ESCOLA
POLITÉCNICA DE
PERNAMBUCO

Trabalho de Conclusão de Curso

Engenharia de Computação

Um algoritmo baseado em *Artificial Bee Colony* para treinamento de Redes Neurais Artificiais.

Autor: Nathalia Maria Temudo

Orientador: Prof. Dr. Mêuser Jorge Silva Valença

UNIVERSIDADE
DE PERNAMBUCO

ESCOLA
POLITÉCNICA DE
PERNAMBUCO

Nathalia Maria Temudo

Um algoritmo baseado em *Artificial Bee Colony* para treinamento de Redes Neurais Artificiais.

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, junho de 2010.

A memória do meu avô Pedro Wanderley Temudo.

Agradecimentos

Primeiramente, agradeço a Deus por ter me dado saúde e força para chegar até esse momento.

Agradeço em especial a minha mãe, Sandra Maria Temudo, a minha avó, Neuza Jatobá Temudo e a toda minha família por tudo! Vocês são minha base e minha inspiração para seguir em frente.

Agradeço em especial a minha tia e madrinha Neuza Maria Temudo por todo apoio e incentivo nos meus estudos ao longo dessa jornada.

Agradeço a meu namorado: Gilliard Alan, pelo apoio, compreensão e suporte no desenvolvimento deste trabalho e pelo companheirismo, atenção e carinho nos demais momentos.

Agradeço ao amigo Arthur Minduca pelas discussões técnicas sobre temas afins e pelo companheirismo dispensado na execução deste trabalho.

Agradeço também aos amigos Caio Cesar, Diego Siqueira, João Fausto, Rafael Galvão, Michele Leitão, Lorena Tablada e Saulo Medeiros e demais colegas pelo apoio mútuo e a amizade durante esses cinco anos de luta.

Agradeço as minhas amigas Amanda Gabriela, Iana Rafaela, Maria Lívia, Mayara Riselle, Nathaly Menelau, Roberta Lucena e Thalita Menelau que me apoiaram sempre que possível e foram responsáveis por bons momentos de descontração.

Agradeço ao Prof. Dr. Mêuser Jorge Silva Valença, pela grande disponibilidade, apoio nas dúvidas técnicas e boa vontade durante toda orientação desse trabalho. Por fim, agradeço a todos os professores da graduação pela boa formação que recebi.

Resumo

Este trabalho propõe uma metodologia de otimização de pesos de Redes Neurais Artificiais (RNAs) com algoritmo de otimização. O sistema proposto busca ajustar os pesos das conexões da rede, dispensando o uso de algoritmos de treinamento por correção de erro. Na metodologia proposta, foi utilizado o algoritmo de otimização baseado no comportamento inteligente das colônias de abelhas, o *Artificial Bee Colony* (ABC), o qual é a principal inovação neste trabalho. Desta forma, esta dissertação possui o objetivo primário de avaliar e comparar o desempenho da metodologia proposta com o algoritmo *BackPropagation* no treinamento da rede neural. Os experimentos foram conduzidos com o propósito de otimizar redes *Multi-Layer Perceptron* (MLP) para problemas de classificação e predição. Os critérios utilizados para a análise de performance do método foram o tempo de convergência, a taxa de acerto e o erro médio probabilístico absoluto (EMPA). Os resultados obtidos foram satisfatórios e indicam que o método proposto possui qualidade de resposta superior ou equivalente a muitos métodos encontrados na literatura.

Palavras-chave: Redes Neurais Artificiais, Otimização de Pesos e *Artificial Bee Colony*.

Abstract

This study introduces a methodology for the optimization of weights of Artificial Neural Networks (ANNs) with optimization algorithms. The proposed system seeks to make the adjustment of weights, avoiding the use of error correction-based training algorithms. In the methodology proposed, we used the optimization algorithm based on the intelligent foraging behavior of a honey bee swarm, *Artificial Bee Colony* (ABC), which is the main innovation of this study. Thus, this primary goal of this dissertation is to evaluate and compare the performance of the proposed methodology with the BackPropagation algorithm in the training of neural networks. The experiments were conducted to optimize the networks Multi-Layer Perceptron (MLP) for purposes of classification and prediction. The criteria used to analyse the performance of the method are the convergence speed, success rate and Mean Average Percentual Error (MAPE). The results were satisfactory and indicate that the proposed method has a quality of response better than or equal many methods in the literature.

Keywords: Artificial Neural Networks, Optimization of Weights and Artificial Bee Colony.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

ABC – *Artificial Bee Colony*

ACO – *Ant Colony Optimization*

EMPA – Erro Médio Probabilístico Absoluto

EMQ – Erro Médio Quadrático

MLP – *Multi-Layer Perceptron*

PSO – *Particle Swarm Optimization*

RN – Rede Neural

RNA – Rede Neural Artificial

Introdução

Este capítulo se inicia descrevendo o problema e a motivação do desenvolvimento deste trabalho de conclusão de curso. Posteriormente são expostos seus principais objetivos. Por fim, ele é encerrado mostrando o conteúdo abordado nos capítulos seguintes.

1.1 Motivação e problema

Redes Neurais Artificiais (RNAs) são técnicas computacionais que foram inspiradas no comportamento biológico do cérebro. Assim como um cérebro humano, cada rede neural é formada por vários neurônios conectados. Um neurônio é capaz de resolver tarefas simples, mas quando agrupados, formam um sistema conexionista bastante eficiente que é capaz de aprender através de um conjunto de exemplos. Este sistema assim formado apresenta diversas características importantes tais como: capacidade de processamento paralelo, capacidade de generalização (por meio de um processo de aprendizado) e tolerância a falhas. Por essa capacidade de aprendizado, as Redes Neurais (RNs) têm sido aplicadas, atualmente, em diversos tipos de problemas, tais como classificação de padrões, previsões, aproximações de funções, otimização e memórias associativas [1,2].

Na predição ou classificação de padrões, as RNs utilizam técnicas de aprendizado supervisionado para mapear dois conjuntos de dados: os padrões de exemplos (atributos de entrada) e os exemplos de classificação (atributos de classe). Após esse mapeamento, a rede poderá classificar padrões desconhecidos para uma das classes dos problemas envolvidos. Entre as redes neurais, o modelo mais utilizado na literatura para classificação e predição é o *perceptron* de múltiplas camadas ou apenas MLP (do inglês *Multi-Layer Perceptron*) [3].

A rede MLP pode operar em dois modos distintos para a predição ou classificação de padrões: treinamento e processamento. No treinamento, vetores de entrada extraídos do conjunto de dados de treinamento são apresentados a rede e

os vetores de saída são calculados com a finalidade de diminuir a diferença entre a saída calculada e a desejada (erro de treinamento). De uma maneira geral, para se modelar uma rede neural se considera 3 conjuntos de dados: um para ajustar os pesos (conjunto de treinamento), um para parar o treinamento (conjunto de validação cruzada) e um para teste (conjunto de teste).

Para o treinamento da rede, são necessárias duas etapas. A primeira etapa consiste na seleção da arquitetura apropriada para o problema e a segunda consiste no ajuste dos pesos que conectam os neurônios. O ajuste dos pesos geralmente é realizado através do algoritmo *BackPropagation*, o qual utiliza gradiente descendente do erro de classificação da rede como informação de retorno para a adaptação dos pesos com a finalidade de diminuir o erro de treinamento.

Nesse contexto, o *BackPropagation* mostra-se satisfatório para a resolução de problemas não-lineares separáveis, mas possui convergência lenta e não há garantia que um treinamento satisfatório seja sempre alcançado [4]. Não raramente, ele pode ficar preso em mínimos locais, o que o impede de avançar em direção ao mínimo global desejado.

Sendo assim, o MLP não conseguirá obter uma generalização eficiente no modo de processamento que é quando a rede produz a sua própria resposta para um padrão de exemplos desconhecido (conjunto de teste). Para suprir essa limitação encontrada no *BackPropagation*, alguns trabalhos de pesquisa vêm empregando técnicas meta-heurísticas de otimização como alternativas viáveis e eficazes [5] para encontrar valores adequados dos pesos das conexões evitando que os mesmos fiquem presos em mínimos locais, bem como com o objetivo de agilizar o processamento.

Nas técnicas meta-heurísticas, uma busca global é feita em vários pontos do espaço de soluções ao mesmo tempo para se encontrar o melhor conjunto de pesos de conexões para o problema a ser resolvido pela RN. O fator que vai indicar o quão bem está uma solução da técnica aplicada em relação ao problema tratado será o percentual da taxa de acerto da MLP para o conjunto de testes quando o problema for de classificação e o erro probabilístico médio absoluto (EMPA) quando o problema for de predição.

Entre as técnicas mais utilizadas na comunidade científica, podemos destacar Otimização por Colônia de Formigas (ACO, do inglês *Ant Colony Optimization*), Otimização por Enxame de Partículas (PSO, do inglês *Particle Swarm Optimization*) [5].

O presente estudo se dedicou a solucionar a limitação do *BackPropagation* através substituição do mesmo no ajuste dos pesos das conexões da MLP pelo algoritmo das abelhas (ABC, do inglês *Artifical Bee Colony*). O ABC foi implementado de acordo com [6] e a realização dos ajustes de parâmetros do algoritmo foram feitas de acordo com o problema a ser solucionado.

1.1.1 Objetivo Geral

Neste trabalho aplicaremos o algoritmo de Colônia de Abelhas para ajuste dos pesos de conexões no *Perceptron* de Múltiplas Camadas.

1.1.2 Objetivos Específicos

- Implementar uma RNA com *BackPropagation*
- Implementar uma RNA com *Artificial Bee Colony*
- Realizar simulações com bases de dados nas RNAs implementadas
- Comparar o desempenho entre os métodos de treinamento da RNA

1.2 Estrutura da monografia

Este trabalho está dividido em 5 capítulos. No primeiro capítulo foi iniciada a introdução ao tema Um algoritmo baseado em *Artificial Bee Colony* para treinamento de Redes Neurais Artificiais, assim como a motivação e a caracterização do problema.

Capítulo 2: Revisão Bibliográfica

Neste capítulo, abordaremos o conteúdo teórico necessário para desenvolver o tema proposto: Redes Neurais Artificiais e *Artificial Bee Colony*. Também apresentaremos as bases de dados que serão utilizadas nos experimentos realizados neste trabalho.

Capítulo 3: Metodologia

Neste capítulo, abordaremos a aplicação do *Artificial Bee Colony* no treinamento da Rede MLP. Sendo assim, explicaremos a técnica do *Artificial Bee Colony* para ajustar os pesos de conexão da MLP.

Capítulo 4: Resultados Obtidos

Neste capítulo, realizaremos simulações com as bases de dados apresentadas no capítulo 2 na MLP com abelhas (ABC) e desenvolvida no capítulo 3. Além disso, compararemos os resultados obtidos das simulações com os resultados obtidos por uma MLP treinada com o *BackPropagation .*

Capítulo 5: Conclusão

Neste capítulo, abordaremos uma visão geral do que foi visto, dificuldades encontradas, discussões e conclusões obtidas, além de melhorias e trabalhos futuros.

Revisão Bibliográfica

Neste capítulo abordaremos todo o conteúdo teórico necessário para solucionar o problema descrito na introdução. A seção 2.1 mostra como funcionam as Redes Neurais Artificiais, uma técnica de Inteligência Computacional para solucionar vários problemas de classificação e predição. Em seguida, a seção 2.2 discorre sobre o Algoritmo das Abelhas, um algoritmo de otimização baseado no comportamento inteligente das abelhas na natureza que visa, entre outras coisas, melhorar o desempenho das RNAs.

2.1 Redes Neurais Artificiais (RNAs)

São técnicas de inteligência artificial baseadas na estrutura do cérebro humano para adquirir conhecimento através do treinamento chamado de aprendizado. E como tal, são compostas por várias camadas de processamento que por sua vez são constituídas por unidades básicas de processamento conhecidas como neurônios artificiais [7]. Nesse contexto, faz-se aqui uma explanação do neurônio biológico de forma que se possa compreender a metáfora para a formação e o funcionamento do neurônio artificial.

2.1.1 Neurônio Biológico

É a unidade básica do sistema nervoso humano (Figura 1). O cérebro é formado por um conjunto de neurônios conectados [8] e cada neurônio é formado basicamente por dendritos (conjuntos de terminais de entrada), corpo celular e por axônios (longos terminais de saída referidos como fibras nervosas) [8].

Os neurônios se comunicam através de sinapses nervosas, região onde dois neurônios entram em contato e transmitem os impulsos nervosos entre si. Essa informação recebida pelos dendritos é processada no corpo celular e propagada pelos terminais axônicos. No entanto, a informação só será transmitida para outros

neurônios se a intensidade do impulso nervoso for maior que o limiar exitatório (Lei do Tudo ou Nada).

Baseado nessas características foi desenvolvido a unidade de processamento de informação fundamental para o funcionamento de uma rede neural: o neurônio artificial.

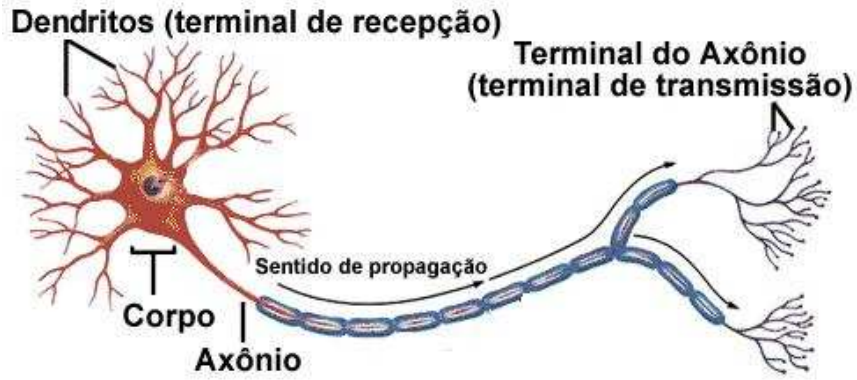

Figura 1. Neurônio biológico.

2.1.2 Neurônio Artificial

A primeira proposta do modelo matemático de um neurônio biológico foi feita por McCulloch Pitts em 1943 (Figura 2) [9].

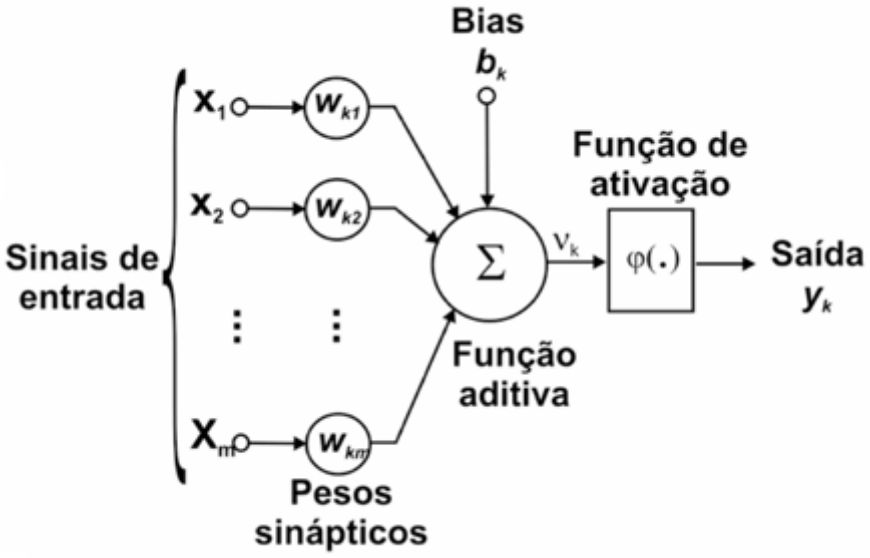

Figura 2. Modelo matemático de um neurônio biológico [10]

Como mostra a Figura 2, no neurônio artificial trocam-se os dendritos por entradas, cuja ligação com o corpo celular artificial é dado por elementos denominados pesos que simulam as sinapses. Nesse contexto, cada sinal (x_i) na entrada da sinapse j conectado ao neurônio $\mathcal{K}$ é multiplicado pelo peso sináptico (w_{ij}). Percebe-se que o sinal (x_o) é predefinido e, quando multiplicado pelo seu peso, representa o valor do limiar exitatório [11]. É relevante esclarecer que o primeiro índice do peso se refere ao neurônio em questão e o segundo se refere ao terminal de entrada ao qual o peso se refere [10]. Ao contrário de uma sinapse no cérebro, o peso sináptico no neurônio artificial pode estar num intervalo que abrange valores positivos e negativos.

Os sinais captados pelas entradas são processados pela função Soma. Esse processamento é feito através de uma soma ponderada de todas as entradas com seus respectivos pesos (net), dada pela Equação 2.1, a qual é usada como parâmetro pela função de ativação para retornar o valor de saída do neurônio (y), Equação 2.2.

$$net_i = \sum_{i=0}^{n} x_i \text{ x } w_{ij} \tag{2.1}$$

$$y = f(net_i) \tag{2.2}$$

Nesse modelo apresentado, a inteligência está nos pesos que o constitui, os quais devem ter valores de forma que, para cada entrada, a saída calculada se aproxime ao máximo da saída desejada [10].

As primeiras redes neurais que utilizaram este modelo de neurônio foram o *Perceptron* e a *Adaline*. Estas redes têm como característica comum possuírem apenas uma camada de neurônios que correspondem às entradas e uma camada de saída que corresponde à resposta da rede. A diferença entre elas é que a primeira lida com saídas discretas e a segunda permite respostas no universo contínuo. Essa diferença está diretamente ligada com a função de ativação de cada neurônio. Enquanto a função mais utilizada para o *Perceptron* é a degrau para a *Adaline* é a sigmóide logística, como podemos ver nas equações 2.3 e 2.4, respectivamente [11].

$$y_i = \begin{cases} 1, \ para\ net_i \ \geq 0 \\ 0, \ para\ net_i \ < 0 \end{cases} \quad (2.3)$$

$$y_i = \frac{1}{1+ e^{-net_i}} \quad (2.4)$$

Entretanto o potencial destas redes é limitado uma vez que só podem resolver problemas linearmente separáveis. Portanto, uma generalização da *Adaline* através da utilização de pelo menos uma camada escondida gera uma rede de grande capacidade de generalização. Logo, a partir da união de neurônios artificiais em pelo menos 3 (três) camadas temos a construção de uma Rede Neural Artificial Multicamadas (RNA). Entre as diversas RNAs, a mais famosa e a mais utilizada é o *Perceptron* de Múltiplas Camadas ou apenas MLP (do inglês *Multi-Layer Perceptron*) pelo fato de resolver com sucesso problemas não linearmente separáveis, além de possuir uma implementação simples. [3]

2.1.3 RNAs *Multi-Layer Perceptron* (MLP)

A extensão do *Perceptron* de uma única camada forma a rede MLP e, sendo assim, sua arquitetura consiste em uma camada com os sinais de entrada responsável por obter esses sinais sem realizar qualquer tipo de processamento, uma ou mais camadas intermediárias conhecidas também como camadas ocultas (escondidas) e uma camada com os sinais de saída (Figura 3). As camadas são formadas por neurônios artificiais e cada neurônio possui função de ativação, que geralmente é a sigmóide logística.

É importante frisar que o número de camadas intermediárias, a quantidade de neurônios em cada camada, a função de ativação utilizada, a semente para inicialização aleatória dos pesos, entre outros, são fatores importantes para a arquitetura da rede, e que influenciarão no resultado final.

Nota-se, então, que a rede possui alto grau de conectividade permitindo interação entre as unidades que a compõe. Isso é um grande diferencial, pois redes com mais de uma camada oculta permitem solucionar problemas complexos através

da aproximação de qualquer função contínua, enquanto que redes com uma camada são capacitadas a resolver apenas problemas linearmente separáveis.

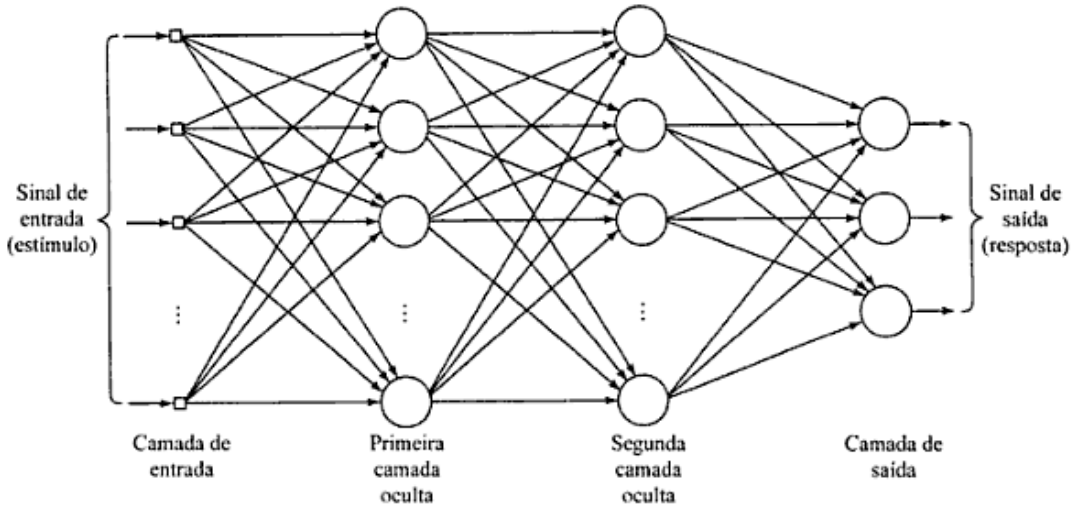

Figura 3. Grafo arquitetural de um P*erceptron* de Múltiplas Camadas [10]

Para resolver tais problemas, a rede necessita explorar sua capacidade de aprendizado (uma das características de todas as RNAs) e dentre os 3 tipos de aprendizado - supervisionado, não-supervisionado e por reforço [7] – o que se aplica ao da MLP é o supervisionado.

No aprendizado supervisionado há o papel do professor que avalia as respostas da rede de acordo com as respostas corretas que possui. Sendo assim, o erro da rede será calculado e os pesos da rede serão ajustados para se adequar melhor as respostas do professor. Entre os algoritmos de aprendizado supervisionado, o mais difundido nesse caso é o *BackPropagation*, que significa propagação recursiva.

2.1.4 BackPropagation

O algortimo *BackPropagation* baseia-se no aprendizado por correção de erros e essa correção é executada em 2 fases: fase *foward* e fase *backward* [11] com o auxílio de dois parâmetros importantes: a *taxa de aprendizado* (α) e o *momentum* (β). A *taxa de aprendizado* revela o quão grande é o passo na direção do erro e o *momentum* acelera a convergência da rede e, assim, diminui a incidência de mínimos locais. [10]

Na fase *foward*, temos a propagação progressiva do sinal da camada de entrada para a camada de saída e assim, pode-se calcular o erro e a resposta da rede. Para cada neurônio do índice *j* da camada oculta, o cálculo do erro δ_j é dado pela Equação 2.5, onde $f'(net_j)$ é a derivada da função de ativação, δ_i é o erro propagado pelo i-ésimo neurônio da camada imediatamente a frente e $w_{i,j}$ é o peso j do neurônio i. Já para o neurônio da camada de saída, o cálculo do erro δ_j é realizado de acordo com a Equação 2.6, onde d_j é a saída desejada, y_j é a saída encontrada pela rede e $f'(net_j)$ é a derivada da função de ativação.

$$\delta_j = f'(net_j) \sum_i \delta_i w_{i,j} \tag{2.5}$$

$$\delta_j = (d_j - y_j) f'(net_j) \tag{2.6}$$

Na fase *backward*, desde a camada de saída até a camada de entrada são feitas as alterações nos pesos sinápticos da rede. As alterações dos pesos são executadas de acordo com a Equação 2.7, onde $w_{i,j}(t+1)$ é o valor do novo peso do neurônio j no instante t+1, $w_{i,j}(t)$ é o valor do i-ésimo peso do neurônio j no instante t (instante atual), α é a taxa de aprendizado, δ_j é o erro do neurônio j, x_i é o sinal de entrada do i-ésimo neurônio, β é o *momentum* e $w_{i,j}(t-1)$ é o valor do i-ésimo peso do neurônio j no instante t - 1 (instante atual).

$$\Delta w_{i,j}(t+1) = w_{i,j}(t) + \alpha \delta_j x_i + \beta (w_{i,j}(t) - w_{i,j}(t-1)) \tag{2.7}$$

No aprendizado supervisionado por correção de erros, o conjunto de padrões é apresentado à rede MLP várias vezes, o que determina o número de ciclos da rede. A cada ciclo os pesos são ajustados para que a resposta obtida pela rede se aproxime cada vez mais do padrão de respostas desejado. Determinar até onde a rede deve treinar dentre uma quantidade de ciclos não é fácil, pois se ela for treinada demasiadamente, ela pode decorar as repostas e perder sua capacidade de

generalização. Para que isso não ocorra, utilizam-se critérios de parada e o mais difundido é o de validação cruzada.

2.1.5 Critério de parada: Validação Cruzada

O critério de validação cruzada divide o conjunto de padrões em 3 tipos: treinamento, validação e teste. Geralmente essa divisão é feita na seguinte proporção: 50%, 25% e 25%, respectivamente. Após a utilização do algoritmo de treinamento em cada ciclo da rede, a rede treinada com os pesos ajustados é testada com o conjunto de padrões de validação. A rede nunca utiliza esse conjunto para treinar a rede e como é algo totalmente novo para a mesma, ela pode continuar generalizando enquanto o erro de validação cruzada estiver diminuindo.

No entanto, quando a quantidade corrente de ciclos de validação cruzada atingir o número máximo de ciclos de validação, o treinamento será interrompido (Figura 4). Após isso, o conjunto de testes é submetido à rede para se realizar a avaliação de desempenho.

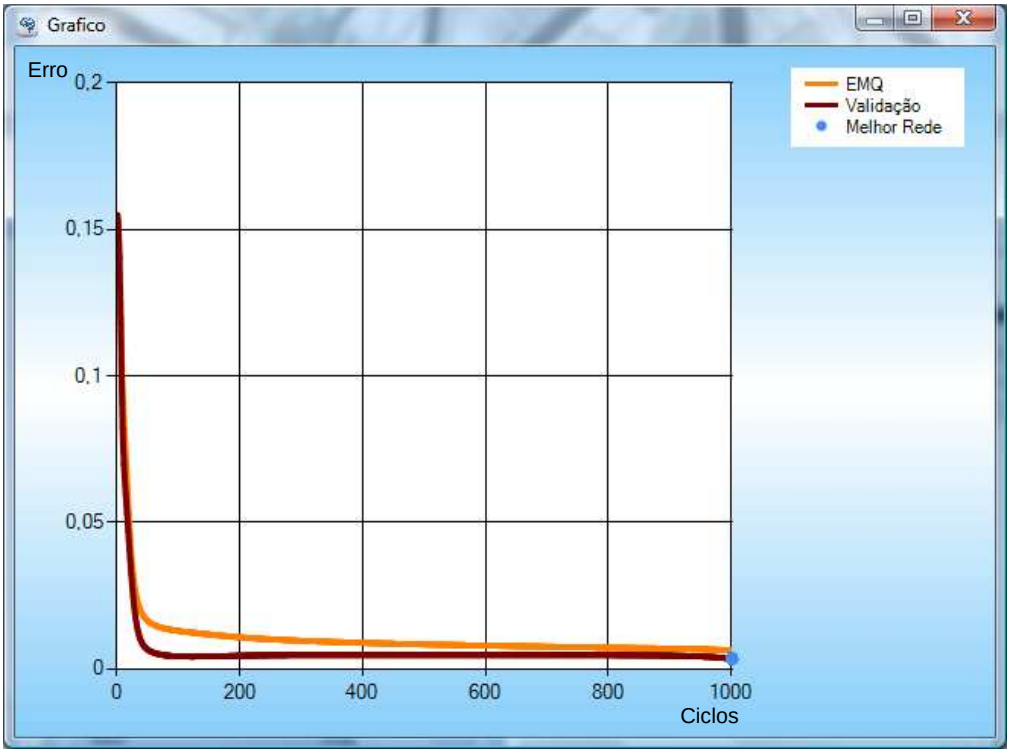

Figura 4. Validação Cruzada

2.2 Artificial Bee Colony (ABC)

Artificial Bee Colony (ABC) ou Algoritmo de Colônia de Abelhas Artificiais foi proposto por Karaboga em 2005 com o intuito de resolver problemas de otimização baseando-se no comportamento inteligente das abelhas na natureza [12, 13]. No mundo da Inteligência Computacional, é um algoritmo tão simples quanto o PSO, usando apenas alguns ajustes de parâmetros, tais como tamanho da colônia de abelhas e número máximo de ciclos.

2.2.1 Abelhas na natureza

Uma colônia de abelhas pode estender-se por grandes distâncias (cerca de 10 km) e em múltiplas direções simultaneamente para explorar diversas fontes de alimentos (Figura 5). Para explorar bem as fontes de alimento a colônia faz uso da seguinte política: as fontes de alimentos mais ricas de néctar receberão mais abelhas para a extração do néctar em detrimento das fontes mais pobres [14].

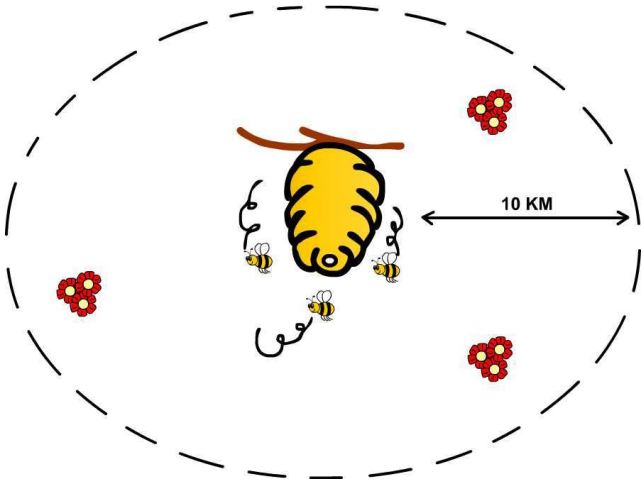

Figura 5. Colônia de abelhas com fontes de alimentos

Numa colônia, existem diversos tipos de abelhas, mas as abelhas utilizadas no algoritmo são de três tipos apenas: campeiras (*scout*), operárias (*employed*) e seguidoras (*recruited*). As campeiras são responsáveis por procurar fontes de alimentos, a partir do momento que uma abelha campeira explora uma fonte de alimento, ela torna-se uma abelha operária. Quando a fonte de alimento esgota, a abelha operária volta a ser uma abelha campeira. E as abelhas seguidoras são as abelhas que ajudam as operárias a explorar a fonte de alimento.

Na época da colheita, as abelhas campeiras procuram aleatoriamente por fontes de alimento onde o néctar é abundante, fácil de ser extraído e rico em açúcar. Ao retornar a colméia, as abelhas campeiras transmitem informações sobre as fontes de alimentos encontras para as abelhas seguidoras através da "*waggle dance*" (dança do requebrado – Figura 6) com o intuito de recrutá-las [14] para ajudá-las na exploração da fonte de alimento.

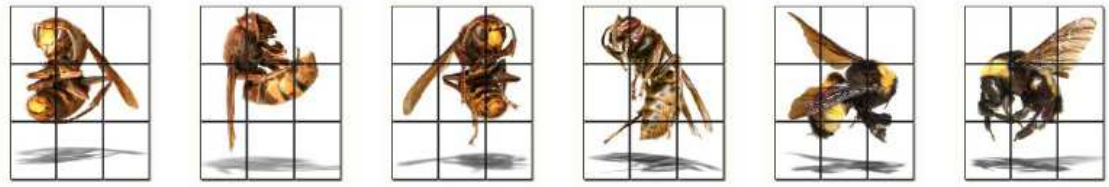

Figura 6. Passos da "*Waggle dance*" [12]

Nessa dança, três informações essenciais do ambiente externo são remetidas às abelhas seguidoras: a direção e a distância em relação à colméia e a qualidade da fonte de alimentação. A direção é transmitida por meio do ângulo que a abelha faz com a colméia e o sol durante a dança como mostra Figura 7 a. A distância, por sua vez, é comunicada pela duração da dança. Já a qualidade, é obtida por meio da frequência da dança, ou melhor, pela velocidade dos círculos de volta feitos pela abelha dançarina como mostra a Figura 7b.

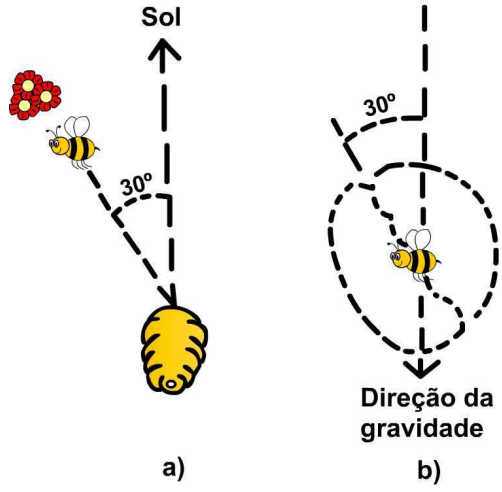

Figura 7. Informações obtidas da "*Waggle dance*"

Após a dança, a abelha operária junto com suas abelhas seguidoras volta à fonte de alimento para explorá-la. Lembrando que quanto mais promissora for uma fonte de alimento, mais abelhas extrairão seu néctar, o que permite um recolhimento

mais rápido e eficiente. Sendo assim, o nível de alimento de uma fonte sempre é mensurado e com isso as fontes serão ranqueadas e divididas entre melhores e não melhores. As fontes melhores serão divididas entre pobres e ricas e ainda continuarão sendo anunciadas na "*waggle dance*" para que mais abelhas explorarem essas fontes. Já as fontes não melhores serão abandonas, pois o alimento não tem uma qualidade que compense a exploração e as abelhas operárias associadas a essas fontes tornar-se-ão campeiras e voltam a procurar novas fontes de alimentos.

2.2.2 Algoritmo de abelhas

O algoritmo foi inspirado, como mencionado anteriormente, no comportamento inteligente das abelhas na natureza para encontrar, de forma otimizada, fontes de alimentos, as quais são pontos no espaço de busca do algoritmo [6]. A Tabela 1 mostra os principais parâmetros do algoritmo e a Figura 8 mostra o fluxograma do algoritmo.

Tabela 1. Parâmetros do algoritmo de abelhas

Sigla	Descrição
nc	Quantidade de abelhas campeiras
nr	Quantidade das fontes ricas
nmf	Quantidade das melhores fontes (ricas + pobres)
nsr	Quantidade de abelhas seguidoras para as fontes ricas
nsp	Quantidade de abelhas seguidoras para as fontes pobres
nciclos	Quantidade de ciclos do algoritmo

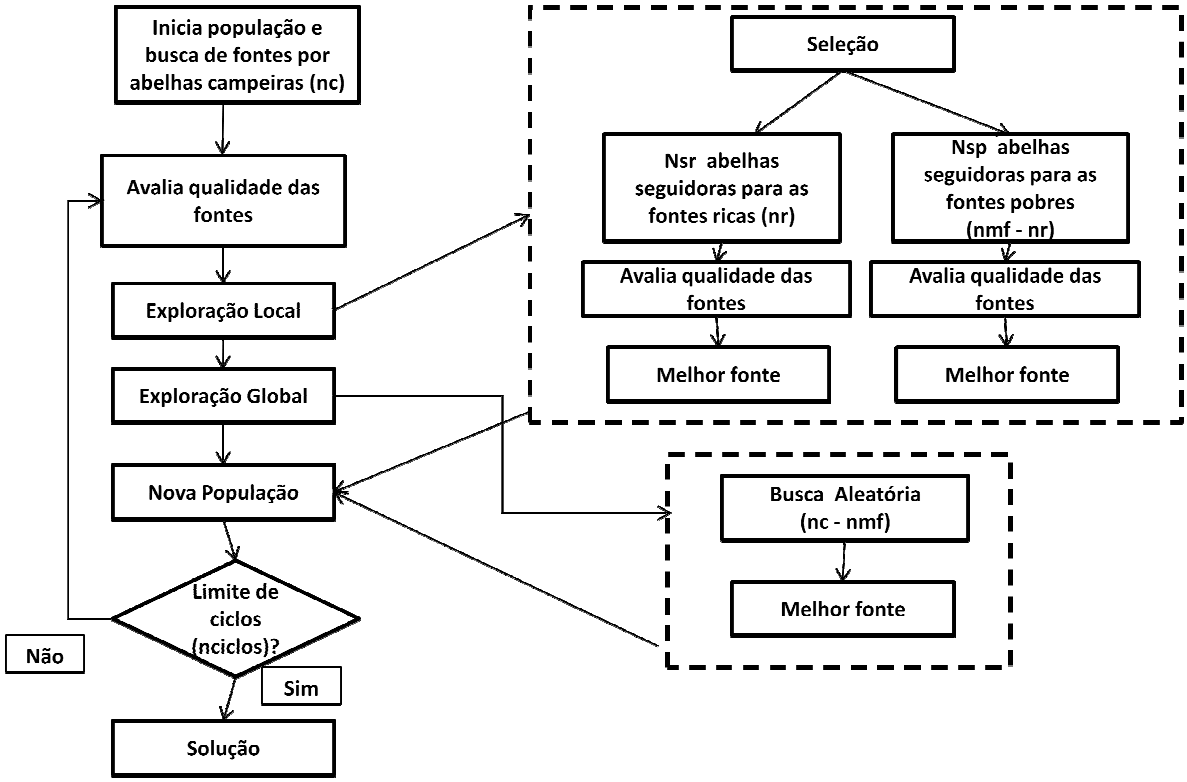

Figura 8. Fluxograma do algoritmo das abelhas

A população das abelhas é iniciada com *nc* abelhas campeiras. As abelhas campeiras procuram aleatoriamente por conjuntos de pesos da rede MLP dentro de um espaço de busca e avaliam esses conjuntos de acordo com o EMQ deles, pois quanto menor for o EMQ, melhor será o conjunto de pesos.

A partir dessa avaliação, os melhores conjuntos de pesos (nmf) das abelhas operárias, os quais possuem os melhores resultados para EMQ, serão repartidas entre dois grupos: ricos (nr) e pobres (nmf - nr). Através da "*waggle dance*", serão recrutadas nsr abelhas seguidoras para explorar a vizinhança dos conjuntos de pesos ricos e nsp abelhas seguidoras para explorar a vizinhança dos conjuntos de pesos pobres com o intuito de encontrar novas soluções. Cada conjunto de pesos da vizinhança terá seu EMQ avaliado em relação ao conjunto de pesos principal e se o EMQ resultante for menor do que o do principal, este conjunto de pesos tornar-se-á o novo conjunto principal da região. Essa exploração de vizinhança é conhecida como "*Local Search*".

Já as abelhas campeiras buscarão novos conjuntos de pesos para substituir os nc-nmf conjuntos, pois os mesmos não tiveram um EMQ com um resultado satisfatório.

Essa busca por novos conjuntos de pesos e exploração de vizinhança continuará até que o conjunto de pesos obtido como solução final (melhor conjunto de pesos das fontes ricas) seja satisfatório para o problema em questão através do critério de validação cruzada ou tenha atingido a quantidade máxima de ciclos do algoritmo (nciclos).

Metodologia

Este capítulo retrata o processo de desenvolvimento e aplicação das teorias apresentadas no Capítulo 2, o qual retrata o problema que o projeto corrente se propôs a resolver. Inicialmente, na Seção 3.1, são descritas as características das bases de dados as quais se aplicam as técnicas. Em seguida, a Seção 3.2 discorre sobre o pré-processamento dos dados. Já a Seção 3.3, explica como a RNA foi otimizada com *Artifical Bee Colony* (ABC) ou Algoritmo de Colônia de Abelhas Artificiais.

3.1 Base de dados

As bases de dados utilizadas neste experimento foram: *Íris* [15] e *Wine* [16], bases disponíveis no *UCI Machine Learning Repository*[17], para problema de classificação. Essas bases são amplamente utilizadas no reconhecimento de padrões, ou seja, informações dessas bases são previamente extraídas para classificar outros dados em um dos padrões apreendidos. Já para previsão, a base utilizada foi a série de vazões médias diárias afluentes da Usina Hidrelétrica Curuá-Una, a qual é mantida e operada pela Eletronorte S/A [18].

A Íris é uma das bases mais famosas na literatura para o reconhecimento de padrões [15]. O conjunto de dados contém 3 classes de 50 casos cada, onde cada classe se refere ao atributo tipo de planta: Íris *Setosa*, Íris *Versicolour* ou Íris *Virginica*. Além do atributo citado, há mais 4 atributos: largura da sépala (cm), comprimento da sépala (cm), largura da pétala (cm) e comprimento da pétala (cm). Lembrando que nesta base, uma das classes é linearmente separável (Figura 9) e as outras duas não são linearmente separáveis (Figura 10).

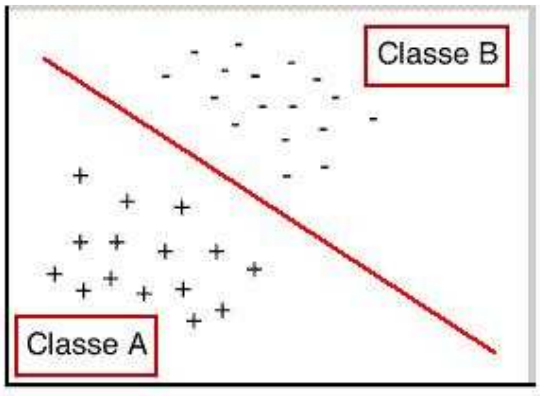

Figura 9. Classes linearmente separáveis.

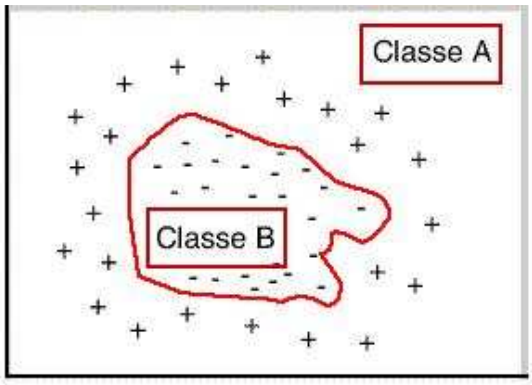

Figura 10. Classes não linearmente separáveis.

A base *Wine* é utilizada para determinar a origem dos vinhos a partir de uma análise química [16]. Essa análise foi feita em 3 cultivos diferentes na Itália, resultando em 13 atributos para distinguir os vinhos de cada cultivo (única classe da base de dados). Esses atributos são: álcool, ácido málico, cinzas, alcalinidade das cinzas, magnésio, fenóis totais, flavanóides, fenóis Nonflavanoid, proanthocyanins, intensidade de cor, matiz OD280/OD315 de vinhos diluídos e proline.

A base de dados da usina hidrelétrica de Curuá-Una possui os registros das vazões médias diárias que vão desde 01 de janeiro de 1978 até 31 de dezembro de 2007. É importante lembrar que essas vazões estão diretamente relacionadas com a geração de energia elétrica pela usina e serão utilizados para predizer vazões até sete dias à frente (sete dias à frente é o padrão utilizado pelo setor elétrico para definição da geração semanal de energia no Brasil).

Essas bases de dados, assim como qualquer outra, geralmente passam pela etapa de pré-processamento para serem estruturas de acordo com a técnica de Inteligência Computacional em questão.

3.2 Pré-processamento

A primeira etapa do pré-processamento é converter a base de dados para o formato padrão do sistema desenvolvido. Para isso, é necessário indicar o separador dos dados, seja esse uma vírgula, ponto e vírgula ou hífen, por exemplo, assim como indicar também a quantidade de saídas dos dados em análise, já que os dados poderão ser classificados de acordo com um conjunto de classes e não apenas de uma única classe.

A próxima etapa se refere aos dados faltantes. Esses dados caracterizados por uma interrogação (?), poderão ser removidos da base ou tratados para fazer parte do processamento. Ao ser tratado, o novo dado será linearmente interpolado.

Na Tabela 2, temos um exemplo de dados faltantes. Esse dado, ao ser tratado, será substituído pela soma de 3,0 + 3,4 divido por 2, logo o novo dado será igual a 3,2. Na ausência do dado posterior ou do dado anterior, iremos atribuir zero na Equação 2.8 para a realização da soma e assim teremos o dado substituto.

Tabela 2. Trecho da base Íris com dados faltantes.

Entradas				Saídas
4,4	2,9	1,4	0,2	Iris-setosa
6,3	3,3	6,0	2,5	Iris-virginica
5,8	2,7	5,1	1,9	Iris-virginica
4,8	3,4	1,6	0,2	Iris-setosa
4,8	3,0	1,4	0,1	Iris-setosa
4,3	?	1,0	1,0	Iris-setosa
5,4	3,4	1,7	0,2	Iris-setosa
7,0	3,2	4,7	1,4	Iris-versicolor
6,4	3,2	4,5	1,5	Iris-versicolor
6,9	3,1	4,9	1,5	Iris-versicolor

A próxima etapa seria o tratamento de dados nominais, caso a base possua dados desse tipo como as saídas da base Íris (Tabela 2), por exemplo. Nesse caso, o atributo de classificação é nominal e terá que ser tratado. O tratamento será através da conversão do dado nominal para dado binário. Tomando a base Íris que

possui dados nominais de 3 tipos (Íris-*Setosa*, Íris-*Virginica* e Íris-*Versicolor*), precisaremos de 3 bits para a conversão, já que a posição do bit “1” identifica o dado como podemos ver na Tabela 3.

Tabela 3. Trecho da base Íris com dados nominais tratados.

Entradas				**Saídas**		
4,4	2,9	1,4	0,2	1	0	0
6,3	3,3	6,0	2,5	0	1	0
5,8	2,7	5,1	1,9	0	1	0
4,8	3,4	1,6	0,2	1	0	0
4,8	3,0	1,4	0,1	1	0	0
4,3	3,2	1,0	1,0	1	0	0
5,4	3,4	1,7	0,2	1	0	0
7,0	3,2	4,7	1,4	0	0	1
6,4	3,2	4,5	1,5	0	0	1
6,9	3,1	4,9	1,5	0	0	1

A última etapa é a de normalização dos dados. É a etapa mais importante do pré-processamento, pois após essa etapa todos os dados receberão o mesmo tratamento na fase de treinamento da RNA. Além disso, os dados devem ter seus valores dentro dos limites da função de ativação usada na camada de saída da RNA e na maioria das vezes os limites são entre 0 e 1. Como a função de ativação utilizada será a sigmóide logística, os dados serão normalizados no intervalo [0,10; 0,90] através da fórmula descrita na Equação 3.2.

$$y = \left[(b-a) * \frac{(x_i - x_{\min})}{(x_{max} - x_{\min})}\right] + a \qquad (3.2)$$

Onde x_i é o dado em questão, $x_{\min}$ é o valor mínimo de todos os dados da mesma classe, assim como x_{max} é o valor máximo, a é o valor mínino do intervalo e b é o valor máximo do intervalo da normalização. Após realizar todo o tratamento necessário e normalizar os dados eles ficam estruturados como padrões para serem utilizados pela RNA como mostra a Tabela 4.

Tabela 4. Trecho da base Íris estruturada na forma padrão.

Entradas				Saídas		
0,144	0,20	0,141	0,167	0,90	0,10	0,10
0,256	0,60	0,141	0,167	0,10	0,90	0,10
0,589	0,367	0,588	0,567	0,10	0,90	0,10
0,411	0,367	0,575	0,50	0,90	0,10	0,10
0,122	0,50	0,141	0,133	0,90	0,10	0,10
0,256	0,60	0,181	0,267	0,90	0,10	0,10
0,278	0,70	0,222	0,20	0,90	0,10	0,10
0,544	0,533	0,778	0,90	0,10	0,10	0,90
0,433	0,333	0,656	0,70	0,10	0,10	0,90
0,722	0,433	0,764	0,767	0,10	0,10	0,90

3.3 Classificação e Predição com RNA utilizando *Artificial Bee Colony*

Após estruturar os dados, o próximo passo será a configuração da RNA para classificar os padrões. Nesse contexto, MLPs ainda revelam certa complexidade em seu uso devido a gama de parâmetros configuráveis, apesar das constantes pesquisas nesse campo [19].

Como o *BackPropagation* foi substituído pelo algoritmo de Colônia de Abelhas Artificiais para o treinamento da RNA, então a configuração da rede se dará em 2 partes: configuração da arquitetura da rede e configuração dos parâmetros do algoritmo das abelhas.

3.3.1 Configuração da arquitetura da rede

Na configuração geral da rede (Figura 14) nas partes destacadas na interface gráfica no sistema desenvolvido neste trabalho, os parâmetros a serem configuráveis são os relacionados à arquitetura da rede, à validação cruzada, ao limiar para taxa de acerto e à base de dados.

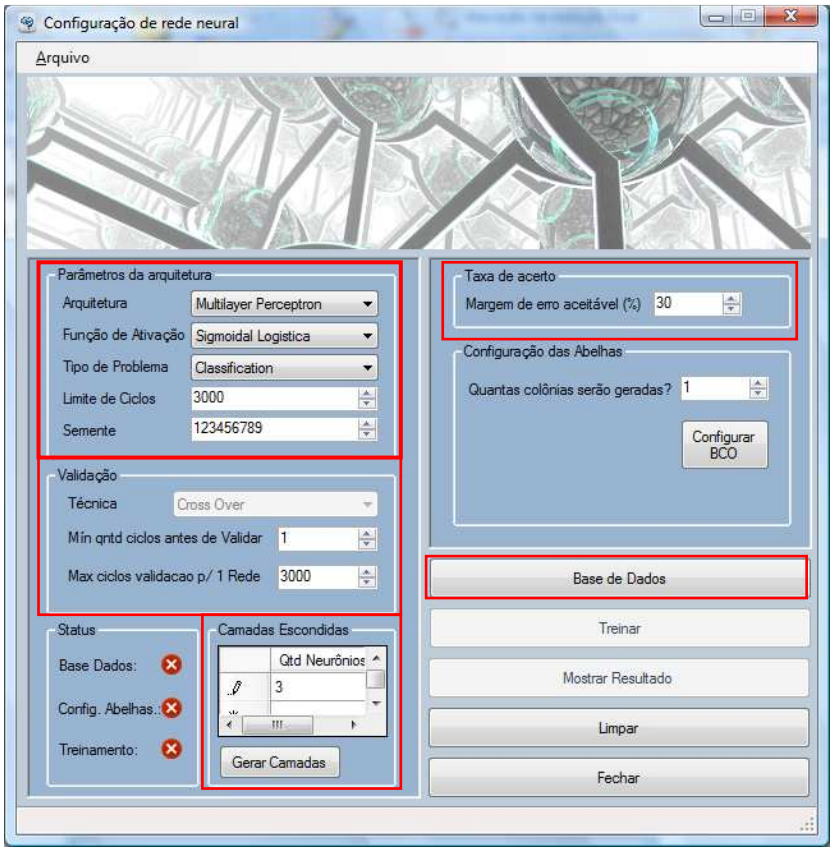

Figura 11. Configuração da Rede Neural

Nos parâmetros sobre arquitetura da rede, os parâmetros abordados serão o tipo de rede, o qual sempre será fixo, pois se trata de uma rede MLP, além da função de ativação, tipo de problema, limite de ciclos e semente.

A função de ativação escolhida foi a Sigmoidal Logística, descrita na Equação 3.3, onde y_i e net_i são, respectivamente, a saída e a média ponderada dos pesos com as entradas do i-ésimo neurônio. Essa função retorna valores dentro do intervalo [0,1].

$$y_i = \frac{1}{1+ e^{-net_i}} \tag{3.3}$$

O tipo de problema a ser resolvido poderá ser tanto de classificação quanto de predição. Dependerá do que é proposto pela base de dados apresentada para ser treinada.

O limite de ciclos de treinamento é um parâmetro variável que dependerá da convergência da rede, ou seja, caso seja verificado no gráfico de treinamento que a rede continuará apreendendo e que a porcentagem de acerto da rede ainda não é

satisfatória esse limite será incrementado como poderá ser decrementado também caso as condições citadas anteriormente tenham sido atendidas antes do limite estabelecido. Após testar várias sementes selecionou-se aquela que apresentava os melhores resultados permanecendo-se com este valor fixo.

Em relação aos parâmetros sobre validação, a técnica utilizada como critério de parada será sempre a de validação cruzada, com 50% do conjunto de padrões para treinamento, 25% para validação e os outros 25% para teste. Essa porcentagem de divisão do conjunto de padrões é configurada no carregamento da base de dados.

O parâmetro sobre mínima quantidade de ciclos antes de validar não é fixo e a rede só começará a validar após igualar a quantidade de ciclos de treinamento corrente a esse parâmetro. A rede tem seu treinamento interrompido de duas maneiras: a primeira é quando o ciclo atual de validação atinge a quantidade máxima de ciclos e não encontrou um erro de validação cruzada melhor do que o atual. A segunda ocorre quando o erro de validação atual for melhor do que o erro do ciclo anterior com uma diferença menor do que 5%.

A arquitetura da MLP pode ser configurável em relação às camadas escondidas, mas como uma única camada escondida é suficiente para alcançarmos os resultados desejados [11], então, o treinamento será realizado com uma única camada escondida com a quantidade de seus neurônios estabelecida por diversos testes. Tanto a quantidade de entradas como a quantidade de saídas da rede vai ser gerado automaticamente, pois são valores que depende do conjunto de padrões utilizado no treinamento. Sendo assim, a arquitetura da rede pode ser vista na Figura 12.

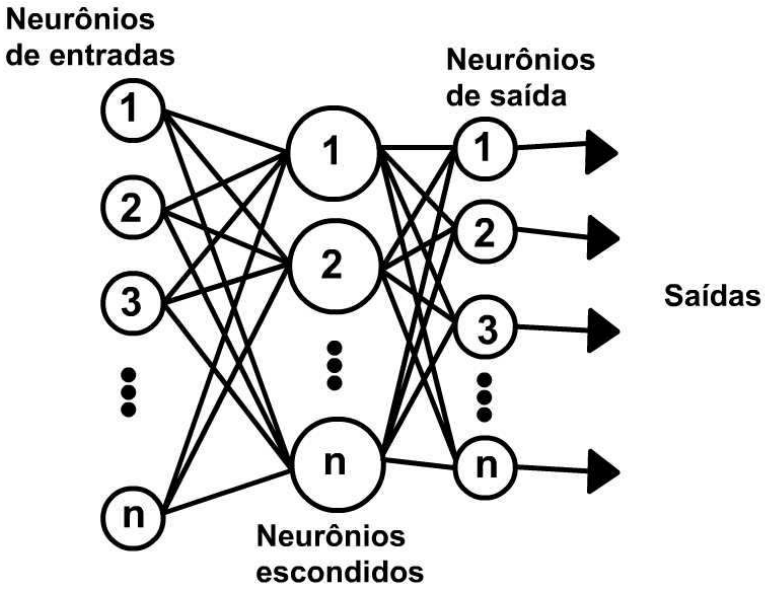

Figura 12. Arquitetura da Rede MLP utilizada na classificação

O parâmetro que estabelece o limiar para a taxa de acerto será de grande influência nos resultados de classificação da rede, principalmente no conjunto de padrões utilizados para teste. Esse parâmetro será mantido fixo com o valor de 30%, sendo assim, um resultado de um exemplo do conjunto de testes será considerado como válido pela rede caso ele esteja dentro da margem para este limiar da taxa de acerto para mais ou para menos em relação ao resultado que se encontra no próprio exemplo.

3.3.2 Configuração dos parâmetros do algoritmo das abelhas

A configuração dos parâmetros do algoritmo das abelhas (Figura 13) é essencial para um bom resultado do treinamento da Rede Neural, pois a otimização dos pesos das conexões da MLP dependerá diretamente dos valores dos parâmetros desse algoritmo, os quais nunca são fixos e podem variar de acordo com a base de dados utilizada na rede.

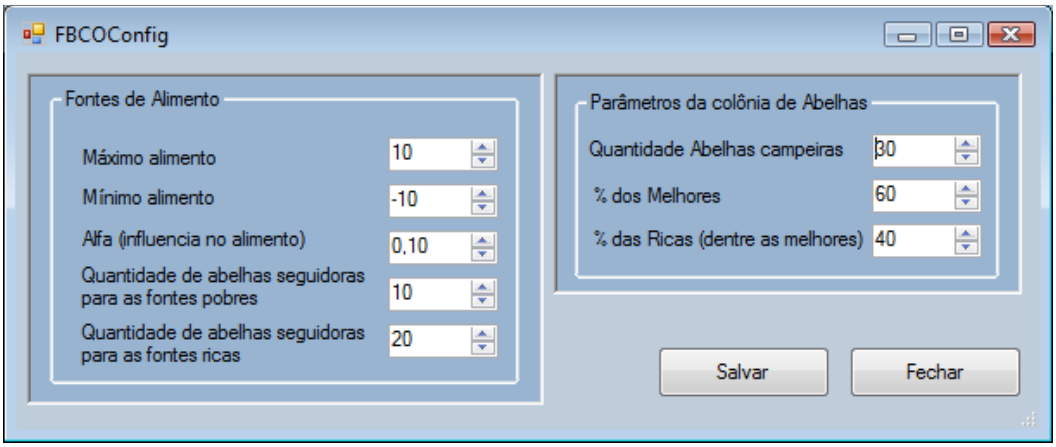

Figura 13. Configuração do Algoritmo das Abelhas

Os parâmetros de configuração vão ser divididos em dois tipos: fontes de alimento e colônia de abelhas. O primeiro grupo representa as características dos conjuntos de pesos que serão exploradas pelas abelhas, já o segundo representa as características das abelhas na colônia.

Contextualizando a configuração do algoritmo com o problema descrito no Capítulo 1 e com o *Artificial Bee Colony* descrito no Capítulo 2, as fontes de alimento que representam o conjunto de alimentos das abelhas será o conjunto de pesos das conexões da rede MLP. Sendo assim, cada peso pertencente ao conjunto será gerado inicialmente, de maneira aleatória.

Para avaliar um conjunto pesos será necessário calcular o erro médio quadrático (EMQ) da MLP como foi mencionado no Capítulo 2. Para calcular o EMQ, os pesos do conjunto serão colocados na arquitetura da MLP e para cada exemplo do conjunto de treinamento, calcula-se, no neurônio de saída, a diferença ao quadrado entre o valor desejado e o calculado.

Após executar essa etapa para cada neurônio de saída, somam-se os valores obtidos e divide pela quantidade de neurônios na camada de saída. Esses resultados obtidos para cada exemplo serão somados ao fim do processamento do ciclo atual de treinamento e divididos pela quantidade de exemplos. Assim, temos como resultado o EMQ do conjunto de pesos.

Os conjuntos de pesos serão gerados de acordo com o parâmetro quantidade de abelhas campeiras, pois cada abelha campeira da colônia estará associada a um conjunto de peso.

De acordo com o EMQ calculado para cada conjunto de pesos, teremos um *rank* desses conjuntos, já que quanto menor for o EMQ melhor será o conjunto de pesos para o nosso problema. A partir desse *rank* então, dividiremos os conjuntos de pesos em melhores e não-melhores de acordo com uma porcentagem previamente definida no parâmetro % das melhores. E dentre as melhores, faremos uma nova divisão entre ricas e pobres, também de acordo com uma porcentagem previamente definida pelo parâmetro % das ricas.

Os conjuntos de pesos classificados como os melhores serão modificados de acordo com a teoria do *waggle dance.* Ou seja, teremos uma quantidade de abelhas seguidoras para explorar tanto os conjuntos de pesos classificados como os classificados como pobres. As quantidades das seguidoras serão configuradas nos parâmetros quantidade de abelhas seguidoras para as fontes elites e quantidade de abelhas seguidoras para as melhores fontes, respectivamente.

Quando as seguidoras forem explorar localmente os conjuntos de pesos, ou seja, a vizinhança desses conjuntos, elas serão associadas a novos conjuntos de pesos através de uma pequena variação do conjunto de pesos associado à abelha operária responsável pelo recrutamento como mostra a Equação 3.5.

$$p_{novo} = p_{antigo} * alfa * \ Random(-1,1) \tag{3.5}$$

Onde p_{novo} é o peso pertencente ao conjunto de pesos da fonte de alimento vizinha, p_{antigo} é o peso pertencente ao conjunto de pesos da fonte de alimento principal, alfa é responsável pela pequena perturbação do peso e é configurado pelo parâmetro Alfa e $Random(-1,1)$ irá definir se a influência do alfa será positiva ou negativa, perturbando o peso para um valor maior ou menor do que o atual.

Se o conjunto de pesos da abelha seguidora tiver um EMQ menor do que o conjunto de pesos da abelha operária, esse conjunto de pesos passa a ser o principal da região explorada. É importante ressaltar que as campeiras que ficaram no grupo das não-melhores, procurarão novamente por novos conjuntos de pesos, ou seja, haverá uma busca global.

Resultados

Este capítulo exibe os resultados a partir da aplicação da metodologia demonstrada no Capítulo 3 nas bases de dados especificadas. Os valores escolhidos para a configuração da arquitetura da rede e para a configuração do algoritmo de abelhas do sistema desenvolvido foram baseados em simulações prévias. Esses valores determinaram a faixa de valores que permitiram a convergência das técnicas utilizadas.

A Seção 4.1 mostra os resultados obtidos aplicando RNAs nas bases de classificação (*Íris e Wine*). Já a seção 4.2 mostra os resultados obtidos com a base de previsão (Curuá-Una).

4.1 Classificação

Nesta seção apresentaremos os resultados obtidos nas bases de classificação. O parâmetro de avaliação das técnicas aplicadas para esse tipo de base será a taxa de acerto obtida pela rede no conjunto de testes perante a configuração da rede, já que é nesse conjunto que a rede produzirá sua própria resposta para um padrão de exemplos desconhecido. Sendo assim, a taxa de acerto indicará o quanto a rede aprendeu e quão boa é a técnica empregada no ajuste dos pesos das conexões da RNA. É importante lembrar que a porcentagem da margem de erro aceitável que tem influência direta na taxa de acerto permaneceu fixa com um valor igual a 30% para todas as simulações.

4.1.1 Base *Íris*

Na base *Íris,* foram realizadas 7 simulações tanto para o *BackPropagation* quanto para o ABC. Alguns parâmetros da configuração da arquitetura da rede como quantidade de neurônios na camada escondida, limite de ciclos (tanto de processamento quanto de validação, pois os dois possuem os mesmos valores) e semente foram alterados da mesma forma para as duas técnicas como mostra a tabela 5.

Tabela 5. Parâmetros da arquitetura da rede para a base Íris.

Simulação	1	2	3	4	5	6	7
Neurônios Escondidos	5	4	6	5	5	5	5
Limite de ciclos	500	400	500	300	100	100	100
Semente	123456789	123456789	123456789	123456789	123456789	2424	1313

Em relação ao *BackPropagation,* os parâmetros alterados foram: a *taxa de aprendizado* (α) e o *momentum* (β) como podem ser visto na Tabela 6.

Tabela 6. Parâmetros do *BackPropagation* para a base *Íris.*

Simulação	1	2	3	4	5	6	7
Taxa de aprendizado (α)	0,01	0,02	0,001	0,4	0,3	0,5	0,8
***Momentum* (β)**	0,02	0,04	0,002	0,8	0,6	0,10	0,16

O melhor resultado obtido pelo *BackPropagation* foi a de simulação número 7, a qual apresentou uma taxa de acerto igual a 94,495% como mostra a tabela 7. Ao comparar com as demais simulações, foi o que teve uma variação na semente utilizada e os maiores valores para alfa e beta, o que foi necessário para a convergência da técnica devido ao baixo valor do limite de ciclos. A Figura 14 mostra a tela de resultado obtida pelo sistema desenvolvido com a melhor configuração do *BackPropagation* .

Tabela 7. Resultados do *BackPropagation* para a base *Íris.*

Simulação	1	2	3	4	5	6	7
Ciclo da melhor rede	500	400	180	301	301	101	101
Tempo de processamento (hora:min:seg)	00:00:17	00:00:07	00:00:15	00:00:08	00:00:07	00:00:02	00:00:03
Taxa de acerto (%)	24,324	40,400	85,000	91,892	86,486	89,189	94,495

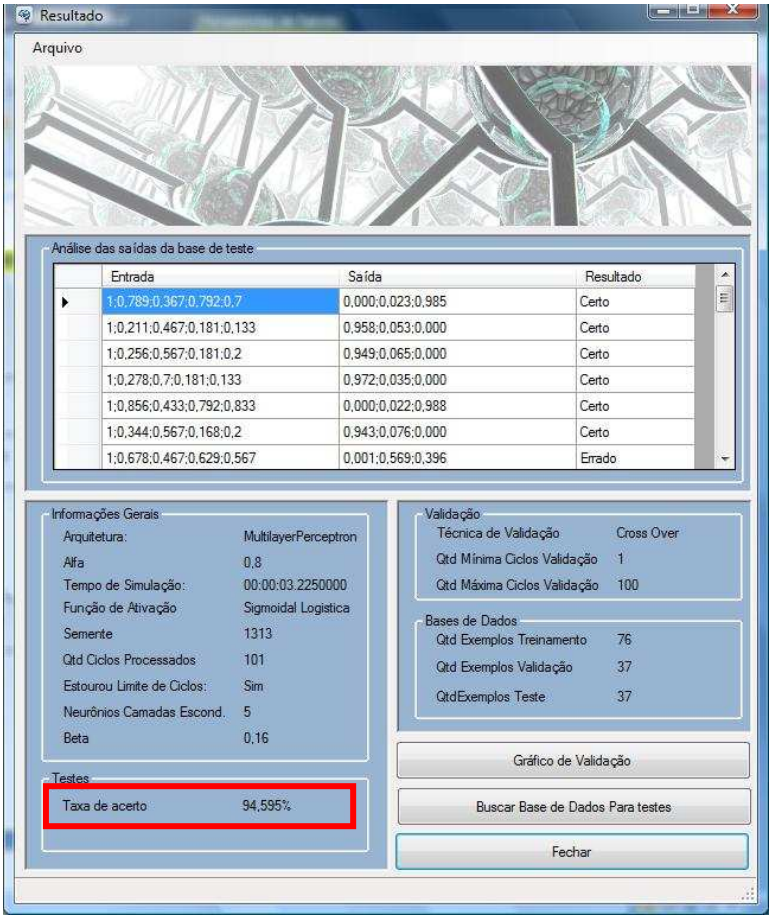

Figura 14. Melhor Resultado do *BackPropagation* para a base *Íris.*

Em relação à configuração do algoritmo das abelhas, todos os parâmetros foram alterados (Tabela 8) para que encontrássemos a configuração que obtivesse o resultado com a melhor taxa de acerto. Então, a partir da primeira simulação com uma taxa de acerto de 97,287% (Tabela 9), foram feitas pequenas alterações

manualmente nos parâmetros para testar a influência dos mesmos na taxa de acerto.

Tabela 8. Parâmetros do ABC para a base *Íris.*

Simulação	1	2	3	4	5	6	7
Máxima qualidade de alimento	20	15	20	25	30	30	10
Mínima qualidade de alimento	-20	-15	-20	-25	-30	-30	-10
Alfa (influência no alimento)	0,08	0,07	0,08	0,10	0,08	0,10	0,10
Quantidade de abelhas recrutadas para as melhores fontes	5	5	5	5	5	5	10
Quantidade de abelhas recrutadas para as fontes elites	10	8	10	10	10	10	20
Tamanho da colônia	40	30	40	40	40	40	30
%das melhores	20	50	20	20	20	20	60
%das Elites (dentre as melhores)	40	40	40	40	20	20	40

Sendo assim, os parâmetros que mais influenciaram na taxa de acerto ao comparar a Tabela 8 com a 9 foram os que estavam diretamente ligados aos ajustes dos pesos de conexões da RNA como Máxima e Mínima qualidade de alimento e Alfa (influência no alimento). E dentre os resultados obtidos (Tabela 9), a melhor simulação foi a de número 5 com uma taxa de acerto de 100,00%. A Figura 15 mostra a tela de resultado obtida pelo sistema desenvolvido com a melhor configuração do ABC.

Tabela 9. Resultados do ABC para a base *Íris.*

Simulação	1	2	3	4	5	6	7
Ciclo da melhor rede	10	210	160	10	22	45	45
Tempo de processamento (hora:min:seg)	0:22:57	0:13:32	0:32:43	0:14:59	0:4:42	0:5:30	0:13:58
Taxa de acerto (%)	97,297	35,05	89,189	89,198	100,000	81,081	82,015

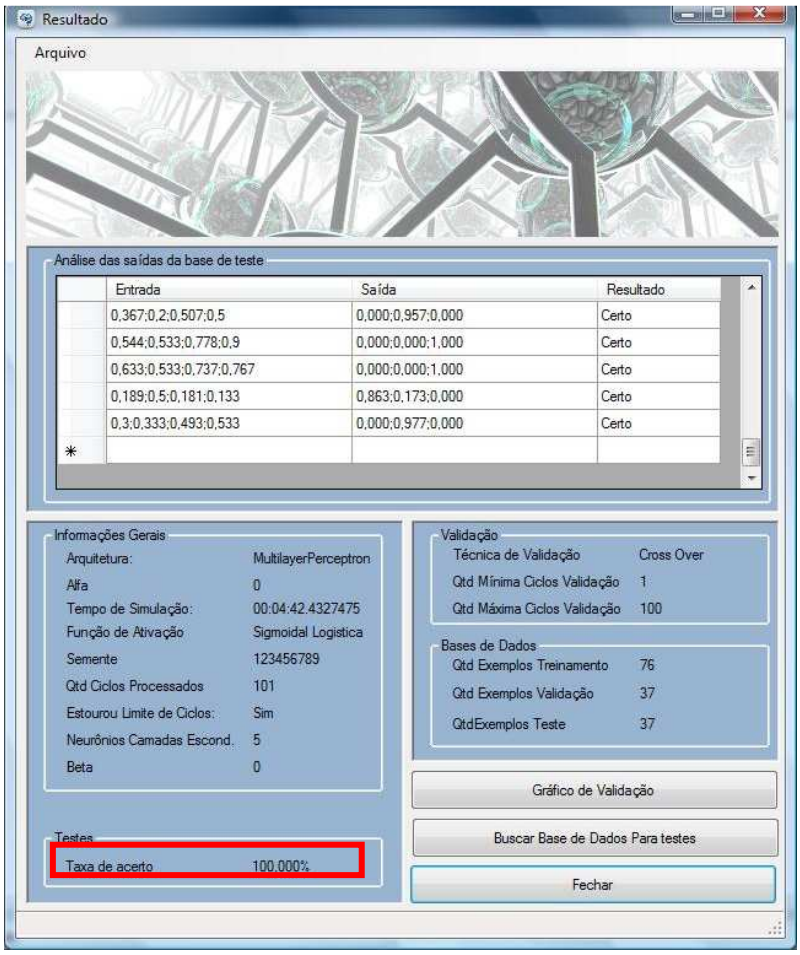

Figura 15. Melhor Resultado do ABC para a base *Íris*

Na Figura 16 tem-se um gráfico comparativo entre as taxas de acerto obtidas nas simulações tanto para o *Backpropagation* quanto para o ABC, e pode-se perceber que independente do melhor resultado obtido por cada técnica utilizada ao final de todas as simulações, em algumas simulações uma técnica foi melhor do que

a outra e vice-versa. Isso ocorreu devido às tentativas realizadas para encontrar os melhores parâmetros para a convergência das técnicas.

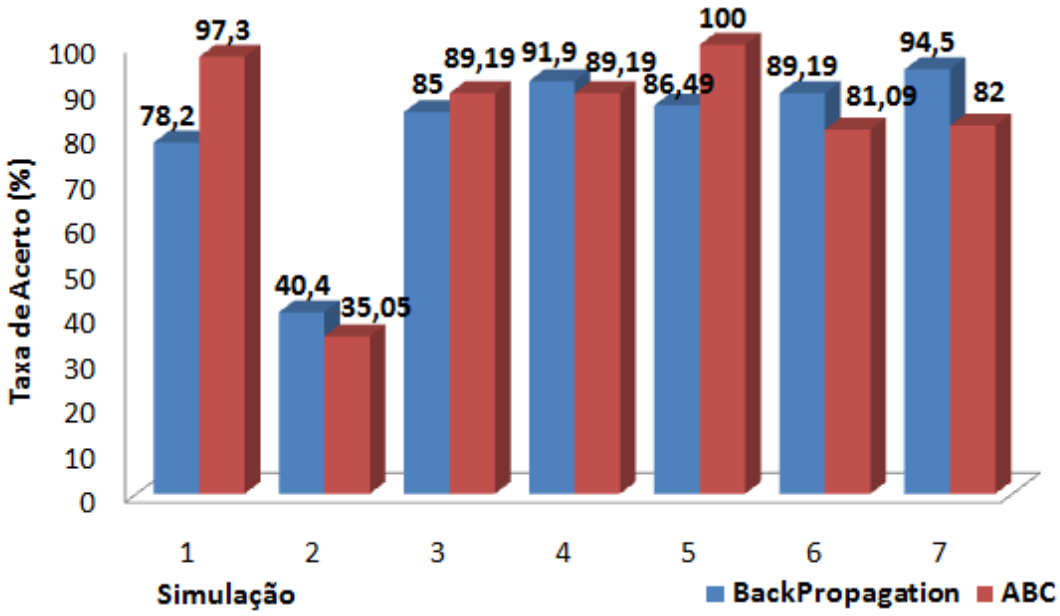

Figura 16. Gráfico comparativo entre *Backpropagation* e ABC para a base *Íris.*

4.1.2 Base Wine

Na base *Wine,* foram realizadas 8 simulações tanto para o *BackPropagation* quanto para o ABC. Alguns parâmetros da configuração da arquitetura da rede como quantidade de neurônios na camada escondida e limite de ciclos (tanto de processamento quanto de validação, pois os dois possuem os mesmos valores) foram alterados da mesma maneira para as duas técnicas (Tabela 10). Vale ressaltar que a variação desses parâmetros para essa base foi bastante sutil.

Tabela 10. Parâmetros da arquitetura da rede para a base *Wine.*

Simulação	1	2	3	4	5	6	7	8
Neurônios Escondidos	5	4	5	4	4	5	5	5
Limite de ciclos	200	100	100	100	100	100	100	100

Em relação ao *BackPropagation* , os parâmetros alterados foram: a *taxa de aprendizado* (α) e o *momentum* (β) (Tabela 11).

Tabela 11. Parâmetros do *BackPropagation* para a base *Wine.*

Simulação	1	2	3	4	5	6	7	8
Taxa de aprendizado (α)	0,01	0,02	0,001	0,3	0,4	0,004	0,8	0,16
***Momentum* (β)**	0,02	0,04	0,002	0,6	0,8	0,008	0,16	0,32

O melhor resultado obtido pelo *BackPropagation* foi o de simulação número 7, o qual apresentou uma taxa de acerto igual a 88,636% (Tabela 12) e ao comparar com as demais simulações, foi o que alcançou a convergência da técnica com valores altos para alfa e beta devido ao baixo valor do limite de ciclos. A Figura 17 mostra a tela de resultado obtida pelo sistema desenvolvido com a melhor configuração do *BackPropagation* .

Tabela 12. Resultados do *BackPropagation* para a base *Wine.*

Simulação	1	2	3	4	5	6	7	8
Ciclo da melhor rede	10	101	101	55	101	101	19	75
Tempo de processamento (hora:min:seg)	0:00:12	0:00:04	0:00:05	0:00:04	0:00:04	0:00:06	0:00:05	0:00:05
Taxa de acerto (%)	34,091	56,818	61,364	84,091	81,818	38,636	88,636	86,364

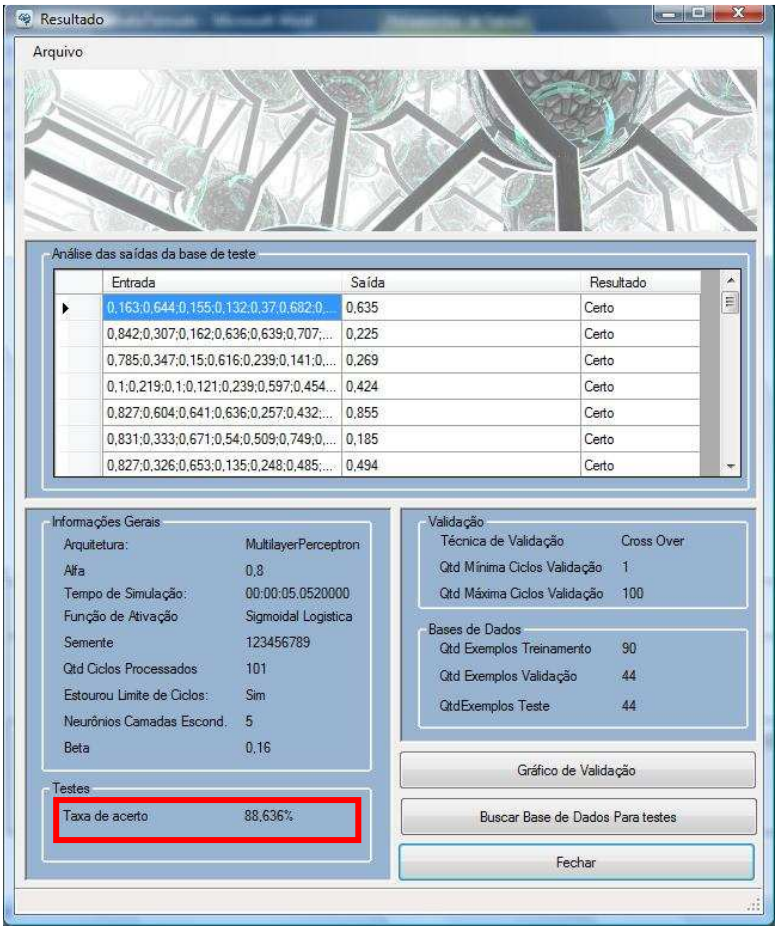

Figura 17. Melhor Resultado do *BackPropagation* para a base *Wine.*

Em relação à configuração do algoritmo das abelhas, os parâmetros alterados foram os que influenciam diretamente na taxa de acerto da RNA, os quais foram constatados na simulação da base *Íris*. São eles: Máxima e Mínima qualidade de alimento e Alfa (influência no alimento). Os outros parâmetros foram mantidos fixos com os seguintes valores: Quantidade de abelhas recrutadas para as melhores fontes (5), Quantidade de abelhas recrutadas para as fontes elites (10), Tamanho da colônia (40), %das melhores (20) e %das Elites (dentre as melhores) (40). Então, a partir da primeira simulação com uma taxa de acerto de 75,000% como mostra a Tabela 14, foram feitas alterações nos 3 parâmetros (Tabela 13) citados anteriormente para testar a influência dos mesmos na taxa de acerto.

Tabela 13. Parâmetros do ABC para a base *Wine.*

Simulação	1	2	3	4	5	6	7	8
Máxima qualidade de alimento	30	20	25	35	40	50	45	50
Mínima qualidade de alimento	-30	-20	-25	-35	-40	-50	-45	-50
Alfa (influência no alimento)	0,05	0,08	0,08	0,05	0,15	0,05	0,05	0,15

Dentre os resultados obtidos na Tabela 14, a melhor simulação foi a de número 8 com uma taxa de acerto de 97,727%, um bom resultado para uma técnica nunca antes utilizada para o treinamento de uma MLP. A Figura 18 mostra a tela de resultado obtida pelo sistema desenvolvido com a melhor configuração do ABC.

Tabela 14. Resultados do ABC para a base *Wine.*

Simulação	1	2	3	4	5	6	7	8
Ciclo da melhor rede	65	37	8	43	92	95	98	61
Tempo de processamento (hora:min:seg)	0:47:34	0:23:52	0:26:29	0:14:36	0:15:59	0:23:36	0:36:55	0:37:18
Taxa de acerto (%)	75,000	59,091	43,182	68,182	63,636	84,091	61,364	97,727

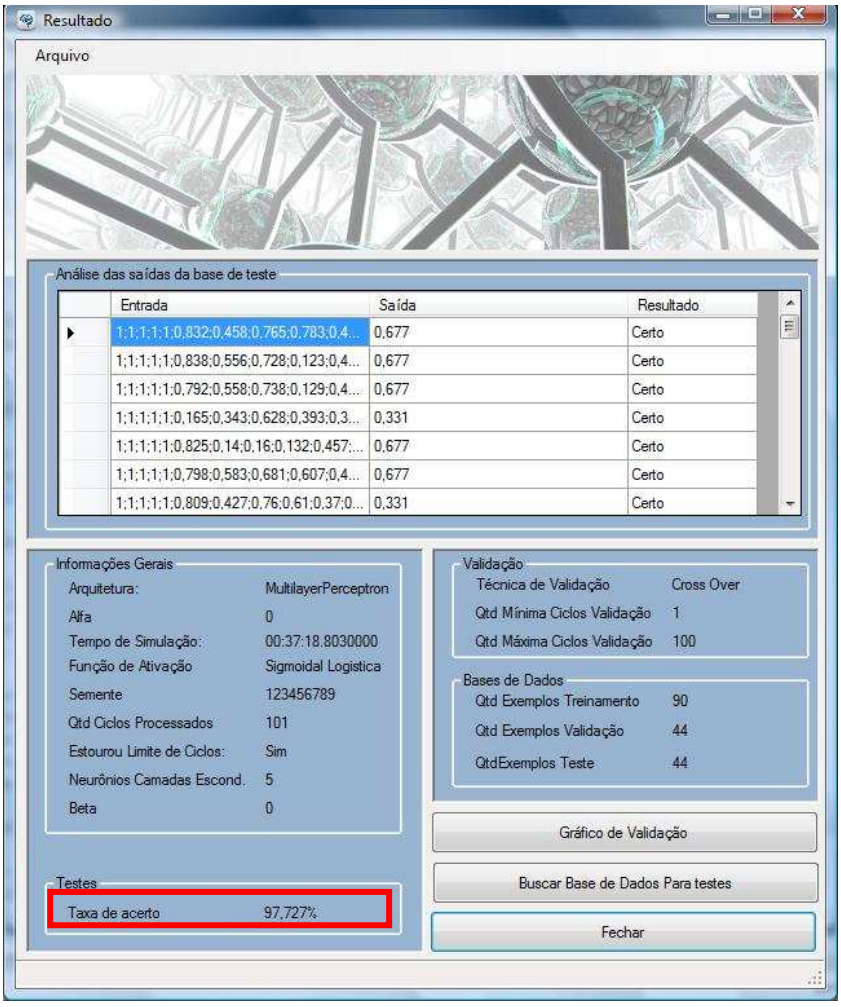

Figura 18. Melhor Resultado do ABC para a base *Wine.*

Na Figura 19 tem-se um gráfico comparativo entre as taxas de acerto obtidas nas simulações tanto para o *Backpropagation* quanto para o ABC, e pode-se perceber que independente do melhor resultado obtido por cada técnica utilizada ao final de todas as simulações, em algumas simulações uma técnica foi melhor do que a outra e vice-versa. Isso ocorreu devido às tentativas realizadas para encontrar os melhores parâmetros para a convergência das técnicas.

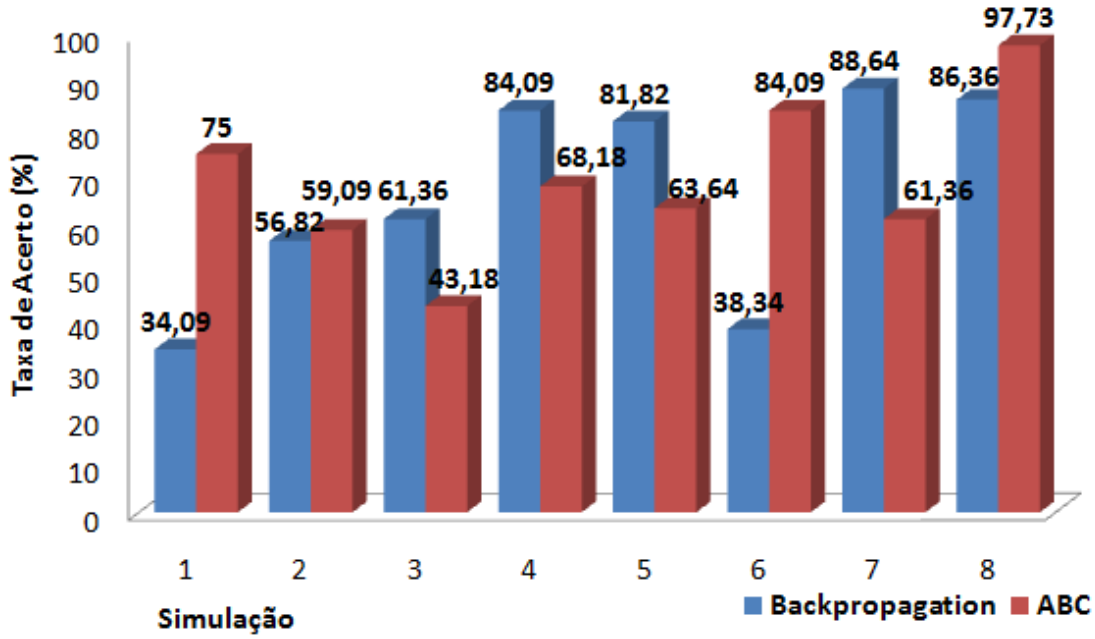

Figura 19. Gráfico comparativo entre *Backpropagation* e ABC para a base *Wine.*

4.2 Predição

Nesta seção apresentaremos os resultados obtidos na base de predição. O parâmetro de avaliação das técnicas aplicadas para esse tipo de base será o Erro Probabilístico Médio Absoluto (EMPA), dado pela equação 4.1, onde N = número de previsões realizadas, N_{out} = números de neurônio na camada de saída da rede, d_i = saída desejada e y_i = saída prevista para a i-ésima predição. Quanto menor for o EMPA, melhor será a predição realizada pela técnica empregada no ajuste dos pesos das conexões da RNA.

$$EMPA = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N_{out}} abs(d_i - y_i) \quad (4.1)$$

4.2.1 Base Curuá-Una

Na base Curuá-Una, foram realizadas 2 simulações tanto para o *BackPropagation* quanto para o ABC. Alguns parâmetros da configuração da arquitetura da rede permaneceram constantes para as duas técnicas com os seguintes valores: 14 neurônios na camada escondida, 100 ciclos (tanto para processamento quanto para validação) e semente igual a 123456789.

Em relação ao *BackPropagation* , os parâmetros alterados foram: a *taxa de aprendizado* (α) e o *momentum* (β) (Tabela 15).

Tabela 15. Parâmetros do *BackPropagation* para a base Curuá-Una

Simulação	1	2
Taxa de aprendizado (α)	0,01	0,1
***Momentum* (β)**	0,01	0,2

O melhor resultado obtido pelo *BackPropagation* foi o de simulação número 1, o qual apresentou o EMPA mais baixo e com valor igual a 21,6% (Tabela 16), além disso, o tempo de convergência entre as duas simulações foram bem próximos. A Figura 20 mostra a tela de resultado obtida pelo sistema desenvolvido com a melhor configuração do *BackPropagation* .

Tabela 16. Resultados do *BackPropagation* para a base Curuá-Una

Simulação	1	2
Ciclo da melhor rede	101	101
Tempo de processamento (hora:min:seg)	0;37:18	0:36:14
EMPA (%)	21,618	23,725

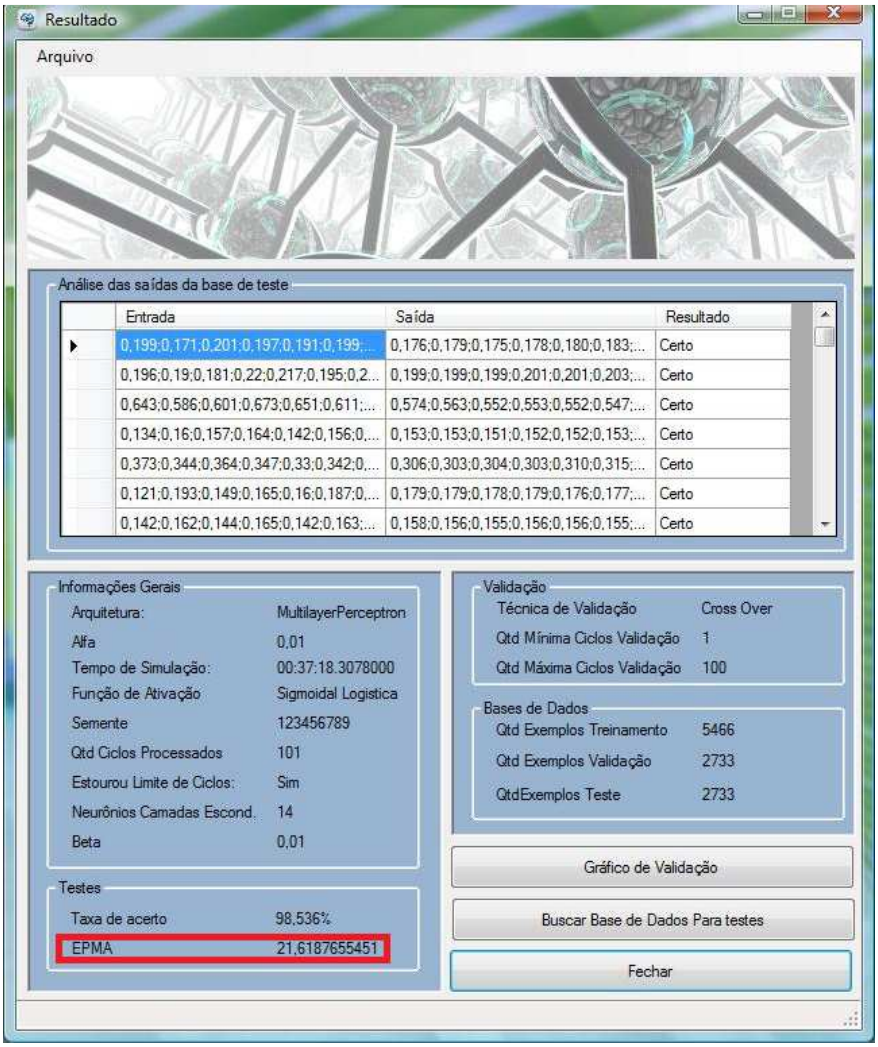

Figura 20. Melhor Resultado do *BackPropagation* para a base Curuá-Una

Em relação à configuração do algoritmo das abelhas, os parâmetros alterados foram os que influenciam diretamente no valor do EPMA (Tabela 17). São eles: Máxima e Mínima qualidade de alimento e Alfa (influência no alimento). Os outros parâmetros foram mantidos fixos com os seguintes valores: Quantidade de abelhas recrutadas para as melhores fontes (1), Quantidade de abelhas recrutadas para as fontes elites (2), Tamanho da colônia (10), %das melhores (50) e %das Elites (dentre as melhores) (50).

Tabela 17. Parâmetros do ABC para a base Curuá-Una

Simulação	1	2
Máxima qualidade de alimento	15	10
Mínima qualidade de alimento	-15	-10
Alfa (influência no alimento)	0,05	0,08

Dentre os resultados mostrados na Tabela 18, a melhor simulação do ABC foi a de número 2 com o EMPA igual de 93,8%, um resultado inferior ao obtido pelo *BackPropagation,* mas que pode ser melhorado através de modificações nos parâmetros do ABC. A Figura 21 mostra a tela de resultado obtida pelo sistema desenvolvido com a melhor configuração do ABC.

Tabela 18. Resultados do ABC para a base Curuá-Una

Simulação	1	2
Ciclo da melhor rede	90	70
Tempo de processamento (dias:hora:min:seg)	3:19:09:06	4:13:40:02
EMPA (%)	99,654	93,829

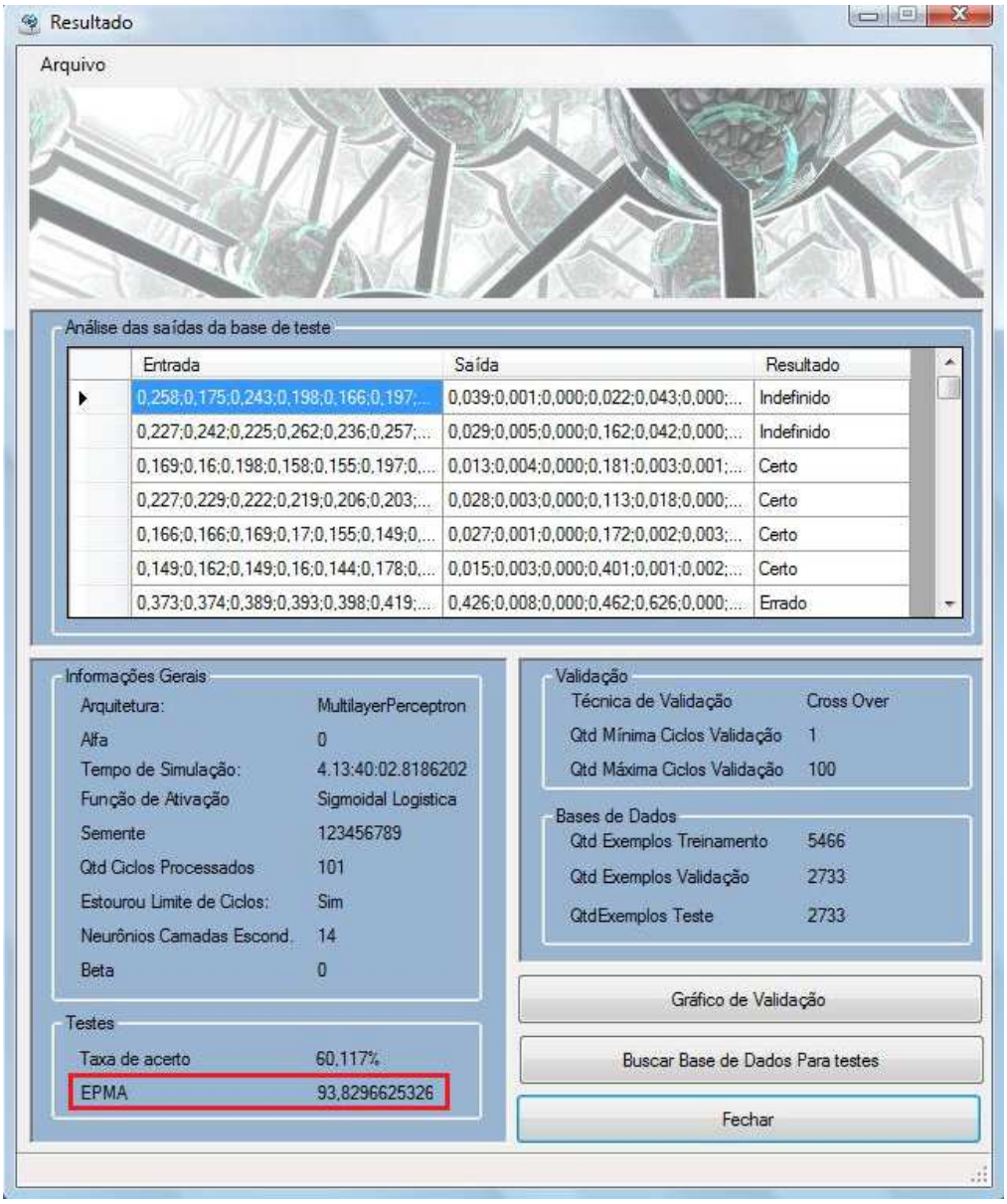

Figura 21. Melhor Resultado do ABC para a base Curuá-Una

Na Figura 22 tem-se um gráfico comparativo entre os EPMAs obtidos nas simulações tanto para o *Backpropagation* quanto para o ABC para predição, e pode-se perceber que o *Backpropagation* obteve os melhores resultados, pois os parâmetros de configuração desta técnica foram obtidos através da experiência do orientador deste trabalho, o professor Dr. Mêuser. Além disso, predição foi implementado como um bônus para este trabalho, e assim sendo, os resultados apresentados pelo ABC ainda são preliminares.

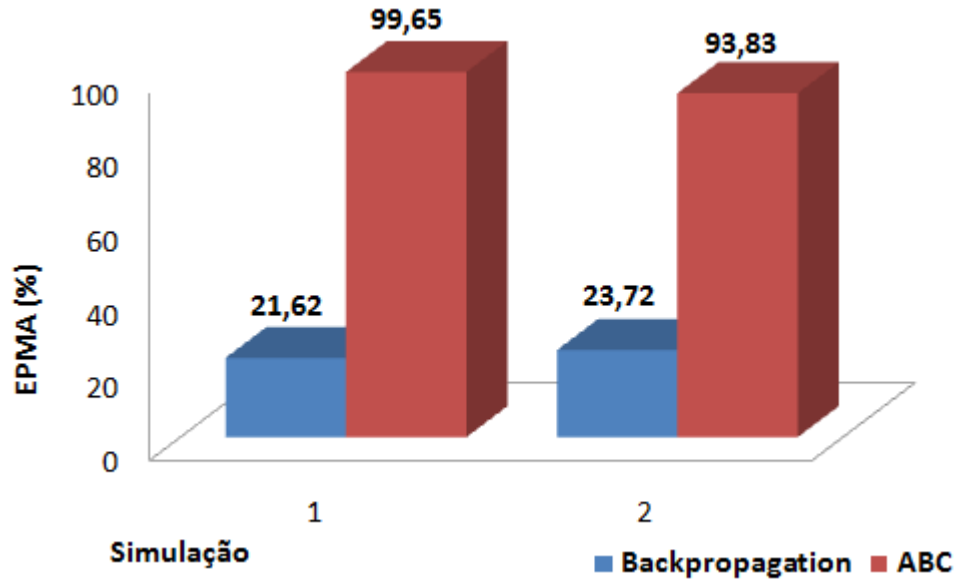

Figura 22. Gráfico comparativo entre *Backpropagation* e ABC para *Curuá-Una.*

Conclusão

Este trabalho descreveu uma proposta de uma metodologia para a substituição do *BackPropagation* pelo algoritmo das abelhas no ajuste dos pesos de conexões de uma RNA. Para validar a nova metodologia de forma satisfatória foram usadas bases de dados tanto para problemas de classificação quanto para problemas de predição.

Com base nas discussões anteriores, o método proposto obteve resultados promissores nas bases de classificação, onde o bom desempenho é evidente tanto no problema da *Íris* quanto no do *Wine*, cujos resultados apontam uma nítida superioridade na taxa de acerto. Entretanto, o tempo de processamento para o treinamento da RNA foi considerável diante do tempo mensurado pelo método tradicional devido ao uso de um espaço de busca configurável ao invés de derivadas de funções objetivas.

Em relação à predição, os resultados obtidos não foram tão promissores na base Curuá-Una devido às poucas simulações realizadas, já que a implementação de predição foi realizada por último e além do mais, é uma simulação que requer um pouco mais de tempo de processamento por causa do tamanho da base de dados.

Uma dificuldade que vale a pena registrar foi a de implementação do próprio ABC, pois, durante as pesquisas nos poucos artigos desse algoritmo de computação natural, tivemos vários pseudocódigos e, por se tratar de um algoritmo recente inclusive na aplicação com RNA, foi necessário implementar a maioria deles até encontrarmos a solução que mais se adequava à proposta do trabalho. Além disso, foram necessárias algumas simulações para se ter uma ideia dos valores dos parâmetros do ABC para a rede convergir. Por outro lado, do ponto de vista de aprendizado da técnica, foi bastante proveitoso.

Como trabalhos futuros, pode-se automatizar os parâmetros do ABC assim como a arquitetura da rede em relação à quantidade de neurônios escondidos. Outra possibilidade é investigar o trabalho com problemas multi-objetivos (EMQ, EMPA), além de aplicar várias funções em predição, regressão e classificação. Em relação à predição (Curuá-Una) tem-se que realizar mais simulações para testar outras

configurações do ABC e investigar possíveis modificações no algoritmo desenvolvido visando minimizar o tempo de processamento do aplicativo durante o treinamento da MLP.

Bibliografia

[1] Dayhoff J. Neural-Network Architectures: An Introduction. New York: Van Nostrand Rein- hold; 1990.

[2] Mehrotra K.; Mohan C.; Ranka S. Elements of Artificial Neural Networks. Cambridge, MA: MIT Press; 1997.

[3] S. Haykin. Neural Networks: A comprehensive Foundation. Prentice Hall, 2 edition, 1998.

[4] D. Marquardt. An algorithm for least squares estimation of non-linear parameters. J. Soc. Ind. Appl. Math., pages 431–441, 1963.

[5] Carvalho, Marcio Ribeiro de. Uma Análise da Otimização de Redes Neurais MLP por Enxames. Dissertação de Mestrado, Universidade Federal de Pernambuco, Recife, PE, 2007.

[6] Pham , DT; Castellani, M. The Bees Algorithm: modelling foraging behaviour to solve continuous optimization problems. Jmes 50th Anniversary Issue Paper, 2009.

[7] BRAGA, A. de P.; CARVALHO, A. P. de Leon Filho de; LUDERMIR, T. B. Redes Neurais Artificiais: Teoria e Aplicações. Rio de Janeiro: LTC, 2000

[8] Machado, Angelo. Neuroanatomia Funcional. Rio de Janeiro: Atheneu, 1988.

[9] MCCULLOCH, W. S.; PITTS, W. A logical calulus of the ideas immanent in nervous activity. Bulletin os Mathematical Biophisics, p. 115_133, 1943.

[10] HAYKIN, S. Redes Neurais: Princípios e Práticas. Recife: Bookman, 2007.

[11] VALENÇA, M. J. S. Fundamentos das Redes Neurais: Exemplos em Java. Recife: Livro Rápido, 2007.

[12] Karaboga, D. An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, 2005.

[13] Basturk, B; Karaboga, D. An artificial bee colony (abc) algorithm for numeric function optimization. IEEE, Swarm Intelligence Symposium 2006, Indiana, USA, 2006.

[14] Pham, D.T; Koç, E.; Ghanbarzadeh, A.; Otri, S. Optimization of the Weights of Multi-Layer Perceptrons Using the Bees Algorithm. Proceedings of 5th International Symposium on Intelligent Manufacturing Systems, p. 38-46, 2006.

[15] Site da base Íris. Disponível em: <http://archive.ics.uci.edu/ml/datasets/Iris>. Último acesso em: 05 de maio de 2010.

[16] Site da base *Wine.* Disponível em: <http://archive.ics.uci.edu/ml/datasets/Wine> Último acesso em: 05 de maio de 2010.

[17] Site que contém as bases de dados utilizadas no projeto. *UCI Machine Learning Repository.* Disponível em: <http://archive.ics.uci.edu/ml/>. Último acesso em: 20 de abril de 2010.

[18] Base de dados sobre medições diárias da Usina Hidrelétrica Curuá-Una.Operados Nacional de Sistema Elétrico. Disponível em: <http://www.ons.org.br>. Último acesso em: 20 de maio de 2010.

[19] Lucas, Tarcísio Daniel Pontes. Utilizando Informações de Tendência *Fuzzy* para Previsão de Vazões com Redes Neurais. Trabalho de Conclusão de Curso, Universidade de Pernambuco, Recife, PE, 2009.