Proceedings of the IV Brazilian Conference on Neural Networks - IV Congresso Brasileiro de Redes Neurais
pp. 078-083, July 20-22, 1999 - ITA, São José dos Campos - SP - Brazil

Treinamento de Redes Neurais Artificiais Utilizando Time Assíncrono

Paulo Akira Saito Junior, Cairo Lúcio Nascimento Júnior, Takashi Yoneyama
Instituto Tecnológico de Aeronáutica
Divisão de Engenharia Eletrônica
12.228-900 - São José dos Campos - SP - Brasil
E-mails: akira@ele.ita.cta.br, cairo@ita.cta.br, takashi@ita.cta.br

Abstract

The training of artificial neural networks can be seen as a hard optimization problem. Any algorithm used to solve this problem will have weak and strong features. In this article we consider the use of Asynchronous Teams to train artificial neural networks. An Asynchronous Team is a general computational structure where several different algorithms run in parallel in different computers and are applied at the same time to solve the same optimization problem. During the computation, several intermediate solutions are analysed and used as new starting points for the different algorithms. We show that, when compared with the solutions obtained by each individual algorithm, the use of the Asynchronous Team (controlled "mixture" of different algorithms) leads to a better solution, that is, better trained artificial neural networks. As a simple example, an artificial neural network is trained to recognize a few simple characters.

1. Introdução

1.1. Apresentação do Trabalho

O problema de Treinamento de Redes Neurais Artificiais pode ser visto como um problema de otimização, onde é desejado minimizar o erro quadrático médio entre a saída desejada e aquela produzida pela Rede Neural Artificial. Existem vários tipos de algoritmos para treinamento das Redes Neurais, alguns com forte supervisão e outros com necessidade reduzida de informações providas pelo meio ambiente.

Em geral, sabe-se que nenhum dos algoritmos de Treinamento de Redes Neurais Artificiais é completo. Alguns algoritmos apresentam boas características tais como alta velocidade de convergência. Entretanto, o erro quadrático médio na saída pode ainda ser relativamente alto. Outros algoritmos atingem pequeno erro quadrático médio na saída, porém apresentam em geral baixa velocidade de convergência.

Neste trabalho é proposta uma estrutura computacional que utiliza vários tipos diferentes de algoritmos de treinamento de redes neurais artificiais trabalhando em paralelo para obter um resultado melhor do que cada algoritmo trabalhando de forma independente. Os algoritmos trabalharão como um time, denominado Time Assíncrono, compartilhando informações disponibilizadas em uma área comum denominada de quadro negro.

Neste artigo mostramos:

- a adaptação de algoritmos para o treinamento de Redes Neurais Artificiais com vistas a utilização em Times Assíncronos;
- uma arquitetura de Time Assíncrono para o treinamento de Redes Neurais Artificiais;
- comparação entre os resultados obtidos com o Time Assíncrono proposto e os resultados obtidos com algoritmos clássicos operando isoladamente.

2. Treinamento de Redes Neurais Artificiais

Embora os Times Assíncronos não se limitem ao treinamento do Perceptron de Múltiplas Camadas [1] [2], este é adotado para a apresentação deste trabalho.

2.1. O Perceptron de Múltiplas Camadas

Este tipo de Rede Neural Artificial é constituída de uma *camada de entrada*, uma ou mais *camadas escondidas* e uma *camada de saída*. A função de ativação das unidades nas *camadas escondidas* é, em geral, não linear. A função das *camadas escondidas* é a recodificação dos padrões de entrada.

A figura 1 ilustra a estrutura de um Perceptron de Múltiplas Camadas com duas camadas escondidas. O número de camadas escondidas pode ser maior do que um, mas normalmente não ultrapassa a três devido a grande dificuldade do treinamento quando o número de camadas escondidas aumenta.

2.2. O Treinamento do Perceptron de Múltiplas Camadas

Existem vários algoritmos para o treinamento do Perceptron de Múltiplas Camadas. Alguns algoritmos utilizam informações do gradiente da função de erro de aproximação como o Back-Propagation [1] [2], enquanto outros algoritmos não se utilizam destas

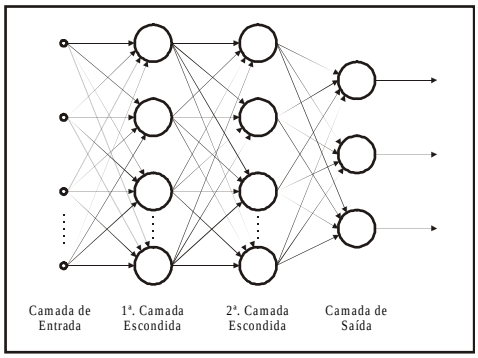

Figura 1: Perceptron de Múltiplas Camadas

informações, como o Algoritmo de Nelder-Mead [3], algoritmo genético [4], algoritmo de Hooke-Juves [3] e outros.

O algoritmo Back-Propagation baseia-se no mesmo princípio da Regra Delta, ou seja, tem o objetivo de minimizar o erro quadrático médio da camada de saída da Rede Neural Artificial e utiliza informações do gradiente da função erro quadrático médio. Para a utilização do algoritmo Back-Propagation é necessário que as funções de ativação usadas pelas unidades das camadas de saída e escondida sejam contínuas [9].

2.3. As Dificuldades dos Algoritmos de Otimização

Um algoritmo para resolver problemas de otimização pode apresentar determinados pontos fracos. No caso específico de Redes Neurais Artificiais existe ainda o problema de generalização da solução e de inicialização dos pesos da rede.

a) A questão da velocidade de convergência: muitos algoritmos utilizados para o treinamento de redes tendem a ter uma baixa velocidade de convergência em determinados períodos do treinamento da rede.

b) A questão do mínimo local: existe o risco do algoritmo ficar preso num dos mínimos locais. Mas, em algum lugar do espaço dos pesos sinápticos pode existir uma outra combinação de pesos que gerem um erro menor ainda.

c) A questão da dimensão e da complexidade do problema: uma Rede Neural Artificial é constituída de muitos neurônios, ou seja, as matrizes que contém os pesos sinápticos e os bias são de grande dimensão.

d) A questão da generalização: a generalização de uma rede é considerada boa quando a aplicação na entrada da rede de um conjunto de padrões não utilizados durante o treinamento, denominado *padrões de teste*, resulta em valores de saída corretos.

e) A questão da inicialização dos pesos da rede: a inicialização dos pesos da rede é normalmente feita usando valores aleatórios pequenos e em torno de zero, o que por vezes pode conduzir o algoritmo ao mesmo ponto de mínimo local. Assim, é interessante a investigação de métodos alternativos para a inicialização dos pesos da rede neural.

3. Times Assíncronos

3.1. Introdução

O treinamento de uma Rede Neural Artificial é um tipo de problema de otimização, onde se quer minimizar uma função custo. Neste caso a função custo é o erro quadrático médio da camada de saída da rede.

Observa-se que alguns dos algoritmos de otimização existentes apresentam algumas dificuldades para obter o ponto de mínimo global da função de custo.

Assim, propõe-se uma estrutura de algoritmos trabalhando de modo paralelo e assíncrono, cooperando um com o outro de modo a obter melhores resultados juntos do que obteriam separadamente. O paralelismo decorre do fato que cada algoritmo estará trabalhando em um computador independente dos outros. A estrutura é assíncrona pois não é exigida a sincronicidade entre os algoritmos, isto é, cada algoritmo é iniciado ou terminado apenas pelo seu controle local, ou seja, não existe inter-dependência forte entre os diferentes algoritmos.

Esta estrutura computacional é conhecida como Time Assíncrono (*A-Team*) [5]. Existem diferentes alternativas para a estrutura de Time Assíncrono. Uma alternativa seria um time formado por várias cópias do mesmo algoritmo, p. ex. Back-Propagation, com diferentes parâmetros (taxa de aprendizado, função de ativação, entre outros). Outra alternativa seria um Time Assíncrono formado por vários tipos de algoritmos. Uma terceira alternativa seria a combinação das duas alternativas anteriores, ou seja, vários tipos de algoritmos com várias cópias de cada tipo de algoritmo.

Estas diferentes combinações de algoritmos poderiam ser comparadas a um *time de futebol*. No primeiro caso, poderíamos ter um time formado apenas por *jogadores atacantes* ou por *jogadores de defesa*. Já no segundo, teríamos vários jogadores, mas apenas um jogador de cada tipo. No terceiro caso, o time seria formado por vários *atacantes*, vários *jogadores de defesa* e vários *jogadores de meio de campo*.

Os resultados das simulações levam a concluir que o terceiro caso parece ser aquele que produz os melhores resultados.

3.2. Componentes do Time Assíncrono

Os Times Assíncronos são formados por elementos autônomos e podem ser implementados através de redes de computadores. Cada computador da rede será um elemento pertencente ao Time Assíncrono. Um esquema do Time Assíncrono implementado pode ser visto na figura 2.

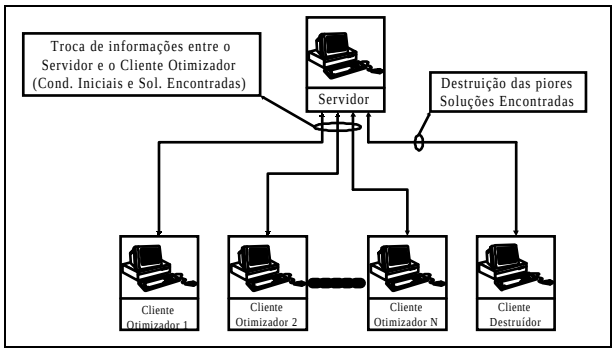

Figura 2: Um Exemplo de Time Assíncrono

Existe um computador que será considerado o *servidor* (*Server*). Todos os outros computadores serão considerados *clientes* (*Clients*). Os *clientes* podem conter: a) algoritmos de otimização, denominados *clientes otimizadores*, ou b) algoritmos de destruição de soluções inferiores, denominados *clientes destruidores*. Assim, temos:

1) Servidor: tem como principal função coordenar todas as informações do Time Assíncrono e sua estrutura em si. A figura 3 mostra o algoritmo do servidor. O servidor: a) pode criar novos clientes ou destruir clientes já existentes, b) é responsável por decidir quando o Time Assíncrono deve parar de trabalhar por ter alcançado aquele que seria considerado um mínimo suficientemente adequado para a função que está sendo minimizada, c) cria novos pontos iniciais para serem utilizados pelos clientes otimizadores.

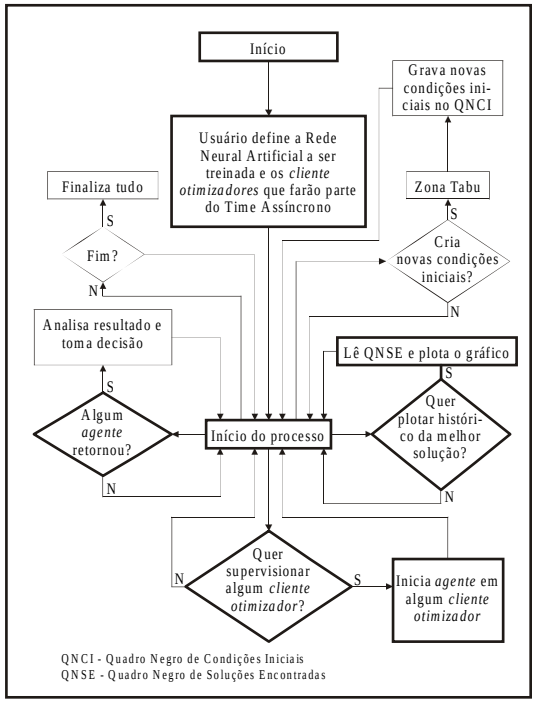

Figura 3: Algoritmo do Servidor

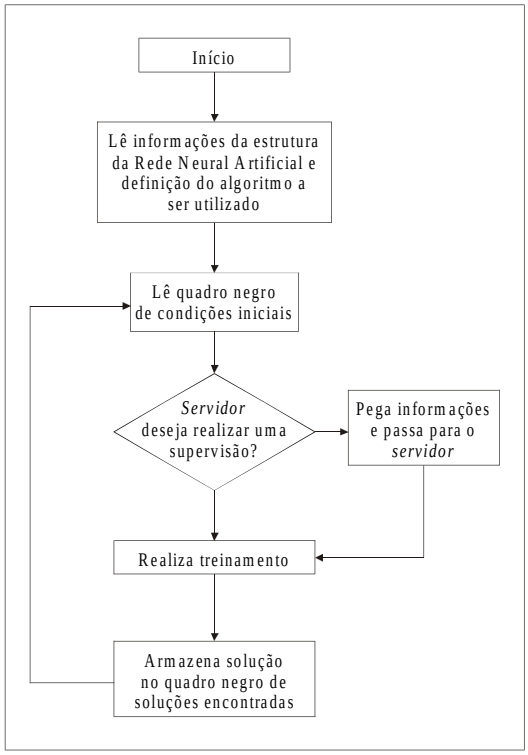

Figura 4: Algoritmo do Cliente Otimizador

2) Cliente Otimizador: tem a função de treinar a Rede Neural Artificial. Ele escolhe uma das *condições iniciais* e treina a rede usando um determinado número de iterações. A solução encontrada ao final do treinamento por este cliente é colocada em uma lista de soluções para acesso geral, denominada *Quadro Negro de Soluções*. Existem três situações que podem ocorrer: a) o algoritmo converge para um ponto de mínimo (seja ele global ou local) pois o erro quadrático médio na saída da rede (EQM) permanece constante; b) o algoritmo não converge pois o EQM fica oscilando; c) o algoritmo diverge pois o EQM cresce de forma instável. Existem vários algoritmos diferentes para o treinamento de redes, tais como Back-Propagation [1] [2] (que utiliza a informação do gradiente da função custo para encontrar o ponto de mínimo), Nelder-Mead [3] (também conhecido como "Poliedros Flexíveis" e que não usa a informação do gradiente da função custo). Ainda podemos contar com algoritmos como Zona Tabu [3] e Algoritmos Genéticos [6]. A figura 4 mostra o algoritmo do cliente otimizador.

3) Cliente Destruidor: a função do *cliente destruidor* é evitar, usando algum critério de decisão, que a lista de soluções armazenada no Quadro Negro de Soluções cresça de forma descontrolada. Diferentes critérios podem ser adotados pelos clientes destruitores, p. ex., simplesmente eliminar as N piores soluções ou calcular a probablidade de retirada da solução do Quadro de Soluções de acordo com a sua distância em relação à melhor solução até então encontrada (quanto

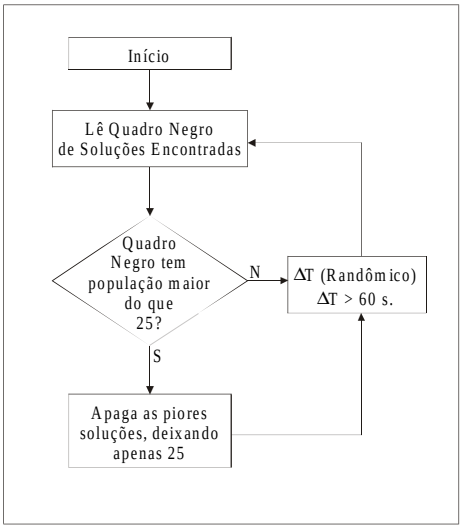

Figura 5: Algoritmo do Cliente Destruídor

maior for o erro maior será a probabilidade da solução ser retirada do Quadro de Soluções e vice-versa). Neste artigo, por simplicadade, utlizamos o primeiro tipo de cliente destruidor. A figura 5 ilustra o algoritmo do cliente destruidor.

4) Agentes: são algoritmos que são executados periodicamente nos *clientes otimizadores* a mando do *servidor* objetivando reportar a este o estado daqueles. Os agentes também podem alterar os parâmetros do cliente otimizador, dependendo do estado em que este estiver. Assim, por exemplo, o agente pode alterar a taxa de aprendizado do algoritmo do cliente otimizador caso o agente note que a razão para a não convergência do algoritmo foi uma taxa de aprendizado excessivamente alta.

3.3. Protocolo de Comunicação

Para que o Time Assíncrono possa trabalhar de modo correto, deve-se criar um protocolo de comunicação entre os componentes do time. Deve-se definir como os clientes vão se comunicar com o servidor, como o agente será iniciado em cada cliente e todas as outras tarefas que forem realizadas pelo Time Assíncrono.

Todas as informações trocadas entre os clientes e o servidor são feitas através de quadros negros. O Quadro Negro de Soluções Encontradas existe para que os clientes otimizadores armazenem as soluções encontradas por eles a cada treinamento. O Quadro Negro de Condições Iniciais é criado pelo servidor do time para disponibilizar as diferentes condições iniciais criadas pelo algoritmo de Zona Tabu [3]. Este algoritmo gera novos pontos iniciais na região onde foram obtidas as melhores soluções, entretanto as outras regiões continuam sendo exploradas, mas com um número menor de pontos.

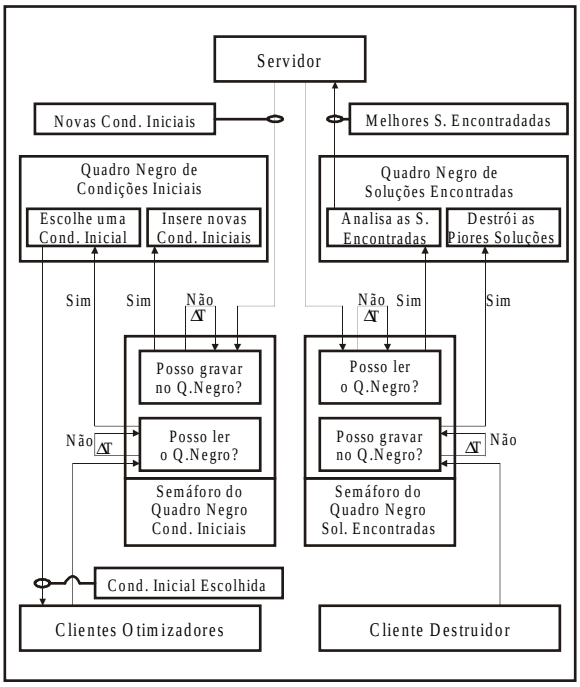

Figura 6: Fluxograma do funcionamento do Time

A figura 6 ilustra o fluxograma do Time Assíncrono durante o seu funcionamento.

A primeira tarefa a ser realizada para iniciar o trabalho do Time Assíncrono é definir a estrutura da Rede Neural Artificial a ser treinada. Esta definição é feita através do servidor. O segundo passo é criar alguns conjuntos de pesos sinápticos iniciais (condições iniciais).

A seguir são escolhidos pelo usuário os algoritmos que serão utilizados no Time Assíncrono. Isto é feito através do servidor.

Depois disto, o servidor disponibiliza estas informações aos clientes que escolhem um dos algoritmos disponíveis e a estrutura da rede a ser treinada.

Após a definição de qual algoritmo cada cliente otimizador executará, cada cliente otimizador escolhe uma das condições iniciais existentes e começa a treinar a rede. São criados, também, os clientes destruidores para evitar que a população de soluções encontradas e armazenadas no Quadro Negro de Soluções não ultrapasse o número máximo permitido de soluções.

Os agentes são “enviados”, de tempo em tempo, aos clientes para que ele possa analisar de forma local o comportamento do cliente. Com estas informações é possível modificar algum parâmetro do algoritmo ou mesmo modificar o algoritmo para que passe a trabalhar de uma maneira mais eficiente.

4. Aplicação Numérica

4.1. Apresentação do Problema

O problema a ser testado com o Time Assíncrono é o treinamento de uma Rede Neural Artificial para reconhecimento de imagens (padrões). As imagens utilizadas para o treinamento são mostradas na figura 7.

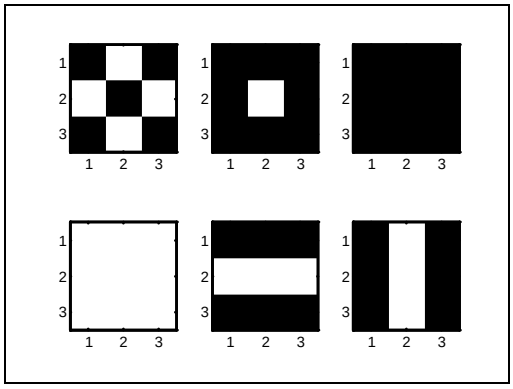

Figura 7: Imagens a serem treinados pela rede

O objetivo deste treinamento será o reconhecimento dos padrões. Cada imagem é constituída de 9 pontos formando um matriz 3 x 3. O Perceptron de Múltiplas Camadas é formado por três camadas: a) uma camada de entrada com 9 unidades (uma unidade para cada ponto da imagem); b) uma camada escondida, com 7 unidades; e c) uma camada de saída, com 6 unidades, onde cada unidade de saída representa um padrão diferente. Assim, uma saída típica da Rede Neural Artificial será uma única unidade igual a 1 e todas as outras iguais a 0. A estrutura da rede utilizada está ilustrada na figura 8.

A Rede Neural Artificial será treinada utilizando o algoritmo Back-Propagation isolado, o algoritmo Nelder-Mead também isolado e um Time Assíncrono composto por 3 clientes otimizadores Back-Propagation e 2 clientes otimizadores Nelder-Mead, conforme a tabela 1.

Os algoritmos Back-Propagation apresentam um número máximo de iterações próximo a 60.000, com um erro quadrático médio desejado de 0,0003 e taxas de aprendizado que variam de 0,008 a 0,05. Estes são os parâmetros que podem ser modificados. Cada um deles apresenta parâmetros diferentes para que possam chegar a resultados diferentes também.

Os algoritmos Nelder-Mead apresentam um número máximo de iterações próximo a 6.000, com erro quadrático médio de 0,003 e 0,005. No caso destes algoritmos, não existem parâmetros a serem modificados, por isto cada algoritmo é uma instanciação do algoritmo de Nelder-Mead.

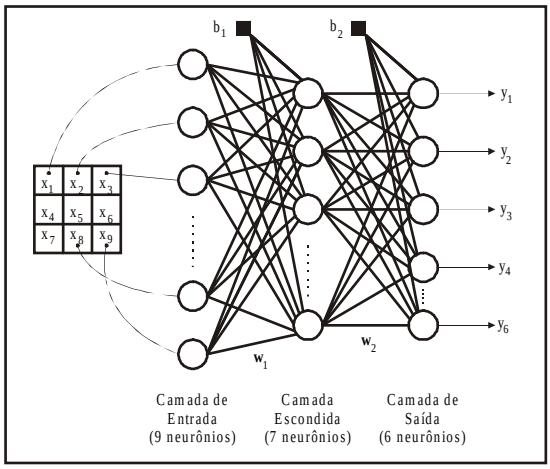

Figura 8: Estrutura da rede a ser treinada

4.2. Resultados para o Algoritmo Back-Propagation

Um primeiro teste foi realizado com o algoritmo back-propagation utilizando os seguintes parâmetros de algoritmo: número máximo de iterações: 60.000; erro quadrático médio desejado: 0,0005; taxa de aprendizado (η): 0,01.

O algoritmo alcançou o erro desejado após 47.000 iterações. A figura 9 ilustra a curva de convergência deste treinamento. O erro médio quadrático obtido foi: $4{,}9999 \times 10^{-4}$.

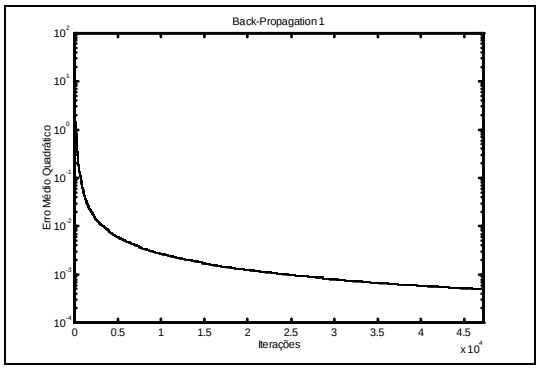

Figura 9: Curva de erro obtido com o Back-Propagation

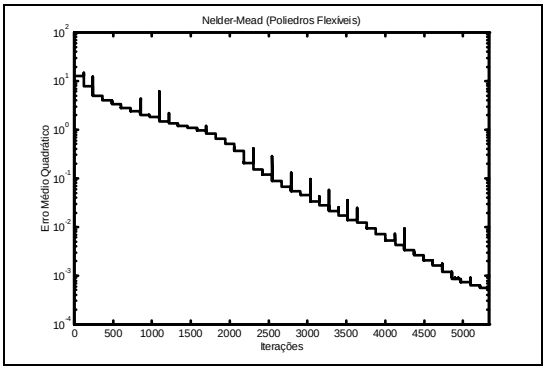

Figura 10: Curva de erro obtido com o Nelder-Mead

4.3. Resultados para o Algoritmo Nelder-Mead

Utilizou-se a versão padrão do algoritmo Nelder-Mead dado pelo Matlab. Assim, os parâmetros que podem ser modificados são: número máximo de iterações: 10.000; erro médio quadrático desejado: 0,0005.

O ponto de partida do treinamento também foi o mesmo do utilizado pelo algoritmo Back-Propagation. O algoritmo conseguiu alcançar o erro médio quadrático desejado em aproximadamente 5.500 iterações. A figura 10 ilustra a curva de convergência do treinamento com o Nelder-Mead. O erro médio quadrático obtido foi: $4{,}7122 \times 10^{-4}$.

4.4. Resultados para o Time Assíncrono

A tabela 1 a seguir mostra os parâmetros usados para cada cliente otimizador do Time Assíncrono. Neste caso o erro médio quadrático obtido foi: $2{,}9147 \times 10^{-4}$.

Tabela 1: Componentes do Time Assíncrono

	Nº max. de iterações	Erro Max. Desejado	η
"Back1" – 1	60.000	0,0003	0,01
"Back2" – 2	75.000	0,0003	0,008
"Back3" – 3	50.000	0,0003	0,05
"Neld1" – 4	5.500	0,005	—
"Neld1" – 5	7.000	0,003	—

A figura 11 ilustra o histórico das primeiras 160 observações do Quadro Negro de Soluções realizadas pelo servidor. Observando-se o gráfico, nota-se que nas primeiras 30 vezes que o servidor analisou o quadro negro, a melhor solução ainda era aquela que ele mesmo havia criado a partir da Zona Tabu. A partir daí, os *clientes otimizadores* com seus algoritmos de treinamento começaram a encontrar melhores soluções. É interessante notar que as soluções intermediárias não foram encontradas exclusivamente por um único tipo de algoritmo.

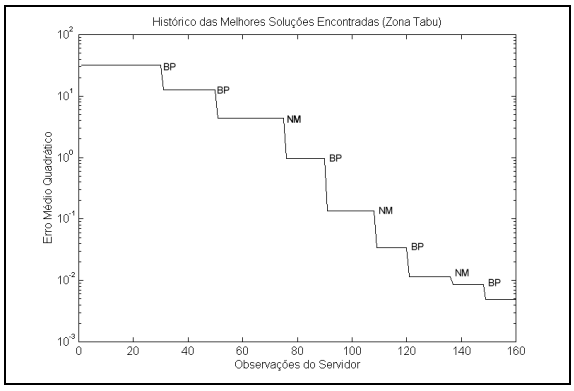

Figura 11: Histórico das Melhores Soluções

5. Conclusões

Neste artigo mostramos como uma rede neural artificial pode ser treinada utilizando-se um Time Assíncrono. Mostramos também a estrutura e os componentes do Time Assíncrono, assim como o seu protocolo de comunicação.

Os resultados obtidos em nossas simulações mostram que é vantajoso o uso de um Time Assíncrono bem escolhido para o treinamento de redes neurais artificiais, visto que a combinação adequada de algoritmos com diferentes características produz redes com menor erro médio quadrático na saída.

Em trabalhos futuros, investigaremos: a) o efeito de um aumento na variedade dos algoritmos usados nos clientes otimizadores, b) diferentes estratégias nos clientes destruidores, c) outros exemplos de aplicação.

Referências Bibliográficas

[1] Zurada, J. M., *Introduction to Artificial Neural Systems*, New York, West Publishing Company, 1992.

[2] Nascimento Jr., C. L. & Yoneyama, T., *Inteligência Artificial em Automação e Controle*, São Paulo, Edgard Blücher, 1999 (a ser publicado).

[3] Himmelblau, D. M., *Applied Nonlinear of Optimization*, New York, McGraw Hill, 1972.

[4] Montana, D. J. & Davis, L., Training Feedforward Neural Networks Using Genetic Algorithms, *Proceedings of the 11th Int. Joint Conf. on Artificial Intelligence (IJCAI-89)*, Detroit, pp. 762-767, 1989.

[5] Talukdar, S.; Baerentzen, L.; Gove. A & Souza, P., *Asynchronous Teams: Cooperation Schemes For Autonomous Agents*, 1989.