Otimização de Reservoir Computing com PSO – Parâmetros Globais, Arquitetura e Pesos

Anderson Tenório Sergio [1], Teresa B. Ludermir
Centro de Informática
Universidade Federal de Pernambuco
Recife, Brasil
ats3@cin.ufpe.br, tbl@cin.ufpe.br
[1] FITec – Fundação para Inovações Tecnológicas

Aida Araújo Ferreira
Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco
Recife, Brasil
aidaaf@gmail.com

***Resumo*—*Reservoir Computing* (RC) é um paradigma de Redes Neurais Artificiais com aplicações importantes no mundo real. RC utiliza arquitetura similar às Redes Neurais Recorrentes sem a necessidade de treinar os pesos da camada intermediária (*reservoir*), além de utilizar uma função de regressão linear simples no treinamento da camada de saída. Entretanto, sua utilização pode ser computacionalmente onerosa e diversos parâmetros influenciam sua eficiência, fazendo com que seja necessário pesquisar alternativas para aumentar sua capacidade. Este trabalho tem o objetivo de utilizar o PSO e duas de suas extensões na tarefa de otimizar os parâmetros globais, arquitetura e pesos do *reservoir* de RC, na previsão de séries temporais. Os resultados alcançados mostraram que a otimização de *Reservoir Computing* com PSO, bem como com as suas extensões selecionadas, apresentaram desempenho satisfatório para todas as bases de dados estudadas.**

Palavras-chave*—*Reservoir Computing; PSO; otimização; previsão de séries temporais.

I. INTRODUÇÃO

Reservoir Computing é um paradigma de Redes Neurais Artificiais com aplicações importantes [1] [2] [3]. RC utiliza arquitetura similar a Redes Neurais Recorrentes (RNR) para processamento temporal sem a dificuldade de treinar a camada intermediária da rede. *Reservoir Computing* foi introduzido independentemente como *Liquid State Machines* (LSM) [4] e *Echo State Networks* (ESN) [5]. De forma geral, o conceito de RC é baseado na construção de uma RNR de maneira randômica (essa camada é denominada *reservoir*), sem alteração dos pesos. Após essa fase, uma função de regressão linear é utilizada para treinar a saída do sistema. Schrauwen et al [6] mostram que a transformação dinâmica não-linear oferecida pelo *reservoir* é suficiente para que a camada de saída, denominada *readout*, consiga extrair os sinais de saída utilizando um mapeamento linear simples.

Assim como ocorre nas redes neurais convencionais, *Reservoir Computing* sofre de algumas desvantagens. Por se tratar de uma abordagem de redes recorrentes, sua utilização pode ser computacionalmente onerosa, mesmo sem a fase de treinamento da camada de *reservoir*. Diversos parâmetros influenciam a eficiência do RC, como, por exemplo, o número de nodos utilizados e o tipo de função de ativação. Definir esses parâmetros sem a experiência na resolução de determinado problema é difícil. Lukosevicius e Jaeger [7] indicam que é improvável que a geração aleatória dos pesos e o treinamento da camada de saída com uma função de regressão linear simples seja a solução ideal para computar RC.

O PSO (*Particle Swarm Optimization*) [8] é um algoritmo de otimização que possui algumas vantagens sobre outras técnicas de busca global. O algoritmo é baseado no comportamento social de revoadas dos pássaros: uma população de soluções é mantida e cada indivíduo busca melhorar seu desempenho baseado na melhor experiência pessoal e na melhor experiência do grupo. Quando comparado aos Algoritmos Genéticos, por exemplo, o PSO possui a vantagem de ter implementação mais simples e, em alguns casos, convergência relativamente mais rápida e custo computacional menor [9] [10].

Este trabalho tem o objetivo de investigar a utilização do PSO na tarefa de otimizar os parâmetros globais, arquitetura e pesos do *reservoir* de RC, no problema de previsão de séries temporais. Além do PSO padrão, também será estudada a utilização de duas de suas extensões (EPUS-PSO e APSO). Para validação do método proposto, serão utilizadas sete séries temporais clássicas e três séries de velocidade média horária dos ventos na região Nordeste do Brasil. As séries de *benchmark* serão utilizadas para comparação com outros trabalhos na literatura, enquanto que as séries de ventos verificarão a adequação do método em um problema de aplicação prática, com relevância econômica.

A seguir, a estrutura do artigo. A Seção II e III apresentam respectivamente *Reservoir Computing* e PSO. A Seção IV discute a otimização de RC, enquanto que a Seção V descreve o método proposto. A Seção VI apresenta as simulações numéricas e, finalmente, a Seção VII descreve as conclusões e as propostas de trabalhos futuros.

II. *RESERVOIR COMPUTING*

Reservoir Computing é uma arquitetura de Redes Neurais Artificiais que foi desenvolvida independentemente como *Liquid State Machines* [4] e *Echo State Networks* [5]. Em comum, ambos os modelos têm a característica de utilizar o poder computacional das Redes Neurais Recorrentes sem as dificuldades relacionadas ao treinamento dessa arquitetura.

De um modo geral, a rede é composta por uma rede recorrente com um número relativamente grande de unidades de processamento, denominada *reservoir*. O *reservoir* recebe os sinais de entrada e envia-os para um circuito menor, chamado *readout*. Enquanto os pesos do *reservoir* são definidos aleatoriamente no início do processo e mantidos inalterados, o *readout* é utilizado para treinar a saída da rede através de uma função que faça regressão linear dos sinais provenientes da camada intermediária. A Fig. 1 mostra o diagrama de uma *Echo State Network* simples.

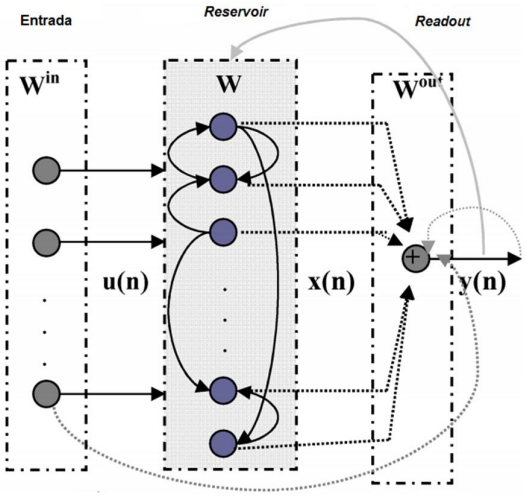

Fig. 1. Arquitetura de uma *Echo State Network*

Como explicitado na Fig. 1, a camada denominada *reservoir* recebe como sinal os valores advindos da camada de entrada, bem como opcionalmente os sinais provenientes da conexão de *feedback* e de bias. O *reservoir*, com certo número de unidades de processamento, é projetado de forma que possua conexões recorrentes. Os pesos dessas conexões são randômicos e não mudam. A camada denominada *readout* faz um mapeamento linear simples da saída do *reservoir* utilizando, por exemplo, pseudo-inversa [11]. Ainda na Fig. 1, as linhas pontilhadas representam conexões que podem ser treinadas e as linhas sombreadas indicam conexões opcionais.

Ao invés de tentar obter uma transformação particular através do ajuste dos pesos, RC utiliza um número maior de neurônios (em comparação às redes convencionais) para alcançar um conjunto diverso de transformações a partir dos sinais de entrada [5]. Geralmente tais transformações não são as desejadas, mas elas podem ser combinadas para alcançá-las. Esta ação é justamente realizada pela camada de *readout*. O cálculo realizado é simples, já que o *readout* não apresenta retroalimentação. É importante notar que um mesmo *reservoir* pode ser utilizado para calcular múltiplas transformações em paralelo, desde que elas tenham diferentes *readouts*.

III. PSO

O PSO é uma técnica de otimização global baseada em uma população de soluções. O algoritmo é baseado no comportamento social de revoadas dos pássaros, onde um indivíduo imita as ações do melhor do grupo (ou mais indicado). O processo tem início com a definição da população de soluções. Cada indivíduo, por vezes chamado de "partícula", é uma solução possível. Cada partícula possui uma posição e uma velocidade, e o processo de atualização é baseado na melhor experiência pessoal e na melhor experiência do grupo. O PSO foi criado por Kennedy e Eberhart em 1995 [8].

Trabalhos como o de Van den Bergh [11], Clerc et al [12] e Trelea [13] analisam matematicamente a convergência do PSO. Tais trabalhos tiveram como resultado orientações sobre quais parâmetros afetam a convergência, divergência ou oscilação do algoritmo, e esses estudos deram origem a diversas variações do PSO original.

Uma dessas variações é o EPUS-PSO (*Efficient Population Utilization Strategy for PSO*) [14]. Com o objetivo de resolver problemas de alta complexidade com busca eficiente e rápida convergência, o EPUS-PSO é baseado em três conceitos: gerenciamento de população (o tamanho da população é variável); compartilhamento de solução (as partículas podem obter conhecimento de outros indivíduos) e o chamado *Search Ranging Sharing* (SRS), que perturba as soluções e aumenta o número de possibilidades na busca.

O *Adaptive Particle Swarm Optimization* (APSO) [15] também foi desenvolvido com o objetivo de aumentar a eficiência da busca e da velocidade de convergência do algoritmo original do PSO. Basicamente, o APSO consiste de dois passos principais. Primeiro, o algoritmo identifica ao longo de sua execução, através da avaliação da distribuição da população e função de aptidão de cada partícula, em qual dos quatro estados evolucionários a geração de soluções se encontra: exploração, explotação, convergência ou salto. Segundo, uma estratégia de aprendizado elitista é seguida, ativada quando o estado evolucionário é classificado como convergência. Para classificar o estado evolucionário da geração, o APSO utiliza a abordagem denominada ESE (*Evolutionary State Estimation*), baseada no comportamento de busca e distribuição da população de soluções do PSO.

IV. OTIMIZAÇÃO DE *RESERVOIR COMPUTING*

Como visto anteriormente, as arquiteturas primordiais de *Reservoir Computing* (LSM e ESN) trabalham com uma rede neural recorrente com pesos fixos, gerados aleatoriamente. Entretanto, Lukosevicius e Jaeger [11] indicam que é improvável que a geração aleatória dos pesos e o treinamento da camada de saída com uma função de regressão linear simples seja a solução ideal para computar RC, sendo necessárias pesquisas de alternativas para geração do *reservoir* e treinamento do *readout*.

Como alternativa para a geração da camada de *reservoir*, é possível criar os pesos da camada intermediária segundo adaptações não supervisionadas ou mesmo a partir de um pré-treinamento supervisionado. A adaptação não supervisionada pressupõe otimizar alguma medida definida no *reservoir* para uma dada entrada, não levando em consideração a saída desejada [9] [16]. Em contrapartida, o pré-treinamento supervisionado leva também em consideração a saída desejada [17] [18].

Além da adaptação do *reservoir*, outras maneiras para otimizar *Reservoir Computing* podem ser consideradas.

Ferreira e Ludermir [19] apresentaram um método para otimizar a escolha dos parâmetros globais utilizando Algoritmos Genéticos, aplicando o sistema a um problema de previsão de séries temporais com aplicação prática. A otimização levou até 22,22% do tempo necessário para realizar uma busca exaustiva dos parâmetros e alcançar resultados semelhantes. Na busca exaustiva, todos os parâmetros são combinados sem utilizar qualquer método de otimização.

Em 2011, Ferreira [20] desenvolveu um método para encontrar o melhor *reservoir* aplicado à tarefa de prever séries temporais, denominado RCDESIGN. O método busca simultaneamente os melhores valores dos parâmetros globais, da topologia da rede e dos pesos, através da combinação de um algoritmo evolucionário com *Reservoir Computing*. Outros dois métodos de otimização foram implementados para realizar a comparação dos resultados. A chamada Busca RS tenta otimizar o tamanho do reservoir, o raio espectral e a densidade de conexão. A Busca TR busca simultaneamente o raio espectral e a topologia da rede. Juntamente com o RCDESIGN, a Busca TR não considera a aproximação de RC com Sistemas Lineares (não reescala a matriz de pesos do *reservoir* pelo raio espectral e permite criação de sistemas com as conexões opcionais de RC). O RCDESIGN apresentou resultados satisfatórios em todas as bases de dados estudadas, sendo melhor que os outros métodos utilizados para comparação. Todas as abordagens foram também aplicadas à previsão da velocidade dos ventos, que é uma tarefa importante para geração de energia eólica.

O PSO é um importante candidato para realizar otimização de *Reservoir Computing*, dado suas vantagens sobre Algoritmos Genéticos [9] [10]. Além disso, as extensões desse algoritmo podem aumentar ainda mais a eficiência da tarefa, visto que tais modificações foram desenvolvidas com esse propósito. Sergio e Ludermir utilizaram o PSO e duas de suas extensões (EPUS-PSO e APSO) para otimizar os parâmetros globais do RC, concluindo que nas cinco séries temporais utilizadas o método proposto reduziu o número de ciclos de treinamento necessários para treinar o sistema [21]. Os parâmetros utilizados na otimização foram o número de nodos no *reservoir*, função de ativação dos neurônios do *reservoir*, raio espectral da matriz de pesos do *reservoir* e presença ou ausência das conexões opcionais.

Em resumo, na tarefa de previsão de séries temporais, otimizar os parâmetros globais e os pesos de *Reservoir Computing* pode trazer resultados satisfatórios, como visto com o RCDESIGN [20]. Além disso, substituir Algoritmos Genéticos pelo PSO pode implicar em aumento de desempenho e computação mais rápida [9] [10]. Resultados preliminares com o PSO indicam sua eficiência [21]. Este trabalho busca dar continuidade à pesquisa iniciada em [21]: baseando-se no RCDESIGN, o artigo propõe um método para otimizar os parâmetros globais, a topologia e os pesos de *Reservoir Computing*, utilizando o PSO e duas de suas extensões.

V. Otimização de *Reservoir Computing* com PSO

O método utilizado será apresentado a seguir. Será descrito como as soluções foram representadas, de que forma a função de aptidão foi calculada, os parâmetros envolvidos e o método experimental. A otimização é adaptada do método proposto por Ferreira, o RCDESIGN [20]. Ao invés de Algoritmos Genéticos, o PSO foi utilizado como algoritmo de otimização. O ESN foi utilizado como método de *Reservoir Computing*.

A. Representação das Soluções

No PSO, existem os conceitos de partícula (uma solução individual) e população de partículas (um conjunto de soluções ou enxame). Ao utilizar esse algoritmo, é necessário definir a representação dessa solução.

A representação utilizada por Ferreira [20] foi adaptada para a inclusão do PSO. Cada partícula é representada por um vetor s^i. A notação s_j^i denota a dimensão j da partícula s^i. A seguir, cada uma dessas dimensões é descrita.

s_1^i (η) – Número de nodos no *reservoir*, inteiro entre 50 e 200. Esse parâmetro define o tamanho das matrizes de pesos.

s_2^i – Conexão entre a camada de entrada e a camada de saída. Número real entre 0 e 1. Se maior que 0.5, há conexão, se menor, não há.

s_3^i – Conexão entre o bias e a camada de saída. Número real entre 0 e 1. Se maior que 0.5, há conexão, se menor, não há.

s_4^i – Conexão de realimentação na camada de saída. Número real entre 0 e 1. Se maior que 0.5, há conexão, se menor, não há.

s_5^i – Conexão entre o bias e a camada de *reservoir*. Número real entre 0 e 1. Se maior que 0.5, há conexão, se menor, não há.

s_6^i – Conexão entre a camada de saída e a camada de *reservoir*. Número real entre 0 e 1. Se maior que 0.5, há conexão, se menor, não há.

s_7^i – Função de ativação dos neurônios. Se 1, tangente hiperbólica, se 2, sigmóide.

s_8^i – Função de treinamento do *readout*. Se 1, pseudo-inversa [9], se 2, *ridge-regress* [22].

s_9^i – *Leak rate*. Número real entre 0,1 e 1. *Leak rate* é um parâmetro que pode ser adicionado aos neurônios, possibilitando o ajuste da dinâmica do RC.

s_{10}^i – Parâmetro de regularização das funções de treinamento. Número real entre 10^{-8} e 10^{-1}. O parâmetro de regularização corresponde a um ruído que pode ser adicionado às respostas do *reservoir*.

s_{11}^i ... $s_{(\eta^2+3\eta+10)}^i$ – Pesos das camadas de *reservoir* (W), entrada (W^{in}), bias (W^{bias}) e realimentação (W^{back}). Número real entre 0,1 e 1.

O tamanho do vetor s^i é variável. Isso acontece porque as últimas posições dependem do número de nodos no *reservoir*, definido pela dimensão j = 1. O intervalo [50; 200] foi definido empiricamente, com a intenção de criar *reservoirs* grande o suficiente para computar os dados e com tamanho viável para realizar as simulações. Se η = 50, o vetor s^i tem 2660 posições. Caso η = 200, o vetor s^i tem tamanho máximo de 40610.

B. Função de Aptidão

Na otimização de *Reservoir Computing* com PSO, foi utilizada como medida de desempenho a função de aptidão do RCDESIGN [21]. Essa função de aptidão é baseada no erro médio quadrático dos sistemas criados (MSE, do inglês *Mean Square Error*). Devido ao fenômeno de *overfitting*, comum em arquiteturas de Redes Neurais Artificiais, ela leva em consideração não somente o MSE do sistema criado no conjunto de treinamento, mas também no de validação. A equação (1) mostra a função de fitness utilizada.

$$f = \overline{MSE}_{Treinamento} + \left\| \overline{MSE}_{Treinamento} - \overline{MSE}_{Validação} \right\| \quad (1)$$

C. Parâmetros

Os parâmetros do PSO neste trabalho foram definidos segundo processo empírico, de tentativa e erro. Diversos conjuntos de parâmetros foram testados aleatoriamente, de acordo com o bom senso do projetista. Os testes também levaram em consideração valores de parâmetros que são consagrados na literatura, como o termo de *momentum* inicial. Foi selecionada a configuração de parâmetros que apresentou melhor desempenho nos testes iniciais. A seguir, os parâmetros utilizados. Tamanho do enxame: 50; número de iterações: 20; termo de *momentum* inicial: 0.9; coeficientes de aceleração global e local: 2.05.

No caso específico do algoritmo do EPUS-PSO, o tamanho do enxame e o número de iterações são variáveis. Nesse caso, os valores iniciais para esses parâmetros foram 60 e 20, respectivamente. Outro ponto importante é a amplitude dos valores dos pesos. Foi detectado empiricamente que em algumas bases de dados os resultados eram melhores com um intervalo menor do que [-1; 1]. Contudo, esses continuaram a serem os valores máximos para todas as bases.

D. Método Experimental

Foram utilizadas sete séries temporais clássicas utilizadas como *benchmark* na literatura e três bases com aplicação prática, contendo dados da velocidade média horária dos ventos na região Nordeste do Brasil. As séries clássicas utilizadas foram a *Narma* ordem 10 (NAR10), *Narma* ordem 30 (NAR30), *Mackey-Glass* caos médio (MGS17), *Mackey-Glass* caos moderado (MGS30), *Multiple Sinewave Oscillator* (MSO), Série Natural do Brilho da Estrela (STAR) e Série Financeira *Dow Jones Industrial Average* (DJIA).

A geração eólica é, dentre outras variáveis, uma função da velocidade dos ventos. Prever adequadamente esses valores pode reduzir sensivelmente as dificuldades de operação em uma fonte eólica e também em fontes tradicionais de energia, já que é possível integrar modelos eficientes de previsões de geração de energia eólica com outros sistemas elétricos de geração [23].

As três séries utilizadas no trabalho de Ferreira [20] foram selecionadas, obtidas no site do projeto SONDA (Sistema de Organização de Dados Ambientais) [24]. O projeto SONDA mantém bases de dados gratuitas dos recursos de energia solar e eólica do Brasil. As bases contêm observações das cidades de Belo Jardim – PE (BJD), São João do Cariri – PB (SJC) e Triunfo – PE (TRI), com 13176, 17520 e 21936 padrões, respectivamente.

No intuito de comparar os resultados obtidos entre si e com outros métodos na literatura, o desempenho da otimização proposta neste trabalho foi calculado segundo diversos índices de erros de previsão. São eles: erro médio quadrático (MSE), raiz quadrada do erro médio quadrático normalizado (NRMSE, do inglês *Normalized Root Mean Square Error*) e erro médio quadrático normalizado (NMSE, do inglês *Normalized Mean Square Error*).

VI. Simulações Numéricas

A tabela I apresenta o NRMSE resultante dos três métodos propostos na fase de teste, juntamente com o método inspirador, o RCDESIGN [20]. Devido às características aleatórias das simulações numéricas, e para realização de teste de hipóteses, as medidas de desempenho estão representadas pelas médias de 30 inicializações de cada base de dados. Os valores entre parêntesis são os desvios-padrão. Em destaque, os melhores desempenhos entre os métodos propostos por esse trabalho.

TABLE I. NRMSE no Teste, 30 Inicializações.

Bases de Dados	**Métodos Propostos**			**Método Inspirador**
	PSO	***EPUS-PSO***	***APSO***	***RCDESIGN***
NAR10	0,0499853761 (0,0141)	0,0562621742 (0,0301)	0,0503082103 (0,0242)	0,08462155 (0,0354)
NAR30	0,1193474167 (0,0353)	0,1257601918 (0,0371)	0,0948630262 (0,0208)	0,20424126 (0,0469)
MGS17	0,0001119798 (0,00001)	0,0001270548 (0,00001)	0,0001099572 (0,00001)	0,00050628 (0,0001)
MGS30	0,0003829910 (0,00002)	0,0003960918 (0,00002)	0,0003778040 (0,00002)	0,00099067 (0,0002)
MSO	0,0000000039 ($8*10^{-10}$)	0,0000000045 (1*10-9)	0,0000000034 (7*10-10)	0,00000120 (0,0000)
STAR	0,0501224563 (0,00001)	0,0506885487 (0,0010)	0,0494933321 (0,0001)	0,08555901 (0,2088)
DJIA	0,3780419257 (0,0008)	0,3787736114 (0,0006)	0,3783935519 (0,0006)	0,23938190 (0,0156)
BJD	0,5513885351 (0,0006)	0,5519262169 (0,0006)	0,5512637857 (0,0007)	0,49230377 (0,0025)
SCR	0,4387079294 (0,0011)	0,4397166635 (0,0012)	0,4384478650 (0,0011)	0,40928555 (0,0030)
TRI	0,3723567172 (0,0075)	0,3726854490 (0,0039)	0,3707453994 (0,0045)	0,43600947 (0,0016)

A tabela II mostra a comparação entre os métodos desenvolvidos de acordo com o teste t de *Student* não pareado ao nível 5% de significância (95% de confiança), em relação ao NRMSE. Nessa tabela, o sinal "=" indica que a hipótese nula não foi rejeitada (a diferença entre as médias dos erros não é estatisticamente relevante) e os modelos apresentam o mesmo desempenho. O sinal "<" indica que a hipótese nula foi rejeitada e que os modelos para comparação (segundo modelo na comparação) tem desempenho inferior ao de seleção (primeiro modelo na comparação). Por fim, o sinal ">" indica que a hipótese nula foi rejeitada e que os modelos para comparação tem desempenho superior ao de seleção. O NRMSE foi escolhido para realização do teste de hipóteses por possuir valores com menos casas decimais, facilitando os cálculos. A análise dos próximos três parágrafos será feita em relação aos métodos propostos por esse trabalho.

De acordo com as tabelas I a II, é possível observar que o EPUS-PSO não superou o desempenho do PSO e do APSO em nenhuma base de dados. O APSO superou o PSO em três bases, sendo que nas outras sete alcançou resultados estatisticamente similares de acordo com o teste de hipóteses realizado. Em relação ao EPUS-PSO, o APSO teve desempenho superior em oito bases de dados, enquanto o PSO superou o EPUS-PSO em sete.

TABLE II. COMPARAÇÃO ESTATÍSTICA NO TESTE – NRMSE.

Comparação	Bases de Dados									
	NAR 10	*NAR 30*	*MGS 17*	*MGS 30*	*MSO*	*STAR*	*DJIA*	*BJD*	*SCR*	*TRI*
PSO x EPUSPSO	=	=	<	<	<	<	<	<	<	=
PSO x APSO	=	>	=	=	>	>	=	=	=	=
EPUSPSO x APSO	=	>	>	>	>	>	>	>	>	=
APSO x RCDESIGN	<	<	<	<	<	=	>	>	>	<

Utilizando os erros de previsão na fase de teste como medida de desempenho para comparação das abordagens propostas, pode-se concluir que o APSO alcançou os melhores resultados. É possível que a estratégia ESE (*Evolutionary State Estimation*), utilizada no APSO, tenha se adequado melhor às características dos dados estudados. No ESE, o algoritmo do PSO tenta identificar em quais dos quatro diferentes estados evolucionários (exploração, explotação, convergência e salto) sua execução se encontra, para poder assim calcular as variáveis necessárias de diferentes maneiras (termo de *momentum*, coeficientes de aceleração e os limites desses coeficientes). A natureza das séries temporais, onde a ordem dos dados é relevante, diferentemente dos modelos de regressão linear, pode ter oferecido um padrão melhor aplicável ao ESE.

Uma explicação para o desempenho inferior do EPUS-PSO seria o fato deste algoritmo possuir um problema da mesma natureza cujo trabalho em questão tenta solucionar. No EPUS-PSO, assim como na configuração de experimentos com redes neurais artificiais, é necessário ao projetista definir o valor inicial de certos parâmetros sem qualquer conhecimento prévio do problema, como por exemplo, o tamanho inicial do enxame.

A última linha da tabela II compara os NRMSE do RCDESIGN [20] com o APSO do método proposto. O APSO foi selecionado para realizar essa comparação por ter alcançando o melhor desempenho de acordo com o erro de previsão entre os métodos desenvolvidos. A otimização de RC com APSO (bem como com as outras abordagens propostas) possui configuração bastante similar ao RCDESIGN. Por exemplo, ambos atuam sobre o mesmo espaço de busca e utilizam a mesma função de aptidão. A diferença fica por conta do algoritmo de otimização utilizado: PSO ou Algoritmos Genéticos. O teste de hipóteses apresentado na tabela II mostra que, em relação ao NRMSE, o APSO foi melhor do que o RCDESIGN em seis das bases de dados investigadas. Obteve desempenho similar em uma base (STAR) e foi superado pelo RCDESIGN em três.

Na execução do algoritmo de otimização de *Reservoir Computing* proposto, seja com a utilização do PSO ou com uma de suas extensões, a tarefa com maior custo computacional é o cálculo da função de aptidão. É nesse ponto que o algoritmo realiza um ciclo completo de treinamento do *reservoir*, para calcular o MSE na fase de treinamento e validação. Logo, o número de vezes que a função de aptidão é calculada está intimamente ligado com o custo computacional do algoritmo como um todo. Uma solução mais eficiente segundo esse ponto de vista busca realizar menos ciclos de treinamento possíveis.

No caso do EPUS-PSO o número de ciclos de treinamento do *reservoir* é variável, já que o tamanho do enxame também o é. Os resultados mostraram que a utilização do EPUS-PSO obteve sempre os melhores resultados segundo o critério de ciclos de treinamento necessários para alcançar os valores ótimos, incluindo o RCDESIGN. Quando em determinada aplicação o tempo de execução é mais crítico do que o desempenho do algoritmo propriamente dito, o EPUS-PSO é mais indicado. O desempenho do EPUS-PSO pode ser ainda melhorado investigando-se uma melhor configuração de parâmetros e valores iniciais para o tamanho inicial do enxame e o número de iterações.

Embora a configuração das simulações seja diferente, é possível comparar o desempenho do APSO com outros trabalhos que utilizaram alguma das bases de dados estudadas, com as restrições decorrentes da não reprodução de seus experimentos. Essa comparação é realizada pelas tabelas III, IV e V.

VII. CONCLUSÃO

Este trabalho apresentou um método para otimizar simultaneamente os parâmetros globais, a topologia e os pesos de *Reservoir Computing* utilizando o PSO. A otimização foi realizada com o algoritmo padrão do PSO e duas de suas extensões presentes na literatura, o EPUS-PSO e o APSO. Dentre os métodos desenvolvidos, o APSO foi o algoritmo que apresentou os melhores resultados segundo os erros de previsão. O EPUS-PSO obteve os melhores resultados quando o critério de ciclos de treinamento necessários para alcançar os valores ótimos foi levado em consideração.

Os resultados alcançados foram comparados com outros trabalhos na literatura. De acordo com os erros de previsão, a otimização com o APSO mostrou-se melhor do que as outras otimizações na maioria das bases de dados investigadas. Entretanto, é importante observar que as simulações numéricas dos trabalhos utilizados para comparação não foram reproduzidos. Tal restrição deve ser levada em consideração. Os resultados obtidos, de acordo com as abordagens utilizadas, fazem com que o método proposto mostre-se como uma solução importante na literatura no que tange a tarefa de otimizar RC.

Propostas de trabalhos futuros: utilizar outras extensões do PSO, utilizar outros algoritmos de otimização e aplicar o método proposto em outras bases de dados de aplicação prática.

TABLE III. COMPARAÇÃO DO MSE.

Métodos	Bases de Dados			
	NAR10	*NAR30*	*MGS17*	*MGS30*
APSO	0,000037	0,000110	0,0000000006	0,0000000096
Sergio e Ludermir [21]	0,000162	0,000131	0,0000394980	0,0000255480

TABLE IV. COMPARAÇÃO DO NMSE.

Métodos	Bases de Dados	
	NAR10	*MGS17*
APSO	0,0031	0,0000000124
Jaeger [25]	0,0081	-
Steil [26]	0,1420	0,0340

TABLE V. COMPARAÇÃO DO NRMSE.

Métodos	Bases de Dados			
	NAR30	*MGS17*	*MGS30*	*MSO*
APSO	0,0948	0,00010	0,00037	0,0000000034
Jaeger [5]	-	0,00012	0,032	-
Wyffels et al [27]	-	0,0065	0,0065	-
Jaeger e Haas [28]	-	0,000063	-	-
Schmidhuber et al [29]	-	-	-	0,0103
Schrauwen et al [30]	0,4600	-	-	-

REFERÊNCIAS

[1] K. Vandoorne, M. Fiers, D. Verstraeten, B. Schrauwen, J. Dambre, P. Bienstman. "Photonic Reservoir Computing: A New Approach to Optical Information Processing". 12th International Conference on Transparent Optical Networks (ICTON), Munich, German, 2010.

[2] P. Buteneers, D. Verstraeten, P. van Mierlo, T.Wyckhuys, D.Stroobandt, R. Raedt, H. Hallez, B. Schrauwen. "Automatic detection of epileptic seizures on the intra-cranial electroencephalogram of rats using reservoir computing". Artificial Intelligence in Medicine, 53, pp. 215-223, 2001.

[3] A. Smerieri, F.Duport, Y.Paquot, M.Haelterman, B.Schrauwen, M.Massar. "Towards Fully Analog Hardware Reservoir Computing For Speech Recognition". International Conference of Numerical Analysis and Applied Mathematics (ICNAAM), 1479, pp.1892-1895, 2012.

[4] W. Maass, T. Natschlager, H. Markram. "Real-time computing without stable states: A new framework for neural computation based on perturbations". Neural Computation, 14(11), pp. 2531–2560, 2002.

[5] H. Jaeger. "The echo state approach to analyzing and training recurrent neural networks". Tech. Rep. GMD 148, German National Resource Center for Information Technology, 2001.

[6] B. Schrauwen, J. Defour, D. Verstraeten, J. Van Campenhout. "The introduction of time-scales in reservoir computing, applied to isolated digits recognition". LNCS, 4668(1), pp. 471–479, 2007.

[7] M. Lukosevicius, H. Jaeger. "Reservoir computing approaches to recurrent neural network training". Computer Science Review, 3(3), pp. 127–149, 2009.

[8] J. Kennedy, R. Eberhart. "Particle swarm Intelligence". Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942-1948, 1995.

[9] R. Hassan, B. Cohanim, O. De Weck, G. Venter. "A Comparison Of Particle Swarm Optimization And The Genetic Algorithm". 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, pp. 1-13, 2005.

[10] S. Panda, N. P. Padhy. "Comparison of Particle Swarm Optimization and Genetic Algorithm for TCSC-based Controller Design". International Journal of Computer Science & Engineering, 1(1), pp. 41, 2007.

[11] F. Van den Bergh. "An Analysis of Particle Swarm Optimizers". PhD thesis, University of Pretoria, Faculty of Natural and Agricultural Science, 2001.

[12] M. Clerc, J. Kennedy. "The particle swarm - explosion, stability, and convergence in a multidimensional complex space". IEEE Transactions on Evolutionary Computation, 6(1), pp. 58–73, 2002.

[13] I. C. Trelea. "The Particle Swarm Optimization Algorithm: convergence analysis and parameter selection". Information Processing Letters, 85, pp. 317–325, 2003.

[14] S. Hsieh, T. Sun, C. Liu, S. J. Tsai. "Efficient Population Utilization Strategy for Particle Swarm Optimizer". IEEE Transactions, Man and Cybernetics, 30, pp. 444-456, 2009.

[15] Z-H. Zhan, J. Zhang, Y. Li, H. Chung, H. "Adaptive Particle Swarm Optimization". IEEE Transactions, Man and Cybernetics, 39, pp. 1362-1381, 2009.

[16] R. Legenstein, W. Maass. "New Directions in Statistical Signal Processing: From Systems to Brain", chapter What makes a dynamical system computationally powerful?, pp. 127–154. MIT Press, 2007.

[17] K. Ishii, T. van der Zant, V. Becanovic, P. Ploger. "Identification of motion with echo state network". Ocean 2004 Conference, 3, pp. 1205–1210, 2004.

[18] K. Bush, B. Tsendjav. "Improving the richness of echo state features using next ascent local search". Artificial Neural Networks In Engineering Conference, pp. 227–232, 2005.

[19] A. A. Ferreira, T. B. Ludermir. "Genetic algorithm for reservoir computing optimization". International Joint Conference on Neural Networks, pp. 811-815, 2009.

[20] A. Ferreira. "Um Método para Design e Treinamento de Reservoir Computing Aplicado à Previsão de Séries Temporais". Tese de Doutorado, Universidade Federal de Pernambuco, CIn, Ciência da Computação, 2011.

[21] A. Sergio, T. B. Ludermir. "PSO for reservoir computing optimization". ICANN 2012, Part I, LNCS 7552, pp. 685-692, 2012.

[22] C. M. Bishop. "Pattern recognition and machine learning". In Information Science and Statistics. Springer, 2006.

[23] R. R. B. Aquino, M. A. Carvalho Junior, M. M S. Lira, O. Nóbrega Neto, G. J. Almeida, S. N. N. Tiburcio. "Recurrent neural networks solving a real large scale mid-term scheduling for power plants". In International Joint Conference on Neural Networks - IJCNN 2010, pp. 3439–3444, Barcelona, 2010.

[24] SONDA, Sistema de Organização Nacional de Dados Ambientais. http://sonda.cptec.inpe.br/, Janeiro 2013.

[25] H. Jaeger. "Adaptive nonlinear system identification with echo state networks". Advances in Neural Information Processing Systems, 15, pp. 593–600, 2003.

[26] J. J. Steil. "Backpropagation–decorrelation: Online recurrent learning witho(n) complexity". In International Joint Conference on Neural Networks, IJCNN 2004, 2, pp. 843–848, 2004.

[27] F. Wyffels, B. Schrauwen, D. Verstraeten, D. Stroobandt. "Band-pass reservoir computing". In International Joint Conference on Neural Networks 2008, pp. 3203–3208, 2008.

[28] H. Jaeger, H. Haas. "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless telecommunication". Science, 308, pp. 78–80, 2004.

[29] J. Schmidhuber, D. Wierstra, M. E. Gagliolo, F. Gomez. "Training recurrent networks by evolino". Neural Computation, 19(3), pp. 757–779, 2007.

[30] B. Schrauwen, M. Wardermann, D. Verstraeten, J. J. Steil, D. Stroobandt. "Improving reservoirs using intrinsic plasticity". Neurocomputing, 71, pp. 1159–1171, 2008.