Implementação de meta-heurísticas para o problema da otimização de arquitetura de redes neurais artificiais do tipo feedforward.

Adenilson R. Carvalho[1], **Fernando M. Ramos**[1], **Antonio A. Chaves**[1]

[1]Laboratório Associado de Computação e Matemática Aplicada – LAC
Instituto Nacional de Pesquisas Espaciais (INPE) – São José dos Campos – SP – Brazil

{adenilson,fernando,chaves}@lac.inpe.br

Resumo. *Este artigo lida com o conceito de rede neural artificial (RNA) evolutiva e tem como objetivo propor uma maneira sistemática e automatizada de estabelecer uma arquitetura de RNA. Para tanto adaptamos 3 metaheurísticas para a resolução do problema representado pela busca da arquitetura ótima de uma RNA do tipo feedForward e introduzimos um novo critério para quantificar a performance de uma RNA baseado na combinação de erro de treinamento e generalização. Adicionalmente, é proposto uma nova metodologia para estimação da complexidade computacional da arquitetura de uma RNA baseada no número de neurônios e no número de épocas necessários para o treinamento dos padrões apresentados à rede. Esta abordagem é implementada em software e testada para o problema da identificação e estimação de Fontes poluentes ([Roberti 2005] e [Paes 2008]). Os resultados mostram que a abordagem computacional proposta apresenta melhor performance em relação a um especialista humano, oferecendo ao mesmo tempo muitas vantagens em relação a abordagens similares encontradas na literatura.*

1. Introdução

Redes neurais artificiais (RNAs) têm sido um tópico de interesse nos institutos de pesquisa no mundo todo. Embora muito tenha sido proposto e estudado a respeito ainda existem certas questões referentes ao modelo de RNAs que necessitam ser melhor compreendidas tais como a ausência de garantia prévia de que o modelo adotado será bem sucedido na execução da tarefa para qual ele foi projetado. Muito se tem estudado e investigado na tentativa de encontrarmos formas sistemáticas de ajustar um modelo de RNA a um problema em particular de maneira a otimizarmos a sua performance global. Essas pesquisas se baseiam em quase todos os aspectos presentes no processo de modelagem de RNA tais como os diferentes tipos de função de ativação, a inicialização dos pesos, a coleção de dados de treinamento, o pré e o pós processamento, os algoritmos de treinamento e as funções de erro. No entanto, baseado em evidências empíricas e teóricas encontradas na literatura, destacamos a busca e a definição de uma arquitetura ótima de RNA como sendo o aspecto mais relevante em termos de desempenho e eficiência computacional do modelo [Bernardos 2004]. Por arquitetura de Rede Neural do tipo feedforward subtende-se um conjunto de elementos constituintes dentre os quais destaca-se o número de camadas, o número de unidades de processamento (neurônios) em cada camada, a função de ativação de cada camada e o algoritmo de treinamento (porque este determina o valor final de pesos e bias).

Além desses elementos existem outros dois que são também importantes na definição de uma arquitetura de RNA "extendida", por assim dizer, e que correspondem à taxa de aprendizagem (α) e à constante de momentum (β). A taxa de aprendizagem é um termo empregado na regra de aprendizagem de uma RNA para atualização dos pesos das conexões. O seu valor fica restrito ao intervalo $[0, 1]$ e determina a quantidade que um peso de uma conexão muda em resposta a um sinal de erro. Se a taxa de aprendizagem selecionada é muito grande, então o mínimo local pode ser ultrapassado constantemente, resultando em oscilações e convergência lenta para um estado de menor erro. Se a taxa de aprendizagem é muito baixa, o número de iterações requeridas pode ser muito grande, resultando em baixa performance [rojas 1996]. Já com a introdução do termo de momentum, uma pequena proporção dos pesos do ciclo anterior é adicionada aos pesos correntes durante a etapa de ajuste de pesos. Tem-se observado que esse método diminui efetivamente o tempo de treinamento da rede e que quanto maior o coeficiente de momentum, mais rápido é a convergência [RUMELHART 1986]).

O problema correspondente a encontrar uma arquitetura de rede quase-ótima pode então ser formulado como uma busca no espaço de arquiteturas onde cada ponto representa uma arquitetura possível. Se associarmos a cada ponto (arquitetura) um nível de performance baseado em algum critério de optimalidade (por exemplo complexidade, erro quadrático mínimo, velocidade de aprendizagem,etc) podemos construir e visualizar uma superfície no espaço tridimensional. Encontrar uma arquitetura ótima consistiria nesse caso em descobrir o ponto mais alto dessa superfície ou seja, o seu ponto de máximo [Yao 1997].

2. Computação evolutiva

Não existe uma indicação clara e inequívoca de como encontrarmos a melhor arquitetura dentre as inúmeras opções que se apresentam. Além disso nem sequer conhecemos de antemão a correta topologia de rede a ser aplicada. Na prática este tipo de problema geralmente é solucionado (parcialmente) mediante o uso de métodos empíricos baseadas em processos repetitivos de tentativa-e-erro. Pode-se dizer que o sucesso desse método depende quase que única e exclusivamente da experiência pregressa e da intuição do especialista humano por trás da cena, envolvendo, portanto, um alto grau de incerteza. Nesses métodos muito tempo de máquina é despedido em simulações desnecessárias e que não abrangem a totalidade das opções de arquitetura disponíveis para um determinado problema. Além disso, quase sempre é encontrada uma solução sub-ótima muito aquém da solução que poderia ser obtida mediante o uso de um método autônomo de busca.

Trata-se, portanto, de uma tarefa árdua projetar e construir uma RNA para um problema específico. Existem alguns poucos princípios rigorosos de projeto à disposição para definição da RNA e, no entanto, uma grande quantidade de parâmetros a serem definidos. Os métodos construtivos são uma alternativa para o projeto de redes neurais artificiais, mas o custo computacional requerido pode ser proibitivo para certas aplicações, além da dificuldade apresentada no tratamento de espaços de entrada de elevada dimensão. Portanto, o objetivo aqui é considerar técnicas avançadas de computação evolutiva, as quais também promovem a obtenção de arquiteturas de redes neurais dedicadas, ou seja, específicas para cada problema.

Podemos dividir as propostas de combinação de computação evolutiva e redes

neurais em dois grupos principais: computação evolutiva para treinamento ou ajuste de pesos de redes neurais e computação evolutiva para definição de arquiteturas de redes neurais [Iyoda 2000].

Neste trabalho apresentamos uma abordagem inovadora para a otimização da arquitetura de uma rede Neural do tipo MLP baseado na utilização de meta-heurísticas de busca global.

3. Metodologia

Para resolver o problema correspondente à busca ótima de uma arquitetura de RNA implementamos o algoritmo conhecido como Variable Neighbourhood Search (VNS) ([Hansen and Mladenovic 1995];[Hansen and Mladenovic 2003] e [Mladenovic and Hansen 1997]), o Generalized Extremal Optimization (GEO) ([Bak 1993]; [Bak 1996] e [Sousa 2003]) e o Simulated Annealing (SA) [Kirkpatrick 1983].

3.1. Representação de uma solução candidata

Em nossa abordagem, empregamos os algoritmos supra-citados para evoluir: (i) o número de camadas ocultas; (ii)o número de elementos de processamento presentes em cada camada oculta;(iii) a taxa de aprendizagem da RNA;(iv)o termo de momentum;(v) a função de ativação da RNA.

A RNA construída a partir da informação codificada nas estruturas binárias é treinada sobre dados simulados gerados a partir do modelo LAMBDA (ver seção 4), que é um modelo direto bem conhecido e amplamente utilizado em estudos de dispersão de poluentes.

Os parâmetros do problema foram codificados, mediante um esquema de codificação indireta, em uma string binária de 22 bits para as simulação executadas nesse paper. O primeiro e o segundo grupos de 6 e 4 bits correspondem à taxa de aprendizagem e momentum respectivamente. O tipo de função de ativação empregada no treinamento da rede fica representado pelo segundo grupo de 2 bits que vem a seguir. Os últimos grupos de 5 bits cada são relativos ao número de neurônios em cada camada. Devido ao esquema de codificação binária e ao número de bits utilizado, o número de neurônios em cada camada oculta fica restrito a $[0, 31]$ já que $1 + 2 + 4 + 8 + 16 = 31$. Para melhor esclarecer o esquema de codificação aplicado, duas diferentes soluções candidatas, junto com suas formas decodificadas, são fornecidas na figura 1. Ao aumentarmos ou diminuirmos o comprimento da string de bits estaremos ampliando ou diminuindo a dimensão do espaço de arquiteturas possíveis garantindo assim a generalidade da metodologia. O limite de até 2 camadas ocultas e 31 bits em cada camada foi adotado nesse trabalho somente por razões práticas (tempo computacional). De qualquer forma, têm-se demonstrado que para RNAs com alimentação à frente (feedforward) e com funções de ativação contínuas e não-lineares, uma camada oculta com um número arbitrariamente grande de neurônios é suficiente para a propriedade de "aproximação universal" ([Bishop 1995];[Ripley 1996] e [Hornik 1993]), de modo que o limite empregado é mais que adequado para a maioria das aplicações.

3.2. 2.Aprendizado e teste da rede

Após codificarmos cada solução candidata a uma arquitetura de rede passamos para a etapa seguinte do algoritmo que corresponde ao treinamento da rede. Ao término do

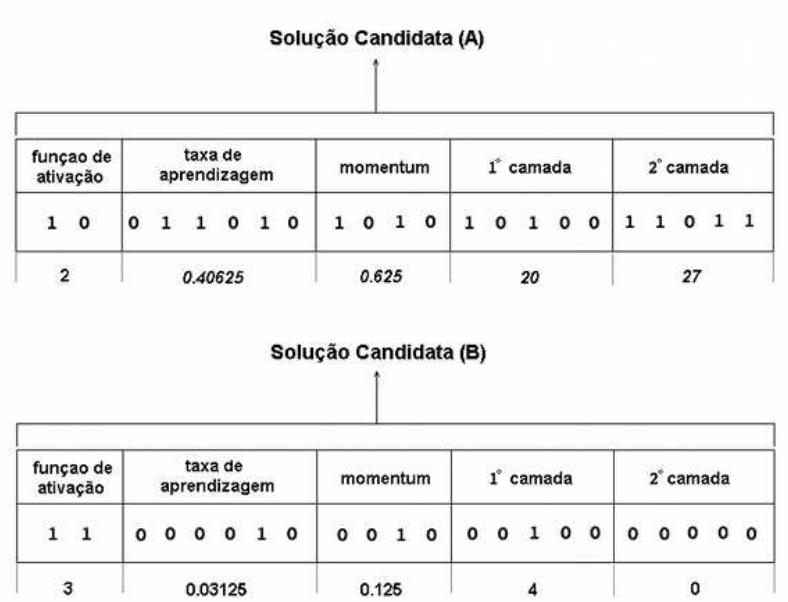


Figura 1. Exemplos de codificação binária de soluções candidatas a arquiteturas de rede e sua representação decimal.

treinamento a rede é testada com o conjunto de dados de teste, o que corresponde ao teste de generalização do modelo. Neste trabalho utilizamos uma função de inicialização de pesos randômicos para todos os pesos e bias com valores randômicos distribuídos no intervalo $[-1, 1]$ e a estratégia de treinamento com parada antecipada usando validação cruzada durante o treinamento da rede [Bishop 1995].

O processo de otimização de um modelo de rede que visa obter uma arquitetura ótima ou sub-ótima de rede segue etapas similares não importando qual seja a meta-heurística utilizada. Essas etapas comuns aos algoritmos de busca são melhor compreendidas e descritas pelo fluxograma exibido na figura 2 abaixo.

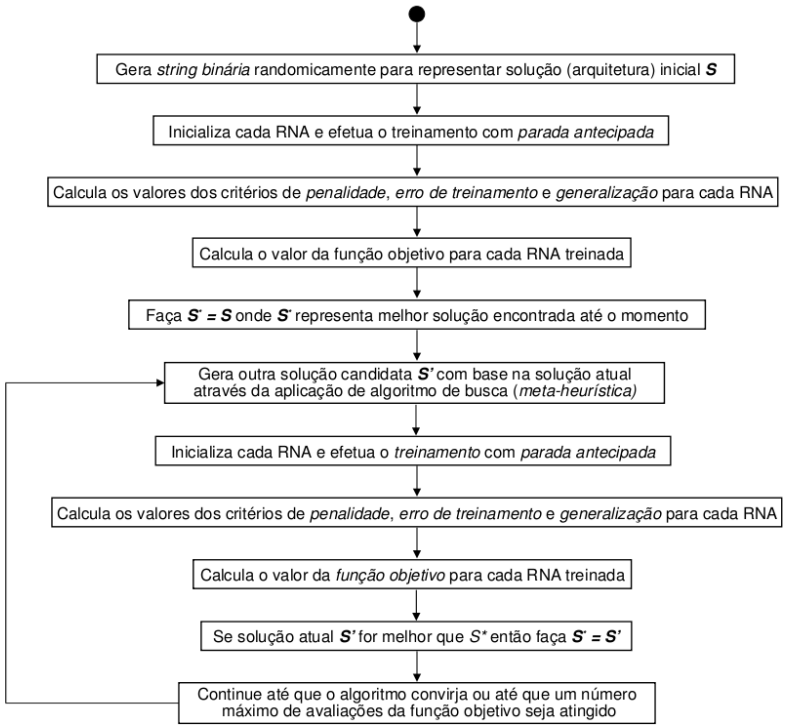


Figura 2. Síntese da metodologia empregada para o problema da otimização de uma arquitetura de RNA do tipo feedforward

3.3. Função objetivo

Neste trabalho lidamos com um problema de otimização combinatória relativamente complexo onde estamos interessados em encontrar uma arquitetura de custo mínimo, que apresente melhor desempenho com menor sobrecarga computacional possível. Para tanto, construímos uma função objetivo que consiste em uma combinação de 2 critérios de erro e um fator de penalização e que pôde ser expressa por :

$$f_{obj} = penalty \times \left(\frac{\alpha \times E_{train} + \beta \times E_{gen}}{\alpha + \beta} \right) \tag{1}$$

onde α e β ($\alpha, \beta > 0; \alpha, \beta \in \mathbb{R}$) são fatores de ajuste utilizados para ponderar o grau de importância atribuído ao erros de treinamento e generalização respectivamente. A presença desses parâmetros de ajuste permite uma maior flexibilidade na hora de avaliarmos a função objetivo. Na maioria dos casos teremos $\alpha < \beta$ pois estamos interessados em modelos que dêem boas respostas quando apresentamos à rede um conjunto de valores de entrada que não pertença ao conjunto de treinamento, ou seja, que saibam generalizar. Em particular, para este trabalho, utilizamos três conjuntos de valores para α e β (ver tabela 1).

A função f_{obj} é a função cujo valor deve ser minimizado pelos algoritmos implementados. Desta forma, fica claro que f_{obj} é constituída pela soma dos erros de treinamento e generalização multiplicado pela penalidade devida à complexidade da arquitetura de rede em questão. O mínimo valor para f_{obj} corresponde a uma arquitetura simples e que exibe comportamento consistente no espaço de soluções aliado a baixos erros de trei-

Tabela 1. Valores utilizados em cada um dos experimentos realizados nesse trabalho para os fatores de ponderação atribuídos ao erro de treinamento (α) e de generalização (β).

Experimento	α	β
#1	1.0	0.1
#2	1.0	1.0
#3	0.1	1.0

namento e generalização. Aqui entendemos por "simples" uma solução cuja arquitetura seja reduzida em termos de números de pesos em tempo de aprendizagem.

3.3.1. Detalhamento dos critérios utilizados na composição da função objetivo

Critério do erro de treinamento: Este termo é importante na medida que nos fornece uma indicação quantitativa do nível de treinamento da rede e está diretamente relacionado à habilidade de memorização da rede. A fórmula utilizada para o cálculo do erro de treinamento e que corresponde à fórmula do Erro quadrático médio foi a seguinte

$$E_{train} = \sqrt{\frac{1}{2N} \times \sum_{i=1}^{N} \sum_{j=1}^{M} \left[Y_{m_{ij}} - Y_{d_{ij}} \right]^2} \tag{2}$$

Onde E_{train} corresponde ao erro de treinamento, $Y_{d_{ij}}$ é o valor de saída desejado (alvo) no *i-ésimo* padrão de treinamento para o *j-ésimo* neurônio neurônio da camada de saída da rede, $Y_{d_{ij}}$ é a resposta da RNA no *i-ésimo* padrão de treinamento para o *j-ésimo* neurônio da camada de saída da rede, N é o número de padrões ou exemplos de treinamento apresentados à rede e M é ó número de neurônios na camada de saída.

Erro de generalização: Este critério refere-se a à capacidade da RNA de identificar e responder a padrões que são similares mas não idênticos aos padrões com os quais a rede foi treinada. Ele revela-se mais importante que o erro de treinamento como parâmetro indicativo da performance do modelo de RNA na maioria das aplicações. A fórmula utilizada para definir o erro de treinamento é a seguinte:

$$E_{gen} = \sqrt{\frac{\sum_{i=1}^{M} \left[Y_{m_i} - Y_{d_i} \right]^2}{M}} \tag{3}$$

onde E_{gen} é o erro de generalização, Y_{d_i} é o valor alvo para o *i-ésimo* neurônio de saída, Y_{m_i} é o valor previsto pela RNA para o *i-ésimo* neurônio de saída, M é o número de neurônios na camada de saída.

Critério de Penalidade devido à Complexidade da rede: O terceiro critério determina a influência que a arquitetura de RNA representa sobre os valores da função objetivo. Podemos definir a complexidade computacional de uma arquitetura de RNA do

tipo feedforward como sendo o número total de pesos e bias presentes em sua estrutura. Quando falamos em otimização de uma arquitetura de RNA estamos interessados em obter modelos ou arquiteturas que apresentem um desempenho o mais próximo possível de um ótimo global para o problema em questão ou, pelo menos, que produzam respostas sub-ótimas melhores do que as respostas fornecidas por um modelo não-otimizado. Em outras palavras, procuramos arquiteturas de RNA com baixos erros de treinamento e generalização. Para tanto, desenvolvemos um critério que favorece arquiteturas mais "leves" através da aplicação de um termo de penalidade à função objetivo. Com arquiteturas menores evitamos o sobre-ajuste (overfitting) dos dados de treinamento da rede [HAYKIN 1994], aceleramos o processo de treinamento, pois existirão menos unidades de processamento e fatores de pesos para serem calculados e aumentamos a sua habilidade de generalização na medida certa. Em um problema de otimização tal penalidade é incorporada à função objetivo de tal forma a exercer um papel de peso multiplicador e de ponderação. Idealmente, o valor da penalidade deveria ser uma função exponencial da complexidade de RNA e não uma função linear como poderíamos supor porque a soma do número de pesos e bias também aumenta de forma não-linear com o número de camadas ocultas e neurônios ocultos. Desta forma, a penalidade aplicada estaria aumentando lentamente enquanto o número total de pesos e bias é relativamente baixo e rapidamente quando a arquitetura é complexa. Portanto a forma geral do critério de penalidade de arquitetura de uma RNA seria dado por

$$P_1 = \alpha \times e^{f(x)} \tag{4}$$

onde α é uma constante e $f(x)$ uma função do número total de pesos e bias. Adicionalmente incluímos restrições relativas ao tempo de aprendizagem dos padrões presentes nos dados de entrada da RNA. Para tanto, introduzimos uma outra função de penalidade que leva em consideração o número de épocas necessárias para se realizar o treinamento. Em nosso trabalho definimos este fator penalidade como sendo uma função linear no número de épocas, ou seja, $g(x)$ é da forma $P_2 = \alpha \times y + \beta$.

Portanto, agrupando os termos definidos acima, obteremos a seguinte expressão global para a penalidade aplicada ao modelo:

$$Penalidade = P_1 + P_2 \tag{5}$$

Especificamente para o nosso trabalho foi utilizada a seguinte expressão global para penalidade devido à complexidade do modelo:

$$Penalidade = \underbrace{5 \times 10^{-8} \times e^{x^2}} + \underbrace{5 \times 10^{-5} \times y + 1} \tag{6}$$

onde x é o número de pesos das conexões e y o número de épocas necessárias no treinamento da rede. É importante salientar que tanto P_1 como P_2 podem ser representadas por qualquer função monótona contínua e estritamente crescente no intervalo $[1, \infty]$. A figura 3 abaixo exibe uma representação tridimensional da função de penalidade aplicada neste trabalho. Ao aplicar tal função de penalidade ao nosso problema estamos daremos preferência à soluções com menos neurônios em cada camada oculta e que, ao mesmo tempo, não necessite de muitas iterações para convergir a uma solução adequada.

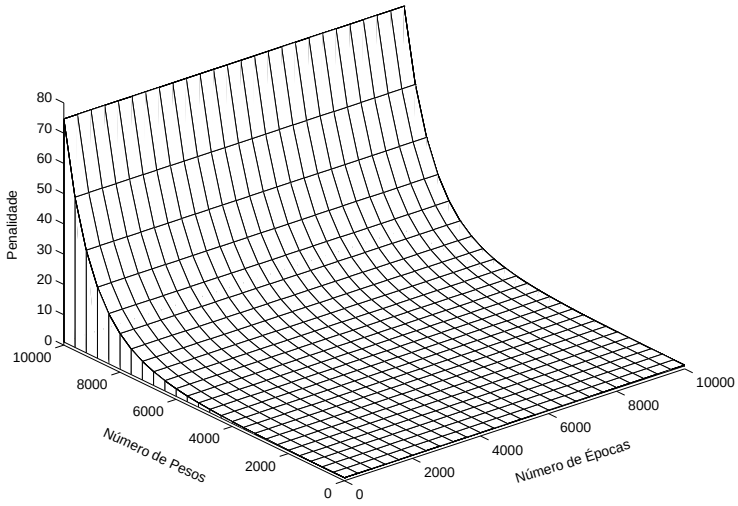


Figura 3. Visualização tridimensional da função de penalidade aplicada ao problema.

4. Estudo de caso: identificação de fontes de poluição atmosférica

Neste trabalho utiliza-se a RNA artificial perceptron de múltiplas camadas (MLP - Multilayer Perceptron) para resolver o problema relativo à estimação de fontes de poluição atmosférica. A metodologia foi testada utilizando dados originários do modelo de dispersão estocástico Lagrangeano LAMBDA (acrônimo para LAgrangean Model for Buoyant Dispersion in Atmosphere), desenvolvido para estudar os processos de transporte e difusão de poluentes sobre terreno plano (Ferrero et al. 1995, Ferrero & Anfossi 1998 a-b). Assim, utiliza-se o modelo direto descrito em [Luz 2007] para gerar 50 conjuntos de dados que são utilizados pelo modelo de inversão. O problema inverso a ser resolvido pelo modelo baseado em RNA consiste então em obter a concentração de poluentes nos 6 sensores a partir das fontes discretizadas em 12 subáreas.

5. Experimentos computacionais

A seguir, apresentamos os resultados computacionais obtidos a partir de execuções realizadas para cada uma das 3 metaheurísticas utilizadas nesses trabalho. Todos os algoritmos foram implementados na linguagem Java e os testes computacionais foram realizados sob o sistema operacional Linux, em um microcomputador com processador AMD Athlon(tm) 64 Processor 3200+ de 1,53 GHz e 1Gb MB de memória RAM.

5.1. Resultados

Foram realizados três experimentos computacionais de acordo com a tabela 1 (ver seção 3.3). Em todos existe a possibilidade de escolha entre quatro funções de ativação distintas: Tanh, Sigmóide, Logarítmica e Gaussiana. Os parâmetros correspondentes à melhor solução encontrada para um determinado número de avaliações da função objetivo são exibidos. Em um primeiro experimento atribuímos um peso maior à capacidade da RNA de aprender os padrões que lhe são apresentados em detrimento da sua capacidade de generalização. Os resultados são apresentados na tabela 2 abaixo.

Tabela 2. Melhores arquiteturas de RNA encontradas pelos algoritmos VNS, SA e GEO. Nesses experimentos os fatores de ponderação atribuídos ao erros de treinamento(α) e generalização(β) foram de 1.0 e 0.1, respectivemente.

Exp	Metaheurística	# Avaliações F_{obj}	Arquitetura	α	η	$\phi(\cdot)$	E_{trein}	E_{gen}
#1	VNS	100	6×31×12×12	0.34375	0.1875	Sigmóide	0.01502	0.5704
#2	VNS	1000	6×5×4×12	0.25	0.375	Sigmóide	0.02673	0.1924
#3	VNS	5000	6×6×15×12	0.25	0.1875	Tanh	0.00913	0.1429
#4	SA	100	6×6×11×12	0.265625	0.125	Logarítmica	0.02054	0.5338
#5	SA	1000	6×15×14×12	0.75	0.5	Sigmóide	0.01518	0.2887
#6	SA	5000	6×10×11×12	0.296875	0.0625	Tanh	0.00706	0.1478
#7	GEO_{var}	100	6×10×7×12	0.546875	0.0625	Sigmóide	0.01906	0.4367
#8	GEO_{var}	1000	6×26×12	0.28125	0.1875	Tanh	0.02034	0.7717
#9	GEO_{var}	5000	6×5×6×12	0.203125	0.0	Tanh	0.0113	0.28417

No segundo experimento o erro de aprendizagem e o erro de generalização da RNA tiveram pesos iguais na composição da função objetivo. Os resultados destas simulações são exibidos na tabela 3 abaixo:

Tabela 3. Melhores arquiteturas de RNA encontradas pelos algoritmos VNS, SA e GEO. Nesses experimentos os fatores de ponderação atribuídos ao erros de treinamento(α) e generalização(β) foram iguais ($\alpha = \beta = 1.0$).

Exp	Metaheurística	# Avaliações F_{obj}	Arquitetura	α	η	$\phi(\cdot)$	E_{trein}	E_{gen}
#1	VNS	100	6×15×16×12	0.125	0.1875	Gauss	0.02992	0.5030
#2	VNS	1000	6×11×30×12	0.578125	0.4375	Sigmóide	0.01842	0.4120
#3	VNS	5000	6×8×8×12	0.046875	0.75	Tanh	0.02333	0.1651
#4	SA	100	6×8×10×12	0.984375	0.5	Sigmóide	0.02580	0.3358
#5	SA	1000	6×14×15×12	0.109375	0.625	Tanh	0.00739	0.1527
#6	SA	5000	6×5×6×12	0.140625	0.5	Tanh	0.01676	0.0756
#7	GEO_{var}	100	6×3×16×12	0.09375	0.25	Tanh	0.02215	0.8788
#8	GEO_{var}	1000	6×7×18×12	0.078125	0.375	Logarítmica	0.02501	0.4381
#9	GEO_{var}	5000	6×23×12	0.1875	0.625	Sigmóide	0.02082	0.6103

Por último, emulamos um cenário mais realístico, no qual a capacidade de generalizar,ou seja, encontrar um valor de saída correto para um padrão desconhecido para a RNA, possui um peso maior do que a capacidade de aprendizagem do modelo. Os resultados são apresentados na tabela 4 abaixo.

Tabela 4. Melhores arquiteturas de RNA encontradas pelos algoritmos VNS, SA e GEO. Nesses experimentos os fatores de ponderação atribuídos ao erros de treinamento(α) e generalização(β) foram de 0.1 e 1.0, respectivemente.

Exp	Metaheurística	# Avaliações F_{obj}	Arquitetura	α	η	$\phi(\cdot)$	E_{trein}	E_{gen}
#1	VNS	100	6×18×25×12	0.484375	0.625	Sigmóide	0.02083	0.4371
#2	VNS	1000	6×9×17×12	0.296875	0.125	Tanh	0.0167	0.1753
#3	VNS	5000	6×17×2×12	0.109375	0.625	Logarítmica	0.01548	0.1126
#4	SA	100	6×12×12×48	0.21875	0.0	Tanh	0.01066	0.3004
#5	SA	1000	6×15×14×12	0.75	0.5	Sigmóide	0.01518	0.2887
#6	SA	5000	6×3×13×12	0.0625	0.5	Logarítmica	0.02228	0.1331
#7	GEO_{var}	100	6×25×12	0.453125	0.0	Tanh	0.02867	1.2825
#8	GEO_{var}	1000	6×22×15×12	0.046875	0.75	Logarítmica	0.01727	0.4073
#9	GEO_{var}	5000	6×5×30×12	0.28125	0.125	Tanh	0.01264	0.3503

A análise dos resultados obtidos neste trabalho (ver tabelas 2-4) evidencia as virtudes de se utilizar uma abordagem evolutiva automatizada na busca ótima de parâmetros de um modelo baseado em RNAs.

Foram geradas arquiteturas com grande capacidade de generalização para os dados de entrada de rede e com baixo custo computacional como por exemplo nos experimentos 3, 6 e 3 das tabelas 2,3 e 4 respectivamente.

A melhor arquitetura de rede encontrada foi obtida após 5000 iterações do algoritmo Simulated Annealing e corresponde aos parâmetros de rede exibidos no sexto experimento da tabela 3. A solução aponta para uma arquitetura de rede com duas camadas ocultas e poucos neurônios em cada uma delas(5 e 6, respectivamente) e sugere a aplicação de uma função de ativação do tipo Tangente Hiperbólica nas camadas intermediárias. Curiosamente essa solução foi obtida atribuindo-se pesos equivalente tanto para treinamento quanto para generalização da rede. Outras soluções igualmente interessantes do ponto de vista do custo computacional requerido e desempenho oferecido foram obtidas nos experimento 2 e 9 da tabela 2 e experimento 3 da tabela 3. Ademais,ao se comparar as tabelas acima, constata-se que os fatores de ponderação atribuídos ao erro de treinamento e generalização têm influência na solução de arquitetura final obtida pelo algoritmo de busca. Como era de se esperar, os resultados obtidos para os erros de generalização exibidos na tabela 2 são maiores, na média, do que os da tabela 4. Isso se explica devido ao fato de atribuírmos um peso maior ao erro de treinamento em detrimento do erro de generalização na composição da função objetivo nos experimentos da tabela 2 e o procedimento inverso com relação aos dados relativos à tabela 4.

Comparativamente, pode-se inferir dos resultados que as arquiteturas produzidas pelas metaheurísticas Simulated annealing e VNS foram superiores às geradas através do GEO_{VAR} sendo que a razão entre a melhor e a pior solução obtida nesses experimentos, levando-se em consideração o mesmo número de avaliações da função objetivo em cada caso, foi de 0.2342, 0.1978 e 0.1238 para 100, 1000 e 5000 avaliações, respectivamente. Esses valores revelam outro fato interessante a respeito da dinâmica evolutiva do modelo proposto: com o aumento no número de avaliações de função objetivo alguns algoritmos ou metaheurísticas tendem a se sobressair em relação aos demais. É importante ressaltar o foco desse trabalho, o qual não diz respeito à utilização e posterior comparação das 3 metaheurísticas aqui empregadas, mas sim a destacar e propor alternativas à estratégia de busca de "às cegas" de uma arquitetura de rede, realizada mediante o processo repetitivo de tentativa-e-erro. Obviamente, os resultados obtidos e exibidos nas tabelas 2, 3 e 4 acima poderiam também ser obtidos com a utilização de outras metaheurísticas. O que podemos inferir desses e de outros resultados encontrados na literatura ([Yao and Liu 1997];[Miller 1989];[Arifovic 2001];[Bernardos 2004] e [Chen 1997]) é que o simples emprego de algum método sistemático e automatizado de busca como o proposto nesse trabalho é preferível à sua ausência.

Por fim, podemos dizer que, embora a abordagem tenha sido aplicada para a resolução de um problema específico, o da estimação de fontes de poluição atmosférica, ela é genérica e mostra-se robusta o suficiente para ser adaptada e aplicada a qualquer problema que possa ser resolvido através de RNAs.

6. Conclusões e trabalhos futuros

Este paper propõe uma nova metodologia para escolha de arquitetura de RNA do tipo feedforward com mínima complexidade e desempenho otimizado. Três meta-heurísticas de busca global evolutivas foram adaptadas e comparadas para treinamento de redes neurais

e para definição de arquiteturas de redes neurais. A correta estimação da intensidade e localização das fontes de poluentes constituiu o estudo de caso para a metodologia desenvolvida. O processo de escolha da rede levou em consideração não apenas o critério baseado no erro de treinamento da rede mas também no erro de generalização. Mais ainda, o erro de generalização não é simplesmente avaliado mas diretamente calculado durante o processo evolucionário utilizando um subconjunto de teste depois que cada arquitetura é treinada. Um novo critério para medição de complexidade de modelos de redes neurais baseado no número de pesos presentes na rede e no número de épocas necessárias para o treinamento é criado e incluído na metodologia. Baseados nos resultados obtidos e naqueles apresentados na literatura, devemos ressaltar que é plenamente justificável evoluir a arquitetura de uma RNA caso se queira obter uma solução dedicada e completamente adaptada ao contexto do problema que se deseja resolver. O tempo necessário para evoluirmos uma arquitetura e encontrarmos uma configuração adequada ao problema é plenamente justificável se comparamos o tempo gasto no processo empírico baseado em tentativa-e-erro. Além disso obtemos uma arquitetura de rede quase-ótima mediante a utilização de um processo autônomo, sistemático e bem comportado e não apenas como resultado da experiência, intuição ou até mesmo sorte.

7. Acknowledgements

O autor agradece o suporte físico e financeiro fornecido à esta pesquisa pela CAPES e CNPq.

8. References

Referências

Arifovic, J., G. R. (2001). Using genetic algorithms to select architecture of a feedforward arti
cial neural network. pages 574–594. Physica A n. 289.

Bak, P. (1996). How nature works. In *How nature works*. New York:Copernicus, Springer-Verlag.

Bak, P., S. K. (1993). Punctuated equilibrium and criticality in a simple model of evolution. Physical Review Letters, v. 71, n. 24, p. 4083-4086.

Bernardos, P. G., V. G. C. (2004). Optimizing feedforward artificial neural network architecture. Engineering Applications of Artificial Intelligence, v.20 n.3, p.365-382.

Bishop, C. (1995). Neural networks for pattern recognition. In *Neural Networks for Pattern Recognition*. OxfordUniversity Press.

Chen, Z., X. J. C. J. (1997). A program for automatic structure search. pages 308–311. Proceedings of the 1997 International Conference on Neural Networks.

Hansen, P. and Mladenovic, N. (1995). A variable neighborhood algorithm: a new metaheuristic for combinatorial optimization. Abstracts of Papers at Optimization Days n. 112.

Hansen, P. and Mladenovic, N. (2003). Variable neighborhood search. In *Handbook of Metaheuristics*, pages 145–184. Glover, F. and Kochenberger, G.

HAYKIN, S. (1994). Neural networks. a comprehensive foundation. In *Neural Networks. A Comprehensive Foundation*. Macmillan, New York, NY.

Hornik, K. (1993). Some new results on neural network approximation. Neural Networks n. 6, 1069-1072.

Iyoda, E. M. (2000). Inteligencia computacional no projeto automatico de redes neurais hibridas e redes neurofuzzy heterogeneas. Tese de Doutorado - Universidade Estadual de Campinas (UNICAMP) - Campinas, SP: [s.n.].

Kirkpatrick, S., G. J. C. D. V. M. P. (1983). Optimization by simulated annealing. Science, 220, N. 4598, pp.671-680.

Luz, E.F.P., C. V. H. B. J. R. D. (2007). Estimating atmospheric area source strength through particle swarm optimization. Inverse Problems, Desing and Optimization Symposium IPDO-2007, April 16-18, Miami (FL), USA, vol 1, p. 354-359.

Miller, G.F., T. P. H. S. (1989). Designing neural networks using genetic algorithms. pages 379–384. Proceedings of the Third International Conference on Genetic Algorithms.

Mladenovic, N. and Hansen, P. (1997). Variable neighbourhood search. pages 1097–1100. Computers and Operations Research, n. 24.

Paes, F.F., C. V. H. F. C. A. L. E. (2008). Identifing atmospheric pollutant sources using artificial neural networks. pages 131–134. AGU 2008 Joint Assembly, Fort Lauderdale (FL), USA.

Ripley, B. (1996). Pattern recognition and neural networks. In *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge.

Roberti, D. R., A. D. C. V. H. F. D. G. A. (2005). Estimating emission rate and pollutant source location. pages 131–134. Ciencia e Natura.

rojas, R. (1996). Neural networks, a systematic introduction. In *Neural Networks, a Systematic Introduction*. Springer-Verlag.

RUMELHART, D.E., H. G. W. R. (1986). Learning internal representations of back-propagating error. Nature, v.323, p.533-536.

Sousa, F.L. Vlassov, V. e. R. F. (2003). Generalized extremal optimization for solving complex optimal design problems. pages 375–376. Lectures Notes on Computer Science n. 2723.

Yao, X., L. Y. (1997). A new evolutionary system for evolving artificial neural networks. IEEE Transactions on Neural Networks 8, 694-713.

Yao, X. and Liu, Y. (1997). A new evolutionary system for evolving artificial neural networks. pages 694–713. IEEE Transactions on Neural Network, 8(3).