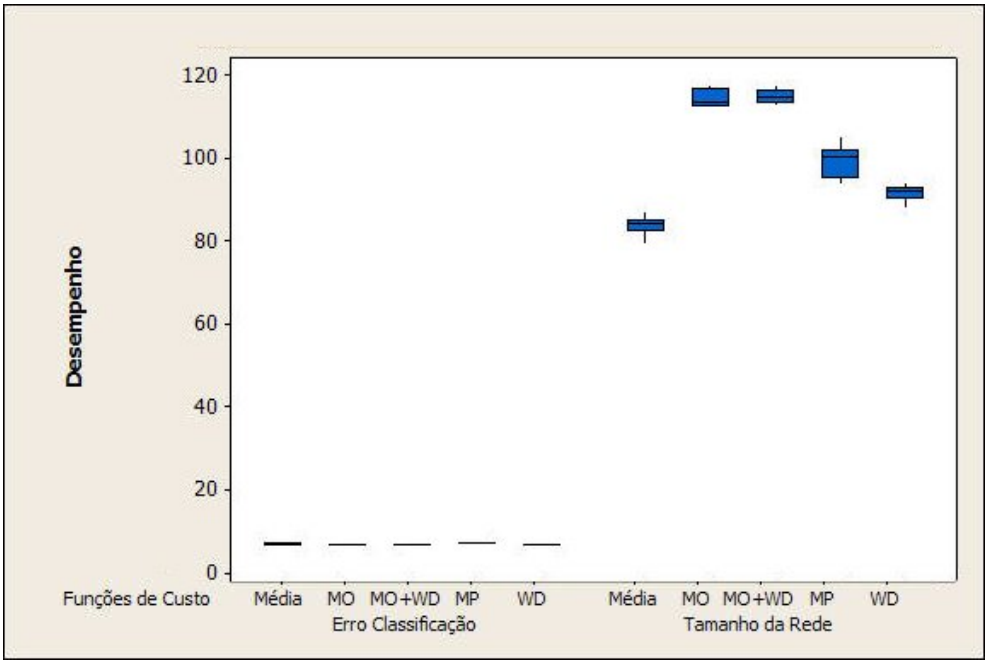

Figura 8.22 - Experimentos sobre a base Tireóide.

O uso da abordagem multiobjetivo nos operadores genéticos apresentou bons resultados em algumas bases de dados, porém também obteve resultados muito ruins em outros problemas investigados. Verificou-se que na abordagem implementada o grande problema foi à construção do *ranking* de Pareto. Possivelmente o algoritmo utilizado para verificar a não-dominância em um conjunto de possíveis soluções não foi eficiente. A maioria dos algoritmos tradicionais apresenta sérios problemas de degradação de performance com o aumento do tamanho da população e número de funções objetivo (Coello, 1996). Nos experimentos realizados a abordagem multiobjetivo gerou também vários resultados *outliers*, estes resultados prejudicaram o desempenho médio da técnica e degradaram também os resultados da combinação de *weight-decay* e otimização multiobjetivo. Outro possível problema foi na escolha da melhor solução quando existiam várias soluções com mesma posição no *ranking* de Pareto. Neste caso foi escolhida a solução com menor erro de classificação. Porém, não se tem garantia que esta solução possua um

pequeno número de conexões. Possivelmente a melhor solução teria sido a escolha da solução com melhor média entre o erro de classificação e o número de conexões.

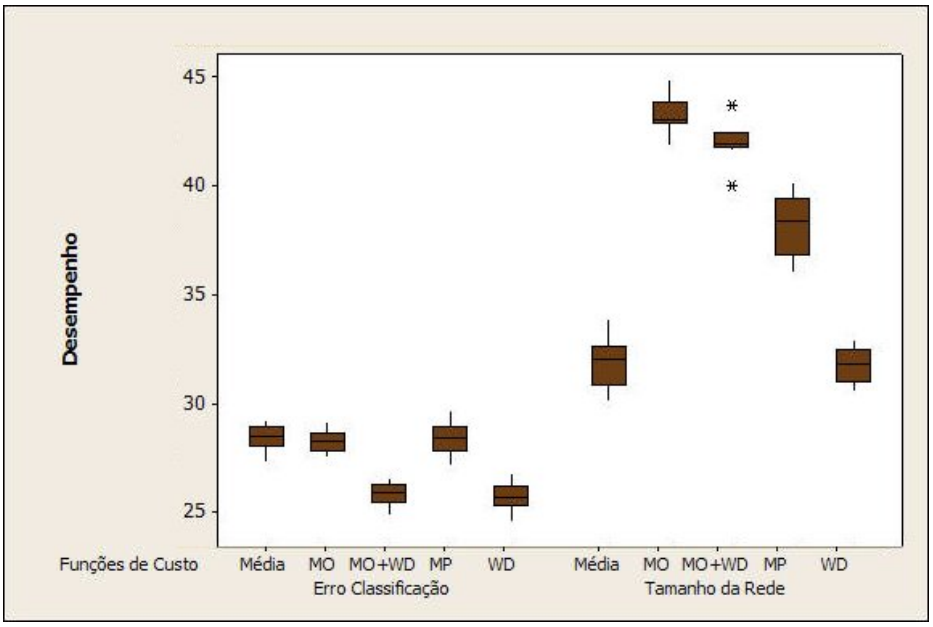

Figura 8.23 - Experimentos sobre a base Diabetes.

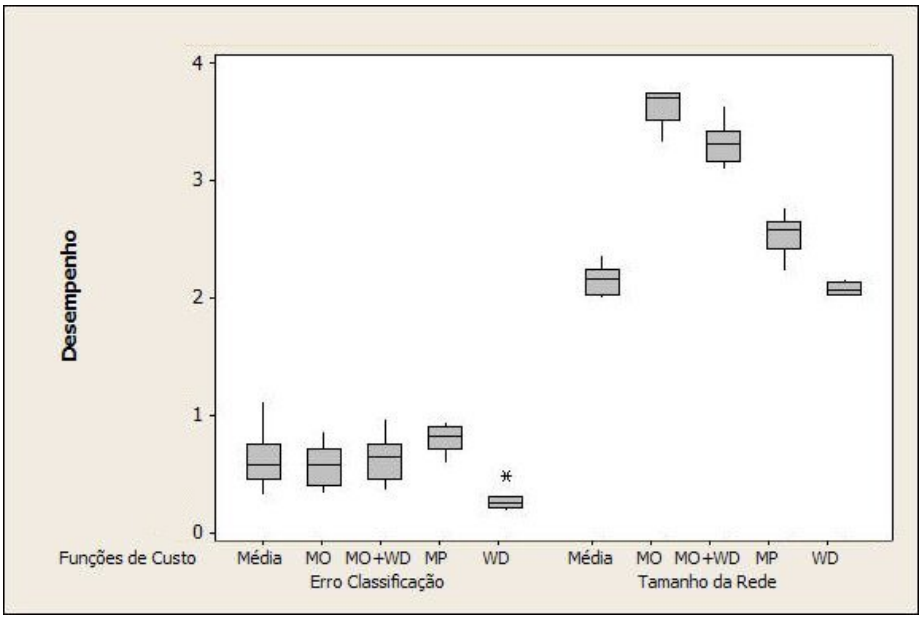

Figura 8.24 - Experimentos sobre a base *Mackey-Glass*.

O desempenho das funções de custo em cada simulação é apresentado na Figura 8.25 e na Figura 8.26. Na Figura 8.25 são realizados experimentos com as bases de dados Nariz Artificial (A) e *Makey-Glass* (B). Na Figura 8.26 são realizados experimentos com as bases de dados Diabetes (A), Íris de Fisher (B) e Tireóide (C). Cada gráfico apresenta o desempenho obtido pelas redes

neurais artificiais otimizadas. Os resultados foram ordenados pelo erro de classificação obtido utilizando cada função de custo. Um erro de classificação homogêneo em cada uma das 10 simulações realizadas geraria uma reta neste tipo de gráfico.

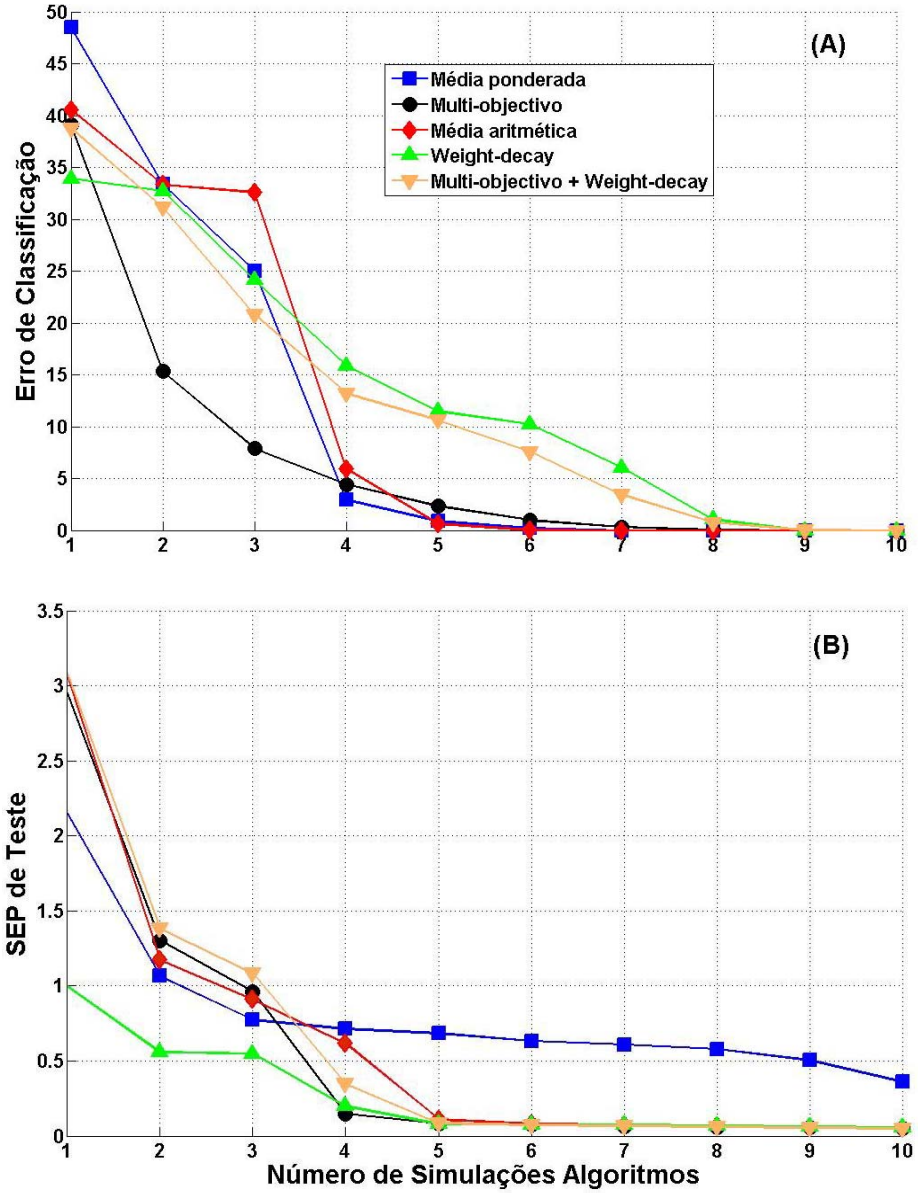

Figura 8.25 – Performance obtida por cada função de custo.

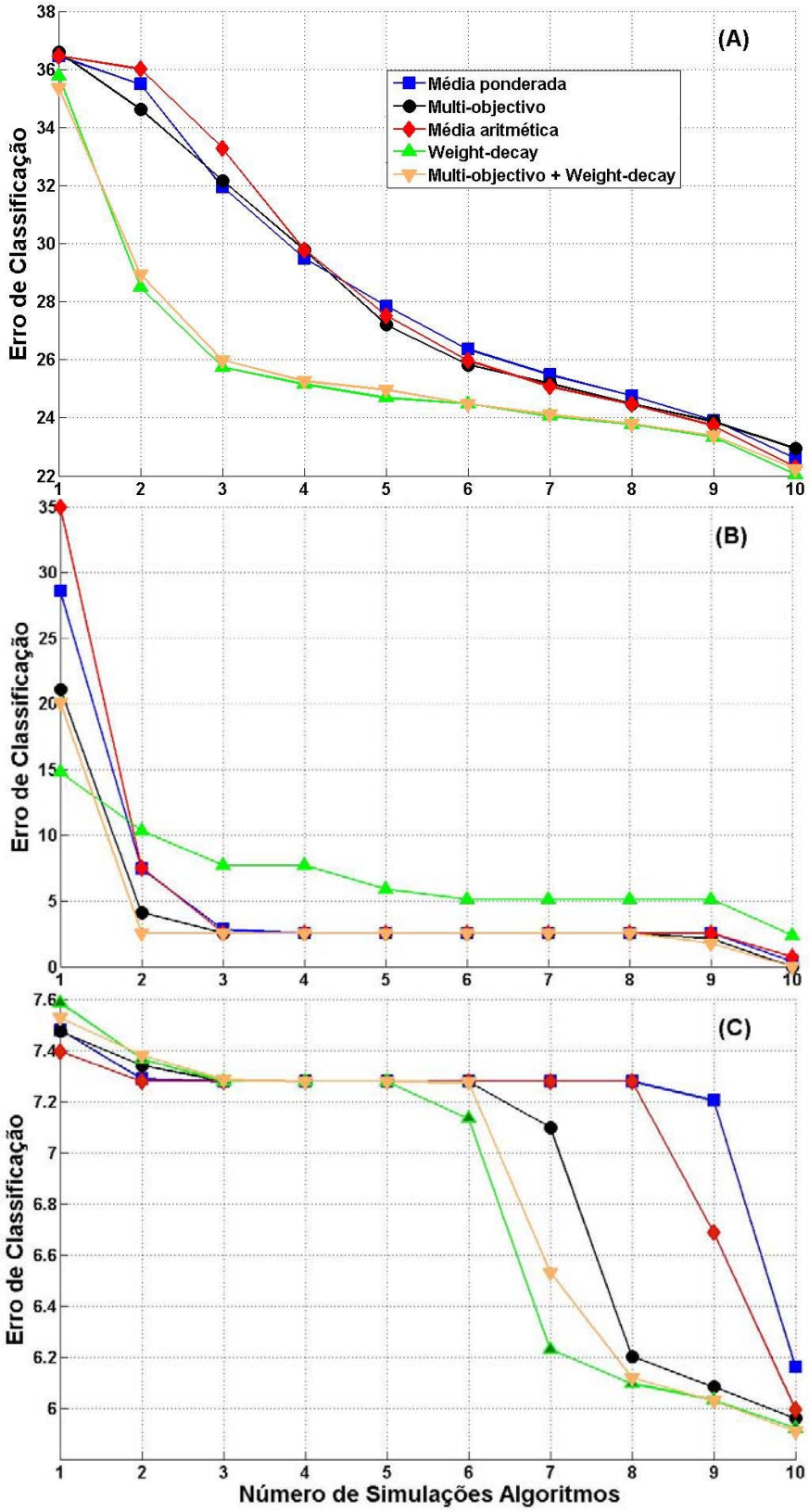

Figura 8.26– Performance obtida por cada função de custo.

Pode ser verificado que as funções de custo apresentaram um comportamento parecido na maioria das bases de dados. Isto pode caracterizar que estas diferenças no erro, obtido entre as diferentes execuções do algoritmo, esteja sendo causada por ruídos nos dados.

Na Figura 8.27 é apresentado um estudo estatístico comparando o desempenho de cada uma das funções de custo. Nesta figura os rótulos *A* e *B* representam o erro de classificação da rede e o tamanho da arquitetura. O teste estatístico *t*-student, com nível de confiança de *5%* (Mitchell, 1997), foi aplicado sobre os experimentos, a fim de confirmar a significância estatística das análises. A melhor exploração do espaço de busca foi obtida com o uso da abordagem *Weight-decay* como função de custo. Este método obteve soluções com menor complexidade e número de erros. A superioridade do método foi estatisticamente verificada nas bases de dado *Mackey-Glass*, Diabetes e Nariz Artificial.

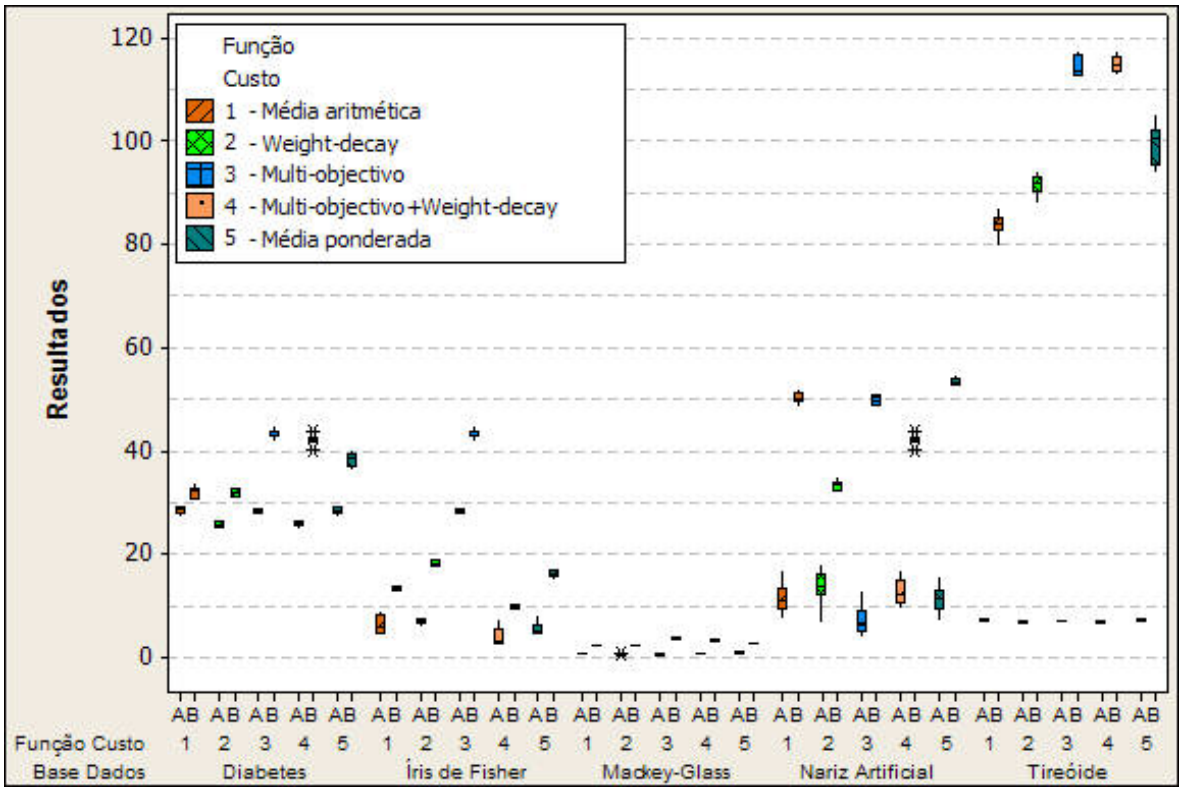

Figura 8.27 – Comparação entre os resultados

Experimentos com Método para Seleção de Atributos Relevantes

Nesta seção são apresentados os experimentos de seleção de atributos importantes realizados com o método proposto. O objetivo destes experimentos é verificar se o método proposto é capaz de realizar satisfatoriamente a seleção dos atributos mais relevantes durante o processo de

otimização dos pesos da rede neural artificial. Estes experimentos estão publicados em Zanchettin e Ludermir (2008).

Best-First e Beam Search

Na Tabela 8.16 e Tabela 8.17 são apresentados os experimentos realizados com os algoritmos *Best-First* e *Beam Search*, respectivamente. A similaridade entre os algoritmos pode ser percebida na execução dos experimentos, pois mesmo o tamanho do *beam* utilizado sendo de *15* e o número de expansões sem melhoria (*k*) igual a *50*, os dois algoritmos apresentaram comportamentos similares. Possivelmente, a diferença entre eles só possa ser percebida em bases de dados com uma maior quantidade de atributos, visto que nestes experimentos o espaço de busca explorado pelos dois métodos foi o mesmo.

Nas tabelas de desempenho apresentadas, os valores relativos à precisão dos algoritmos são referentes à classificação utilizando validação cruzada com *10 folds*, o mesmo método de avaliação utilizado para avaliar a precisão do conjunto com todos os atributos (utilizando o mesmo conjunto de instâncias). Os resultados da variante *Aleatória* apresentados correspondem à média de dez execuções do algoritmo, devido às características aleatórias do modelo.

Apesar destas características, os resultados com *Best-First* e *Beam Search* foram obtidos com grande esforço computacional. Como exemplo, na base de dados Tireóide, foram avaliados mais de *2.000* subconjuntos antes do algoritmo retornar a melhor solução encontrada. Resultado obtido mesmo com o número de expansões sem melhoria não ser alto.

Tabela 8.16– Desempenho do algoritmo *Best-First.*

Base	Todos Atributos		Best First - *Para Trás*		Best First - *Aleatória*		Best First - *Para Frente*	
	Atrib.	Class.	Atrib.	Class.	Atrib.	Class.	Atrib.	Class.
Diabetes	8	27.53%	5	31.58%	4.5	30.79%	2	30.00%
Iris	4	6.00%	2	4.29%	2	3.71%	1	4.29%
Tireóide	21	7.04%	10	1.44%	8.2	2.61%	5	1.08%

Tabela 8.17 – Desempenho do algoritmo *Beam Search.*

Base	Todos Atributos		Beam - *Para Trás*		Beam- *Aleatória*		Beam- *Para Frente*	
	Atrib.	Class.	Atrib.	Class.	Atrib.	Class.	Atrib.	Class.
Diabetes	8	27.53%	5	31.58%	3.9	30.21%	4	31.32%
Iris	4	6.00%	2	4.29%	1.9	4.14%	1	4.29%
Tireóide	21	7.04%	10	1.44%	8.9	2.19%	5	1.08%

O número de atributos selecionados refletiu as maiores diferenças entre as variantes do *Beam Search*. A variante *Para Frente* obteve soluções com número menor de atributos, principalmente na base de dados Tireóide, na qual conseguiu uma redução superior a *76%* na dimensionalidade da base.

Quando comparado ao método de otimização proposto os métodos *Best-First* e *Beam Search* apresentaram bons resultados. Primeiro deve-se destacar o bom desempenho do classificador *k-NN* sobre a base de dados Tireóide. O classificador *K-NN* apresentou bons resultados inclusive quando comparado com a MLP totalmente conectada com todos os atributos (*7.38%* de erro sobre o conjunto de teste). Este resultado pode ser justificado pelas características da base de dados, que apresenta uma distribuição pouco balanceada visto que distribuição de probabilidades das classes é de *5.1%*, *96.2%* e *2.3%*, respectivamente.

O objetivo da estratégia de busca *Beam Search* é limitar a quantidade de soluções a serem investigadas aos ramos mais promissores. Se não é imposto limite à quantidade de soluções investigadas, o método realiza uma busca exaustiva; se for limitada em apenas um elemento, a busca é equivalente à busca *Hill-Climbing.* Os métodos *Best-First* e *Beam Search* apresentaram resultados promissores, porém não são recomendados para bases de dados com um número de atributos muito grande.

Hill-Climbing

Os resultados dos experimentos realizados com o algoritmo *Hill-Climbing* são apresentados na Tabela 8.18. Os resultados da variante *Aleatória* apresentados correspondem à média de dez execuções do algoritmo, devido às características aleatórias do modelo. Pode ser verificado que não houve diferenças significativas entre estes e os resultados obtidos pelas abordagens *Best-First* e *Beam Search*. Diferente da estratégia anterior, o algoritmo *Hill-Climbing* é um algoritmo guloso cujas decisões são baseadas em informações locais e que utiliza uma função estimativa heurística para escolher o próximo elemento da solução.

Tabela 8.18 – Desempenho do algoritmo *Hill-Climbing.*

Base	Todos Atributos		Hill-limbing - *Para Trás*		Hill-Climbing - *Aleatória*		Hill-Climbing – *Para Frente*	
	Atrib.	Class.	Atrib.	Class.	Atrib.	Class.	Atrib.	Class.
Diabetes	8	27.53%	5	31.58%	5.4	31.50%	2	30.00%
Íris	4	6.00%	2	4.29%	1.9	15.14%	1	4.29%
Tireóide	21	7.04%	10	1.44%	12.9	3.98%	5	1.08%

Diferente da estratégia *Beam Search*, a variante *Para Frente* do algoritmo apresentou resultados com menor performance que as variantes *Para Trás* e *Aleatória*. Uma das características da variante *Para Frente* é a avaliação de poucos subconjuntos, esta tendência sugere uma deficiência do método em escapar de mínimos locais. Nos experimentos realizados, bases de dados com muitos atributos tiveram apenas alguns selecionados, normalmente levando a uma reduzida precisão de classificação.

Devido a sua estratégia de busca o método tende a ficar preso em mínimos locais, especialmente utilizando a variante *Para Frente*.

Ramdom Bit Climber

Diferente da metodologia utilizada nos experimentos anteriores, nos experimentos com *Random Bit Climber* foram utilizados três diferentes estados iniciais – nenhum atributo na solução inicial, solução com todos os atributos originais da base de dados e uma solução com atributos aleatoriamente selecionados – referenciados respectivamente como: *sem atributos*, *todos atributos* e *aleatório*.

A Tabela 8.19 apresenta os resultados dos experimentos com *Ramdom Bit Climber*. Nesta tabela os resultados correspondem à média de dez execuções com o algoritmo. Pode ser verificada uma pequena melhora do método, em relação aos métodos anteriores, quanto ao erro de classificação do modelo e, principalmente, uma evolução no número de atributos selecionados.

Tabela 8.19 – Desempenho do algoritmo *Ramdom Bit Climber*.

Base	Todos Atributos		Random Bit –Todos Atributos		Random Bit – Aleatório		Random Bit - Sem Atributos	
	Atrib.	Class.	Atrib.	Class.	Atrib.	Class.	Atrib.	Class.
Diabetes	8	27.53%	4	28.68%	4.4	30.68%	2	30.00%
Iris	4	6.00%	2	4.29%	1.8	4.71%	2	4.29%
Tireóide	21	7.04%	7	1.08%	7	1.32%	5	1.08%

A quantidade de estados avaliados nesta abordagem não apresentou grandes diferenças entre as três variações de inicializações. Porém, a busca que iniciou sem nenhum atributo avaliou menos atributos durante o processo de avaliação da qualidade dos subconjuntos. A superioridade do método *Random Bit Climber* em relação ao *Hill-Climbing* pode demonstrar uma menor tendência do *Random Bit Climber* em ficar preso em mínimos locais.

O método mostrou-se eficaz mesmo quando comparada com métodos mais elaborados como *Beam Search* e pode ser considerada uma boa alternativa para minimizar o número de estados avaliados. O método *Random Bit Climber* apresentou ainda bons resultados quando comparado às soluções proposta nesta tese. Se confrontados os resultados, apenas na base Diabetes o método proposto apresentou resultados superiores.

Las Vegas

Os resultados das simulações realizadas com o algoritmo *Las Vegas* são apresentados na Tabela 8.20. Nesta tabela os resultados correspondem à média de dez execuções com o algoritmo. Nas simulações realizadas o número máximo de subconjuntos gerados sem melhoria no desempenho da solução foi definida como *50*. O método não apresenta variações quanto à

estratégia de busca, já que a busca *Las Vegas* se caracteriza por ser aleatória, não se encaixando nas estratégias *Para Frente* ou *Para Trás*.

Tabela 8.20 – Desempenho do algoritmo *Las Vegas.*

Base	Todos Atributos		*Las Veagas*	
	Atrib.	Class.	Atrib.	Class.
Diabetes	8	27.53%	4	31.32%
Iris	4	6.00%	1	4.29%
Tireóide	21	7.04%	8,9	1.78%

O algoritmo utiliza força bruta, gerando subconjuntos de atributos aleatoriamente, sem qualquer heurística para otimizar o processo. Com esta estratégia de busca, o número de avaliações é extremamente alto, mesmo quando o espaço de busca é pequeno. Porém algoritmos aleatórios têm se mostrado promissores e relativamente simples para problemas de busca.

Nos experimentos realizados o algoritmo *Las Vegas* e *Beam Search* (variante *Aleatória*) apresentaram os melhores resultados na seleção de atributos na base de dados Íris – somente um atributo foi selecionado. Para as bases de dados Diabetes e Tireóide, os melhores algoritmos foram *Hill-Climbing* e *Random Bit Climber* com variante *Aleatória* – foram selecionados dois e cinco atributos, respectivamente. A performance de classificação dos algoritmos foi parecida em praticamente todos os experimentos. A degradação da classificação em relação aos experimentos com todos os atributos foi pequena.

A maioria dos algoritmos, utilizando a variante *Para Frente*, apresentaram uma avaliação menor de atributos, em relação às estratégias *Para Trás* e *Aleatória*. A avaliação de poucos subconjuntos pode impedir que atributos com inter-relacionamentos (atributos que combinados produzem melhores resultados do que separadamente) sejam selecionados.

Métodos de Otimização Global

Na Tabela 8.21 são apresentados os desempenhos dos métodos de otimização global investigados na tese, aplicados na seleção de atributos relevantes.

Tabela 8.21 – Desempenho dos algoritmos de otimização.

Base	Todos Atributos		TS		AS		GA		TSa	
	Atrib.	Class.	Atrib.	Class.	Atrib.	Class.	Atrib.	Class.	Atrib.	Class.
Diabetes	8	27,53%	7.78	27.40%	7.76	27.15%	4.96	25.99%	7.56	25.87%
Íris	4	6,00%	2.87	12.47%	2.85	12.64%	2.43	3.03%	2.71	4.61%
Tireóide	21	7,04%	20.77	7.34%	20.77	7.38%	12.83	7.28%	20.37	7.33%

Os quatro métodos de otimização global apresentaram resultados muito parecidos na seleção de atributos relevantes. A abordagem que mais se destacou foram os algoritmos genéticos, superando as demais abordagens nas três bases de dados investigadas.

Método Proposto GaTSa

Em relação ao método proposto é possível afirmar que ele apresentou resultados promissores na seleção de atributos relevantes. Nas três bases de dados investigadas o método apresentou resultados muito próximos dos métodos clássicos de seleção de atributos. Apesar de não ter sido desenvolvido especificamente para seleção de atributos relevantes e sim para otimização de conexões em redes neurais artificiais, o método, mesmo que de forma indireta é capaz de eliminar as entradas menos relevantes para o algoritmo de aprendizado. Na Tabela 8.22 são apresentados os resultados das simulações realizadas com o método.

Tabela 8.22 – Desempenho do algoritmo GaTSa.

Base	Todos Atributos		GaTSa	
	Atrib.	Class.	Atrib.	Class.
Diabetes	8	27.53%	1.51	27.06%
Iris	4	6.00%	3.36	5.23%
Tireóide	21	7.04%	7.12	7.15%

Uma das características observadas com o método proposto foi o fato dele apresentar dificuldade em realizar uma boa seleção de atributos relevantes quando falta informação na base de dados para que a tarefa de classificação seja desempenhada com erros baixos. Esta característica foi observada em algumas das execuções do algoritmo, na qual as redes obtidas não apresentavam uma boa performance de classificação. Isto ocorre porque a eliminação não é feita levando em consideração cada nodo de entrada da rede, de modo que a eliminação de uma unidade de entrada só acontece se forem eliminadas todas as conexões que partem da mesma. Dessa forma, a exclusão de um atributo ocorre como conseqüência do processo de redução de conexões. Assim o método proposto não verifica a contribuição de cada atributo de entrada durante a otimização, pois a ênfase está na contribuição de cada conexão da rede. Se o foco da otimização for à seleção de atributos relevantes o projetista possivelmente poderia considerar outras características na função de avaliação além do erro de classificação e o número de conexões da rede.

Alem disso, outras diferenças são perceptíveis entre os vários métodos de seleção de atributos e o algoritmo proposto, uma das principais são as características aleatórias do modelo. Neste tipo de algoritmo o valor esperado irá depender de escolhas aleatórias realizadas no algoritmo e não de distribuições impostas sobre os atributos de entrada dos dados. Este comportamento diferencia o

tempo esperado de processamento, presente nos algoritmos aleatórios, da complexidade de caso médio, normalmente utilizada na análise de algoritmos determinísticos.

Vale ressaltar ainda que o comportamento dos algoritmos com características aleatórias poderá variar, mesmo quando aplicado repetidas vezes para uma mesma entrada. Desta forma, o tempo de processamento se torna uma variável aleatória e a análise de tempo de processamento exige que se tenha uma maior compreensão da sua distribuição de probabilidade associada.

Intuitivamente, quanto maior o número de atributos em uma base de dados, maior o poder discriminatório do classificador e a facilidade de extrair modelos de conhecimento da base. Porém, quando a teoria vira prática, o mundo real apresenta provas de que nem sempre isso é verdade. Isso porque, em primeiro lugar, muitos métodos sofrem da maldição da dimensionalidade, ou seja, o tempo computacional do algoritmo aumenta de acordo com o número de atributos existentes na base (Lee, 2005). Além disso, experimentos comprovam que o número de exemplos utilizados para garantir uma certa taxa de classificação cresce exponencialmente com o número de atributos irrelevantes presentes (Langley e Iba 1993).

Não é possível comparar de forma justa o desempenho de classificação dos modelos utilizados. Os algoritmos de seleção de atributos clássicos implementaram, em um modelo *wrapper*, o algoritmo de classificação *k-NN*, enquanto que o algoritmo proposto e os demais métodos de otimização global implementam redes neurais artificiais *Multi-Layer Perceptron*. Apesar da maior complexidade do modelo proposto, nos experimentos realizados, com as três bases de dados, foi verificado que as características de algumas bases de dados privilegiaram o modelo *k-NN*. Como exemplo a base de dados Tireóide que possui uma distribuição de probabilidades que privilegia muito uma das classes (92.6% dos dados são de uma das classes).

Na Tabela 8.23 são apresentados os atributos selecionados por cada uma das técnicas investigadas. Nesta tabela são apresentados os atributos que na média foram destacados como de maior relevância para a base de dados. Os atributos estão ordenados segundo a ordem de importância, definida pelo número de citações em cada um dos experimentos realizados.

Tabela 8.23 - Atributos relevantes selecionados por cada umas das técnicas

Método	**Base de dados Tireóide (Atributos)**											
Beam Search	3	8	5	17	7	18	19	4	6	12	13	15
Best-First	3	8	17	7	21	19	13	12	5	18	15	9
Hill-Climbing	3	12	8	5	17	15	9	4	12	18	19	11
Random Bit Climber	3	8	13	17	15	21	7					
Las Vegas	3	17	8	21	7	10	2	4	5			
GaTSa	3	8	5	17	21	7	12	15	13			

	Base de dados Diabetes (Atributos)						
Beam Search	2	7	5	8	1		
Best-First	2	8	4	5	6	1	
Hill-Climbing	2	8	1	4	6	5	7
Random Bit Climber	2	1	8	6	5		
Las Vegas	2	5	7	8			
GaTSa	2	8					
	Base de dados Íris de Fisher (Atributos)						
Beam Search	4	3					
Best-First	4	3					
Hill-Climbing	4	3	2				
Random Bit Climber	3	4					
Las Vegas	4						
GaTSa	4	3	2				

Pode ser verificado que o método proposto apresenta resultados muito próximos aos demais métodos investigados. Esta performance média pode ser utilizada como base na assertiva de que o método apresenta resultados similares aos métodos de seleção de atributos relevantes clássicos e pode ser utilizado com segurança neste tipo de tarefa.

Se considerarmos somente os algoritmos de otimização global, o método proposto apresentou o melhor desempenho em relação à eliminação das unidades de processamento de entrada e escondida. Na eliminação das unidades de entrada foi o método que conseguiu eliminar em média mais unidades de processamento.

O teste estatístico *t*-student, com nível de confiança de 5% (Mitchell, 1997), foi aplicado sobre os experimentos, a fim de confirmar a significância estatística das análises. O melhor desempenho da técnica GaTSa na eliminação das unidades de processamento de entrada foi estatisticamente comprovado. Na eliminação das unidades de processamento da camada intermediária todos os métodos apresentaram desempenho estatisticamente equivalente, apesar do método GaTSa ter apresentado desempenho superior quando testada com arquitetura fixa. Estes resultados não são ruins, servem para comprovar que a estratégia construtivista funciona. Nestes experimentos o número de unidades de processamento na camada intermediária foi otimizado em conjunto com os demais parâmetros da rede neural artificial.

Na Figura 8.28 são apresentados o desempenho de cada um dos métodos de otimização global na eliminação das unidades de processamento de entrada e escondida.

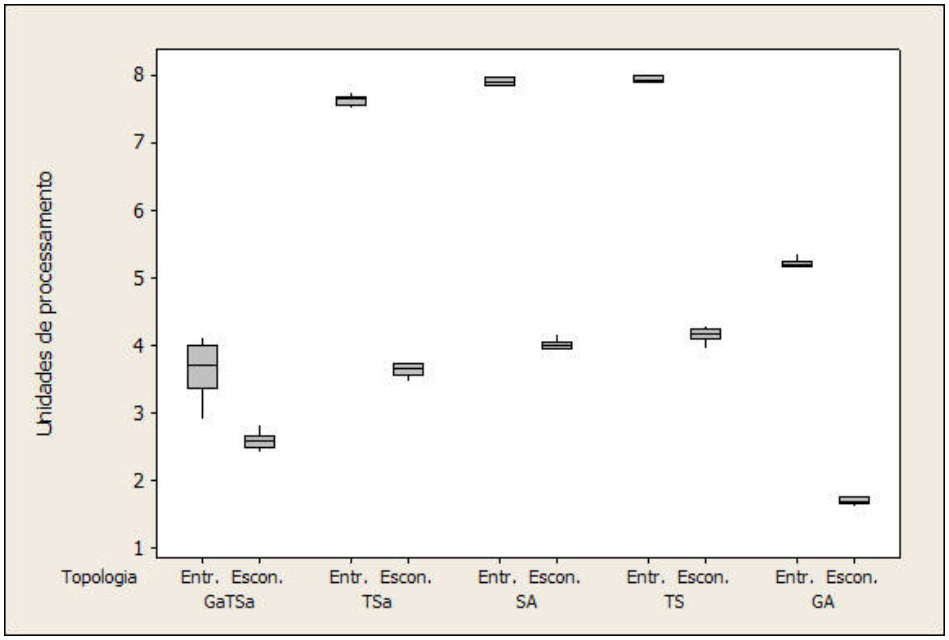

Figura 8.28 - Número de unidades de Entrada (Entr.) e Escondidas (Escon.).

Planejamento Fatorial de Experimentos com o Método Proposto

O planejamento fatorial de experimentos foi utilizado com o intuito de verificar o comportamento do método proposto de forma sistemática e controlada. Nesta análise também foi possível identificar os fatores (parâmetros) que possuem maior influência no desempenho do sistema. A análise da influência de cada um dos parâmetros de configuração pode ser muito útil no projeto e utilização do modelo, visto a quantidade de parâmetros configuráveis, isso devido à integração de diferentes técnicas. A expectativa é que uma quantidade pequena de parâmetros possua grande influência média no desempenho do modelo. Nesta análise são verificados os parâmetros mais influentes e também a interação e inter-relação entre eles.

Neste estudo optou-se pela realização de um experimento fatorial com dois níveis (experimento fatorial 2^k), objetivando diminuir a quantidade de experimentos a serem realizados. Os fatores controlados na investigação são apresentados na Tabela 8.24.

Tabela 8.24 - Fatores controlados nos experimentos com o método GaTSa.

Fatores		Níveis	
		Inferior (-1)	Superior (+1)
A	Quantidade de Soluções Vizinhas Geradas a cada Iteração	5	10
B	Fator de Redução da Temperatura	0.7	0.9
C	Maximo de Iterações Permitidas	50	100
D	Número de Micro-evoluções dos Operadores Genéticos	5	10
E	Taxa de Recombinação dos Operadores Genéticos	0.7	0.9
F	Taxa de Mutação dos Operadores Genéticos	0.1	0.3
G	Tamanho da Lista *Tabu*	10	20

Os fatores *H = sigmóide logística* (tipo da função de ativação da rede), *I = Backpropagation com momentum* (método de otimização local dos pesos da rede), *J = 0.001* (taxa de aprendizado para ajuste dos pesos da rede), *K = 0.6* (termo de *momentum* para o *Backpropagation*), *L = 10* (quantidade de iterações entre reduções sucessivas de temperatura), *M = 5* (parâmetro *k* do critério de parada do treinamento $GL_{(k)}$ do *Proben1*), *N = 0.01* (limiar de comparação entre duas soluções para identificar soluções similares) e *O = 1* (temperatura inicial) foram mantidos fixos durante os experimentos.

Os ensaios foram realizados em ordem aleatória, sendo considerados sete fatores de controle (definidos na Tabela 8.24), cada um deles com dois níveis, resultando em *128* combinações. Considerando as replicações necessárias na execução do método, cada uma das combinações dos níveis dos fatores de controle foi realizada cinco vezes, num total de *640* ensaios. Devido às características aleatórias do modelo em estudo nas inicializações dos pesos, cada um dos *640* ensaios corresponde a uma média de *30* execuções, desta forma foram realizados *19.200* simulações com o método proposto.

Para as simulações foi utilizada a base de dados Íris de Fischer. Com esta base de dados foram construídas cinco partições de dados que representam as replicações nas simulações realizadas.

Com base na análise de variância do experimento fatorial - ANOVA (Montgomery e Runger, 2003) realizado, considerando o nível estatístico de significância $\alpha = 5\%$ na função de distribuição *F*, dois fatores foram identificados com maior influência sobre o desempenho de classificação das redes neurais artificiais otimizadas pelo modelo. Estas análises são apresentadas na Tabela 8.25.

Os fatores mais relevantes são: Número de Micro-evoluções dos Operadores Genéticos (D), explicando $\approx 32.64\%$ da variância do sistema; e Taxa de Recombinação dos Operadores Genéticos (E), que explica $\approx 24.04\%$ da variância. Também foi identificada interação entre os fatores: Número de Micro-evoluções dos Operadores Genéticos (D) e Taxa de Recombinação dos Operadores Genéticos (E), correspondendo a $\approx 10.39\%$ da variância do sistema; e Quantidade de Soluções Vizinhas Geradas a cada Iteração (A) e Tamanho da Lista *Tabu* (G), com $\approx 1.82\%$ da variância total dos dados.

Tabela 8.25 - Resultados da tabela ANOVA considerando o erro de classificação obtido.

	Soma Quadrática	Graus de Liberdade	Média Quadrática	F_0	Nível de significância
Fatores					
D	317.128	1	317.128	72.43	0.00
E	233.301	1	233.301	53.28	0.00
Interações					
DE	100.888	1	100.888	23.04	0.00
AG	17.692	1	17.692	4.04	0.00

É interessante verificar que apesar da grande quantidade de parâmetros configuráveis do método, poucos possuem influência significativa na qualidade de classificação das redes neurais artificiais otimizadas. Esta característica é interessante, pois até mesmo projetistas inexperientes podem obter sucesso em sua utilização. Os parâmetros que mais influenciaram no desempenho do método foram as variáveis que controlam a evolução das soluções no espaço de busca. O método foi robusto às demais configurações, pois estas não apresentaram alterações significativas no desempenho das redes geradas. Não é possível afirmar neste momento se este comportamento se reflete em bases de dados com outras características, principalmente com bases de dados de previsão.

Os parâmetros que apresentaram maior interação também estão relacionadas aos operadores genéticos. Os parâmetros que ajustam a exploração do espaço de busca, como a Quantidade de Soluções Vizinhas Geradas a cada Iteração (A) e Tamanho da Lista *Tabu* (G). A influência de cada fator e a interação entre eles pode ser verificada na Figura 8.29.

Na Figura 8.30 são apresentados os principais efeitos de cada fator sobre a resposta do modelo. Nesta análise, pode ser verificado que um menor número de gerações e uma taxa de recombinação mais alta contribuem para redes com uma melhor taxa de classificação. Diferenças no número de soluções geradas a cada iteração, valor inicial da temperatura inicial e o tamanho da lista *tabu* não influenciaram significativamente no resultado do modelo. Também foi verificado que um menor número de iterações e uma maior taxa de mutação contribuem para o sucesso do modelo.

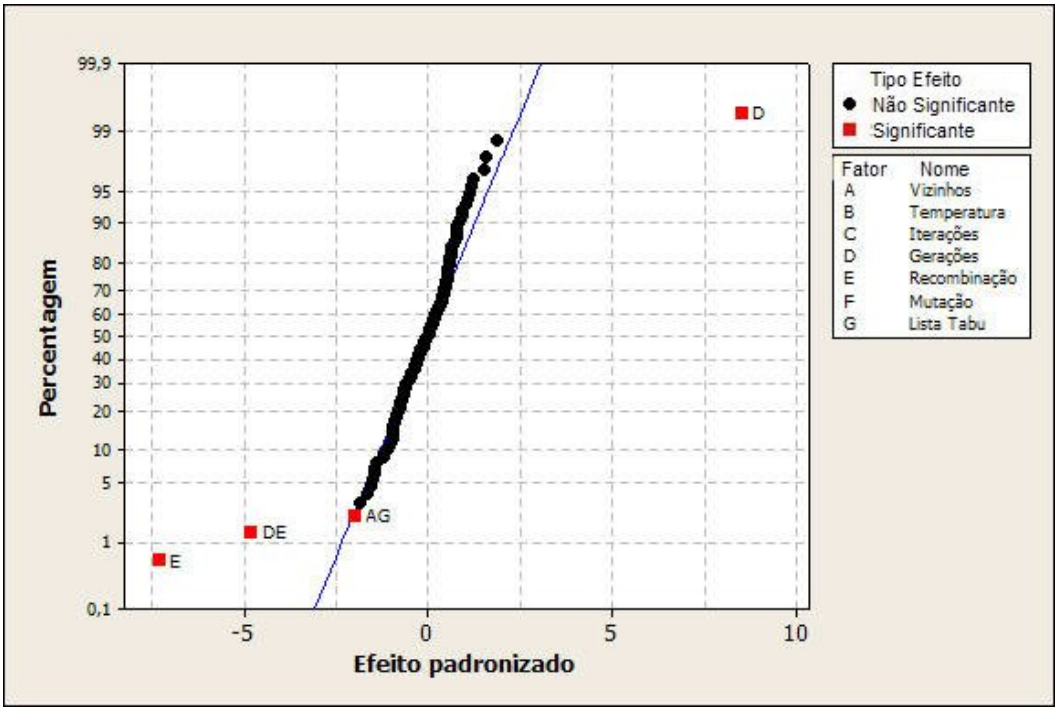

Figura 8.29 - Significância estatística de cada fator.

Na Tabela 8.26 e Figura 8.31 são apresentados os resultados obtidos dos experimentos quando tentou-se verificar a influência de cada um dos parâmetros do sistema na obtenção de redes com poucas conexões. Os fatores mais relevantes são: Número de Micro-evoluções dos Operadores Genéticos (D), explicando $\approx 28.19\%$ da variância do sistema; Taxa de Recombinação dos Operadores Genéticos (E), que explica $\approx 17.85\%$ da variância, Maximo de Iterações (C) permitidas que explica $\approx 6.56\%$ da variância; e Quantidade de Soluções Vizinhas Geradas a cada Iteração (A) que explica $\approx 3.40\%$ da variância total do sistema. Também foi identificada interação entre vários fatores. As interações e inter-relações com maior relevância estatística foram: Número de Micro-evoluções dos Operadores Genéticos (D) e Taxa de Recombinação dos Operadores Genéticos (E), correspondendo a $\approx 10.63\%$ da variância do sistema; Quantidade de Soluções Vizinhas Geradas a cada Iteração (A) e Taxa de Recombinação dos Operadores Genéticos (E), com $\approx 6.96\%$ da variância; e Maximo de Iterações Permitidas (C) e Taxa de Recombinação dos Operadores Genéticos (F) com $\approx 3.27\%$ da variância total dos dados.

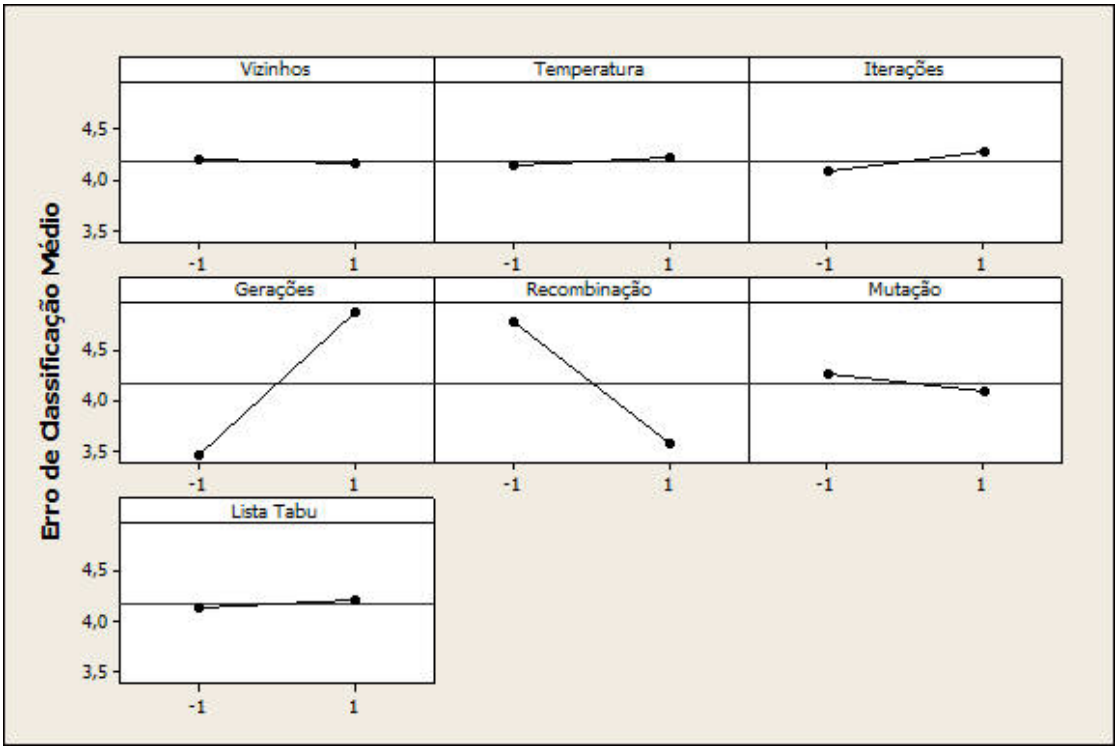

Figura 8.30 - Efeito causado por cada fator.

Tabela 8.26 - Resultados da tabela ANOVA considerando o número de conexões obtido.

	Soma Quadrática	Graus de Liberdade	Média Quadrática	F_0	Nível de significância
Fatores					
D	1227.664	1	1227.664	833.27	0.00
E	777.483	1	777.483	527.71	0.00
C	285.958	1	285.958	194.09	0.00
A	148.033	1	148.033	100.48	0.00
Interações					
DE	463.080	1	463.080	314.31	0.00
AE	303.326	1	303.326	205.88	0.00
CE	142.318	1	142.318	96.60	0.00
CD	25.760	1	25.760	17.48	0.00
AC	25.361	1	25.361	17.21	0.00
CEG	10.404	1	10.404	7.06	0.008
ACFG	9.216	1	9.216	6.26	0.013
BE	8.418	1	8.418	5.71	0.017
ABCEF	8.327	1	8.327	5.65	0.018
CDEG	7.700	1	7.700	5.23	0.023
ACDE	7.396	1	7.396	5.02	0.025
ACE	7.268	1	7.268	4.93	0.027
BDE	6.241	1	6.241	4.24	0.040
CDEF	6.084	1	6.084	4.13	0.043
ABCDF	5.700	1	5.700	3.87	0.050
ABDFG	5.439	1	5.439	3.69	0.055

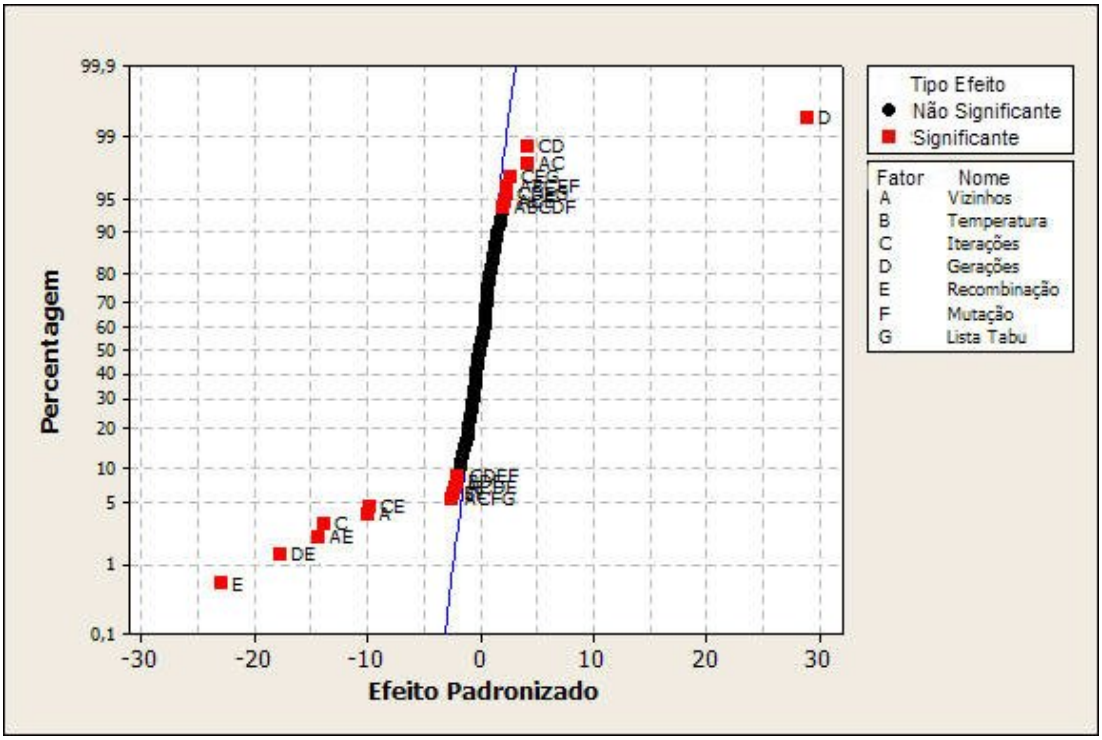

Figura 8.31 - Significância estatística de cada fator.

Nesta análise pode ser verificado que diferente do momento em que se privilegiou o erro de classificação da rede neural artificial, quando se analisou o número de conexões final da rede um maior número de fatores influenciou no resultado do modelo. Também se verificou vários fatores com interações e inter-relações. Novamente os parâmetros genéticos, que controlam a geração de novas soluções foram os mais influentes. O valor inicial para o parâmetro temperatura e o tamanho da lista *tabu*, novamente não apresentaram grande influência nos resultados. Os fatores que mais influenciaram positivamente o resultado do modelo foram um maior número de soluções geradas a cada iteração, um maior número de iterações, um menor número de micro-evoluções e uma taxa maior de recombinação.

Na Figura 8.32 e na Figura 8.33 são apresentada a influência e interação entre os fatores investigados. Os parâmetros que apresentaram maior interação com os demais foram o número de micro-evoluções e a taxa de recombinação nos operadores genéticos. Os demais fatores apresentaram comportamento praticamente similar quando se variava os demais parâmetros do sistema.

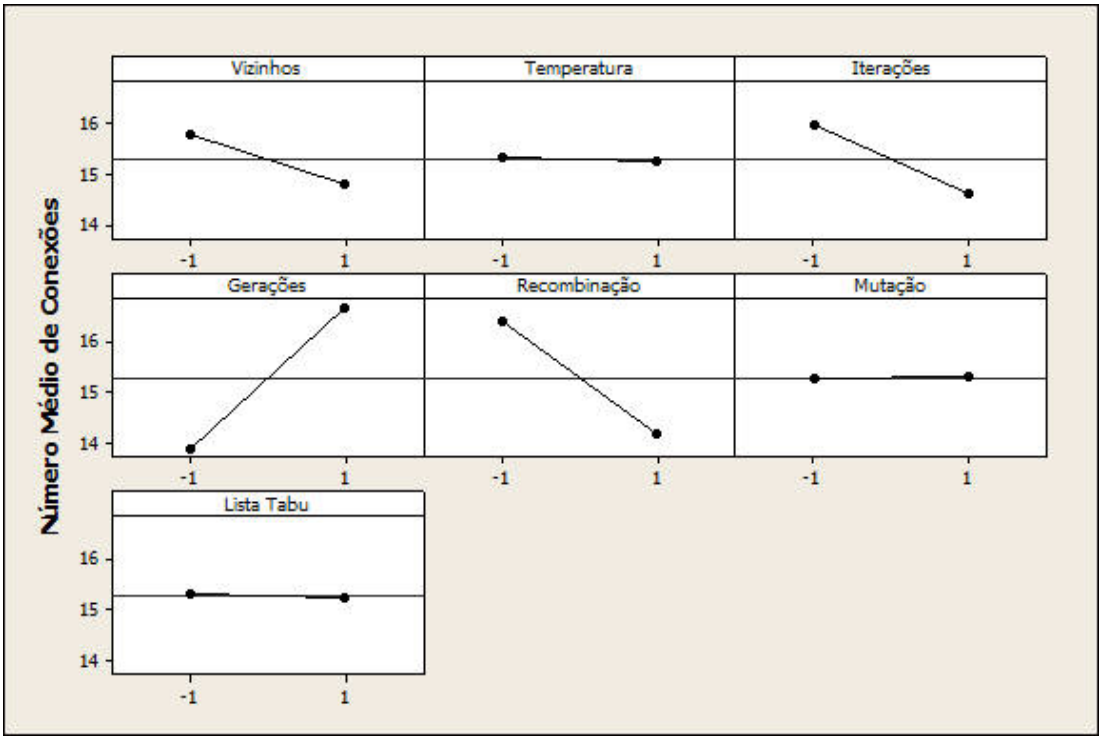

Figura 8.32 - Efeito causado por cada fator.

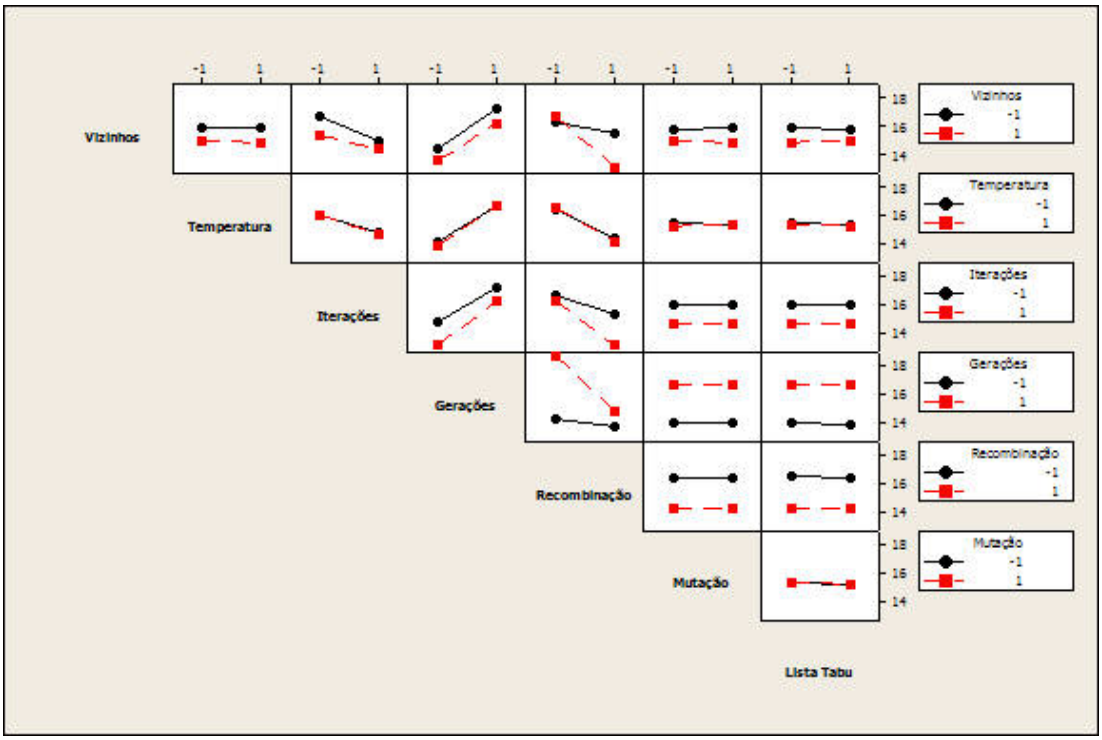

Figura 8.33 - Gráfico de interação entre os fatores.

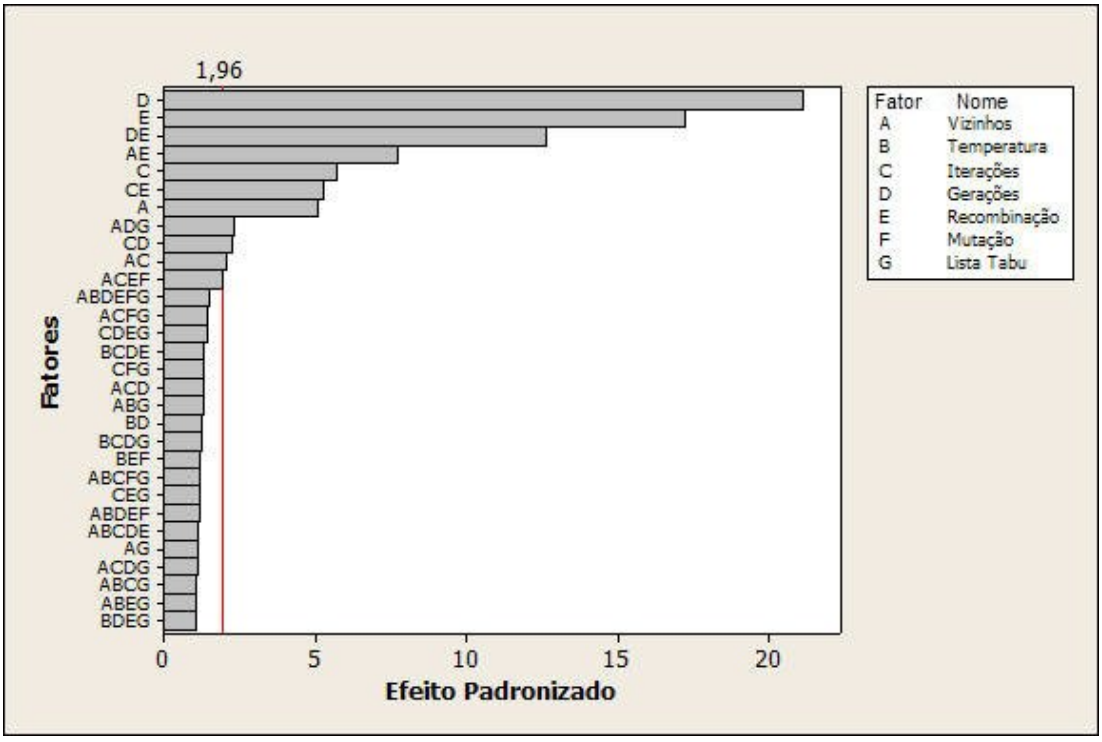

Figura 8.34 - Significância estatística de cada fator – 30 primeiros fatores mais significativos.

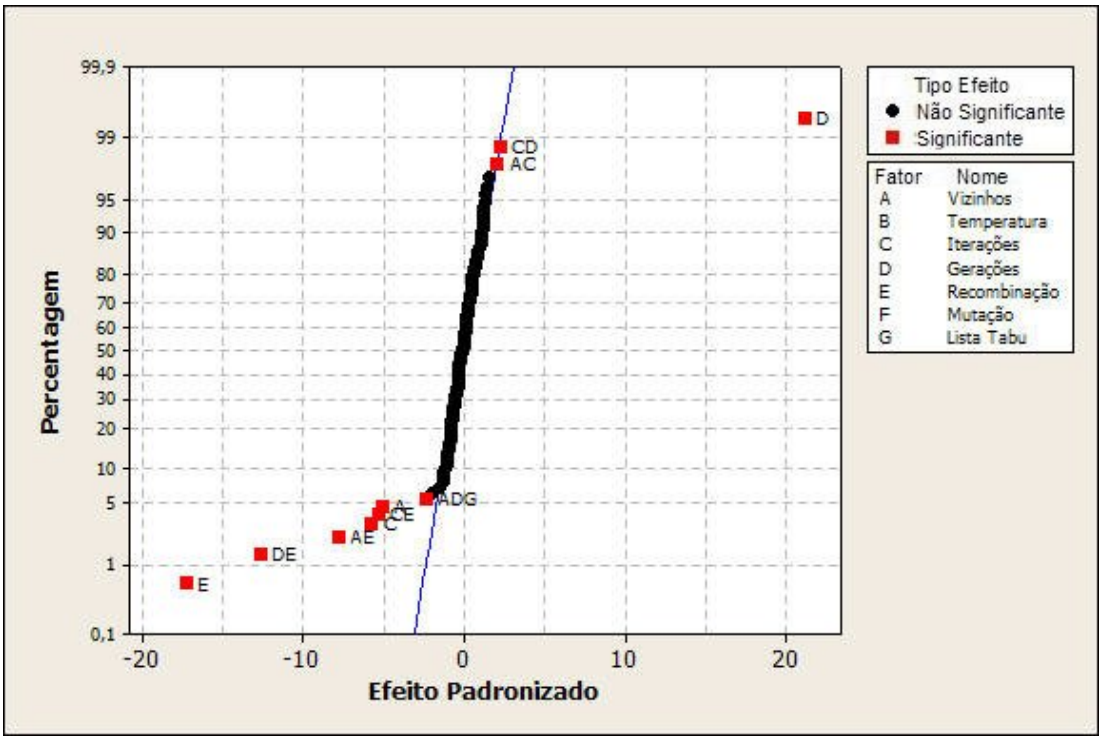

Figura 8.35 - Significância estatística de cada fator.

Na Figura 8.34 e Figura 8.35 são apresentadas as análises realizadas sobre a média entre o número de conexões e o erro médio de classificação das redes neurais artificiais obtidas ao final da

otimização. Novamente se repetem os fatores mais influentes nas simulações anteriores. Estas análises sugerem que os resultados podem ser robustos para outros experimentos e variáveis de desempenhos investigadas.

Capítulo 9

Conclusões e Trabalhos Futuros

Esta tese apresentou um novo método de otimização baseado na integração das heurísticas de busca *Simulated Annealing* (Kirkpatrick, 1983), *Tabu Search* (Glover e Laguna, 1997), Algoritmos Genéticos (Goldberg, 1989) e *Backpropagation* (Rumelhart et al, 1986). A técnica proposta combinou estratégias de busca global e local apresentando excelentes resultados na relação, abrangência do espaço de soluções investigado, custo computacional e tempo de busca.

O método proposto foi desenvolvido como um algoritmo iterativo geral. Desta forma, ele pode facilmente implementar uma diversidade de problemas, bastando para esta tarefa, adaptar a representação da solução, função de custo e o mecanismo para gerar novas soluções sobre o espaço de busca.

A efetividade deste método é investigada na otimização simultânea da topologia e dos valores dos pesos das conexões entre as unidades de processamento de redes neurais artificiais *Multi-layer Perceptron* (Rumelhart e Weigend, 1990). O objetivo desta otimização é gerar topologias de rede com poucas conexões e alto desempenho para qualquer conjunto de dados. A heurística proposta, diferente de outras abordagens para o problema, realiza a busca de forma construtiva e baseada na poda das conexões entre as unidades de processamento da rede neural artificial. Assim, devem ser geradas redes com arquitetura variável e que podem ser ajustadas para cada problema de forma automática.

Para verificar o desempenho do modelo proposto foram realizados experimentos com redes neurais *Multy-layer Perceptron* totalmente conectadas e com as técnicas *Simulated Annealing*, *Tabu Search* e Algoritmos Genéticos de forma isolada. Também foram realizados experimentos com uma técnica que combina as heurísticas de *Simulated Annealing*, *Tabu Search* e *Backpropagation* (Yamazaki, 2004). Nestes experimentos, foi verificada uma superioridade da

técnica proposta tanto quando é analisado o tamanho das redes neurais artificiais geradas quanto no erro de classificação obtido ao final da otimização.

Diferente das demais técnicas investigadas, a técnica proposta não precisa que o projetista defina uma arquitetura máxima de rede para ser realizada a poda das conexões. O algoritmo realiza a busca de forma construtiva e vai eliminando conexões desnecessárias de forma automática. Nas simulações realizadas, as soluções iniciais possuem o tamanho mínimo de uma arquitetura válida – número de entradas e saídas relacionadas ao problema investigado e uma unidade de processamento na camada intermediária. Foram realizados experimentos utilizando uma arquitetura inicial, na qual a camada intermediária possuía tamanho variável. O número de unidades de processamento na camada intermediária era um número aleatório no intervalo [*1, número de saídas*]. Porém, testes estatísticos não confirmaram melhora no desempenho do modelo ou no tempo de processamento do método.

Também foi comprovado experimentalmente que a codificação das soluções utilizando dois vetores – um com os valores dos pesos e outro representando a conectividade da rede – é interessante para a generalização das redes geradas e para a velocidade na otimização destes modelos. Quando se tentou codificar soluções apenas com os valores dos pesos da rede (eliminando os bits de conectividade) e considerando inexistentes as conexões de peso nulo, obteve-se uma degradação na velocidade do algoritmo. Apesar do tamanho do vetor que representa as soluções diminuir e intuitivamente oferecer um ganho na performance do algoritmo, na prática se verificou que quando os valores das conexões são perdidos e não podem ser reaproveitados quando conexões anteriormente eliminadas são re-estabelecidas, o processo de aprendizado sofre deterioração.

No desenvolvimento da tese havia a preocupação em realizar experimentos de forma controlada e identificar quais parâmetros mais influenciavam no desempenho dos modelos. O planejamento fatorial de experimentos (Montgomery, 1997) foi utilizado com este fim. Pra verificar o potencial desta técnica, primeiramente ela foi utilizada para verificar quais fatores possuem maior influência no desempenho dos sistemas neuro-difusos *Adaptative Neuro Fuzzy Inference System* (Jang, 1993) e *Evolving Fuzzy Neural Network* (Kasabov, 1998b). Foram obtidas interessantes conclusões sobre os principais parâmetros de configuração destes dois modelos. Estas informações são muito importantes para projetistas deste tipo de técnicas, especialmente pelo fato destes modelos apresentarem grande quantidade de parâmetros que precisam ser ajustados para a correta operação do sistema. Este problema de configuração se agrava ao considerarmos que estes parâmetros são muitas vezes variáveis difusas e que estes sistemas são modelos que operam em tempo real. A técnica de planejamento fatorial de experimentos também foi utilizada para avaliar o comportamento do método proposto. Neste experimento foi verificado o comportamento do

método em diferentes situações, identificada a relevância, interação e inter-relação dos parâmetros de configuração do modelo.

Posteriormente foi investigada a influência das funções de custo no desempenho do método proposto. As funções de custo são responsáveis por avaliar a qualidade de cada uma das soluções geradas pelo algoritmo e direcionar a busca para um possível ponto de ótimo (local ou global). Foram investigadas cinco funções de custo com características totalmente distintas: *(i)* média aritmética entre o erro de classificação e a percentagem de conexões utilizadas pela rede neural artificial; *(ii)* média aritmética ponderada entre a percentagem de conexões, percentagem de unidades escondidas e o erro de classificação da rede; *(iii)* um mecanismo inspirado na técnica *weight-decay* (Hinton, 1986); *(iv)* uma estratégia de otimização multiobjetivo; e *(v)* a combinação das estratégias *weight-decay* e otimização multiobjetivo. A melhor exploração do espaço de busca foi obtida com o uso da abordagem *weight-decay* como função de custo.

Outra questão abordada foi a capacidade do método em selecionar atributos relevantes. O objetivo destes experimentos era verificar se o método proposto conseguia realizar satisfatoriamente a seleção dos atributos relevantes durante o processo de otimização dos pesos da rede neural artificial. Para avaliar o desempenho do método proposto nesta questão, métodos clássicos de seleção de atributos com diferentes características foram utilizados. Foram realizados experimentos com as técnicas *Hill-Climbing* (Vafaie e De Jong, 1993), *Random Bit Climber* (Davis, 1991), *Best-First* (Aha e Banker, 1995), *Beam Search* (Aha e Banker, 1995) e *Las Vegas* (Liu e Setiono, 1996).

Apesar do método ter se mostrado promissor na seleção de atributos relevantes, superando outras técnicas de otimização global investigadas, foi observado uma dificuldade do modelo em obter bom desempenho quando falta informação na base de dados para que a tarefa de classificação seja desempenhada com erros baixos. Esta característica foi observada em algumas das execuções do algoritmo, na qual as redes obtidas não apresentavam uma boa performance de classificação. Isto ocorre porque a eliminação não é feita levando em consideração cada nodo de entrada da rede. A eliminação de uma unidade de entrada só acontece se forem eliminadas todas as conexões que partem da mesma. Dessa forma, a exclusão de um atributo ocorre como conseqüência do processo de redução de conexões. Assim, o método proposto não verifica a contribuição de cada atributo de entrada durante a otimização, pois a ênfase está na contribuição de cada conexão da rede. Se o foco da otimização for à seleção de atributos relevantes o projetista possivelmente poderia considerar outras características na função de avaliação além do erro de classificação e o número de conexões da rede.

De forma geral o método proposto se mostrou eficiente e promissor nos experimentos realizados, se tornando uma alternativa interessante para otimização simultânea da topologia e dos

valores dos pesos das conexões entre as unidades de processamento de redes neurais artificiais *Multi-layer Perceptron*.

9.1 Avaliação das Contribuições Propostas pela Tese

Originalmente, foram propostos os seguintes avanços e contribuições técnicas:

(i) **Proposta de um método de otimização que combine as potencialidades das principais técnicas de otimização global e local.** A técnica proposta foi descrita no Capítulo 7 e integra as principais características das heurísticas de busca *Simulated Annealing*, *Tabu Search*, Algoritmos Genéticos e *Backpropagation*. A heurística de busca foi estruturada de forma construtiva e baseada em poda, na qual as soluções são iniciadas com a menor topologia possível e vão crescendo de acordo com a necessidade. Desta forma, foram combinadas estratégias de busca global e local apresentando excelentes resultados na relação, abrangência do espaço de soluções investigado, custo computacional e tempo de busca. No desenvolvimento da técnica proposta foram investigadas cinco diferentes funções de custo (média aritmética, média aritmética ponderada, *weight-decay*, multiobjetivo e a combinação de multiobjetivo e *weight-decay*) para o método.

(ii) **Otimização de redes neurais artificiais *Multi-layer Perceptron* de forma construtiva e baseada na poda das conexões da rede.** No Capítulo 9 foram apresentadas as simulações realizadas com diversas abordagens na otimização da topologia e dos valores dos pesos de redes neurais artificiais *Multi-layer Perceptron*. Dentre as técnicas investigadas a abordagem proposta apresentou os melhores resultados na maioria das bases de dados. Apesar do problema investigado envolver uma questão crítica, a relação estabilidade versus plasticidade no treinamento de redes neurais artificiais, os resultados obtidos mostraram-se promissores.

(iii) **Verificar se a abordagem pode ser utilizada para seleção de atributos.** Durante a otimização da arquitetura da rede, unidades de processamento de entrada podem ser eliminadas de acordo com sua relevância para o desempenho do modelo. No Capítulo 5 foi realizada uma revisão sobre as técnicas clássicas de seleção de atributos e estudadas algumas técnicas que apresentavam características similares às da abordagem proposta. No Capítulo 9 foram apresentadas simulações que comprovaram a eficiência da técnica na seleção de atributos relevantes durante o processo de otimização das redes neurais artificiais. A técnica proposta apresentou resultados similares às técnicas de seleção de

atributos clássicas e desempenho superior às técnicas de otimização global investigadas na seleção de atributos relevantes.

(iv) **Identificar os parâmetros mais relevantes do desempenho dos modelos baseados em aprendizagem, verificando suas influências e possivelmente diminuindo a quantidade de parâmetros ajustáveis no projeto destes modelos.** No Capítulo 9 foram apresentados experimentos realizados com a técnica de planejamento fatorial de experimentos (descrita no Capítulo 8 e no Anexo A) com dois modelos neuro-difusos. Nestes experimentos foram obtidas interessantes conclusões sobre os principais parâmetros de configuração das duas técnicas. Estas informações são muito importantes para projetistas deste tipo de técnicas, especialmente pelo fato destes modelos apresentarem grande quantidade de parâmetros livres que precisam ser ajustados para a correta operação do sistema em determinado problema. Este problema de configuração se agrava ao considerarmos que estes parâmetros são muitas vezes variáveis difusas e que estes sistemas são modelos que operam em tempo real.

(v) **Identificar o comportamento do método de forma sistemática e controlada, utilizando técnicas estatísticas de experimentação científica.** É difícil afirmar se os resultados apresentados nesta tese podem ser estendidos para as demais classes de problemas. Wolpert e Macready (1997) apresentaram um estudo atestando que a performance média de qualquer par de algoritmos iterativos (determinísticos ou não) sobre todos os tipos de problemas é idêntica. Sendo assim, se um algoritmo obtém uma performance muito boa sobre uma classe de problemas, então necessariamente paga por isso com uma degradação de performance no conjunto remanescente de problemas. Desta forma, durante as simulações realizadas buscou-se controlar as simulações o máximo possível. Nos experimentos nem sempre foram utilizados as mesmas partições de dados, inicializações de pesos, testes de hipótese e diferentes tipos de problemas (previsão e classificação, com diferentes tamanhos e complexidades). Também foi realizada no Capítulo 9 uma simulação utilizando a técnica de planejamento fatorial de experimentos para avaliar o comportamento do método proposto. Neste experimento foi verificado o comportamento do método em diferentes situações, identificada a relevância, interação e inter-relação dos parâmetros de configuração do modelo.

(vi) **Verificar o desempenho da técnica para diferentes tipos de problema – classificação e previsão.** Nos experimentos realizados nesta tese foram investigadas diferentes bases de dados, com diferentes características, tamanhos, números de atributos e classes. As características das bases de dados utilizadas estão descritas no

Capítulo 8. Na análise das simulações realizadas foram considerados o erro de classificação e o erro percentual quadrático obtido pelas redes neurais artificiais otimizadas.

(vii) **Comparação entre a técnica e diferentes métodos de otimização.** O método proposto foi comparado com técnicas de otimização que a compõe de forma isolada e com outro modelo que integra as técnicas *Simulate Annealing*, *Tabu Search* e *Backpropagation*. Os resultados obtidos pelo método proposto foram promissores, porém, em alguns casos, a utilização de micro-evoluções genéticas tornou o método lento em relação às demais abordagens. Isto era esperado e o que também se espera é que a qualidade das soluções obtidas possa compensar o aumento no tempo de processamento. Como a otimização nas redes neurais artificiais é realizada de forma *off-line*, esta característica não deve prejudicar o método.

Pode ser destacado ainda que durante a realização do trabalho, vários artigos científicos foram submetidos para diferentes periódicos e conferências objetivando validar cada uma das contribuições propostas (Zanchettin et al, 2005; Zanchettin e Ludermir, 2005a; Zanchettin e Ludermir, 2005b; Ludermir et al, 2006; Zanchettin e Ludermir, 2006; Zanchettin e Ludermir, 2006a; Zanchettin e Ludermir, 2007; Zanchettin e Ludermir, 2007b; Zanchettin et al, 2007; Zanchettin e Ludermir, 2008). Desta forma, vários preceitos propostos puderam ser avaliados pela comunidade científica, contribuindo fortemente para o desenvolvimento da tese.

Analisando-se as contribuições acima enumeradas, acredita-se que a meta geral desta tese tenha sido alcançada, pelo menos em parte. A consolidação, em longo prazo, das contribuições vindas do método proposto e dos estudos realizados depende de mais estudos teóricos, empíricos e provas analíticas, em adição aos apresentados neste documento. A seção seguinte relaciona algumas das próximas etapas planejadas e visualizadas para um maior embasamento, aprimoramento e versatilidade para o método de otimização proposto.

9.2 Propostas para Novos Trabalhos

Esta seção sugere alguns temas para futuras pesquisas com o método de otimização proposto, as quais foram planejadas a partir de limitações observadas e da visualização de tópicos interessantes ainda não desenvolvidos. As principais extensões deste trabalho devem incluir:

(i) **Investigar o comportamento do método com diferentes funções de esfriamento e custo.** No trabalho foi utilizado somente um tipo de esquema de esfriamento, porém, Lins (2005) obteve desempenhos estatisticamente diferentes em

experimentos utilizando outros esquemas de esfriamentos. Desta forma é possível que o esquema de esfriamento tenha uma influência relevante na convergência do algoritmo. Também se faz necessária uma investigação mais precisa sobre o desempenho da função de custo baseada em operadores multiobjetivos. Nas simulações realizadas estes operadores não apresentaram um bom desempenho. Na literatura este tipo de operador tem apresentado resultados que superam a maioria das funções de custo baseadas em um único objetivo, é necessário investigar o motivo pelo qual este desempenho não se repetiu quando a técnica foi utilizada no algoritmo proposto. É interessante também investigar outros tipos de operadores genéticos e incorporar outros métodos de busca local.

(ii) **Verificar o desempenho da técnica para diferentes tipos de problema.** Nos experimentos realizados foram abordados problemas de previsão e classificação. Porém, ainda não foi investigado o comportamento do método com diferentes medidas de erro. Também é importante investigar o comportamento do método em diferentes tipos de problemas. Problemas com complexidade maior, maior número de atributos, classes e atributos ausentes podem expor deficiências ou potencialidades da técnica proposta.

(iii) **Considerar o tempo de execução do método.** Nas simulações realizadas nesta tese foi verificado que o método apresenta um tempo de processamento próximo ao dos algoritmos genéticos, no pior dos casos. Porém, tal aspecto ainda merece maior atenção, sendo importante estudar e comparar o desempenho de processamento e tempo do método com diferentes técnicas, a fim de encontrar possíveis melhorias e verificar se o desempenho atual é satisfatório em relação aos resultados de classificação e topologia de rede obtidos ao final da otimização. Vale ressaltar, a importância de analisar os tempos de execução separando a fase de otimização global da fase de refinamento local.

(iv) **Investigar o desempenho do método na otimização de outras abordagens conexionistas.** Nesta tese foi analisado o comportamento do método proposto na otimização dos valores dos pesos e conexões de redes neurais artificiais *Multi-Layer Perceptron*. É necessário investigar o comportamento do método em outros tipos de arquiteturas.

(v) **Investigar o desempenho do método como uma técnica de otimização iterativa geral.** O método proposto foi desenvolvido como um algoritmo iterativo geral.

Desta forma, ele pode facilmente implementar uma diversidade de problemas, bastando para esta tarefa, adaptar a representação da solução, função de custo e o mecanismo para gerar novas soluções sobre o espaço de busca. Nesta tese somente foi focado um tipo de problema, a otimização simultânea da topologia e dos valores dos pesos das conexões entre as unidades de processamento de redes neurais artificiais *Multi-layer Perceptron*. Assim é necessário verificar o desempenho do método sobre outras classes de problemas.

Referências Bibliográficas

Abraham A e Nath B. (2000) “Optimal Design of Neural Nets using Hybrid Algorithms”, *PRICAI 2000*, Melbourne, pp. 510–20.

Abraham, A. e Nath, B. (2000a). “Hybrid Intelligent Systems Design: A Review of a Decade of Research”, technical report, School of Computing and Information Technology, Monash University, Australia, http://citeseer.nj.nec.com/523093.html.

Aha, D. W. e Bankert, R. L. (1995) “A Comparative Evaluation of Sequential Feature Selection Algorithms” *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics*, vol. 1, n. 7, pp. 13–40.

Almuallim, H. e Dietterich, T. G. (1992) “Efficient Algorithms for Indetifying Relevant Features”, *Proceedings of the 9th Canadian Conference on Artificial Intelligence*, pp. 35–45.

Arantes Jr., G. M., França, F. M. G. e Martinhon, C. A. (2005) “Gerando Orientações Acíclicas com Algoritmos Randômicos Distribuídos”, *Pesquisa Operacional*, vol. 25, pp. 301–312.

Bäck, T., Fogel, D. e Michalewicz, Z. (1997). “Handbook of Evolutionary Computation”, Oxford Univ. Press, 1130p.

Baker, J. E. (1987). “Reducing Bias and Inefficiency in the Selection Algorithm”, *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, Lawrence Erlbaum Associates, pp. 14–21.

Bartlett, P.L. (1997) “For Valid Generalization, the Size of the Weights is more Important than the Size of the Network, em Mozer, M.C., Jordan, M.I., and Petsche, T., (eds.) *Advances in Neural Information Processing Systems*, The MIT Press, vol. 9, pp. 134–140.

Beale, R. e Jackson, T. (1990). “Neural Computing - An Introduction”, Adam Hilger Publishers, 256p.

Blum, C. e Roli, A. (2003). “Metaheuristics in Combinatorial Optimisation: Overview and Conceptual Comparison”, ACM Computing Surveys (CSUR), vol. 35, n. 3, 268–308.

Box, G. e Jenkins, G. (1970). “Time Series Analysis, Forcasting and Control”, *Holden Day*, San Francisco, pp. 532–533.

Braga, A.P., Carvalho, A.P.L. e Ludermir, T.B. (2000). “Redes Neurais Artificiais: Teoria e Aplicações”, Rio de Janeiro: Livros Técnicos e Científicos, 262p.

Brassard, G. e Bratley, P. (1996) “Fundamentals of Algorithms”, Prentice Hall, New Jersey, 524p.

Cannas, B., Fanni, A., Marchesi, M. e Pilo, F. (1999). “A Tabu Search Algorithm for Optimal Sizing of Locally Recurrent Neural Networks”, *Proc. of 10th Int. Symp. On Theoretical Electrical Engineering*, Magdeburg, Germany, pp. 267–272.

Chen, M.S. e Liou, R. (1999). “An Efficient Learning Method of Fuzzy Inference System”, *Proceedings of FUZZ-IEEE’99*, Seoul, Korea. pp 634–638.

Coello, C.A.C. (1996). “An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design”, *Tese de Doutorado*, Tulane University, New Orleans, LA, 437p.

Corana, A., Marchesi, M., Martini, C. e Ridella, S. (1987). “Minimizing Multimodal Functions of Continuous Variables with the Simulated Annealing Algorithm”, *ACM Transactions on Mathematical Software*, vol. 13, pp. 262–280.

Davis, L. (1991) “Bit-Climbing, Representational Bias, and Test Suite Design”, *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kauffman, pp. 18–23.

Deb, K. (2001). “Multi-Objective Optimization using Evolutionary Algorithms” John Wiley & Sons. Chichester, England, 419p.

Dowsland, K.A. (1993). “Simulated Annealing”, In Reeves, C.R. (ed), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publications, pp. 20–69.

Du, X., Li, Y., Chen, W., Zhang, Y e Yao, D. (2006). “A Markov Random Field Based Hybrid Algorithm with Simulated Annealing and Genetic Algorithm for Image Segmentation”, *Advances in Natural Computation*, Springer Berlin / Heidelberg, Vol. 4221, pp. 706–715.

Duda, R. O., Hart, P. E. e Stork D. G. (2001). “Pattern Classification”, 2nd ed, Stork-John Wiley and Sons, 654p.

Fisher, R.A. (1936). “The Use of Multiple Measurements in Taxonomic Problems”, *Annals of Eugenics*, n. 7, pp. 179–188.

Forina, M. et al. (1991). “PARVUS - An Extendible Package for Data Exploration, Classification and Correlation”, *Institute of Pharmaceutical and Food Analysis and Technologies*, Via Brigata Salerno, 16147 Genova, Italy.

Fox, B.L. (1992). “Uniting Probabilistic Methods for Optimization”, *Proceedings of the 24th conference on Winter simulation*, pp. 500–505.

García-Pedrajas, N., Ortiz-Boyer, D. e Hervás-Martínez, C. (2006). “An Alternative Approach for Neural Network Evolution with a Genetic Algorithm: Crossover by Combinatorial Optimization”, *Neural Networks*, vol. 19, pp. 514–528

Geman, S., Bienenstock, E. e Doursat, R., (1992) “Neural Networks and the Bias/Variance Dilemma”, *Neural Computation*, vol. 4, pp. 1–58.

Gennari, J. H., Langley, P. e Fisher, D. (1989) “Models of Incremental Concept Formation”, *Artificial Intelligence*, n. 40, pp. 11–62.

Gepperth, A. e Roth, S. (2006). “Applications of Multi-objective Structure Optimization”, *Neurocomputing*, vol. 69, pp. 701–713.

Glover, F. (1986). “Future Paths for Integer Programming and Links to Artificial Intelligence”, *Computation Operations Research*, vol. 13, pp.533-549.

Glover, F. e Laguna, M. (1997). “Tabu Search”, Kluwer academic Publishers, Boston, 408p.

Glover, F.; Laguna, M. (1995). “Tabu Search”, COLIN, R. R. Modern heuristic techniques for combinatorial problems. New York: McGraw-Hill, pp.70-150.

Goldberg, D.E. (1989). “Genetic Algorithms in Search, Optimization and Machine Learning”, Massachusets: Addison-Wesley Co, 432p.

Goonatilake, S. e Khebbal, S. (1995). “Intelligent Hybrid Systems”, John Wiley & Sons, 316p.

Hamm, L., Brorsen, B.W. e Hagan, M. T. (2002). “Global Optimization of Neural Network Weights”, *Proceedings of the 2002 International Joint Conference on Neural Networks*, vol. 2, pp. 1228–1233.

Harp, S. A., Samad, T. e Guha, A. (1989), “Towards the Genetic Synthesis of Neural Networks”, *Proc. of the Third Int'l Conf. on Genetic Algorithms and Their Applications* (J. D. Schaffer, ed.), Morgan Kaufmann, San Mateo, CA, pp. 360–369.

Haykin, S. (2001). “Neural Networks – A Comprehensive Foundation”, New Jersey, Prentice Hall, 842p.

Hinton, G. E. (1986). “Learning Distributed Representations of Concepts”, *Eighth Annual Conference of the Cognitive Science Society*, pp. 1–12.

Holland, J. H. (1986). “Escaping Brittleness: The Possibilities of General-purpose Learning Algorithms Applied to Parallel Rule-based Systems”, In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning II*, Morgan Kaufmann , pp. 593–623.

Hongbo, S., Shuxia L., Degang, G. e Peng, L. (2006). "Genetic simulated annealing algorithm-based assembly sequence planning", *International Technology and Innovation Conference 2006 (ITIC 2006)*, pp. 1573–1579.

Jang, J. S. (1993). “ANFIS: Adaptive-network-based Fuzzy Inference System”, *IEEE Transaction on Systems, Man and Cybernetics*, vol. 23, n. 3, pp. 665–685.

Jiang, T., Li, X. e Kruggel, F. (2000). “Global Optimization Approaches to MEG Source Localization” *Proceedings of the IEEE International Symposium on Bio-Informatics and Biomedical Engineering*, pp.223–230.

John, G., Kohavi, R. e Pfleger, K. (1994) “Irrelevant Features and the Subset Selection Problem”, *Proceedings of the 11th International Conference on Machine Learning*, vol. 1, pp. 121–129.

Karaboga, D. e Kalinli, A. (1997). “Training Recurrent Neural Networks for Dynamic System Identification using Parallel Tabu Search Algorithm”, *12th IEEE Int. Symp. on Intelligent Control*, Istanbul, Turkey, pp.113–118.

Kasabov, N. (1988a). “The ECOS Framework and the 'Eco' Training Method for Evolving Connectionist Systems”, *Journal of Advanced Computational Intelligence*, vol. 2, n. 6, pp. 1–8.

Kasabov, N. (1998b). “Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation”, em Yamakawa T and Matsumoto G (Eds), *Methodologies for the Conception, Design and Application of Soft Computing*, World Scientific, pp. 271–274.

Kasabov, N. (1999). “Evolving Connectionist Systems for On-line, Knowledge-based Learning: Principles and Applications”, *Technical Report TR99/02*, Department of Information Science, University of Otago, 31p.

Kira, K. e Rendell, L. (1992) “Practical Approach to Feature Selection”, *Proceedings of the 9th International Conference on Machine Learning*, Morgan Kaufmann, pp. 2490–2506.

Kirkpatrick, S., Gellat, D.C. e Vecchi, M.P. (1983) “Optimization by Simulated Annealing”, Science, n. 220, pp. 671–680.

Köhn, P. (1996). “Genetic Encoding Strategies for Neural Networks”, *Knowledge-Based Systems*, vol. 2, pp. 947–950.

Koller, D. e Sahami, M. (1996) “Toward Optimal Feature Selection”, *Proceedings of the 13th International Conference on Machine Learning*, pp. 284–292.

Kramer, A. H. e Sangiovanni-Vincentelli, A. (1988). “Efficient Parallel Learning Algorithms for Neural Networks”, em D. S. Touretsky, editor, *Advances in Neural Information Processing Systems I*, San Mateo: Morgan Kaufmann, pp. 40–48.

Lacerda, E.G.M. (2003). “Seleção de Modelos de Redes Neurais RBF via Algoritmos Genéticos”, *Tese de Doutorado*, Centro de Informática, Universidade Federal de Pernambuco.

Langley, P. e Iba, W. (1993) “Average-case Analysis of a Nearest Neighbor Algorithm” *Proceedings of 13th International Conference on Artificial Intelligence*, pp. 889–894.

Lee, H. D. (2005) “Seleção de Atributos Importantes para a Extração de Conhecimento de Bases de Dados”, *Tese de Doutorado* – USP, São Carlos, 154p.

Lehman, E. (1986) “Testing Statistical Hypothesis (2nd ed.)”, Wiley, 600p.

Li, W.D., Ong, S.K., e Nee, A.Y.C. (2002). “Hybrid Genetic Algorithm and Simulated Annealing Approach for the Optimization of Process Plans for Prismatic Parts”, *International Journal of Production Research*, vol. 40, n. 8, pp.1899–1922.

Lins, A.P.S. (2005). “Métodos de Otimização para Definição de Arquiteturas e Pesos de Redes Neurais MLP”, *Dissertação de Mestrado*, Centro de Informática, Universidade Federal de Pernambuco, Recife, PE, 135p.

Liu, H. E e Yu, L. (2005) “Toward Integrating Feature Selection Algorithms for Classification and Clustering”, *IEEE Transactions on Knowledge and Data Engineering*, vol. 4, n. 17, pp. 491–502.

Liu, H. e Motoda, H. (1998) “Feature Selection for Knowledge Discovery and Data Mining”, Kluwer Academic Publishers, 244p.

Liu, H. e Setiono, R. (1996) “Feature Selection and Classification - A Probabilistic Wrapper Approach” *Proceedings of 9th International Conference on Industrial and Engineering Applications of Artificial Intelligence*, pp. 284–292.

Liu, Y., Ma, L. e Zhang, J. (2000). “GA/SA/TS Hybrid Algorithms for Reactive Power Optimization”, *IEEE Power Engineering Society Summer Meeting*, pp. 245–249.

Ludermir, T.B., Yamazaki, A. e Zanchettin, C. (2006). “An Optimization Methodology for Neural Network Weights and Architectures”, *IEEE Transactions on Neural Networks*, IEEE Computer Society, vol. 17, n. 5, pp. 1452–1459.

Lundy, M e Mess, A. (1986). “Convergence of an Annealing Algorithm”, *Mathematical Programming*, vol. 34, pp.111–124.

Mackey, M. C. and Glass. L. (1977). “Oscillation and Chaos in Physiological Control Systems”, *Science*, vol. 197, pp.287–289.

Mantawy, A.H., Abdel-Magid, Y.L. e Selim, S.Z. (1999). “Integrating Genetic Algorithms, Tabu Search, and Simulated Annealing for the Unit Commitment Problem”, *IEEE Transactions on Power Systems*, vol. 14, n. 3, pp. 829-836.

Martí, R. e El-Fallahi, A. (2004). “Multilayer Neural Networks: an Experimental Evaluation of On-line Training Methods”, *Computers & Operations Research*, vol. 31, pp.1491–1513.

Medsker, L.A. e Bailey, D.L. (1992). “Models and Guidelines for Integrating Expert Systems and Neural Networks”, *Hybrid Architectures for Intelligent Systems*, ed. Kandel and Langholz, CRC Press, pp. 153–171.

Metin N.G., Sahiner, B., Chan, H., Hadjiiski, L. e Petrick, N. (2000). “Optimal Selection of Neural Network Architecture for CAD Using Simulated Annealing”, *Proceedings of the 22th Annual EMBS International Conference*, pp. 1325–1330.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M. N., Teller, A. H., e Teller, E. (1953). “Equation of State Calculations by fast Computing Machines”, *The Journal of Chemical Physics*, vol. 21, n. 6, pp. 1087–1092.

Michie, D., Spiegelhalter, D. J. e Taylor, C. C. (1994) “Machine Learning, Neural and Statistical Classification”, Ellis Horwood, 289p.

Mitchell, T. (1997) “Machine Learning”, McGraw-Hill, 352p.

Montgomery, D. C. (1997). “Design and Analysis of Experiments”, Fourth ed., Wiley, New York, 660p.

Montgomery, D.C. e Runger, G.C. (2003). “Estatística Aplicada e Probabilidade para Engenheiros”, Rio de Janeiro: LTC Editora, 2a edição, 476p.

Moody, J.E. (1992) “The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems”, em Moody, J.E., Hanson, S.J., e Lippmann, R.P., *Advances in Neural Information Processing Systems*, vol. 4, pp. 847–854.

Osman, I.H. (1993). “Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem”, *Ann. Operational Research*, vol. 41, n. 1-4, pp. 421–451.

Palmes, P. P., Hayasaka, T. e Usui, S. (2005). “Mutation-Based Genetic Neural Network”, *IEEE Transactions on Neural Networks*, vol. 16, n. 3, pp. 587–600.

Panos Y. Papalambros e Douglass J. Wilde (2000) “Principles of Optimal Design : Modeling and Computation”, Cambridge University Press, 412p.

Porto, V.W., Fogel, D.B., e Fogel, L.J. (1995). “Alternative Neural Network Training Methods”, *IEEE Expert*, vol. 10, n. 3, pp. 16–22.

Prehelt, L. (1994). “PROBEN1 – A Set of Neural Network Benchmark Problems and Benchmarking Rules”, Technical report 21/94, Universität Karlsruhe, 38p.

Quinlan, J. (1987). “Simplifying Decision Trees”, *International Journal of Man-Machine Studies*, vol. 27, pp. 221–234.

Ritchie, B. C. W., Parker, J. S., Hahn, L. W. e Moore, J. H. (2003). “Optimization of Neural Network Architecture using Genetic Programming Improves Detection and Modeling of Gene-gene Interactions in Studies of Human Diseases”, *BMC Bioinformatics*, vol. 4, n. 28, pp. 1–14.

Rizzi, A., Frattale, F. M., & Martinelli, G. (1999). “Automatic Training of ANFIS Networks”, *Proceedings of FUZZ-IEEE’99*, Seoul, Korea. pp 1655–1660.

Rumelhart, D. E., Hinton, G. E. e Williams, R. J. (1986). “Learning Representations by Backpropagation Errors”, *Nature*, vol. 323, pp. 533–536.

Rumelhart, D.E., Weigend, S.A. (1990). “Predicting the Future: A Connectionist Approach”, *International Journal of Neural Systems*, pp. 193–209.

Sait, S.M. e Youssef, H. (1999). “Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems”, *IEEE Computer Society Press*, 387p.

Santoro, D. M. (2005) “Sobre o Processo de Seleção de Subconjuntos de Atributos - As Abordagens Filtro e Wrapper” *Dissertação de Mestrado*, Universidade Federal de São Carlos, 141p.

Santos, M. S. (2000). “Construção de um Nariz Artificial usando Redes Neurais”, *Tese de Doutorado*, Curso de Pós-graduação em Ciência da Computação, Centro de Informática, Universidade Federal de Pernambuco, Recife-PE, 120p.

Sexton, R. J., Alidaee, B., Dorsey, R. E., e Johnson, J. D. (1998). “Global Optimization for Artificial Neural Networks: A tabu search application”, *European Journal of Operational Research*, vol. 106, n. 2-3, pp. 570–584.

Sexton, R.S., Dorsey, R.E., e Johnson, J.D. (1999). “Optimization of Neural Networks: A Comparative Analysis of the Genetic Algorithm and Simulated Annealing”, *European Journal of Operational Research*, vol. 114, pp. 589–601.

Shieh H. e Peralta, R. C. (2005). “Optimal in Situ Bioremediation Design by Hybrid Genetic Algorithm-Simulated Annealing”, *Journal of Water Resources Planning and Management*, vol. 131, n. 1, pp. 67–78.

Skiena, S. (1990) “Breadth-First and Depth-First Search”, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Reading, MA: Addison-Wesley, pp. 95–97.

Smith, M. (1996) “Neural Networks for Statistical Modeling”, Boston: International Thomson Computer Press, 235p.

Smith, K.I., Everson, R.M., e Fieldsend, J.E. (2004). “Dominance Measures for Multi-objective Simulated Annealing”, *Congress on Evolutionary Computation*, vol. 1, pp. 23–30

Souza, J.E.G., Neto, B.B., Santos, F.L., Melo, C.P., Santos, M. S. e Ludermir, T.B. (1999) “Polypyrrole Based Aroma Sensor”, *Synthetic Metals*, vol. 102, pp. 1296-1299.

Stepniewski, S. W. and Keane, A. J. (1997). “Pruning Backpropagation Neural Networks using Modern Stochastic Optimization Techniques”, *Neural Computing and Applications*, vol. 5, pp. 76–98.

Sywerda, G. (1989). “Uniform Crossover in Genetic Algorithms”, *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers Inc, pp. 2–9.

Takagi, T. e Sugeno, M. (1985). “Fuzzy Identification of Systems and its Applications to Modeling and Control”, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, pp. 116–132.

The Mathworks INC. Matlab 6.0 (R12). Fuzzy Logic Toolbox User’s Guide – Version 2, 2000. Disponível em: http://www.mathworks.com. Acesso em: 03 de janeiro de 2008.

Tsai, J., Chou, J. e Liu, T. (2006). “Tuning the Structure and Parameters of a Neural Network by Using Hybrid Taguchi-Genetic Algorithm”, *IEEE Transactions on Neural Networks*, vol. 17, n. 1, pp. 69–80.

Vafaie, H. e De Jong, K. (1993) “Robust Feature Selection Algorithms”, *Proceedings of the Fifth IEEE International Conference on Tools for Artificial Intelligence*, pp. 356–363.

Weigend, A. (1994) “On Overfitting and the Effective Number of Hidden Units”, *Proceedings of the 1993 Connectionist Models Summer School*, pp. 335–342.

Weigend, A. S., Rumelhart, D. E. e Huberman, B. A. (1991). “Generalization by Weight Elimination with Application to Forecasting”. em J. E. M. . D. S. T. Lippmann, editor, *Advances in Neural Information Processing Systems III*, San Mateo: Morgan Kaufmann, pp. 875–882.

Wolpert, D. H. e Macready, W. G. (1997). “No Free Lunch Theorems for Optimization”, *IEEE Transactions on Evolutionary Computation*, vol.1, pp. 67–82.

Yamazaki, A. (2004).“Uma Metodologia para Otimização de Arquiteturas e Pesos de Redes Neurais”, *Tese de Doutorado*, Centro de Informática, Universidade Federal de Pernambuco, 135p.

Yamazaki, A., de Souto, M., e Ludermir, T.B. (2002). “Optimization of Neural Network Weights and Architectures for Odor Recognition using Simulated Annealing”, *Proceedings International Joint Conference on Neural Networks*, pp. 547–552.

Yao, X. (1999). “Evolving Artificial Neural Networks”, *Proceedings of the IEEE*, vol. 87, n. 9, pp. 1423–1447.

Yogeswaran, M., Ponnambalam, S. G., Tiwari, M. K. (2007). “An Hybrid Heuristic using Genetic Algorithm and Simulated Annealing Algorithm to Solve Machine Loading Problem in FMS”, *IEEE International Conference on Automation Science and Engineering*, pp. 182-187.

Yu, L. e Liu, H. (2004) “Efficient Feature Selection Via Analysis of Relevance and Redundancy”, *Jornal of Machine Learning Research*, n. 5, pp. 1205–1224.

Zanchettin, C. e Ludermir, T.B. (2005a). “Hybrid Technique for Artificial Neural Network Architecture and Weight Optimization”, *Proceedings of the 16th European Conference on Machine Learning (ECML) and the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD).* Springer-Verlag Lecture Notes in Artificial Intelligence. Porto - Portugal, vol. 3721. p. 709–716.

Zanchettin, C. e Ludermir, T.B. (2005b). “Técnica Híbrida de Otimização para Pesos e Arquiteturas de Redes Neurais Artificiais”, *Anais do Encontro Nacional de Inteligência Artificial (ENIA 2005)*. São Leopoldo - Brasil, vol. 1, pp. 47–52.

Zanchettin, C. e Ludermir, T.B. (2006). “The Influence of Different Cost Functions in Global Optimization Techniques”, *Proceedings of the Brazilian Symposium on Neural Networks (SBRN 2006).* IEEE Computer Society, Ribeirão Preto – Brasil, vol. 1, pp. 17–23.

Zanchettin, C. e Ludermir, T.B. (2006a). “A Methodology to Training and Improve Artificial Neural Networks` Weights and Connections”, *Proceedings of the Internation Joint Conference on Neural Networks (IJCNN 2006)*, Vancouver – Canada, vol. 1, pp. 10723–10730.

Zanchettin, C. e Ludermir, T.B. (2007). “Global Optimization Methods for Designing and Training Feedforward Artificial Neural Networks”, *Dynamics of Continuous, Discrete & Impulsive Systems, Series A: Mathematical Analysis – A Supplement*, Advances in Neural Networks, Watam Press, vol. 14(S1), pp. 328–337.

Zanchettin, C. e Ludermir, T.B. (2007b). “Comparison of the Effectiveness of Different Cost Functions in Global Optimization Techniques”, *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2007)*, IEEE Computer Society, Orlando – EUA, vol. 1, pp. 2701–2706.

Zanchettin, C., Minku, F.L. e Ludermir, T.B. (2005). “Design of Experiments in Neuro-Fuzzy Systems”, *Proceedings of the Hybrid Intelligent Systems (HIS2005)*, IEEE Computer Society, Rio de Janeiro – Brasil, pp. 218–226, 2005.

Zanchettin, C., Minku, F.L. e Ludermir, T.B. (2007). “Design of Experiments in Neuro-Fuzzy Systems” (submetido - artigo convidado pelo jornal). *Journal of Intelligent & Fuzzy Systems*, IOS Press.

Zanchettin, C. e Ludermir, T.B. (2008). “Feature Subset Selection in a Methodology for Training and Improving Artificial Neural Network Weights and Connections” (Artigo submetido), *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2008)*, IEEE Computer Society, Hong Kong – China.

Apêndice A

Planejamento Fatorial de Experimentos

O planejamento fatorial de experimentos (Montgomery, 1997) fornece um método para investigar simultaneamente os efeitos de variáveis múltiplas (fatores) em uma variável de saída (resposta). Este estudo consiste em uma série de verificações, ou testes, nas quais alterações deliberadas são realizadas para inserção de variáveis ou fatores. Os resultados destas alterações são coletados em cada verificação para avaliação do comportamento do método investigado.

Abaixo são apresentados os principais conceitos das técnicas de planejamento e análise de experimentos:

Variáveis de resposta: são as variáveis dependentes que sofrem efeito nos testes, quando estímulos são introduzidos propositalmente nos fatores que regulam ou ajustam os processos.

Fatores de controle: são os fatores alterados deliberadamente no experimento. O objetivo principal de introduzir estímulos nos fatores de controle é avaliar o efeito produzido nas variáveis de resposta e, com isso poder determinar os principais fatores do processo.

Fatores de ruído: são os fatores, conhecidos ou não, que influenciam nas variáveis de resposta do experimento.

Níveis dos fatores: são as condições de operação dos fatores de controle investigados nos experimentos. Os níveis são identificados por baixo (-1) nível alto (+1).

Tratamentos: é a combinação dos níveis de fatores de controle, isto significa que cada uma das execuções do experimento representará um tratamento.

Efeito principal: é a diferença média observada na resposta quando se muda o nível do fator investigado.

Efeito da interação: é a metade da diferença entre os efeitos principais de um fator nos níveis de outro fator.

Matriz de experimentos: é o plano formal construído para conduzir os experimentos. Nesta matriz são incluídos os fatores de controle, os níveis e tratamentos do experimento.

Aleatorização: é o processo de definir a ordem dos tratamentos da matriz experimental, através de sorteios por limitações específicas dos testes.

Repetição: é o processo de repetir cada uma das combinações da matriz experimental sob as mesmas condições de experimentação. Este conceito permite encontrar uma estimativa do erro experimental, que é utilizado para determinar se as diferenças observadas entre os dados são estatisticamente significativas.

Blocos: é a técnica utilizada para controlar e avaliar a variabilidade produzida pelos fatores perturbadores (controláveis ou não) dos experimentos.

A.1 Planejamento Fatorial

No caso de haver vários fatores potencialmente importantes em um experimento, um planejamento fatorial de experimentos deve ser utilizado. Um experimento fatorial é aquele em que os fatores são variados em conjunto. Segundo Montgomery e Runger (2003), "por um planejamento fatorial, queremos dizer que, em cada tentativa completa ou réplica do experimento, todas as combinações possíveis dos níveis dos fatores são investigadas".

Para ilustrar o funcionamento desta técnica, considere-se um experimento com dois fatores (A e B), cada um destes parâmetros serão testados com *a* níveis para o fator *A* e *b* níveis para o fator *B*. Desta forma, existem *ab* combinações de teste com *n* réplicas. A matriz de planejamento para este experimento é apresentada na Tabela . Essa organização também representa o caso geral do experimento fatorial de dois fatores, para uma resposta y_{ijk}, observada quando o fator *A* está no *i*-ésimo nível ($i=1,2,...,a$).

Tabela A.1 Arranjo dos dados para um Planejamento de Experimentos com Dois Fatores (Montgomery, 1991)

		Fator B					
		1	2	...	*b*	Totais	Médias
Fator A	1	$y_{111},y_{112},..,y_{11n}$	$y_{121},y_{122},..,y_{12n}$		$y_{1b1},y_{1b2},..,\ y_{1bn}$	$y_{1..}$	$y_{1..}$
	2	$y_{211},y_{212},..,y_{21n}$	$y_{221},y_{222},..,y_{22n}$		$y_{2b1},y_{2b2},..,\ y_{2bn}$	$y_{2..}$	$Y_{2..}$
	.						
	.						
	.						
	a	$y_{a11},y_{a12},..,y_{a1n}$	$y_{a21},y_{a22},..,y_{a2n}$		$y_{ab1},y_{ab2},..,\ y_{abn}$	$y_{a..}$	$y_{a..}$
	Totais	$y_{.1.}$	$y_{.2.}$		$y_{.b.}$	$y...$	
	Médias	$y_{.1.}$	$y_{.2.}$		$y_{.b.}$		$y...$

Segundo Montgomery (1997), com o experimento organizado desta forma é possível verificar se:

(i) A resposta é alterada significativamente quando muda o nível do fato A;

(ii) A resposta é alterada significativamente quando muda o nível do fator B;

(iii) A interação dos fatores (coluna x linha) altera significativamente a resposta.

Os autores também definem que o modelo estatístico do planejamento fatorial é dado pela Equação A.1.

$$y_{ijk} = \mu + \tau_i + \beta_j + (\tau\beta)_{ij} + \varepsilon_{ijk} \qquad \text{(A.1)}$$

sendo que,

μ é a média dos resultados

τ_i é o efeito principal do fator A

β_j é o efeito principal do fator B

$(\tau\beta)_{ij}$ é o efeito da interação dos fatores A e B

ε_{ijk} é o erro experimental

Um dos métodos que pode ser utilizado para determinar os coeficientes da Equação A.1 é a Análise de Variância (ANOVA), que também pode ser utilizada para verificar se os efeitos são significativos nas respostas.

Alguns valores devem ser calculados para que se obtenha essa análise de variância. As representações do total das observações feitas no *i*-ésimo nível do fator A, $y_{i..}$, e suas médias correspondentes são dadas pelas Equações A.2 e A.3.

$$y_{i..} = \sum_{j=1}^{b}\sum_{k=1}^{n} y_{ijk} \qquad \text{(A.2)} \qquad \bar{y}_{i..} = \frac{y_{i..}}{bn} \qquad \text{(A.3)} \qquad i = 12,...,a$$

As representações do total das observações feitas no *i*-ésimo nível do fator B, $y_{.j.}$, e suas médias correspondentes são dadas pelas Equações A.4 e A.5.

$$y_{.j.} = \sum_{i=1}^{a}\sum_{k=1}^{n} y_{ijk} \qquad \text{(A.4)} \qquad \bar{y}_{.j.} = \frac{y.j.}{an} \qquad \text{(A.5)} \qquad j = 1,2,...,b$$

As representações do total das observações feitas na *ij*-ésima célula da Tabela A.1, $y_{ij.}$, e suas médias correspondentes são dadas pelas equações A.6 e A.7.

$$y_{ij.} = \sum_{k=1}^{n} y_{ijk} \qquad \text{(A.6)} \qquad \bar{y}_{ij.} = \frac{y_{ij.}}{n} \qquad \text{(A.7)} \qquad i = 1,2,...,a; \;\; j = 1,2,...,b$$

As representações do total global de todas as observações, $y_{...}$, e suas médias correspondentes são dadas pelas Equações A.8 e A.9.

$$y_{...} = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n} y_{ijk} \qquad \text{(A.8)} \qquad \bar{y}_{...} = \frac{y_{...}}{abn} \tag{A.9}$$

Conforme dito anteriormente, a análise da variância testará três hipóteses: a de nenhum efeito principal do fator A, a de nenhum efeito principal do fator B, e de nenhum efeito de interação AB. Para isso, tal procedimento de análise fará a decomposição da variabilidade total nos dados em partes componentes, comparando, então, os vários elementos nessa decomposição.

A *variabilidade total* é medida pela soma quadrática total das observações:

$$SQ_T = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n} (y_{ijk} - \bar{y}_{...})^2 \tag{A.10}$$

Sendo que a decomposição da soma quadrática (ou seja, a decomposição da variabilidade total, a qual testa as hipóteses acima) é dada por:

$$\begin{aligned} SQ_T = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n} (y_{ijk} - \bar{y}_{...})^2 = bn\sum_{i=1}^{a} (\bar{y}_{i..} - \bar{y}_{...})^2 + an\sum_{j=1}^{b} (\bar{y}_{.j.} - \bar{y}_{...})^2 + \\ + n\sum_{i=1}^{a}\sum_{j=1}^{b} (\bar{y}_{ij.} - \bar{y}_{i..} - \bar{y}_{.j.} + \bar{y}_{...})^2 + \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n} (y_{ijk} - \bar{y}_{ij.})^2 \end{aligned} \tag{A.11}$$

A equação acima é chamada de a *identidade da soma quadrática para a análise de variância com dois fatores*, e pode ser escrita simbolicamente como:

$$SQ_T = SQ_A + SQ_B + SQ_{AB} + SQ_E \tag{A.12}$$

Nesta Equação A.11, SQ_A representa a soma quadrática devido ao fator A, SQ_B, a soma quadrática devido ao fator B, SQ_{AB} representa a soma quadrática devido à interação entre A e B, e SQ_E representa a soma quadrática devido ao erro.

As Equações A.13 a A.17 fornecem os cálculos para as **somas quadráticas** em uma análise de variância com dois fatores:

$$SQ_T = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n} y_{ijk}^2 - \frac{y_{...}^2}{abn} \tag{A.13}$$

$$SQ_A = \sum_{i=1}^{a} \frac{y_{i..}^2}{bn} - \frac{y_{...}^2}{abn} \tag{A.14}$$

$$SQ_B = \sum_{j=1}^{b} \frac{y_{.j.}^2}{an} - \frac{y_{...}^2}{abn} \tag{A.15}$$

$$SQ_{AB} = \sum_{i=1}^{a}\sum_{j=1}^{b} \frac{y_{ij.}^2}{n} - \frac{y_{...}^2}{abn} \quad - SQ_A - \quad SQ_B \tag{A.16}$$

$$SQ_E = SQ_T - SQ_{AB} - SQ_A - SQ_B \tag{A.17}$$

Os efeitos principais de A, B, e o efeito de interação possuem, respectivamente, (*a-1*), (*b-1*) e (*a-1*)(*b-1*) **graus de liberdade**. No total, existem (*abn-1*) graus de liberdade. Dentro de cada uma das células *ab* da Tabela A.1, há *n-1* graus de liberdade entre as *n* réplicas, podendo haver alguma diferença devido ao erro aleatório. Conseqüentemente, há (*ab*)(*n-1*) graus de liberdade para o erro. Portanto, os graus de liberdade são divididos de acordo com:

$$abn\text{-}1 = (a\text{-}1) + (b\text{-}1) + (a\text{-}1)(b\text{-}1) + (ab)(n\text{-}1) \qquad \text{(A.18)}$$

Em geral, uma **média quadrática** é calculada dividindo-se uma soma quadrática por seu número de graus de liberdade. Sendo assim, as Equações A.19 a A.22 fornecem as médias quadráticas do planejamento de experimentos em questão.

$$MQ_A = \frac{SQ_A}{a-1} \qquad \text{(A.19)}$$

$$MQ_{AB} = \frac{SQ_{AB}}{(a-1)(b-1)} \qquad \text{(A.20)}$$

$$MQ_B = \frac{SQ_B}{b-1} \qquad \text{(A.21)}$$

$$MQ_E = \frac{SQ_E}{ab(n-1)} \qquad \text{(A.22)}$$

Finalmente, para testar o fato de os efeitos dos fatores nas linhas serem iguais a zero, é utilizada a razão: $F_0 = MQ_A/MQ_E$, que terá distribuição *f*, com (a-1) e ab(n-1) graus de liberdade se a hipótese nula for verdadeira. A hipótese nula será rejeitada com um nível de significância μ se $f_0 > f_{\mu,\,(a-1),\,ab(n-1)}$. Testes similares devem ser feitos para as outras hipóteses nulas: $F_0 = MQ_B/MQ_E$ e $F_0 = MQ_{AB}/MQ_E$.

Tese de Doutorado apresentada por **Cleber Zanchettin** a Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Otimização Global em Redes Neurais Artificiais"**, orientada pela **Profa. Teresa Bernarda Ludermir** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Francisco de Assis Tenório de Carvalho
Centro de Informática / UFPE

Prof. Tsang Ing Ren
Centro de Informática / UFPE

Prof. Marcilio Carlos Pereira de Souto
Departamento de Informática e Matemática Aplicada / UFRN

Prof. Andre Carlos Ponce de Leon Ferreira de Carvalho
Instituto de Ciências Matemática e Computação / USP

Prof. Gerson Zaverucha
Programa de Engenharia de Sistemas / UFRJ

Visto e permitida a impressão.
Recife, 21 de fevereiro de 2008.

Prof. FRANCISCO DE ASSIS TENÓRIO DE CARVALHO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.