Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia Elétrica

Algoritmos Genéticos e Processamento Paralelo aplicados à Definição e Treinamento de Redes Neurais Perceptron de Múltiplas Camadas

Ana Claudia Medeiros Lins de Albuquerque

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Norte, como parte dos requisitos necessários para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: **Prof. Dr. Jorge Dantas de Melo**
Co-orientador: **Prof. Dr. Adrião Duarte Dória Neto**

Natal-RN, Fevereiro de 2005

Divisão de Serviços Técnicos

Catalogação da Publicação na Fonte. UFRN / Biblioteca Central Zila Mamede

Albuquerque, Ana Claudia Medeiros Lins de.
Algoritmos genéticos e processamento paralelo aplicados à definição e treinamento de redes neurais perceptron de múltiplas camadas / Ana Claudia Medeiros Lins de Albuquerque. – Natal, RN, 2005.
76 f. : il.

Orientador : Jorge Dantas de Melo.
Co-orientador : Adrião Duarte Dória Neto.

Dissertação (Mestrado) – Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica.

1. Redes neurais – Tese. 2. Redes neurais artificiais - Tese. 3. Algoritmos genéticos – Tese. 4. Computação evolutiva – Tese. 5. Processamento paralelo – Tese. I. Melo, Jorge Dantas de. II. Dória Neto, Adrião Duarte. III. Título.

RN/UF/BCZM CDU 004.032.26 (043.2)

Ana Claudia Medeiros Lins de Albuquerque

Algoritmos Genéticos e Processamento Paralelo aplicados à Definição e Treinamento de Redes Neurais Perceptrons de Múltiplas Camadas

Tese de mestrado aprovada em 1 de fevereiro de 2005 pela Comissão Examinadora formada pelos seguintes membros:

Prof. Jorge Dantas de Melo

Prof. Adrião Duarte Dória Neto

Prof. Leandro dos Santos Coelho

Prof. André Laurindo Maitelli

Prof. Estefane George Macedo de Lacerda

Natal, 1 de fevereiro de 2005

Para Juliana Medeiros Lins de Albuquerque (*In memorian*).

Agradecimentos

Ao Prof. Jorge Dantas de Melo e ao Prof. Adrião Duarte Dória Neto pelo estímulo, amizade e dedicação demonstrados na orientação e desenvolvimento deste trabalho.

Aos meus pais, Eudenilson e Rosa, e às minhas irmãs Juliana (*In memorian*) e Ana Beatriz, pelo carinho, compreensão, apoio e incentivo.

Aos amigos, Cynthia, Marcelo, Marconi, Miriam e Valnaide, por suas valiosas e verdadeiras amizades.

A Wedson, pelas sugestões e colaborações.

A CAPES, pelo apoio financeiro.

Resumo

Neste trabalho foi desenvolvido um algoritmo genético paralelo cooperativo com diferentes comportamentos evolutivos para o treinamento e definição de redes neurais Perceptron de Múltiplas Camadas. As redes neurais Perceptron de Múltiplas Camadas são ferramentas poderosas e tiveram seu uso intensificado já que são capazes de proporcionar bons resultados para diversas aplicações. A combinação de algoritmos genéticos e de processamento paralelo aplicados no processo de treinamento e na definição de redes neurais Perceptron de Múltiplas Camadas é interessante uma vez que o processo de aprendizagem geralmente é lento e a maioria dos algoritmos de treinamento existente realiza apenas o ajuste dos pesos sinápticos da rede neural. Sabe-se que, sem conhecimento prévio da aplicação, é difícil definir uma arquitetura ideal para a rede neural. Desta maneira, tem-se que técnicas para automatizar a definição da arquitetura de redes neurais são de interesse. Além disso, o uso de cooperação no algoritmo genético permite a exploração de áreas promissoras do espaço de busca encontradas por diferentes populações, pode evitar mínimos locais e possibilita a re-introdução nas populações de informações previamente perdidas. Por fim, através da incorporação de diferentes comportamentos evolutivos, intensifica-se a diversidade dos indivíduos e, assim, a busca por uma solução promissora.

Abstract

In this work, it was developed a parallel cooperative genetic algorithm with different evolution behaviors to train and to define architectures for Multilayer Perceptron neural networks. Multilayer Perceptron neural networks are very powerful tools and had their use extended vastly due to their ability of providing great results to a broad range of applications. The combination of genetic algorithms and parallel processing can be very powerful when applied to the learning process of the neural network, as well as to the definition of its architecture since this procedure can be very slow, usually requiring a lot of computational time. Also, research work combining and applying evolutionary computation into the design of neural networks is very useful since most of the learning algorithms developed to train neural networks only adjust their synaptic weights, not considering the design of the networks architecture. Furthermore, the use of cooperation in the genetic algorithm allows the interaction of different populations, avoiding local minima and helping in the search of a promising solution, accelerating the evolutionary process. Finally, individuals and evolution behavior can be exclusive on each copy of the genetic algorithm running in each task enhancing the diversity of populations.

Sumário

Lista de Figuras

Capítulo 1

Introdução

1.1 Apresentação

As redes neurais artificiais são ferramentas poderosas, apresentando um modelo matemático inspirado na estrutura neural de organismos inteligentes e adquirindo conhecimento através da experiência. Mais especificamente, as redes neurais Perceptron de Múltiplas Camadas constituem uma importante classe de redes neurais, com propriedades relevantes, tais como simplicidade, versatilidade, eficiência e precisão computacional. Além disso, por possuírem um alto grau de aplicabilidade, as redes neurais Perceptron de Múltiplas Camadas são cada vez mais utilizadas em diferentes áreas. [1].

Entretanto, sabe-se que, na maioria das vezes, a definição de uma arquitetura específica para uma rede neural artificial é uma tarefa árdua. Pode-se até afirmar que a escolha de uma arquitetura para determinadas aplicações é, basicamente, um processo de tentativa e erro, dependendo, em grande parte, de experiências passadas do projetista com problemas similares [2].

O problema de definição de arquiteturas também se aplica às redes neurais Perceptron de Múltiplas Camadas. Algoritmos de treinamento desenvolvidos, tais como o algoritmo de retro-propagação de erro, gradiente conjugado, Levenberg-Marquardt e quase-Newton, são utilizados apenas no ajuste dos pesos sinápticos da rede neural. Desta maneira, técnicas para automatizar a definição da arquitetura de redes neurais são de grande interesse e candidatas naturais à aplicação de algoritmos evolutivos.

Diversas pesquisas onde computação evolutiva é combinada e aplicada à definição de arquiteturas para redes neurais artificiais vêm sendo realizadas [3], [4], [5]. Neste trabalho, os algoritmos genéticos são utilizados, simultaneamente, no treinamento e na definição da arquitetura da rede neural. Algumas das conclusões da combinação de algoritmos genéticos e processamento paralelo ao treinamento e definição de redes neurais Perceptron de Múltiplas Camadas obtidas no desenvolvimento deste trabalho encontram-se publicadas na literatura especializada [6], [7], [8], [9].

Redes neurais e algoritmos genéticos podem ser combinados de modo que uma população de redes neurais possa competir entre si. Cada indivíduo da população codifica uma rede neural específica com diferentes arquiteturas e pesos sinápticos. Aqueles indivíduos que codificam as melhores configurações de redes neurais são combinados e passados à geração seguinte. Após um certo número de iterações, uma configuração final e otimizada da rede neural pode ser obtida.

Mesmo com a utilização de algoritmos genéticos, tem-se que o processo de aprendizagem de Perceptron de Múltiplas Camadas pode ser lento, variando conforme o tamanho da rede neural e da configuração adotada para o algoritmo genético. Desta maneira, o processamento paralelo é incorporado

ao algoritmo genético desenvolvido visando minimizar o tempo demandado na etapa de treinamento da rede.

1.2 Objetivos

O objetivo deste trabalho consiste no desenvolvimento de um algoritmo para realizar o treinamento de redes neurais Perceptron de Múltiplas Camadas e, simultaneamente, realizar a definição automática de arquitetura para a rede.

Apesar das redes neurais Perceptron de Múltiplas Camadas serem amplamente aplicadas na resolução de diversos problemas através do seu treinamento com o algoritmo de retro-propagação de erro, tem-se que tanto a robustez quanto a velocidade desse algoritmo são sensíveis à escolha de diversos parâmetros de treinamento, tais como taxa de aprendizado e constante do momento. Além disso, o algoritmo de retro-propagação de erro não pode ser aplicado na definição automática de arquiteturas para a rede.

Os algoritmos genéticos, por sua vez, quando aplicados ao treinamento de redes neurais Perceptron de Múltiplas Camadas apresentam mais flexibilidade que o algoritmo de retro-propagação de erro tanto em termos de estrutura quanto no aprendizado de parâmetros da rede. Além disso, são menos susceptíveis a mínimos locais e podem ser aplicados ao problema de definição automática de arquitetura para redes neurais.

Sendo assim, utiliza-se, neste trabalho, algoritmos genéticos para o ajuste dos pesos sinápticos e para a definição de arquitetura de redes neurais Perceptron de Múltiplas Camadas.

Vale salientar, também, que o treinamento de redes neurais Perceptron de Múltiplas Camadas pode ser lento, principalmente no uso de arquiteturas complexas. Com o intuito de acelerar esta etapa, utiliza-se, no algoritmo genético, processamento paralelo.

Desta forma, diversos processos paralelos são criados, cada um deles correspondendo a uma população diferente. Todas as populações criadas evoluem simultaneamente. Ao final de um número pré-determinado de gerações, as diferentes populações comunicam-se e trocam informações a respeito dos melhores indivíduos selecionados. Este procedimento continua até que o sinal de erro produzido pela rede neural seja aproximadamente zero e, conseqüentemente, a mesma possa ser considerada treinada. Ao final do processo de treinamento, o indivíduo selecionado contém o conjunto de pesos sinápticos da rede neural, assim como sua arquitetura.

Além de processamento paralelo, são introduzidos, ao algoritmo genético desenvolvido, trocas de informações entre as populações e diferentes critérios de reprodução.

A troca de informações entre as populações é importante, permitindo a cooperação e a exploração de áreas promissoras do espaço de busca encontradas por outras populações. Além disso, através da cooperação, mínimos locais são evitados e o material genético previamente perdido é re-introduzido às populações.

O uso de diferentes comportamentos evolutivos, por sua vez, contribui na manutenção da diversidade dos indivíduos, intensificando, desta maneira, a busca por uma solução promissora.

Sendo assim, ao final, obtém-se um algoritmo genético paralelo cooperativo e com diferentes comportamentos evolutivos para o treinamento e definição

de arquitetura de redes neurais Perceptron de Múltiplas Camadas.

1.3 Algumas Abordagens para o Treinamento de Redes Neurais Presentes na Literatura

Existem diversos trabalhos publicados na literatura especializada contendo diferentes tipos de abordagens de treinamento de redes neurais artificiais. A seguir, são citados alguns deles.

Em [5] aplica-se neuroevolução e aprendizagem por reforço em problemas de controle. Nele, é desenvolvido um método, *NeuroEvolution of Augmenting Topologies - NEAT*, cujo esquema de codificação é definido de modo a permitir que genes correspondentes sejam facilmente alinhados durante a recombinação na etapa de reprodução. Os genes, por sua vez, são representações da conectividade da rede. Cada gene possui uma lista de conectividade contendo informações sobre o neurônio de origem, o neurônio de destino e o peso sináptico associado. Além disso, faz-se uso de mutações capazes de modificar tanto os pesos sinápticos quanto a estrutura da rede.

Em [10], são utilizados processamento paralelo e o algoritmo de retropropagação de erro no treinamento redes neurais Perceptron de Múltiplas Camadas. Cada tarefa paralela opera sobre uma cópia da rede neural inicializada com conjunto de pesos sinápticos distintos. Após ser subdividido, o conjunto de treinamento é enviado para cada uma das tarefas paralelas. Ao final de um número pré-determinado de épocas, permuta-se informações entre as tarefas paralelas. Em seguida, inicializa-se uma delas com o melhor

conjunto de pesos encontrados e, nas demais, aplica-se heurísticas tais como média de todos os conjuntos de pesos, inserção de vírus e combinação dos elementos.

Em [11] utiliza-se algoritmos genéticos paralelos no treinamento de redes neurais recorrentes aplicadas a problemas de otimização. Nele, são desenvolvidos dois métodos. O primeiro, *Co-operating Populations with Different Evolution Behaviours - CoPDEB*, aumenta o desempenho do algoritmo genético paralelo através da introdução de diferentes comportamentos evolutivos em cada uma das sub-populações existentes. O segundo método, *Varying Fitness Function - VFF*, é aplicado a problemas de otimização restritos e auxilia o algoritmo genético a evitar soluções ruins, além de ajudá-lo a localizar a área de ótimo global mais facilmente.

1.4 Organização da Dissertação

Esta dissertação está organizada em seis capítulos. O Capítulo 1 consiste em uma visão geral do trabalho e de seus objetivos. No Capítulo 2, é realizada uma breve introdução às redes neurais Perceptron de Múltiplas Camadas, são detalhadas algumas vantagens e desvantagens do algoritmo de retro-propagação de erro e de algoritmos genéticos no treinamento de Perceptron de Múltiplas Camadas. Além disso, são ilustradas maneiras de se realizar a validação da rede neural, assim como de se definir, automaticamente, arquiteturas para a rede. No Capítulo 3, é apresentada uma introdução aos algoritmos genéticos. Além disso, mostra-se como treinar e definir arquiteturas para redes neurais Perceptron de Múltiplas Camadas usando algoritmos genéticos, e como aplicar processamento paralelo no treinamento

de redes neurais Perceptron de Múltiplas Camadas com algoritmos genéticos. No Capítulo 4 é detalhada toda a estrutura do algoritmo genético paralelo cooperativo com diferentes comportamentos evolutivos desenvolvidos. No Capítulo 5, são ilustrados os resultados obtidos com o algoritmo desenvolvido em aproximação de funções e na auto-compensação de sensores. Por fim, no Capítulo 6, são apresentadas as conclusões obtidas com o trabalho e as perspectivas de trabalhos futuros.

Capítulo 2

Redes Neurais Perceptron de Múltiplas Camadas

2.1 Introdução

As redes neurais Perceptron de Múltiplas Camadas constituem uma importante classe de redes neurais e foram desenvolvidas para a resolução de problemas complexos, que não poderiam ser resolvidos pelo modelo de neurônio básico proposto por Rosenblatt [12]. A organização em camadas das redes neurais Perceptron de Múltiplas Camadas encontra-se ilustrada na Figura 2.1.

Desta maneira, a rede neural Perceptron de Múltiplas Camadas é formada por uma camada de entrada, uma ou mais camadas intermediárias e uma camada de saída. A camada de entrada é formada pelo conjunto de treinamento da rede neural. As camadas intermediárias e a de saída, por sua vez, são compostas por um ou mais neurônios artificiais. Note que a rede

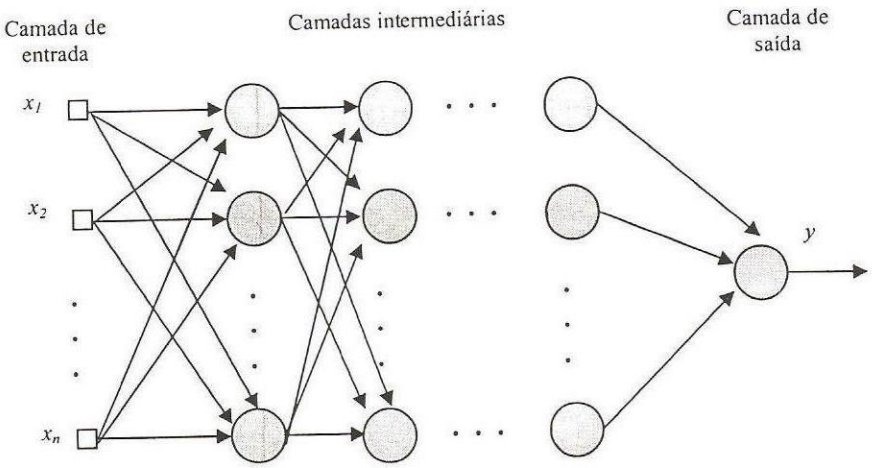

Figura 2.1: Organização em camadas da rede neural Perceptron de Múltiplas Camadas.

neural Perceptron de Múltiplas Camadas utilizada é totalmente conectada. Ou seja, um neurônio artificial em qualquer camada da rede está conectado a todos os outros neurônios artificiais da camada anterior. O fluxo de sinal através da rede, por sua vez, progride para frente, da esquerda para direita e de camada em camada.

Cada neurônio artificial apresenta um conjunto de sinapses, caracterizada por um peso (w_{ij}) próprio. Sendo assim, um sinal x_p localizado na entrada da sinapse p e conectado ao neurônio n é multiplicado pelo peso sináptico w_{np}. Todos os sinais de entrada ponderados pelas respectivas sinapses do neurônio são, em seguida, adicionados por um somador (Σ). A saída do somador (v_n) é a entrada de uma função de ativação ($\varphi(.)$) que, por sua vez, irá produzir a saída do neurônio. Utiliza-se a função de ativação para limitar o intervalo permissível de amplitude do sinal de saída do neurônio a um valor finito. Tipicamente, o intervalo normalizado da amplitude de saída de um neurônio

é escrito como o intervalo unitário fechado [0,1] ou como o intervalo fechado [-1,1]. O bias (b_n) aplicado ao neurônio n tem o seu efeito representado por uma sinapse de peso $w_{n0} = b_n$ conectada a uma entrada fixa igual a +1. A função de ativação ($\varphi(.)$) utilizada foi a sigmóide, mostrada na equação 2.1.

$$y_j = \frac{1}{1 + \exp(-v_j)} \tag{2.1}$$

A Figura 2.2 ilustra a configuração do neurônio artificial.

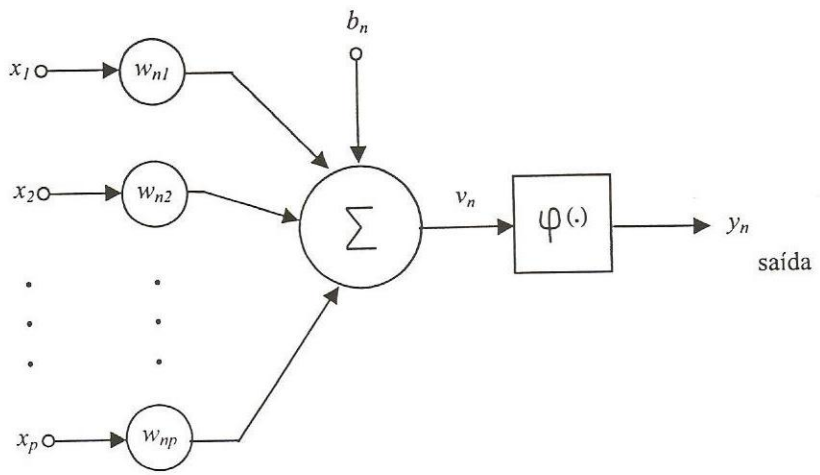

Figura 2.2: Neurônio artificial.

2.2 Algoritmos de Treinamento: Retropropagação de Erro x Algoritmos Genéticos

Usualmente, as redes neurais do tipo Perceptron de Múltiplas Camadas vêm sendo aplicadas com êxito para resolver diversos tipos de problemas

através do seu treinamento de forma supervisionada com o algoritmo de retropropagação de erro. Entretanto, a aplicação de algoritmos genéticos ao ajuste dos pesos sinápticos da rede neural já se popularizou há algum tempo. A utilização de algoritmos genéticos se faz presente, em Redes Neurais Artificiais, tanto na realização do treinamento propriamente dito (ajuste dos pesos sinápticos), quanto para a seleção automática de arquiteturas da rede neural.

Neste trabalho, utiliza-se algoritmos genéticos uma vez que o objetivo do algoritmo desenvolvido consiste em realizar o treinamento da rede neural simultâneo à obtenção automática da arquitetura mais apropriada para uma aplicação específica.

Em linhas gerais, os algoritmos genéticos consistem em métodos de otimização estocásticos baseados em características de seleção natural e evolução biológica. Uma vantagem dos algoritmos genéticos sobre o algoritmo de retropropagação de erro consiste no fato deles apresentarem mais flexibilidade em termos de estrutura e no aprendizado de parâmetros da rede. Além disso, tem-se que algoritmos genéticos, comparados ao algoritmo de retropropagação de erro, são menos susceptíveis a mínimos locais [13].

No algoritmo de retropropagação de erro, o sinal de erro produzido é retro-propagado e os pesos sinápticos são ajustados através do cálculo do gradiente dos mesmos, o que requer diferenciabilidade. Sendo assim, o algoritmo de retropropagação de erro não suporta critérios de otimização não contínuos. Além disso, quando um mínimo global encontra-se cercado de mínimos locais, o algoritmo pode ficar preso a um mínimo local, fazendo com que o resultado do treinamento seja prejudicado. Existe, entretanto, um método de retropropagação de erro com fator de *momentum* que reduz

a sensibilidade do algoritmo de retropropagação de erro a pequenos detalhes presentes na superfície do erro [13], [14], [15]. Este método irá, então, ajudar a rede neural a evitar mínimos locais que muitas vezes impedem a busca de uma solução ideal. Porém, tem-se que tanto a robustez quanto a velocidade de convergência do algoritmo de retropropagação de erro são sensíveis à escolha de diversos parâmetros de treinamento, tais como a taxa de aprendizado e a constante do momento. Os valores desses parâmetros, por sua vez, geralmente mudam de acordo com a aplicação [16], [17].

Por fim, a complexidade do algoritmo de retropropagação de erro cresce exponencialmente à medida que se aumenta o grau de complexidade da aplicação [13]. Além disso, vale ressaltar que este algoritmo apenas realiza o ajuste dos pesos sinápticos da rede neural, não sendo capaz de ser aplicado na definição uma arquitetura ideal para a mesma.

2.3 Generalização

Durante o processo de aprendizagem da rede neural Perceptron de Múltiplas Camadas, aplica-se um conjunto de treinamento e calcula-se os pesos sinápticos da rede utilizando o algoritmo genético desenvolvido. A rede neural obtida, por sua vez, é capaz de realizar corretamente o mapeamento de entrada e saída tanto para os dados de testes utilizados quanto para aqueles que não foram utilizados (capacidade de generalização).

O processo de aprendizagem da rede neural pode ser visto como um problema de ajuste de curva e a própria rede pode ser considerada um mapeamento não linear de entrada e saída do problema. Deste modo, a generalização é conseqüência de uma boa interpolação não linear sobre os dados

de entrada [18]. A rede, por sua vez, realiza boa interpolação fundamentalmente porque o Perceptron de Múltiplas Camadas com funções de ativação contínuas produz funções de saída que também são contínuas.

A Figura 2.3 ilustra um exemplo de generalização em uma rede neural hipotética.

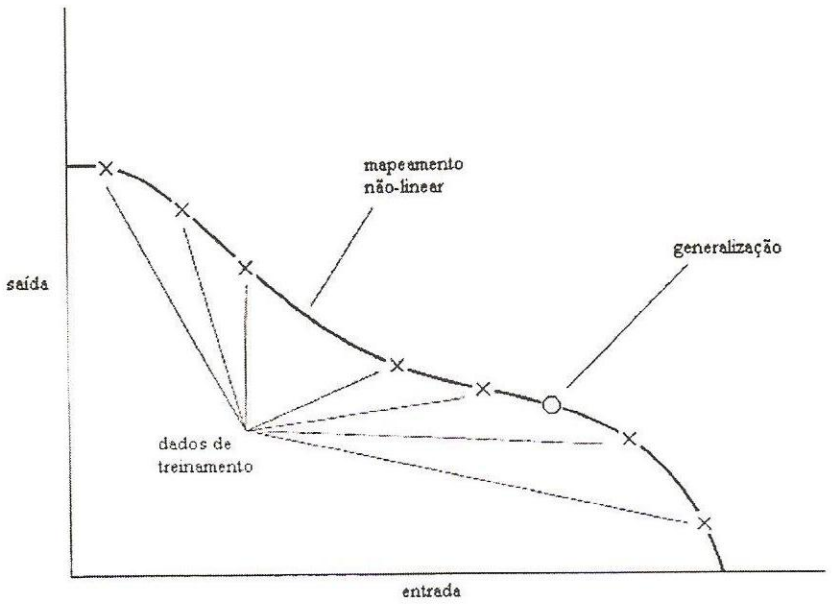

Figura 2.3: Boa generalização, saídas estimadas pela rede adequadamente.

Uma rede neural que apresente uma boa generalização é capaz de produzir um mapeamento de entrada e saída correto mesmo quando a entrada em questão for diferente daquelas fornecidas à rede durante o processo de aprendizagem. A curva ilustrada na Figura 2.3 representa o mapeamento não linear de entrada e saída da rede para os dados de treinamento fornecidos. O ponto marcado com um círculo sobre a curva como generalização, por sua vez, é o resultado da interpolação realizada pela rede.

Vale salientar que quando uma rede neural aprende um número excessivo de exemplos de treinamento pode acabar memorizando os dados. Para tal, basta a rede encontrar uma característica que está presente nos dados de treinamento, mas não na função que deve ser modelada. Sendo assim, treinando-se excessivamente a rede neural (sobre-treinamento), a mesma perde a habilidade de generalizar entre padrões de entrada e saída similares.

A Figura 2.4 ilustra um exemplo em que ocorre uma má generalização pela rede neural devido à memorização para os mesmos dados mostrados na Figura 2.3.

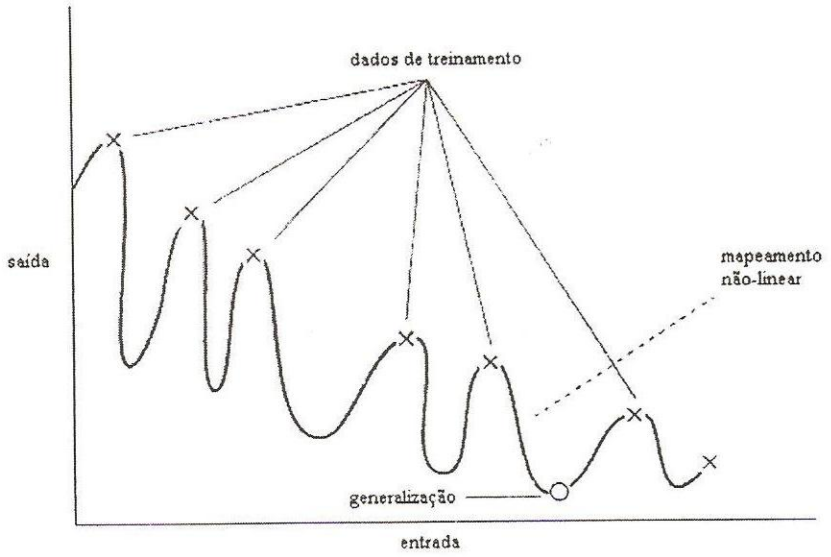

Figura 2.4: Má generalização, saídas estimadas pela rede inapropriadamente.

A generalização é influenciada por três fatores [1]:

1. Tamanho do conjunto de treinamento e o quão representativo ele é;

2. Arquitetura da rede neural;

3. Complexidade física do problema em questão.

No que diz respeito a complexidade física do problema em questão, não se pode ter nenhum tipo de controle. No contexto dos outros dois fatores, entretanto, pode-se ver a questão da generalização sob duas perspectivas distintas [19]:

- A arquitetura da rede é fixa e a questão a ser resolvida é determinar o tamanho do conjunto de treinamento necessário para que ocorra uma boa generalização.
- O tamanho do conjunto de treinamento é fixo e a questão de interesse é determinar a melhor arquitetura para alcançar uma boa generalização.

2.4 Validação Cruzada

Uma rede neural é considerada bem treinada quando aprende o suficiente sobre o passado para que possa generalizar no futuro. Sendo assim, o processo de aprendizagem se transforma em uma escolha de parametrização da rede neural para um conjunto específico de dados. A validação cruzada surge, então, com o objetivo de evitar que a rede neural seja treinada excessivamente e que apresente uma má generalização.

Para realizar a validação da rede neural, divide-se o conjunto de dados, aleatoriamente, em um conjunto de treinamento e em um conjunto de teste. O conjunto de treinamento, por sua vez, é dividido em dois subconjuntos disjuntos. O primeiro deles consiste no subconjunto de estimação, utilizado para selecionar a rede neural. O segundo deles, por sua vez, consiste no subconjunto de validação, utilizado para testar ou validar a rede neural. Desta

maneira, faz-se a validação da rede neural com um conjunto de dados diferente daquele utilizado para estimar os parâmetros. Pode-se, então, utilizar o conjunto de treinamento para avaliar o desempenho dos diferentes modelos de rede neural e, assim, escolher o melhor deles. Entretanto, vale ressaltar que o modelo assim selecionado pode acabar ajustando excessivamente o subconjunto de validação. Daí a importância de analisar o desempenho de generalização do modelo selecionado sobre o conjunto de teste, que é diferente do subconjunto de validação. A validação cruzada pode, ainda, ser utilizada para determinar a rede neural Perceptron de Múltiplas Camadas com o melhor número de neurônios e, também, para determinar o melhor momento de parar o seu treinamento.

2.4.1 Método de Treinamento com Parada Antecipada

Durante o treinamento das redes neurais Perceptron de Múltiplas Camadas, observa-se que o sinal de erro médio quadrático inicia com um valor elevado, decresce rapidamente, e continua a diminuindo lentamente conforme a rede é otimizada em direção a um mínimo local na superfície do erro. Sabe-se, entretanto, que a rede neural pode ser excessivamente ajustada aos dados de treinamento. Neste caso, tem-se que a capacidade de generalização da rede neural é prejudicada.

A escolha do melhor momento para encerrar o treinamento da rede neural é difícil levando-se em consideração apenas a sua a curva de aprendizagem. Porém, o início do excesso de treinamento pode ser identificado através do uso de validação cruzada, onde, conforme mencionado, os dados de treinamento são divididos em um subconjunto de estimação e um subconjunto de validação.

Desta maneira, utiliza-se o subconjunto de estimação para treinar a rede neural, interrompendo-se o processo de aprendizagem periodicamente, após um determinado número de épocas, e testa-se a rede com o subconjunto de validação após cada período de treinamento. Mais especificamente, após um período de estimação (treinamento propriamente dito), os pesos sinápticos e os níveis de bias da rede neural são todos fixos e a rede opera normalmente, em seu modo direto. O erro de validação é, então, medido para cada exemplo do conjunto de validação. Quando a fase de validação é completada, a fase de estimação é reiniciada para um novo período e o processo é repetido. Esse procedimento é denominado de método de treinamento com parada antecipada.

A Figura 2.5 ilustra a regra de parada antecipada baseada na validação cruzada.

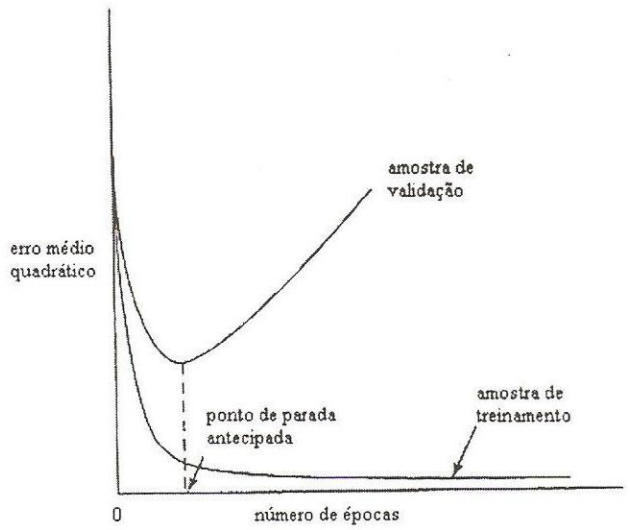

Figura 2.5: Parada antecipada do aprendizado da rede neural baseada na validação cruzada.

Observa-se, na Figura 2.5, as formas conceituais de duas curvas de aprendizagem. Uma das curvas é relativa às medidas sobre o subconjunto de estimação. A outra curva, por sua vez, é relativa às medidas sobre o subconjunto de validação. Tem-se que, tipicamente, o modelo não funciona tão bem sobre o subconjunto de validação, quando comparado ao subconjunto de estimação, sobre o qual o baseia-se o projeto.

Conforme pode ser visto na Figura 2.5, a curva de aprendizagem de estimação decresce monotonicamente para um número crescente de épocas, enquanto que a curva de aprendizagem de validação decresce monotonicamente para um número mínimo e, então, começa a crescer conforme o treinamento continua.

Sendo assim, ao analisar a curva de aprendizagem de estimação separadamente, pode parecer que o desempenho da rede neural possa ser melhorado continuando-se o treinamento, ou seja, indo além do ponto mínimo da curva de aprendizagem de validação. Entretanto, ao continuar o treinamento, tem-se que a rede neural irá aprender, basicamente, o ruído contido nos dados de treinamento. Portanto, esta heurística sugere que o ponto mínimo na curva de aprendizagem de validação seja utilizado como critério de parada do processo de treinamento da rede neural.

2.4.2 Validação Cruzada Múltipla

Quando o conjunto de treinamento não é grande o suficiente, pode-se fazer uso de validação cruzada múltipla [1]. Neste caso, o conjunto disponível de N exemplos é dividido em K subconjuntos, onde $K > 1$ e divisível por N. Em seguida, treina-se o modelo com todos os subconjuntos, com exceção de um deles que é utilizado para medir o erro de validação. O procedimento

é repetido K vezes. Em cada uma das K vezes, utiliza-se um subconjunto diferente para validação, conforme ilustrado na Figura 2.6, onde K vale quatro.

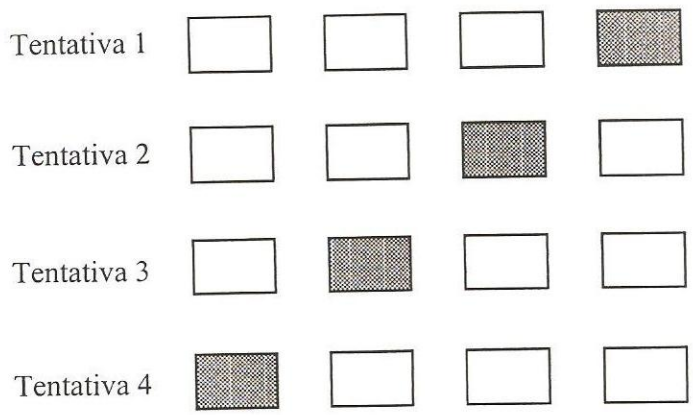

Figura 2.6: Exemplo de validação cruzada múltipla.

Avalia-se o desempenho do modelo realizando-se a média do erro quadrático obtido com a validação sobre todas as tentativas do experimento. Uma vez que o modelo deve ser treinado K vezes, onde $1 < K \leq N$, tem-se que a desvantagem da validação cruzada consiste em requerer uma quantidade excessiva de cálculos. Entretanto, este problema não existe com a utilização do paralelismo, uma vez que os diferentes conjuntos de treinamento são utilizados no processo de aprendizagem das redes neurais simultaneamente.

2.5 Definição de Arquiteturas do Perceptron de Múltiplas Camadas

Usualmente, faz-se necessária a utilização de redes neurais altamente estruturadas e com tamanho elevado para solucionar problemas do mundo real. Entretanto, é menos provável que uma rede neural com tamanho mínimo aprenda os ruídos presentes nos dados treinamento podendo, assim, generalizar melhor sobre novos dados. Além disso, sabe-se que o processo de aprendizagem de redes neurais de tamanhos reduzidos é mais rápido e menos complexo. Por esses motivos, é importante conseguir minimizar o tamanho da rede neural com a manutenção de um bom desempenho da mesma. Tal objetivo pode ser alcançado pelo crescimento da rede, onde se inicia com uma rede neural pequena e adiciona-se, gradativamente, um novo neurônio ou uma nova camada de neurônios enquanto as especificações do projeto ainda não forem satisfeitas ou pela poda da rede, onde se inicia com uma rede neural grande e, em seguida, reduz-se ou elimina-se certos pesos sinápticos de forma seletiva e ordenada [1].

Utilizando os algoritmos genéticos no treinamento da rede neural pode-se, ainda, incorporar à função custo do mesmo, um fator de penalização, cuja função consiste em impossibilitar o crescimento desordenado da rede neural.

No capítulo seguinte, uma análise mais detalhada do uso de algoritmos genéticos na definição de arquiteturas de redes neurais e no treinamento das mesmas é apresentada.

Capítulo 3

Algoritmos Genéticos

3.1 Introdução

Os algoritmos genéticos consistem em métodos estocásticos de busca, baseados nos mecanismos de seleção e evolução natural. Possuem como objetivo encontrar o indivíduo ótimo de uma população geneticamente refinada. O processo de refinamento das populações dá-se de geração a geração, com a renovação da população, obedecendo-se aos critérios probabilísticos de seleção e evolução naturais [4],[20], [21].

Uma implementação de um algoritmo genético inicia-se com uma população de indivíduos gerada aleatoriamente. Cada um desses indivíduos apresenta um material genético, dito cromossomo, que codifica uma solução distinta do problema. O grau de adequabilidade de cada um dos indivíduos da população é medido por uma função custo. Maiores oportunidades de reprodução são fornecidas aos indivíduos mais adaptados segundo a função custo. Neste contexto, a idéia de sobrevivência dos mais adaptados é de extrema

importância aos algoritmos genéticos.

Para codificar as características envolvidas no processo de otimização, freqüentemente são utilizadas cadeias do código binário. Durante a fase de execução, por sua vez, o comprimento da cadeia normalmente permanece fixo, dependendo do grau de precisão requerido para a solução do problema ou da quantidade de características em observação.

Nas técnicas de busca e otimização tradicionais, tais como métodos de Newton-Raphson, Gauss-Seidel, entre outros, uma única solução é iterativamente manipulada através do uso de algumas heurísticas (estáticas) diretamente associadas ao problema a ser solucionado. Algoritmos genéticos, por sua vez, podem realizar buscas paralelas em diferentes áreas do espaço de solução intensificando, desta maneira, a procura pela solução adequada.

3.2 Algoritmos Genéticos e Redes Neurais Artificiais

Os algoritmos genéticos vêm sendo aplicados às redes neurais artificiais para definição da arquitetura da rede, assim como para o ajuste do conjunto de pesos sinápticos das mesmas [4], [6], [7], [8], [20].

A escolha de uma arquitetura de rede neural adequada e específica para uma determinada aplicação consiste em uma tarefa complexa para o usuário. Sendo assim, cada vez mais o uso de algoritmos genéticos na definição de arquiteturas para redes neurais artificiais vem se popularizando.

Além disso, algoritmos genéticos são eficazes quando aplicados à etapa de treinamento de redes neurais artificiais. Apesar de serem utilizados no ajuste

de pesos sinápticos apenas recentemente, a idéia de treinar redes neurais artificiais com algoritmos genéticos pode ser encontrada no livro *Adaptation in Natural and Artificial Systems* escrito em 1975 por John Holland.

3.2.1 Treinamento de Perceptron de Múltiplas Camadas com Algoritmos Genéticos

Existem inúmeros argumentos indicando vantagens na aplicação de algoritmos genéticos ao treinamento de redes neurais Perceptron de Múltiplas Camadas em substituição ao algoritmo de retro-propagação de erro [22]. Um deles é a capacidade de realização de buscas globais no espaço de pesos sinápticos pelos algoritmos genéticos, o que pode evitar os mínimos locais. Outras vantagens dos algoritmos genéticos sobre o algoritmo de retro-propagação consistem no fato deles apresentarem mais flexibilidade em termos de estrutura e do aprendizado de parâmetros da rede e de serem aplicáveis em problema de otimização contínuo e discreto. Entretanto, vale mencionar que algoritmos genéticos não garantem a convergência para a solução ótima.

Durante o treinamento, cada indivíduo de cada população do algoritmo genético codifica uma rede neural específica. Portanto, em cada população existe um número finito de indivíduos apresentando diferentes conjuntos de pesos sinápticos e, geralmente, a mesma arquitetura de rede neural. Ao final do processo de treinamento, os pesos sinápticos encontram-se ajustados e o material genético do indivíduo selecionado contém a configuração final da rede neural.

Os passos principais da implementação do algoritmo genético encontram-

se ilustrados no fluxograma da Figura 3.1.

O algoritmo genético desenvolve-se da seguinte maneira: uma população inicial de indivíduos é gerada aleatoriamente. Cada material genético de um indivíduo, consistindo de um vetor de 0's e 1's, contém uma descrição de uma rede neural específica. Posteriormente, os indivíduos são decodificados e avaliados de acordo com a função custo. Os melhores indivíduos são aqueles cujos valores da função custo são próximos de zero.

Uma função que converte um vetor de 0's e 1's em um número real é criada para decodificar as informações de cada cromossomo do indivíduo.

A função custo, por sua vez, é calculada para cada conjunto de amostras de treinamento adicionando-se todos os sinais de erro obtidos para cada elemento localizado na saída da rede neural e dividindo o resultado pelo número de amostras aplicadas à rede. O sinal de erro, por outro lado, é o resultado da subtração entre uma resposta desejada previamente definida e a resposta estimada pela rede.

Se a condição de parada ainda não for alcançada, o processo de aprendizagem continua e um certo número de indivíduos é escolhido, de acordo com os critérios de seleção, para serem os pais da próxima geração. Em geral, pode ser dito que para formar uma nova população, os indivíduos são selecionados de acordo com sua capacidade de minimizarem a função custo. Deste modo, os indivíduos que apresentam o menor sinal de erro médio quadrático têm uma chance melhor de reprodução, enquanto que os outros são mais prováveis de desaparecerem.

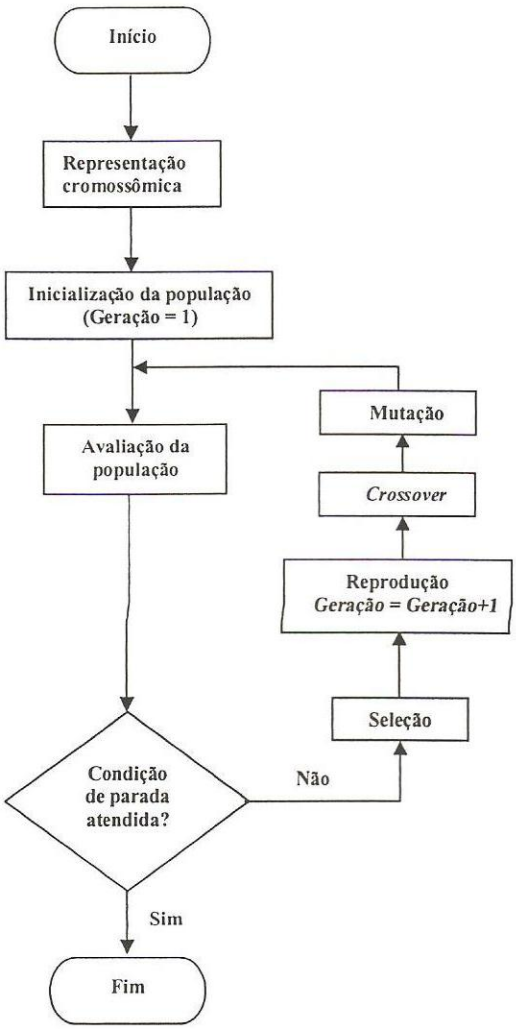

Figura 3.1: Fluxograma básico do algoritmo genético.

3.2.2 Algoritmos Genéticos Aplicados à Definição de Arquiteturas de Perceptron de Múltiplas Camadas

Os algoritmos genéticos vêm sendo amplamente utilizados na definição de arquiteturas de redes neurais artificiais [2], [3], [25]. Neste caso, o material genético de cada indivíduo codifica uma arquitetura específica de rede neural. Sendo assim, cada indivíduo contém informações sobre a dimensão de entrada da rede, a dimensão de saída, o número de camadas da rede e o número de neurônios por camada. O algoritmo genético é, então, utilizado para determinar qual arquitetura da rede é mais adequada ao problema em questão. Após definida a arquitetura da rede, podem ser utilizados algoritmos de treinamento, como o algoritmo de retro-propagação de erro para o refinamento dos pesos sinápticos.

Entretanto, os algoritmos genéticos podem ser aplicados na definição da arquitetura e no refinamento dos pesos sinápticos da rede neural simultaneamente, com a produção de bons resultados [6], [7], [8]. Neste caso, são codificadas informações no material genético dos indivíduos a respeito da arquitetura e dos pesos sinápticos da rede neural. Sendo assim, cada um dos indivíduos irá representar uma rede neural com arquiteturas e conjunto de pesos sinápticos distintos. Os indivíduos irão evoluir conforme a função custo estabelecida e, por fim, o indivíduo ótimo irá conter a configuração final da rede neural.

3.3 Processamento Paralelo Aplicado ao Treinamento de Redes Neurais Perceptron de Múltiplas Camadas e Algoritmos Genéticos

As técnicas de processamento paralelo vêm sendo aplicadas ao processo de aprendizagem de redes neurais Perceptron de Múltiplas Camadas com o intuito de minimizar o tempo gasto durante a etapa de treinamento [26].

Abordagens de paralelismo foram exploradas na literatura [23], [24]. Em [10], um novo algoritmo paralelo cooperativo para o treinamento de redes neurais Perceptron de Múltiplas Camadas foi desenvolvido. Diferentes estratégias paralelas foram aplicadas a múltiplas cópias de redes neurais em cada uma das tarefas. Informações eram periodicamente trocadas entre as tarefas, permitindo que o procedimento de busca fosse eficientemente direcionado. Por fim, utilizou-se o algoritmo de retro-propagação de erro no ajuste dos pesos sinápticos.

O conceito de algoritmos paralelos cooperativos difere da abordagem tradicional de desenvolvimento de algoritmos paralelos [27]. Ao invés de dividir a carga computacional entre cada uma das tarefas e deixá-las apenas competindo entre si, problemas completos são resolvidos em cada uma das tarefas (abordagem competitiva) e suas soluções são aceleradas a partir de informações provenientes das outras tarefas (abordagem cooperativa).

Algoritmos genéticos paralelos consistem em uma abordagem bastante interessante. A aplicação de processamento paralelo aos algoritmos genéticos permite uma melhora de desempenho tanto em termos de velocidade quanto em termos de qualidade de busca. Além disso, tem-se que a existência de

populações diversas evoluindo paralelamente evita convergências prematuras (mínimos locais).

Em algoritmos genéticos, técnicas de paralelismo são aplicadas e exploradas em diferentes níveis, produzindo soluções com baixa e alta granularidade. No modelo de baixa granularidade, apenas um membro da população é alocado a uma unidade de processamento. Sendo assim, os indivíduos só podem reproduzir e trocar material genético com outros indivíduos situados em uma região limitada. O modelo de alta granularidade, por sua vez, é o mais utilizado. Nele, divide-se uma população grande em diversas sub-populações com tamanho reduzido e utiliza-se, ou não, troca de informações entre as tarefas [28]. Cada um dos múltiplos processadores executa um algoritmo genético seqüencial nas populações. A cooperação pode ser aplicada ao modelo de alta granularidade. Neste caso, as diferentes populações presentes em cada um dos processadores comunicam-se e trocam informações entre si. Além disso, vale ressaltar que a existência de diferentes sub-populações isoladas ajuda na manutenção da diversidade genética. Os indivíduos, assim como os comportamentos evolutivos, podem ser exclusivos em cada uma das cópias do algoritmo genético presente nos processadores. Por fim, o uso de cooperação entre as populações permite que o processo evolutivo seja acelerado e que mínimos locais sejam evitados.

Capítulo 4

Algoritmo Genético Cooperativo Paralelo com Diferentes Comportamentos Evolutivos

4.1 Introdução

O algoritmo genético cooperativo paralelo com diferentes comportamentos evolutivos desenvolvido é aplicado, simultaneamente, no ajuste dos pesos sinápticos de redes neurais Perceptron de Múltiplas Camadas e na definição de uma arquitetura apropriada para as mesmas.

Em cada uma das diferentes populações evoluindo paralelamente, há um número finito de indivíduos codificando redes neurais específicas, com pesos sinápticos e arquiteturas distintas. À medida que as populações evoluem,

realiza-se o ajuste dos pesos sinápticos da rede neural, juntamente com a definição da arquitetura. Ao final do processo, extrai-se, a partir do material genético do melhor indivíduo obtido pelo algoritmo genético, a configuração final da arquitetura e dos pesos sinápticos da rede neural.

O desempenho das redes neurais Perceptron de Múltiplas Camadas, em diversas aplicações, encontra-se intrinsecamente ligado à definição de uma arquitetura adequada. Sendo assim, vale a pena ressaltar, mais uma vez, a importância de automatizar o processo de definição de arquitetura para as redes neurais Perceptron de Múltiplas Camadas. Quando feito manualmente, a definição de arquiteturas consiste em um processo de tentativa e erro, onde, na maioria das vezes, as experiências anteriores com aplicações similares são tomadas como base. O uso de algoritmos genéticos consiste, desta maneira, em uma alternativa natural e intuitiva de automatizar o processo de definição de arquiteturas para redes neurais Perceptron de Múltiplas Camadas.

Conforme mencionado, além de ser utilizado na definição da arquitetura da rede neural, o algoritmo genético também realiza, simultaneamente, o ajuste dos pesos sinápticos. Com o intuito de melhorar as buscas pelo indivíduo ótimo, diferentes critérios de reprodução são incorporados ao algoritmo genético. O uso de diferentes critérios de reprodução contribui no aumento e na manutenção da diversidade das populações acelerando, desta forma, a busca pela solução ideal. Neste caso, cada um dos indivíduos de cada uma das populações evolui diferentemente dos outros, seguindo um critério de reprodução próprio. Por fim, faz-se uso de cooperação entre as populações, permitindo, desta maneira, a exploração mais ampla do espaço de busca e evitando-se mínimos locais.

4.2 Representação do Material Genético

O material genético de cada indivíduo contém dois campos principais. O primeiro campo codifica um índice para uma tabela de arquiteturas. O segundo campo, por sua vez, codifica o conjunto de pesos sinápticos para a arquitetura definida no primeiro campo. Cada peso sináptico é representado por uma cadeia binária de 32 bits. O índice para a tabela de arquiteturas também é representado por uma cadeia binária de 32 bits. Durante a inicialização do algoritmo, as cadeias binárias representando a arquitetura da rede neural e seus pesos sinápticos são gerados aleatoriamente. À medida que as populações evoluem, as melhores arquiteturas e os melhores pesos sinápticos são mantidos. Por fim, o indivíduo selecionado contém a configuração final da rede neural.

A representação do material genético de cada indivíduo encontra-se ilustrada na Figura 4.1. Note que a tabela de arquiteturas contém diferentes configurações de rede neural, com números distintos de camadas e de neurônios por camadas. Apenas as dimensões de entrada e de saída da rede são pré-definidas, uma vez que dependem e variam conforme a aplicação.

Uma vez que diferentes arquiteturas de redes neurais são codificadas por cada um dos indivíduos, tem-se que o tamanho do material genético dos mesmos varia conforme o tamanho da rede neural representada. Ou seja, um indivíduo codificando uma rede neural mais simples apresenta um conjunto de pesos sinápticos com tamanho inferior quando comparado a um indivíduo que codifica uma rede neural com uma arquitetura mais complexa.

Entretanto, é necessário realizar um tratamento na representação dos materiais genéticos uma vez que, com a existência de tamanhos variados

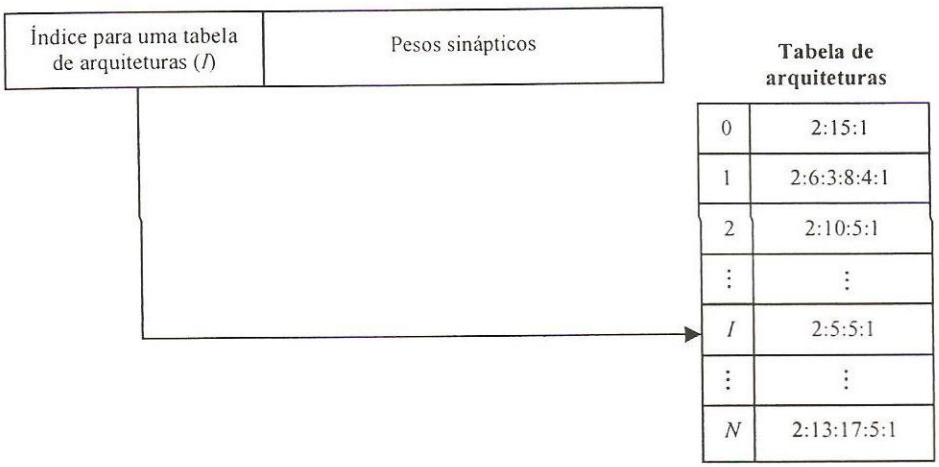

Figura 4.1: Representação do material genético de um indivíduo.

de cromossomos, pode-se, eventualmente, utilizar algum lixo da memória durante a etapa de reprodução, onde indivíduos são recombinados.

Uma solução para a questão do tamanho variado de material genético consiste em considerar a configuração de rede neural que apresenta o maior conjunto de pesos sinápticos e tomá-la como padrão. Desta maneira, todos os indivíduos apresentam o mesmo tamanho de cromossomo e as camadas, os neurônios e as conexões inexistentes são preenchidas com zero.

Porém, com esta solução, a rede pode crescer indiscriminadamente e, cada vez mais, o tamanho do material genético pode aumentar. Além de exigir muita memória e de tornar o processo de reprodução mais lento, essa solução eleva o tempo gasto nas trocas de mensagens entre as tarefas paralelas, afetando o desempenho final do algoritmo.

Uma solução viável encontrada para o problema consiste na especificação de um intervalo admissível de pesos sinápticos para a rede neural. Sendo

assim, todas as arquiteturas de rede neural que se encontram dentro do intervalo permitido são levadas em consideração. Desta maneira, impede-se que o material genético dos indivíduos cresça de forma indiscriminada, evita-se o desperdício de memória e, principalmente, acelera-se a troca de materiais genéticos entre as tarefas paralelas.

Portanto, o tamanho do material genético dos indivíduos será igual a uma cadeia binária de 32 bits, que codifica um índice para a tabela de arquiteturas, adicionado ao número máximo de pesos sinápticos permitido multiplicado por 32 bits, uma vez que cada peso sináptico é representado por uma cadeia binária de 32 bits.

Durante a inicialização das populações, especifica-se a quantidade de indivíduos desejada, o intervalo de pesos sinápticos permitido e gera-se, aleatoriamente, com distribuição uniforme, o material genético dos mesmos.

A Figura 4.2 ilustra a inicialização dos indivíduos do algoritmo genético.

```
Determine o número de populações existentes;
Determine a quantidade de indivíduos de cada população;
Determine o valor máximo do conjunto de pesos sinápticos (P);
Determine o valor mínimo do conjunto de pesos sinápticos (P_min);
Determine a porcentagem de mutação;


Para i = 0 até i < tamanho do material genético
  sorteie um valor "0" ou "1" para compor o material genético do
  indivíduo;
```

Figura 4.2: Inicialização dos indivíduos do algoritmo genético.

4.3 Definição da Tabela de Arquiteturas

Após fornecido o intervalo de pesos sinápticos permitido e conhecendo-se a dimensão de entrada e de saída do problema, o próximo passo do algoritmo genético consiste na construção de uma tabela que contém todas as configurações possíveis de arquiteturas para a rede neural.

No caso de uma rede neural com duas camadas (uma camada intermediária e uma de saída), conhecendo-se a dimensão de entrada (C_0) e saída (C_2) do problema e o número máximo de pesos sinápticos (P), pode-se calcular a quantidade C_1 de neurônios da camada intermediária da seguinte maneira:

Rede neural com duas camadas: $C_0 : C_1 : C_2$

$$C_0C_1 + 1C_1 + C_1C_2 + 1C_2 = P$$
$$C_1(C_0 + 1) + C_2(C_1 + 1) = P$$
$$(C_0 + 1)C_1 + C_2C_1 + C_2 = P$$
$$(C_0 + 1 + C_2)C_1 = P - C_2$$

$$C_1 = \frac{P - C_2}{C_0 + C_2 + 1} \tag{4.1}$$

Para uma rede neural com três camadas (duas camadas intermediárias e uma de saída), conhecendo-se a dimensão de entrada (C_0) e saída (C_3) do problema, o número máximo de pesos sinápticos (P) e incrementando gradativamente a quantidade de neurônios da primeira camada intermediária (C_1), calcula-se a quantidade C_2 de neurônios da segunda camada intermediária da seguinte maneira:

Rede neural com três camadas: $C_0 : C_1 : C_2 : C_3$

$$C_0C_1 + 1C_1 + C_1C_2 + 1C_2 + C_2C_3 + 1C_3 = P$$
$$C_1(C_0 + 1) + C_2(C_1 + 1) + C_3(C_2 + 1) = P$$
$$(C_1 + 1 + C_3)C_2 = P - (C_0 + 1)C_1 - C_3$$

$$C_2 = \frac{P - C_3(C_0 + 1)C_1}{C_1 + 1 + C_3} \tag{4.2}$$

Para uma rede neural com quatro camadas (três camadas intermediárias e uma de saída), conhecendo-se a dimensão de entrada (C_0) e saída (C_4) do problema, o número máximo de pesos sinápticos (P) e incrementando gradativamente a quantidade de neurônios da primeira camada intermediária (C_1) e da segunda camada intermediária (C_2), calcula-se a quantidade C_3 de neurônios da terceira camada intermediária da seguinte maneira:

Rede neural com quatro camadas: $C_0 : C_1 : C_2 : C_3 : C_4$
$$(C_0 + 1)C_1 + (C_1 + 1)C_2 + (C_2 + 1)C_3 + (C_3 + 1)C_4 = P$$
$$(C_2 + 1 + C_4)C_3 = P - C_4 - (C_0 + 1)C_1 - (C_1 + 1)C_2$$

$$C_3 = \frac{P - C_4 - (C_0 + 1)C_1 - (C_1 + 1)C_2}{C_2 + 1 + C_4} \tag{4.3}$$

Sendo assim, generalizando as equações 4.1 a 4.3 tem-se que:

$$C_{n-1} = \frac{P - C_n - \sum_{i=0}^{n-3}(C_i + 1)C_{i+1}}{1 + C_n + C_{n-2}} \tag{4.4}$$

Desta forma, a partir da equação 4.4, pode-se gerar todas as arquiteturas de redes neurais da seguinte maneira:

- Fixando um intervalo para o número de pesos sinápticos;
- Fixando a dimensão de entrada e de saída (C_0 e C_n);

- Variando o número de neurônios em C_i, exceto para a camada intermediária C_{n-1} que antecede a camada de saída e é calculado pela equação 4.4.

4.4 Critérios de Reprodução

A etapa de reprodução de um algoritmo genético pode ser realizada segundo diferentes heurísticas. No algoritmo genético desenvolvido, os indivíduos são ordenados conforme os valores da função custo produzidos por eles. A quantidade de indivíduos candidatos a pais da próxima geração é escolhida aleatoriamente. Note que os pais da próxima geração são os indivíduos que produziram os menores sinais de erro. A partir do conjunto de indivíduos classificados, dois são escolhidos aleatoriamente para terem os seus materiais genéticos combinados, produzindo um filho. O material genético dos indivíduos pode ser combinado em um ou dois pontos distintos. Um pequeno número de mutações, também obtido aleatoriamente, é introduzido à nova população. A Figura 4.3 ilustra dois exemplos de reprodução do algoritmo genético.

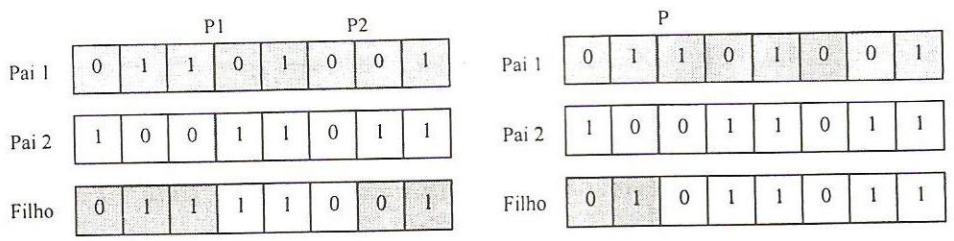

Figura 4.3: Ilustração de dois exemplos de reprodução do algoritmo genético, onde P, P_1 e P_2 são os pontos de corte obtidos aleatoriamente.

Inicialmente, escolhe-se um ou dois pontos aleatórios do material genético dos indivíduos selecionados. Caso apenas um ponto aleatório seja escolhido, a parte esquerda do material genético do filho é igual ao material genético do Pai 1 e a parte direita é igual ao material genético do Pai 2. No caso da escolha de dois pontos aleatórios, a parte esquerda ao primeiro ponto é igual ao material genético do Pai 1 e a parte direita igual ao material genético do Pai 2. Por fim, faz-se a parte direita ao segundo ponto aleatório igual ao material genético do Pai 1.

O algoritmo de reprodução dos indivíduos encontra-se ilustrado na Figura 4.4.

4.5 Cooperação

Além de apresentarem comportamentos evolutivos distintos, as diversas populações são capazes de se comunicarem entre si e trocarem informações importantes sobre o melhor indivíduo até então encontrado por cada uma delas. A existência de cooperação no algoritmo genético é importante, uma vez que a troca de informações entre as populações evita mínimos locais. Além disso, o uso de cooperação permite a exploração de um intervalo mais amplo do espaço de busca e ajuda a re-introduzir materiais genéticos previamente perdidos.

Desta maneira, em todas as populações paralelas, ordena-se cada um dos indivíduos de acordo com o valor da função custo produzido por cada um deles. O indivíduo que melhor conseguir minimizar a função custo, ou seja, que produzir o menor sinal de erro médio quadrático, é selecionado e enviado para a estrutura mestre. Esta estrutura, por sua vez, encarrega-se de enviar

```
// Sorteia a quantidade de indivíduos que serão mantidos na próxima geração
// qtd representa a quantidade de indivíduos que serão mantidos na próxima geração
// nind representa todos os indivíduos existentes na geração atual

qtd = sorteio(nind);

// Entre os indivíduos selecionados, sorteiam-se dois para terem o material genético recombinado

Pai 1 = sorteio(qtd);
Pai 2 = sorteio(qtd);

// Sorteia os pontos aleatórios

Função ponto_aleatorio( Pai1, Pai2, Filho ) {
  ponto = sorteio( filho.tamanho() ); // Sorteia ponto em qualquer lugar do material genético
  Retorne ponto;
}

// Recombinação do material genético

Função Recombinar (Pai1, Pai2, Filho) {
  Se quantidade de pontos igual a 1 {
    Para i = 0 até i < ponto
        Filho.v[i] = Pai1.v[i]; // Faz primeira parte material genético do filho igual ao Pai 1

    Para i = ponto até i < filho.v.size()
        Filho.v[i] = Pai2.v[i]; // Faz segunda parte material genético do filho igual ao Pai 2
}
  Se quantidade de pontos igual a 2 {
    Para i = 0 até i < ponto1
        Filho.v[i] = Pai1.v[i]; // Faz primeira parte material genético do filho igual ao Pai 1

    Para i = ponto1 até i < ponto2
        Filho.v[i] = Pai2.v[i]; // Faz segunda parte material genético do filho igual ao Pai 2
    Para i = ponto2 até i < filho.v.size()
        Filho.v[i] = Pai1.v[i]; // Faz terceira parte material genético do filho igual ao Pai 1
   Retorne Filho;
}

// Sorteia número de mutações

Para k = sorteio(n) até k = 0  // Sorteia número de mutações
  k = k - 1;
  // i = indivíduo em questão
  // j = posição do material genético do indivíduo em questão
  ind[i].v[j] = !ind[i].v[j];
```

Figura 4.4: Reprodução dos indivíduos.

a todas as populações paralelas o melhor indivíduo até então encontrado.

A implementação das trocas de mensagens entre as tarefas paralelas foi feita com o *software* PVM - *Parallel Virtual Machine*.

Na Figura 4.5 encontra-se a representação do envio do melhor indivíduo de cada uma das populações para a estrutura mestre. Na Figura 4.6 encontra-se a escolha do melhor indivíduo recebido pela estrutura mestre. Na Figura 4.7 encontra-se a recepção do melhor indivíduo produzido por cada uma das populações paralelas.

```
// Cálculo do valor da função de adequabilidade
// X: vetor de entrada da rede neural
// D: vetor de saídas desejadas
ind[i].custo = erro(rede, X, D);
// Ordenação dos indivíduos
sort(ind, ind + nind);
// Envia para o mestre o melhor indivíduo
    pvm_initsend(PvmDataDefault);
    pvm_pkdouble(&ind[0].custo, 1, 1);
    pvm_pkint(&ind[0].v[0], ind[0].v.size(), 1);
    pvm_send(pvm_parent(), 1);
```

Figura 4.5: Envio do melhor indivíduo de cada uma das populações.

O fluxograma da Figura 4.8 ilustra a cooperação entre as populações do algoritmo genético.

```
// Recebe melhor de cada escravo

for (int i = 0; i < nescravos; ++i) {
  cerr << "Recebendo do escravo " << i << endl;
  pvm_recv(tids[i], -1);
  pvm_upkdouble(&ind[i].custo, 1, 1);
  pvm_upkint(&ind[i].v[0], ind[i].v.size(), 1);
}

cout << "Melhor: " << ind[0].custo << endl;

// Envio do melhor para todos os escravos

  pvm_initsend(PvmDataDefault);
  pvm_pkdouble(&ind[0].custo, 1, 1);
  pvm_pkint(&ind[0].v[0], ind[0].v.size(), 1);
  pvm_mcast(tids, nescravos, 1);
  }
```

Figura 4.6: Recepção do melhor indivíduo de cada uma das populações e envio do melhor indivíduo produzido para cada uma das populações.

```
// Desloca todo mundo um para direita, para abrir espaço
para o novo melhor

copy( ind, ind + nind - 1, ind + 1 );

// Recebe na última posição o melhor do mestre

pvm_recv(-1, -1);
pvm_upkdouble(&ind[0].custo, 1, 1);
pvm_upkint(&ind[0].v[0], ind[0].v.size(), 1);
```

Figura 4.7: Recepção do melhor indivíduo produzido.

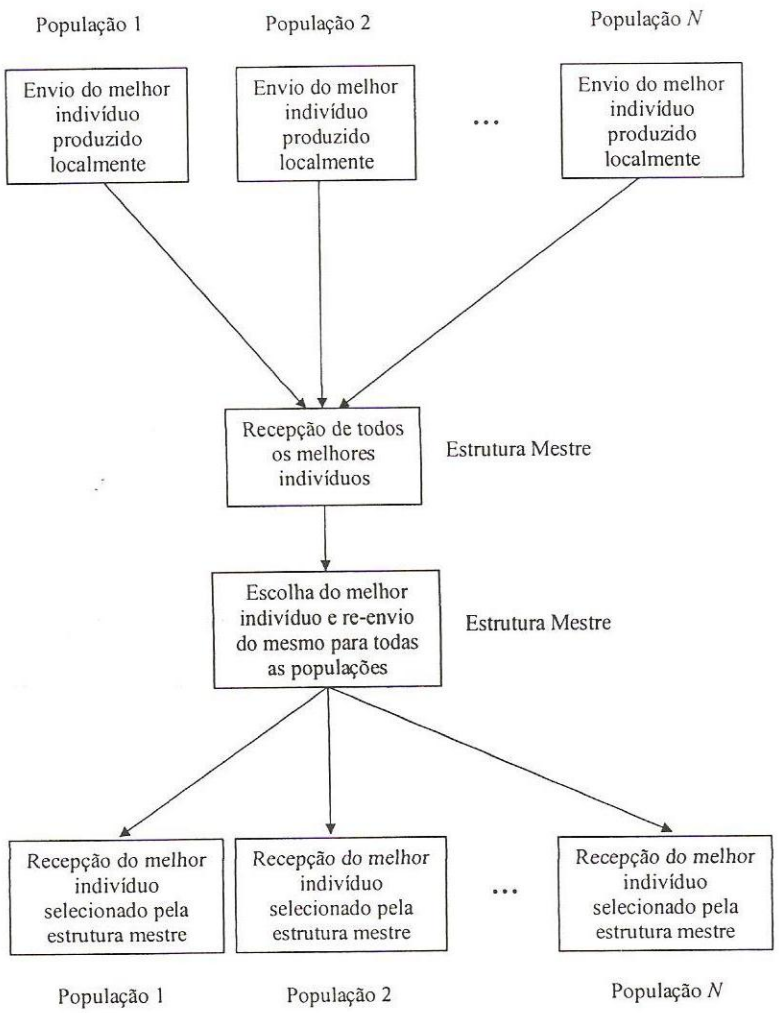

Figura 4.8: Cooperação entre as populações do algoritmo genético.

4.6 Validação Cruzada Múltipla

A utilização de validação cruzada no algoritmo genético paralelo desenvolvido objetiva evitar que a rede neural seja treinada excessivamente, apresentando uma má generalização. Desta maneira, o conjunto de dados fornecido à rede neural é dividido, aleatoriamente, em um conjunto de treinamento e em um conjunto de validação. O conjunto de treinamento é utilizado para definir a arquitetura e realizar o ajuste dos pesos sinápticos da rede neural. Após a realização do ajuste dos pesos sinápticos, faz-se a validação da rede neural com o conjunto de validação.

Uma vez que no algoritmo genético paralelo desenvolvido há diversas populações evoluindo paralelamente, optou-se pela utilização da validação cruzada múltipla descrita no Capítulo 2. O conjunto de dados é dividido em K subconjuntos, onde K equivale ao número de populações paralelas existentes. Para cada uma das diferentes populações existentes, realiza-se o treinamento com todos os subconjuntos, com exceção de um deles, utilizado para medir o erro de validação. Entretanto, para cada uma das populações paralelas existentes utiliza-se um conjunto de validação distinto. Desta forma, o procedimento é repetido K vezes paralelamente, acelerando, também, o custo computacional necessário para a validação.

A Figura 4.9 ilustra a validação cruzada múltipla implementada. Os dados de treinamento são utilizados para o ajuste dos pesos sinápticos e os dados de validação (hachurados) são utilizados para validar a rede neural.

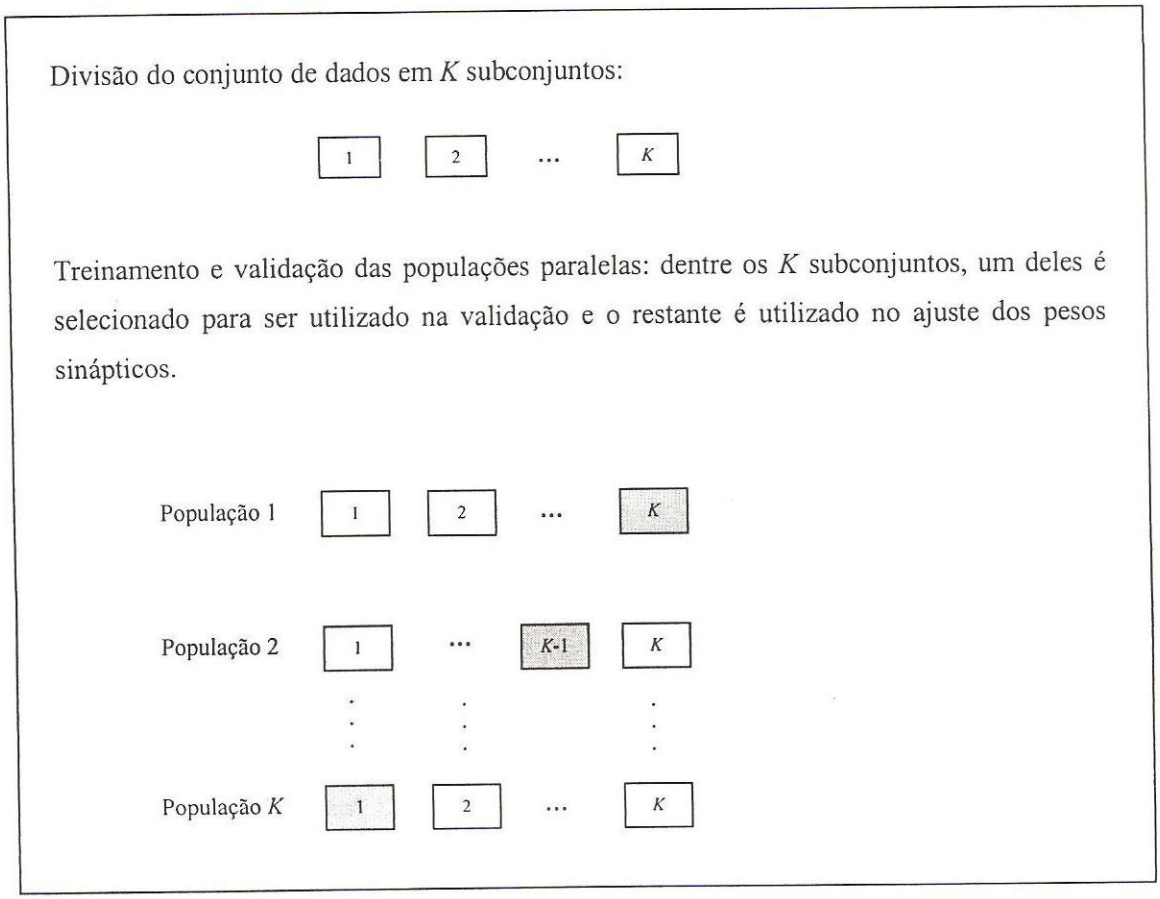

Figura 4.9: Validação cruzada múltipla implementada.

4.7 Definição de Arquiteturas

Para evitar que redes neurais muito grandes apresentem sempre menores valores de erro médio quadrático adicionou-se à função custo do algoritmo genético, um fator de penalização. Para a obtenção do fator de penalização, calcula-se e ordena-se a quantidade de pesos sinápticos codificada por cada um dos indivíduos da população. Em seguida, realiza-se a divisão entre a quantidade de pesos sinápticos codificada pelo indivíduo em questão pela maior quantidade de pesos sinápticos encontrada na população.

A equação 4.5 ilustra a função custo do algoritmo genético paralelo:

$$FuncaoCusto = EMQ + \lambda \frac{NumeroPesos}{NumeroMaximoPesos} \tag{4.5}$$

onde EMQ equivale ao cálculo do erro médio quadrático e $\lambda = 1$.

A Figura 4.10 ilustra o cálculo da função custo.

```
// Número de indivíduos = n

Para i = 0 até i < n
  Calcule a quantidade de pesos sinápticos codificada e armazene o
  resultado em um vetor (v);
Ordene o vetor com a quantidade de pesos sinápticos codificada;

// A última posição do vetor contém o maior valor de pesos sinápticos codificados

//Cálculo da função custo

F = Erro Médio Quadrático + v[i]/v[n];
```

Figura 4.10: função custo implementada.

4.8 Estrutura Paralela

A estrutura paralela adotada na implementação do algoritmo genético paralelo encontra-se ilustrada na Figura 4.11.

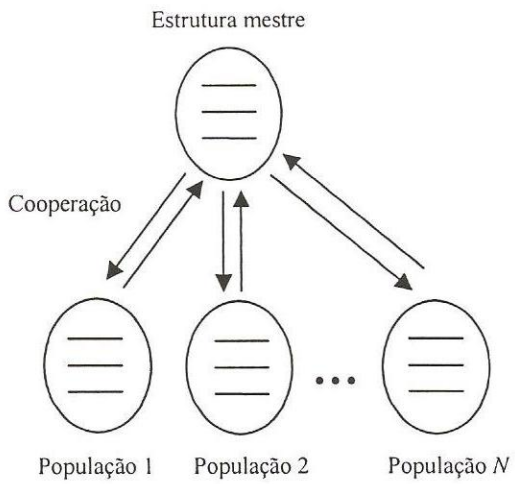

Figura 4.11: Estrutura paralela adotada no algoritmo genético

Conforme pode ser visto através da Figura 4.11, existe uma estrutura mestre que é responsável pela inicialização das populações. Cada população evolui simultaneamente, segundo critérios exclusivos de reprodução. Após um número pré-determinado de gerações, cada uma das populações envia à estrutura mestre o melhor indivíduo até então encontrado por cada uma delas. A estrutura mestre seleciona o melhor indivíduo a partir do conjunto de melhores indivíduos recebidos de todas as populações coexistentes. Finalmente, a estrutura mestre envia uma cópia do melhor indivíduo selecionado para cada uma das populações coexistentes e assim sucessivamente, até o término do algoritmo.

O ambiente utilizado na implementação do algoritmo paralelo foi o PVM. O *software* PVM consiste em um ambiente de programação paralela e distribuída. Ele permite ao usuário criar e acessar um sistema de computação paralelo composto por uma coleção de processadores distribuídos, como também tratar o sistema resultante como uma máquina virtual única, daí o nome máquina virtual paralela. O *software* PVM é baseado no modelo de programação paralela de troca de mensagem. Deste modo, tem-se que mensagens são permutadas entre as tarefas através de uma cadeia de conexão.

No capítulo seguinte são apresentados os resultados obtidos, possibilitando uma análise do algoritmo desenvolvido neste trabalho.

Capítulo 5

Resultados

O algoritmo genético paralelo desenvolvido foi utilizado no processo de treinamento de redes neurais Perceptron de Múltiplas Camadas para realizar aproximação de funções e implementar uma função de auto-compensação das medidas de sensores inteligentes.

5.1 Aproximação de Funções

Os resultados da aproximação da função $f(x_1, x_2) = \cos(2\pi x_1)\cos(2\pi x_2)$, onde $0 \leq x_1 \leq 0{,}5$ e $0 \leq x_2 \leq 0{,}5$ são inicialmente ilustrados. O conjunto de treinamento é composto por 2500 pontos da função escolhidos aleatoriamente. A vantagem do algoritmo genético paralelo cooperativo desenvolvido consiste na existência de diferentes populações, com diferentes comportamentos evolutivos, evoluindo simultaneamente. Para ilustrar os ganhos obtidos com o paralelismo, faz-se, inicialmente, uma comparação entre o algoritmo genético seqüencial e o algoritmo genético paralelo.

A Figura 5.1 representa o sinal de erro médio quadrático obtido com o algoritmo genético seqüencial padrão para a aproximação da função $f(x_1, x_2) = \cos(2\pi x_1)\cos(2\pi x_2)$. Nela, há quatro populações evoluindo de forma independente e cada linha do gráfico representa o melhor indivíduo selecionado por cada uma das quatro populações.

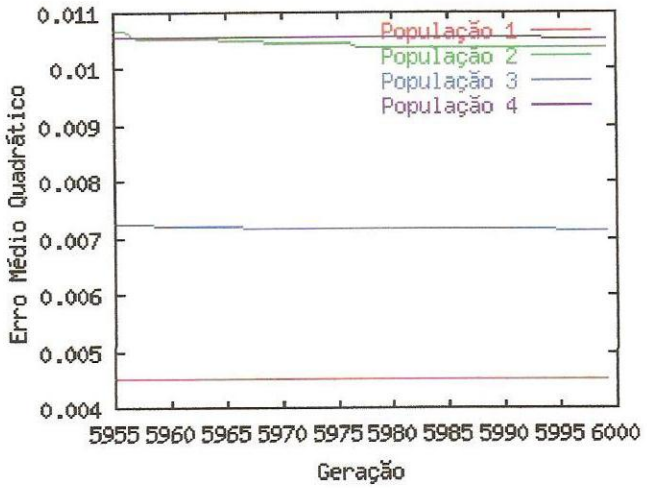

Figura 5.1: Erro médio quadrático obtido com o algoritmo genético seqüencial para a função $f(x_1, x_2) = \cos(2\pi x_1)\cos(2\pi x_2)$.

Uma vez que foi utilizada a abordagem seqüencial do algoritmo genético tem-se que as quatro populações evoluem independentemente, sem nenhum grau de cooperação ou comunicação entre elas. As novas gerações de indivíduos de cada uma das quatro populações são obtidas apenas através da recombinação dos materiais genéticos dos indivíduos locais.

Entretanto, uma vez que a inicialização dos indivíduos que compõem as populações seqüenciais é aleatória, pode-se ter, em cada uma das quatro populações, sinais de erro médio quadrático mais elevados do que outros.

Além disso, dependendo também da inicialização, é freqüente encontrar populações presas em mínimos locais. Uma vez que na abordagem seqüencial não há trocas de materiais genéticos entre os indivíduos das diferentes populações, a existência de populações presas em mínimos locais tem maior probabilidade de ocorrência.

Conforme ilustrado na Figura 5.1, percebe-se que, durante a geração 5955 a População 1 apresenta o menor valor de erro médio quadrático, enquanto que a População 2 apresenta o maior valor de erro médio quadrático. As populações seguem a sua evolução, separadamente, até atingirem a geração 6000. Ao longo da evolução, é percebido que as Populações 1, 3 e 4 encontram-se estagnadas em um mínimo local. Apenas a População 2 continua a reduzir, lentamente, o seu valor de erro médio quadrático. Entretanto, devido a inicialização dos indivíduos da sua população, observa-se, na geração 6000, que o menor sinal de erro médio quadrático encontrado corresponde à População 1.

Os resultados da implementação paralela do algoritmo genético desenvolvido encontra-se ilustrada na Figura 5.2. Nela, há quatro populações evoluindo simultaneamente e cada linha do gráfico representa o melhor indivíduo selecionado por cada uma das quatro populações.

Inicialmente, as quatro populações paralelas evoluem simultaneamente, porém de forma independente, sem trocas de informações. Entretanto, a cada quinze gerações, as populações paralelas comunicam-se entre si, trocando os melhores materiais genéticos produzidos por cada uma.

Devido a presença de cooperação entre os indivíduos que compõem as populações paralelas, evita-se que, por causa de uma má inicialização, uma população seja sujeita a elevados valores de erro médio quadrático ou, até

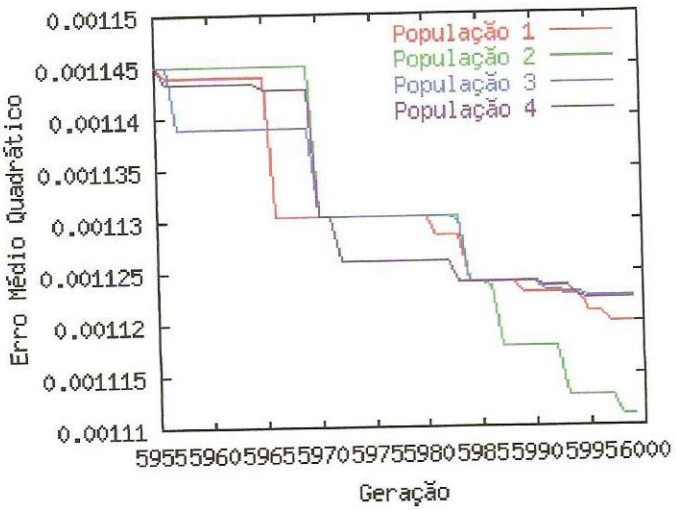

Figura 5.2: Erro médio quadrático obtido com o algoritmo genético paralelo cooperativo para a função $f(x_1, x_2) = \cos(2\pi x_1)\cos(2\pi x_2)$.

mesmo, que fique presa em algum mínimo local.

Além disso, com a utilização de diferentes critérios de reprodução, aumenta-se a variabilidade genética dos indivíduos intensificando, conseqüentemente, a busca pela melhor solução.

Nota-se pela Figura 5.2 que durante a geração 5955, houve comunicação entre as populações. Sendo assim, as Populações 1 a 4 apresentam o mesmo valor de erro médio quadrático. As quatro populações paralelas evoluem de forma independente e competitiva até a geração 5970, onde ocorre troca de material genético. Após a troca, nota-se que as Populações 2, 3 e 4 são beneficiadas pela População 1, reduzindo o seu sinal de erro médio quadrático para o valor mais baixo até então encontrado. As populações paralelas seguem a evolução de forma independente por mais quinze gerações. Durante a geração 5985, outra troca de material genético acontece e, desta

vez, as Populações 1, 2 e 3 são beneficiadas pela População 4. Após estabelecida a cooperação, as populações seguem sua evolução individualmente até que, na geração 6000, observa-se que a População 2 apresenta o menor erro médio quadrático. Sendo assim, a partir do material genético do melhor indivíduo encontrado pela População 2, pode-se obter informações sobre a arquitetura e os pesos sinápticos da rede neural.

Utilizou-se, no algoritmo genético paralelo, a validação cruzada múltipla. Conforme mencionado, na validação cruzada múltipla, treina-se o modelo com todos os subconjuntos, com exceção de um deles que é utilizado para medir o erro de validação e repete-se o procedimento K vezes. Em cada uma das K vezes, utiliza-se um subconjunto diferente para validação. No algoritmo genético paralelo implementado, para calcular o erro de validação, divide-se o conjunto de dados em K subconjuntos, equivalente ao número de populações paralelas existentes. Para cada uma das diferentes populações existentes, realiza-se o treinamento com todos os subconjuntos, com exceção de um deles, utilizado para medir o erro de validação. Além disso, é utilizado, em cada uma das populações paralelas existentes, um conjunto de validação distinto. Neste caso, o procedimento é repetido K vezes, paralelamente, acelerando, também, o tempo necessário para a validação.

O cálculo do erro de validação é efetuado de 15 em 15 gerações. Cada uma das populações paralelas produz um erro de validação distinto. Em seguida, realiza-se a média do erro quadrado obtido na validação de cada uma das populações paralelas. O resultado obtido através do cálculo do erro médio quadrático de validação é utilizado para avaliar o desempenho da rede neural.

O sinal de erro de validação obtido com o algoritmo genético paralelo

encontra-se ilustrado na Figura 5.3. A Figura 5.4 ilustra o erro de validação para as gerações 5955 a 6000.

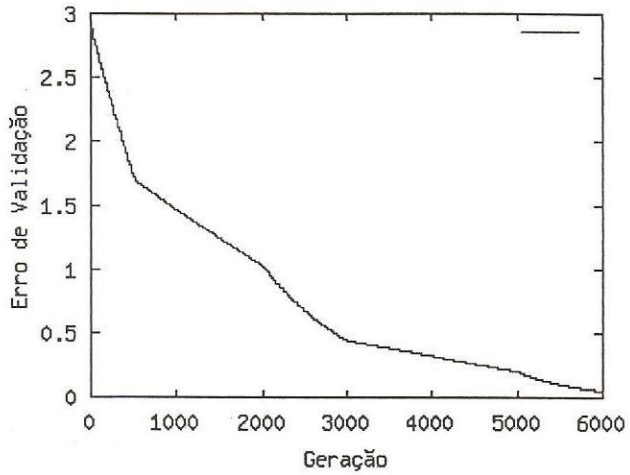

Figura 5.3: Erro de validação da função $f(x_1, x_2) = \cos(2\pi x_1)\cos(2\pi x_2)$.

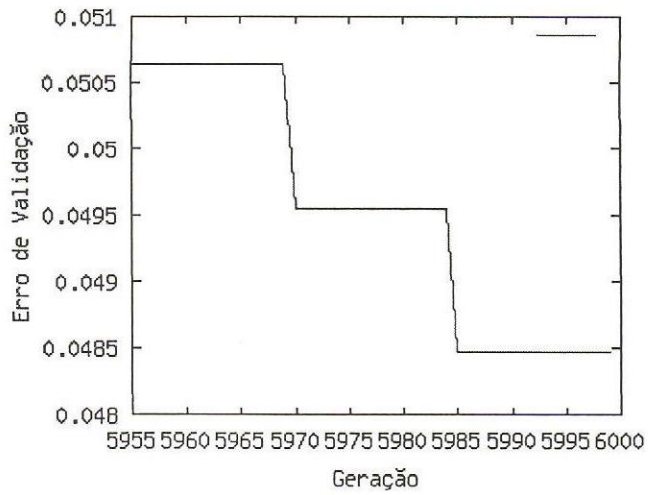

Figura 5.4: Erro de validação da função $f(x_1, x_2) = \cos(2\pi x_1)\cos(2\pi x_2)$ para a geração 5955 a 6000.

O algoritmo genético paralelo desenvolvido também foi utilizado na definição da arquitetura da rede neural Perceptron de Múltiplas Camadas. Foram utilizadas diversas configurações de redes neurais. Fez-se o número de camadas intermediárias variar de um a três e fez-se o número de neurônios por camada variar de dois a quinze. A dimensão de entrada da rede neural é igual a dois e a dimensão de saída da rede neural é igual a um. A configuração final da arquitetura da rede neural Perceptron de Múltiplas Camadas obtida com o algoritmo genético paralelo apresentou uma camada de entrada com duas entradas, uma camada de saída com um neurônio e uma camada intermediária com doze neurônios.

A saída reconstruída da função $f(x_1, x_2) = \cos(2\pi x_1)\cos(2\pi x_2)$ obtida através da rede neural encontra-se ilustrada na Figura 5.5. A saída original da função $f(x_1, x_2) = \cos(2\pi x_1)\cos(2\pi x_2)$, por sua vez, encontra-se ilustrada na Figura 5.6.

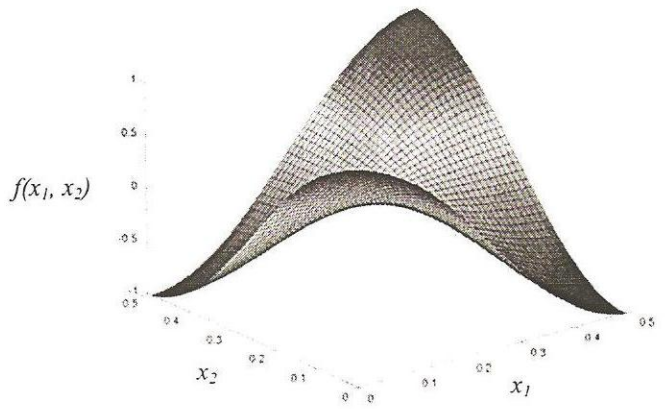

Figura 5.5: Saída reconstruída obtida pela rede neural Perceptron de Múltiplas Camadas através do treinamento com o algoritmo genético paralelo para a função $f(x_1, x_2) = \cos(2\pi x_1)\cos(2\pi x_2)$.

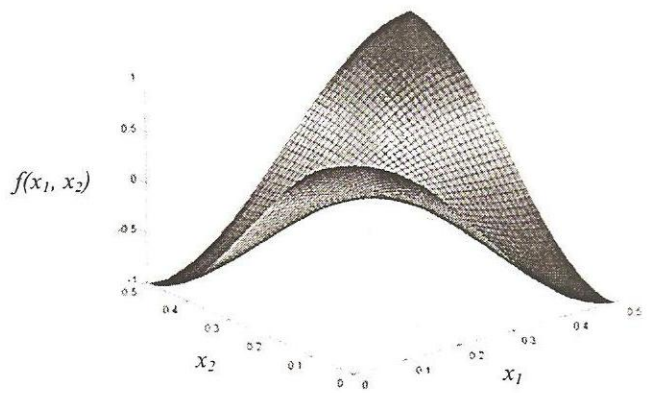

Figura 5.6: Saída original da função $f(x_1, x_2) = \cos(2\pi x_1)\cos(2\pi x_2)$.

O segundo resultado ilustra a aproximação da função $z = sen(r)/r$, onde $r = \sqrt{x^2 + y^2}$ e -7,5 $\leq x \leq$ 7,5 e -7,5 $\leq y \leq$ 7,5. O conjunto de treinamento é composto por 960 pontos da função obtidos aleatoriamente. Primeiramente, é utilizado o algoritmo genético seqüencial para aproximar a função. Em seguida, faz-se uso da abordagem paralela do algoritmo genético desenvolvido para ilustrar os benefícios obtidos com o paralelismo.

A Figura 5.7 representa o sinal de erro médio quadrático obtido com o algoritmo genético seqüencial padrão para a aproximação da função.

As quatro populações evoluem independentemente, sem o estabelecimento de troca de informação, de modo que as novas gerações de indivíduos são obtidas através da recombinação dos materiais genéticos dos indivíduos locais. Uma vez que não há trocas de materiais genéticos entre os indivíduos das diferentes populações, pode-se observar, mais facilmente, a existência de populações presas em mínimos locais.

Conforme ilustrado na Figura 5.7, percebe-se que, durante a geração

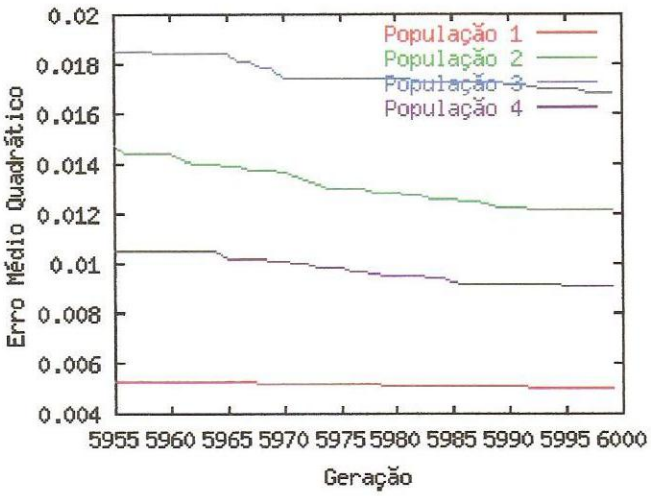

Figura 5.7: Erro médio quadrático obtido com o algoritmo genético seqüencial para a função $z = sen(r)/r$.

5955, a População 3 apresenta o maior valor de erro médio quadrático, enquanto que a População 1 apresenta o menor valor de erro médio quadrático. As populações seguem a sua evolução, separadamente, até atingirem a geração 6000. Ao longo da evolução, é claro perceber que a População 2, a População 3 e a População 4 seguem reduzindo seus valores de erro médio quadrático. A População 1, por sua vez, provavelmente encontra-se presa em um mínimo local. Na geração 6000, observa-se que a População 3 apresenta o maior valor de erro médio quadrático e a População 1 apresenta o menor valor.

A abordagem paralela do algoritmo genético desenvolvido encontra-se ilustrada na Figura 5.8. Nela, há quatro populações evoluindo simultaneamente e cada linha do gráfico representa o melhor indivíduo selecionado por cada uma das quatro populações.

As quatro populações evoluem paralelamente, fazendo uso de cooperação

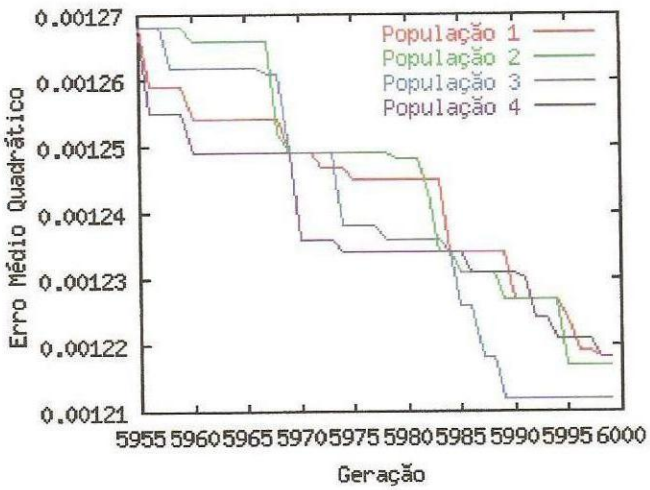

Figura 5.8: Erro médio quadrático obtido com o algoritmo genético paralelo cooperativo para a função $z = sen(r)/r$

periodicamente. As novas gerações de indivíduos de cada uma das quatro populações são obtidas através da recombinação dos materiais genéticos dos indivíduos locais. Os critérios de reprodução, por sua vez, variam entre as populações paralelas. Após um número pré-definido de gerações, as populações paralelas se comunicam, trocando informações sobre o melhor indivíduo produzido por cada uma.

A troca de materiais genéticos entre as populações paralelas ocorre a cada quinze gerações. Nos intervalos das cooperações, as quatro populações evoluem simultaneamente, porém de forma independente, sem trocas de informações. Sendo assim, as populações são estimuladas a competir e cooperar entre si.

Novamente, a presença de cooperação entre os indivíduos que compõem as populações paralelas previne mínimos locais e ajuda àquelas populações que apresentam má inicialização.

Através da Figura 5.8, percebe-se que durante a geração 5955, as populações paralelas comunicam-se, trocando informações sobre o melhor indivíduo até então encontrado. Desta maneira, tem-se que, nesta geração, as Populações 1, 2, 3 e 4 apresentam o mesmo valor de erro médio quadrático. As quatro populações paralelas evoluem de forma independente e competitiva até a geração 5970, onde ocorre troca de material genético. Após a troca, nota-se que as Populações 1, 2 e 3 são beneficiadas pela População 4, reduzindo o seu sinal de erro médio quadrático para o valor mais baixo até então encontrado. As populações paralelas seguem a evolução de forma independente por mais quinze gerações. Durante a geração 5985, outra troca de material genético acontece e as Populações 1, 2 e 3 são novamente beneficiadas pela População 4. Após estabelecida a cooperação, as populações seguem sua evolução individualmente até que, na geração 6000, observa-se que a População 3 apresenta o menor erro médio quadrático. Além disso, a partir do material genético do melhor indivíduo encontrado pela População 3, pode-se obter informações sobre a arquitetura e os pesos sinápticos da rede neural.

O erro de validação obtido com o algoritmo genético paralelo encontra-se ilustrado na Figura 5.9. A Figura 5.10 ilustra o erro de validação para as gerações 5955 a 6000.

O algoritmo genético paralelo desenvolvido também foi utilizado na definição da arquitetura da rede neural Perceptron de Múltiplas Camadas. Foram utilizadas diversas configurações de redes neurais. Fez-se o número de camadas intermediárias variar de um a três e fez-se o número de neurônios por camada variar de dois a quinze. A dimensão de entrada da rede neural é igual a dois e a dimensão de saída da rede neural é igual a um. A configuração final da arquitetura da rede neural Perceptron de Múltiplas Camadas

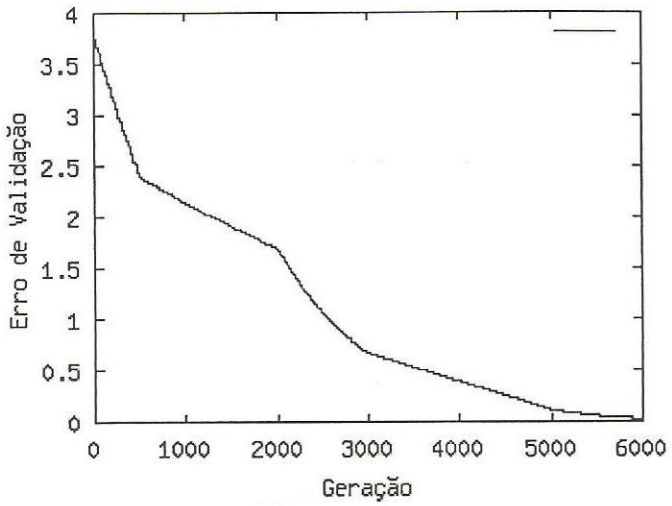

Figura 5.9: Erro de validação da função $z = sen(r)/r$.

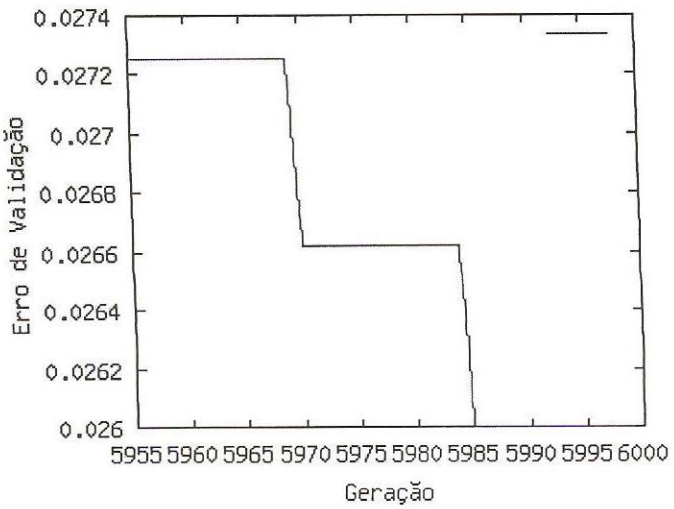

Figura 5.10: Erro de validação da função $z = sen(r)/r$ para a geração 5955 a 6000.

obtida com o algoritmo genético paralelo apresentou uma camada de entrada com duas entradas, uma camada de saída com um neurônio e duas camadas intermediárias com oito e seis neurônios, respectivamente.

A saída reconstruída da função $z = sen(r)/r$ obtida com a rede neural Perceptron de Múltiplas Camadas encontra-se ilustrada na Figura 5.11. Na Figura 5.12, é ilustrada a saída original da função.

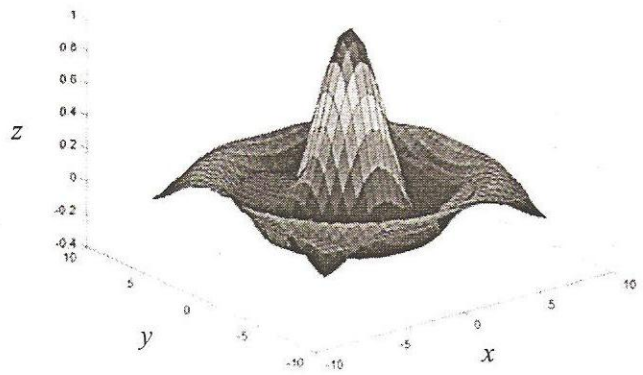

Figura 5.11: Saída reconstruída obtida pela rede neural Perceptron de Múltiplas Camadas através do treinamento com o algoritmo genético paralelo para a função $z = sen(r)/r$.

5.2 Aplicação em Sensores Inteligentes

O funcionamento de um sensor corresponde à uma função matemática que recebe uma variável de processo como entrada e fornece como saída o valor equivalente de tensão, corrente elétrica ou um valor digital, por exemplo. Devido a desgastes e outros fatores, para que um sensor funcione adequada-

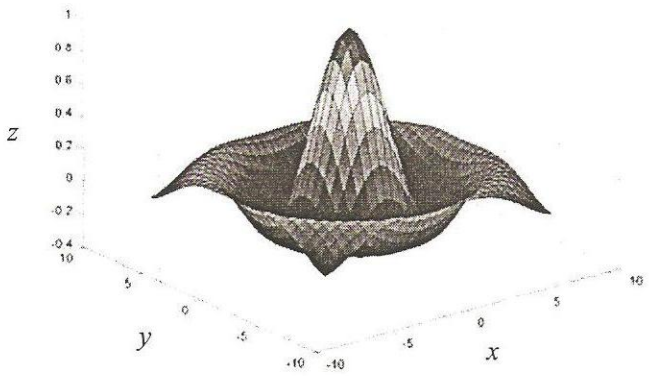

Figura 5.12: Saída original da função $z = sen(r)/r$.

mente, faz-se necessário a realização de calibrações periódicas do mesmo. Um sensor descalibrado, ao receber uma entrada, fornece uma resposta diferente daquela produzida por um sensor calibrado. Para se calibrar um sensor, o mesmo é retirado do campo e levado ao laboratório, o que pode levar muito tempo e apresentar alto custo já que faz-se necessário interromper a produção ou trabalhar com redundância. Além disso, uma vez que a periodicidade ideal para retirada do sensor é desconhecida, o mesmo pode ser sub-utilizado, caso retirado prematuramente ou pode produzir medidas incorretas, caso retirado tardiamente.

Em [29] foram desenvolvidos algoritmos de auto-compensação, auto-calibração e auto-validação para sensores *Foundation Fieldbus* utilizando redes neurais Peceptron de Múltiplas Camadas. Nesta sessão, serão ilustrados os resultados referentes a auto-compensação de sensores utilizando redes neurais Perceptron de Múltiplas Camadas treinadas com o algoritmo genético paralelo desenvolvido.

O algoritmo de auto-compensação é utilizado para corrigir o erro provocado pelas alterações na curva de calibração do sensor. Utiliza-se uma rede neural Perceptron de Múltiplas Camadas que recebe como entrada a medida de um sensor em campo e o seu tempo de funcionamento e gera como saída a medida equivalente de um sensor calibrado.

O sistema de treinamento da rede neural para a auto-compensação é ilustrado na Figura 5.13.

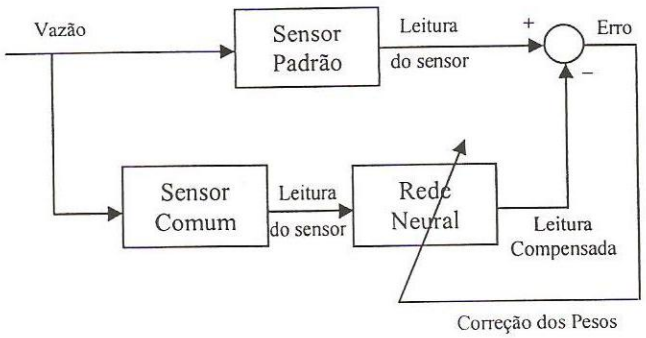

Figura 5.13: Implementação da auto-compensação de sensores com redes neurais.

O algoritmo genético paralelo desenvolvido foi utilizado no treinamento de redes neurais Perceptron de Múltiplas Camadas para realizar a auto-compensação de sensores.

Os testes foram realizados com um conjunto de curvas que modelavam a variação com o tempo das curvas de calibração de um sensor empírico da seguinte forma:

$$v(t) = \eta(1 - e^{\alpha f(t)}) \tag{5.1}$$

onde η e α são constantes que definem a forma da curva, $f(t)$ é a grandeza

a ser medida na entrada do sensor em um tempo t e $v(t)$ é o sinal de saída gerado pelo sensor que corresponde à medida efetuada da variável de entrada.

O comportamento do sensor calibrado encontra-se ilustrado na Figura 5.14. As curvas que representam a variação do comportamento do sensor com o tempo são ilustradas na Figura 5.15. As curvas ilustradas na Figura 5.15 representam o comportamento do sensor durante um período de 0 a 36 meses após a calibração e foram fornecidas como entrada para a rede neural.

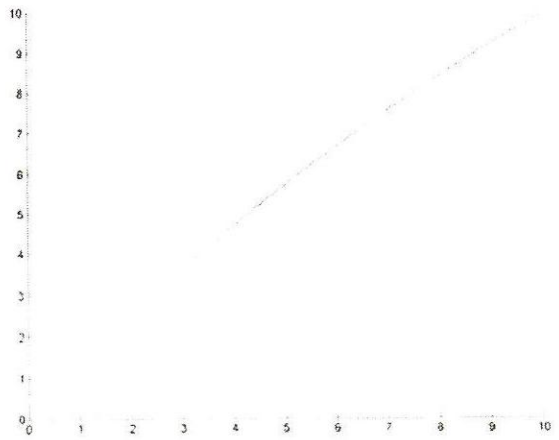

Figura 5.14: Curva obtida com o sensor calibrado.

Inicialmente, para ilustrar os ganhos obtidos com o paralelismo, apresenta-se os resultados obtidos com a auto-compensação do sensor utilizando-se a abordagem seqüencial do algoritmo genético.

A Figura 5.16 representa o sinal de erro médio quadrático obtido com o algoritmo genético seqüencial padrão. Nela, há quatro populações evoluindo de forma independente e cada linha do gráfico representa o melhor indivíduo selecionado por cada uma das quatro populações.

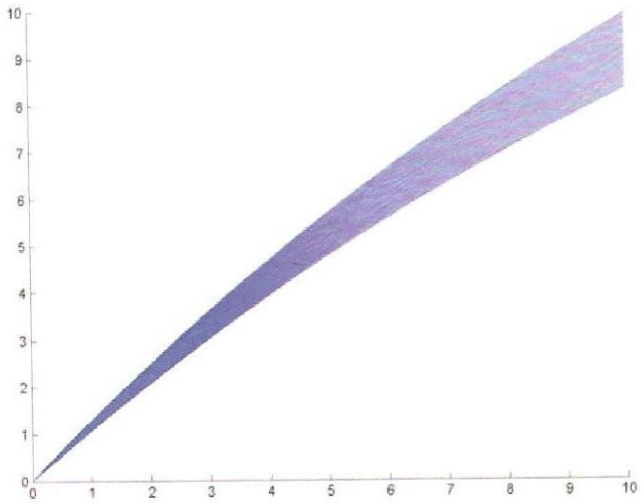

Figura 5.15: Curvas obtidas com sensor após 36 meses de uso.

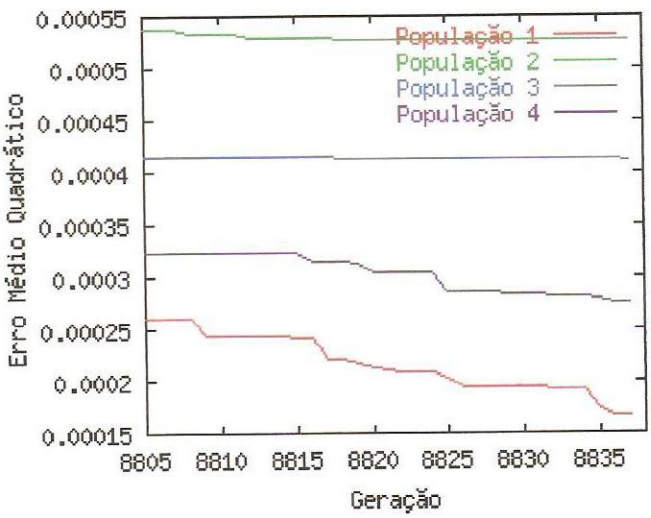

Figura 5.16: Erro médio quadrático obtido com o algoritmo genético seqüencial para a auto-compensação.

As quatro populações do algoritmo genético evoluem de forma separada e independente, sem a existência de cooperação ou comunicação. As novas gerações de indivíduos de cada uma das quatro populações são obtidas apenas através da recombinação dos materiais genéticos dos indivíduos locais.

A existência de sinais de erro médio quadrático mais elevados que outros é resultado da inicialização aleatória dos indivíduos que compõem as populações seqüenciais. Além disso, dependendo também da inicialização, é freqüente encontrar populações presas em mínimos locais. Uma vez que na abordagem seqüencial não há trocas de materiais genéticos entre os indivíduos das diferentes populações, a existência de populações presas em mínimos locais torna-se mais propícia.

Conforme ilustrado na Figura 5.16, percebe-se que, durante a geração 8805 a População 1 apresenta o menor valor de erro médio quadrático, enquanto que a População 2 apresenta o maior valor de erro médio quadrático. As populações seguem a sua evolução, separadamente, até atingirem a geração 8837. Ao longo da evolução, é nota-se que a População 2 e a População 3 encontram-se estagnadas em um mínimo local. Apenas a População 1 e a População 4 continuam a reduzir o valor de erro médio quadrático.

A abordagem paralela do algoritmo genético desenvolvido encontra-se ilustrada na Figura 5.17. Nela, há quatro populações evoluindo simultaneamente e cada linha do gráfico representa o melhor indivíduo selecionado por cada uma das quatro populações.

Na abordagem paralela, tem-se que, a princípio, as quatro populações evoluem simultaneamente, porém de forma independente. A cooperação entre as populações só ocorre a cada quinze gerações, onde é estabelecida a comunicação e a troca dos melhores materiais genéticos produzidos por cada

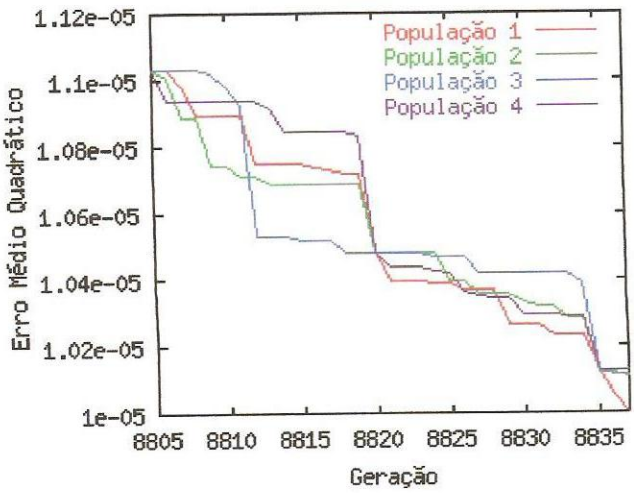

Figura 5.17: Erro médio quadrático obtido com o algoritmo genético paralelo cooperativo para a auto-compensação.

uma das populações paralelas.

Devido a presença de cooperação entre os indivíduos que compõem as populações paralelas, evita-se que más inicializações resultem em mínimos locais ou elevados valores de erro médio quadrático.

Além disso, com a utilização de diferentes critérios de reprodução, a variabilidade genética dos indivíduos é intensificada, assim como a busca por solução adequada.

Conforme percebe-se através da Figura 5.17, durante a geração 8805, houve comunicação entre as populações. Sendo assim, as Populações 1, 2, 3 e 4 apresentam o mesmo valor de erro médio quadrático. As quatro populações paralelas evoluem de forma independente e competitiva até a geração 8820, onde ocorre troca de material genético. Após a troca, nota-se que as Populações 1, 2 e 4 são beneficiadas pela População 3, reduzindo o seu

sinal de erro médio quadrático para o valor mais baixo até então encontrado. As populações paralelas seguem a evolução de forma independente por mais quinze gerações. Durante a geração 8835, outra troca de material genético acontece e, desta vez, as Populações 2, 3 e 4 são beneficiadas pela População 1. Após estabelecida a cooperação, as populações seguem sua evolução individualmente até a geração 8837, onde observa-se que a População 1 apresenta o menor erro médio quadrático. Sendo assim, a partir do material genético do melhor indivíduo encontrado pela População 1, pode-se obter informações sobre a arquitetura e os pesos sinápticos da rede neural.

A validação cruzada múltipla também foi utilizada, no algoritmo genético paralelo durante a auto-compensação.

O sinal de erro de validação obtido com o algoritmo genético paralelo encontra-se ilustrado na Figura 5.18. A Figura 5.19 ilustra o erro de validação para as gerações 8805 a 8837.

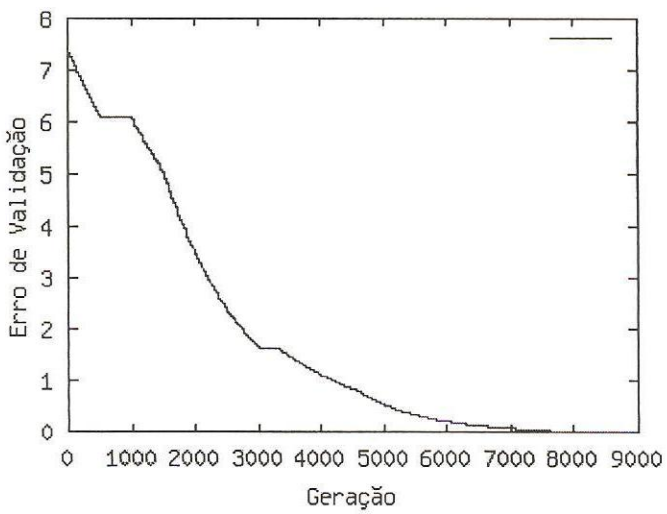

Figura 5.18: Erro de validação para a auto-compensação.

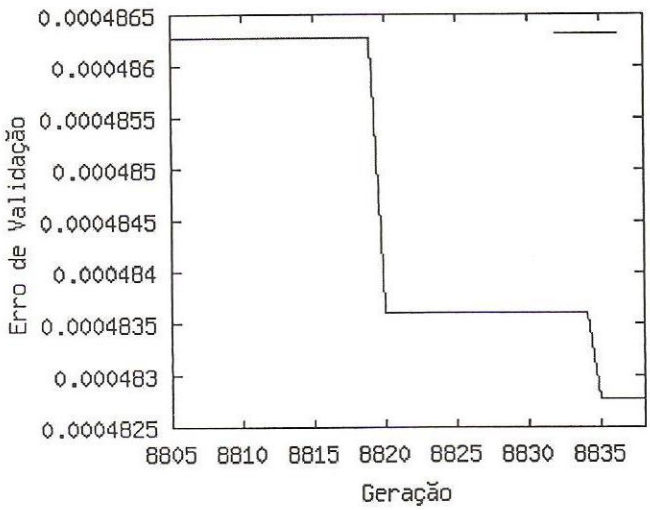

Figura 5.19: Erro de validação da auto-compensação para as gerações 8805 a 8837.

O algoritmo genético paralelo desenvolvido também foi utilizado na definição da arquitetura da rede neural Perceptron de Múltiplas Camadas. Foram utilizadas diversas configurações de redes neurais. Fez-se o número de camadas intermediárias variar de um a três e fez-se o número de neurônios por camada variar de dois a quinze. A dimensão de entrada da rede neural é igual a dois e a dimensão de saída da rede neural é igual a um. A *configuração final da arquitetura da rede neural Perceptron de Múltiplas Camadas* obtida com o algoritmo genético paralelo apresentou uma camada de entrada com duas entradas, uma camada de saída com um neurônio e duas camadas intermediárias com quatro e dois neurônios, respectivamente.

O resultado obtido com a auto-compensação é ilustrado na Figura 5.20. Os pontos pretos correspondem à saída compensada do sensor descalibrado pela rede neural. Embaixo dos mesmos, encontra-se a curva correspondente ao sensor padrão. A saída do sensor descalibrado no trigésimo sexto mês

também encontra-se ilustrada.

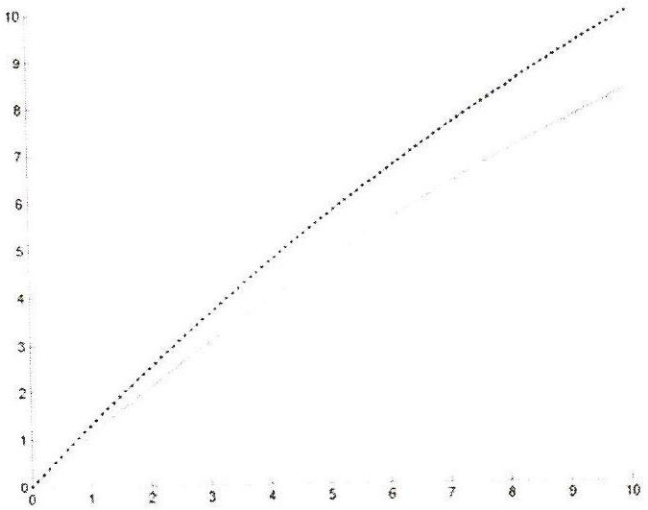

Figura 5.20: Auto-compensação do sensor realizada pela rede neural Perceptron de Múltiplas Camadas.

Capítulo 6

Conclusões

O algoritmo genético cooperativo paralelo desenvolvido foi aplicado, simultaneamente, no ajuste dos pesos sinápticos de redes neurais Perceptron de Múltiplas Camadas e na definição de uma arquitetura apropriada para as mesmas.

O algoritmo genético desenvolvido foi aplicado ao problema de aproximação de funções e de auto-compensação de sensores, podendo ter seu uso estendido a diversas aplicações, tais como previsão de séries temporais, filtragem, entre outras.

Através da análise dos resultados obtidos com a aplicação da abordagem seqüencial e com a aplicação da abordagem paralela do algoritmo genético percebe-se que a abordagem paralela mostrou ser eficiente quando comparada à abordagem seqüencial.

Conforme pôde ser observado através da análise dos resultados obtidos, o simples fato de permitir trocas de informações entre as diferentes populações proporcionou considerável melhora na desempenho final do algoritmo

genético. Mostrou-se, através de cooperação, com a troca periódica entre as populações paralelas do material genético do melhor individuo obtido na geração que antecede a comunicação, que as diferentes populações puderam interagir entre si e ajudar umas as outras, evitando mínimos locais e auxiliando a busca pela solução ideal.

Desta maneira, o algoritmo genético deixa de ser puramente competitivo. Com a abordagem implementada, as populações competem entre si na busca do individuo ótimo apenas até a geração que antecede a comunicação, onde foi estabelecida a cooperação entre as populações. Sendo assim, conseguiu-se evitar que populações ficassem presas a elevados valores de erro médio quadrático.

Portanto, vale a pena ressaltar que o uso de diferentes comportamentos evolutivos em cada uma das populações permitiu uma maior diversificação dos indivíduos, acelerando, também, a busca por uma solução adequada.

Por fim, o algoritmo genético foi usado, simultaneamente, para refinar os pesos sinápticos da rede neural e para definir a arquitetura da mesma. Sabe-se que, em sua maioria, a definição de arquiteturas para redes neurais consiste em uma tarefa árdua. Além disso, a maioria dos algoritmos de treinamento desenvolvidos para as redes neurais artificiais de forma geral, apenas realiza o ajuste dos pesos sinápticos. Desta maneira, a combinação e aplicação de algoritmos genéticos na definição, assim como no ajuste dos pesos sinápticos de redes neurais Perceptron de Múltiplas Camadas é interessante.

As perspectivas futuras do trabalho desenvolvido incluem a utilização do algoritmo cooperativo implementado para:

- Realizar re-treinamento de redes neurais do tipo *feed-forward*;

- Realizar treinamento de redes neurais recorrentes;
- Desenvolver aplicações em compressão, controle, identificação de sistemas e classificação de padrões;
- Desenvolver métodos híbridos de retropropagação de erro e algoritmos genéticos cooperativos;
- Realizar testes com populações com tamanhos diferentes;
- Comparar resultados com outras abordagens da literatura.

Referências Bibliográficas

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Maxwell Macmillan International, 1994.

[2] K. Balakrishnan, V. Honavar, Evolutionary Design of Neural Architectures, *Technical Report ISU CS-TR 95-01*, Iowa State University, January 1995.

[3] D. Curran, C. ORiordan, Applying Evolutionary Computation to Designing Neural Networks: A Study of the State of the Art, *Technical Report: NUIG-IT 111002, 12 pages*, National University of Ireland, Galway, Ireland, October 2002.

[4] D. Whitley, Genetic Algorithms and Neural Networks, *Genetic Algorithms in Engineering and Computer Science*, Winter, Periaux, Galan and Cuesta, eds., pp. 203-216, John Wiley, 1995.

[5] K.O. Stanley, R. Miikkulainen, Efficient Reinforcement Learning through Evolving Neural Network Topologies, *Proceedings of GECCO*, 2002.

[6] A.C.M.L. Albuquerque, J.D. Melo, A.D. Dória Neto, Evolutionary Computation and Parallel Processing Applied to the Design of Multilayer Perceptron, *Evolvable Machines-Theory and Practice*, N. Nedjah

and L. M. Mourelle, Eds., Springer-Verlag, Vol. 161, Chapter 8, pp. 181-203, Germany, 2004.

[7] A.C.M.L. Albuquerque, J.D. Melo, A.D. Dória Neto, Parallel Genetic Algorithm with Different Evolution Behavior for Multilayer Perceptron Design and Learning, *Learning and Nonlinear Models - Revista da Sociedade Brasileira de Redes Neurais*, Vol. 1, No. 2, pp.195-207, 2003.

[8] A.C.M.L. Albuquerque, J.D. Melo, A.D. Dória Neto, Algoritmos Genéticos e Processamento Paralelo Aplicado ao Treinamento e Definição de Redes Neurais Perceptron de Múltiplas Camadas, *XV Congresso Brasileiro de Automática*, Gramado - RS, 2004.

[9] A.C.M.L. Albuquerque, J.D. Melo, A.D. Dória Neto, Algoritmo Genético Paralelo com Diferentes Comportamentos Populacionais para o Treinamento de Perceptrons de Múltiplas Camadas, *VI Congresso Brasileiro de Redes Neurais*, São Paulo - SP, 2003.

[10] R. Alves, J.D. Melo, A.D. Dória Neto, A.C.M.L. Albuquerque, New Parallel Algorithms for Backpropagation Learning, *INNS-IEEE International Joint Conference on Neural Networks* , Honolulu, USA, 2002.

[11] P. Adamidis, S. Kazarlis, V. Petridis, Advanced Methods for Evolutionary Computation, *Symposium on Large Scale Systems - LSS 98*, Greece, 1998.

[12] F. Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, *Psychological Review*, Vol. 65. pp. 386-408, 1958.

[13] M. Siddique, M. Tokhi, Training Neural Networks: Backpropagation vs Genetic Algorithms, *INNS-IEEE International Joint Conference on Neural Networks*, Washington DC, 14-19, July 2001.

[14] V. Phansalkar, P. Sastry, Analysis of the Backpropagation Algorithm with Momentum, *IEEE Trans. on Neural Networks*, Vol. 5, No. 3, pp. 505-506, 1994.

[15] T. Vogt, J. Mangis, A. Rigler, W. Zink, D. Alkon, Accelerating the Convergence of the Back-propagation Method, *Biological Cybernetics*, Vol. 59, pp. 257, 1988.

[16] H. Mühlenbein, Limitations of Multilayer Perceptron Networks - Steps Towards Genetic Neural Networks, *Parallel Computing*, Vol. 14, pp. 249-260, 1990.

[17] L. Davis, Mapping Neural Networks into Classifier Systems, *Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA)*, Fairfax, Virginia, USA, 375-378, 1989.

[18] A. Wieland, R. Leighton, Geometric Analysis of Neural Network Capabilities, 1st *IEEE International Conference on Neural Networks*, Vol. III, pp. 385-392, San Diego, USA, 1987.

[19] D. Hush, B. Horne, Progress in Supervised Neural Networks: What's New since Lippman?, *IEEE Signal Processing Magazine*, Vol. 10, pp. 8-39, 1993.

[20] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1992.

[21] D. Whitley, A Genetic Algorithm Tutorial, *Statistics and Computing* Vol. 4, pp. 65-85, 1994.

[22] J. Schaffer, D. Whitley, L. Eshelman, Combination of Genetic Algorithms and Neural Networks: A Survey of the State of the Art, *International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pp. 1-37, Baltimore, USA, 1992.

[23] J. Torresen, *Parallelization of Back-propagation Training for Feed-Forward Neural Networks*, PhD Thesis, The Norwegian Institute of Technology, 1996.

[24] S. Foo, P. Saratchandran, N. Sundararajan, Parallel Implementation of Back-propagation Neural Networks on a Heterogeneous Array of Transputers, *IEEE Trans. Systems, Man and Cybernetics - Part B*, Vol. 27, No. 1, pp. 118-126, 1997.

[25] K. Stanley, B. Bryant, R. Miikkulainen, Evolving Adaptative Neural Networks with and without Adaptative Synapses, *Proceedings of the IEEE Congress on Evolutionary Computation*, 2003.

[26] N. Sundarajan, P. Saratchandran, *Parallel Architectures for Artificial Neural Networks: Paradigms and Implementations*, Wiley-IEEE Computer Science Press, November 1998.

[27] T. Crainic , M. Gendreau, Cooperative Parallel Tabu Search for Capacitated Network Design, *Journal of Heuristics*, Vol. 8, pp. 601-627, 2002.

[28] A. Muhammad, A. Bargiela, G. King, Fine-grained Parallel Genetic Algorithm: A Global Convergence Criterion, *Int. Journal of Computer Mathematics*, Vol. 73, No. 2, pp. 139-155, 1999.

[29] D. Pereira, J. Bezerra, A.D. Dória Neto, J.D. Melo, Instrumentação Inteligente em Rede Aplicada ao Processo de Medição de Vazão e BSW, *XV Congresso Brasileiro de Automática*, Gramado - RS, 2004.