Estratégia Evolucionária e Algoritmo Genético para Otimização Dinâmica de Parâmetros de EFuNNs

Fernanda L. Minku e Teresa B. Ludermir
Universidade Federal de Pernambuco, Centro de Informática
Recife, PE 50732-970
Telefone: +55 81 2126-8430 Emails: {flm,tbl}@cin.ufpe.br

***Abstract*— Redes neurais difusas com estrutura adaptativa (EFuNNs) são normalmente utilizadas para modelar processos dinâmicos, que se desenvolvem e se modificam de maneira contínua no tempo. Este tipo de rede neural possui alguns parâmetros fixos, que não se alteram durante o aprendizado. O melhor conjunto de parâmetros fixos normalmente depende dos dados apresentados, de forma que se os dados mudam, o melhor conjunto de parâmetros também muda. Este artigo apresenta duas abordagens que utilizam computação evolucionária para realizar a otimização *on-line* desses parâmetros. Uma delas utiliza um algoritmo genético e a outra utiliza uma estratégia evolucionária. As redes neurais foram utilizadas para previsão de séries temporais caóticas Mackey-Glass. Um estudo comparativo entre estas abordagens e algumas de suas variações é apresentado.**

I. Introdução

Processos dinâmicos[1] 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 dinâmicos 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 a longo prazo [1].

Os sistemas conexionistas com estrutura adaptativa[2] (ECOSs) [1] formam um paradigma criado para facilitar a modelagem de processos dinâmicos. As redes neurais difusas com estrutura adaptativa[3] (EFuNNs) [2] formam um exemplo de uma classe de ECOSs que facilita a representação e a extração de conhecimento através de regras difusas.

Embora os ECOSs desenvolvam suas estruturas de acordo com os dados de entrada no tempo, eles ainda possuem alguns parâmetros fixos, que não são ajustados durante o aprendizado. Estes parâmetros definem o aprendizado. Neste artigo são apresentadas duas abordagens que utilizam computação evolucionária (CE) para realizar a otimização *on-line*, ou seja, durante o aprendizado, desses parâmetros. Uma delas utiliza um algoritmo genético (AG), a outra utiliza uma estratégia evolucionária (EE) e ambas foram baseadas no método de otimização introduzido por [3].

[1]Em inglês, foram chamados de *Evolving Processes*, por [1].
[2]Em inglês, *Evolving Connectionist Systems*.
[3]Em inglês, *Evolving Fuzzy Neural Networks*.

Foram realizados experimentos para comparar os resultados da otimização de alguns dos parâmetros que definem o aprendizado das EFuNNs utilizando as abordagens apresentadas e algumas de suas variações. Em especial, é comparado o resultado da otimização utilizando as abordagens apresentadas e uma variação que as torna semelhantes à otimização realizada em [3]. As EFuNNs foram utilizadas para previsão de séries temporais caóticas Mackey-Glass com mudança na dinâmica.

II. ECOSs e EFuNNs

Os ECOSs apresentados em [1] são sistemas compostos por uma ou mais redes neurais e com as seguintes características:

- Facilitam a modelagem de processos dinâmicos.
- Facilitam a representação e extração de conhecimento.
- Possuem 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 aprendidos anteriormente e sem a necessidade de realizar um novo treinamento com os dados antigos.
  - Rápido: possivelmente através de uma só passada pelos dados.
  - Local: particionam o espaço do problema localmente, permitindo adaptação rápida e tratamento de processos dinâmicos no tempo;
- Aprendem tanto como sistemas individuais, quanto como parte de uma população evolucionária de sistemas.
- Utilizam estruturas adaptativas e aprendizado construtivo.
- Evoluem em espaço aberto, não necessariamente de dimensões fixas.

As EFuNNs [2] constituem uma classe de ECOSs que une as características funcionais das redes neurais ao poder expressivo da lógica difusa. Elas possuem uma arquitetura composta por cinco camadas, como mostra a figura 1. A primeira camada representa o vetor de entrada, a segunda representa a quantificação difusa do vetor de entrada, a terceira representa associações entre o espaço de entradas difusas e o espaço de saídas difusas, a quarta representa a quantificação difusa do vetor de saída e a quinta representa o vetor de saída.

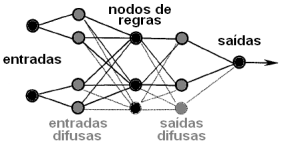

Fig. 1. Arquitetura das EFuNNs

O aprendizado ocorre na camada de nodos de regras. Cada nodo r_j desta camada é representado por dois vetores ($W1(r_j)$ e $W2(r_j)$) de pesos de conexões. O vetor $W1(r_j)$ representa as coordenadas do nodo r_j no espaço de entradas difusas e é ajustado através de aprendizado não supervisionado. O vetor $W2(r_j)$ representa as coordenadas do nodo de regra r_j no espaço de saídas difusas e é ajustado através de aprendizado supervisionado. As regras de aprendizado são as seguintes:

$$W1(r_j^{(t+1)}) = W1(r_j^{(t)}) + lr1(r_j^{(t)})(x_f - W1(r_j^{(t)}))$$
$$W2(r_j^{(t+1)}) = W2(r_j^{(t)}) + lr2(r_j^{(t)})(y_f - A2)A1(r_j^{(t)})$$

onde: x_f e y_f são os vetores de entradas difusas e saídas difusas, respectivamente; $lr1(r_j^{(t)})$ e $lr2(r_j^{(t)})$ são as taxas de aprendizado para os vetores de pesos $W1$ e $W2$ do nodo r_j, no tempo t; $A2$ é o vetor de ativação da camada de saídas difusas e $A1(r_j^{(t)})$ é o valor escalar de ativação do nodo r_j, no tempo t.

O algoritmo de aprendizado da EFuNN é brevemente explicado a seguir. É recomendada a leitura de [2] para maiores detalhes.

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 do campo receptivo inicial $R(r_j) = 1 - S$, 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 do campo receptivo $Rmax$ 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 $x2$ de tamanho N é definida por: $D(x1, x2) = \sum_{i=0}^{N-1} |x1_i - x2_i| / \sum_{i=0}^{N-1} |x1_i + x2_i|$
   b) Calcule as ativações $A1$ dos nodos de regras. Um exemplo de como $A1$ pode ser calculada é:
   $$A1(r_j) = 1 - D(W1(r_j)), x_f)$$
   c) Selecione o nodo de regras r_k que possui a menor distância $D(W1(r_k), x_f)$, ativação $A1(r_k) >= S(r_k)$ e que, no caso de acomodar o exemplo, não fique com raio R_k maior que o raio máximo permitido $Rmax$. No caso de aprendizado *m-de-n*, selecione *m* nodos ao invés de um nodo.
   d) Se não existe tal nodo (ou se não existirem *m* nodos com essas características), crie um novo nodo de regras para acomodar o exemplo (x_f, y_f).
   e) Senão
      i) Determine a ativação $A2$ da camada de saídas difusas e o erro de saída normalizado:
      $$Err = \sum_{i=0}^{Nout-1} |y_i - y_i'| / Nout$$
      onde y é a saída desejada, y' é a saída obtida e $Nout$ é o número de nodos da camada de saída.
      ii) Se $Err > E$, 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 $W2(r_k)$ e atualize $S(r_k)$, $R(r_k)$ e $AT(r_k)$. $AT(r_k)$ é a ativação total do nodo r_k, que pode ser, por exemplo, a soma das ativações $A1$ causadas por todos exemplos que o nodo r_k acomoda. 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) Incremente $Idade(r_i)$, para todos os nodos r_i.
   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 $T2$.

De acordo com o algoritmo 2.1, existem parâmetros que são ajustados durante o aprendizado (nodos de regras e seus pesos) e parâmetros que não mudam durante o aprendizado, mas o definem (número de funções de pertinência, *E, Nagg, OLD, Pr,* valor *m-de-n, Rmax, T1* e *T2*) [3].

III. Otimização on-line de parâmetros de ECOSs utilizando computação evolucionária

Através do uso de diferentes parâmetros do tipo que define o aprendizado, os ECOSs atingem diferentes desempenhos 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.

Um método que utiliza CE para otimizar parâmetros de ECOSs foi introduzido em [3]. Este método utiliza uma função de aptidão baseada somente na raiz quadrada do erro quadrático médio (RMSE) gerado pela previsão realizada pela EFuNN sobre um determinado conjunto de dados. Deste modo, se o intervalo de valores permitidos pelo algoritmo evolucionário para os parâmetros a serem otimizados não for

pequeno o suficiente, as EFuNNs geradas se tornam muito grandes em comparação com o número de exemplos de treinamento apresentados. Além disso, embora o método possa ser utilizado com outros algoritmos, [3] realizaram testes somente com AG, que nem sempre é a melhor técnica evolucionária para o problema em mãos.

Duas abordagens que utilizam CE para otimização *on-line* de parâmetros de EFuNNs foram implementadas no presente trabalho. Uma delas utiliza um AG, a outra utiliza uma EE e ambas foram baseadas em [3]. As subseções a seguir descrevem estas abordagens.

A. Otimização baseada em algoritmos genéticos

Esta abordagem utiliza AG [4] para realizar a otimização dos parâmetros que definem o aprendizado das EFuNNs. Assim como no AG utilizado em [3], 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 o 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 juntamente com o tamanho da EFuNN para determinar o valor de aptidão a ser minimizado (aptidão$=W_{rmse}RMSE+W_{tam}tamanho$). W_{rmse} e W_{tam} são pesos pré-definidos pelo usuário.
   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 por $P-D$ pontos anteriores e D pontos novos).

B. Otimização baseada em estratégia evolucionária

Esta abordagem utiliza uma EE [5] para realizar a otimização dos parâmetros que definem o aprendizado das EFuNNs. As EEs possuem as seguintes características, desejáveis para otimização de parâmetros de EFuNNs utilizadas com processos dinâmicos:

- São boas otimizadoras de parâmetros reais [5].
- Normalmente são utilizadas com auto-ajuste dos parâmetros de mutação [5]. Isto é desejado porque a superfície de aptidão não é conhecida e muda de acordo com as variações nas características dos dados de teste.
- Possuem um baixo tempo de *takeover* [5]. Esta característica permite uma rápida adaptação às características do momento.

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 [5]. Para gerar os números aleatórios da distribuição gaussiana, foram utilizados números aleatórios de uma distribuição normal obtidos através do método Polar [6].

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. Esta representação é mais adequada para otimização de parâmetros numéricos ordinais (como os dos ECOSs) que a representação binária, pois, na binária, pequenas modificações no genótipo podem causar grandes modificações no fenótipo [5].

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 [5].

A seleção de pais realizada é aleatória, com probabilidades iguais para todos os indivíduos da população, 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 [5].

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.
4) Determine a aptidão de cada EFuNN da população da mesma forma que no algoritmo 3.1.
5) Repita até os dados de treinamento/teste acabarem:

a) Selecione aleatoriamente N pais, utilizando probabilidades iguais para todos os indivíduos.
b) Aplique mutação com os desvios-padrões auto-ajustáveis e cruzamento com probabilidade Pc, gerando um filho para cada par de pais.
c) Desloque a janela de tempo de treinamento (a nova janela será composta por $P - D$ pontos anteriores e D pontos novos). Desloque do mesmo modo a janela de tempo de teste, se utilizada.
d) Realize o aprendizado de cada filho.
e) Determine a aptidão de cada filho.
f) Realize seleção de sobreviventes (μ, λ), ou seja, selecione deterministicamente os μ (μ é o tamanho da população) melhores indivíduos dentre todos os λ filhos gerados. O número de filhos gerados é $\lambda = N/2$ e $\lambda > \mu$.

IV. Experimentos

Os algoritmos 3.1 e 3.2 foram utilizados para realizar otimização *on-line* de alguns dos parâmetros que definem o aprendizado das EFuNNs. As EFuNNs foram utilizadas para previsão de séries temporais caóticas Mackey-Glass com mudança na dinâmica, como foi feito nos experimentos realizados por [3].

A. Série temporal caótica Mackey-Glass

A série temporal caótica Mackey-Glass [7] é uma série *benchmark* 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 trabalho foram utilizados os valores $a = 0,2$; $b = 0,1$; $x(0) = 1,2$ e $x(t) = 0$ para $t < 0$, conforme foi utilizado em [3]. O valor τ utilizado sofreu 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 na dinâmica da série não seja drástica, ela ainda assim pode causar falhas na previsão da série com o parâmetro modificado [1].

Foram realizados dois tipos de experimentos. No primeiro, 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 segundo experimento, 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 com $\tau = 17$ e de 1800 a 2399 com $\tau = 19$.

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

TABELA I
Execuções realizadas

Ab	Pd	Tes = Trei	W_{tam}	Conj Int	Num
AG	S	S	0	2	10
			0.0005	2	10
		N	0	2	10
			0.002	2	10
	N	S	0	1	10
			0	2	1
			0.0005	2	10
		N	0	1	10
			0	2	1
			0.002	2	10
EE	S	S	0	2	10
			0.0005	2	10
		N	0	2	10
			0.002	2	10
	N	S	0	1	10
			0	2	1
			0.0005	2	10
		N	0	1	10
			0	2	1
			0.002	2	10

B. Parâmetros utilizados e objetivos dos experimentos

Os parâmetros da EFuNN otimizados foram *E, Mrad, m-de-n* e, em algumas execuções, *OLD*.

Os experimentos foram realizados com o intuito de comparar os resultados das técnicas considerando e não considerando os tamanhos das EFuNNs na função de aptidão, com e sem poda de nodos de regras, com o RMSE obtido a partir da previsão feita sobre a mesma janela de tempo utilizada para treinamento e com o RMSE obtido a partir da previsão feita sobre uma janela de tempo diferente da utilizada para treinamento.

Dois conjuntos de valores permitidos para os parâmetros a serem otimizados foram utilizados. O conjunto 1 é igual ao que foi utilizado por [3], numa tentativa de gerar resultados similares para os experimentos feitos utilizando $W_{tam} = 0$. Neste conjunto, o intervalo permitido para *m-de-n* é $[1; 8] \in Z$, o intervalo para *Mrad* é $[0,75; 0,95] \in R$ e o intervalo para *E* é $[0,05; 0,25] \in R$. Os experimentos feitos por [3] não realizaram a otimização do parâmetro de poda *OLD* e não utilizaram poda. O intervalo de valores permitidos para *OLD* utilizado no presente trabalho foi $[10; 266] \in Z$. O segundo conjunto de parâmetros utilizado foi *m-de-n* ($[1; 15] \in Z$), *Mrad* ($[0,01; 0,8] \in R$), *E* ($[0,01; 0,6] \in R$) e *OLD* ($[10, 266] \in Z$). Este conjunto possui os valores normalmente permitidos pelas EFuNNs para realizar o aprendizado.

Os experimentos sem poda não realizaram a otimização do parâmetro *OLD* e utilizaram $Pr = 0$. Os experimentos com poda utilizaram $Pr = 1$. O parâmetro *Nagg* foi fixado em 0 (não foi realizada agregação), os parâmetros *T1* e *T2* foram fixados em 0 (não foi realizada extração de regras) e o parâmetro *S* foi fixado em 0,9. O tamanho da população utilizado foi de 12 indivíduos, o peso do RMSE no cálculo da aptidão foi $W_{rmse} = 10$ e o tamanho da janela de tempo foi $P = 200$, com 90% de sobreposição entre janelas consecutivas ($D = 20$). Note que a modificação do parâmetro τ sempre ocorre na geração 21.

O AG utilizou $Pc = 0,7$ e $Pm = 0,01$. O genótipo para

o conjunto 1 de intervalos de valores permitidos foi formado por 3 bits para o valor *m-de-n*, 6 para *Mrad*, 6 para *E* e 8 para *OLD*, quando este parâmetro foi utilizado. Para o conjunto 2, foi utilizado um genótipo composto por 4 bits para o valor *m-de-n*, 10 para *Mrad*, 9 para *E* e 8 para *OLD*, quando este parâmetro foi utilizado.

A EE utilizou $Pc = 0,7$ e $N = 96$. O genótipo foi composto por 6 variáveis reais quando não foi otimizado o parâmetro *OLD* e por 8 quando foi otimizado o parâmetro *OLD*.

A tabela I resume as execuções realizadas. Nesta tabela, Ab significa abordagem, Pd significa poda, Tes=Trei significa que a janela de tempo de teste é igual à janela de tempo de treinamento, W_{tam} indica o peso utilizado na função de aptidão, Conj Int significa conjunto de intervalos de valores permitidos, Num significa número de execuções, S significa sim e N significa não.

C. Resultados

Esta seção explica as principais análises das execuções mostradas na tabela I. Em alguns casos os resultados serão considerados somente após a geração 15, para não serem afetados pela rápida queda do RMSE das primeiras gerações.

As execuções sem poda, com $W_{tam} = 0$ e com o conjunto 2 de intervalos de valores permitidos geraram EFuNNs muito grandes[4] em comparação com o número de exemplos de treinamento (1200). Devido a isso, elas foram finalizadas na geração 28. No final desta geração, o número de nodos da EFuNN de melhor aptidão foi 1214 na execução do AG com janela de tempo de teste igual à janela de tempo de treinamento , foi 3373 na execução do AG com janela de tempo de teste diferente da de treinamento , foi 5239 na execução da EE com janela de tempo de teste igual à de treinamento e foi 2275 na execução da EE com janela de tempo de teste diferente da de treinamento. As redes da geração 28 foram também as maiores redes, dentre as EFuNNs de melhor aptidão, de todas as gerações até a geração 28. As redes se tornaram muito grandes porque a função de aptidão não foi penalizada pelo tamanho das redes. Utilizando o conjunto 1 de intervalos de valores permitidos, de maneira similar às execuções realizadas em [3], as EFuNNs não se tornaram muito grandes em comparação com o número de exemplos de treinamento, mas este conjunto possui intervalos pequenos em comparação com o conjunto 2. As próximas análises utilizarão sempre o conjunto 2.

Nas execuções sem poda e com $W_{tam} \neq 0$, as maiores EFuNNs, dentre as EFuNNs de melhor aptidão, de todas as gerações, obtiveram menos que 838 nodos de regras nas 10 execuções do AG com janela de tempo de teste igual à de treinamento, menos que 201 nas 10 execuções do AG com janela de tempo de teste diferente da de treinamento, menos que 911 nas 10 execuções da EE com janela de tempo de teste igual à de treinamento e menos que 196 nas 10 execuções da EE com janela de tempo de teste diferente da de treinamento.

Comparando a EE sem poda e $W_{tam} = 0.0005$, a EE com poda e $W_{tam} = 0.0005$ e a EE com poda e $W_{tam} = 0$, para janela de tempo de teste igual à de treinamento, a EE sem poda obteve as maiores médias dos números de nodos de regras e a com poda e $W_{tam} = 0.0005$ obteve as menores, como mostra a figura 2(a). A EE sem poda obteve as maiores médias de RMSEs e a EE com poda e $W_{tam} = 0$ obteve as menores, na maioria das gerações após a geração 15, como mostra a figura 2(a). Note que as execuções com poda e $W_{tam} = 0.0005$ geraram redes menores que as execuções com poda e $W_{tam} = 0$, conforme esperado, já que $W_{tam} = 0.0005$ penaliza o tamanho das redes. O AG obteve comportamento similar. Testes estatísticos T-Student [8] executados para o AG e para a EE, entre as médias dos seus números de nodos de regras e as médias dos seus RMSEs, com nível de significância (α) de 5%, confirmaram essas análises.

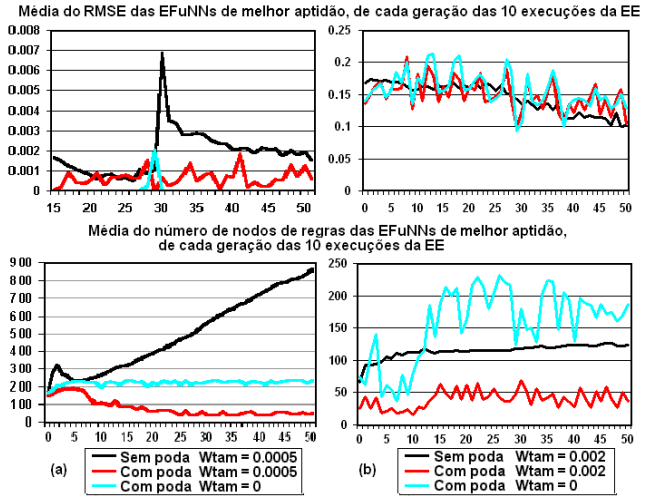

Fig. 2. Média do número de nodos de regras e do RMSE para EE com janela de tempo de teste (a) igual e (b) diferente da janela de tempo de treinamento

Comparando EE sem poda e $W_{tam} = 0.002$, EE com poda e $W_{tam} = 0.002$ e EE com poda e $W_{tam} = 0$, para janela de tempo de treinamento diferente da de teste, pode ser observado que com poda as médias dos números de nodos de regras e as dos RMSEs variam mais que nas execuções sem poda, conforme a figura 2(b). Também pode ser observado que, para médias de RMSEs semelhantes, a EE com poda e $W_{tam} = 0.002$ gerou EFuNNs com médias dos números de nodos de regras menores que EE com poda e $W_{tam} = 0$. Isto era esperado, já que $W_{tam} = 0.002$ penaliza o tamanho das EFuNNs. O AG obteve comportamento semelhante.

Note que, para essas execuções da EE, o uso de janela de tempo de teste diferente da de treinamento, quando $W_{tam} = 0$ ou quando não foi realizada poda, obteve o efeito de diminuir a média dos tamanhos das EFuNNs em comparação com o uso de janelas de tempo de teste iguais às de treinamento, na maioria do tempo. Efeito similar ocorreu com os AGs. Esta análise foi confirmada com testes T-Student com nível de significância de 5% para a EE e para o AG. Os RMSEs das EFuNNs resultantes das execuções com janela de tempo de teste igual à de treinamento e das EFuNNs resultantes das execuções com janela de tempo de teste diferente da de trei-

[4]Os termos tamanho, EFuNN grande e EFuNN pequena serão utilizados em referência ao número de nodos de regras.

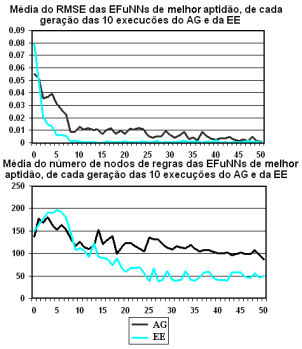

Fig. 3. Média do RMSE e do número de nodos de regras do AG e da EE com poda, $W_{tam} = 0.0005$ e janela de tempo de teste igual à de treinamento

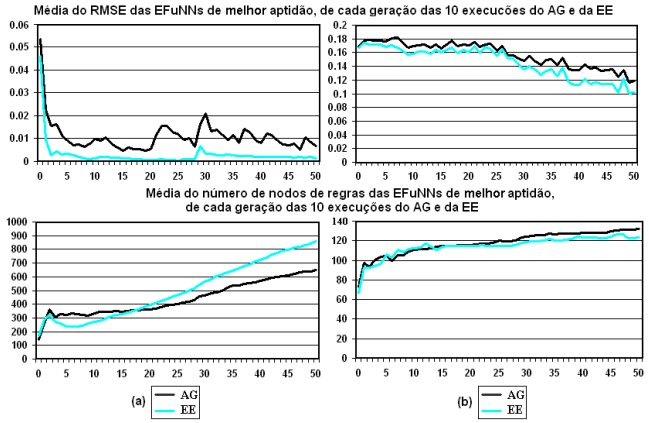

Fig. 4. Média do RMSE e do número de nodos de regras do AG e da EE (a) sem poda, com $W_{tam} = 0.0005$ e janela de tempo de teste igual à de treinamento e (b) sem poda, com $W_{tam} = 0.002$ e janela de tempo de teste diferente da de treinamento

namento não podem ser comparados porque foram calculados sobre diferentes dados de teste. Os valores maiores obtidos pelas execuções com janelas de tempo distintas ocorrem como uma conseqüência direta do uso de um conjunto de teste diferente do conjunto de treinamento.

Comparando AG e EE com poda, $W_{tam} = 0.0005$ e janela de tempo de teste igual à de treinamento, pode ser observado que as médias dos RMSEs e dos números de nodos de regras da EE foram menores que as do AG, na maior parte do tempo. A figura 3 ilustra esses efeitos, que foram confirmados através de testes T-Student com $\alpha = 5\%$.

Esta análise não foi feita para execuções com poda e janela de tempo de teste diferente da janela de tempo de treinamento porque as variações das médias dos RMSEs e dos nodos de regras foram muito grandes, como foi explicado anteriormente.

Comparando AG e EE sem poda, com $W_{tam} = 0.0005$ e janela de tempo de teste igual à de treinamento, pode ser observado que as médias dos RMSEs da EE foram menores que as médias do AG e as médias dos números de nodos de regra da EE foram maiores que as do AG na maior parte do tempo. A figura 4(a) ilustra esses efeitos, que foram confirmados através de testes T-Student com $\alpha = 5\%$.

Comparando AG e EE sem poda, com $W_{tam} = 0.002$ e janela de tempo de teste diferente da de treinamento, pode ser observado que as médias dos RMSEs e dos números de nodos de regras da EE foram menores que as do AG na maior parte do tempo. A figura 4(b) ilustra esses efeitos, que foram confirmados através de testes T-Student com $\alpha = 5\%$.

V. Conclusão

As abordagens de otimização utilizadas neste artigo fazem as principais características dos ECOSs, como aprendizado adaptativo, ainda mais úteis para aplicações que utilizam dados que mudam suas características com o tempo.

As abordagens utilizadas com $W_{tam} \neq 0$ geraram EFuNNs com tamanho reduzido em comparação com $W_{tam} = 0$, mesmo quando não foi utilizada poda, permitindo o uso de maiores intervalos de valores permitidos para os parâmetros a serem otimizados.

As abordagens utilizadas com poda e janela de tempo de teste diferente da de treinamento possuem uma maior variação das médias dos RMSEs e dos números de nodos de regras com o passar das gerações do que quando não foi utilizada poda.

O uso de janela de tempo de teste diferente da de treinamento age, em geral, como um redutor do número de nodos de regras das EFuNNs quando não é utilizada poda ou quando o tamanho da rede não é considerado no cálculo da aptidão. Quando uma janela de tempo de teste diferente não é disponível, o uso de poda é uma boa maneira para reduzir o número de nodos de regras das EFuNNs, resultando em redes de menores médias de RMSEs do que quando não é realizada poda.

O RMSE das EFuNNs geradas utilizando EE foi menor que o RMSE das EFuNNs geradas utilizando AG, nas execuções sem poda. Efeito similar foi observado nas execuções com poda e janela de tempo de teste igual à de treinamento.

Trabalhos futuros incluem a investigação do efeito da variação dos pesos da função de aptidão, a criação de um método para variar no tempo os pesos da função de aptidão e o uso de técnicas multi-objetivo.

Referências

[1] N. Kasabov, *Evolving Connectionist Systems*. Great Britain: Springer, 2003.

[2] N.Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised on-line, knowledge-based learning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 31, no. 6, pp. 902–918, 2001.

[3] N. Kasabov, Q. Song, and I. Nishikawa, "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*, vol. 1, Portland, Oregon, 2003, pp. 438–443.

[4] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc., 1989.

[5] A. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Berlin: Springer, 2003.

[6] B. E. Knuth, *The Art of Computer Programming - Seminumerical Algorithms*, 3rd ed. Massachusetts: Reading Mass Addison-Wesley Pub. Co., 1998, vol. 2.

[7] M. C. Mackey and L. Glass, "Oscillations and chaos in physiological control systems," *Science*, vol. 197, pp. 287–289, 1977.

[8] I. H. Witten and E. Frank, *Data Mining: Pratical Machine Learning Tools and Techniques with Java Implementations*. San Francisco: Morgan Kaufmann Publishers, 2000.