Computação evolucionária para otimização dinâmica de parâmetros de EFuNNs

L. Minku[1] **, T. Ludermir**[1] **, A. Araújo**[1]

[1]Centro de Informática – Universidade Federal de Pernambuco
Caixa Postal 7851 – 50732-970 Recife, PE

{flm,tbl,aluizioa}@cin.ufpe.br

Abstract. *This paper compares two approaches of on-line optimization of evolving connectionist systems (ECOSs) using evolutionary computation. One of the approaches uses a genetic algorithm and the other one uses an evolutionary strategy. Evolving connectionist systems make easier the evolutive processes modeling task, have evolving structure and do local, on-line, lifelong, incremental and fast learning. The optimization consisted of the adjustment of some evolving fuzzy neural networks (EFuNNs) parameters. EFuNNs constitute a class of evolving connectionist systems. These networks were used to Mackey-Glass chaotic time series prediction.*

Resumo. *Este artigo compara duas abordagens de otimização on-line de sistemas conexionistas evolutivos (ECOSs) utilizando computação evolucionária. Uma das abordagens faz uso de um algoritmo genético e a outra faz uso de uma estratégia evolucionária. Os ECOSs facilitam a modelagem de processos evolutivos, possuem estrutura evolutiva e realizam aprendizado local, on-line, lifelong, incremental e rápido. A otimização consistiu no ajuste de alguns parâmetros de redes neurais difusas evolutivas (EFuNNs), que constituem uma classe de ECOSs. Estas redes foram utilizadas para previsão da série temporal caótica Mackey-Glass.*

1. Introdução

Processos evolutivos são processos que se desenvolvem e se modificam de maneira contínua no tempo. Como exemplos de processos desse tipo, podem ser citados diversos problemas do mundo real, como processamento de dados biológicos, previsão de carga elétrica e reconhecimento adaptativo de palavras. Processos evolutivos apresentam dificuldades de modelagem, pois alguns de seus parâmetros podem não ser conhecidos *a priori*, perturbações ou mudanças inesperadas podem ocorrer durante o seu desenvolvimento e eles não são previsíveis em longo prazo [Kasabov, 2003].

Os sistemas conexionistas evolutivos (ECOSs) formam um paradigma criado para facilitar a modelagem de processos evolutivos. As redes neurais difusas evolutivas (EFuNNs) [Kasabov, 2001] formam um exemplo de uma classe de ECOSs que facilitam a representação e a extração de conhecimento através de regras difusas.

Embora os ECOSs evoluam suas estruturas de acordo com os dados de entrada no tempo, eles ainda possuem alguns parâmetros fixos, que não são ajustados durante o aprendizado. Muitas vezes o conjunto ótimo desse tipo de parâmetros depende dos dados utilizados. Assim, para modelar processos evolutivos, é importante que esses parâmetros possam se modificar conforme os dados utilizados. Para realizar o ajuste desses parâmetros pode ser utilizada computação evolucionária (CE).

A combinação de redes neurais com CE têm sido explorada em diversos trabalhos, por exemplo, [Braun and Weisbrod, 1993], [Angeline et al., 1994], [Branke, 1995], [Lee and Kim, 1996], [Yao and Liu, 1997], [Yao and Liu, 1998], [Yao, 1999], [Watts and Kasabov, 2001], [Stanley and Miikkulainen, 2002]. Exemplos de tarefas que podem ser realizadas através de algoritmos evolucionários são treinamento dos pesos das conexões, otimização da arquitetura, adaptação da taxa de aprendizado, seleção de atributos de entrada, inicialização dos pesos das conexões, extração de regras, etc.

Neste artigo são apresentadas duas variações de um método (introduzido por [Kasabov et al., 2003]) que realiza o ajuste *on-line*, utilizando CE, de parâmetros das EFuNNs que não são ajustados dentro do algoritmo de aprendizado. A primeira variação utiliza um algoritmo genético (AG), conforme o experimento feito em [Kasabov et al., 2003], e a segunda utiliza uma estratégia evolucionária (EE). As EFuNNs foram utilizadas para previsão de séries temporais caóticas.

2. ECOSs e EFuNNs

Os ECOSs são sistemas formados por uma ou mais redes neurais e que possuem as seguintes características [Kasabov, 2003]:

- facilitam a modelagem de processos evolutivos;
- facilitam a representação e a extração de conhecimento;
- realizam aprendizado:
  - *lifelong*: aprendem durante toda a sua existência a partir de dados que vêm continuamente de um ambiente mutante;
  - *on-line*: aprendem cada exemplo separadamente, enquanto o sistema opera (muitas vezes em tempo real);
  - incremental: aprendem novos dados sem destruir totalmente os padrões aprendidos anteriormente e sem a necessidade de realizar um novo treinamento com os dados antigos;
  - rápido, possivelmente através de uma única passada pelos dados;
  - local, permitindo adaptação rápida e tratamento de processos evolutivos no tempo;
- possuem estrutura evolutiva, através de construtivismo;
- evoluem em um espaço aberto, não necessariamente de dimensões fixas.

As EFuNNs [Kasabov, 2001] constituem uma classe de ECOSs que unem as funcionalidades das redes neurais ao poder de expressividade da lógica difusa. Elas possuem arquitetura composta por cinco camadas, conforme a figura 1. A primeira camada recebe o vetor de entrada, a segunda representa a quantificação difusa do vetor de entrada, a terceira representa associações entre o espaço de entradas e o de saídas difusas, a quarta representa a quantificação difusa do vetor de saída e a quinta representa o vetor de saída.

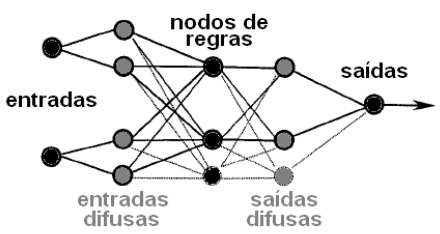

Figura 1: Arquitetura de uma rede EFuNN

O aprendizado ocorre na camada de nodos de regras. Cada nodo r_j desta camada realiza uma associação entre uma hiperesfera do espaço de entradas difusas e uma hiperesfera do espaço de saídas difusas. O centro das hiperesferas de entrada e saída são representados pelos vetores de pesos $W1(r_j)$ e $W2(r_j)$, respectivamente. Os pesos $W1$ são ajustados através de aprendizado não supervisionado, para criar agrupamentos no espaço de entradas difusas. Já os pesos $W2$ são ajustados através de aprendizado supervisionado, de forma a fazer com que a rede obtenha a saída desejada. As regras de aprendizado são:

- $W1(r_j(t+1)) = W1(r_j(t)) + lr1(r_j(t)) * (x_f - W1(r_j(t)))$
- $W2(r_j(t+1)) = W2(r_j(t)) + lr2(r_j(t)) * (y_f - A2) * A1(r_j(t))$
  Onde: x_f e y_f são os vetores de entradas e saídas difusas, respectivamente; $lr1(r_j(t))$ e $lr2(r_j(t))$ são as taxas de aprendizado para ajuste dos pesos $W1$ e $W2$ do nodo r_j no tempo t; $A2$ é o vetor de ativações das saídas difusas e $A1(r_j(t))$ é o valor escalar de ativação do nodo de regras r_j no tempo t.

Essas regras de aprendizado fazem com que as hiperesferas do espaço de entradas difusas e de saídas difusas desloquem seus centros na direção dos exemplos de treinamento apresentados.

A seguir é brevemente apresentado o algoritmo de treinamento das EFuNNs. Para maiores detalhes, consulte [Kasabov et al., 2003].

Algoritmo 2.1 (Treinamento das EFuNNs)

1. Estabeleça os parâmetros iniciais do sistema: número de funções de pertinência; limiar de sensibilidade inicial S dos nodos (que também é utilizado para determinar o raio inicial da hiperesfera de entradas difusas $R(r_j) = 1 - S$, para todos os nodos r_j criados); limiar de erro E; parâmetro de agregação $Nagg$; parâmetros de poda OLD e Pr; valor *m-de-n* (que é o número de nodos de maior ativação utilizados para o treinamento); raio máximo da hiperesfera de entradas difusas $Rmax$ e limiares $T1$ e $T2$ para extração de regras.
2. Inicie o primeiro nodo de regras r_0 de forma que ele memorize o primeiro exemplo (x, y):
   $$W1(r_0) = x_f \text{ e } W2(r_0) = y_f$$
   onde os vetores x_f e y_f representam as quantificações difusas do vetor x e do vetor y, respectivamente.
3. Repita para cada novo par de entrada-saída (x, y):
   (a) Determine a distância difusa local normalizada D entre x_f e os pesos $W1$ existentes. A distância D entre dois vetores difusos $x1$ e $x2$ é definida por:
   $$D(x1, x2) = subabs(x1, x2)/sumabs(x1, x2)$$
   Onde $subabs(x1, x2)$ é a soma de todos os valores absolutos do vetor obtido a partir da subtração dos vetores difusos $x1$ e $x2$ e $sumabs(x1, x2)$ é a soma de todos os valores absolutos do vetor obtido a partir da soma dos vetores difusos $x1$ e $x2$.
   (b) Calcule as ativações $A1$ dos nodos de regras. Um exemplo de como $A1$ pode ser calculada é $A1 = 1 - D(W1(r_j)), x_f)$.
   (c) Selecione o nodo de regras r_k que possui a menor distância $D(x_f, W1(r_k))$ e ativação $A1(r_k) >= S(r_k)$. No caso de aprendizado *m-de-n* selecione *m* nodos ao invés de um nodo.
   (d) Se não existe tal nodo
      i. Crie um novo nodo de regras para acomodar o exemplo (x_f, y_f).
   (e) Senão

i. Determine a ativação $A2$ da camada de saída e o erro de saída normalizado $Err = subabs(y, y')/Nout$, onde y é a saída desejada, y' é a saída obtida e $Nout$ é o número de nodos da camada de saída.
ii. Se $Err > E$
    A. Crie um novo nodo de regras para acomodar o exemplo (x_f, y_f).
iii. Senão
    A. Aplique as regras de aprendizado a $W1(r_k)$ e $W2(r_k)$ (no caso de aprendizado *m-de-n* as regras são aplicadas a todos os *m* nodos de regras).

(f) Aplique o procedimento de agregação depois da apresentação de cada grupo de Nagg exemplos.
(g) Atualize os parâmetros $S(r_k)$, $R(r_k)$, $Age(r_k)$ e $TA(r_k)$. $TA(r_k)$ pode ser, por exemplo, a soma das ativações $A1$ causadas por todos exemplos que o nodo r_k acomoda.
(h) Realize poda dos nodos de regras, se necessário, de acordo com os parâmetros OLD e Pr.
(i) Realize extração de regras, de acordo com os parâmetros $T1$ e $T2$.

Como pode ser visto no algoritmo 2.1, existem parâmetros que são ajustados durante o aprendizado (nodos de regras e seus pesos) e parâmetros que definem o aprendizado (número de funções de pertinência, *E, Nagg, OLD, Pr,* valor *m-de-n, Rmax, T1 e T2*) [Kasabov et al., 2003].

3. Otimização *on-line* de parâmetros de ECOSs baseada em computação evolucionária

Através do uso de diferentes parâmetros do tipo que define o aprendizado, os ECOSs atingem diferentes performances e diferentes pesos são aprendidos. Normalmente o conjunto de parâmetros ótimos depende dos dados de entrada e saída apresentados. O objetivo de aplicar otimização *on-line* de parâmetros nos ECOSs é determinar, ao mesmo tempo em que ocorre o aprendizado, o melhor conjunto de parâmetros para o momento.

Em [Kasabov et al., 2003], foi introduzido um método que utiliza computação evolucionária para realizar otimização *on-line* de parâmetros de ECOSs. A seguir serão explicadas as duas variações deste método que foram implementadas no presente trabalho.

3.1. Otimização através de algoritmo genético

A primeira variação do método introduzido em [Kasabov et al., 2003] utiliza um AG, conforme o experimento realizado em [Kasabov et al., 2003], para realizar a otimização de parâmetros de EFuNNs utilizadas com processos dinâmicos.

A representação utilizada é binária, a mutação é realizada a partir da inversão de bits, realizada bit-a-bit, o cruzamento é de um ponto, a seleção de pais é proporcional à aptidão, através do método da roleta, e a seleção de sobreviventes é generacional. Maiores detalhes podem ser observados no algoritmo 3.1.

Algoritmo 3.1 (Otimização através de AG)

1. Inicialize a população com indivíduos criados a partir da escolha aleatória de valores para cada um dos bits do genótipo (o genótipo codifica os parâmetros a serem otimizados).
2. Defina uma janela com *P* pontos no tempo para treinamento das EFuNNs e, opcionalmente, defina uma janela com *P* pontos no tempo para teste.

3. Repita até os dados de treinamento/teste acabarem:
   (a) Realize o aprendizado de cada EFuNN da população a partir dos *P* pontos da janela de tempo de treinamento, utilizando os parâmetros determinados através dos genótipos dos indivíduos.
   (b) Determine a raiz quadrada do erro quadrático médio (RMSE) de cada EFuNN (a partir da previsão feita sobre os mesmos dados da janela de tempo utilizada para treinamento, ou a partir da previsão feita sobre os dados da janela de tempo de teste, caso utilizada) e utilize-o como valor de aptidão a ser minimizado.
   (c) Selecione pais através do método da roleta e de probabilidades determinadas a partir da maior aptidão da geração menos a aptidão do indivíduo.
   (d) Embaralhe os pais e aplique cruzamento e mutação com probabilidades *Pc* e *Pm*, respectivamente, para gerar novos indivíduos (os nodos de regras dos pais são herdados pelos filhos).
   (e) Realize seleção de sobreviventes generacional.
   (f) Desloque a janela de tempo de treinamento e a de teste, caso utilizada (as novas janelas serão compostas de $P - D$ pontos anteriores e de D pontos novos).

3.2. Otimização através de estratégia evolucionária

A segunda variação do método introduzido em [Kasabov et al., 2003] utiliza uma EE para realizar a otimização de parâmetros de EFuNNs utilizadas com processos dinâmicos.

As EEs possuem as seguintes características [Eiben and Smith, 2003] desejáveis para realizar a otimização dos parâmetros de EFuNNs utilizadas com processos dinâmicos:

- São boas otimizadoras de valores reais.
- Normalmente são utilizadas com auto-ajuste dos parâmetros de mutação. Como a superfície de aptidão não é conhecida e pode variar com o tempo (de acordo com as variações das características dos dados de teste), isto é desejável.
- Possuem um baixo tempo de *takeover*, fazendo com que a população se ajuste mais rapidamente às características dos dados do momento. Isto é bom para otimização *on-line* porque não é desejável que a população não tenha se adequado antes de ocorrerem outras mudanças nas características dos dados.

A EE utilizada realiza mutação por perturbação gaussiana com um desvio padrão auto-ajustável para cada variável que representa um parâmetro da EFuNN a ser otimizado. Este tipo de mutação é adequado para otimização de parâmetros numéricos ordinais. Para a geração dos números aleatórios da distribuição gaussiana foram utilizados números aleatórios de uma distribuição normal, obtidos através do método Polar [Knuth, 1998].

A representação é real e possui dois genes para cada parâmetro da EFuNN a ser otimizado, sendo um para representar o próprio parâmetro e o outro para representar o desvio padrão auto-ajustável correspondente a este parâmetro.

O tipo de cruzamento utilizado é local discreto para as variáveis que representam os parâmetros a serem ajustados e local intermediário para as variáveis que representam os desvios padrões auto-ajustáveis, conforme é sugerido em [Eiben and Smith, 2003].

A seleção de pais realizada é aleatória e a seleção de sobreviventes é (μ, λ). Esta última foi escolhida porque ela é melhor que a $(\mu + \lambda)$ para seguir pontos ótimos que se movem no espaço de busca, para escapar de ótimos locais do espaço de busca e para utilização com auto-ajuste dos parâmetros de mutação [Eiben and Smith, 2003].

Maiores detalhes sobre a EE utilizada podem ser observados no algoritmo 3.2.

Algoritmo 3.2 (Otimização através de EE)

1. Inicialize a população com indivíduos criados a partir da escolha aleatória de valores para cada um dos parâmetros a serem otimizados (o genótipo codifica os parâmetros a serem otimizados).
2. Defina uma janela com *P* pontos no tempo para treinamento das EFuNNs e, opcionalmente, defina uma janela com *P* pontos no tempo para teste.
3. Realize o aprendizado de cada EFuNN da população a partir dos *P* pontos da janela de tempo de treinamento, utilizando os parâmetros determinados através dos genótipos de cada indivíduo.
4. Determine o RMSE de cada EFuNN da população (a partir da previsão feita sobre os mesmos dados da janela de tempo utilizada para treinamento, ou a partir da previsão feita sobre os dados da janela de tempo de teste, caso utilizada) e utilize-o como valor de aptidão a ser minimizado.
5. Repita até os dados de treinamento/teste acabarem:
   (a) Selecione aleatoriamente N pais, utilizando probabilidades iguais para todos os indivíduos da população.
   (b) Aplique cruzamento e mutação, gerando um filho para cada par de pais.
   (c) Desloque a janela de tempo de treinamento (a nova janela será composta de $P - D$ pontos anteriores e de D pontos novos). Desloque, da mesma forma, a janela de tempo de teste, caso ela esteja sendo usada.
   (d) Realize o aprendizado de cada filho.
   (e) Determine o RMSE de cada filho.
   (f) Realize seleção de sobreviventes (μ, λ), *i.e.* selecione deterministicamente os μ (μ é o tamanho da população, que neste caso é 12) melhores indivíduos dentre todos os λ filhos gerados. O número de filhos gerados é $\lambda = N/2$ e $\lambda > \mu$.

4. Experimentos realizados

Foram utilizados os algoritmos apresentados nas seções 3.1 e 3.2 para realizar a otimização *on-line* de alguns parâmetros que definem o aprendizado de EFuNNs, as quais foram utilizadas para previsão de séries temporais caóticas Mackey-Glass. Estas séries, bem como os parâmetros da otimização utilizados, foram escolhidos numa tentativa de reproduzir ao máximo o experimento realizado em [Kasabov et al., 2003]. Porém, este experimento não pôde ser reproduzido por completo porque o artigo não informa todos os parâmetros necessários para a sua execução.

4.1. Otimização de EFuNNs para previsão de séries temporais caóticas

Determinadas séries temporais manifestam comportamento caótico, *i.e.* elas apresentam alguns padrões vagos de repetição no tempo e são aproximadamente previsíveis no futuro próximo, mas não a longo termo [Kasabov, 2003].

A série utilizada neste trabalho é a *benchmark* Mackey-Glass [Mackey and Glass, 1977], descrita pela seguinte equação diferencial:

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1 + x^{10}(t-\tau)} - bx(t)$$

Esta série possui comportamento caótico para alguns valores dos parâmetros τ, a, b e para o valor inicial $x(0)$. Neste artigo são utilizados os valores $a = 0,2$; $b = 0,1$; $x(0) = 1,2$ e $x(t) = 0$ para $t < 0$. O valor τ sofre uma mudança de 17 para 19 durante

a execução dos algoritmos, causando uma modificação na natureza caótica e no atrator da série. Embora esta modificação não seja drástica, ela ainda assim pode causar falhas na previsão da série com o parâmetro modificado [Kasabov, 2003]. Os valores utilizados fazem com que a série apresente comportamento caótico.

Foram realizados dois tipos de experimentos. No experimento 1, o RMSE utilizado como aptidão dos indivíduos foi obtido a partir da predição feita sobre a mesma janela de tempo utilizada para treiná-los. Para este experimento foram utilizados os pontos de 0 a 599 com $\tau = 17$ e os pontos de 600 a 1199 com $\tau = 19$. No experimento 2, o RMSE de predição utilizado como aptidão dos indivíduos foi obtido a partir de uma janela de tempo diferente da utilizada para treiná-los. Para este experimento foram utilizados para treinamento os pontos da série de 0 a 599 com $\tau = 17$ e os pontos de 1200 a 1799 com $\tau = 19$ e, para teste, os pontos de 600 a 1199 utilizando $\tau = 17$ e de 1800 a 2399 utilizando $\tau = 19$. A figura 2 possui as séries utilizadas.

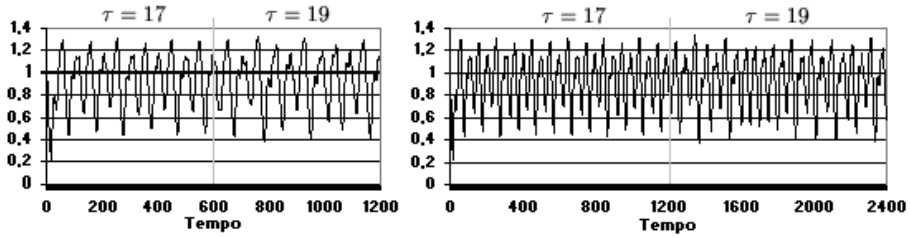

Figura 2: Séries Mackey-Glass utilizadas nos experimentos

A tarefa de previsão a ser realizada pelas EFuNNs é determinar o valor de $x(t+6)$, dados os vetores de dados $[x(t-18), x(t-12), x(t-6), x(t)]$.

4.2. Parâmetros da otimização

Os parâmetros otimizados e os intervalos de valores permitidos pelas representações dos algoritmos evolucionários utilizados nos experimentos são:

- *m-de-n*: valores inteiros no intervalo $[1; 8]$;
- *Rmax*: valores reais no intervalo $[0,75; 0,95]$;
- *E*: valores reais no intervalo $[0,05; 0,25]$.

Na representação utilizada pelo AG, cada genótipo é composto por 15 bits (3 para representar o valor *m-de-n*, 6 para o *Rmax* e 6 para o *E*). A taxa de mutação usada foi de 1% e a de cruzamento foi de 70%.

Na representação escolhida para ser utilizada com a estratégia evolucionária, cada genótipo é composto por 6 variáveis do tipo real (uma variável para cada parâmetro da EFuNN a ser otimizado e uma para cada desvio padrão auto-ajustável). Note que esta representação é mais adequada para otimização de valores numéricos ordinais que a representação binária, já que na binária pequenas alterações no genótipo podem causar grandes alterações no fenótipo [Eiben and Smith, 2003]. O número de filhos utilizado foi quatro vezes maior que o número de indivíduos da população ($N = 96$).

O tamanho utilizado para a janela de tempo foi de $P = 200$ pontos de dados, com 90% de sobreposição entre janelas consecutivas ($D = 20$). Note que, desta forma, a mudança no valor de τ ocorre na geração 21, tanto para o experimento 1, quanto para o experimento 2. O tamanho da população utilizado foi 12.

4.3. Testes executados

Foram executados experimentos de acordo com as seções 4.1 e 4.2. Conforme já havia sido explicado na seção 4.1, os experimentos foram divididos em dois tipos. Para cada

tipo foram realizadas dez execuções do AG e dez execuções da EE, para verificar se a EE adotada é capaz de produzir melhores resultados que o AG, já que ela possui características mais adequadas para o problema abordado.

4.4. Resultados obtidos

A figura 3 possui um gráfico com a média (dentre as 10 rodadas do AG do experimento 1) da aptidão obtida pelo melhor indivíduo de cada geração do AG e com a média (dentre as 10 rodadas da EE do experimento 1) da aptidão obtida pelo melhor indivíduo de cada geração da EE e um outro gráfico do mesmo tipo, mas para o experimento 2. Note que a EE obteve um aumento de aptidão menor com a mudança do valor de τ (que ocorre na geração 21) do que o AG, no experimento 1 (lembre que a aptidão deve ser minimizada neste problema). No experimento 2 as mudanças nas aptidões causadas pela mudança no valor de τ foram semelhantes.

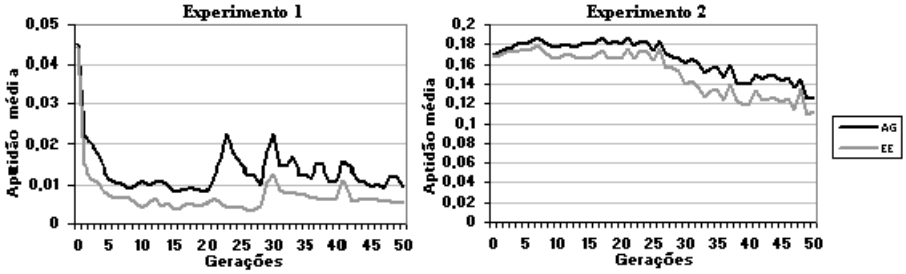

Figura 3: Média da aptidão do melhor indivíduo de cada geração das 10 rodadas

A partir de testes T-Student [Meyer, 1983] com nível de significância de 5% foi determinado que, no experimento 1, somente na geração 0 não houve diferença estatisticamente significativa entre as médias das aptidões geradas pelos dois algoritmos. Em todas as outras gerações pode-se afirmar que a otimização através de EE obteve aptidões médias estatisticamente menores que as obtidas pelo AG, *i.e.* as médias dos RMSEs das melhores EFuNNs das gerações de 1 a 50 foram estatisticamente menores para a EE do que para o AG.

Também foi determinado que, no experimento 2, nas gerações 0, 1, 2, 26, 48 e 50, não houve diferença estatisticamente significativa entre as médias das aptidões geradas pelos dois algoritmos de otimização. Em todas as outras gerações pode-se afirmar que a otimização através de EE obteve aptidões médias estatisticamente menores que as obtidas pelo AG, *i.e.* as médias dos RMSEs das melhores EFuNNs destas gerações foram estatisticamente menores para a EE do que para o AG.

A partir de testes F [Meyer, 1983] com nível de significância de 5% foi determinado que, no experimento 1, a variância das aptidões de ambos os algoritmos foi estatisticamente igual nas gerações de 0 a 6, de 8 a 10 e na geração 30. Em todas as outras, as variâncias obtidas pela otimização com EE foram menores que as obtidas pela com AG. Já no experimento 2, as variâncias das aptidões de ambos os algoritmos foram consideradas estatisticamente iguais em todas as gerações, exceto na geração 32, em que a variância da otimização com EE foi estatisticamente maior que a da otimização com AG.

A figura 4 possui um gráfico com o número de nodos de regras do melhor indivíduo de cada geração da rodada que obteve o melhor indivíduo na última geração do experimento 1 utilizando AG (rodada 2) e da rodada que obteve o melhor indivíduo na última geração do experimento 1 utilizando EE (rodada 4) e um outro gráfico do mesmo tipo, mas obtido a partir do experimento 2, utilizando a rodada 2 do AG e a 6 da EE.

Esses gráficos foram gerados com o intuito de analisar o número de nodos de regras de uma única rodada, ao invés da média do número de nodos de regras das rodadas.

Pode-se observar que o número de nodos de regras para estas rodadas da otimização através de AG e de EE no experimento 1 foi semelhante. Já no experimento 2, o número de nodos obtidos pelos melhores indivíduos da EE foi predominantemente maior do que o do AG. Nas últimas gerações do AG e da EE, o número de nodos gerados pela melhor EFuNN foi maior para o experimento 1 do que para o 2, mostrando que o uso de janelas de tempo de treinamento diferentes das de teste fez com que as redes geradas tivessem tamanho menor.

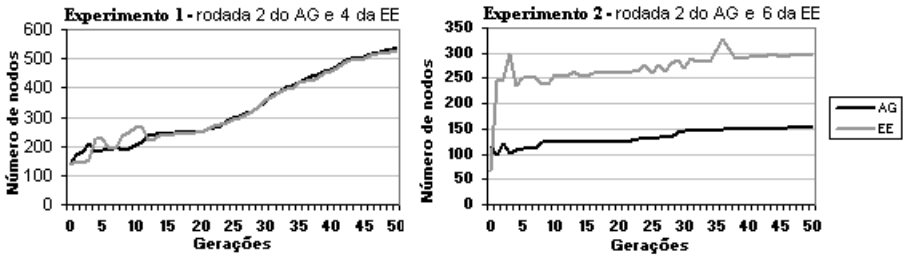

Figura 4: Número de nodos de regras do melhor indivíduo de cada geração

Deve-se ressaltar que, embora as aptidões médias dos melhores indivíduos gerados pela EE sejam, em sua grande maioria, melhores que as obtidas pelo AG, o número de avaliações de aptidão feito pela EE foi maior que o feito pelo AG. O tempo médio de execução de uma rodada para a EE, no experimento 1, foi cerca de 1,8 vezes maior que o de uma rodada para o AG. No experimento 2, o tempo de execução médio de uma rodada pela EE foi de cerca de 2,5 vezes maior que o do AG.

O fato de o tempo de execução da EE em relação ao AG no experimento 2 ter sido maior que no experimento 1 indica que, provavelmente, em média, a diferença entre o número de nodos de regras gerados pelas rodadas da EE e do AG no experimento 2 foi maior que esta diferença no experimento 1 e que, em média, o número de nodos da EE no experimento 2 foi maior que o do AG no experimento 2, já que o teste de uma rede com maior número de nodos leva mais tempo que o de uma com menos nodos. Também se deve observar que não foi utilizada uma função de aptidão multi-objetivo para penalizar uma rede de maior tamanho e este maior número de nodos foi resultante do processo de otimização da EE, que gerou parâmetros melhores adaptados que os do AG.

5. Conclusões

O método de otimização utilizado neste artigo faz as principais características dos ECOSs, como aprendizado adaptativo, ainda mais úteis para aplicações *on-line*.

A utilização desse método com EE fez com que, devido às suas características desejáveis para a otimização dos parâmetros de EFuNNs usadas para previsão de séries temporais caóticas, o RMSE das melhores EFuNNs geradas fosse mais baixo na utilização para previsão das séries Mackey-Glass do que a utilização do método com AG. Porém, as rodadas da EE levaram, em média, maior tempo de execução que as do AG.

Como trabalho futuro, pode ser feito um estudo maior sobre variações do AG e da EE para realizar otimização de parâmetros das EFuNNs. Uma modificação que pode ser feita é a inclusão de uma técnica para otimização com multi-objetivos, visando tanto a otimização dos parâmetros de forma a reduzir o RMSE das EFuNNs, quanto a produção de redes de tamanho reduzido. Podem também ser feitos experimentos para realizar a otimização de outros parâmetros que definem o aprendizado das EFuNNs e um maior estudo sobre o efeito da variação dos parâmetros do AG e da EE nos resultados obtidos.

Referências

Angeline, P. J., Saunders, G. M., and Pollack, J. P. (1994). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1):54–65.

Branke, J. (1995). Evolutionary algorithms in neural network design and training – A review. In Alander, J. T., editor, *Proc. of the First Nordic Workshop on Genetic Algorithms and their Applications, (1NWGA)*, volume 1, pages 145–163, Vaasa, Finnland.

Braun, H. and Weisbrod, J. (1993). Evolving feedforward neural networks. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, ANNGA'93*, pages 25–32, Innsbruck. Springer-Verlag.

Eiben, A. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Springer, Berlin.

Kasabov, N. (2001). Evolving fuzzy neural networks for supervised/unsupervised online, knowledge-based learning. *IEEE Transactions on Systems, Man and Cybernetics*, 31(6):902–918.

Kasabov, N. (2003). *Evolving Connectionist Systems*. Springer, Great Britain.

Kasabov, N., Song, Q., and Nishikawa, I. (2003). Evolutionary computation for dynamic parameter optimization of evolving connectionist systems for on-line prediction of time series with changing dynamics. In *IEEE Proceedings, IJCNN'2003*, volume 1, pages 438–443, Portland, Oregon.

Knuth, B. E. (1998). *The Art of Computer Programming - Seminumerical Algorithms*, volume 2. Reading Mass Addison-Wesley Pub. Co., Massachusetts, 3 edition.

Lee, C.-H. and Kim, J.-H. (1996). Evolutionary ordered neural network with a linked-list encoding scheme. In *Proc. 1996 IEEE Int. Conf. Evolutionary Computation, ICEC'96*, pages 665–669.

Mackey, M. C. and Glass, L. (1977). Oscillations and chaos in physiological control systems. *Science*, 197:287–289.

Meyer, P. L. (1983). *Testes de hipóteses, probabilidade - aplicações à estatística*. Livros Técnicos e Científicos Editora S.A., Rio de Janeiro.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

Watts, M. and Kasabov, N. (2001). Dynamic optimisation of evolving connectionist system training parameters by pseudo-evolution strategy. In *Proceedings of Congress on Evolutionary Computation 2001, CEC'2001*, volume 2, pages 1335–1342, Seoul.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447.

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

Yao, X. and Liu, Y. (1998). Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 28(3):417–425.