"Um Método para Design e Treinamento de Reservoir Computing Aplicado à Previsão de Séries Temporais"

Por

Aida Araújo Ferreira

Tese de Doutorado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, 28 de Março de 2011

Universidade Federal de Pernambuco
Centro de Informática

Aida Araújo Ferreira

UM MÉTODO PARA DESIGN E TREINAMENTO DE RESERVOIR COMPUTING APLICADO À PREVISÃO DE SÉRIES TEMPORAIS

Tese apresentada à Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientadora: *Profa. Dra. Teresa Bernarda Ludermir*

RECIFE
28 de Março de 2011

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Ferreira, Aida Araújo
Um método para design e treinamento de reservoir computing aplicado à previsão de séries temporais / Aida Araújo Ferreira - Recife: O Autor, 2011.
xviii, 126 folhas : il., fig., tab.

Orientador: Teresa Bernarda Ludermir.
Tese (doutorado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2011.

Inclui bibliografia e apêndice.

1. Inteligência artificial. 2. Redes neurais. 3. Otimização. 4. Previsão de séries temporais. I. Ludermir, Teresa Bernarda (orientadora). II. Título.

006.3 CDD (22. ed.) MEI2011 – 089

Tese de Doutorado apresentada por **Aida Araújo Ferreira** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Um Método para Design e Treinamento de Reservoir Computing Aplicado à Previsão de Séries Temporais"**, orientada pela **Profa. Teresa Bernarda Ludermir** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Marcilio Carlos Pereira Souto
Centro de Informática / UFPE

Prof. Ricardo Bastos Cavalcante Prudêncio
Centro de Informática / UFPE

Prof. Cleber Zanchettin
Centro de Informática / UFPE

Prof. Mêuser Jorge Silva Valença
Escola Politécnica de Pernambuco / UPE

Prof. André Carlos Ponce de Leon Ferreira de Carvalho
Instituto de Ciências Matemáticas e de Computação / USP

Visto e permitida a impressão.
Recife, 28 de março de 2011.

Prof. NELSON SOUTO ROSA
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Agradecimentos

Em primeiro lugar, agradeço a Deus, que sempre esteve presente iluminando todos os momentos da minha vida.

Agradeço à minha amada mãe Maria José (*in memorian*) que sonhou comigo este sonho.

Agradeço a meu amando pai José Frazão (*in memorian*).

Agradeço a meu querido marido Ronaldo que é o meu grande incentivador.

Agradeço às minhas filhas Amanda, Ana Clara e Alice que souberam compreender os muitos momentos em que estive ausente e que são a razão da minha vida.

Agradeço às minhas irmãs Ermelinda, Idalina e Fátima pois elas sempre foram para mim modelos de dedicação e seriedade com os estudos.

Quero agradecer especialmente a Profa. Dra. Teresa Bernarda Ludermir pela orientação e pela atenção.

Eu dedico esta tese à minha querida Mainha, a meu amado Ronaldo e às minhas filhas Amanda, Aninha e Alice.

...

I have a dream
a song to sing
to help me cope
with anything
if you see the wonder
of a fairy tale
you can take the future
even if you fail
I believe in angels
something good in
everything I see
I believe in angels
when I know the time
is right for me ...

—BENNY ANDERSSON E BJÖRN ULVAEUS (ABBA)

Resumo

Reservoir Computing é um tipo de rede neural recorrente que permite uma modelagem caixa-preta para sistemas dinâmicos (não-lineares). Em contraste com outras abordagens de redes neurais recorrentes, com *Reservoir Computing* não existe a necessidade de treinamento dos pesos da camada de entrada e nem dos pesos internos da rede (*reservoir*), apenas os pesos da camada de saída (*readout*) são treinados. No entanto, é necessário ajustar os parâmetros e a topologia da rede para a criação de um *reservoir ótimo* que seja adequado a uma determinada aplicação. Neste trabalho, foi criado um método, chamado RCDESIGN, para encontrar o melhor *reservoir* aplicado à tarefa de previsão de séries temporais. O método desenvolvido combina um algoritmo evolucionário com *Reservoir Computing* e busca simultaneamente pelos melhores valores dos parâmetros, da topologia da rede e dos pesos, sem reescalar a matriz de pesos do *reservoir* pelo raio espectral. A ideia do ajuste do raio espectral dentro de um círculo unitário no plano complexo, vem da teoria dos sistemas lineares que mostra claramente que a estabilidade é necessária para a obtenção de respostas úteis em sistemas lineares. Contudo, este argumento não se aplica necessariamente aos sistemas não-lineares, que é o caso de *Reservoir Computing*. O método criado considera também o *Reservoir Computing* em toda a sua não–linearidade, pois permite a utilização de todas as suas possíveis conexões, em vez de usar apenas as conexões obrigatórias. Os resultados obtidos com o método proposto são comparados com dois métodos diferentes. O primeiro, chamado neste trabalho de *Busca RS*, utiliza algoritmo genético para otimizar os principais parâmetros de *Reservoir Computing*, que são: tamanho do *reservoir*, raio espectral e densidade de conexão. O segundo, chamado neste trabalho de *Busca TR*, utiliza algoritmo genético para otimizar a topologia e pesos de *Reservoir Computing* baseado no raio espectral. Foram utilizadas sete séries clássicas para realizar a validação acadêmica da aplicação do método proposto à tarefa de previsão de séries temporais. Um estudo de caso foi desenvolvido para verificar a adequação do método proposto ao problema de previsão da velocidade horária dos ventos na região nordeste do Brasil. A geração eólica é uma das fontes renováveis de energia com o menor custo de produção e com a maior quantidade de recursos disponíveis. Dessa forma, a utilização de modelos eficientes de previsão da velocidade dos ventos e da geração eólica pode reduzir as dificuldades de operação de um sistema elétrico composto por fontes tradicionais de energia e pela fonte eólica.

Palavras-chave: Reservoir Computing, Redes Neurais Recorrentes, Otimização, Previsão de Séries Temporais, Energia Eólica

Abstract

Reservoir Computing is a type of recurrent neural network that allows for the black box modeling of (nonlinear) dynamical systems. In contrast to other recurrent neural network approaches, Reservoir Computing does not train the input and internal weights of the network, only the readout is trained. However, it is necessary to adjust parameters and topology to create a "good"reservoir for a given application. This work created a method, called RCDESIGN, to find the best Reservoir applied to the task of time series forecasting. The developed method combines an evolutionary algorithm with Reservoir Computing and simultaneously looks for the best values of parameters, topology and weight matrices without rescaling the reservoir matrix by the spectral radius. The idea of rescaling the weight matrices to adjust the spectral radius within the unit circle in the complex plane comes from the linear system theory and shows clearly that the stability is necessary to obtain useful responses from linear systems. However, this argument does not necessarily apply to nonlinear systems, which is the case of Reservoir Computing. The method also considers Reservoir Computing in all its non–linearity, because it allows the possibility for all its connections, instead of using only required connections. The results obtained with the proposed method are compared with two different methods. The first, called in this work by *RS Searching*, uses genetic algorithm to optimize the major parameters that determine the reservoir dynamics and performance: size, spectral radius and connection density. The second, called in this work by *TR Searching*, uses genetic algorithm to optimize the topology and weights based on spectral radius. We used seven sets to validate the academic application of the method to the task of forecasting time series. A case study was developed to examine the adequacy of the proposed method to the problem of hourly wind speed in the northeast region of Brazil. The wind energy is a renewable energy with the lowest cost producer and the largest amount of resources available. Thus, the use of efficient models to forecast wind speed and wind power can reduce the difficulties of operating a power system composed of traditional energy sources and wind power.

Keywords: Reservoir Computing, Recurrent Neural Network, Optimization, Time Series Forecasting, Wind Power

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

APRL Atiya-Parlos Recurrent Learning

BPDC Backpropagation-DeCorrelation

EP Elementos de Processamento

ESN Echo State Network

EVS Eigenvalue Spread

GL Generalization Loss - Perda da capacidade de generalização

LIF Leaky Integrate and Fire

LSM Liquid State Machine

MAPE Mean Absolute Percentage Error

MGS Mackey-Glass Time Series

MLP Multi-layer Perceptron

MSE Mean Squared Error

NMSE Normalized Mean Squared Error

MSO Multiple Sinewave Oscillator

NARMA Nonlinear Autoregressive Moving Average

NMSE Normalized Mean Squared Error

PSO Particle Swarm Optimization

RC Reservoir Computing

RCDESIGN RC Design e treinamento

RLS Recursive Least Squares

RNA Rede Neural Artificial

RNR Rede Neural Recorrente

1 Introdução

Teoricamente, as redes neurais recorrentes (RNRs) são ferramentas muito poderosas para resolução de tarefas temporais complexas de aprendizagem de máquina. No entanto, vários fatores ainda dificultam a implantação em maior escala das RNRs em aplicações práticas. Existem vários algoritmos de aprendizagem e a maioria deles sofre com taxas de convergência lenta, limitando assim sua aplicabilidade [VSDS07]. Em 2001, uma nova abordagem para o design e treinamento de RNR foi proposta de forma independente sob o nome de *Liquid State Machines* (LSMs) [MNM02] e de *Echo State Networks* (ESNs) [Jae01]. Esta abordagem, que teve predecessores na neurociência computacional [Dom95] e ramificações posteriores como o *Backpropagation-Decorrelation* [Ste04], agora é frequentemente citada como *Reservoir Computing* (RC) [LJ09]. O conceito básico é construir aleatoriamente uma RNR e deixar os pesos desta rede inalterados. Uma função de regressão linear é treinada separadamente sobre as respostas da rede para os sinais de entrada. A ideia básica é que a camada intermediária (*reservoir*) da RNR, construída aleatoriamente, sofre uma transformação não-linear complexa da dinâmica dos sinais de entrada que permite que a camada de saída (*readout*) extraia a saída desejada utilizando um mapeamento linear simples. *Reservoir Computing* oferece uma solução para o uso do poder de processamento temporal das RNRs sem apresentar o problema de taxa de convergência lenta que é comum aos algoritmos de treinamento de RNRs [SDVC07a].

O desempenho de *Reservoir Computing* depende da seleção apropriada dos parâmetros globais e da topologia da rede. Algumas métricas foram descritas na literatura, entretanto não existe ainda uma clara indicação de que valores são *ótimos* para cada tarefa. Além disso, os valores dessas métricas ainda não foram relacionados à dinâmica real do sistema. Alguns dos guias mais genéricos para produzir *boas* redes do tipo *Reservoir Computing* foram apresentados nos trabalhos que introduziram ESNs [Jae01] e

[Jae02b]. Porém, de acordo com Lukosevicius e Jaeger[LJ09], os métodos clássicos ainda não exploraram todo o potencial das RNRs pois eles usam uma camada intermediária aleatória, que é improvável que seja *ótima*, e uma camada linear de saída (*readout*), que é bastante limitada pela qualidade dos sinais que ela combina; e finalmente, a separação entre a camada intermediária e a camada de saída fornece uma plataforma muito boa para experimentar muitos tipos de método de adaptação dos pesos da rede e analisar como eles podem realmente melhorar o desempenho das RNRs geradas aleatoriamente.

Em muitas áreas como previsão de séries temporais e reconhecimento de dígitos isolados, a aplicação de técnicas de *Reservoir Computing* são o *estado-da-arte* [SVH10], [LJ09].

Várias tarefas de previsão são realizadas diariamente pelas concessionárias de energia elétrica, como por exemplo previsão de carga, previsão de vazão, previsão da geração eólica entre outras. Essas previsões podem ser em curto, médio ou logo prazo e os dados das previsões são utilizados como entrada para vários estudos, tais como planejamento, operação e comercialização de energia elétrica. Tradicionalmente, as técnicas de previsão utilizam métodos estatísticos de análise de séries temporais, que incluem a regressão linear, amortecimento exponencial e *Box Jenkins*, porém as técnicas de inteligência artificial, como as redes neurais artificiais, têm sido usadas como métodos alternativos e apresentam resultados promissores [ANL+07].

Esta tese desenvolve diferentes estudos de caso com o método proposto, que utiliza redes neurais artificiais, para previsão de séries temporais. Para realizar a validação acadêmica da aplicação do método proposto à tarefa de previsão de séries temporais foram utilizadas séries clássicas e para verificar a adequação do método proposto a um problema real, foram escolhidas séries de velocidade média horária dos ventos na região nordeste do Brasil.

A previsão da velocidade dos ventos é muito importante para previsão da geração de energia eólica. O governo brasileiro passou a demonstrar um grande interesse na geração de energia eólica após o racionamento de energia de 2001, criando o PROINFA (Programa de Incentivo a utilização de Fontes Alternativas de Energia Elétrica) [end10b] com os objetivos de diversificar a matriz energética brasileira e de garantir maior confiabilidade e segurança de abastecimento. Em relação ao abastecimento de energia elétrica do país, o PROINFA será um instrumento de complementaridade energética sazonal à energia hidráulica, que, segundo o relatório PEN de 2010 da ONS (Operador Nacional do Sistema Elétrico), é atualmente responsável por cerca de 65% da geração do país [end10a]. Na região Nordeste, a energia eólica servirá como complemento ao abastecimento hidráulico,

já que o período de chuvas é inverso ao de ventos. O mesmo ocorrerá com a biomassa nas regiões Sul e Sudeste, onde a colheita de safras propícias à geração de energia elétrica (cana–de–açúcar e arroz, por exemplo) ocorre em período diferente do chuvoso [end10b].

A geração eólica é uma das fontes renováveis de energia com o menor custo de produção e com a maior quantidade de recursos disponíveis e está suficientemente madura para representar uma maior contribuição na matriz energética [KHM+06]. A integração em larga escala da geração de energia eólica com qualquer sistema de geração de energia seja ele interconectado ou autônomo, impõe muitas dificuldades para operação do sistema. Isto acontece devido às características de intermitência e variabilidade da geração eólica. O caráter aleatório dos ventos dificulta o planejamento, a operação e a estabilidade do sistema elétrico, pois as redes elétricas são compostas por fontes diversificadas de energia, as tradicionais as quais se destacam hidroelétrica e as térmicas, e as alternativas em que a fonte eólica é uma das principais. Como a geração eólica é função basicamente da velocidade dos ventos [AJL+10], a integração de modelos eficientes de previsão da velocidade dos ventos ao controle e operação do sistema elétrico pode reduzir as dificuldades de operação de um sistema composto por fontes tradicionais de energia e pela fonte eólica.

1.1 Objetivo

O objetivo deste trabalho é a proposição, implementação e a análise de um método para o design e treinamento simultâneo de *Reservoir Computing* (RC) aplicado à tarefa de previsão de séries temporais, especificamente ao problema de previsão da velocidade média dos ventos em curto prazo na região Nordeste do Brasil.

Embora a otimização de RC seja um desafio, por outro lado, verificar o desempenho de uma rede resultante é relativamente barato. Isto faz com que os métodos evolucionários, para o pre–treinamento de RC sejam uma estratégia natural [LJ09] para a busca da melhor rede para uma determinada tarefa. Algumas abordagens evolucionárias de otimização de RC foram apresentadas em [IvdZBP04] e [BT05], entretanto elas usavam a mesma ideia de separação da topologia e dos pesos de RC com o objetivo de reduzir o espaço de busca e também da busca por um raio espectral menor que 1 para garantir a propriedade de *echo state*.

O método proposto neste trabalho, chamado RCDESIGN (*Reservoir Computing* Design e Treinamento), utiliza uma estratégia evolucionária para pesquisar simultaneamente pelos melhores valores dos parâmetros globais, pela melhor topologia e pelos pesos da

camada intermediária, sem a limitação apresentada em trabalhos anteriores da redução do espaço de busca [LJ09] e sem reescalar a matriz de pesos da camada intermediária pelo raio espectral. A string (ou vetor) genética usada pelo método proposto é muito maior do que as que foram utilizadas em outras abordagens evolucionárias para otimização de RC e mesmo assim o desempenho do método foi bastante satisfatório, pois o uso do mesmo evitou a necessidade de se utilizar operações matemáticas computacionalmente caras para criação de RC.

Neste trabalho utilizamos sete séries clássicas e três séries de velocidade média horária dos ventos na região Nordeste do Brasil. A ideia é utilizar as séries clássicas para validar a aplicação do método proposto à tarefa de previsão de séries temporais e utilizar as séries de velocidade média horária dos ventos para verificar a adequação do método proposto ao problema de previsão da velocidade média horária dos ventos em curto prazo na região Nordeste do Brasil.

1.2 Contribuições da Tese

Em especial, este trabalho propõe-se a trazer os seguintes avanços e contribuições:

(i) Proposição, implementação e a análise de um método para o design e treinamento simultâneo de *Reservoir Computing* (RC) aplicado à tarefa de previsão de séries temporais, especificamente ao problema de previsão da velocidade média dos ventos em curto prazo na região Nordeste do Brasil. Comparação dos resultados obtidos pelo método RCDESIGN com o resultado de outros modelos de previsão de séries temporais com *Reservoir Computing*.

(ii) Implementação de um método para otimização de *Reservoir Computing* (RC) utilizando os parâmetros clássicos de criação de RC para integração com o *toolbox* de *Reservoir Computing* que está disponível em [SVH10].

(iii) Avaliação do comportamento dos sistemas dinâmicos, criados com o método proposto, devido à definição dos pesos da camada intermediária de forma independente do raio espectral.

Partes destes tópicos geraram as seguintes publicações científicas:

- A. A. Ferreira and T. B. Ludermir. Evolutionary strategy for simultaneous optimization of parameters, topology and reservoir weights in echo state networks. In *International Joint Conference on Neural Networks - IJCNN 2010* [FL10].

- A. A. Ferreira and T. B. Ludermir. Genetic algorithm for reservoir computing optimization. In *International Joint Conference on Neural Networks - IJCNN 2009* [FL09].

- A. A. Ferreira, T. B. Ludermir, et al. Investigating the use of reservoir computing for forecasting the hourly wind speed in short-term. In *International Joint Conference on Neural Networks - IJCNN 2008* [FLdA+08].

- A. A. Ferreira and T. B. Ludermir. Using reservoir computing for forecasting time series: Brazilian case study. In *8th International Conference on Hybrid Intelligent Systems - HIS 2008* [FL08b].

- A. A. Ferreira and T. B. Ludermir. Algoritmo genético para otimização de reservoir computing: Uma primeira tentativa. In *II Workshop on Computational Intelligence - WCI 2008* [FL08a].

1.3 Organização do Documento

Neste capítulo introdutório, os objetivos e as contribuições desta tese foram apresentados. Os demais capítulos seguem a seguinte organização:

- Capítulo 2: *Reservoir Computing*

  Este capítulo apresenta conceitos básicos de Redes Neurais Recorrentes, principalmente sobre *Reservoir Computing*, que é o foco do trabalho. O capítulo apresenta a arquitetura de RC, os principais métodos, otimização de RC, diferentes maneiras para geração de *Reservoirs*, aplicações e questões abertas sobre RC, a fim de familiarizar o leitor com o objetivo e as potencialidades do trabalho.

- Capítulo 3: Método Proposto

  O capítulo apresenta a motivação do trabalho, o método proposto que integra Algoritmos Genéticos e *Reservoir Computing* e dois métodos desenvolvidos para otimização de RC e que se baseiam no raio espectral. São descritas as principais características de funcionamento e detalhes de implementação dos métodos.

- Capítulo 4: Método Experimental

  Neste capítulo são descritas as bases de dados que foram utilizadas nos experimentos, os métodos utilizados para análise e comparação dos resultados dos experimentos e o fluxo básico dos experimentos.

- Capítulo 5: Experimentos e Análise

  Este capítulo apresenta o planejamento dos experimentos, uma descrição da realização de cada experimento, os resultados obtidos com o método proposto, a comparação dos resultados com outros métodos e uma análise dos resultados alcançados.

- Capítulo 6: Conclusões e Proposta de Trabalhos Futuros

  Neste capítulo apresentam-se as conclusões obtidas com o trabalho desenvolvido, bem como propostas de trabalhos futuros que dele se originam.

2

Reservoir Computing

Em aprendizagem de máquina, estruturas *feedforward*, como redes neurais artificiais, modelos gráficos *Bayesianos* e métodos *Kernel*, têm sido estudadas extensivamente para o processamento de problemas não temporais. Porém existem muitas aplicações do mundo real que são temporais, como: tarefas de previsão, sistemas de controle ou de identificação, filtragem adaptativa, redução de ruídos, robótica, sistemas de visão e de reconhecimento da fala [SVC07].

Através do uso de estruturas *feedforward* é possível a solução de problemas temporais. Na área de modelagem de sistemas dinâmicos, Takens [Tak81] propôs que o estado do sistema dinâmico pode ser reconstruído usando uma representação implícita. O tempo é representado pelo efeito que tem sobre o processamento de sinais de forma implícita. Por exemplo, o sinal de entrada é amostrado uniformemente, e a sequência de pesos sinápticos de cada neurônio conectado à camada de entrada da rede sofre uma convolução com uma sequência diferente de amostras de entrada. Dessa forma, a estrutura temporal do sinal de entrada é inserida na estrutura espacial da rede [Hay01].

Outra possível solução de problemas temporais é através da representação explícita, ou seja, através da adição de conexões recorrentes às arquiteturas *feedforward* [Hay01]. Essas conexões recorrentes transformam o sistema em um sistema dinâmico complexo [SVC07]. Na verdade, redes neurais artificiais recorrentes (RNRs), são métodos atraentes para resolução de tarefas complicadas da engenharia. Além da capacidade de processamento temporal, as RNRs possuem também as vantagens das redes alimentadas adiante, que incluem robustez, aprendizado a partir de exemplos e habilidade para modelar sistemas não-lineares [VSDS07]. A Figura 2.1 apresenta a estrutura típica de uma rede alimentada adiante e de uma rede recorrente.

Recentemente, foram sugeridas duas soluções diferentes para os problemas das RNRs: *Liquid State Machine* (LSM) [MNM02] e *Echo State Networks* (ESN) [Jae01]. Elas

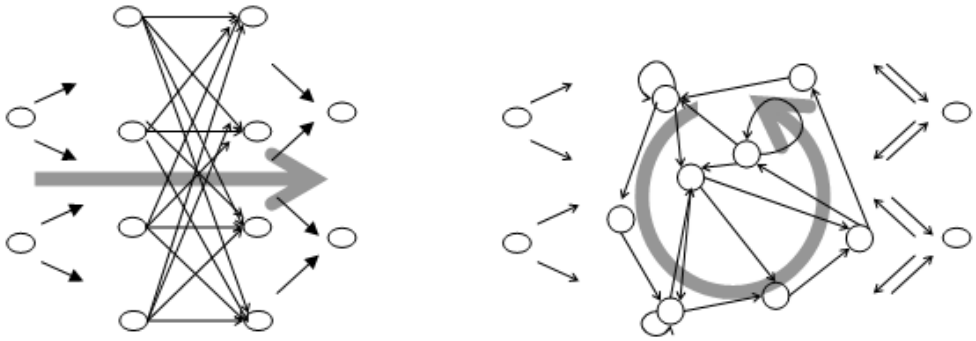

Figura 2.1 Estrutura típica de uma rede alimentada adiante (esquerda) e de uma rede recorrente (direita).

foram criadas independentemente, com diferentes tipos de aplicações e com diferentes parâmetros [JMP07]. Essas soluções descrevem a possibilidade de usar uma RNR sem adaptar os pesos das conexões internas. Em princípio a saída pode ser gerada usando algum tipo de classificador ou regressor, que pode ser um *Perceptron* [MP69] ou uma *Support Vector Machine* [Vap95]. Na maioria das aplicações um discriminante linear simples ou um algoritmo de regressão é usado para calcular a saída desejada. Segundo Verstraeten et al [VSDS07], este tipo de função *readout* oferece algumas vantagens: são simples de treinar e oferecem ótimas garantias no sentido dos mínimos quadrados além de apresentam bons resultados.

Um novo paradigma de RNRs, proposto por Verstraeten, Schrauwen, D'Haen e Stroobandt [VSDS07] e denominado de *Reservoir Computing* (RC), propõe a unificação em um único campo de pesquisa das LSM e ESN. Esse paradigma tem apresentado resultados promissores [LJ09]. Na verdade, as ideias investigadas rigorosamente sob o nome de LSM e de ESN foram sugeridas anteriormente no livro de Rosenblatt em 1962 [Ros62], foram comentadas por Hinton em 1981 [Hin81] e também por Buonomano e Merzenich em 1995 [BM95].

2.1 Descrição da Arquitetura

RNRs são exemplos de modelos computacionais neurais que manipulam o tempo sem a necessidade de pré-processar linhas de atraso de tempo. RNRs criam internamente a memória requerida para armazenar a história dos padrões de entrada através de suas conexões recorrentes [SOP+07]. RNRs têm sido largamente utilizadas em muitas apli-

cações como sistemas de identificação e controle de sistemas dinâmicos [DKW95], [KZM94], [PF95].

Reservoir Computing apresenta uma arquitetura de RNR na qual sua estrutura básica é formada por dois componentes principais: uma topologia recorrente não-linear de elementos de processamento, chamada *reservoir*, e uma camada de saída linear, chamada *readout*. Os estados do *reservoir* são chamados de *echo states* e eles contêm informação sobre a história dos padrões de entrada. A propriedade interessante de RC é que só a camada de saída (*readout*) é treinada enquanto que a topologia recorrente tem pesos aleatórios [SOP+07]. Os neurônios usados em RC podem ser de tipos diferentes (*sigmoid, leaky integrator*, ou *spiking*). A Figura 2.2 apresenta a abordagem RC com K unidades de entrada, N unidades internas e L unidades de saída.

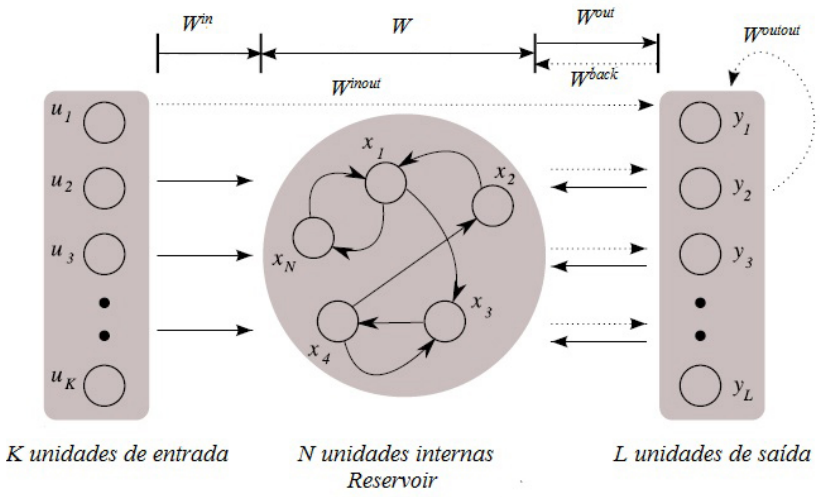

Figura 2.2 Arquitetura de *Reservoir Computing*. As linhas tracejadas indicam conexões opcionais.

2.2 Métodos de *Reservoir Computing*

Nesta seção apresentaremos os dois métodos pioneiros de RC, que são as *Echo States Networks* e as Liquid State Machines.

2.2.1 Echo State Networks

As *Echo state networks* (ESNs) representam um dos métodos pioneiros de *Reservoir Computing* e se baseiam na observação que uma RNR aleatória possui certas propriedades algébricas e que o treinamento apenas da camada de saída (*readout*) é frequentemente suficiente para alcançar bons resultados em aplicações práticas. A parte fixa (não treinada) é chamada de *reservoir* dinâmico, e os estados resultantes $\mathbf{x}(n)$ são *echo states* do histórico das entradas. ESNs usam neurônios sigmóides simples, isto é, neurônios cuja função de ativação é uma função sigmóide.

Uma ESN é composta de camada intermediária (*reservoir*) e de uma camada de saída linear (*readout*) que mapeia os estados do *reservoir* nas saídas desejadas [ASS08]. A equação geral de atualização de estados (equação (2.1)) e a equação do *readout* (equação (2.2)) são descritas a seguir:

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{back}}\mathbf{y}(n) + \mathbf{W}^{\text{bias}}), \tag{2.1}$$

$$\mathbf{y}(n+1) = \mathbf{f}^{\text{out}}(\mathbf{W}^{\text{inout}}\mathbf{u}(n+1) + \mathbf{W}^{\text{out}}\mathbf{x}(n+1) + \mathbf{W}^{\text{outout}}\mathbf{y}(n) + \mathbf{W}^{\text{biasout}}), \tag{2.2}$$

onde $\mathbf{u}(n)$ denota a entrada no tempo n; $\mathbf{x}(n)$ representa o estado do *reservoir* no tempo n; $\mathbf{y}(n)$ é a saída no tempo n; e $\mathbf{f}()$ é a função de ativação das unidades do *reservoir* (geralmente funções sigmóides) e $\mathbf{f}^{\text{out}}$ é a função de ativação das unidades da camada de saída. A matriz $\mathbf{W}$ representa as conexões do *reservoir*; a matriz $\mathbf{W}^{\text{in}}$ representa as conexões entre a camada de entrada e o *reservoir*; a matriz $\mathbf{W}^{\text{back}}$ representa as conexões entre a camada de saída e o *reservoir* e a matriz $\mathbf{W}^{\text{bias}}$, opcional, representa as conexões entre bias e *reservoir*. A matriz $\mathbf{W}^{\text{out}}$ representa as conexões entre *reservoir* e camada de saída; a matriz $\mathbf{W}^{\text{inout}}$, opcional, representa as conexões entre camada de entrada e camada de saída; a matriz $\mathbf{W}^{\text{outout}}$, opcional, representa as conexões recorrentes da camada de saída; e a matriz $\mathbf{W}^{\text{biasout}}$, opcional, representa as conexões de bias e camada de saída. Apenas as conexões direcionadas para a camada de saída são treinadas ($\mathbf{W}^{\text{inout}}$, $\mathbf{W}^{\text{out}}$, $\mathbf{W}^{\text{outout}}$ e $\mathbf{W}^{\text{biasout}}$).

É possível adicionar um parâmetro, α (*leak rate*), aos neurônios sigmóides. A adição deste parâmetro, às unidades básicas, é chamada geralmente de *leaky integrator neurons* [LJ09]. As duas principais possibilidades são a integração antes da aplicação da função de ativação ou aplicação da integração depois da função de ativação. O valor de α,

pode efetivamente ajustar a dinâmica de RC. Se α é escolhido corretamente, a dinâmica pode ser ajustada para coincidir com a escala de tempo da entrada, possibilitando ao sistema um melhor desempenho [ASS08]. A equação (2.3) apresenta às mudanças que ocorrem na equação geral de atualização de estados (equação (2.1)) de uma ESN quando se adiciona o parâmetro α antes da função de ativação.

$$\mathbf{x}(n+1) = \mathbf{f}((1-\alpha)\mathbf{x}(n) + \alpha(\mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{back}}\mathbf{y}(n) + \mathbf{W}^{\text{bias}})). \quad (2.3)$$

2.2.2 Liquid State Machines

Liquid State Machines (LSMs) representam o outro método pioneiro de *Reservoir Computing*. Uma LSM é composta por uma grande coleção de unidades (chamados de nós, ou neurônios). Os nós são conectados aleatoriamente uns aos outros. As LSMs recebem entradas contínuas e variantes no tempo que representam estímulos aferentes e, depois de processados por uma rede recorrente, obtém-se uma saída de mesma natureza, isto é, contínua e variante no tempo, que representa uma integração temporal dos estímulos prévios.

Uma LSM é formada por dois subsistemas separados: *o líquido*, que é usado para obter um vetor de estados muito complexo e variante no tempo e a *função readout*, um subsistema sem memória (geralmente formado por um perceptron simples) que é usado para extrair informações do *líquido*.

As LSMs, que possuem neurônios pulsantes, são modelos mais sofisticados de sinapse e são mais difíceis de implementar, de inicializar corretamente e tipicamente são mais caras em termos computacionais do que as ESNs. Assim elas são menos utilizadas em aplicações da engenharia [LJ09]. Entretanto, neurônios pulsantes são capazes de executar um processamento mais complicado devido à codificação do tempo da informação em seus sinais.

As LSMs precisam obedecer a uma propriedade muito restritiva, a propriedade de separação *point-wise* [MNM02], a fim de serem computacionalmente úteis. Na prática, a maioria das descrições do uso de LSMs é feita a partir de um modelo relativamente simples de neurônio pulsante chamado de neurônio *Leak Integrate and Fire* (LIF) [MB01] com um modelo de sinapse dinâmico, porém o uso de portas lógicas de limiar (TLGs - *threshold logic gates*) também foi descrito em [NBL04]. Modelos de neurônio pulsantes são mais complexos do que neurônios sigmóides, mas eles têm demonstrado serem computacionalmente mais poderosos [Maa97]. A camada de saída (*readout*) é especificada de

maneira muito ampla, ela precisa ser capaz de aproximar qualquer função contínua sobre um conjunto compacto de $x \subseteq \Re^N$ dos sinais de entrada. Quando ambas as condições são satisfeitas, garante-se que a LSM resultante é capaz de aproximar qualquer função invariante no tempo.

2.3 Criação de Reservoir Computing

A receita clássica para produzir RC foi proposta por Jaeger em [Jae01] e em [Jae02b] e é descrita em [LJ09] da seguinte maneira:

- Gere uma camada intermediária (*reservoir*) grande;
- Esparsa (em torno de 20% das possíveis conexões) e
- Aleatoriamente conectada (a partir de uma distribuição uniforme com média 0 e variância 1).

Os pesos da camada de entrada para camada intermediária ($\mathbf{W}^{\text{in}}$) e os pesos opcionais da camada de saída para camada intermediária ($\mathbf{W}^{\text{back}}$) são geralmente gerados aleatoriamente a partir de uma distribuição normal, com média 0 e variância 1. A escala exata de ambas as matrizes e uma constante opcional que pode ser adicionada a entrada ($\mathbf{u}(n)$) são alguns dos poucos parâmetros que podem ser escolhidos pelo método clássico. O dimensionamento da matriz $\mathbf{W}^{\text{in}}$ e a mudança da entrada dependem da não-linearidade que se precisa da unidade de processamento: se as entradas estão próximas a 0, os neurônios com função de ativação tangente hiperbólica tendem a operar com ativações próximas a 0, onde são essencialmente lineares, enquanto que as entradas longe de 0 tendem a levá-los mais para a saturação onde eles exibem mais não-linearidade. A mudança da entrada pode ajudar a superar as consequências indesejáveis da simetria em torno de 0 dos neurônios *tanh* no que diz respeito ao sinal de suas entradas. Efeitos semelhantes são produzidos por escalar a matriz $\mathbf{W}^{\text{bias}}$ (corresponde a constante da entrada, que muitas vezes tem um fator de escala diferente do resto de $\mathbf{W}^{\text{in}}$). A escala da matriz $\mathbf{W}^{\text{back}}$ na prática é limitada por um *threshold* no qual a ESN começa a apresentar um comportamento instável, ou seja, a conexão recorrente começa a ampliar as saídas (e os erros).

A propriedade de *echo state* é muito importante para o funcionamento das ESNs. Esta condição indica que o efeito de um estado anterior ($\mathbf{x}(n)$) e de uma entrada anterior ($\mathbf{u}(n)$) em um estado futuro ($\mathbf{x}(n+k)$) deve desaparecer gradualmente à medida que o tempo passa (ou seja, $k \longrightarrow \infty$), e não deve persistir ou mesmo ser amplificado. Na prática

a propriedade de *echo state* é assegurada se a matriz de pesos $\mathbf{W}$ é dimensionada de modo que seu raio espectral $\rho(\mathbf{W})$ (maior valor absoluto entre os autovalores) satisfaz a condição de $\rho(\mathbf{W}) < 1$ [Jae01]. O fato que $\rho(\mathbf{W}) < 1$ quase sempre garante a propriedade de *echo state* tem levado a um equívoco que aparece em muitos trabalhos de RC, em que $\rho(\mathbf{W}) < 1$ seria uma condição necessária e suficiente para a propriedade de *echo state* [LJ09]. A conexão correta entre o raio espectral e a propriedade de *echo state* é que a propriedade de *echo state* é violada se $\rho(\mathbf{W}) > 1$ apenas em RC que usam a função de ativação *tanh* e que não possuem entrada. Contrariamente a esse equívoco, a propriedade de *echo state* pode ser obtida mesmo que $\rho(\mathbf{W}) > 1$ em sistemas com entrada diferente de zero, e ela também pode ser perdida mesmo se $\rho(\mathbf{W}) < 1$, embora seja muito difícil construir sistemas onde isso ocorra.

O valor ótimo de $\rho(\mathbf{W})$ pode ser escolhido de acordo com a necessidade de memória e de não-linearidade da tarefa. A regra introduzida por Jaeger [Jae01] é que $\rho(\mathbf{W})$ deve ser próximo de 1 para tarefas que exigem muita memória e, consequentemente, menor para as tarefas onde uma memória muito longa poderia ser prejudicial. Grandes valores de $\rho(\mathbf{W})$ também podem ter o efeito de conduzir o sinal $\mathbf{x}(t)$ para regiões mais lineares das unidades *tanh* (próximas de 0).

A abordagem clássica foi utilizada por Ferreira e Ludermir em [FLdA+08] e em [FL08b] para criação de modelos para previsão da velocidade média dos ventos em diferentes bases de dados. Os resultados alcançados nestes trabalhos mostraram que a previsão realizada por ESNs criadas a partir da seleção empírica dos parâmetros estocásticos (número de neurônios e raio espectral) foi melhor do que a previsão realizada por redes *Multi-layer Perceptron* (MLPs) ou do que a previsão utilizando o método da persistência (conhecido também como método ingênuo e explicado na Seção 4.2.5).

2.4 Geração e Treinamento de *Reservoir Computing*

A ideia básica de *Reservoir Computing* (LSM e ESN) é trabalhar com uma rede neural recorrente (RNR), grande, distribuída e dinâmica, com pesos fixos e chamada de *reservoir* e usar uma função de regressão linear para treinar os pesos da camada de saída (*readout*). O uso de RC oferece uma metodologia intuitiva no uso do processamento temporal das RNRs sem os inconvenientes do treinamento das mesmas [MNM02], [Jae01], [Ste04].

Segundo Lukosevicius e Jaeger [LJ09] para explorar melhor o potencial de *Reservoir Computing* são necessárias pesquisas de alternativas para geração do *reservoir* e treinamento da camada de saída (*readout*), pois é improvável que a geração aleatória dos pesos

do *reservoir* e a utilização de regressão linear para treinar o *readout* sejam a solução ideal para RC.

Neste capítulo apresentamos vários trabalhos que apresentam alternativas à ideia básica de RC, tanto na geração do *reservoir* quanto no treinamento do *readout*.

2.5 Geração do *Reservoir*

O método clássico para geração de RC, apresentado na Seção 2.3, utiliza a ideia de gerar a camada intermediária com pesos aleatórios com topologia e pesos dependentes de alguns parâmetros que são selecionados manualmente. Porém é possível criar os pesos da camada intermediária a partir de adaptações não-supervisionadas ou a partir de um pré-treinamento supervisionado. Os métodos não-supervisionados de geração/treinamento da camada intermediária tentam minimizar alguma medida definida nas ativações do *reservoir* para uma dada entrada ($\mathbf{x}(n)$), mas desconsideram a saída desejada ($\mathbf{y}(n)$) enquanto que o pré-treinamento supervisionado leva em consideração além das entradas, $\mathbf{u}(n)$, a saída desejada $\mathbf{y}_{\text{target}}(n)$.

2.5.1 Adaptação Não-supervisionada

Os métodos não-supervisionados de geração/treinamento de *reservoirs* tentam otimizar alguma medida definida nas ativações do *reservoir* ($\mathbf{x}(n)$), para uma data entrada ($\mathbf{u}(n)$), mas não levam em consideração a saída desejada ($\mathbf{y}_{\text{target}}(n)$). Algumas medidas, além do raio espectral (descrito na Seção 2.3), que podem ser utilizadas para estimar a qualidade do *reservoir* são descritas nesta seção.

A medida do poder computacional, ou qualidade do *kernel*, que foi descrita em [MLB05] é obtida da seguinte maneira: Tome k diferentes sequências de entrada (ou segmentos do mesmo sinal) $\mathbf{u}^i(n)$, onde $i = 1,...,k$ e $n = 1,...,T_k$. Para cada entrada i tomam-se os estados resultantes do *reservoir* $\mathbf{x}^i(n_0)$ e coleciona-os em uma matriz $\mathbf{M} \in \Re^{k x N_x}$, onde n_0 é algum tempo fixo após o aparecimento da entrada $\mathbf{u}^i(n)$. Em seguida, o *rank* r da matriz $\mathbf{M}$ é medido. Se $r = k$, isto significa que todas as entradas apresentadas podem ser separadas por uma camada linear (*readout*), e assim se diz que o *reservoir* tem a propriedade de *separação linear*. Para se estimar a capacidade de generalização do *reservoir*, o mesmo procedimento pode ser realizado com s ($s \gg k$) entradas $u^j(n), j = 1,...,s$, que representam o conjunto de todas as possíveis entradas. Se o *rank* resultante r é substancialmente menor do que o tamanho do conjunto de

treinamento, o *reservoir* generaliza bem. Estas duas medidas são mais orientadas para tarefas de classificação.

Outra medida que pode ser utilizada é a minimização da faixa dos autovalores (EVS - *Eigenvalue Spread*), que é a relação entre os valores máximo e mínimo dos autovalores, da matriz de correlação cruzada das ativações $\mathbf{x}(n)$. Um pequeno valor de EVS é necessário para um treinamento online das saídas da ESN por um algoritmo estocástico de gradiente descendente, computacionalmente barato [LJ09]. Em ESNs clássicas o valor de EVS chega algumas vezes a 10^{12} ou mesmo maior, o que torna inviável a utilização do treinamento estocástico de gradiente descendente.

A adaptação não-supervisionada do *reservoir* também pode ser feita baseada na entropia da distribuição de $\mathbf{x}(n)$. Neste caso deve-se buscar uma grande entropia de $\mathbf{x}(n)$ [LJ09].

A região conhecida como *beira do caos* é uma região dos parâmetros de um sistema dinâmico no qual ele opera na fronteira entre o regime estável e o regime caótico. Em muitos tipos de sistemas dinâmicos, incluindo *reservoirs*, a *beira do caos* indica que o sistema possui um alto poder computacional [LJ09]. Em *reservoirs* a *beira do caos* só pode aparecer quando o efeito das entradas não desaparece rapidamente; assim *reservoirs* podem ter maior capacidade de memória [LM07]. Entretanto isso não implica em *reservoirs ótimos*. A *beira do caos* pode ser detectada empiricamente (mesmo para as redes biológicas), medindo expoentes de *Lyapunov* [LM07], embora tal medida não seja trivial (e necessite muitas vezes de um especialista) para sistemas de alta-dimensionalidade. Para *reservoirs* de simples unidades de *threshold* binário isto pode ser feito de maneira mais simples, calculando a distância de *Hamming* entre as trajetórias dos estados [BN04].

Há também um modelo de observação empírica que, ao se alterar as definições de diferentes parâmetros de um *reservoir*, o melhor desempenho em uma dada tarefa se correlaciona com um expoente de *Lyapunov* específico para essa tarefa [VSDS07]. O expoente ótimo está relacionado com o total de memória necessária para a tarefa. Foi observado em ESNs sem entrada que quando $\rho(\mathbf{W})$ é ligeiramente superior a 1, os sinais gerados internamente são oscilações periódicas, e quando valores de $\rho(\mathbf{W})$ maiores que 1 são usados, os sinais são mais irregulares ou mesmo caóticos [OP05]. Além disso, as entradas $\mathbf{u}(n)$ podem colocar a dinâmica do *reservoir* fora do regime caótico e assim torná-lo útil [OP05].

2.5.2 Adaptação Supervisionada

A otimização é o processo de encontrar a melhor solução em um conjunto de possíveis soluções para um problema. O processo de busca normalmente parte de uma solução inicial ou de um conjunto delas, realizando melhoramentos progressivos até chegar a outro conjunto que contenha uma ou todas as melhores soluções possíveis dentro do espaço de busca. A solução de um problema de otimização pode ser caracterizada localmente e globalmente [Roc10].

Problemas de otimização ocorrem com frequência em todas as áreas de ciência e de engenharia, surgindo sempre que há uma necessidade de minimizar (ou maximizar) uma *função objetivo*, a qual depende de um conjunto de *variáveis de decisão*, enquanto satisfizer um conjunto de *restrições* sobre a escolha das variáveis. A função objetivo permite fazer comparações de diferentes escolhas possíveis das variáveis para determinar qual dessas escolhas seria a *melhor*. Entre as aplicações usuais de otimização podemos relacionar a minimização de custos, maximização de lucros, minimização de erros, etc. Problemas de otimização podem ser formalmente especificados definindo-se um vetor de variáveis $\mathbf{x} \in \Re^n$, uma função objetivo $f : \Re^n \longrightarrow \Re$, e funções de restrições $g : \Re^n \longrightarrow \Re^m$. Porém, grande parte dos problemas do mundo real envolve a otimização de múltiplos objetivos. A maioria dos métodos disponíveis para resolução desses problemas evita as complexidades que um problema multiobjetivo envolve e converte os problemas multiobjetivos em problemas com um único objetivo [Coe99].

Redes do tipo *Reservoir Computing* diferem basicamente em tamanho, conectividade, tipo de neurônio, topologia e algoritmo de treinamento da camada de saída. Esquemas de otimização desses sistemas têm atraído a atenção de vários pesquisadores [SS05], [SWGG07], [ZBI+04] e [DBN07]. Embora a otimização de RC seja um desafio, a avaliação do seu desempenho para uma determinada tarefa, é relativamente muito simples, fazendo com que os métodos evolucionários de pré-treinamento de RC sejam uma estratégia natural de busca da melhor solução para uma determinada tarefa [LJ09].

Alguns métodos de busca de topologia de ESN foram publicados. Podemos citar o trabalho de mestrado de Benjamin Liebald [Lie04] que testou o crescimento espacial de ESNs, livres de escala. Este trabalho também explorou o método da força bruta para busca de pequenas redes (*reservoirs* com 4 neurônios).

Ishii et al utilizaram algumas abordagens evolucionárias para otimização de *reservoirs* em [IvdZBP04]. A primeira abordagem foi a de realizar uma pesquisa evolucionária sobre os parâmetros de geração da matriz do *reservoir* ($\mathbf{W}$): N (número de neurônios no *reservoir*), $\rho(\mathbf{W})$ (raio espectral), e a densidade de conexões de $\mathbf{W}$ para criar um sistema

para movimentação de um robô submarino. Em seguida, um algoritmo evolucionário foi usado em indivíduos (formados por todas as matrizes de pesos $\mathbf{W}^{\text{in}}$, $\mathbf{W}$ e $\mathbf{W}^{\text{back}}$) para pequenos *reservoirs* (5 neurônios). Foi testada também a busca apenas dos pesos de $\mathbf{W}$, sempre utilizando pequenos *reservoirs* com o objetivo de reduzir o espaço de busca. Segundo o autor [IvdZBP04], os resultados empíricos deste trabalho foram superiores aos resultados alcançados por outros métodos.

Outra abordagem de otimização de *reservoirs* usando uma pesquisa evolucionária foi apresentada em [BT05]. Este trabalho também utilizou a mesma ideia de separar a topologia e os pesos do *reservoir* com o objetivo de reduzir o espaço de busca, porém a busca foi realizada apenas na topologia da rede. Segundo o autor [BT05], os erros na previsão de comportamento de um sistema de amortecedor foram 50% menores (e muito mais estáveis) do que os erros apresentados sem a otimização genética. Mais uma vez foram utilizados pequenos *reservoirs* (com 20 neurônios).

Outra maneira de reduzir o espaço de busca dos parâmetros do *reservoir* foi apresentada em [RJH08] através da construção de um grande *reservoir* de forma fractal aplicando repetidamente multiplicação *Kronecker* em uma primeira matriz de pequeno porte, chamado de *Kronecker kernel*. Os resultados deste trabalho mostraram que a pesquisa evolucionária só precisa encontrar os bons pesos do *kernel Kronecker* para produzir bons *reservoirs*.

2.6 Treinamento do *Readout*

Nesta seção vamos nos referenciar a todos os pesos do *readout* ($\mathbf{W}^{\text{inout}}$, $\mathbf{W}^{\text{out}}$, $\mathbf{W}^{\text{outout}}$ e $\mathbf{W}^{\text{biasout}}$) usando a nomenclatura $\mathbf{W}_{\text{out}}$.

Conceitualmente, a formação do *readout* (camada de saída) a partir de um *reservoir* é uma tarefa supervisionada não-temporal de mapeamento de $\mathbf{x}(n)$ para $\mathbf{y}_{\text{target}}(n)$. Este é um domínio bem investigado em aprendizagem de máquina e existe uma grande variedade de métodos disponíveis, onde, em princípio, qualquer um deles pode ser aplicado para solução deste problema.

Na modalidade *batch*, o aprendizado dos pesos do *readout* ($\mathbf{W}_{\text{out}}$) pode ser redigido como a solução de um sistema de equações lineares (2.4).

$$\mathbf{W}_{\text{out}}(X) = \mathbf{Y}_{\text{target}}, \tag{2.4}$$

onde $\mathbf{X} \in \Re^{\text{N}x\text{T}}$ são todos os estados do *reservoir*, $\mathbf{x}(n)$, produzidos pela apresentação das entradas $\mathbf{u}(n)$. $\mathbf{Y}_{\text{target}} \in \Re^{\text{N}_y x\text{T}}$ são todos os $\mathbf{y}_{\text{target}}(n)$, ambos colecionados em

suas respectivas matrizes durante o período de treinamento $n = 1, ..., T$. Geralmente os dados $\mathbf{x}(n)$ do início do treinamento são descartados porque eles estão contaminados por transientes iniciais. O objetivo do treinamento é minimizar um erro quadrático $E(\mathbf{Y}_{\text{target}}, \mathbf{W}_{\text{out}}(X))$.

Para resolver a equação (2.4) utiliza-se com frequência métodos para encontrar as soluções dos mínimos quadrados em sistemas de equações lineares, problema também conhecido como regressão linear. Os métodos *Moore-Penrose pseudo-inverse* (*pseudo-inversa*) [LJ09] e a Regressão *Ridge* (*ridge-regress*) [Bis06] são comumente utilizados para o treinamento do *readout*, mas existem outras alternativas como a decomposição ponderada ou utilização de busca evolucionária [LJ09].

Algumas aplicações necessitam de modelos de adaptação *online*, nesses casos, minimiza-se um erro que é exponencialmente descontado voltando no tempo [LJ09]. $\mathbf{W}_{\text{out}}$ funciona aqui como um combinador linear adaptativo. A maneira mais simples é usar um treinamento estocástico de gradiente descendente. Um dos métodos utilizados é o algoritmo *Least Mean Squares* (LMS), descrito em [FB98], porém a convergência deste método é lenta quando há uma grande faixa de autovalores de $\mathbf{X}\mathbf{X}^{\text{T}}$.

O algoritmo *Recursive Least Squares* (RLS) é uma alternativa ao LMS, ele é largamente utilizado em processamento de sinais, é insensível aos efeitos negativos da grande faixa de autovalores de $\mathbf{X}\mathbf{X}^{\text{T}}$ e apresenta convergência mais rápida, pois é um método de segunda ordem [Jae03], porém segundo [Jae03], o RLS é computacionalmente mais caro.

O algoritmo *BackPropagation-DeCorrelation* (BPDC) [Ste04] é um outro método poderoso para o treinamento *online* de *readouts* de camada única. O BPDC combina princípios importantes: um passo para retro-propagação dos erros e uso de memória temporal de redes dinâmicas. A regra BPDC é derivada e teoricamente justificada sobre a aprendizagem como um problema de otimização. Segundo Steil [Ste04], ela é muito fácil de ser implementada e tem uma pequena complexidade.

Segundo Lukosevicius e Jaeger [LJ09], o *reservoir* pode ser considerado um *Kernel* temporal, dessa forma o *readout* pode ser treinado usando *Support Vector Machines* (SVMs) ou *Support Vector Regression* (SVR). A utilização de versões diferentes desses algoritmos para o treinamento de *Echo State Networks* foi investigada por Shin e Han em [SH07].

Embora a maior parte dos trabalhos utilize o treinamento supervisionado do *readout*, uma regra de aprendizado por reforço, chamada de *Spike-Time-Dependent Plasticity* (STDP), foi utilizada com sucesso para o treinamento do *readout* em [LPM09].

Uma busca genética para otimização não-supervisionada do *readout* foi utilizada por

Xu et al em [XLP05], o método foi aplicado ao controle de sistemas não-lineares. Eles demonstraram que o design do controle adaptativo usando ESNs e algoritmos genéticos apresentou um bom desempenho com uma baixa complexidade de treinamento.

Outra estratégia evolucionária foi utilizada para otimização do treinamento de ESNs em um *benchmark* clássico da embriologia artificial multicelular (conhecida por "flag") por Devert et al em [DBS08]. Segundo Devert et al, a abordagem utilizada apresentou resultados muito bons.

Redes *Multi–Layer Perceptron* (MLPs) foram utilizadas como *readout* e treinadas com o algoritmo *backpropagation* desde o princípio das pesquisas com LSMs [MNM02]. As redes MLPs são teoricamente mais poderosas para o mapeamento de $\mathbf{x}(n)$ para $\mathbf{y}(n)$ do que a regressão linear simples, porém são significativamente mais difíceis de treinar. Bush e Anderson utilizaram *readouts* formados por MLPs treinadas pelo algoritmo *backpropagation* em [BA05].

Jaeger introduziu em [Jae07] uma arquitetura hierárquica de ESNs, que utiliza o fluxo *bottom-up* da arquitetura para extrair as características do sinal de entrada e o fluxo *top-down* para passar expectativas de recursos. A arquitetura é treinada por um método estocástico de gradiente descendente.

2.7 Aplicações

Algumas aplicações de sucesso de RC tanto em problemas *abstratos* quanto em aplicações de engenharia do mundo real têm sido reportadas na literatura. Aplicações abstratas incluem classificação de padrões dinâmicos [Jae02a], geração de sinos autônomos [Jae01] ou computação de funções não-lineares em taxas instantâneas de trens pulsantes [MNM04]. Em robótica, LSMs foram usadas para controlar um braço de um robô simulado [JM04], para modelar um controlador de robô [Bur05b], para executar a previsão da trilha e do movimento de um objeto [Bur05a], [MLM02], detecção de eventos [Jae05] ou em algumas aplicações da competição *Robocup* (principalmente controle de motor) [SP05], [OLSB05]. ESNs foram usadas no contexto de aprendizado por reforço [BA05]. Também em aplicações no campo do processamento de sinais digitais têm tido muito sucesso, como reconhecimento da fala [MNM03],[VSSC05] ou modelagem de ruído [JH04]. E finalmente, o uso de *Reservoir Computing* para geração e previsão de séries temporais caóticas também foi reportado em [Jae02a], [Jae03], [Ste05] e [Ste06].

Em muitas áreas como previsão de séries temporais caóticas e reconhecimento de dígitos isolados, técnicas RC são o *estado-da-arte* [SVH10], [LJ09]. Um caso que chama

a atenção é demonstrado por Jaeger em [JH04], segundo o autor a previsão da série temporal caótica de *Mackey-Glass* foi melhorada por um fator de 2400 em relação às técnicas anteriores.

2.8 Implementação

O conceito de RC foi introduzido usando uma RNR, mas na verdade RC é um conceito maior, onde qualquer sistema dinâmico de alta dimensionalidade com a propriedade correta de dinâmica pode ser usado como um *Kernel* temporal para pré-processar os dados que podem depois ser facilmente processados com técnicas lineares.

Recentemente um *toolbox* de código aberto que implementa uma grande faixa de técnicas RC foi liberado [SVH10]. O *toolbox* consiste de funções que permitem a utilização de conjunto de dados, definição das topologias de RC, parâmetros dos experimentos e análise dos resultados. Ele suporta neurônios analógicos (linear, sign, tanh) e neurônios pulsados (LIF). Os neurônios pulsados são simulados e otimizados com o simulador de eventos C++.

Atualmente já existe implementações de RC em hardware digital [SDVC07b].

2.9 Questões Abertas para Pesquisa

RC é uma área de pesquisa relativamente nova, muito atrativa, que tem se expandido rapidamente, mas que ainda possui muitos problemas abertos para serem pesquisados. O sucesso dos métodos de treinamento de RC superando RNRs tradicionais em muitas tarefas (mas nem todas) estabeleceu um marco importante, ou até mesmo uma mudança nas pesquisas sobre o treinamento de RNRs. As várias modificações da ideia original estão aumentando gradualmente o desempenho de RC. O fato de utilizar uma camada intermediária (*reservoir*) de pesos fixos, gerada aleatoriamente, e de treinar apenas a camada linear de saída (*readout*), consistentemente são o *estado-da-arte* dos métodos de treinamento de RNRs e têm várias consequências:

- Segundo Lukosevicius e Jaeger [LJ09] nós ainda não sabemos bem como treinar RNRs, e alguma coisa nova é necessária. Os métodos de *backpropagation*, que causaram um grande avanço no treinamento das redes neurais *feedforward*, e também se tornaram os métodos mais populares de treinamento de RNRs, dificilmente obtém o potencial pleno das RNRs.

- Os métodos clássicos de treinamento de RC ainda não exploram o pleno potencial das RNRs, pois eles utilizam uma camada intermediária aleatória (*reservoir*), que é improvável que seja ideal, e uma camada linear de saída (*readout*), que é bastante limitada pela qualidade dos sinais que ela combina. Mas eles nos dão um desempenho muito superior quando comparados a métodos mais sofisticados [LJ09].

- A separação entre camada intermediária e camada linear de saída fornece uma boa plataforma para experimentar vários tipos de métodos de adaptação no *reservoir* e para observar o quanto esses métodos de adaptação podem melhorar o desempenho sobre os *reservoirs* criados aleatoriamente. Isto é particularmente bem aplicado para o teste de vários mecanismos evolucionários de adaptação de RNRs.

- Do ponto de vista teórico, é muito importante um entendimento da dinâmica de RC e a geração de medidas dessa dinâmica. Foi dito em [VSDS07] que a condição de reescalar a matriz de pesos da camada intermediária (*reservoir*) por um raio espectral próximo a 1 fornece uma medida da qualidade computacional desta camada e tudo pode ser deduzido a partir desta matriz, isto é, sem simular o sistema explicitamente com algum sinal de entrada. Entretanto, não está claro como o raio espectral próximo a 1 se relaciona com a real dinâmica do sistema.

- Ainda são desconhecidas quais propriedades de *Reservoir Computing* são responsáveis pela força ou fraqueza do mesmo para uma tarefa particular. Portanto a escolha dos parâmetros da rede como quantidade de neurônios, topologia e pesos é um dos problemas mais difíceis quando se usa RC. A criação de um método supervisionado para realizar a adaptação dos pesos do *reservoir* e o treinamento do *readout*, que considere o desempenho baseado nas entradas e nas saídas produzidas, que possibilite a criação de diferentes topologias e que considere a real dinâmica do sistema deve ser investigada. Dentro deste contexto o método proposto neste trabalho foi inicialmente investigado em [FL10].

3

Método Proposto

Embora as RNAs tenham se mostrado uma técnica eficiente para a solução de um grande número de problemas, elas não são suficientes para resolver qualquer problema de Inteligência Artificial (IA). As RNAs apresentam vários problemas e limitações que não permitem seu uso exclusivo para a solução de uma quantidade significativa de problemas. Sistemas Híbridos combinam dois ou mais diferentes subsistemas, para forma um sistema heterogêneo com características inteligentes. A ideia principal do desenvolvimento de Sistemas Híbridos é a de que uma única técnica, devido às suas limitações ou deficiências, pode não ser capaz, por si só, de resolver um dado problema. Neste caso, a combinação de duas ou mais técnicas pode levar a uma solução mais robusta e eficiente [BdLFCL07]. O principal foco das pesquisas de Sistemas Híbridos tem sido em geral o de combinar técnicas fortemente baseadas em dados (RNA, por exemplo) com técnicas que se utilizam de conhecimento (Lógica *Fuzzy*, por exemplo) [Zad71]. Os Algoritmos de Otimização Global (GOA -*Global Optimization Algorithms*) também têm sido utilizados na criação dos Sistemas Inteligentes Híbridos (SIH), principalmente para a configuração dos valores *ótimos* dos parâmetros dos sistemas inteligentes.

Yamazaki criou em [Yam04] Sistemas Inteligentes Híbridos que combinavam técnicas de *tabu search, simulated annealing* e *backpropagation* para otimização da arquitetura e pesos em redes neurais artificiais *Multi-layer Perceptron*. O algoritmo proposto combina características de *simulated annealing* e *tabu search*, além de fazer uso de treinamento híbrido, através da utilização de uma técnica local de treinamento.

Zanchettin propôs em [Zan08] um método de otimização global e local, baseado na integração das heurísticas das técnicas *Simulated Annealing, Tabu Search*, Algoritmos Genéticos e *Backpropagation*. O desempenho foi investigado na otimização simultânea da topologia e dos valores dos pesos das conexões entre as unidades de processamento de redes neurais artificiais *Multi-layer Perceptron*, a fim de gerar topologias com poucas

conexões e alto desempenho para qualquer conjunto de dados. A heurística proposta realiza a busca de forma construtiva e baseada na poda das conexões entre as unidades de processamento da rede. Assim, são geradas redes com arquitetura variável e que podem ser ajustadas para cada problema de forma automática.

Ferreira e Ludermir criaram Sistemas Inteligentes Híbridos que utilizavam a abordagem evolucionária para otimização de *Reservoir Computing* em [FL08a] e [FL09]. Esses trabalhos também utilizaram a ideia de separar a topologia e os pesos da camada intermediária (*reservoir*) para reduzir o espaço de busca. O algoritmo genético foi utilizado para buscar a melhor configuração de alguns parâmetros estocásticos (número de neurônios no *reservoir*, função de ativação, raio espectral, etc) de geração de ESNs para previsão de diferentes séries de velocidade média dos ventos. Os resultados alcançados mostraram que a busca genética gastou apenas 20% do tempo necessário para busca exaustiva dos mesmos parâmetros.

Em seguida, Ferreira e Ludermir [FL10] utilizaram a busca evolucionária de forma inovadora, buscando simultaneamente parâmetros, topologia e pesos de *Reservoir Computing*, sem se preocuparem com o espaço da busca e com o raio espectral. Os resultados do trabalho mostraram que o método proposto é promissor, pois os erros foram menores do que os obtidos em trabalhos anteriores e também porque o método era computacionalmente barato.

3.1 Motivação

Os algoritmos genéticos (ver Apêndice A) são muito eficientes para busca de soluções ótimas (ou pelo menos aproximadamente ótimas) em uma grande variedade de problemas porque eles não impõem muitas das limitações encontradas nos métodos tradicionais [Hol92]. Por este motivo decidiu-se pela investigação de seu uso na escolha de (*Reservoir Computing*) para problemas de previsão de séries temporais. Geralmente a pesquisa pelos parâmetros de *Reservoir Computing* é feita de maneira exaustiva ou através de experimentos sistemáticos, que em geral levam muito tempo para serem realizados e utilizam muitos recursos computacionais.

A busca de *Reservoir Computing* usando algoritmo genético pode ser realizada de três formas diferentes: a) como um processo evolucionário que opera diretamente na topologia da rede; b) como um operador evolucionário que trabalha nos principais parâmetros que geram o sistema e c) ambos ao mesmo tempo. Segundo Verstraeten et al [VSDS07] e Ishii [IvdZBP04] os parâmetros principais que determinam a dinâmica de RC são o

número de neurônios, o raio espectral, o percentual de conexões, e o tipo da função de ativação dos neurônios.

Se o método evolucionário opera diretamente nas conexões (opções a e c do parágrafo anterior) o espaço de busca pode ser muito grande. Para uma rede com N neurônios o vetor genético é na ordem de N^2 e são necessários operadores especiais para combinar indivíduos com tamanhos diferentes [IvdZBP04]. A segunda forma (opção b do parágrafo anterior) foi usada em vários trabalhos [IvdZBP04], [FLdA+08] e [FL09]. Um problema com essa estratégia é a investigação imprecisa dos parâmetros globais, especialmente porque as interconexões da rede e os pesos são gerados aleatoriamente. Por outro lado, o vetor genético e o espaço de busca é significativamente menor. Embora a estratégia de busca de pesos e conexões tenha sido evitada até o presente momento, devido principalmente ao grande espaço de busca, o método proposto nesta tese adota a terceira opção e apresentou resultados promissores [FL10].

Jaeger descreve em [Jae01] as condições necessárias para que um *reservoir*, construído com neurônios sigmóides e função de ativação *tanh*, tenha a propriedade de *Echo State*. A propriedade de *Echo State* diz que o estado de ativação de uma RNR, $\mathbf{x}(n)$, é função do histórico de entradas $(\mathbf{u}(n), \mathbf{u}(n-1), ...)$ apresentados à rede. Mais precisamente, sobre certas condições, existe uma função *echo* $E = (e_1, ..., e_n)$, onde $e_1 : U^{-N} \longrightarrow \Re$, na qual, para todo o histórico de entradas $(..., \mathbf{u}(n-1), \mathbf{u}(n) \in U^{-N})$, o estado da rede é dado pela equação (3.1):

$$\mathbf{x}(n) = E(..., \mathbf{u}(n-1), \mathbf{u}(n)). \tag{3.1}$$

As condições descritas por Jaeger em [Jae01] e em [Jae02b] são baseadas no maior valor singular, $\sigma = \sigma_{max}(\mathbf{W})$, e no raio espectral, $\rho = |\lambda_{max}|(\mathbf{W})$, da matriz de pesos do *reservoir* ($\mathbf{W}$). A primeira condição diz que para haver a propriedade de *Echo State*, o maior valor singular (σ) deve respeitar a seguinte restrição, $\sigma < 1$. A segunda condição, conhecida como $\rho > 1$, diz que se o raio espectral (ρ) da matriz de conexões do *reservoir* é maior do que 1 a rede tem um estado nulo assintoticamente instável, desta forma não possui a propriedade de *Echo State*.

A segunda condição é descrita em [VSDS07] na forma $\rho < 1$, e significa que todos os autovalores da matriz de conexões do *reservoir* devem estar dentro de um círculo unitário no plano complexo. Essa condição expressa que o reservatório é localmente assintoticamente estável ao redor da origem e é uma condição necessária, mas não suficiente para garantir a propriedade de *Echo State*.

Como a primeira condição ($\sigma < 1$) é considerada muito restritiva por Jaeger em

[Jae01], ele utiliza a segunda condição, para sugerir um método heurístico para construção de *reservoirs* em [Jae02b]: construir a matriz de pesos do *reservoir* (**W**) aleatoriamente a partir de uma distribuição normal com média 0 e variância 1. Reescalar **W**, primeiro, dividindo pelo maior valor absoluto dos seus autovalores e depois multiplicando por um fator escolhido pelo especialista (geralmente próximo a 1, para ser próximo da fronteira de estabilidade).

Apesar de Jaeger ter proposto um método heurístico em [Jae02b], baseado no que foi descrito no parágrafo anterior, ele também destacou que redes sem a propriedade de *Echo State* algumas vezes podem ser transformadas em redes com *Echo State* pela aplicação das entradas ao sistema e que isso pode ser observado em redes neurais biológicas.

O raio espectral é um parâmetro importante que controla a dinâmica do *reservoir*. Ele escala globalmente os autovalores da matriz de conexões do mesmo. Segundo Verstraeten et al [VS09], o *reservoir* pode ser aproximado como um sistema linear discreto invariante no tempo:

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k], \tag{3.2}$$

$$\mathbf{y}[k+1] = \mathbf{C}\mathbf{x}[k+1] + \mathbf{D}\mathbf{u}[k+1], \tag{3.3}$$

onde $\mathbf{x}[k]$ representa o estado do *reservoir* (o vetor das ativações dos neurônios) no tempo k, e $\mathbf{u}[k]$ e $\mathbf{y}[k]$ representam a entrada e a saída do sistema respectivamente. A matriz **A** contém os pesos do *reservoir* (**W**) e a matriz **B** contém os pesos da entrada para o *reservoir* ($\mathbf{W}^{in}$), e **C** e **D** contêm os pesos treinados do *reservoir* para saída e da entrada para saída ($\mathbf{W}^{out}$ e $\mathbf{W}^{inout}$). Segundo [VS09] a matriz **D** é normalmente igual a zero. Como podemos observar nas equações acima (equação 3.2 e equação 3.3), esta modelagem não considera as conexões opcionais da camada de saída para o *reservoir* ($\mathbf{W}^{back}$) e nem as de bias ($\mathbf{W}^{bias}$ e $\mathbf{W}^{biasout}$) [OXP07].

A ideia de criar RC com raio espectral próximo a 1 se baseia na teoria dos sistemas lineares, pois em um sistema linear, se a matriz **A** tem todos os valores singulares menores que 1, ela é definida como estável (o maior ganho em qualquer direção do espaço de estados será menor que 1), enquanto que se qualquer valor absoluto dos seus autovalores (raio espectral) é maior que 1, o sistema será instável (ele irá se desviar sem barreiras, a partir do ponto fixo, quando iniciado a partir de um estado diferente de zero) [VS09]. Isto não acontece com as redes utilizadas neste trabalho, pois elas são do tipo ESN e seus neurônios utilizam uma função de ativação não-linear (*tanh* ou *sign*) que neutraliza o

crescimento sem limites das saídas de RC (ver Figura 3.1).

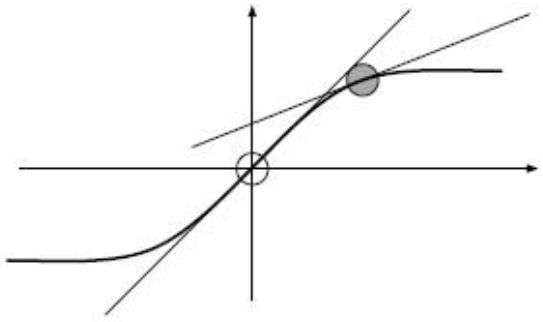

Figura 3.1 Ao redor da origem o ganho da não-linearidade é maior (círculo branco). Uma vez que o neurônio pode ser guiado por uma entrada externa ou um bias, o ponto de trabalho se desloca (círculo cinza) e o ganho local diminui.

Verstraeten et al [VSDS07] reforçaram que as condições, descritas por Jaeger, para a propriedade de *Echo State* descrevem uma medida para calcular a qualidade de RC, e que essa medida pode ser deduzida a partir da matriz de peso do *reservoir* sem a necessidade de simular o mesmo explicitamente com sinais de entrada. Porém, eles também comentam que não fica claro como essa medida está relacionada com a verdadeira dinâmica do *reservoir* já que a mesma não leva em consideração os estados resultantes da aplicação das entradas ao sistema [VSDS07].

Ozturk et al, em [OXP07], comprovaram que a dinâmica de RC não é determinada apenas pelos seus pesos fixos e que os sinais de entrada influenciam esta dinâmica. Eles demonstraram que existem várias matrizes de pesos com o mesmo raio espectral, e infelizmente elas não possuem o mesmo desempenho quando o MSE (erro médio quadrático) é calculado, especificamente em problemas de aproximação de função.

Ozturk e Príncipe relembram, em [OP05], que a ideia de reescalar as matrizes para ajustar o raio espectral dentro de um círculo unitário no plano complexo vem da teoria dos sistemas lineares que mostra que essa condição é necessária para obterem-se respostas úteis em um sistema linear. No entanto, este argumento, não se aplica necessariamente a sistemas não-lineares, que é o caso de RC. Ozturk e Príncipe apresentaram também que um sistema não-linear pode ser instável, pela condição do raio espectral maior que 1, e ainda assim ser capaz de ter sua dinâmica controlada pela entrada, quando a mesma for aplicada [OP05].

Sendo assim, esta tese propõe um método que busca simultaneamente pelos parâmetros globais de *Reservoir Computing*, pela topologia e pelos pesos do *reservoir*,

considerando que a dinâmica de RC não é determinada apenas pelos seus pesos fixos e que os sinais de entrada influenciam nesta dinâmica. O método proposto não considera a aproximação de RC com sistemas lineares, pois não reescala a matriz de pesos do *reservoir* pelo raio espectral e também considera RC em toda a sua não-linearidade, pois permite a criação de sistemas que utilizem também as conexões opcionais de RC. O método é chamado de RCDESIGN (Reservoir Computing Design e Treinamento) e é descrito na Seção 3.2.

Nesta tese desenvolvemos também dois métodos de busca para comparar com RCDESIGN. O primeiro método, chamado neste trabalho de *Busca RS* e descrito na Seção 3.3, se baseia no método clássico de criação de ESNs (ver Seção 2.3), que usa a ideia de reescalar a matriz de pesos do *reservoir* pelo raio espectral menor ou igual a 1 e uma topologia fixa (apenas com as conexões obrigatórias de RC), para aproximar RC de Sistemas Lineares. Este método busca os principais parâmetros de geração de RC, que são o tamanho do *reservoir*, o raio espectral e a densidade de conexão. O segundo método, chamado neste trabalho de *Busca TR* e descrito na Seção 3.4, busca simultaneamente pelo raio espectral e pela topologia da rede sem fazer a aproximação entre RC e Sistemas Lineares.

Os métodos desenvolvidos (RCDESIGN, *Busca RS* e *Busca TR*) serão aplicados em problemas de previsão de séries temporais. Os resultados apresentados e as comparações dos métodos são apresentados no Capítulo 5. Os resultados mostram que é possível encontrar redes bem ajustadas de forma automática, independente do conhecimento detalhado de cada parâmetro de RC, eles mostram também que a busca dos pesos, sem a restrição da propriedade de *Echo State*, permite que se encontrem soluções mais adequadas para o problema.

3.2 Método Proposto: RCDESIGN

O método proposto foi integrado ao *toolbox* de RC [SVH10] e utiliza Algoritmo Genético para buscar simultaneamente pelos parâmetros globais de *Reservoir Computing*, pela topologia e pelos pesos do *reservoir*. O método é chamado de RCDESIGN (RC Design e Treinamento) e desconsidera a aproximação de RC com sistemas lineares.

3.2.1 Função de Aptidão

Embora o método RCDESIGN possa trabalhar com diferentes funções de aptidão, neste trabalho a função de aptidão adotada leva em consideração o desempenho no conjunto de

treinamento e no conjunto de validação, de maneira a adaptar para o ambiente de RC o critério de perda da capacidade de generalização (GL – *Generalization Loss*) do Proben1 [Pre94]. Isto é necessário devido ao fenômeno de *overfitting*: para duas redes treinadas para o mesmo problema, a rede que apresenta maior erro no conjunto de treinamento pode na verdade ser melhor do que a rede que apresenta o menor erro, pois a rede com menor erro pode ter se concentrado nas peculiaridades do conjunto de treinamento e pode ter perdido a regularidade necessária para uma boa generalização. A função *aptidão* proposta é apresentada na equação (3.4).

$$f = \overline{MSE}_{Treinamento} + ||\overline{MSE}_{Treinamento} - \overline{MSE}_{Validacao}||, \tag{3.4}$$

onde f é o valor a ser minimizado pelo algoritmo evolucionário e MSE é calculado segundo a equação (3.5).

$$MSE = \frac{1}{N.P}\sum_{i=1}^{P}\sum_{j=1}^{N}(T_{ij} - L_{ij})^2. \tag{3.5}$$

P é o número total de padrões no conjunto; N é o número de unidades de saída da rede; T_{ij} e L_{ij} são o valor desejado de saída (target) e o valor calculado pela rede para o *i-th* neurônio da camada de saída, respectivamente.

3.2.2 Representação das Soluções

Seja P^n uma coleção (população) dos vetores s^i, onde n representa uma geração do algoritmo genético e s^i representa um indivíduo da população. O valor máximo de n é o número máximo de gerações (parâmetro N_G) e o tamanho do conjunto P^n é definido pelo parâmetro T_P do algoritmo. A notação s^i_j denota a característica (gene) j do indivíduo i.

- s^i_1 (η) - Define o tamanho das matrizes de pesos $\mathbf{W}$, $\mathbf{W}^{in}$, $\mathbf{W}^{bias}$ e $\mathbf{W}^{back}$. $[50;200]$.
- s^i_2 - Conexão entre a entrada e saída. Se 1 há conexão, se 0 não há.
- s^i_3 - Conexão entre bias e saída. Se 1 há conexão, se 0 não há.
- s^i_4 - Conexão de realimentação na saída. Se 1 há conexão, se 0 não há.
- s^i_5 - Conexão entre bias e *reservoir*. Se 1 há conexão, se 0 não há.
- s^i_6 - Conexão entre saída e *reservoir*. Se 1 há conexão, se 0 não há.

- s_7^i - Função de ativação dos neurônios. Se 1 é *tanh*, se 2 é *sign*.
- s_8^i - Função de treinamento do *readout*. Se 1 é *pseudo-inversa*, se 2 é *ridge-regress*.
- s_9^i - *Leak rate*. $[0.1;1]$.
- s_{10}^i - Parâmetro de regularização das funções de treinamento. $[10^{-8};10^{-1}]$.
- $s_{11}^i \ldots s_{(\eta^2+3\eta+10)}^i$ - Pesos das matrizes $\mathbf{W}$, $\mathbf{W}^{in}$, $\mathbf{W}^{bias}$ e $\mathbf{W}^{back}$. $[-1;1]$.

Como o tamanho do vetor s^i dependente do valor de s_1^i (η), o intervalo de [50;200] foi escolhido de maneira empírica de forma que o método criasse *reservoirs* grande o suficiente para solução dos problemas mas também possibilitando a viabilidade computacional da realização dos experimentos. O vetor s^i terá tamanho mínimo de 2.660 quando η for igual a 50 e terá tamanho máximo de 40.610 quando η for igual a 200. A Figura 3.2 apresenta a divisão conceitual do vetor s^i.

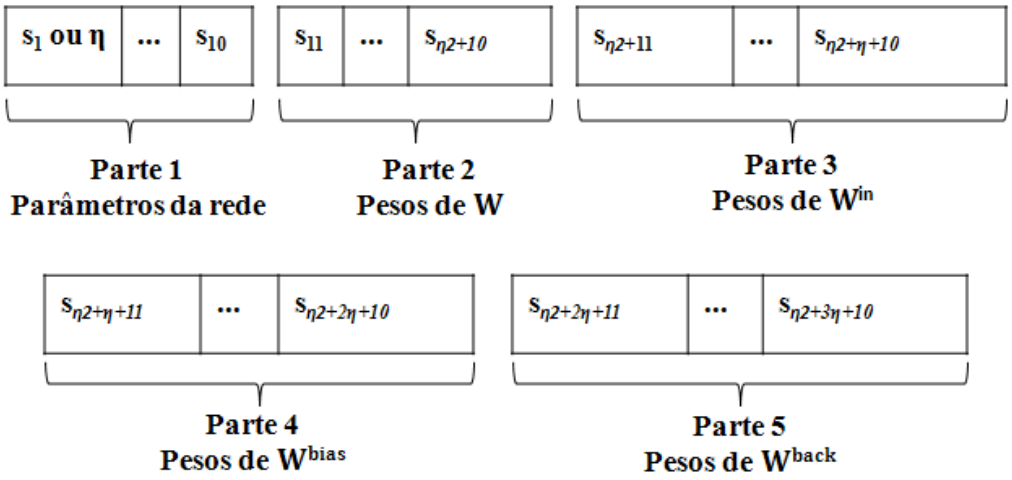

Figura 3.2 Divisão conceitual de s^i.

O parâmetro de regularização utilizado por RCDESIGN corresponde ao ruído que é adicionado às respostas do *reservoir* durante o treinamento com a *Moore-Penrose pseudo-inverse* (*pseudo-inversa*) [LJ09] ou o λ do treinamento com Regressão *Ridge* (*ridge-regress*) [Bis06].

3.2.3 Descrição do algoritmo

O processo de busca do algoritmo genético envolve uma sequência de passos, onde um conjunto de soluções passa por processos de seleção e reprodução. Para criar a próxima geração, o algoritmo seleciona certos indivíduos da população atual, chamados de *pais*, e utiliza-os para criar indivíduos na próxima geração, chamados *filhos*. Normalmente, o algoritmo é mais propenso a escolher os *pais* que tenham melhor desempenho, baseado na *aptidão*. O método RCDESIGN utiliza a seleção *uniforme estocástica* (ver Apêndice A.3) e cria três tipos de *filhos* para a próxima geração: a) Elite, b) Cruzamento e c) Mutação. A etapa de criação dos conjuntos de treinamento e validação é descrita na Seção 5.1.

Algoritmo 1: Pseudo-código de **RCDESIGN**

```
Criar população inicial e aleatória, P^0, com tamanho T_P;
Chamar função CriaIndividuos;
n = 1;
enquanto (n ≤ N_G) e não convergir faça
    Selecionar indivíduos da população P^(n−1) (pais);
    Gerar indivíduos da população P^n por elite, cruzamento, e mutação;
    Chamar função CriaIndividuos;
    n = n + 1;
Retornar a melhor solução de P^n;
Criar rede (Reservoir Computing) de acordo a melhor solução;
Calcular os erros no conjunto de teste;
```

Algoritmo 2: Pseudo-código da função ***CriaIndividuos***

```
enquanto i ≤ T_P (tamanho da população) faça
    Criar rede (Reservoir Computing) de acordo com s^i;
    enquanto fold ≤ 10 (validação cruzada) faça
        Criar conjunto de treinamento (nove partições) e conjunto de validação
        (uma partição);
        Simular rede com conjunto de treinamento;
        Treinar pesos da camada linear de saída (readout);
        Calcular erros no conjunto de treinamento;
        Calcular erros no conjunto de validação;
    Calcular o valor de aptidão de s^i;
    i = i + 1;
```

3.2.4 Descrição dos operadores de Cruzamento, Mutação e Elitismo

O operador de cruzamento utilizado por RCDESIGN é uma adaptação do cruzamento uniforme para populações de indivíduos com tamanhos diferentes. A ideia de combinar indivíduos de tamanhos diferentes já foi utilizada em outros trabalhos de redes neurais artificiais [Zan08], mas é nova no ambiente de RC.

Para cada par de pais o vetor de maior dimensão é definido como s^A e o menor como s^B. Os elementos de s^{filho} são combinados a partir de uma máscara ($s^{Mascara}$) criada aleatoriamente. A máscara indica quais as características de s^{filho} que serão herdadas de s^A e quais as características que serão herdadas de s^B. A Figura 3.3 ilustra funcionamento do operador de cruzamento na primeira parte de s^{filho} (do gene 1 ao gene 10).

s^A	s^A_1	s^A_2	...	s^A_9	s^A_{10}
s^B	s^B_1	s^B_2	...	s^B_9	s^B_{10}
$s^{Mascara}$	1	0	...	1	0
s^{Filho}	s^A_1	s^B_2	...	s^A_9	s^B_{10}

Figura 3.3 Funcionamento do operador de cruzamento na primeira parte de s^{filho}.

A segunda parte de s^{filho}, corresponde aos pesos da matriz $\mathbf{W}$. Caso s^{filho} tenha herdado o gene s_1 de s^B, então o operador de cruzamento vai combinar s^A e s^B até o gene $s^B_{(\eta^2)}$. Caso s^{filho} tenha herdado o gene s_1 de s^A, então o operador de cruzamento vai combinar s^A e s^B até o gene $s^B_{(\eta^2)}$ e então copiar os genes s^A até $s^A_{(\eta^2)}$. O operador de cruzamento funciona da mesma forma na terceira ($\mathbf{W}^{in}$), quarta ($\mathbf{W}^{bias}$) e quinta partes ($\mathbf{W}^{back}$) de s^{filho}. A Figura 3.4 ilustra o funcionamento do operador de cruzamento na segunda parte de s^{filho}. O operador de cruzamento funciona da mesma forma para terceira, quarta e quinta partes de s^{filho}.

O operador de mutação utilizado por RCDESIGN também é uma adaptação da mutação para populações de indivíduos com tamanhos diferentes. Para cada indivíduo selecionado como pai, (s^A), cria-se uma máscara ($s^{Mascara}$). O gene s^A_1 é copiado para s^{Filho}. A mutação pode ocorrer a partir do segundo gene de s^A até o último gene de s^A com taxa de mutação definida pelo parâmetro denominado de $Taxa_M$.

RCDESIGN utiliza elitismo com valor definido pelo parâmetro chamado *número de elite*.

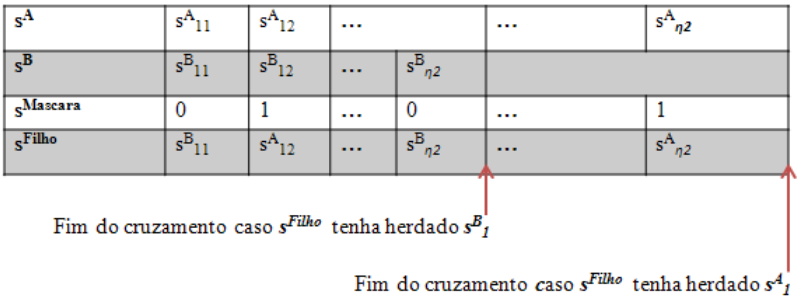

Figura 3.4 Funcionamento do operador de cruzamento na segunda parte de s^{filho}.

s^A	s^A_1	s^A_2	s^A_3	s^A_4	s^A_5	...	$s^A_{\eta 2+3\eta+10}$
$s^{Mascara}$	-	0	1	0	1	...	0
s^{Filho}	s^A_1	s^A_2	Novo	s^A_4	Novo	...	$s^A_{\eta 2+3\eta+10}$

Figura 3.5 Operador de mutação.

3.2.5 Parâmetros

Ao utilizar AGs para solução de um problema, é importante analisar como a escolha dos parâmetros utilizados pode influenciar no seu comportamento para que se possa definir valores conforme as necessidades do problema e dos recursos disponíveis. Os principais parâmetros são: tamanho da população, taxa de cruzamento e taxa de mutação. Embora existam algumas heurísticas, a escolha desses parâmetros geralmente ocorre por meio de um processo de tentativa e erro [BdLFCL07].

Foram realizados experimentos aleatórios para definição dos parâmetros do algoritmo genético utilizado por RCDESIGN. A configuração dos parâmetros que apresentou menor erro nos testes experimentais foi escolhida para ser utilizada nos experimentos. RCDESIGN utiliza os seguintes parâmetros:

- Tamanho da População (T_P) – 120
- Número de Gerações (N_G) – 10
- Número de Elite – 2

- Probabilidade de Cruzamento ($Prob_C$) – 75%
- Taxa de Mutação ($Taxa_M$) – 3%

3.3 Método para Comparação: *Busca RS*

Este método foi desenvolvido com o objetivo de comparar os resultados de RCDESIGN com os resultados de um método que busque apenas os parâmetros globais de geração de RC e que seja baseado no método clássico de criação de ESNs (Seção 2.3). Os parâmetros considerados foram o *número de neurônios no reservoir, o raio espectral e o percentual de conexões*. De acordo com o método proposto por Jaeger ([Jae02b]), que considera a aproximação de RC com sistemas lineares, foi definido que a *Busca RS* trabalha com redes com topologia fixa e composta apenas das conexões obrigatórias (entrada e *reservoir*, *reservoir* e *reservoir* e *reservoir* e saída) e raio espectral máximo igual a 1. A função de ativação dos neurônios é a função *tanh* e a função de treinamento da camada linear de saída (*readout* é a *pseudo-inversa*).

3.3.1 Função de Aptidão

A função de aptidão adotada pela *Busca RS* é igual à função *aptidão* adotada por RCDESIGN e apresentada na equação (3.4).

3.3.2 Representação das Soluções

Seja P^n uma coleção (população) dos vetores s^i, onde n representa uma geração do algoritmo genético e s^i representa um indivíduo da população. O valor máximo de n é o número máximo de gerações (parâmetro N_G) e o tamanho do conjunto P^n é definido pelo parâmetro T_P do algoritmo. A notação s^i_j denota a característica (gene) j do indivíduo i.

- s^i_1 - Define o tamanho das matrizes de pesos $\mathbf{W}$ e $\mathbf{W}^{in}$. $[50;200]$.
- s^i_2 - Define o raio espectral. $[0,7;1]$.
- s^i_3 - Define a conectividade das matrizes de pesos $\mathbf{W}$ e $\mathbf{W}^{in}$. $[10\%;100\%]$.

3.3.3 Descrição do algoritmo

A *Busca RS* utiliza a seleção *uniforme estocástica* (ver Apêndice A.3) e cria três tipos de filhos para a próxima geração: a) Elite, b) Cruzamento e c) Mutação.

A *Busca RS* trabalha com o cruzamento uniforme [Syw89]. No cruzamento uniforme uma máscara, de mesmo tamanho do cromossomo, é criada de forma aleatória e a paridade dos *bits* na máscara indicam os pontos do cromossomo pai onde as informações serão trocadas.

O operador de mutação utilizado pela *Busca RS* (ver Apêndice A.4.2) é aplicado aos indivíduos com uma probabilidade dada pela *taxa de mutação*. A *taxa de mutação* é geralmente pequena, pois uma taxa muito alta torna a busca essencialmente aleatória.

Algoritmo 3: Pseudo-código da ***Busca RS***

```
Criar Cria uma população inicial e aleatória, P^0, com tamanho T_P;
Chamar função CriaIndividuosRS;
n = 1;
enquanto (n ≤ N_G) e não convergir faça
    Selecionar indivíduos da população P^(n−1) (pais);
    Gerar indivíduos da população P^n por elite, cruzamento, e mutação;
    Chamar função CriaIndividuosRS;
    n = n + 1;
Retornar a melhor solução de P^n;
Criar rede (Reservoir Computing) de acordo a melhor solução;
Calcular os erros no conjunto de teste;
```

Algoritmo 4: Pseudo-código da função ***CriaIndividuosRS***

```
enquanto i ≤ T_P (tamanho da população) faça
    Criar rede (Reservoir Computing) de acordo com s^i;
    Calcular autovalores da matriz W;
    Dividir W pelo maior valor absoluto dos seus autovalores;
    Multiplicar W pelo raio espectral definido em s_2^i;
    enquanto fold ≤ 10 (validação cruzada) faça
        Criar conjunto de treinamento (nove partições) e conjunto de validação
        (uma partição);
        Simular rede com conjunto de treinamento;
        Treinar pesos da camada linear de saída (readout);
        Calcular erros no conjunto de treinamento;
        Calcular erros no conjunto de validação;
    Calcular o valor de aptidão de s^i;
    i = i + 1;
```

3.3.4 Parâmetros

Os parâmetros de tamanho da população, número de gerações e número de elite que foram escolhidos para RCDESIGN foram repetidos para a *Busca RS*.

Foram realizados experimentos aleatórios para definição dos parâmetros de probabilidade de cruzamento e taxa de mutação. A configuração dos parâmetros que apresentou menor erro nos testes experimentais foi escolhida e é apresentada a seguir:

- Probabilidade de Cruzamento ($Prob_C$) – 85%
- Taxa de Mutação ($Taxa_M$) – 3%

3.4 Método para Comparação: *Busca TR*

Este método também foi desenvolvido com o objetivo de comparar os resultados de RCDESIGN, porém a *Busca TR* é um método intermediário entre a *Busca RS* e RCDESIGN. A *Busca TR* trabalha da mesma forma que a *Busca RS*, otimizando apenas os parâmetros de geração de RC, porém ela desconsidera a aproximação de RC com sistemas lineares da mesma forma que RCDESIGN. Os parâmetros utilizados são o *número de neurônios no reservatório, raio espectral, percentual de conexões e conexões opcionais*. Neste caso, o raio espectral máximo é igual a $1,5$; valor superior ao sugerido por Jaeger em [Jae02b]). A função de ativação dos neurônios é a função *tanh* e a função de treinamento da camada linear de saída (*readout* é a *pseudo-inversa*).

A *Busca TR* utiliza o mesmo algoritmo, mesmos operadores de cruzamento e mutação, e mesma função de aptidão da *Busca RS* (ver Seção 3.3). A diferença dos dois métodos se concentra na representação das soluções.

3.4.1 Representação das Soluções

Seja P^n uma coleção (população) dos vetores s^i, onde n representa uma geração do algoritmo genético e s^i representa um indivíduo da população. O valor máximo de n é o número máximo de gerações (parâmetro N_G) e o tamanho do conjunto P^n é definido pelo parâmetro T_P do algoritmo. A notação s^i_j denota a característica (gene) j do indivíduo i.

- s^i_1 - Define o tamanho das matrizes de pesos $\mathbf{W}$, $\mathbf{W}^{in}$, $\mathbf{W}^{bias}$ e $\mathbf{W}^{back}$. $[50;200]$.
- s^i_2 - Define o raio espectral. $[0,7;1,5]$.

- s_3^i - Define a conectividade (percentual) das matrizes de pesos $\mathbf{W}$, $\mathbf{W}^{in}$, $\mathbf{W}^{bias}$ e $\mathbf{W}^{back}$. $[10\%; 100\%]$.
- s_4^i - Conexão entre a entrada e saída. Se 1 há conexão, se 0 não há.
- s_5^i - Conexão entre bias e saída. Se 1 há conexão, se 0 não há.
- s_6^i - Conexão de realimentação na saída. Se 1 há conexão, se 0 não há.
- s_7^i - Conexão entre bias e *reservoir*. Se 1 há conexão, se 0 não há.
- s_8^i - Conexão entre saída e *reservoir*. Se 1 há conexão, se 0 não há.

3.4.2 Parâmetros

Os parâmetros de tamanho da população, número de gerações e número de elite que foram escolhidos para RCDESIGN foram repetidos para a *Busca TR*.

Foram realizados experimentos aleatórios para definição dos parâmetros de probabilidade de cruzamento e taxa de mutação. A configuração dos parâmetros que apresentou menor erro nos testes experimentais foi escolhida e é apresentada a seguir:

- Probabilidade de Cruzamento ($Prob_C$) – 85%
- Taxa de Mutação ($Taxa_M$) – 5%

4

Método Experimental

Neste capítulo são descritas as bases de dados utilizadas nas simulações e as principais características dos experimentos realizados. Os experimentos foram desenvolvidos com sete séries clássicas com o objetivo de realizar a validação acadêmica da aplicação do método RCDESIGN à tarefa de previsão de séries temporais e um estudo de caso de previsão da velocidade média horária dos ventos na região nordeste do Brasil com três bases diferentes.

- Séries Clássicas
  - Série NARMA ordem 10 e ordem 30;
  - Série *Mackey-Glass* caos médio e caos moderado;
  - Série *Multiple Sinewave Oscillator* (MSO);
  - Série Natural do Brilho da Estrela (STAR);
  - Série Financeira (*Dow Jones Industrial Average*).
- Estudo de Caso de Previsão da Velocidade Média Horária dos Ventos na Região NE do Brasil
  - Série de Belo Jardim (BJD);
  - Série de São João do Cariri (SCR);
  - Série de Triunfo (TRI).

4.1 Bases de Dados

4.1.1 Séries Clássicas

- **Séries Narma ordem 10 e Narma ordem 30**

  A série NARMA (Nonlinear Autoregressive Moving Average) é uma série discreta formada pela equação (4.1):

$$y(t+1) = 0.3y(t) + 0.05y(t)[\sum_{i=0}^{k-1} y(t-i)] + 1.5u(t-(k-1)) * u(t) + 0.1, \quad (4.1)$$

  onde a entrada $u(t)$ do sistema é um ruído aleatório uniforme, t é o tempo, k é a ordem do sistema e y é a saída. A propriedade interessante deste sistema é que a saída corrente depende tanto da entrada corrente quanto do histórico das entradas. A modelagem com este sistema é geralmente muito difícil devido à função não-linear e também à necessidade de uma memória longa, isto é y pode depender de muitas entradas e saídas [Lie04]. Como entrada, foi utilizado $u(k)$ aleatório, gerado de uma distribuição uniforme no intervalo [0, 1] e os valores de ordem (k) utilizados foram $k = 10$ e $k = 30$.

- **Séries *Mackey-Glass* caos médio e *Mackey-Glass* caos moderado**

  A série *Mackey-Glass* é uma série temporal contínua, unidimensional e apresenta oscilações quase periódicas. Ela é um *benchmark* padrão para teste de previsão de séries temporais. A série é dada pela equação (4.2):

$$y(t+1) = \frac{0.2y(t-\tau)}{1+y(t-\tau)^{10}} - 0.1y(t), \quad (4.2)$$

  onde $y(t)$ é a saída no tempo t, τ é um parâmetro de atraso que influencia no nível de caos da série. O sistema é caótico sempre que $\tau > 16.8$. Este trabalho usa valores de τ igual a 17 para gerar a série com "caos médio"e $\tau = 30$ para gerar a série com "caos moderado", onde 17 é mais fácil que 30. Embora a saída não dependa de nenhuma entrada, utilizou-se uma entrada bias, que é usada simplesmente pela rede na sua rotina de atualização dos estados.

- *Multiple Sinewave Oscillator* (MSO)

A série *Multiple Sinewave Oscillator* (MSO) é gerada pela equação (4.3). Ela é utilizada para criar um sistema de geração de múltiplos sinos.

$$d(n) = sin(0.2 * n) + sin(0.311 * n), \tag{4.3}$$

onde $d(n)$ é o valor do sinal no tempo n.

- **Série natural do Brilho da Estrela (STAR)**

  A série natural do brilho da estrela (STAR) consiste de 600 observações sucessivas à meia noite. Esta série esta disponível em http://robjhyndman.com/TSDL/physics/.

- **Série financeira (*Dow Jones Industrial Average*)**

  Esta série corresponde às observações diárias do Índice *Dow Jones Industrial Average*. Embora o índice exista desde 12 de Janeiro de 1906, os dados utilizados neste trabalho correspondem às observações diárias deste índice, no período de 02 Janeiro de 1998 a 26 de Agosto de 2003, constituindo uma base com de 1.444 pontos. Esta série está disponível em http://www.djindexes.com/.

4.1.2 Previsão da Velocidade Média Horária dos Ventos na Região Nordeste do Brasil

Sistemas de geração com predominância hidráulica, como o brasileiro, possuem acoplamento temporal e espacial, o que torna o planejamento da operação um problema de grande porte [AJL+10]. A principal ferramenta para a realização deste planejamento é o despacho de geração [Aqu01], que é a alocação ótima de sistemas geradores de energia para atender uma determinada demanda de energia. Dessa forma, com a inserção de um parque gerador eólico, o despacho de geração passa a considerar, além das fontes geradoras hidráulicas e térmicas, também a fonte eólica.

Como a geração eólica é função basicamente da velocidade dos ventos [AJL+10], a integração de modelos eficientes de previsões de velocidades de ventos ao controle e operação do sistema elétrico pode reduzir as dificuldades de operação de um sistema composto por fontes tradicionais de energia e a fonte eólica. A potência convertida por uma turbina eólica pode ser expressa através da equação (4.4):

$$P_{turbina} = \frac{1}{2} * \rho * A * v^3 * C_p(\lambda, \theta), \tag{4.4}$$

onde $P_{turbina}$ é a potência elétrica instantânea fornecida por uma turbina eólica em watts, v é a velocidade do vento, A é a área varrida pelas pás em m^2, ρ é a densidade do ar em kg/m^3 e $C_p(\lambda, \theta)$ é o coeficiente de potência (um parâmetro que pode ser calculado ou estimado para uma turbina eólica com base na eficiência da conversão eólica e eletromecânica).

Foram escolhidas três séries de velocidade dos ventos para realizar os experimentos de aplicação do modelo proposto ao problema de previsão da velocidade média dos ventos em curto prazo na região Nordeste do Brasil. Essas séries foram obtidas no site do projeto SONDA (Sistema de Organização Nacional de Dados Ambientais) [end10c]. O projeto SONDA é um projeto do Instituto Nacional de Pesquisas Espaciais (INPE) que implementa uma infraestrutura física e humana para desenvolver um banco de dados dos recursos de energia solar e eólica no Brasil e disponibiliza esses dados de forma gratuita para pesquisas.

- **Série de Belo Jardim** - Município de Pernambuco localizado a uma latitude 0820′08” sul e a uma longitude 3625′27” oeste, estando a uma altitude de 608 metros. A estação instalada pelo projeto SONDA está a 718 metros de altitude. Esta série é constituída pelos valores médios horários obtidos pela central eólica de Belo Jardim, entre 01 de Julho de 2004 e 31 de Dezembro de 2005, com um total de 13.176 padrões.

- **Série de São João do Cariri** - Município da Paraíba, localizado na região do semiárido. Cidade da região da Borborema, microrregião Cariri Oriental. Está localizada a 458m de altitude, porém a estação instalada pelo projeto SONDA está a 718 metros de altitude. Esta série é constituída pelos valores médios horários obtidos pela central eólica de São João do Cariri, entre 01 de Janeiro de 2006 e 31 de Dezembro de 2007, com um total de 17.520 padrões.

- **Série de Triunfo** - Município de Pernambuco localizado a uma latitude 0750′17” sul e a uma longitude 3806′06” oeste, com altitude média de 1004 metros, o que proporciona temperaturas mais baixas que a média da Região. A estação instalada pelo projeto SONDA está a 1123 metros de altitude. Esta série é constituída pelos valores médios horários obtidos pela central eólica de Triunfo, entre 01 de Julho de 2004 e 31 de Dezembro de 2006, com um total de 21.936 padrões.

As bases de dados foram analisadas, inicialmente para definição da janela de tempo, com relação ao parâmetro de autocorrelação dos dados das séries de ventos. Como podemos ver na Figura 4.1, a base de dados de Belo Jardim apresenta uma alta autocorrelação nos passos $1,2,3...$ que vai decrescendo até o passo 12 e depois volta a subir, alcançando um novo limite superior no passo 24. O comportamento se repete para os passos 25 a 48, com índices de autocorrelação menores. A Figura 4.2 apresenta a autocorrelação da série de velocidade média horária dos ventos de João do Cariri, onde verificamos o mesmo comportamento apresentado pela base de Belo Jardim. A Figura 4.3 apresenta a autocorrelação da série de velocidade média horária dos ventos de Triunfo, porém neste caso a curva de autocorrelação é mais suave do que a apresentada pelas outras séries.

Com base na análise da autocorrelação e também com base em um bom intervalo para o planejamento da operação, adotou-se um modelo para previsão dessas séries em uma janela de tempo de 24 horas.

As bases foram pré-processadas e os valores das velocidades médias horárias foram transformadas para o intervalo [0,1] usando a equação (4.5).

$$y = (y_{max} - y_{min}) * (x - x_{min}) / (x_{max} - x_{min}) + y_{min}, \tag{4.5}$$

onde y é o valor de velocidade média horária transformado para o intervalo [0,1], y_{max} é o valor máximo do intervalo, ou seja 1, y_{min} é o valor mínimo do intervalo, ou seja 0, x_{max} e x_{min} são os valores máximo e mínimo de velocidade média horária encontrada na série de ventos e x é valor original de velocidade média horária. O valor máximo utilizado para normalização foi acrescido de 20% em todas as 3 bases.

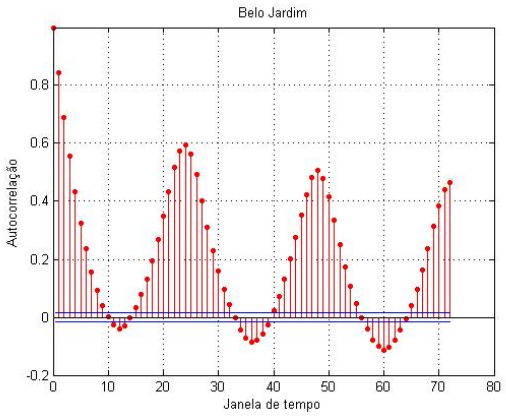

Figura 4.1 Análise da autocorrelação da série de Belo Jardim.

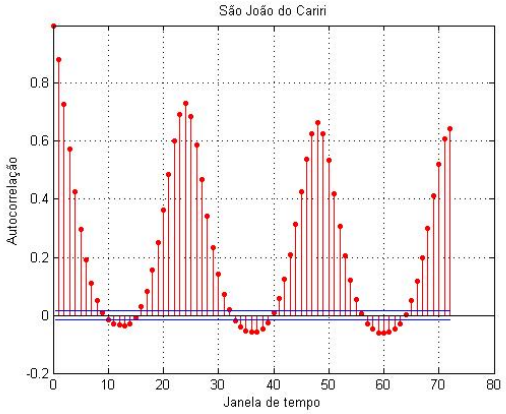

Figura 4.2 Análise da autocorrelação da série de São João do Cariri.

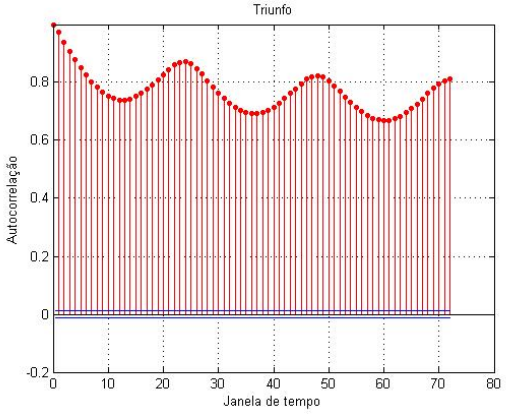

Figura 4.3 Análise da autocorrelação da série de Triunfo.

4.2 Abordagens para Comparação dos Métodos

Neste trabalho utilizamos duas abordagens diferentes para comparar o método RCDESIGN com a *Busca RS* e com a *Busca TR*. Os métodos serão comparados em relação à complexidade computacional e em relação aos erros de previsão.

4.2.1 Comparação pela Complexidade Computacional

O termo complexidade, no contexto de algoritmos, refere-se aos requerimentos de recursos necessários para que um algoritmo possa resolver um problema sob o ponto de vista computacional, ou seja, à quantidade de trabalho despendido pelo algoritmo [TV01]. Quando o recurso é o tempo, são escolhidas uma ou mais operações fundamentais e então contados os números de execuções desta operação fundamental na execução do algoritmo.

A complexidade do tempo de um problema é o número de passos que se toma para resolver uma instância de um problema, a partir do tamanho da entrada utilizando o algoritmo mais eficiente à disposição. Intuitivamente, caso se tome uma instância com entrada de longitude n que pode resolver-se em n^2 passos, se diz que esse problema tem uma complexidade em tempo de n^2. Para não ter que falar do custo exato de um cálculo se utiliza a notação assimptótica. Quando um problema tem custo dado em tempo O(n^2) em uma configuração de computador e linguagem, este custo será o mesmo em todos os computadores, de maneira que esta notação generaliza a noção de custo independentemente do equipamento utilizado.

Através da análise dos algoritmos propostos pelos métodos RCDESIGN (Seção 3.2), *Busca RS* (Seção 3.3) e *Busca TR* (Seção 3.4) pode-se perceber que existem três diferenças básicas entre eles:

- Tamanho dos indivíduos (T_i)
- Taxas dos operadores genéticos (T_{op})
- Reescalonamento da matriz de pesos da camada intermediária (W) pelo raio espectral.

Segundo [dA98] a complexidade de um algoritmo genético depende do tamanho do indivíduo (T_i), do número de gerações (N_G), do tamanho da população (T_P) e das taxas (ou probabilidade) dos operadores genéticos (T_{op}). A complexidade do procedimento criação da população pode ser definida de forma simplificada como O($T_P T_i$) e a complexidade do

procedimento de reprodução como O($N_G T_P T_i T_{op}$). A diferença do tamanho dos indivíduos nos três métodos implica em diferença na complexidade dos procedimentos de criação da população inicial e também nas operações de reprodução (cruzamento e mutação). Os indivíduos de RCDESIGN possuem tamanho variável (entre 2.660 e 40.610) e os indivíduos da *Busca RS* e da *Busca TR* possuem tamanho fixo e respectivamente iguais a 3 e 8.

O reescalonamento da matriz de pesos da camada intermediária (W) pelo raio espectral diz respeito à etapa de geração de RC. Esta operação possui um alto custo computacional, pois o processo envolve o cálculo de todos os autovalores da matriz W, a divisão da matriz W pelo maior valor absoluto entre seus autovalores e finalmente a multiplicação W pelo raio espectral. Segundo [ABBea99] a complexidade de tempo para o cálculo de um auto valor de uma matriz (nxn) é O(n^3), porém o procedimento de reescalonamento necessita de todos os autovalores (n), logo para o cálculo de todos os autovalores a complexidade é igual a O(n^4). Então definimos, de forma simplificada, que a complexidade do reescalonamento da matriz de pesos é O($N_G T_P n^4$), sendo N_G o número de gerações, T_P o tamanho da população e n a quantidade de neurônios na camada intermediária (entre 50 e 200). A Tabela 4.1 apresenta as principais diferenças dos três métodos em relação à complexidade computacional.

Tabela 4.1 Comparação da complexidade computacional

	RCDESIGN	*Busca RS*	*Busca TR*
Criação da População Inicial	O($T_i T_P$)	O($T_i T_P$)	O($T_i T_P$)
Reprodução	O($N_G T_P T_i T_{op}$)	O($N_G T_P T_i T_{op}$)	O($N_G T_P T_i T_{op}$)
Reescalonamento da matriz de pesos	-	O($N_G T_P n^4$)	O($N_G T_P n^4$)

Poderíamos supor que um método de busca dos pesos de RC fosse mais custoso computacionalmente, devido ao grande espaço de busca, do que métodos de busca dos parâmetros de geração de RC, porém de acordo com a Tabela 4.1, podemos observar que o custo associado ao grande espaço de busca é compensado pelo fato do método não possuir o custo associado com reescalonamento da matriz de pesos da camada intermediária (W) pelo raio espectral.

A complexidade de tempo de cada método será apresentada também no Capítulo 5 como tempo médio gasto pelo método para cada rede. Todos os experimentos foram realizados em um computador com processador Inter(R) Core(TM)2 Quad CPU Q6600, 3.5GB de RAM e 2.4 GHz. O tempo médio é calculado pela equação (4.6):

$$\overline{tempo} = (\frac{1}{N_G T_P} \sum tempo_i), \tag{4.6}$$

onde N_G é o número de todas as gerações executadas pelo algoritmo (inclusive a primeira geração aleatória), T_P é o tamanho da população e $tempo_i$ é o tempo gasto pelo algoritmo desde a criação até a avaliação do individuo i.

4.2.2 Comparação pelos Erros de Previsão

Na previsão de séries temporais, a medida de desempenho mais intuitiva é o próprio erro da previsão que pode ser calculado pela equação (4.7):

$$erro_i = (T_i - L_i), \tag{4.7}$$

onde T_i e L_i são respectivamente o valor real e o valor calculado pelo sistema no tempo i. Porém esta medida não é usada na prática para analisar o desempenho de modelos de previsão, pois ela pode apresentar problemas na comparação dos resultados.

O desempenho dos métodos foi avaliado por diferentes medidas, que são mais precisas que o simples cálculo do erro de previsão. O objetivo de apresentar tantas medidas diferentes é permitir a comparação dos resultados deste trabalho com o resultado de trabalhos de outros autores. As medidas utilizadas são: erro médio quadrático (MSE), especificado na equação (3.5), raiz quadrada do erro médio quadrático normalizado (NRMSE), especificado na equação (4.8), erro médio quadrático normalizado, especificado na equação (4.9), e para as bases reais pela percentagem de erro absoluto (MAPE), especificado na equação 4.10 e pelo erro médio absoluto (MAE), em m/s, especificado na equação (4.11):

$$NRMSE = \frac{1}{N*P} \sum_{i=1}^{P} \sum_{j=1}^{N} sqrt(\frac{(T_{ij} - L_{ij})^2}{var(t)}), \tag{4.8}$$

$$NMSE = \frac{1}{N*P} \sum_{i=1}^{P} \sum_{j=1}^{N} \frac{(T_{ij} - L_{ij})^2}{var(t)}, \tag{4.9}$$

$$MAPE = 100 * \frac{1}{P*N} \sum_{i=1}^{P} \sum_{j=1}^{N} \mid \frac{T_{ij} - L_{ij}}{T_{ij}} \mid, \tag{4.10}$$

$$MAE = \frac{1}{P*N} \sum_{i=1}^{P} \sum_{j=1}^{N} \mid (T_{ij} - L_{ij}) \mid, \tag{4.11}$$

onde P é o número total de padrões no conjunto, N é o número de neurônios na

camada de saída da rede, T_{ij} e L_{ij} são respectivamente valores reais (targets) e os valores calculados pela rede do $i-th$ neurônio da camada de saída e $var(T)$ é a variância dos valores no conjunto de saídas desejadas.

O MSE, NMSE e o NRMSE diferem do MAE e do MAPE por elevarem o erro ao quadrado antes de o agrupar, o que significa que os maiores erros irão penalizar mais a avaliação final. O NMSE e o NRMSE normalizam os resultados tendo em consideração o domínio dos valores de entrada, permitindo desta forma que os resultados obtidos possam ser comparados com os resultados de outras experiências [Fer09]. O MAE é a medida que indica a média do afastamento de todos os valores previstos pelos sistema e o seus valores reais e o MAPE é a medida de precisão da previsão realizada pelo sistema.

4.2.3 Validação Cruzada

É relativamente fácil criar sistemas que comentam poucos erros nos exemplos de treinamento, porém é muito mais difícil obter esses resultados em novos casos. Os sistemas de previsão devem ter seus resultados analisados por uma técnica capaz de estimar a taxa de erro verdadeira, a fim de prever qual será seu desempenho em novos casos. Essas técnicas fornecem a base para comparação do desempenho de diferentes sistemas.

Para análise do desempenho dos métodos apresentados neste trabalho, o método de validação cruzada (*k-fold cross-validation*) foi escolhido. Ao optar pelo o método de validação cruzada, é importante decidir quantas partições serão utilizadas, pois um número grande de partições pode levar a uma taxa de erro menor e a um tempo computacional maior, enquanto que um número pequeno de partições pode levar a uma taxa de erro maior e um tempo computacional menor.

Testes extensivos em diferentes conjuntos de dados, com diferentes técnicas de aprendizado, têm mostrado que 10 é o número de partições que permite a melhor estimativa de erro. Dessa forma, validação cruzada com 10 partições tem se tornado um método padrão em termos práticos [WF00].

4.2.4 Análise Estatística dos Resultados

Outro ponto importante quando comparamos diferentes métodos é a significância estatística dos resultados apresentados. Segundo Wainer [Wai07], as técnicas estatísticas de comparação de conjuntos de medidas devem ser usadas para determinar se existem diferenças significativas entre os resultados de métodos diferentes. O tipo do dado (categórico, intervalar, ordinal, nominal, ...) define que tipo de estatística é possível usar

para sumarizar os dados, e que tipo de teste estatístico deverá ser usado para verificar se dois conjuntos de dados são significativamente diferentes ou não.

Testes de hipóteses são procedimentos estatísticos que fazem uma pressuposição sobre os dados, a chamada hipótese nula, e mais uma série de outras pressuposições (condições do teste) sobre os dados, e calculam a probabilidade que alguma propriedade relacionada aos dados seja verdadeira, dadas as pressuposições [Wai07]. Essa probabilidade calculada é chamada valor p ou ($p-value$). Se as condições do teste são verdadeiras, e o $p-value$ é suficientemente baixo, então se pode assumir que a hipótese nula é falsa, ou como é normalmente fraseado, "há evidencias suficientes para rejeitar a hipótese nula". O valor do $p-value$ abaixo do qual se assume que a hipótese é falsa é usualmente $0,05$ ou $0,01$. Se o valor de corte é $0,05$, então diz-se que a significância do teste é $(1-0,05)$, ou 95%. Se o valor de corte é $0,01$, diz-se que a significância do teste é 99%. Se $p-value$ calculado é maior que o valor de corte, então a conclusão que se tira é que "não há evidências para rejeitar a hipótese nula".

Os testes de hipótese podem ser bilaterais, geralmente utilizados para testar se os valores são estatisticamente iguais, ou unilaterais, usados para comparar se os valores são estatisticamente maiores ou menores.

O teste de hipóteses utilizado para comparação dos métodos apresentados neste trabalho é o teste *t* de *Student*. O teste *t* de *Student* é um teste de hipóteses paramétrico, sendo os seus pressupostos de que as duas variáveis envolvidas são intervalares e que ambas tem distribuição normal. Sendo tais pressupostos verdadeiros, a variável analisada resultante pode ser matematicamente transformada, de modo a formar a distribuição *t* de *Student*; em função da qual se determina se um dado resultado é ou não estatisticamente significativo.

Foram realizados testes pareados dos resultados obtidos nos experimentos realizados com os três métodos para cada base de dados com significância de $0,05$. O teste bilateral foi executado entre RCDESIGN e a *Busca RS* e também entre RCDESIGN e a *Busca TR* para verificar se as médias eram estatisticamente iguais.

A comparação do desempenho dos métodos também pode ser realizada de forma visual, através do gráfico *Boxplot*. Este gráfico apresenta os resultados de cada método em caixas, onde a marca central é a mediana, as bordas da caixa são os percentis 25 e 75 e os pontos considerados outliers são apresentados individualmente no gráfico.

4.2.5 Comparação com o Método da Persistência

O método da persistência ou método ingênuo é utilizado também para comparação dos resultados de previsão das bases de velocidade média horária dos ventos. Os resultados são comparados apenas no conjunto de teste, já que não existe um "treinamento"do modelo da persistência.

O método da persistência assume que as condições de tempo da previsão não mudam, ou seja, o método apenas repete o valor realizado como valor previsto. Por exemplo, se é hoje é um dia ensolarado com 40^o *Celsius*, a previsão para amanhã, usando o método da persistência, será dia ensolarado com 40^o *Celsius*. O método da persistência trabalha muito bem quando os padrões climáticos mudam muito lentamente. Ele se adequa bem em lugares como o nordeste do Brasil, onde o clima varia muito pouco de um dia para o outro. Os métodos de persistência são usados normalmente como benchmark de comparação com outros métodos [Wal05].

5

Experimentos e Análise

Este capítulo apresenta, analisa e compara os resultados da aplicação do método proposto, RCDESIGN (Reservoir Computing Design e Treinamento), aplicado à tarefa de previsão de séries temporais. Os métodos utilizados nos experimentos são descritos no Capítulo 3 e as bases de dados são descritas no Capítulo 4. Os resultados alcançados por RCDESIGN são comparados com os resultados obtidos pela *Busca RS*, e também com os resultados obtidos pela *Busca TR*. Os resultados alcançados por RCDESIGN também são comparados com resultados de outros trabalhos com o objetivo de comprovar a viabilidade deste método.

A Seção 5.2 apresenta o desempenho médio dos três métodos em 30 inicializações para todas as bases de dados, o MSE detalhado por iteração pode ser analisado no Apêndice C. A Seção 5.3 apresenta a análise da complexidade computacional dos três métodos, a Seção 5.5 analisa os resultados alcançados pelo método RCDESIGN e a Seção 5.4 apresenta um dos sistemas criados pelos três métodos para todas as base de dados.

5.1 Fluxo Básico do Experimento

Todos os experimentos foram realizados utilizando o fluxo apresentado nesta seção. Devido às características aleatórias dos três métodos foram realizadas 30 inicializações dos mesmos em todas as bases de dados, dessa forma consideramos que um experimento consiste de 30 inicializações dos três métodos (RCDESIGN, *Busca RS* e *Busca TR*) para uma mesma base de dados.

As bases de dados foram divididas em 2 conjuntos: treinamento e teste. O conjunto de treinamento possui 75% dos padrões de cada série e o conjunto de teste possui os 25% restante. Como estamos utilizamos o método de validação cruzada, com 10 partições, para

avaliar o desempenho dos sistemas, foi necessário subdividir os dados do treinamento em 10 partições para a criação do conjunto de treinamento e do conjunto de validação.

O algoritmo que controla a realização dos experimentos é descrito a seguir:

Algoritmo 5: Pseudo-código do experimento

```
Criar base de dados;
Dividir a base de dados em duas partes, 75% para o treinamento e os últimos 25%
da série para teste;
Criar 10 partições (validação cruzada);
Salvar partições e dados do teste;
n = 1;
enquanto (n ≤ 30) faça
    Executar RCDESIGN;
    Executar Busca RS;
    Executar Busca TR
Calcular desempenho médio de RCDESIGN;
Calcular desempenho médio da Busca RS;
Calcular desempenho médio da Busca TR;
```

5.2 Análise do Desempenho

5.2.1 Resultados no Treinamento

Esta seção apresenta o desempenho dos três métodos no treinamento, em relação às medidas de MSE, NMSE e NRMSE, e também a comparação estatística entre o MSE de RCDESIGN e o MSE da *Busca RS* e da *Busca TR*. Devido às características aleatórias dos métodos, os valores apresentados para cada medida são valores médios obtidos pelos sistemas criados nas 30 inicializações e os valores entre parêntesis são os valores de desvio padrão.

Observamos na Tabela 5.1, que o método RCDESIGN apresentou os melhores resultados em todas as bases de dados, exceto para a base Narma ordem 30, onde o método RCDESIGN não foi superior ao método da *Busca TR*.

Para afirmar se as diferenças entre os métodos são estatisticamente significativas, a Tabela 5.2 apresenta o resultado do teste *t* de *Student* pareado, com significância de 5%, entre o MSE de RCDESIGN e da *Busca RS* e também entre o MSE de RCDESIGN e da

[2]O *p-value* é a probabilidade de observar um resultado, dado que a hipótese nula é verdadeira. Valores pequenos para o *p-value* laçam dúvidas sobre a validade da hipótese nula.

Tabela 5.1 MSE no Treinamento - 30 inicializações

	RCDESIGN	*Busca RS*	*Busca TR*
Narma ordem 10	$0,00008852(0,0001)$	$0,00190439(0,0003)$	$0,00046076(0,0001)$
Narma ordem 30	$0,00049597(0,0002)$	$0,00197742(0,0004)$	$0,00047873(0,0001)$
MGS médio	$0,00000001(0,0000)$	$0,00000160(0,0000)$	$0,00000013(0,0000)$
MGS moderado	$0,00000006(0,0000)$	$0,00000788(0,0000)$	$0,00000044(0,0000)$
MSO	$0,00000000(0,0000)$	$0,00018657(0,0001)$	$0,00000051(0,0000)$
STAR	$0,00010459(0,0000)$	$0,00031142(0,0000)$	$0,00017808(0,0000)$
Down-Jones	$0,00076126(0,0000)$	$0,00154268(0,0000)$	$0,00076718(0,0000)$
BJD	$0,00498988(0,0000)$	$0,01273617(0,0001)$	$0,00540878(0,0001)$
SCR	$0,00371390(0,0000)$	$0,01097276(0,0001)$	$0,00424655(0,0001)$
TRI	$0,00087448(0,0000)$	$0,00457273(0,0001)$	$0,00092412(0,0000)$

Busca TR. Na Tabela 5.2 podemos observar o tipo do teste de hipótese utilizado, o *p-value* [2] e a decisão obtida em cada comparação. De acordo com resultados apresentados, é possível rejeitar a hipótese que o MSE de RCDESIGN é igual ao MSE da *Busca RS* para todas as bases de dados no treinamento. Na comparação entre o MSE de RCDESIGN e o MSE da *Busca TR* é possível rejeitar a hipótese que o desempenho dos dois métodos, para todas as bases de dados, é igual no treinamento, exceto para a base Narma ordem 30, onde o teste de hipótese com significância de 5%, não rejeitou a hipótese de desempenho igual.

Tabela 5.2 Comparação estatística - Métodos são iguais no treinamento?

$H_0 : (MSE_{RCDESIGN} - MSE_{Metodo}) = 0$ $H_1 : (MSE_{RCDESIGN} - MSE_{Metodo}) \neq 0$				
	$Metodo = Busca\ RS$		$Metodo = Busca\ TR$	
	Decisão	$p-value$	Decisão	$p-value$
Narma ordem 10	Rejeita H_0	$4,41E-24$	Rejeita H_0	$9,51E-19$
Narma ordem 30	Rejeita H_0	$1,38E-16$	Aceita H_0	$6,77E-01$
MGS médio	Rejeita H_0	$1,36E-24$	Rejeita H_0	$2,29E-20$
MGS moderado	Rejeita H_0	$1,23E-23$	Rejeita H_0	$9,38E-25$
MSO	Rejeita H_0	$6,57E-17$	Rejeita H_0	$6,54E-11$
STAR	Rejeita H_0	$6,96E-21$	Rejeita H_0	$1,18E-19$
Down-Jones	Rejeita H_0	$6,81E-53$	Rejeita H_0	$6,81E-04$
BJD	Rejeita H_0	$1,43E-53$	Rejeita H_0	$1,30E-24$
SCR	Rejeita H_0	$9,41E-55$	Rejeita H_0	$3,38E-25$
TRI	Rejeita H_0	$5,29E-56$	Rejeita H_0	$5,93E-18$

Tabela 5.3 NMSE no Treinamento - 30 inicializações

	RCDESIGN	*Busca RS*	*Busca TR*
Narma ordem 10	0,00734622(0,0071)	0,16179120(0,0214)	0,03861271(0,0065)
Narma ordem 30	0,04217974(0,0175)	0,16622939(0,0295)	0,04083304(0,0065)
MGS médio	0,00000029(0,0000)	0,00003437(0,0000)	0,00000287(0,0000)
MGS moderado	0,00000097(0,0000)	0,00011741(0,0000)	0,00000656(0,0000)
MSO	0,00000000(0,0000)	0,00018633(0,0001)	0,00000051(0,0000)
STAR	0,00153967(0,0002)	0,00459180(0,0006)	0,00262400(0,0002)
Down-Jones	0,01717322(0,0002)	0,03481399(0,0002)	0,01730801(0,0000)
BJD	0,25726089(0,0018)	0,65650251(0,0060)	0,27885249(0,0033)
SCR	0,16680545(0,0018)	0,49280906(0,0048)	0,19073040(0,0031)
TRI	0,04711310(0,0003)	0,24621008(0,0027)	0,04979020(0,0007)

A Tabela 5.3 apresenta o NMSE do treinamento dos três métodos em todas as bases. Os resultados do NMSE do treinamento foram equivalentes aos resultados apresentados na Tabela 5.1, onde RCDESIGN foi melhor em todas as bases/métodos, exceto na comparação com a base Narma ordem 30 e o método da *Busca TR*.

A Tabela 5.4 apresenta o NRMSE do treinamento dos três métodos em todas as bases. Neste caso, RCDESIGN foi melhor em todas as bases, tanto na comparação com a *Busca RS* quanto na comparação com a *Busca TR*.

Tabela 5.4 NRMSE no Treinamento - 30 inicializações

	RCDESIGN	*Busca RS*	*Busca TR*
Narma ordem 10	0,08002522(0,0312)	0,40137066(0,0265)	0,19582617(0,0165)
Narma ordem 30	0,20073622(0,0441)	0,40606099(0,0366)	0,20145930(0,0159)
MGS médio	0,00051347(0,0002)	0,00584377(0,0005)	0,00168732(0,0002)
MGS moderado	0,00096043(0,0002)	0,01079309(0,0010)	0,00255599(0,0002)
MSO	0,00000123(0,0000)	0,01348980(0,0021)	0,00069132(0,0002)
STAR	0,03913336(0,0024)	0,06750947(0,0048)	0,05113318(0,0019)
Down-Jones	0,13066480(0,0008)	0,18602903(0,0006)	0,13117511(0,0002)
BJD	0,50716864(0,0017)	0,81022126(0,0037)	0,52802343(0,0031)
SCR	0,40838565(0,0022)	0,70196885(0,0034)	0,43669594(0,0036)
TRI	0,21689316(0,0007)	0,49586402(0,0028)	0,22296342(0,0016)

5.2.2 Resultados nos Testes

Esta seção apresenta o desempenho dos três métodos no teste de todas as bases, a comparação estatística entre o MSE de RCDESIGN com os métodos da *Busca RS* e da *Busca TR* e também comparações entre os resultados de RCDESIGN e os resultados de trabalhos de outros autores para algumas bases. Como foi dito na seção anterior, devido às características aleatórias dos métodos, os valores apresentados para cada medida são valores médios obtidos pelos sistemas criados nas 30 inicializações e os valores entre parêntesis são os valores de desvio padrão. O MSE detalhado por iteração pode ser analisado no Apêndice C. Os valores de MSE, NMSE e NRMSE são apresentados para todas as bases de dados. A comparação com o método da Persistência e os valores de MAPE e MAE, são apresentados apenas para as bases de velocidade média horária dos ventos.

Outra maneira de verificar a diferença do desempenho dos métodos pode ser realizada de forma visual, através do gráfico *Boxplot*. O Apêndice B apresenta o *Boxplot* com o MSE do teste dos três métodos em todas as bases de dados.

Tabela 5.5 MSE no teste - 30 inicializações

	RCDESIGN	*Busca RS*	*Busca TR*
Narma ordem 10	0,00009201(0,0001)	0,00229437(0,0009)	0,00047734(0,0001)
Narma ordem 30	0,00050483(0,0002)	0,00199516(0,0006)	0,00052719(0,0001)
MGS médio	0,00000001(0,0000)	0,00000161(0,0000)	0,00000014(0,0000)
MGS moderado	0,00000007(0,0000)	0,00000831(0,0000)	0,00000045(0,0000)
MSO	0,00000000(0,0000)	0,00018968(0,0001)	0,00000051(0,0000)
STAR	0,00379918(0,0198)	0,00038424(0,0000)	0,00021858(0,0000)
Down-Jones	0,00072697(0,0001)	0,00117886(0,0000)	0,00069018(0,0000)
BJD	0,00396731(0,0000)	0,01137915(0,0001)	0,00445515(0,0001)
SCR	0,00315583(0,0000)	0,01154520(0,0002)	0,00393969(0,0001)
TRI	0,00099199(0,0000)	0,00357787(0,0001)	0,00106601(0,0000)

Como podemos observar na Tabela 5.5, o método RCDESIGN apresentou os melhores resultados em todas as bases de dados, exceto para as bases para as bases *STAR* e *Down-Jones* onde o desempenho de RCDESIGN foi inferior ao desempenho da *Busca TR*. Para afirmar se as diferenças entre os métodos são estatisticamente significativas, a Tabela 5.6 apresenta o resultado do teste *t* de *Student* pareado, com significância de 5%, entre o MSE de RCDESIGN e da *Busca RS* e também entre o MSE de RCDESIGN e da *Busca TR*.

De acordo com resultados apresentados na Tabela 5.6, é possível rejeitar a hipótese que o MSE de RCDESIGN é igual ao MSE da *Busca RS* para todas as bases de dados no teste, exceto para a base *STAR*, onde não é possível afirmar com significância de 5%, que RCDESIGN tem desempenho diferente da *Busca RS*.

Na comparação entre o MSE de RCDESIGN e o MSE da *Busca TR* é possível rejeitar a hipótese que o desempenho dos dois métodos é igual no teste em todas as bases, exceto para as bases Narma ordem 30, *STAR* e *Down-Jones*. Embora o MSE de RCDESIGN seja menor do que o MSE da *Busca TR* para a base Narma ordem 30, não é possível afirmar, com significância de 5%, que RCDESIGN tem desempenho diferente da *Busca TR*. Para as bases *STAR* e *Down-Jones* o MSE de RCDESIGN é um pouco maior que o MSE da *Busca TR* e não é possível afirmar, com significância de 5%, que RCDESIGN tem desempenho diferente da *Busca TR*.

Tabela 5.6 Comparação estatística - Métodos são iguais no teste?

$H_0 : (MSE_{RCDESIGN} - MSE_{Metodo}) = 0$ $H_1 : (MSE_{RCDESIGN} - MSE_{Metodo}) \neq 0$				
	$Metodo = Busca\ RS$		$Metodo = Busca\ TR$	
	Decisão	$p - value$	Decisão	$p - value$
Narma ordem 10	Rejeita H_0	$7,62E-14$	Rejeita H_0	$5,42E-20$
Narma ordem 30	Rejeita H_0	$7,62E-13$	Aceita H_0	$6,16E-01$
MGS médio	Rejeita H_0	$4,51E-21$	Rejeita H_0	$7,33E-19$
MGS moderado	Rejeita H_0	$7,88E-19$	Rejeita H_0	$3,45E-21$
MSO	Rejeita H_0	$5,63E-17$	Rejeita H_0	$5,49E-11$
STAR	Aceita H_0	$3,53E-01$	Aceita H_0	$3,31E-01$
Down-Jones	Rejeita H_0	$2,09E-19$	Aceita H_0	$7,37E-02$
BJD	Rejeita H_0	$9,30E-54$	Rejeita H_0	$2,15E-25$
SCR	Rejeita H_0	$1,01E-51$	Rejeita H_0	$1,06E-25$
TRI	Rejeita H_0	$2,37E-44$	Rejeita H_0	$2,74E-16$

A Tabela 5.7 apresenta o NMSE do teste dos três métodos em todas as bases de dados. Podemos observar nesta tabela que o desempenho de RCDESIGN foi superior ao desempenho dos outros dois métodos em todas as bases de dados, exceto para as bases Narma ordem 30, *STAR* e *Down-Jones* onde o desempenho de RCDESIGN foi inferior ao desempenho da *Busca TR*. É possível também comparar os resultados de NMSE das bases Narma ordem 10 e *Mackey-glass* caos médio com trabalhos de outros autores.

Tabela 5.7 NMSE no teste - 30 inicializações

	RCDESIGN	*Busca RS*	*Busca TR*
Narma ordem 10	$0,00837240(0,0089)$	$0,18194143(0,0395)$	$0,04387851(0,0105)$
Narma ordem 30	$0,04384370(0,0190)$	$0,17399053(0,0380)$	$0,04356045(0,0072)$
MGS médio	$0,00000028(0,0000)$	$0,00003454(0,0000)$	$0,00000289(0,0000)$
MGS moderado	$0,00000104(0,0000)$	$0,00012801(0,0001)$	$0,00000694(0,0000)$
MSO	$0,00000000(0,0000)$	$0,00018593(0,0001)$	$0,00000050(0,0000)$
STAR	$0,04945553(0,2580)$	$0,00500185(0,0006)$	$0,00284537(0,0002)$
Down-Jones	$0,05753908(0,0084)$	$0,09330531(0,0028)$	$0,05462694(0,0010)$
BJD	$0,24236893(0,0024)$	$0,69516923(0,0065)$	$0,27217168(0,0041)$
SCR	$0,16752339(0,0025)$	$0,61286343(0,0081)$	$0,20913412(0,0057)$
TRI	$0,19010669(0,0014)$	$0,68567226(0,0170)$	$0,20429252(0,0043)$

Embora a configuração dos experimentos não seja igual, a Tabela 5.8 apresenta uma comparação dos resultados de médios de NMSE de RCDESIGN com os resultados de outros trabalhos para as mesmas bases de dados. Para a base Narma ordem 10, Jaeger [Jae03] usou um esquema de adaptação online dos pesos da camada de saída (*readout*), baseado no algoritmo *Recursive Least Squares* (RLS). Steil [Ste04] realizou experimentos com o algoritmo *Atiya-Parlos Recurrent Learning* (APRL) e o algoritmo *Backpropagation-DeCorrelation* (BPDC) para treinar RC com quatro bases de dados, uma delas foi a Narma ordem 10. O melhor resultado alcançado por seus experimentos para a base Narma ordem 10 foi utilizando o algoritmo BPDC. Steil [Ste04] também realizou experimentos com o algoritmo APRL e o algoritmo BPDC para treinar sistemas para *Mackey-glass* caos médio. Novamente o melhor resultado alcançado por ele no conjunto de teste foi utilizando o algoritmo BPDC.

A Tabela 5.9 apresenta o NRMSE do teste dos três métodos em todas as bases de dados. Podemos observar nesta tabela que o desempenho de RCDESIGN foi superior ao desempenho dos outros dois métodos em todas as bases de dados, exceto para as bases *STAR* e *Down-Jones* onde o desempenho de RCDESIGN foi inferior ao desempenho da *Busca TR*. É possível também comparar os resultados de NRMSE das bases Narma

Tabela 5.8 Comparação do NSME

	NMSE	Raio Espectral	Tamanho do *Reservoir*
Narma ordem 10			
RCDESIGN	0,0084	1,15(0,08)	174(16,72)
Jaeger [Jae03]	0,0081	-	400
Steil [Ste04]	0,1420	-	40
MGS caos médio			
RCDESIGN	0,00000028	1,59(0,28)	82(31,20)
Steil [Ste04]	0,0340	-	40

ordem 10, Narma ordem 30, *Mackey-glass* caos moderado e MSO, com trabalhos de outros autores.

Tabela 5.9 NRMSE no teste - 30 inicializações

	RCDESIGN	*Busca RS*	*Busca TR*
Narma ordem 10	0,08462155(0,0354)	0,42436505(0,0438)	0,20807155(0,0246)
Narma ordem 30	0,20424126(0,0469)	0,41465769(0,0460)	0,20796290(0,0180)
MGS médio	0,00050628(0,0001)	0,00584469(0,0006)	0,00168976(0,0002)
MGS moderado	0,00099067(0,0002)	0,01114130(0,0020)	0,00260901(0,0004)
MSO	0,00000120(0,0000)	0,01347762(0,0021)	0,00068574(0,0002)
STAR	0,08555901(0,2088)	0,07059365(0,0044)	0,05330699(0,0020)
Down-Jones	0,23938190(0,0156)	0,30542720(0,0045)	0,23371477(0,0021)
BJD	0,49230377(0,0025)	0,83375926(0,0039)	0,52168618(0,0040)
SCR	0,40928555(0,0030)	0,78283935(0,0052)	0,45727073(0,0062)
TRI	0,43600947(0,0016)	0,82799176(0,0103)	0,45196280(0,0048)

Embora a configuração dos experimentos não seja igual, a Tabela 5.10 apresenta uma comparação dos resultados de médios de NRMSE de RCDESIGN com os resultados de outros trabalhos para as mesmas bases de dados. Os sistemas criados por RCDESIGN apresentam raio espectral maior do que o raio utilizado nos trabalhos apresentados na Tabela 5.10, bem como NRMSE foi menor em quase todas as comparações e os sistemas criados por RCDESIGN são pequenos (menos de 100 neurônios), exceto para a base Narma ordem 10, onde em média os sistemas possuem 174 neurônios no *reservoir*.

Verstraeten et al apresentaram em [VSDS07] um estudo sobre a relação entre os parâmetros de RC e a dinâmica do mesmo. Eles realizaram várias configurações diferentes de RC para algumas bases de dados, entre elas a Narma ordem 10. Eles concluíram que para esta base, o tamanho da rede não era o parâmetro mais importante, porém o tipo de

função de ativação e o raio espectral foram parâmetros muito importantes.

Boedecker et al investigaram em [BOMA09] um método para inicialização da matriz de pesos da camada intermediária usando permutação de matrizes. Eles realizaram testes com a base Narma ordem 30 entre outras bases. Schrauwen et al apresentaram em [SWV+08] uma adaptação da regra de *Intrinsic Plasticity* (IP) para neurônios *tanh* para a base Narma ordem 30.

Jaeger [Jae01] utilizou a base *Mackey-Glass* para realizar experimentos de RC com $\tau = 17$ (caos médio) e $\tau = 30$ (caos moderado). Em [JH04] Jaeger e Haas também executaram experimentos para *Mackey-glass* caos médio. Em [WSVS08], Wyffels et al executaram experimentos com neurônios chamados "filtro passa-banda"para as bases *Mackey-Glass* caos médio e *Mackey-Glass* caos moderado.

Em [SWGG07] Schmidhuber et al realizaram experimentos com o método EVOLINO (*EVOlution of recurrent systems with LINear Outputs*) para a previsão da série **MSO**.

Tabela 5.10 Comparação do NRSME

	NMSE	Raio Espectral	Tamanho do *Reservoir*
Narma ordem 10			
RCDESIGN	0,085	1,15(0,08)	174(16,72)
Verstraeten et al [VSDS07]	0,400	1	200
Narma ordem 30			
RCDESIGN	0,2042	1,64(0,18)	87(21,90)
Boedecker et al [BOMA09]	0,4542	0,95	100
Schrauwen et al [SWV+08]	0,4600	-	100
MGS caos médio			
RCDESIGN	0,00051	1,59(0,28)	82(31,20)
Jaeger [Jae01]	0,00012	0,95	400
Jaeger e Haas [JH04]	0,000063	-	1.000
Wyffels et al [WSVS08]	0,0065	-	1.000
MGS caos moderado			
RCDESIGN	0,000991	1,58(0,31)	84(32,51)
Jaeger [Jae01]	0,032	-	400
Wyffels et al [WSVS08]	0,0065	0,98	-
MSO			
RCDESIGN	0,00000120	1,57(0,18)	80(19,68)
Schmidhuber et al [SWGG07]	0,0103	-	400

A Tabela 5.11 apresenta o MAPE do teste do método RCDESIGN, da *Busca RS*, da *Busca TR* e do método da Persistência para as bases de velocidade média horária dos ventos. O desempenho de RCDESIGN foi bem superior ao desempenho da *Busca RS* e

do método da Persistência e foi próximo ao desempenho da *Busca TR*.

Tabela 5.11 MAPE(%) no teste

	RCDESIGN	*Busca RS*	*Busca TR*	*Persistência*
BJD	12,08(0,11)	20,27(0,11)	12,47(0,07)	23,99(0,25)
SCR	13,29(0,28)	24,75(0,43)	13,70(0,51)	25,06(0,61)
TRI	9,86(0,04)	20,63(0,24)	10,23(0,09)	22,46(0,23)

A Tabela 5.12 apresenta o MAE (em metros por segundo) do teste do método RCDESIGN, da *Busca RS*, da *Busca TR* e do método da Persistência para as bases de velocidade média horária dos ventos. Em relação ao MAE, RCDESIGN também foi melhor do que os outros métodos nas três bases, porém os valores de MAE de RCDESIGN foram próximos aos valores de MAE da *Busca TR*.

Tabela 5.12 MAE(m/s) no teste

	RCDESIGN	*Busca RS*	*Busca TR*	*Persistência*
BJD	0,62(0,00)	1,08(0,01)	0,66(0,00)	1,22(0,96)
SCR	0,63(0,00)	1,23(0,01)	0,70(0,01)	1,34(1,05)
TRI	0,87(0,00)	1,71(0,02)	0,90(0,01)	1,92(1,61)

Comparando os resultados de RCDESIGN com o modelo criado por Ferreira et al em [FLdA+08], para a base de Belo Jardim, observamos que RCDESIGN foi melhor. O modelo criado em [FLdA+08] obteve no conjunto de teste MSE, MAPE e MAE respectivamente iguais a 0,0411[3], 18,21 e 0,71 em uma rede com 200 neurônios. O desempenho médio de RCDESIGN para esta base foi melhor com MSE, MAPE e MAE de 0,00397, 12,08 e 0,62.

Comparando com o modelo criado por Ferreira et al em [FL08b], para a base de Triunfo, RCDESIGN foi melhor. O modelo criado em [FL08b] obteve MSE e MAE de 0,0095[3] e 0,88 no conjunto de teste em uma rede com 400 neurônios no *reservoir*, enquanto que RCDESIGN obteve MSE e MAE de 0,000992 e 0,87. Outro modelo criado por Ferreira et al em [FL09] para a base de Triunfo obteve MSE igual a 0,0096[3] no conjunto de teste, em uma rede muito grande, com 547 neurônios no *reservoir* e raio espectral de 0,947. O desempenho médio de RCDESIGN para esta base foi muito melhor com MSE igual a 0,000992.

[3]Valor convertido para MSE.

5.3 Análise da Complexidade Computacional

Esta seção apresenta uma análise da complexidade computacional dos três métodos baseada no tempo médio de criação de uma rede por cada método e também no tempo total de busca de cada método. Os valores apresentados são valores médios obtidos pelos sistemas criados nas 30 inicializações e os valores entre parêntesis são os valores de desvio padrão.

Podemos observar na Tabela 5.13 que o tempo médio, em segundos, de criação dos sistemas com RCDESIGN foi sempre bem inferior ao tempo médio da *Busca RS* e da *Busca TR*, exceto para a base SCR, o que significa mais uma vantagem do método RCDESIGN em comparação com a *Busca RS* e a *Busca TR*, pois, além de alcançar sistemas com baixas taxas de erro, o método RCDESIGN tem um custo computacional inferior ao custo computacional dos outros dois. Como foi explicado na Seção 4.2.2, embora o método RCDESIGN tenha um espaço de busca muito maior do que o espaço de busca da *Busca RS* e da *Busca TR*, pois RCDESIGN trabalha com indivíduos com tamanho de até 40.610 genes, a complexidade de tempo de RCDESIGN é compensada com a ausência das operações para reescalar a matriz de pesos da camada intermediária pelo raio espectral.

Tabela 5.13 Comparação do tempo médio para criação de uma rede (em segundos)

Modelo	RCDESIGN	Busca RS	Busca TR
Narma ordem 10	4,54(0,43)	5,83(0,45)	6,20(0,51)
Narma ordem 30	3,77(0,29)	5,98(0,40)	6,17(0,63)
MGS médio	3,89(0,21)	5,58(0,47)	5,54(0,40)
MGS moderado	3,91(0,20)	5,62(0,52)	5,35(0,42)
MSO	3,83(0,23)	5,84(0,50)	5,52(0,52)
STAR	0,11(0,01)	0,14(0,01)	0,14(0,01)
Down-Jones	0,20(0,01)	0,23(0,02)	0,29(0,02)
BJD	2,57(0,09)	3,62(0,11)	3,35(0,14)
SCR	10,99(23,28)	10,34(1,04)	9,95(1,07)
TRI	5,94(0,17)	7,76(0,16)	7,75(0,26)

Na Tabela 5.13 podemos observar que o método RCDESIGN também apresentou um tempo total de busca, em todas as bases de dados, bem inferior ao tempo total dos outros dois métodos, porém é importante destacar que o tempo total das buscas está relacionado com a complexidade estudada na Seção 4.2.2 e também com o parâmetro de tamanho da população, logo se os métodos utilizassem valores diferentes para este

parâmetro poderíamos observar outros métodos com tempo total de busca inferior ao de RCDESIGN.

Tabela 5.14 Comparação do tempo total de busca em cada método (em minutos)

Modelo	RCDESIGN	Busca RS	Busca TR
Narma ordem 10	77,38(26,51)	127,99(10,29)	134,80(12,37)
Narma ordem 30	54,35(18,34)	128,28(15,68)	135,92(13,81)
MGS médio	31,18(1,68)	44,75(3,78)	44,40(3,21)
MGS moderado	31,38(1,57)	45,06(4,15)	42,88(3,40)
MSO	30,71(1,83)	119,75(24,44)	44,28(4,18)
STAR	1,53(0,44)	2,97(0,45)	2,79(0,58)
Down-Jones	2,17(0,35)	4,58(0,58)	4,31(1,46)
BJD	38,41(11,61)	75,14(11,58)	70,59(10,25)
SCR	125,51(182,54)	225,05(26,17)	212,87(39,74)
TRI	58,59(9,87)	166,84(14,85)	151,21(34,23)

5.4 Apresentação dos Sistemas Criados

Nesta seção são apresentados a o número médio de gerações, o raio espectral médio e o tamanho médio do *reservoir* dos três métodos para todas as bases de dados bem como um dos 30 sistemas criado por RCDESIGN, um dos 30 sistemas criado pela *Busca RS* e um dos 30 sistemas criado pela *Busca TR* para todas as bases de dados. O critério para seleção dos sistemas que serão exibidos a seguir é que eles possuam MSE próximo à média do desempenho do método.

A topologia ou arquitetura dos sistemas é apresentada, nesta seção, graficamente como uma matriz. Nessa matriz todas as conexões partindo do bias (B), entrada (I), *reservoir* (R) e saída (O) são apresentadas na horizontal e chegando ao *reservoir* (R) ou a camada de saída (O) são apresentadas na vertical. Um retângulo branco indica a existência da conexão e um retângulo preto indica a ausência da mesma. Embora RCDESIGN não faça busca do parâmetro raio espectral, o mesmo será exibido em todos os sistemas (independentemente do método) através da exibição da pole dos pesos do *reservoir* no plano complexo.

A Tabela 5.15, a Tabela 5.16 e a Tabela 5.17 apresentam a quantidade média de gerações, o raio espectral médio e o número médio de neurônios no *reservoir* para os três métodos em todas as bases de dados. Os valores médios apresentados nestas tabelas correspondem à média das 30 inicializações de cada método para cada base de dados.

Exceto para a base *STAR* e para a base *Down-Jones*, podemos observar na Tabela 5.15 que o raio espectral médio dos sistemas criados por RCDESIGN foi acima do limite de 1 para a propriedade de *Echo State* [Jae01], e mesmo assim os sistemas apresentaram um excelente resultado na previsão das séries em todas as medidas analisadas (ver Seção 5.2.2). Os bons resultados obtidos pelos sistemas criados por RCDESIGN, mesmo sem a propriedade de *Echo State*, foram explicados por Ozturk e Príncipe em [OP05]. Neste trabalho eles provam que a dinâmica real dos *reservoirs* só pode ser encontrada com o sistema em uso, ou seja, *reservoirs* com raio espectral acima de 1 também podem produzir bons resultados.

Observamos também na Tabela 5.15 que o número de neurônios no *reservoir* foi sempre muito menor do que 200, limite máximo estabelecido para esta variável, exceto para a base Narma ordem 10, onde o tamanho médio do reservoir é 174 neurônios. Os sistemas criados por RCDESIGN foram muito menores do que os sistemas criados pelos outros dois métodos, exceto para a base *Down-Jones*.

Tabela 5.15 RCDESIGN - Resultado Médio por Base de Dados

Modelo	Qtd Gerações	Raio Espectral	Neurônios no Reservoir
Narma ordem 10	8,5	1,15(0,08)	174(16,72)
Narma ordem 30	7,5	1,64(0,18)	87(21,90)
MGS médio	4,0	1,59(0,28)	82(31,20)
MGS moderado	4,0	1,58(0,31)	84(32,51)
MSO	4,0	1,57(0,18)	80(19,68)
STAR	7,1	0,97(0,07)	57(6,71)
Down-Jones	5,3	0,99(0,12)	57(6,68)
BJD	7,4	1,53(0,22)	79(23,09)
SCR	6,8	1,73(0,20)	98(22,89)
TRI	4,5	1,59(0,26)	86(25,91)

Podemos observar na Tabela 5.16 que o raio espectral médio dos sistemas criados pela *Busca RS* são todos menores ou iguais a 1, isso se deve ao fato que os sistemas criados por este método deveriam ter a propriedade de *Echo State* [Jae01], logo o limite para o raio espectral é 1. Mesmo com a propriedade de *Echo State* os sistemas criados pela *Busca RS* não apresentaram desempenho superior ao dos sistemas criados sem esta restrição (ver Seção 5.2.2). Observamos na Tabela 5.16 que o número de neurônios no *reservoir* foi próximo a 200 para todas as bases de dados, exceto para a base *STAR* onde o tamanho médio do *reservoir* é igual a 101 e para a base *Down-Jones* onde o tamanho médio do *reservoir* é igual a 54. A maioria dos sistemas criados pela *Busca RS* é grande, com

tamanho do *reservoir* próximo ao limite máximo de 200 neurônios que foi estabelecido para esta variável neste método.

Tabela 5.16 *Busca RS* - Resultado Médio por Base de Dados

Modelo	Qtd Gerações	Raio Espectral	Neurônios no Reservoir
Narma ordem 10	11,0	0,97(0,05)	196(3,50)
Narma ordem 30	10,7	0,97(0,04)	195(5,38)
MGS médio	4,0	1,00(0,00)	189(8,15)
MGS moderado	4,0	1,00(0,02)	194(7,48)
MSO	10,2	1,00(0,00)	194(5,66)
STAR	10,5	0,99(0,03)	101(12,13)
Down-Jones	10,3	0,70(0,00)	54(3,28)
BJD	10,4	1,00(0,00)	177(23,11)
SCR	10,9	1,00(0,00)	185(13,73)
TRI	10,5	1,00(0,00)	173(21,46)

Podemos observar na Tabela 5.17 que o raio espectral médio dos sistemas criados pela *Busca TR* ficaram próximos a 1, porém para algumas bases o raio espectral foi um pouco maior do que 1. Observamos na Tabela 5.16 que o número de neurônios no *reservoir* foi próximo a 200 para todas as bases de dados, exceto para as bases *STAR* e *Down-Jones*, onde o tamanho médio do reservoir é próximo a 70 neurônios. A maioria dos sistemas criados pela *Busca TR* é grande, com tamanho do *reservoir* próximo ao limite máximo estabelecido para esta variável.

Tabela 5.17 *Busca TR* - Resultado Médio por Base de Dados

Modelo	Qtd Gerações	Raio Espectral	Neurônios no Reservoir
Narma ordem 10	10,9	0,93(0,05)	193(5,39)
Narma ordem 30	11,0	0,92(0,04)	193(6,56)
MGS médio	4,0	1,00(0,05)	187(13,08)
MGS moderado	4,0	0,90(0,06)	182(12,15)
MSO	4,0	0,81(0,08)	187(9,49)
STAR	9,9	0,91(0,11)	71(12,66)
Down-Jones	7,5	1,39(0,35)	62(9,65)
BJD	10,5	1,30(0,11)	123(24,20)
SCR	10,6	1,39(0,09)	178(15,72)
TRI	9,5	1,21(0,09)	174(23,92)

5.4.1 Narma

A base de dados NARMA tem sido utilizada como *benchmark* de tarefas de previsão de séries temporais. Nesta seção apresentamos os resultados de um dos sistemas criados pelos três métodos para a base Narma (ordem 10 e ordem 30). A Figura 5.1 apresenta a topologia e a distribuição dos pesos dos sistemas criados por RCDESIGN, pela *Busca RS* e pela *Busca TR* para base Narma 10 e a Figura 5.2 apresenta as mesmas informações para a base Narma 30.

Podemos observar na Figura 5.1, que os sistemas criados por RCDESIGN (A) e pela *Busca TR* (E) utilizaram, além das conexões obrigatórias ($\mathbf{W}^{in}$ ou I-R ,$\mathbf{W}$ ou R-R e $\mathbf{W}^{out}$ ou R-O), algumas conexões opcionais de RC. Como a *Busca RS* considera a aproximação de RC à sistemas lineares, observamos em (C) e (D) que o sistema criado possui apenas as conexões obrigatórias de RC e raio espectral dentro do limite de 1, que foi sugerido por Jaeger [Jae02b] para a propriedade de *Echo State*. O sistema criado por RCDESIGN (B), apresenta raio espectral igual a $1,2139$, valor acima do limite de 1.

Analisando a Figura 5.2 percebemos que para a base Narma ordem 30, o sistema criado por RCDESIGN (B) também apresenta um raio espectral acima do limite de 1, enquanto que os sistemas criados pela *Busca RS* (D) e pela *Busca TR* (F) apresentam raio espectral dentro deste limite. Com relação à arquitetura, os sistemas criados por RCDESIGN (A) e pela *Busca TR* (E) utilizaram as conexões opcionais de RC.

A Tabela 5.18 apresenta a quantidade de neurônios, a função de ativação dos neurônios, a função de treinamento e o raio espectral dos sistemas criados pelos três métodos para a base Narma ordem 10 e a Tabela 5.19 apresenta as mesmas informações para a base Narma ordem 30.

Para a base Narma ordem 10 (Tabela 5.18), os sistemas criados pelos três métodos são muito parecidos, apresentam quase mesmo tamanho, mesma função de ativação e mesma função de treinamento, porém o raio espectral do sistema criado por RCDESIGN foi bem maior que o raio espectral dos outros dois sistemas.

Tabela 5.18 NARMA ordem 10. Configuração dos Parâmetros

Modelo	Tamanho	Ativação	Treino	Raio Spec.
RCDESIGN	192	*Tanh*	*Pseudo*	$1,2139$
Busca RS	198	*Tanh* [1]	*Pseudo* [1]	1
Busca TR	198	*Tanh* [1]	*Pseudo* [1]	$0,9$

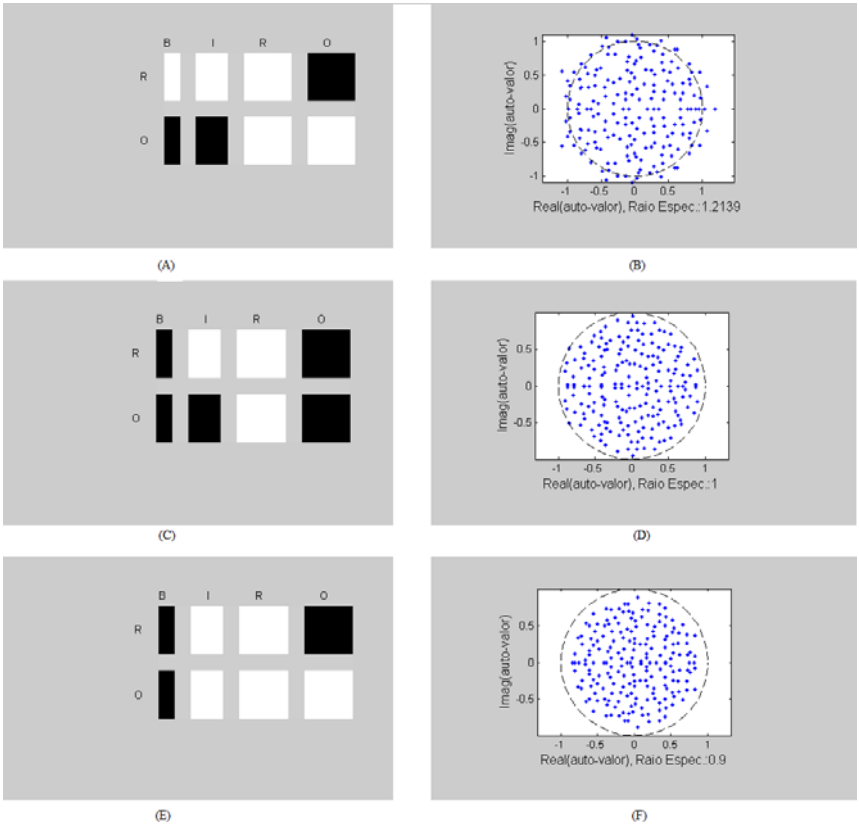

Figura 5.1 Narma ordem 10. RCDESIGN: (A) Topologia e (B) distribuição dos pesos. *Busca RS*: (C) Topologia e (D) distribuição dos pesos. *Busca TR*: (E) Topologia e (F) distribuição dos pesos.

Para a base Narma ordem 30 (Tabela 5.19), os sistemas criados pela *Busca RS* e pela *Busca TR* são maiores do que o sistema criado por RCDESIGN. Os três sistemas utilizam a mesma função de ativação, porém o sistema criado por RCDESIGN usou uma função de treinamento diferente da função utilizada pelos outros dois, além apresentar um raio espectral bem maior do que o raio espectral dos outros dois sistemas.

A Figura 5.3 apresenta graficamente uma comparação entre os últimos 50 pontos da série real e da série prevista com RCDESIGN em (A), com a *Busca RS* (B) e com a *Busca TR* (C) para a base Narma ordem 10, e a Figura 5.4 apresenta as mesmas informações para a base Narma ordem 30.

A Figura 5.5 e a Figura 5.6 apresentam o melhor valor de *aptidão* de cada geração da busca realizada pelos três métodos para as bases Narma ordem 10 e para a base Narma ordem 30 respectivamente. Para as duas bases Narma, observamos que a evolução do

[1] Valor fixo para o método

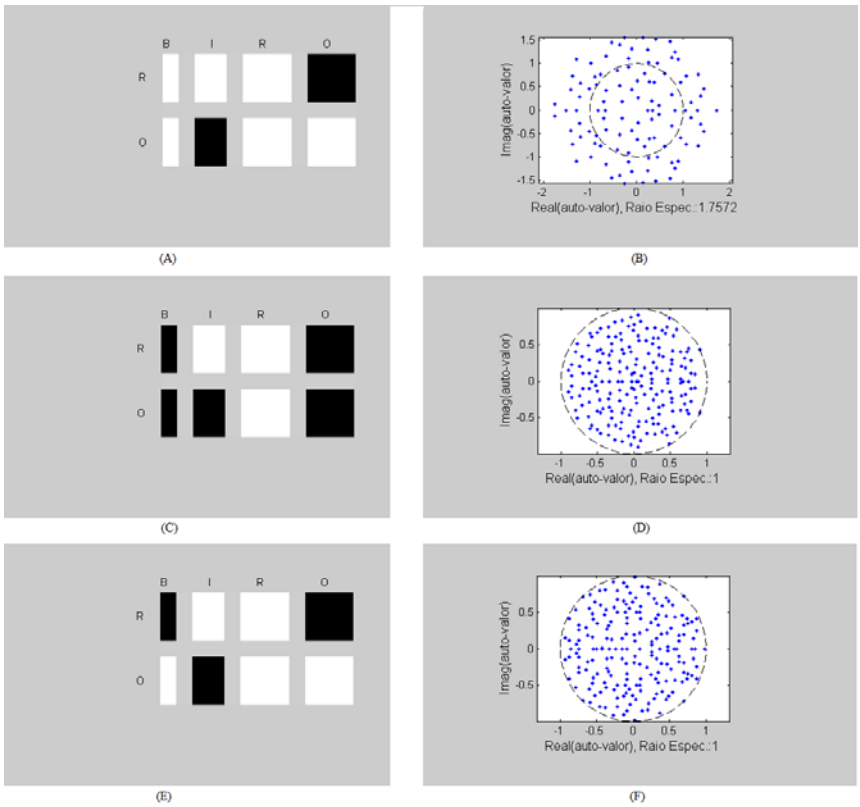

Figura 5.2 Narma ordem 30. RCDESIGN: (A) Topologia e (B) distribuição dos pesos. *Busca RS*: (C) Topologia e (D) distribuição dos pesos. *Busca TR*: (E) Topologia e (F) distribuição dos pesos.

melhor indivíduo pelo método RCDESIGN é mais estável do que a evolução com os outros dois métodos, embora os três métodos utilizem elite igual a 2. Como o método RCDESIGN utiliza o operador de elite e também como ele cria sistemas com os pesos que estão definidos nos cromossomos de cada indivíduo, é possível observar uma evolução regular da aptidão do melhor indivíduo em cada geração de RCDESIGN. A utilização do elitismo na *Busca RS* e na *Busca TR* não garante uma evolução regular do melhor indivíduo em cada geração, pois esses métodos criam sistemas com pesos baseados no raio espectral e como foi dito por Ozturk et al [OXP07], existem várias matrizes de pesos com o mesmo raio espectral, e elas não possuem o mesmo desempenho quando o MSE (erro médio quadrático) é calculado. Para a base Narma ordem 10 (Figura 5.5), a busca realizada por RCDESIGN (A) terminou na oitava geração, a *Busca RS* (B) e *Busca TR* (C) terminaram na décima geração. Para a base Narma ordem 30 (Figura 5.6), os três métodos terminaram na décima geração. Na Tabela 5.15, na Tabela 5.16 e na Tabela 5.17

Tabela 5.19 NARMA ordem 30. Configuração dos Parâmetros

Modelo	Tamanho	Ativação	Treino	Raio Spec.
RCDESIGN	106	*Tanh*	*Ridge*	1,7572
Busca RS	192	*Tanh* [1]	*Pseudo* [1]	1
Busca TR	197	*Tanh* [1]	*Pseudo* [1]	1

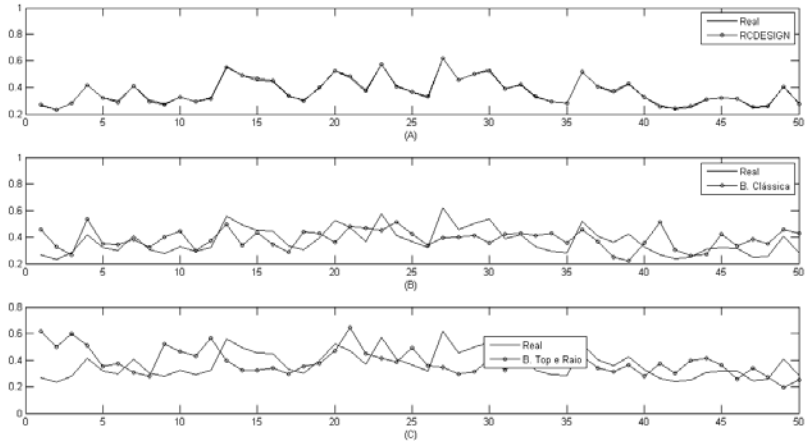

Figura 5.3 Narma ordem 10. (A) Real x RCDESIGN, (B) Real x *Busca RS* e (C) Real x *Busca TR*.

observamos que os valores médios de gerações para os 3 métodos nestas bases foram próximos ao limite de 11, isso se deve à complexidade das séries, logo são necessárias várias gerações para que se encontre uma solução satisfatória.

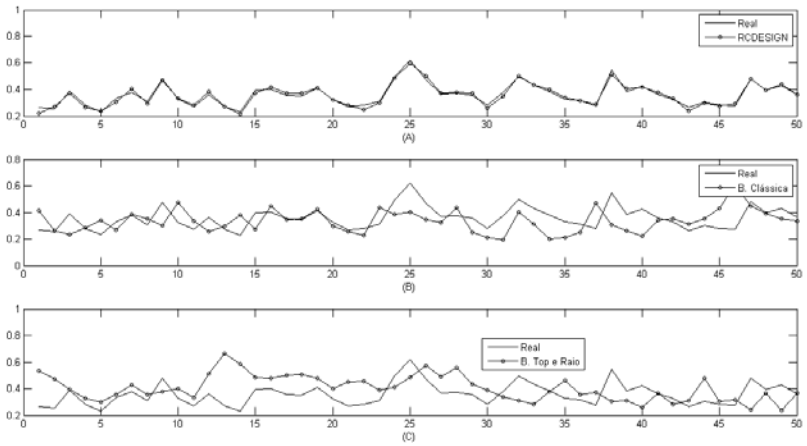

Figura 5.4 Narma ordem 30. (A) Real x RCDESIGN, (B) Real x *Busca RS* e (C) Real x *Busca TR*.

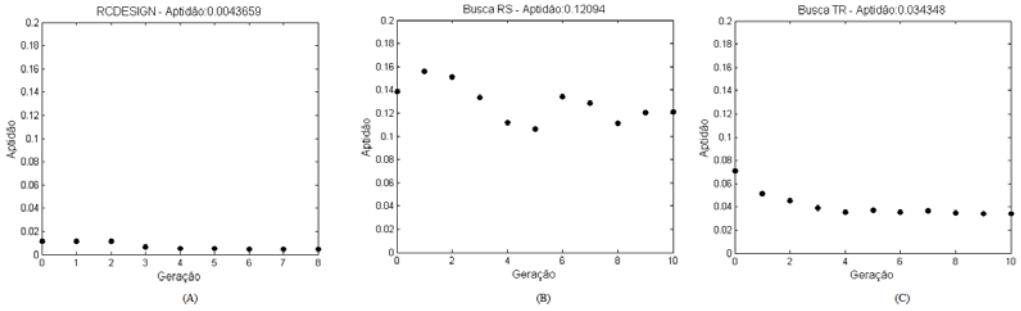

Figura 5.5 NARMA ordem 10. (A) Evolução da aptidão com RCDESIGN, (B) evolução da aptidão com a *Busca RS* e (C) evolução da aptidão com a *Busca TR*.

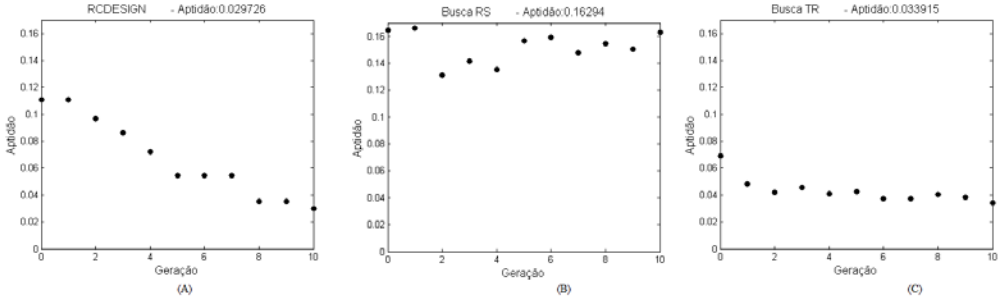

Figura 5.6 NARMA ordem 30. (A) Evolução da aptidão com RCDESIGN, (B) evolução da aptidão com a *Busca RS* e (C) evolução da aptidão com a *Busca TR*.

5.4.2 *Mackey-Glass* (MGS)

A tarefa de previsão da base *Mackey-glass* também é usada como *benchmark* em várias pesquisas e sistemas criados com RC têm apresentado excelentes resultados para esta base. O caos médio ($\tau = 17$) tem sido o parâmetro mais usado nos trabalhos de RC, porém existem alguns poucos trabalhos que utilizam o caos moderado ($\tau = 30$). A Figura 5.7 apresenta a topologia e a distribuição dos pesos dos sistemas criados por RCDESIGN, pela *Busca RS* e pela *Busca TR* para base *Mackey-glass* caos médio e a Figura 5.8 apresenta as mesmas informações para a base *Mackey-glass* caos moderado.

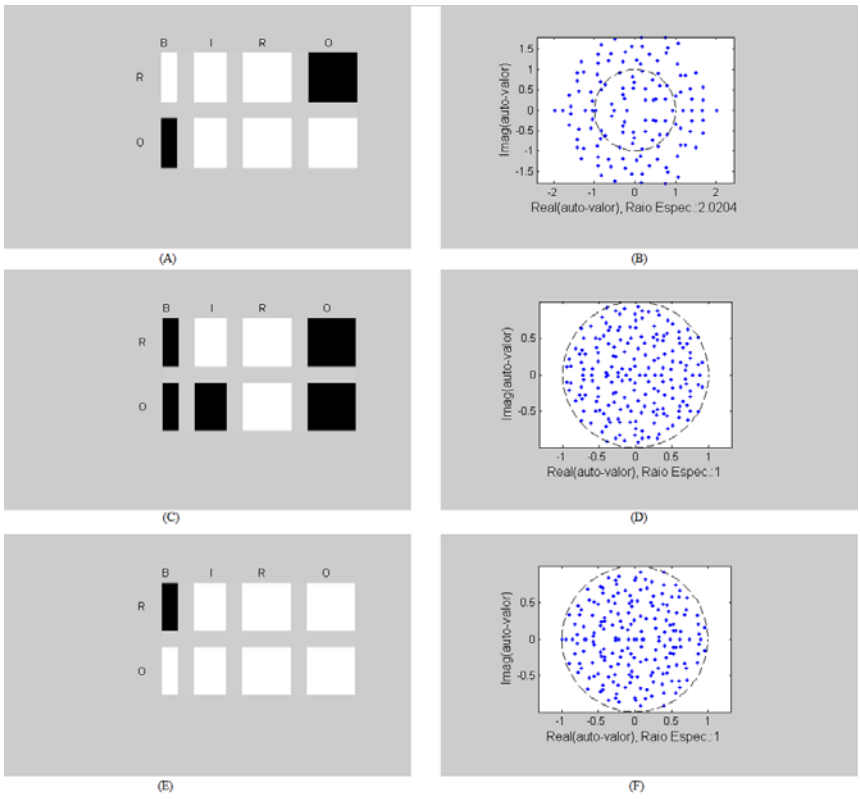

Figura 5.7 MGS caos médio. RCDESIGN: (A) Topologia e (B) distribuição dos pesos. *Busca RS*: (C) Topologia e (D) distribuição dos pesos. *Busca TR*: (E) Topologia e (F) distribuição dos pesos.

Para a base *Mackey-glass* caos médio, o sistema criado por RCDESIGN (Figura 5.7) apresenta raio espectral igual a $2,0204$ e distribuição não uniforme dos pesos do *reservoir* no plano complexo (B), pois os mesmos foram escolhidos de maneira independente do raio espectral por este método. Os sistemas criados pela *Busca RS* (D) e pela *Busca*

TR (F) apresentam raio espectral dentro do limite de 1 para propriedade de *echo state*. Observamos também que os sistemas criados por RCDESIGN (A) e pela *Busca TR* (E) possuem muitas ligações em sua topologia fazendo com que as aproximações de um sistema linear, que foram explicadas em Seção 3.1, não sejam válidas para eles.

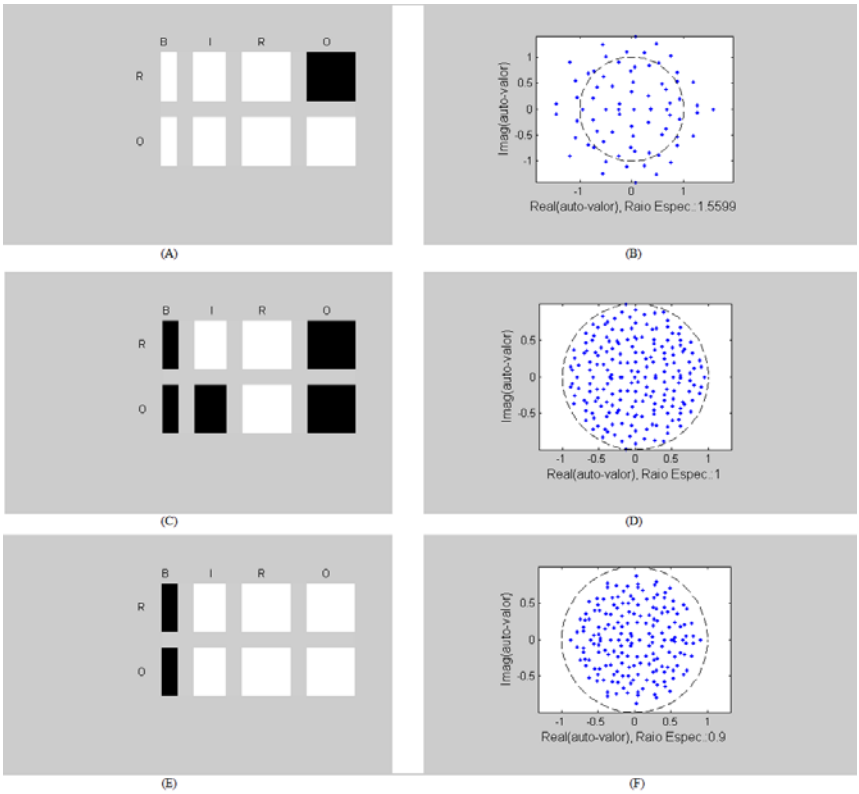

Figura 5.8 MGS caos moderado. RCDESIGN: (A) Topologia e (B) distribuição dos pesos. *Busca RS*: (C) Topologia e (D) distribuição dos pesos. *Busca TR*: (E) Topologia e (F) distribuição dos pesos.

Para a base *Mackey-glass* caos moderado (Figura 5.8), observam-se resultados semelhantes aos observados na base *Mackey-glass* caos médio, onde o sistema criado por RCDESIGN (B) apresenta raio espectral maior que 1 e os sistemas criados por RCDESIGN (A) e pela *Busca TR* (E) utilizam as conexões opcionais de RC. Vale ressaltar que para as duas bases *Mackey-glass*, os sistemas criados por RCDESIGN apresentaram os melhores resultados de previsão (ver Seção 5.2.2) quando comparados com a *Busca RS* e com a *Busca TR*.

A Tabela 5.20 apresenta a quantidade de neurônios, a função de ativação dos neurônios, a função de treinamento e o raio espectral dos sistemas criados pelos três métodos para a

base *Mackey-glass* caos médio e a Tabela 5.21 apresenta as mesmas informações para a base *Mackey-glass* caos moderado.

O sistema criado por RCDESIGN para a base *Mackey-glass* caos médio (Tabela 5.20), tem 132 neurônios no *reservoir*, e é menor do que os sistemas criados pela *Busca RS* e pela *Busca TR*. Os três sistemas utilizaram a mesma função de ativação, *Tanh*, e a mesma função de treinamento, a *pseudo-inversa*.

Tabela 5.20 MGS caos médio: Configuração dos Parâmetros

Modelo	Tamanho	Ativação	Treino	Raio Spec.
RCDESIGN	132	*Tanh*	*Pseudo*	2,0204
Busca RS	192	*Tanh* [1]	*Pseudo* [1]	1
Busca TR	191	*Tanh* [1]	*Pseudo* [1]	1

RCDESIGN também criou um sistema bem menor para a base *Mackey-glass* caos moderado (Tabela 5.21) do que os outros dois métodos, com 76 neurônios no *reservoir*, enquanto que os outros dois métodos criaram sistemas com quase 200 neurônios. Os três sistemas utilizam a mesma função de ativação e a mesma função de treinamento.

Tabela 5.21 MGS caos moderado: Configuração dos Parâmetros

Modelo	Tamanho	Ativação	Treino	Raio Spec.
RCDESIGN	76	*Tanh*	*Pseudo*	1,5599
Busca RS	199	*Tanh* [1]	*Pseudo* [1]	1
Busca TR	181	*Tanh* [1]	*Pseudo* [1]	0,9

A Figura 5.9 apresenta graficamente uma comparação entre os últimos 50 valores reais e previstos por RCDESIGN (A), pela *Busca RS* (B) e pela *Busca TR* (C) para a base *Mackey-glass* caos médio, e a Figura 5.10 apresenta as mesmas informações para a base *Mackey-glass* caos moderado. Podemos observar nessas figuras que as previsões realizadas pelos sistemas criados por RCDESIGN, para as duas bases, foram melhores do que as previsões realizadas pelos sistemas criados pela *Busca RS* e pela *Busca TR*. A análise dos erros de previsão, Seção 5.2.2, confirma o que pode ser observado nessas figuras, pois os erros de previsão de RCDESIGN foram muito menores do que os erros de previsão da *Busca RS* e da *Busca TR* em todas as medidas apresentadas.

A Figura 5.11 apresenta a evolução da *aptidão* dos três métodos para a base *Mackey-glass* caos médio e a Figura 5.12 apresenta a evolução da *aptidão* dos três métodos para a

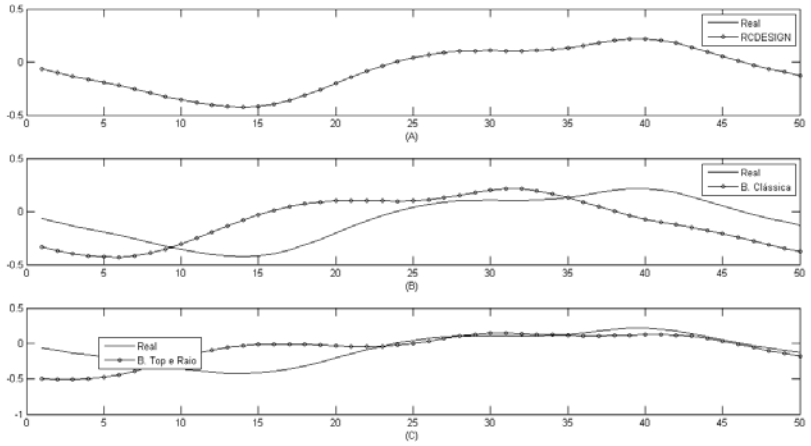

Figura 5.9 MGS caos moderado. (A) Real x RCDESIGN, (B) Real x *Busca RS* e (C) Real x *Busca TR*.

base *Mackey-glass* caos moderado. Como as populações dos três métodos são grandes (120 indivíduos) e como as duas séries não apresentam comportamento muito complexo é provável que uma solução satisfatória seja encontrada em poucas gerações, pois podemos observar que os métodos encerraram as buscas com apenas 3 gerações. Se observarmos a Tabela 5.15, a Tabela 5.16 e a Tabela 5.17 veremos que o valor médio de gerações para os 3 métodos com estas 2 bases é semelhante ao apresentado nestes exemplos, pois a média de gerações (das 30 inicializações) para os métodos nessas bases foi 4.

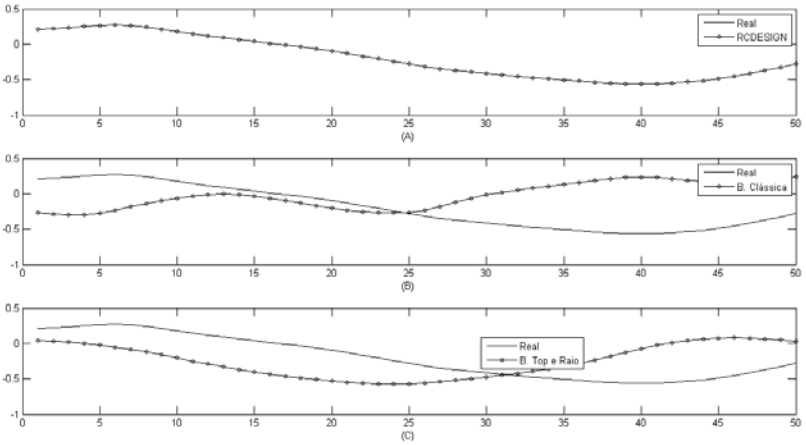

Figura 5.10 MGS caos médio. (A) Real x RCDESIGN, (B) Real x *Busca RS* e (C) Real x *Busca TR*.

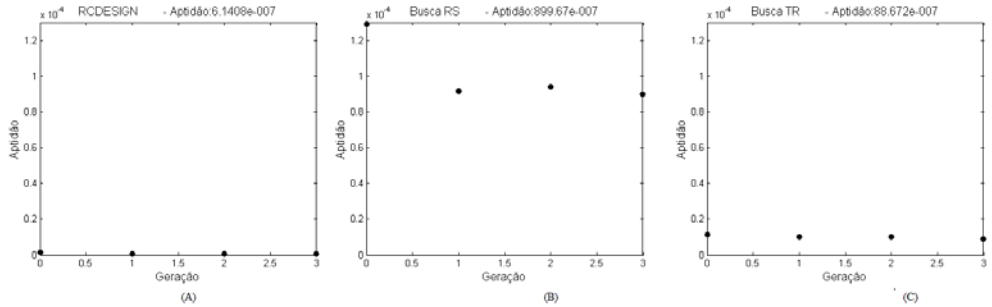

Figura 5.11 MGS caos médio. (A) Evolução da aptidão com RCDESIGN, (B) evolução da aptidão com a *Busca RS* e (C) evolução da aptidão com a *Busca TR*.

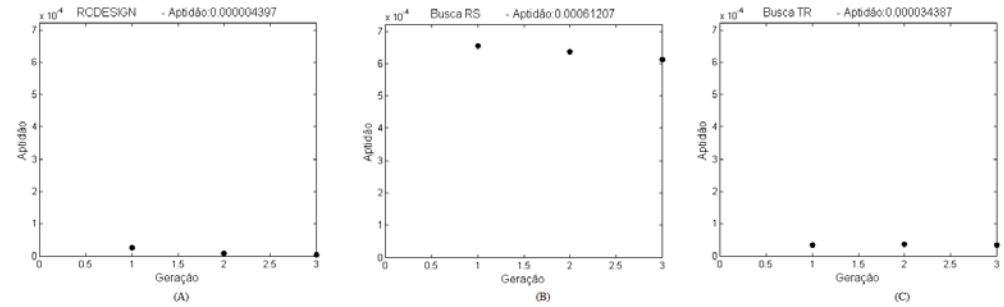

Figura 5.12 MGS caos moderado. (A) Evolução da aptidão com RCDESIGN, (B) evolução da aptidão com a *Busca RS* e (C) evolução da aptidão com a *Busca TR*.

5.4.3 *Multiple Sinewave Oscillator* (MSO)

O problema conhecido como *Multiple Sinewave Oscillator* (MSO) é uma simples superposição de ondas senoidais, este problema tem sido usado como *benchmark* para RC. A Figura 5.13 apresenta a topologia e a distribuição dos pesos dos sistemas criados pelos métodos RCDESIGN, *Busca RS* e *Busca TR* para previsão desta série. O sistema criado por RCDESIGN possui todas as possíveis conexões de RC (A) e raio espectral igual a $1,6144$. O raio espectral do sistema criado pela *Busca RS* (D) foi igual a 1 e o raio espectral do sistema criado pela *Busca TR* (F) foi igual a $0,8$. Os sistemas criados pela *Busca RS* (C) e pela *Busca TR* (E) utilizam menos conexões do que o sistema criado por RCDESIGN.

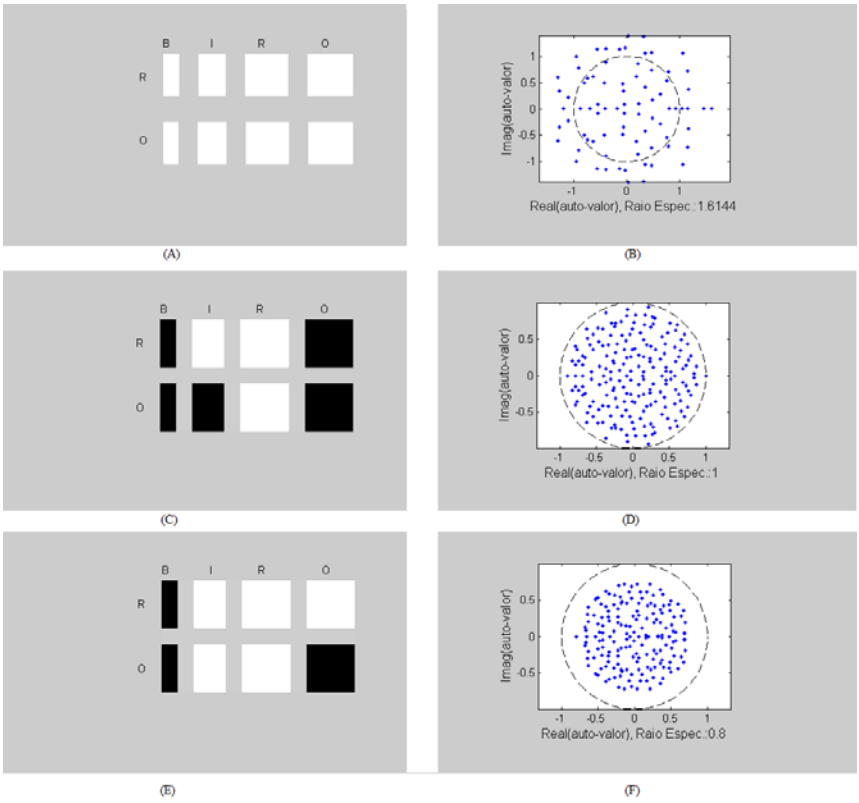

Figura 5.13 MSO. RCDESIGN: (A) Topologia e (B) distribuição dos pesos. *Busca RS*: (C) Topologia e (D) distribuição dos pesos. *Busca TR*: (E) Topologia e (F) distribuição dos pesos.

Na Tabela 5.22 observamos que o método RCDESIGN criou um sistema com 79 neurônios no *reservoir*, enquanto que a *Busca RS* e a *Busca TR* criaram sistemas muito maiores com respectivamente 193 e 172 neurônios. Como o sistema criado por

RCDESIGN é bem menor do que os outros dois sistemas, podemos afirmar que o sistema criado por RCDESIGN será mais rápido do que os outros dois quando em operação. Como relação à função de ativação e à função de treinamento, os três sistemas utilizam os mesmos valores.

Tabela 5.22 MSO. Configuração dos Parâmetros

Modelo	Tamanho	Ativação	Treino	Raio Spec.
RCDESIGN	79	*Tanh*	*Pseudo*	1,6144
Busca RS	193	*Tanh* [1]	*Pseudo* [1]	1
Busca TR	172	*Tanh* [1]	*Pseudo* [1]	0,8

A Figura 5.14 apresenta graficamente uma comparação entre os últimos 50 pontos da série real e da série prevista pelo sistema criado com RCDESIGN (A), com a *Busca RS* (B) e com a *Busca TR* (C).

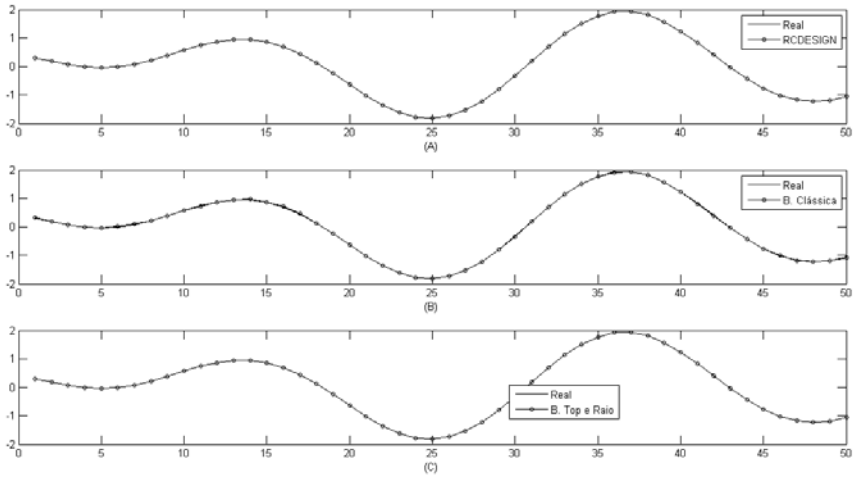

Figura 5.14 MSO. (A) Real x RCDESIGN, (B) Real x *Busca RS* e (C) Real x *Busca TR*.

Para esta base, as previsões realizadas pelos três métodos, apresentaram erros muito pequenos, dessa forma a avaliação da diferença do desempenho dos mesmos só pode ser realizada através da análise dos erros de previsão (Seção 5.2.2), já que visualmente os métodos possuem resultados semelhantes. RCDESIGN criou um sistema com desempenho melhor do que o desempenho dos sistemas criados pelos outros dois métodos,

obtendo os menores valores de MSE, NMSE e NRMSE, tanto no treinamento quanto no teste.

A Figura 5.15 apresenta a evolução do melhor indivíduo com RCDESIGN (A), com a *Busca RS* (B) e com a *Busca TR* (C). O método RCDESIGN (A) e a *Busca TR* (C) precisaram de apenas 3 gerações para encontrarem uma solução enquanto que a *Busca RS* (B) precisou de 5 gerações. Como as populações dos três métodos são grandes (120 indivíduos) e como a série não apresenta comportamento muito complexo é provável que uma solução satisfatória seja encontrada em poucas gerações. Isso se confirma quando analisamos as médias de gerações para RCDESIGN (ver Tabela 5.15) e para a *Busca TR* (ver Tabela 5.17) com relação à esta base, onde os dois métodos apresentam uma média (nas 30 inicializações) de 4 gerações.

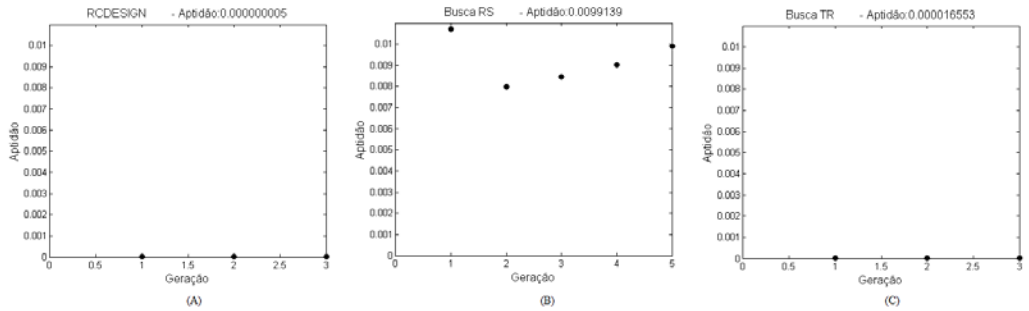

Figura 5.15 MSO. (A) Evolução da aptidão com RCDESIGN, (B) evolução da aptidão com a *Busca RS* e (C) evolução da aptidão com a *Busca TR*.

5.4.4 Série Natural do Brilho da Estrela (*STAR*)

O estudo com a série natural do brilho da estrela (*STAR*) faz parte de um dos estudos desenvolvidos para realizar a validação acadêmica da aplicação do método RCDESIGN à tarefa de previsão de séries temporais. Não foram encontrados trabalhos de RC com esta série, mas os resultados dos experimentos realizados com os três métodos, para esta série, serão apresentados neste capítulo. A Figura 5.16 apresenta a topologia e a distribuição dos pesos dos sistemas criados pelos três métodos. Podemos observar nesta figura que os sistemas criados pelos três métodos, (B), (D) e (F), apresentam raio espectral próximo a 1. Com relação às conexões opcionais, observamos que o sistema criado por RCDESIGN (A) apresenta várias conexões opcionais e o sistema criado pela *Busca TR* (E) apresenta apenas uma conexão opcional ($\mathbf{W}^{out\,out}$ ou O-O).

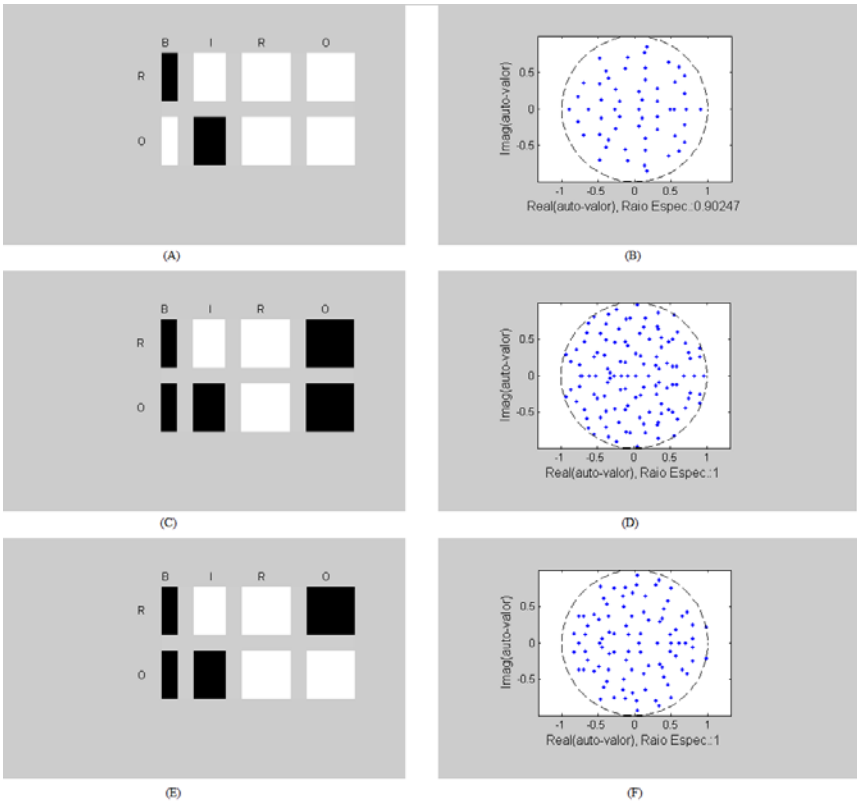

Figura 5.16 *STAR*. RCDESIGN: (A) Topologia e (B) distribuição dos pesos. *Busca RS*: (C) Topologia e (D) distribuição dos pesos. *Busca TR*: (E) Topologia e (F) distribuição dos pesos.

A Tabela 5.23 apresenta a quantidade de neurônios, a função de ativação dos neurônios, a função de treinamento e o raio espectral dos sistemas criados pelos três métodos para

a base *STAR*. O método RCDESIGN criou um sistema pequeno, com 56 neurônios no *reservoir*, enquanto que a *Busca RS* criou um sistema muito maior, com 193 neurônios, e a *Busca TR* criou um sistema com 91 neurônios. Os três sistemas utilizam a mesma função de ativação, porém o sistema criado por RCDESIGN utiliza uma função de treinamento diferente dos outros dois sistemas.

Tabela 5.23 *STAR*. Configuração dos Parâmetros

Modelo	Tamanho	Ativação	Treino	Raio Spec.
RCDESIGN	56	*Tanh*	*Ridge*	0,90247
Busca RS	120	*Tanh* [1]	*Pseudo* [1]	1
Busca TR	91	*Tanh* [1]	*Pseudo* [1]	1

A Figura 5.17 apresenta graficamente uma comparação entre os últimos 50 pontos da série real e da série prevista com RCDESIGN (A), com a Busca RS (B) e a *Busca TR* (C). Embora RCDESIGN tenha criado um sistema com erros de previsão menores do que os erros dos sistemas criados pelos outros dois métodos, eles foram muito próximos e a avaliação da diferença do desempenho dos três sistemas só pode ser realizada através da análise dos erros de previsão (Seção 5.2.2), pois não é possível verificar as diferenças nesta figura.

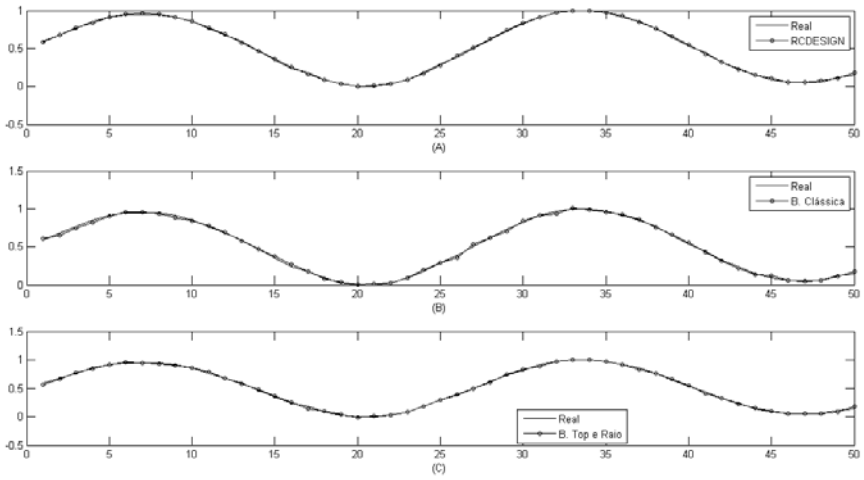

Figura 5.17 *STAR*. (A) Real x RCDESIGN, (B) Real x *Busca RS* e (C) Real x *Busca TR*.

A Figura 5.18 apresenta o melhor valor de *aptidão* de cada geração na criação do

sistema com RCDESIGN (A), com a *Busca RS* (B) e com a *Busca TR* (C). O método RCDESIGN só precisou de 3 gerações para encontrar uma solução e os outros dois métodos precisaram de 10 gerações. Para RCDESIGN a evolução do melhor indivíduo ao longo das gerações foi regular, porém para a *Busca RS* e para a *Busca TR*, a evolução do melhor indivíduo ao longo das gerações foi irregular. O comportamento apresentado nesses exemplos é semelhante ao comportamento médio dos métodos (nas 30 inicializações) para esta base.

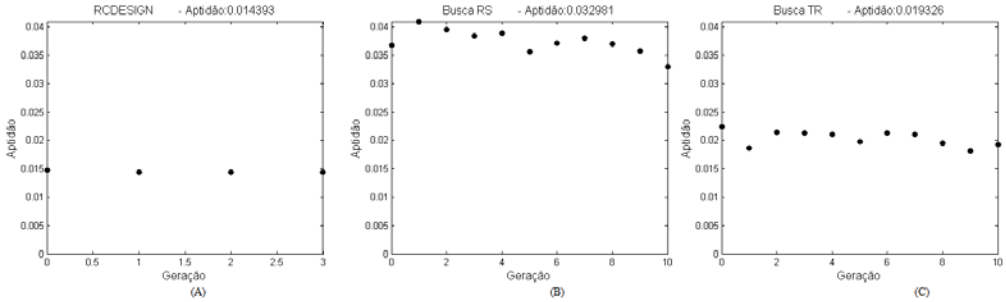

Figura 5.18 *STAR*. (A) Evolução da aptidão com RCDESIGN, (B) evolução da aptidão com a *Busca RS* e (C) evolução da aptidão com a *Busca TR*.

5.4.5 *Dow Jones Industrial Average* (*Down-Jones*)

A série Financeira (*Dow Jones Industrial Average* ou simplesmente *Down-Jones*) foi escolhida para validação acadêmica do método RCDESIGN, embora não tenham sido encontrados trabalhos de RC com esta série. Podemos observar na Figura 5.19, que o sistema criado por RCDESIGN (A) utiliza conexões opcionais de RC e apresenta raio espectral igual $1,4222$ (B). O sistema criado pela *Busca RS* (D) tem um raio espectral bem pequeno e igual a $0,7$. O sistema criado pela *Busca TR* (E) também utiliza as conexões opcionais de RC e tem um raio espectral iguala $1,5$ (F). Apenas para esta base, não foi possível afirmar, com significância de 5%, que o sistema criado pelo método RCDESIGN é melhor do que o sistema criado pelo método da *Busca TR*.

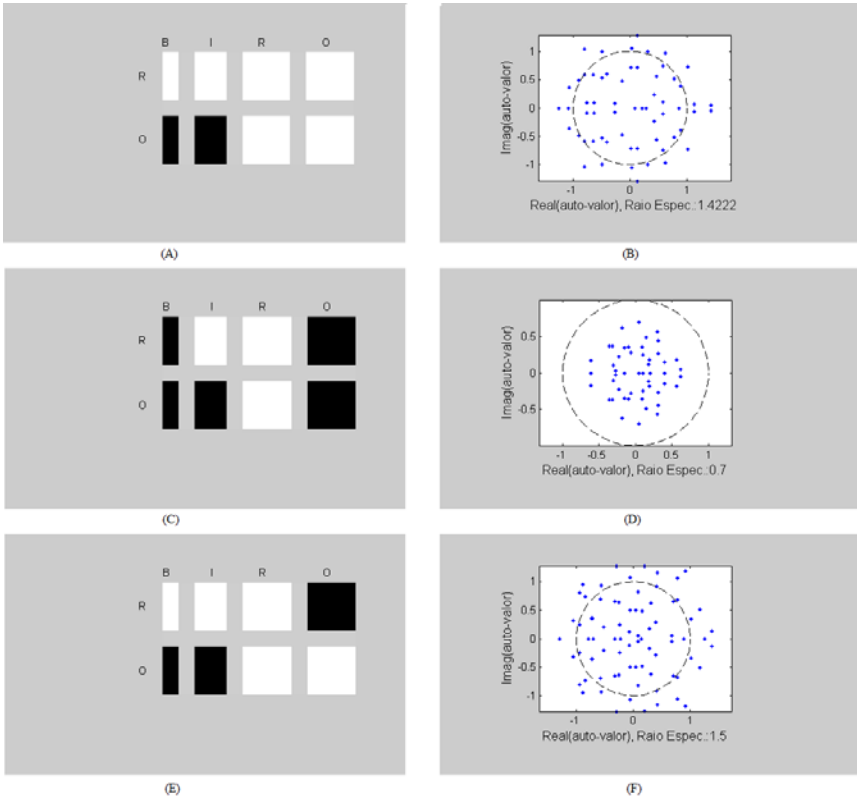

Figura 5.19 *Down-Jones*. RCDESIGN: (A) Topologia e (B) distribuição dos pesos. *Busca RS*: (C) Topologia e (D) distribuição dos pesos. *Busca TR*: (E) Topologia e (F) distribuição dos pesos.

A Tabela 5.24 apresenta a quantidade de neurônios, a função de ativação dos neurônios, a função de treinamento e o raio espectral dos sistemas criados pelos três métodos para a base *Down-Jones*. Os três métodos criaram sistemas bem pequenos, menos de 80

neurônios no *reservoir*, e utilizam a mesma função de ativação. RCDESIGN usou a regressão *Ridge-Regress* para o treinamento dos pesos do *readout*, enquanto que os outros dois métodos utilizam a *pseudo-inversa*.

Tabela 5.24 *Down-Jones*. Configuração dos Parâmetros

Modelo	Tamanho	Ativação	Treino	Raio Spec.
RCDESIGN	60	*Tanh*	*Ridge*	1,4222
Busca RS	50	*Tanh* [1]	*Pseudo* [1]	0,7
Busca TR	78	*Tanh* [1]	*Pseudo* [1]	1,5

A Figura 5.20 apresenta graficamente uma comparação entre os últimos 50 pontos da série real e da série prevista com RCDESIGN (A), com a Busca RS (B) e a *Busca TR* (C). A previsão realizada pelo sistema criado por RCDESIGN foi equivalente a previsão realizada pela *Busca TR* e foi um pouco melhor do que a previsão realizada pela *Busca RS* (ver Seção 5.2.2).

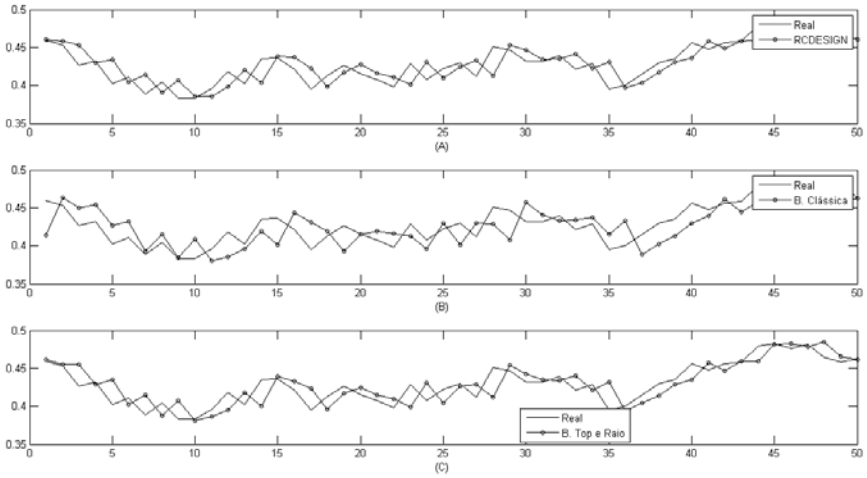

Figura 5.20 *Down-Jones*. (A) Real x RCDESIGN, (B) Real x *Busca RS* e (C) Real x *Busca TR*.

A Figura 5.21 apresenta o melhor valor de *aptidão* de cada geração na criação do sistema com RCDESIGN (A), com a *Busca RS* (B) e com a *Busca TR* (C). O método RCDESIGN precisou de 4 gerações para encontrar uma solução e os outros dois métodos precisaram de 10 gerações. A Tabela 5.15, a Tabela 5.16 e a Tabela 5.17 indicam que o

comportamento dos métodos nesses exemplos está dentro do comportamento médio dos métodos para esta base.

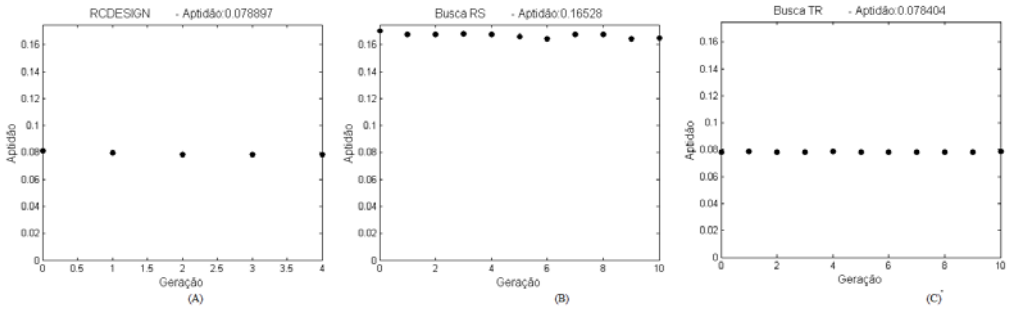

Figura 5.21 *Down-Jones*. (A) Evolução da aptidão com RCDESIGN, (B) evolução da aptidão com a *Busca RS* e (C) evolução da aptidão com a *Busca TR*.

5.4.6 Belo Jardim (BJD)

A base de dados de Belo Jardim é uma das três bases do estudo de caso da previsão da velocidade média horária dos ventos na região NE do Brasil. Podemos observar na Figura 5.22 que os sistemas criados por RCDESIGN e pela *Busca TR* são semelhantes no aspecto da utilização das conexões opcionais de RC, (A) e (E), e também pelo fato que o raio espectral dos dois sistemas, (B) e (F), é maior que o limite de 1 (propriedade de *echo state*). A distribuição dos pesos do *reservoir*, dos sistemas criados por RCDESIGN (B) e pela *Busca TR* (F) não é uniforme dentro de um círculo unitário no plano complexo. O método da *Busca RS* considera a aproximação de RC à sistemas lineares e só cria sistemas dentro dessa aproximação, onde o sistema criado pela *Busca RS* possui apenas as conexões obrigatórias de RC (C) e raio espectral dentro do limite de 1 (D), com pesos uniformemente distribuídos dentro de um círculo unitário no plano complexo.

A Tabela 5.25 apresenta a quantidade de neurônios, a função de ativação dos neurônios, a função de treinamento e o raio espectral dos sistemas criados pelos três métodos para a base de Belo Jardim. Os sistemas criados por RCDESIGN e pela *Busca TR* também são semelhantes no tamanho do *reservoir*, com 80 e 92 neurônios respectivamente. O sistema criado pela *Busca RS* é bem maior do que os outros dois, com 197 neurônios no *reservoir*. Os três sistemas utilizam a mesma função de ativação e mesma função de treinamento do *readout*.

Podemos observar na Figura 5.23 que RCDESIGN (A) e a *Busca TR* (C) criaram sistemas que realizam uma boa previsão para a base de Belo Jardim e que o sistema

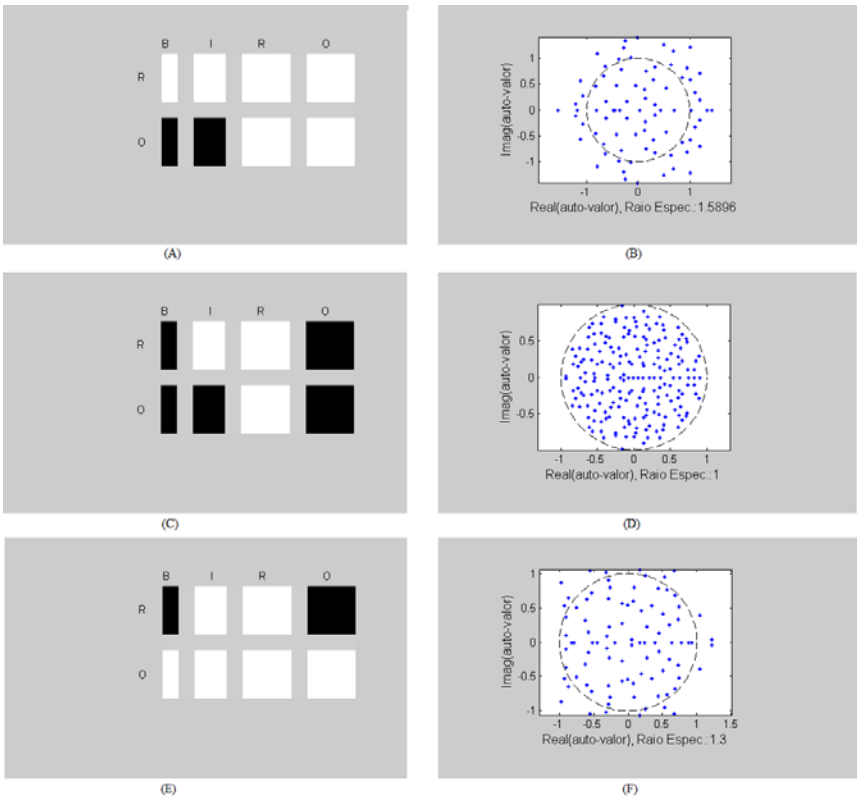

Figura 5.22 BJD. RCDESIGN: (A) Topologia e (B) distribuição dos pesos. *Busca RS*: (C) Topologia e (D) distribuição dos pesos. *Busca TR*: (E) Topologia e (F) distribuição dos pesos.

criado pela *Busca RS* (B) tem um desempenho inferior em relação ao desempenho dos outros dois sistemas. Nesta figura são exibidos os últimos 50 padrões do conjunto de teste que foram previstos pelos três métodos. O que se percebe visualmente é confirmado através da análise dos erros de previsão da Seção 5.2.2, onde RCDESIGN apresenta os menores erros de previsão para a base de Belo Jardim, seguido pelo sistema criado pela *Busca TR*, com desempenho inferior, porém muito próximo ao do sistema criado por RCDESIGN, e por fim pelo sistema criado pela *Busca RS*, que apresentou os maiores erros de previsão em todas as medidas calculadas.

A Figura 5.24 apresenta o melhor valor de *aptidão* de cada geração dos três métodos. Foram necessárias apenas 6 gerações para que o método RCDESIGN (A) encontrasse uma boa solução para o problema. A *Busca RS* (B) e a *Busca TR* (C) terminaram na décima geração.

Tabela 5.25 BJD: Configuração dos Parâmetros

Modelo	Tamanho	Ativação	Treino	Raio Spec.
RCDESIGN	80	*Tanh*	*Pseudo*	1,6144
Busca RS	197	*Tanh* [1]	*Pseudo* [1]	1
Busca TR	92	*Tanh* [1]	*Pseudo* [1]	1,3

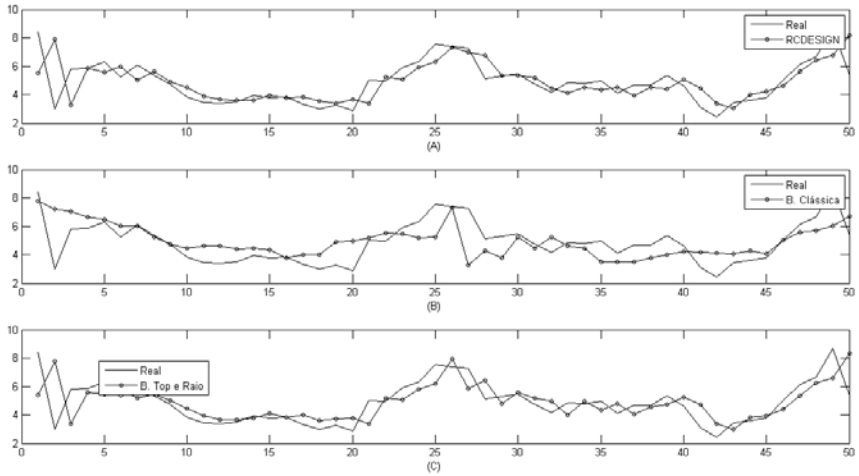

Figura 5.23 BJD. (A) Real x RCDESIGN, (B) Real x *Busca RS* e (C) Real x *Busca TR*.

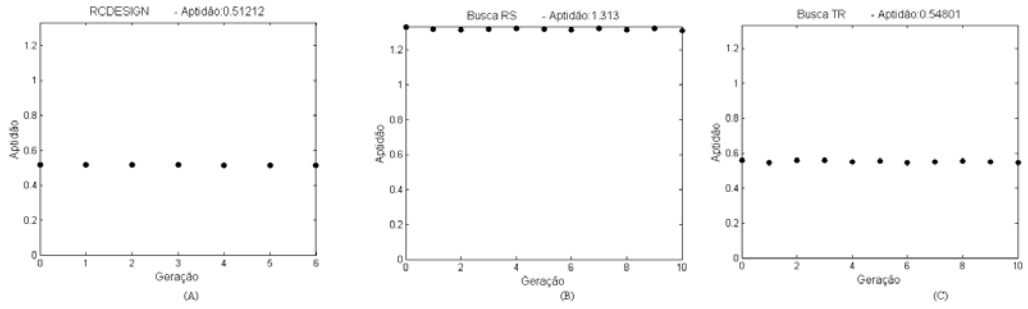

Figura 5.24 BJD. (A) Evolução da aptidão com RCDESIGN, (B) evolução da aptidão com a *Busca RS* e (C) evolução da aptidão com a *Busca TR*.

5.4.7 São João do Cariri (SCR)

A base de dados de São João do Cariri é outra base do estudo de caso da previsão da velocidade média horária dos ventos na região NE do Brasil. Para esta base, os sistemas criados por RCDESIGN e pela *Busca TR* são semelhantes no aspecto da utilização das conexões opcionais de RC, (A) e (E), e também em relação ao raio espectral, pois o método RCDESIGN (B) e o método da *Busca TR* (F) criaram sistemas com raio espectral bem maior que 1. O sistema criado pela *Busca RS* possui apenas as conexões obrigatórias de RC (C) e raio espectral dentro do limite de 1 (D).

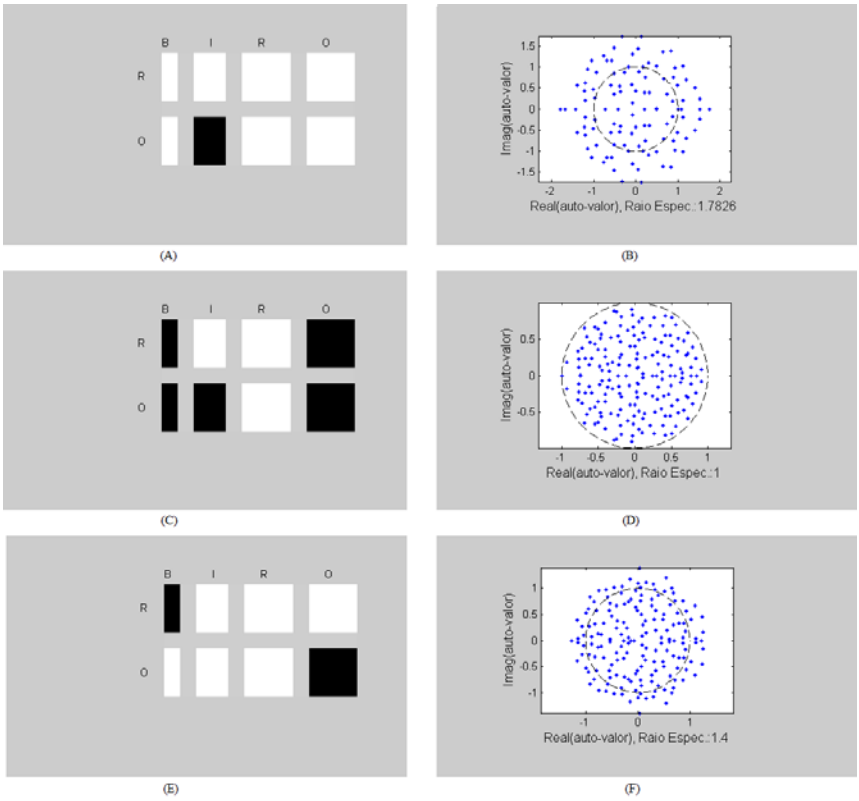

Figura 5.25 SCR. RCDESIGN: (A) Topologia e (B) distribuição dos pesos. *Busca RS*: (C) Topologia e (D) distribuição dos pesos. *Busca TR*: (E) Topologia e (F) distribuição dos pesos.

A Tabela 5.26 apresenta a quantidade de neurônios, a função de ativação, a função de treinamento e o raio espectral dos sistemas criados pelos três métodos para a base de São João do Cariri. Para esta base, o sistema criado por RCDESIGN é bem menor do que os sistemas criados pela *Busca RS* e pela *Busca TR*. RCDESIGN criou um sistema com 106 neurônios no *reservoir*, a *Busca RS* criou um sistema com 179 neurônios e a *Busca TR*

criou um sistema com 170 neurônios. Outra diferença importante entre os três sistemas é em relação à função de treinamento do *readout*, que para RCDESIGN é a *Ridge-regress* e para os outros dois é a *Pseudo-inversa*.

Tabela 5.26 SCR: Configuração dos Parâmetros

Modelo	Tamanho	Ativação	Treino	Raio Spec.
RCDESIGN	106	*Tanh*	*Ridge*	1,7826
Busca RS	179	*Tanh* [1]	*Pseudo* [1]	1
Busca TR	170	*Tanh* [1]	*Pseudo* [1]	1,4

A Figura 5.26 apresenta graficamente uma comparação entre os últimos 50 pontos da série real e da série prevista por RCDESIGN (A), pela *Busca RS* (B) e pela *Busca TR* (C). Os três métodos foram capazes de criar sistemas que realizam boa previsão para esta base, porém podemos perceber que o desempenho do sistema criado por RCDESIGN (A) foi levemente melhor do que o desempenho do sistema criado pela *Busca TR* (C) e também melhor do que o desempenho do sistema criado pela *Busca RS* (B).

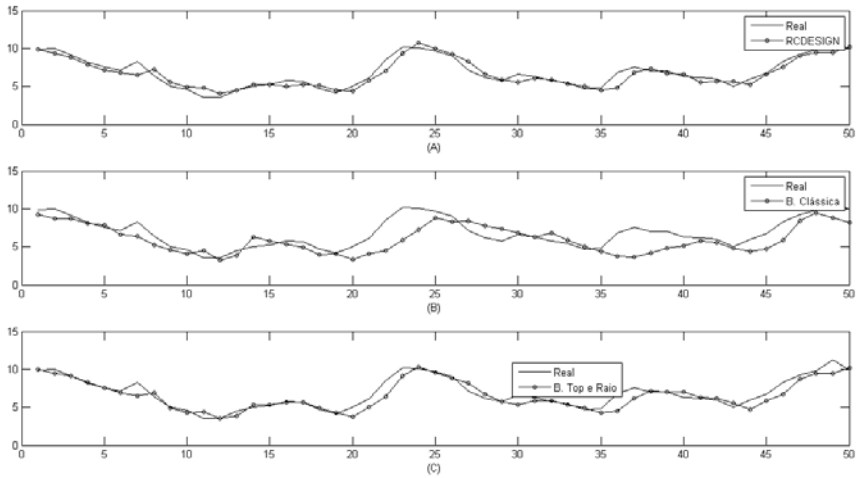

Figura 5.26 SCR. (A) Real x RCDESIGN, (B) Real x *Busca RS* e (C) Real x *Busca TR*.

A Figura 5.27 apresenta o melhor valor de *aptidão* de cada geração na criação dos sistemas com os três métodos. Foram necessárias 7 gerações para o método RCDESIGN (A), e o valor de *aptidão* ao final da busca foi igual a 0,38386. A *Busca RS* (B) terminou

na décima geração, com valor de aptidão igual a $1,1296$, porém a evolução da aptidão foi irregular. A *Busca TR* (C) terminou na quinta geração, com valor de aptidão igual a $0,44044$ e também com evolução irregular da aptidão. Para os métodos da *Busca RS* e da *Busca TR*, o uso do operador de elite não garante uma evolução regular da aptidão do melhor indivíduo em cada geração, pois esses métodos se baseiam na busca do raio espectral do *reservoir* e na geração aleatória dos pesos.

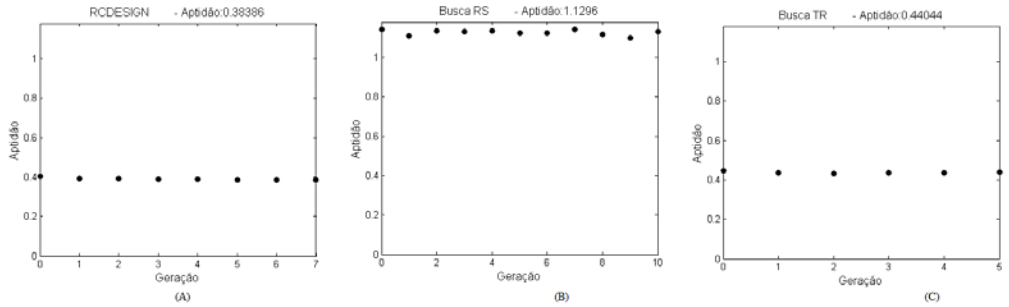

Figura 5.27 SCR. (A) Evolução da aptidão com RCDESIGN, (B) evolução da aptidão com a *Busca RS* e (C) evolução da aptidão com a *Busca TR*.

5.4.8 Triunfo (TRI)

A base de dados de Triunfo é a terceira e última base do estudo de caso da previsão da velocidade média horária dos ventos na região NE do Brasil. RCDESIGN criou um sistema com quase todas as possíveis conexões de RC (A) e com raio espectral igual a $1,5558$ (B). A *Busca RS* criou um sistema apenas com as conexões obrigatórias de RC (C) e a *Busca TR* (E) criou um sistema que utiliza, além das conexões obrigatória de RC, a conexão de realimentação da camada de saída para o *reservoir* ($\mathbf{W}^{back}$ ou O-R).

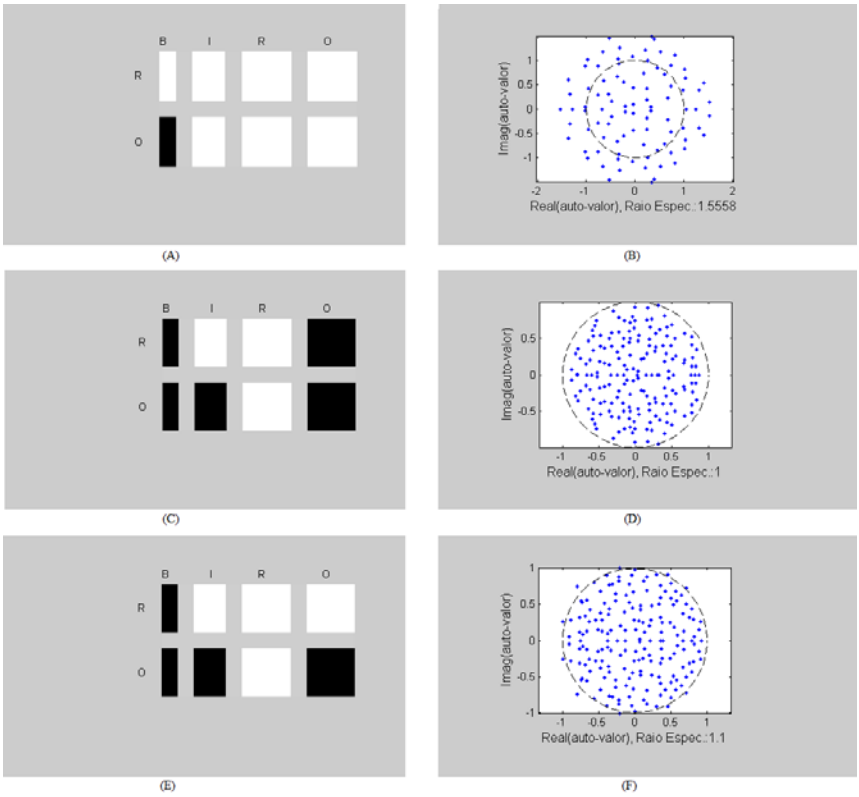

Figura 5.28 TRI. RCDESIGN: (A) Topologia e (B) distribuição dos pesos. *Busca RS*: (C) Topologia e (D) distribuição dos pesos. *Busca TR*: (E) Topologia e (F) distribuição dos pesos.

A Tabela 5.27 apresenta a quantidade de neurônios, a função de ativação dos neurônios, a função de treinamento e o raio espectral dos sistemas criados pelos três métodos. O método RCDESIGN criou um sistema com 91 neurônios no *reservoir*, enquanto que a *Busca RS* e *Busca TR* criaram sistemas maiores, com 176 e 182 neurônios respectivamente. A função de treinamento da camada de saída do sistema criado por RCDESIGN é diferente da função de treinamento dos outros dois sistemas.

Tabela 5.27 TRI: Configuração dos Parâmetros

Modelo	Tamanho	Ativação	Treino	Raio Spec.
RCDESIGN	91	*Tanh*	*Ridge*	1,5558
Busca RS	176	*Tanh* [1]	*Pseudo* [1]	1
Busca TR	182	*Tanh* [1]	*Pseudo* [1]	1,1

A Figura 5.29 apresenta graficamente uma comparação entre os últimos 50 pontos da série real e da série prevista por RCDESIGN (A), pela *Busca RS* (B) e pela *Busca TR*(C). Percebemos nesta figura, que as previsões realizadas pelos sistemas criados por RCDESIGN (A) e pela *Busca TR* foram melhores do que as previsões realizadas pelo sistema criado pela *Busca RS* (B).

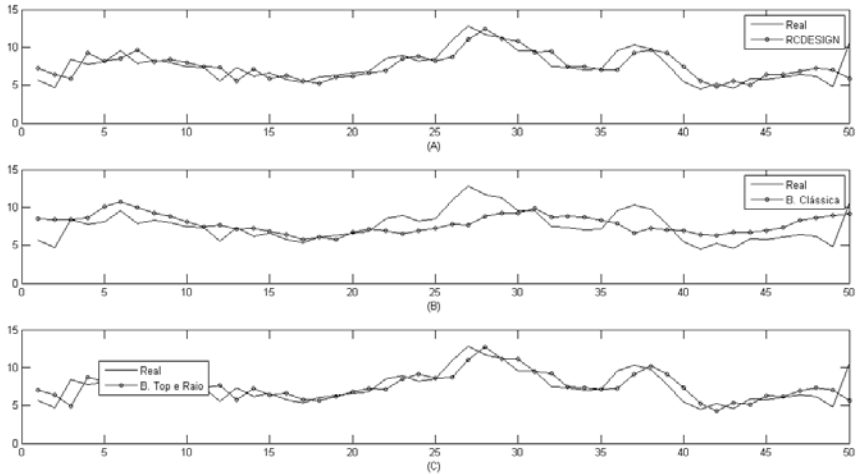

Figura 5.29 TRI. (A) Real x RCDESIGN, (B) Real x *Busca RS* e (C) Real x *Busca TR*.

A Figura 5.30 apresenta a evolução do melhor indivíduo em cada geração dos três métodos para a base de Triunfo. O método RCDESIGN (A) precisou de apenas 3 gerações para encontrar uma solução, e o valor de aptidão ao final da busca foi igual a 0,089393. A *Busca RS* (B) terminou na décima geração com valor de aptidão igual a 0,47062 e a *Busca TR* (C) terminou na oitava geração com aptidão igual a 0,094402. Se observarmos a Tabela 5.15, a Tabela 5.16 e a Tabela 5.17 veremos que o valor médio de gerações para os 3 métodos com esta base é semelhante ao apresentado nestes exemplos.

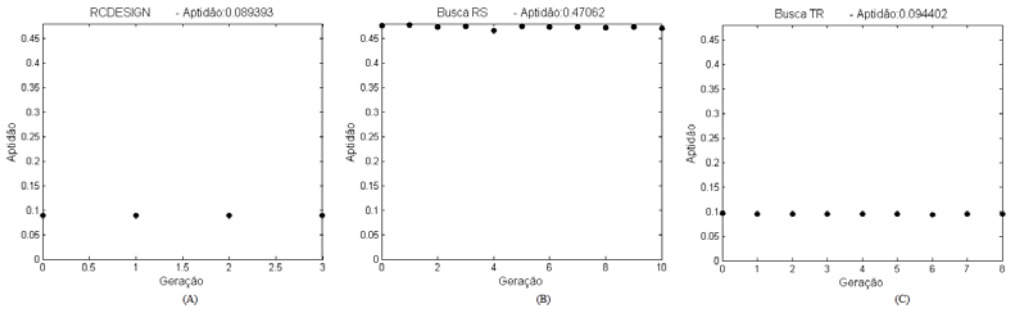

Figura 5.30 TRI. (A) Evolução da aptidão com RCDESIGN, (B) evolução da aptidão com a *Busca RS* e (C) evolução da aptidão com a *Busca TR*.

5.5 Análise dos Resultados de RCDESIGN

De acordo com Lukosevicius e Jaeger [LJ09] a separação do *reservoir* e da camada linear de saída (*readout*) provê uma plataforma onde vários tipos de métodos de Redes Neurais Recorrentes podem ser testados com RC. Esta separação também é particularmente útil para que mecanismos evolucionários de adaptação sejam testados no treinamento supervisionado de RC. Considerando esta questão, este trabalho apresenta um método, chamado RCDESIGN, para o design e treinamento de RC, que possibilita a criação de *reservoirs* mais versáteis, pois não impõe limites na topologia da rede, no raio espectral e nem nos pesos do *reservoir*.

Os resultados apresentados neste trabalho, tanto para as séries clássicas quanto para as séries reais, mostraram que é possível otimizar parâmetros, topologia e pesos simultaneamente, sem as limitações dos trabalhos anteriores, de redução do espaço de busca. Além do mais, os experimentos realizados mostraram que o custo computacional de RCDESIGN foi muito menor do que o custo computacional da *Busca RS* e da *Busca TR*.

Os resultados apresentados nas bases estudadas evidenciam que não é necessária uma distribuição uniforme das poles dos pesos da camada intermediária dentro de um círculo unitário no plano complexo para que os *reservoirs* apresentem bons resultados, já que a dinâmica real dos *reservoirs* só pode ser encontrada com o sistema em uso. Isto está de acordo com o que foi dito por Ozturk e Príncipe em [OP05].

Os resultados apresentados pelos métodos *Busca RS* e *Busca TR*, em todas as bases, também evidenciam o que foi apresentado por Ozturk et al em [OXP07]. Eles demonstraram que existem várias matrizes de pesos com o mesmo raio espectral, e infelizmente

elas não possuem o mesmo desempenho quando o MSE é calculado. Isso pode ser observado claramente na evolução da aptidão da *Busca RS* e da *Busca TR* em todas as bases. Embora os dois métodos criem sistemas com um raio espectral determinado previamente, eles apresentam uma evolução irregular do valor de *aptidão*, pois eles buscam pelos parâmetros de criação das redes e isso implica que os pesos das redes serão criados de forma aleatória, ou seja, não é possível garantir que redes criadas com o mesmo cromossomo (parâmetros) terão o mesmo desempenho. Para RCDESIGN é possível observar a evolução regular do valor de *aptidão*, pois o método RCDESIGN busca simultaneamente pela topologia, pelos parâmetros de geração de RC e pelos pesos, dessa forma não existe variação entre as redes criadas com o mesmo cromossomo, logo existe uma garantia de que essas redes irão apresentar o mesmo desempenho para os mesmos dados.

Foi possível observar que alguns dos sistemas escolhidos pela *Busca TR* apresentaram raio espectral maior que o valor sugerido por Jaeger em [Jae01]. Provavelmente isto aconteceu devido ao uso da topologia variável, com a possibilidade de usar todas as possíveis conexões de RC, o que invalidou em muitas redes a aproximação de RC com um sistema linear. Neste caso foi possível observar que as entradas fornecidas aos sistemas criados pela *Busca TR* foram capazes de colocar esses sistemas em um estado estável durante sua utilização.

O método RCDESIGN realiza a busca dos pesos da camada intermediária sem impor nenhum limite ao raio espectral dessa matriz. Através da análise dos resultados de RCDESIGN observamos que a maioria dos sistemas criados por RCDESIGN apresentaram raio espectral maior que o limite de 1, que foi sugerido por Jaeger em [Jae01]. Como foi dito para a *Busca TR*, as entradas fornecidas aos sistemas criados por RCDESIGN são capazes de colocar esses sistemas em um estado estável durante sua utilização.

Embora, nas séries clássicas, o método RCDESIGN possa ser superado, o que consideramos natural, devido ao grande número de trabalhos com essas bases, RCDESIGN alcançou bons resultados em todas as bases estudadas e os resultados comprovaram que é possível construir RC utilizando uma abordagem diferente da tradicional que se preocupa excessivamente com a propriedade de *Echo State* e que também limitam as possibilidades de conexões do sistema, pois trabalham geralmente apenas com as conexões obrigatórias.

Outro ponto importante a ser destacado é a *função de aptidão* implementada no método RCDESIGN que implementa no ambiente de RC o critério de perda da capacidade de generalização (GL) do Proben1 [Pre94] e que não havia sido implementado antes neste ambiente.

O método RCDESIGN permite a investigação de diferentes soluções de RC para um determinado problema de forma mais rápida e mais eficiente do que através de experimentos aleatórios ou através de busca exaustiva.

Outra vantagem de RCDESIGN em relação aos métodos tradicionais de busca dos parâmetros de geração de RC é o custo computacional, pois como o método RCDESIGN não realiza as operações necessárias para reescalar a matriz de pesos da camada intermediária pelo raio espectral, ela apresenta um custo computacional inferior ao custo computacional dos outros dois. Os sistemas criados por RCDESIGN foram menores dos que os sistemas criados pelos outros métodos, e também menores que os sistemas de muitos outros trabalhos, o que implica em um menor custo computacional desses sistemas quando colocados em operação.

6

Conclusões e Proposta de Trabalhos Futuros

Este capítulo apresenta as conclusões e a proposta de trabalhos futuros.

6.1 Considerações Finais

Esta tese apresentou três métodos de otimização de *Reservoir Computing*, sendo dois métodos criados para otimização dos parâmetros de geração de RC, *Busca RS* e *Busca TR*, e um método para otimização simultânea dos parâmetros, da topologia da rede e dos pesos da camada intermediária, RCDESIGN.

O método da *Busca RS* cria sistemas com topologia fixa, apenas com as conexões obrigatórias de RC, dessa forma são válidas, para este método, as aproximações de RC a sistemas lineares. A *Busca RS* se baseia na receita clássica para produzir bons *reservoirs* que foi proposta por Jaeger em [Jae01] e em [Jae02b] e busca apenas pela quantidade de neurônios na camada intermediária, pelo raio espectral da matriz de pesos da camada intermediária e pela conectividade desta matriz. O raio espectral é um parâmetro muito importante neste método, pois ele considera a aproximação de RC a sistemas lineares, dessa forma o raio espectral neste método tem valor máximo de 1. Como foi explicado no Capítulo 3, a matriz de pesos da camada intermediária é considerada estável quando o raio espectral da mesma for menor ou igual a 1 [VS09].

O método da *Busca TR* não considera a aproximação de RC a sistemas lineares, dessa forma os sistemas criados por ele não possuem topologia fixa e o valor máximo do raio espectral da matriz de pesos da camada intermediária é $1,5$, valor bem superior ao limite de estabilidade. Mesmo sem considerar a aproximação de RC a sistemas lineares, a *Busca TR* ainda se baseia na receita clássica de criação de RC, pois os pesos da camada

intermediária são criados aleatoriamente e a matriz é reescalada por um raio espectral.

O método RCDESIGN é diferente das outras abordagens de criação de RC, pois ele não considera a aproximação de RC a sistemas lineares e nem utiliza a receita clássica de criação de RC. RCDESIGN combina algoritmo genético com RC e busca simultaneamente pelos melhores valores dos parâmetros, da topologia da rede e dos pesos, sem reescalar a matriz de pesos do *reservoir* pelo raio espectral. As vantagens desse método são que o design do sistema não depende da experiência do analista, já que ele permite a investigação automática da combinação de diferentes topologias, parâmetros e pesos, menor custo computacional, pois ela não realiza as operações necessárias para reescalar a matriz de pesos da camada intermediária pelo raio espectral e desempenho muito bom nas bases analisadas.

De forma geral o método RCDESIGN se mostrou eficiente e promissor em todas as bases estudadas, pois os resultados obtidos pelos sistemas criados por RCDESIGN foram melhores do que os resultados obtidos pela *Busca RS* e do que os resultados da *Busca TR*. Os resultados de RCDESIGN foram melhores que os resultados de muitos trabalhos acadêmicos.

6.2 Proposta de Trabalhos Futuros

Esta seção sugere alguns temas para pesquisas futuras usando o método RCDESIGN. Entre as perspectivas futuras, temos:

1. **Utilizar outros métodos de otimização dos pesos de *Reservoir Computing***. O método proposto, RCDESIGN, utiliza algoritmos genéticos para realizar a busca simultânea de melhores valores dos parâmetros, da topologia da rede e dos pesos da camada intermediária de RC. É interessante que outras técnicas de otimização como, por exemplo, otimização por enxame de partícula (PSO - *Particle Swarm Optimization*) sejam investigadas.

2. **Investigar o desempenho da técnica com outras bases reais**. O desempenho do método RCDESIGN foi avaliado com algumas séries clássicas e com três séries de velocidade média horária dos ventos na região nordeste do Brasil. É importante investigar a aplicação do método RCDESIGN em outras bases reais como, por exemplo, séries de velocidade dos ventos em outras localidades, séries de velocidade dos ventos em intervalo diferente do intervalo horário, séries de vazão, séries de demanda de energia elétrica.

3. **Verificar outros modelos para previsão da velocidade dos ventos**. Para o caso específico das bases reais foi adotado um modelo que utiliza uma entrada, uma saída e o horizonte de previsão de 24. Porém é importante verificar a aplicação de RCDESIGN com outras configurações de horizontes de previsão, ou com modelagens diferentes como, por exemplo, modelos diferentes para cada hora ou para cada estação do estação do ano, ou modelos que utilizem também como entrada a direção dos ventos ou a temperatura.

4. **Utilizar diferentes funções de ativação**. O método RCDESIGN pesquisa entre a função de ativação *tanh* e *sign*, mas existem outras possibilidades em RC como, por exemplo, a função linear ou neurônios pulsantes (LIF - *Leaky Integrate and Fire Neuron*), que devem ser investigadas.

5. **Adaptar o método proposto para problemas de classificação**. RCDESIGN foi utilizado com problemas de previsão de séries temporais, mas RC é uma técnica de inteligência artificial que tem apresentado excelentes resultados com problemas de classificação. É interessante a adaptação do método proposto para problemas de classificação.

6. **Investigar diferentes técnicas para criação do *reservoir***. Os resultados alcançados por RCDESIGN provam a necessidade de se investigar diferentes técnicas para criação da matriz de pesos da camada intermediária (*reservoir*), pois a tradicional, que cria os pesos aleatoriamente e depois reescalona por um raio espectral, é muito limitada e tem um custo computacional muito alto, principalmente devido à necessidade de calcular todos os autovalores desta matriz.

7. **Investigar o método RCDESIGN do ponto de vista matemático**. É necessária uma investigação matemática dos resultados alcançados por RCDESIGN em relação aos outros métodos.

Referências Bibliográficas

[ABBea99] E. Anderson, Z. Bai, C. Bischof, and et al. *LAPACK User's Guide.* SIAM, Dezembro 1999.

[AJL+10] R. R. B. Aquino, M. A. Carvalho Junior, M. M S. Lira, O. Nóbrega Neto, G. J. Almeida, and 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*, pages 3439–3444, Barcelona, 2010.

[ANL+07] R. R. B. Aquinoand, O. Nóbrega Neto, M. M. S. Lira, A. A. Ferreira, M. A. Carvalho Junior, G. B. Silva, , , and J. B. Oliveira. Development of an artificial neural network by genetic algorithm to mid-term load forecasting. In *International Joint Conference on Neural Networks - IJCNN 2007*, pages 1726–1731, Orlando, 2007.

[Aqu01] R. R. B Aquino. Redes neurais artificiais recorrentes: Uma aplicação a otimização da operação de sistemas hidrotérmicos de geração. Tese de Doutorado, Coordenação do Programa de Pós-Graduação em Engenharia Elétrica- COPELE/UFPB, Jan 2001.

[ASS08] E. A. Antonelo, B. Schrauwen, and D. Stroobandt. Event detection and localization for small mobile robots using reservoir computing. *Neural Networks*, 21(6):862–871, 2008.

[BA05] K. Bush and C. Anderson. Modeling reward functions for incomplete state representations via echo state networks. In *International Joint Conference on Neural Networks, IJCNN 2005*, volume 5, pages 2995–3000, 2005.

[BdLFCL07] A. P. Braga, A. C. P. de L. F. Carvalho, and T. B. Ludermir. *Redes Neurais Artificiais: Teoria e Aplicações.* 2ª edição, Livros Técnicos e Científicos, 2007.

[Bis06] C. M. Bishop. Pattern recognition and machine learning. In *Information Science and Statistics*. Springer, 2006.

[BM95] D. V. Buonomano and M. M. Merzenich. Temporal information transformed into a spatial code by a neural network with realistic properties. *Science*, 267:1028–1030, 1995.

[BN04] N. Bertschinger and T. Natschläger. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7):1413–1436, 2004.

[BOMA09] J. Boedecker, O. Obst, N. M. Mayer, and M. Asada. Studies on reservoir initialization and dynamics shaping in echo state networks. In *17th European Symposium on Artificial Neural Networks*, 2009.

[BT05] K. Bush and B. Tsendjav. Improving the richness of echo state features using next ascent local search. In *Artificial Neural Networks In Engineering Conference*, pages 227–232, 2005.

[Bur05a] H. Burgsteiner. On learning with recurrent spiking neural networks and their applications to robot control with real-world devices. Tese de Doutorado, Graz University of Technology, 2005.

[Bur05b] H. Burgsteiner. Training networks of biological realistic spiking neurons for real-time robot control. In *9th international conference on engineering applications of neural networks*, pages 129–136, 2005.

[Coe99] C.A.C. Coello. An updated survey of evolutionary multiobjective optimization techniques: state of the art and future trends. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, pages 3 vol. (xxxvii+2348), 1999.

[dA98] M. S. de Aguiar. Análise formal da complexidade de algoritmos genéticos. Dissertação de Mestrado, Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Abril 1998.

[DBN07] X. Dutoit, H. V. Brussel, and M. Nuttin. A first attempt of reservoir pruning for classification problems. In *15th European Symposium on Artificial Neural Networks, ESANN 2007*, pages 507–512, 2007.

[DBS08] A. Devert, N. Bredeche, and M. Schoenauer. Unsupervised learning of echo state networks: a case study in artificial embryogeny. In *8th International Conference on Artificial Evolution*, volume 4926, pages 278–290. Springer, 2008.

[DKW95] A. Delgado, C. Kambhampati, and K. Warwick. Dynamic recurrent neural network for system identification and control. In *IEEE Proceedings of Control Theory and Applications*, volume 142(4), pages 307–314, 1995.

[Dom95] P. F. Dominey. Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. *Biological Cybernetics*, 73(3):265–274, 1995.

[end10a] *ONS, Operador Nacional do Sistema Elétrico.* http://www.ons.org.br, Dezembro 2010.

[end10b] *PROINFA, Programa de Incentivo às Fontes Alternativas de Energia Elétrica, Ministério das Minas e Energia.* http://www.mme.gov.br/programas/proinfa, Agosto 2010.

[end10c] *SONDA, Sistema de Organização Nacional de Dados Ambientais.* http://sonda.cptec.inpe.br/, Agosto 2010.

[FB98] B. Farhang-Boroujeny. *Adaptive Filters: Theory and Applications.* Wiley, 1998.

[Fer09] J. M. A. Ferreira. *Preservação de Longa Duração de Informação Digital no Contexto de um Arquivo Histórico.* PhD thesis, Escola de Engenharia da Universidade do Minho, 2009.

[FL08a] A. A. Ferreira and T. B. Ludermir. Algoritmo genético para otimização de reservoir computing: Uma primeira tentativa. In *II Workshop on Computational Intelligence (WCI), 2008. Anais do II Workshop on Computational Intelligence (WCI)*, Salvador, 2008.

[FL08b] A. A. Ferreira and T. B. Ludermir. Using reservoir computing for forecasting time series: Brazilian case study. In *International Conference on Hybrid Intelligent Systems - HIS 2008*, pages 602–607, Los Alamitos, CA, USA, 2008.

[FL09] A. A. Ferreira and T. B. Ludermir. Genetic algoritm for reservoir computing optimization. In *International Joint Conference on Neural Networks - IJCNN 2009*, pages 811–815, Atlanta, 2009.

[FL10] A. A. Ferreira and T. B. Ludermir. Evolutionary strategy for simultaneous optimization of parameters, topology and reservoir weights in echo state networks. In *International Joint Conference on Neural Networks - IJCNN 2010*, pages 1870–1876, Barcelona, 2010.

[FLdA+08] A. A. Ferreira, T. B. Ludermir, R. R. B. de Aquino, M. M. Lira, and O. N. Neto. Investigating the use of reservoir computing for forecasting the hourly wind speed in short-term. In *International Joint Conference on Neural Networks - IJCNN 2008*, pages 1950–1957, Hong Kong, 2008.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, 1989.

[Hay01] S. Haykin. *Redes Neurais: Princípios e Prática.* ARTMED Editora S. A, Brasil, 2001.

[Hin81] G. E. Hinton. Implementing semantic networks in parallel hardware. *Parallel models of associative memory*, pages 161–187, 1981.

[Hol92] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology Control and Artificial Intelligence.* MIT Press, Cambridge, MA, USA, 1992.

[IvdZBP04] K. Ishii, T. van der Zant, V. Becanovic, and P. Ploger. Identification of motion with echo state network. In *OCEANS 2004 MTS/IEEE – TECHNO-OCEAN 2004 Conference*, volume 3, pages 1205–1210, 2004.

[Jae01] H. Jaeger. The echo state approach to analyzing and training recurrent neural networks. Technical report, GDM 148, German National Resource Center for Information Technology, 2001.

[Jae02a] H. Jaeger. Short-term memory in echo state networks. Technical report, GDM 152, German National Resource Center for Information Technology, 2002.

[Jae02b] H. Jaeger. Tutorial on training recurrent neural networks, covering bptt, rtrl, ekf and the echo state network approach. Technical report, GDM 159, German National Resource Center for Information Technology, 2002.

[Jae03] H. Jaeger. Adaptive nonlinear system identification with echo state networks. *Advances in Neural Information Processing Systems*, 15:593–600, 2003.

[Jae05] H. Jaeger. Reservoir riddles: Suggestions for echo state network research. In *International Joint Conference on Neural Networks - IJCNN 2005*, volume 3, pages 1460–1462, 2005.

[Jae07] H. Jaeger. Discovering multiscale dynamical features with hierarchical echo state networks. Technical Report 9, Jacobs University Bremen, 2007.

[JH04] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless telecommunication. *Science*, 308:78–80, 2004.

[JM04] P. Joshi and W. Maass. Movement generation and control with generic neural microcircuits. *Lecture Notes in Computer Science, Biologically Inspired Approaches to Advanced Information Technology First International Workshop, BioADIT 2004*, 3141:16–31, 2004.

[JMP07] H. Jaeger, W. Maass, and J. Principe. Special issue on echo state networks and liquid state machines. *Neural Networks*, 20(3):287–289, 2007.

[KHM+06] G. N. Kariniotakis, J. Halliday, I. Marti, A. Palomares, H. Madsen, U. Focken, G. Kallos, N. Hatziargyriou, P. Frayssinet, and et al. Next generation short-term forecasting of wind power - overview of the anemos project. In *European Wind Energy Conferences and Exhibitions, EWEC*, Athens, Greece, 2006.

[KZM94] G. Kechriotis, E. Zervas, and E. S. Manolakos. Using recurrent neural networks for adaptive communication channel equalization. *IEEE transactions on Neural Networks*, 5(2):267–278, 1994.

[Lie04] B. Liebald. Exploration of effects of different network topologies on the esn signal crosscorrelation matrix spectrum. Dissertação de Mestrado, Jacobs University Bremen, 2004.

[LJ09] M. Lukosevicius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[LM07] R. Legenstein and W. Maass. *New Directions in Statistical Signal Processing: From Systems to Brain*, chapter What makes a dynamical system computationally powerful?, pages 127–154. MIT Press, 2007.

[LPM09] R. Legenstein, D. Pecevski, and W. Maass. A learning theory for reward-modulated spiketimingdependent plasticity with application to biofeedback. *PLoS Computational Biology*, 4(10), 2009.

[Maa97] W. Maass. *Advances in neural information processing systems*, volume 9, chapter Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons., pages 211–217. MIT Press, Cambridge, 1997.

[MB01] W. Maass and C. Bishop. *Pulsed neural networks.* MIT Press, Cambridge, MA, 2001.

[MLB05] W. Maass, R. A. Legenstein, and N. Bertschinger. Methods for estimating the computational power and generalization capability of neural microcircuits. *Advances in Neural Information Processing Systems NIPS 2004*, 17:865–872, 2005.

[MLM02] W. Maass, R. A. Legenstein, and H. Markram. A new approach towards vision suggested by biologically realistic neural microcircuit models. *Second Workshop on Biologically Motivated Computer Vision*, 2525:282–293, 2002.

[MNM02] W. Maass, T. Natschlager, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[MNM03] W. Maass, T. Natschlager, and H. Markram. *Advances in Neural Information Processing Systems 15, NIPS 2002*, volume 15, chapter A model for real-time computation in generic neural microcircuits., pages 213–220. MIT Press, Cambridge, MA, 2003.

[MNM04] W. Maass, T. Natschlager, and H. Markram. Fading memory and kernel properties of generic cortical microcircuits models. *Journal of Physiology*, 98(4-6):315–330, 2004.

[MP69] M. Minsky and S. Papert. *Perceptrons: an introduction to computational geometry*. MIT Press, 1969.

[NBL04] T. Natschlager, N. Bertschinger, and R. Legenstein. At the edge of chaos: Real-time computations and self-organized criticality in recurrent neural networks. In *Neural Information Processing Systems - NIPS 2004*, pages 145–152, 2004.

[OLSB05] M. Oubbati, P. Levi, M. Schanz, and T. Buchheim. Velocity control of an omnidirectional robocup player with recurrent networks. In *Robocup Symposium 2005*, pages 691–701, 2005.

[OP05] M. C. Ozturk and J. C. Principe. Computing with transiently stable states. In *International Joint Conference on Neural Networks, IJCNN 2005*, volume 3, pages 1467–1472, 2005.

[OXP07] M. C. Ozturk, D. Xu, and J. C. Principe. Analysis and design of echo state networks. *Neural Computation*, 19(1):111–138, 2007.

[PF95] G. V. Puskorius and L. A. Feldkamp. Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks. *IEEE Transactions on Neural Networks*, 5(2):279–297, 1995.

[Pre94] L. Prechelt. Proben1 - a set of neural network benchmark problems and benchmarking rules. Technical report, 21/94, Fakultät für Informatik, Universität Karlsruhe, Germany, 1994.

[RJH08] A. A. Rad, M. Jalili, and M. Hasler. Reservoir optimization in recurrent neural networks using kronecker kernels. In *IEEE International Symposium on Circuits and Systems 2008*, pages 868–871, 2008.

[Roc10] R. T. Rockafellar. *Fundamentals of Optimization.* http://www.math.washington.edu/ rtr/fundamentals.pdf, 2010.

[Ros62] F. Rosenblatt. *Principles of neurodynamics: Perceptrons and the theory of brain mechanismis.* Spartan Books, 1962.

[SDVC07a] B. Schrauwen, J. Defour, D. Verstraeten, and J. Van Campenhout. The introduction of time-scales in reservoir computing, applied to isolated digits recognition. *17th International Conference on Artificial Neural*

Networks, in: Lecture Notes in Computer Science, 4668, Part I:471–479, 2007.

[SDVC07b] B. Schrauwen, M. D'Haener, D. Verstraeten, and J. Van Campenhout. Compact hardware for real-time speech recognition using a liquid state machine. In *International Joint Conference on Neural Networks, IJCNN 2007*, pages 1097–1102, 2007.

[SH07] Z.i Shi and M. Han. Support vector echostate machine for chaotic time-series prediction. *IEEE Transactions on Neural Networks*, 18(2):359–372, 2007.

[SOP+07] R. Sacchi, M. C. Ozturk, J. C. Principe, A. A. F. Carneiro, and I. N. Silva. Water inflow forecasting using the echo state network: a brazilian case study. In *International Conference on Neural Networks, IJCNN 2007*, pages 2403–2408, Orlando, Florida, 2007.

[SP05] M. Salmen and P. G. Ploger. Echo state networks used for motor control. In *International Conference on Robotics and Automation, ICRA 2005*, pages 1953–1958, 2005.

[SS05] U. D. Schiller and J. J. Steil. Analyzing the weigtht dynamic of recurrent learning algorithms. *Neurocomputing*, 63C:5–23, 2005.

[Ste04] J. J. Steil. Backpropagation–decorrelation: Online recurrent learning with o(n) complexity. In *International Joint Conference on Neural Networks, IJCNN 2004*, volume 2, pages 843–848, 2004.

[Ste05] J. J. Steil. Memory in backpropagation-decorrelation o(n) efficient online recurrent learning. In *15th International Conference on Artificial Neural Networks, in: Lecture Notes in Computer Science*, volume 3697, pages 649–654, 2005.

[Ste06] J. J. Steil. Online stability of backpropagation-decorrelation recurrent learninge. *Neurocomputing*, 69:642–650, 2006.

[SVC07] B. Schrauwen, D. Verstraeten, and J. V. Campenhout. An overview of reservoir computing: theory, applications and implementations. In *15th European Symposium on Artificial Neural Networks*, pages 471–482, 2007.

[SVH10] B. Schrauwen, D. Verstraeten, and M. D Haene. *Reservoir Computing Toolbox Manual*. http://reslab.elis.ugent.be/rctoolbox, Dezembro 2010.

[SWGG07] J. Schmidhuber, D. Wierstra, M. E. Gagliolo, and F. Gomez. Training recurrent networks by evolino. *Neural Computation*, 19(3):757–779, 2007.

[SWV+08] B. Schrauwen, M. Wardermann, D. Verstraeten, J. J. Steil, and D. Stroobandt. Improving reservoirs using intrinsic plasticity. *Neurocomputing*, 71:1159–1171, 2008.

[Syw89] G. Sywerda. Uniform crossover in genetic algorithms. In *Third International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann Publishers Inc, 1989.

[Tak81] F. Takens. Detecting strange attractors in turbulence. In David Rand and Lai-Sang Young, editors, *Dynamical Systems and Turbulence, Warwick 1980*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. Springer Berlin / Heidelberg, 1981.

[TV01] L. V. Toscani and P. A. S. Veloso. *Complexidade de algoritmos: análise, projeto e métodos.* Bookman, Porto Alegre, 2001.

[Vap95] V. Vapnik. *The nature of statistical learning theory.* Spring-Verlag, New York, 1995.

[VS09] D. Verstraeten and B. Schrauwen. On the quantification of dynamics in reservoir computing. *19th International Conference on Artificial Neural Networks, in: Lecture Notes in Computer Science*, 5768:985–994, 2009.

[VSDS07] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403, 2007.

[VSSC05] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout. Isolated word recognition with the liquid state machine: a case study. *Information Processing Letters*, 95(6):521–528, 2005.

[Wai07] J. Wainer. Métodos de pesquisa quantitativa e qualitativa para a ciência computação. *Atualização em Informática da Sociedade Brasileira de Computação*, pages 221–262, 2007.

[Wal05] S. Walson. Fresh forecasts [wind power forecasting]. *Power Engineer*, 19(2):36–38, 2005.

[WF00] I. H Witten and E. Frank. *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann Publishers, 2000.

[WSVS08] F. Wyffels, B. Schrauwen, D. Verstraeten, and D. Stroobandt. Band-pass reservoir computing. In *International Joint Conference on Neural Networks 2008*, pages 3203–3208, 2008.

[XLP05] D. Xu, J. Lan, and J. C. Príncipe. Direct adaptive control: An echo state network and genetic algorithm approach. In *International Joint Conference on Neural Networks 2005*, pages 1483–1486, 2005.

[Yam04] A. Yamazaki. Uma metodologia para otimização de arquiteturas e pesos de redes neurais. Tese de Doutorado, Centro de Informática, UFPE, 2004.

[Zad71] L. A. Zadeh. *Aspects of networks and system theory.*, chapter Toward a theory of fuzzy system, pages 469–490. Rinehart and Winston, 1971.

[Zan08] C. Zanchettin. Otimização global em redes neurais artificiais. Tese de Doutorado, Centro de Informática, UFPE, 2008.

[ZBI+04] T. Van Der Zant, V. Becanovic, K. Ishii, H. U. E. Kobialka, and P. G. Plöger. Finding good echo state networks to control an underwater robot using evolutionary computations. In *5th symposium on intelligent autonomous vehicles - IAV 2004*, 2004.

Appendices

A

Algoritmo Genético

Algoritmos genéticos são métodos para resolver problemas de otimização com ou sem restrições baseado no processo de seleção natural, o processo que guiou a evolução biológica. O algoritmo genético modifica a população de soluções individuais repetidas vezes. Em cada passo, o algoritmo seleciona aleatoriamente indivíduos da população atual para serem pais e os utiliza para produzirem filhos para a próxima geração. Depois de sucessivas gerações, a população evolui para uma solução. Algoritmos genéticos podem ser aplicados na solução de uma grande variedade de problemas de otimização.

Os algoritmos genéticos incorporam o conhecimento específico sobre o domínio do problema a fim de realizar o processo de otimização, tolerando alguns elementos não determinantes que os ajudam a escapar de mínimos locais. Além disso, eles possuem uma função de custo adequada para cada problema, tornando-se amplamente utilizado.

Segundo Goldberg [Gol89] os algoritmos genéticos difere dos métodos tradicionais de busca e otimização em quatro aspectos principais, sumarizados na Tabela A.1

A.1 Terminologia

- **Função objetivo**

  A função objetivo é a função que se deseja otimizar.

- **Indivíduos**

  Um indivíduo é qualquer ponto no qual é possível aplicar a função objetivo. O valor da função objetivo para um indivíduo é sua *aptidão*. Um indivíduo é algumas vezes referenciado como um genoma e o vetor de entrada (ou string genética) de um indivíduo como cromossomo.

Tabela A.1 Algoritmos Genéticos x Métodos Tradicionais de Busca e Otimização

Algoritmos Genéticos	***Métodos Tradicionais de Busca e Otimização***
Trabalham com uma codificação do conjunto de parâmetros.	Trabalham com os próprios parâmetros.
Trabalham com uma população de soluções candidatas em cada iteração. O melhor ponto na população aproxima uma solução ótima.	Trabalham com uma única solução em cada iteração. A sequência de pontos aproxima uma solução ótima.
Utilizam informação de custo ou recompensa.	Utilizam informações derivadas de funções.
Utilizam regras de transição probabilísticas.	Utilizam regras determinísticas.

- **População e Gerações**

  Uma população é um conjunto de indivíduos. Por exemplo, se o tamanho da população é 100 e o número de variáveis na função objetivo é três, essa população será representada por uma matriz $100x3$. O mesmo indivíduo pode aparecer mais de uma vez na população. Em cada iteração, o algoritmo genético executa uma série de cálculos na população atual para produzir uma nova população. Cada população sucessiva é chamada de nova geração.

- **Diversidade**

  Diversidade se refere à distância média entre os indivíduos de uma população. Uma população tem alta diversidade se a distância média é alta; caso contrário ela tem uma baixa diversidade. Diversidade é essencial para algoritmo genético porque ela habilita o algoritmo a pesquisar em um grande espaço de busca.

- **Pais e Filhos**

  Para criar a próxima geração, o algoritmo genético seleciona certos indivíduos na população corrente, chamados pais, e os utiliza para criar os indivíduos na próxima geração, chamados filhos.

- **Valor de Aptidão e Melhor Valor de Aptidão** O valor de *aptidão* de um indivíduo é o valor de avaliação da função de *aptidão* para aquele indivíduo. O melhor valor

de *aptidão* para uma população é um menor (para problemas de minimização) valor de *aptidão* entre todos os valores de *aptidão* dos indivíduos da população.

A.2 Pseudo-código do Algoritmo Genético

1. O algoritmo começa criando uma população inicial aleatória.
2. O algoritmo cria uma sequência de novas populações. Em cada passo, o algoritmo usa os indivíduos da população corrente para criar uma nova população. Para criar a nova população o algoritmo executa os seguintes passos:
   (a) Cada membro da população corrente recebe uma pontuação de acordo com seu valor de *aptidão*.
   (b) Os membros da população corrente são selecionados como *pais* baseado em seus valores de *aptidão* (ver Seção A.3).
   (c) *Filhos* são produzidos a partir de mudanças aleatórias (*mutação*) dos genes de um único indivíduo ou combinando os genes de um par de indivíduos (*cruzamento*).
   (d) A população corrente é trocada com os *filhos* para formar a próxima geração.
3. O algoritmo pára quando um dos critérios de parada é alcançado (ver Seção A.5).

A.3 Regras de Seleção

As funções de seleção trabalham com valores de *aptidão* para escolherem os *pais* para a próxima geração. Um indivíduo pode ser escolhido mais de uma vez como pai, dessa forma ele vai contribuir com seus genes para mais de um filho. A função de seleção atribui uma maior probabilidade de seleção para os indivíduos com maiores valores em escala.

O método de seleção afeta o desempenho do algoritmo genético. Se usarmos apenas pais com excelentes valores de *aptidão* poderemos estar jogando fora bons esquemas presentes nos indivíduos "ruins", porém se permitimos muito que os indivíduos com valores de *aptidão* ruins participem do processo reprodutivo, os esquemas que os tornam ruins não desaparecerão da população.

Alguns dos critérios para seleção de pais em algoritmos genéticos são:

- **Uniforme Estocástica** - Os pais são escolhidos por uma linha em que cada indivíduo corresponde a um segmento da linha. O tamanho de cada segmento é proporcional ao valor da *aptidão* do indivíduo que está sendo mapeado. O algoritmo se move ao longo da linha em passos de mesmo tamanho. Os indivíduos *donos* dos segmentos apontados são selecionados como pais.

- **Roleta** - Escolhe pais pela simulação de uma roleta, onde os indivíduos são ordenados de acordo com seu valor de *aptidão*, com probabilidade decrescente de serem escolhidos. A escolha é feita então aleatoriamente de acordo com essas probabilidades. Dessa forma são escolhidos como pais os mais bem adaptados, sem deixar de lado a diversidade dos menos adaptados.

- **Torneio** - Escolhe os pais pela simulação de um torneio (competição). Cada torneio escolhe indivíduos aleatoriamente e então esses indivíduos passam a competir usando o valor de *aptidão* como arma. O indivíduo vencedor é escolhido como um pai.

- **Ranking** - O valor bruto de *aptidão* da cada indivíduo é substituído pelo seu valor de *rank*. O valor de *rank* de um indivíduo é sua posição entre todas as *aptidões* ordenadas. Por exemplo: o *rank* do indivíduo mais apto é 1, do segundo mais apto é 2, e assim por diante. Uma vez estabelecido o valor de *rank* de cada indivíduo utiliza-se qualquer outro método para seleção dos pais.

A.4 Reprodução

Os indivíduos escolhidos para serem pais podem passar por combinação ou modificação para produzirem indivíduos da próxima geração. Essas combinações e modificações são realizadas por operadores genéticos. Os operadores genéticos são utilizados para transformarem uma população através de sucessivas gerações, estendendo a busca até a obtenção de um resultado satisfatório. Os principais operadores de reprodução são: cruzamento (ou crossover) e mutação. Eles são utilizados para assegurar que a nova geração apresente novos indivíduos, mantendo características desejáveis adquiridas em gerações anteriores. Para prevenir a eliminação dos melhores indivíduos utiliza-se o operador de elitismo [BdLFCL07].

A.4.1 Cruzamento

O operador de cruzamento dos algoritmos genéticos é responsável pela recombinação de características genéticas dos pais, permitindo que elas sejam herdadas pelas próximas gerações. Ele é considerado o operador predominante e é aplicado com uma probabilidade, chamada de neste trabalho de *probabilidade de cruzamento*.

A.4.2 Mutação

O operador de mutação é responsável pela introdução e manutenção da diversidade genética na população, alterando arbitrariamente um ou mais genes de um indivíduo (cromossomo) escolhido aleatoriamente. Este operador fornece meios para introdução de novos elementos na população. A mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será zero, além de contornar o problema de mínimos locais. O operador de mutação é aplicado aos indivíduos com uma probabilidade dada pela *taxa de mutação*, que é geralmente pequena, pois uma taxa muito alta torna a busca essencialmente aleatória. A Figura A.1 ilustra a operação da mutação.

Figura A.1 Operador de mutação.

A.4.3 Elitismo

O operador de elite é responsável por copiar o(s) melhor(es) indivíduo(s) da geração atual na próxima geração a fim de evitar que todos os bons cromossomos sejam alterados pelo cruzamento e pela mutação, desta forma, o uso de elitismo permite que o algoritmo convirja mais rapidamente para uma solução.

A.5 Critérios de Parada

O processo evolucionário termina quando algum critério de parada é atingido. Os critérios de parada mais comuns em algoritmos genéticos são:

- **Gerações** - O algoritmo pára quando o número máximo de gerações é alcançado.
- **Limite de tempo** - Tempo máximo de execução do algoritmo.
- **Limite da função objetivo** - O algoritmo pára quando o valor da função objetivo do melhor ponto da população atual é menor do que o limite definido.
- **Convergência** - Não ocorre melhora significativa na solução durante um dado número de gerações.

B

Comparação do Desempenho dos Métodos pelo Boxplot

A Figura B.1, Figura B.2, Figura B.3, Figura B.4, Figura B.5, Figura B.6, Figura B.7, Figura B.8, Figura B.9 e Figura B.10 exibem graficamente a comparação do *MSE* dos três métodos no conjunto de teste.

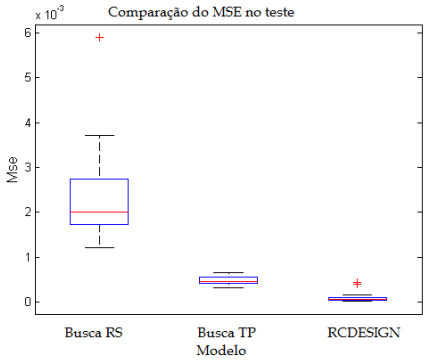

Figura B.1 NARMA ordem 10: Comparação do desempenho dos métodos no conjunto de teste.

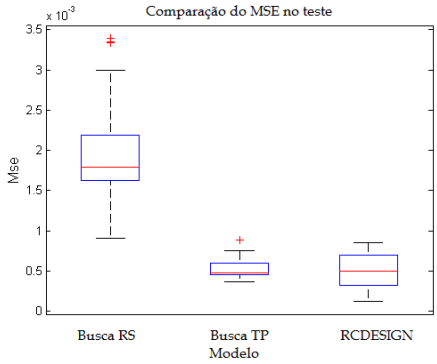

Figura B.2 NARMA ordem 30: Comparação do desempenho dos métodos no conjunto de teste.

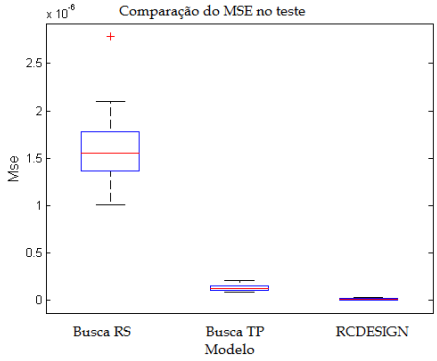

Figura B.3 MGS caos médio: Comparação do desempenho dos métodos no conjunto de teste.

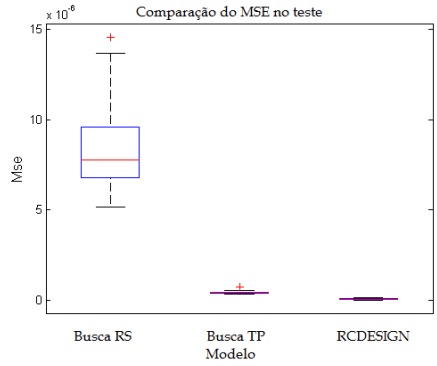

Figura B.4 MGS caos moderado: Comparação do desempenho dos métodos no conjunto de teste.

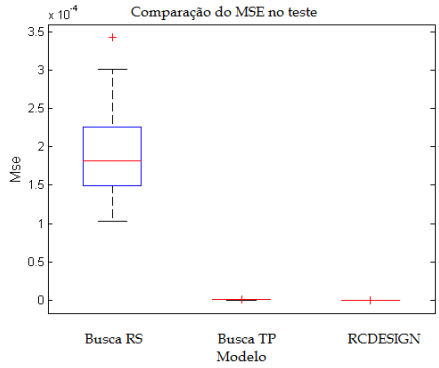

Figura B.5 MSO: Comparação do desempenho dos métodos no conjunto de teste.

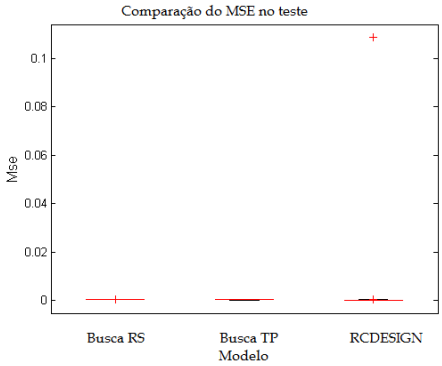

Figura B.6 *STAR*: Comparação do desempenho dos métodos no conjunto de teste.

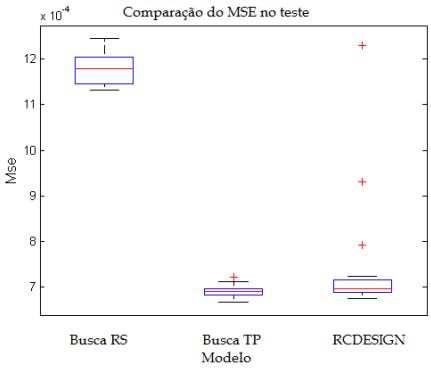

Figura B.7 *Down-Jones*: Comparação do desempenho dos métodos no conjunto de teste.

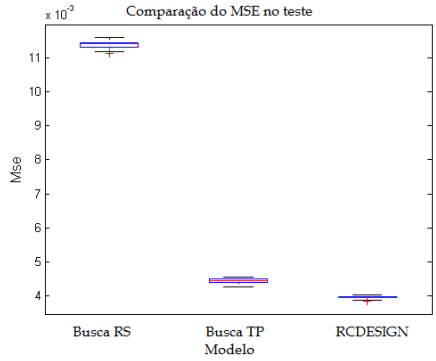

Figura B.8 Belo Jardim: Comparação do desempenho dos métodos no conjunto de teste.

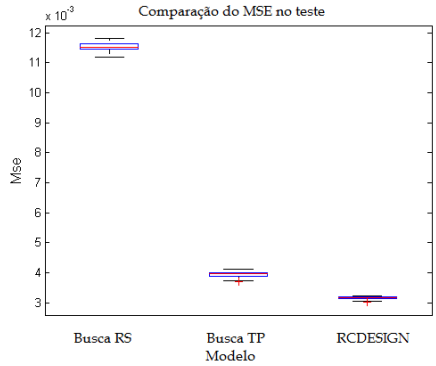

Figura B.9 São João do Cariri: Comparação do desempenho dos métodos no conjunto de teste.

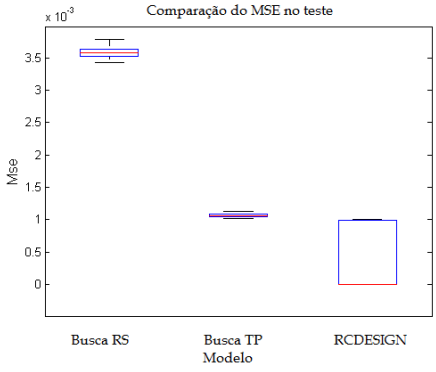

Figura B.10 Triunfo: Comparação do desempenho dos métodos no conjunto de teste.

C

MSE no Teste Por Inicialização

Tabela C.1 Narma ordem 10

	RCDESIGN	*Busca RS*	*Busca TR*
1	0,00008239(0,0001)	0,00191964(0,0036)	0,00048057(0,0009)
2	0,00004383(0,0001)	0,00166667(0,0030)	0,00036245(0,0006)
3	0,00003739(0,0001)	0,00273763(0,0056)	0,00043436(0,0007)
4	0,00002816(0,0000)	0,00371295(0,0114)	0,00056603(0,0011)
5	0,00008195(0,0001)	0,00288910(0,0065)	0,00057906(0,0011)
6	0,00004161(0,0001)	0,00588921(0,0186)	0,00042343(0,0008)
7	0,00011791(0,0002)	0,00257173(0,0059)	0,00037342(0,0008)
8	0,00005238(0,0001)	0,00216958(0,0036)	0,00056707(0,0009)
9	0,00003485(0,0001)	0,00154227(0,0023)	0,00059518(0,0010)
10	0,00016673(0,0002)	0,00163369(0,0023)	0,00049487(0,0009)
11	0,00003723(0,0001)	0,00202081(0,0039)	0,00036552(0,0007)
12	0,00010416(0,0005)	0,00290436(0,0042)	0,00039854(0,0007)
13	0,00004442(0,0001)	0,00179442(0,0034)	0,00046645(0,0009)
14	0,00012711(0,0002)	0,00182542(0,0038)	0,00045605(0,0007)
15	0,00004451(0,0001)	0,00122034(0,0025)	0,00050281(0,0010)
16	0,00040386(0,0007)	0,00184274(0,0031)	0,00064618(0,0012)
17	0,00007054(0,0001)	0,00185263(0,0028)	0,00048162(0,0010)
18	0,00005763(0,0001)	0,00241730(0,0046)	0,00050476(0,0009)
19	0,00010008(0,0002)	0,00172477(0,0027)	0,00046082(0,0009)
20	0,00005705(0,0001)	0,00167395(0,0024)	0,00057350(0,0011)
21	0,00004008(0,0001)	0,00160762(0,0026)	0,00043071(0,0008)
22	0,00007035(0,0002)	0,00206670(0,0035)	0,00047680(0,0009)
23	0,00005571(0,0001)	0,00162321(0,0026)	0,00055496(0,0010)
24	0,00005987(0,0001)	0,00286347(0,0047)	0,00045727(0,0007)
25	0,00004792(0,0001)	0,00176440(0,0027)	0,00046231(0,0008)
26	0,00010333(0,0002)	0,00238546(0,0040)	0,00042205(0,0007)
27	0,00006723(0,0001)	0,00287267(0,0057)	0,00045013(0,0007)
28	0,00003838(0,0001)	0,00302387(0,0061)	0,00035069(0,0007)
29	0,00043226(0,0008)	0,00199384(0,0028)	0,00066058(0,0012)
30	0,00011147(0,0002)	0,00262059(0,0048)	0,00032198(0,0005)
Média	0,00009201(0,0002)	0,00229437(0,0045)	0,00047734(0,0009)
Desv. Pad.	0,00009460(0,0002)	0,00088976(0,0032)	0,00008597(0,0002)

Tabela C.2 Narma ordem 30

	RCDESIGN	*Busca RS*	*Busca TR*
1	0,00069233(0,0011)	0,00245242(0,0036)	0,00047955(0,0008)
2	0,00058557(0,0010)	0,00154042(0,0022)	0,00046759(0,0007)
3	0,00031837(0,0005)	0,00177550(0,0026)	0,00055553(0,0010)
4	0,00018397(0,0003)	0,00091122(0,0016)	0,00044572(0,0009)
5	0,00057297(0,0009)	0,00335343(0,0085)	0,00053574(0,0008)
6	0,00047411(0,0008)	0,00338958(0,0092)	0,00054617(0,0010)
7	0,00046274(0,0008)	0,00299364(0,0067)	0,00059487(0,0011)
8	0,00083813(0,0014)	0,00213305(0,0028)	0,00036235(0,0006)
9	0,00038530(0,0006)	0,00182286(0,0027)	0,00042103(0,0007)
10	0,00077813(0,0012)	0,00162205(0,0025)	0,00066688(0,0012)
11	0,00029999(0,0005)	0,00291928(0,0056)	0,00087944(0,0015)
12	0,00042413(0,0008)	0,00170452(0,0033)	0,00045081(0,0009)
13	0,00026564(0,0004)	0,00181975(0,0026)	0,00037482(0,0007)
14	0,00030966(0,0005)	0,00334121(0,0065)	0,00045289(0,0010)
15	0,00020939(0,0003)	0,00226655(0,0041)	0,00059378(0,0010)
16	0,00084614(0,0013)	0,00152465(0,0024)	0,00042409(0,0009)
17	0,00029568(0,0005)	0,00175372(0,0024)	0,00073904(0,0014)
18	0,00056460(0,0009)	0,00146670(0,0025)	0,00046168(0,0008)
19	0,00057294(0,0010)	0,00168829(0,0025)	0,00047779(0,0009)
20	0,00071839(0,0014)	0,00149710(0,0022)	0,00043905(0,0008)
21	0,00070615(0,0010)	0,00219006(0,0048)	0,00075708(0,0013)
22	0,00052567(0,0009)	0,00126927(0,0023)	0,00056989(0,0010)
23	0,00043159(0,0007)	0,00142356(0,0019)	0,00046195(0,0008)
24	0,00012546(0,0002)	0,00173104(0,0029)	0,00051514(0,0009)
25	0,00042673(0,0007)	0,00180238(0,0028)	0,00046958(0,0007)
26	0,00078625(0,0013)	0,00205728(0,0037)	0,00062359(0,0011)
27	0,00034991(0,0006)	0,00198545(0,0034)	0,00046635(0,0009)
28	0,00073458(0,0013)	0,00170774(0,0027)	0,00060716(0,0010)
29	0,00065590(0,0013)	0,00206688(0,0034)	0,00043959(0,0008)
30	0,00060434(0,0010)	0,00164506(0,0024)	0,00053664(0,0010)
Média	0,00050483(0,0008)	0,00199516(0,0036)	0,00052719(0,0009)
Desv. Pad.	0,00020485(0,0004)	0,00063196(0,0019)	0,00011728(0,0002)

Tabela C.3 MGS caos médio

	RCDESIGN	*Busca RS*	*Busca TR*
1	0,00000001(0,0000)	0,00000182(0,0000)	0,00000011(0,0000)
2	0,00000002(0,0000)	0,00000120(0,0000)	0,00000013(0,0000)
3	0,00000001(0,0000)	0,00000210(0,0000)	0,00000009(0,0000)
4	0,00000003(0,0000)	0,00000149(0,0000)	0,00000015(0,0000)
5	0,00000001(0,0000)	0,00000101(0,0000)	0,00000017(0,0000)
6	0,00000001(0,0000)	0,00000116(0,0000)	0,00000011(0,0000)
7	0,00000002(0,0000)	0,00000175(0,0000)	0,00000011(0,0000)
8	0,00000001(0,0000)	0,00000171(0,0000)	0,00000012(0,0000)
9	0,00000001(0,0000)	0,00000155(0,0000)	0,00000021(0,0000)
10	0,00000001(0,0000)	0,00000149(0,0000)	0,00000012(0,0000)
11	0,00000001(0,0000)	0,00000142(0,0000)	0,00000010(0,0000)
12	0,00000002(0,0000)	0,00000201(0,0000)	0,00000011(0,0000)
13	0,00000001(0,0000)	0,00000167(0,0000)	0,00000010(0,0000)
14	0,00000002(0,0000)	0,00000126(0,0000)	0,00000011(0,0000)
15	0,00000003(0,0000)	0,00000184(0,0000)	0,00000016(0,0000)
16	0,00000001(0,0000)	0,00000132(0,0000)	0,00000019(0,0000)
17	0,00000001(0,0000)	0,00000153(0,0000)	0,00000014(0,0000)
18	0,00000002(0,0000)	0,00000165(0,0000)	0,00000018(0,0000)
19	0,00000001(0,0000)	0,00000154(0,0000)	0,00000010(0,0000)
20	0,00000001(0,0000)	0,00000189(0,0000)	0,00000015(0,0000)
21	0,00000001(0,0000)	0,00000115(0,0000)	0,00000012(0,0000)
22	0,00000001(0,0000)	0,00000165(0,0000)	0,00000014(0,0000)
23	0,00000002(0,0000)	0,00000131(0,0000)	0,00000013(0,0000)
24	0,00000001(0,0000)	0,00000149(0,0000)	0,00000014(0,0000)
25	0,00000001(0,0000)	0,00000178(0,0000)	0,00000014(0,0000)
26	0,00000001(0,0000)	0,00000206(0,0000)	0,00000020(0,0000)
27	0,00000003(0,0000)	0,00000136(0,0000)	0,00000013(0,0000)
28	0,00000002(0,0000)	0,00000278(0,0000)	0,00000015(0,0000)
29	0,00000001(0,0000)	0,00000155(0,0000)	0,00000010(0,0000)
30	0,00000001(0,0000)	0,00000167(0,0000)	0,00000013(0,0000)
Média	0,00000001(0,0000)	0,00000161(0,0000)	0,00000014(0,0000)
Desv. Pad.	0,00000001(0,0000)	0,00000035(0,0000)	0,00000003(0,0000)

Tabela C.4 MGS caos moderado

	RCDESIGN	*Busca RS*	*Busca TR*
1	$0,00000002(0,0000)$	$0,00001103(0,0000)$	$0,00000039(0,0000)$
2	$0,00000004(0,0000)$	$0,00000642(0,0000)$	$0,00000053(0,0000)$
3	$0,00000005(0,0000)$	$0,00000781(0,0000)$	$0,00000043(0,0000)$
4	$0,00000008(0,0000)$	$0,00001001(0,0000)$	$0,00000041(0,0000)$
5	$0,00000003(0,0000)$	$0,00000518(0,0000)$	$0,00000047(0,0000)$
6	$0,00000010(0,0000)$	$0,00000794(0,0000)$	$0,00000037(0,0000)$
7	$0,00000002(0,0000)$	$0,00000773(0,0000)$	$0,00000045(0,0000)$
8	$0,00000006(0,0000)$	$0,00000713(0,0000)$	$0,00000048(0,0000)$
9	$0,00000007(0,0000)$	$0,00000583(0,0000)$	$0,00000050(0,0000)$
10	$0,00000010(0,0000)$	$0,00000679(0,0000)$	$0,00000046(0,0000)$
11	$0,00000005(0,0000)$	$0,00001110(0,0000)$	$0,00000039(0,0000)$
12	$0,00000003(0,0000)$	$0,00001454(0,0000)$	$0,00000073(0,0000)$
13	$0,00000007(0,0000)$	$0,00000674(0,0000)$	$0,00000040(0,0000)$
14	$0,00000007(0,0000)$	$0,00000700(0,0000)$	$0,00000047(0,0000)$
15	$0,00000006(0,0000)$	$0,00000701(0,0000)$	$0,00000054(0,0000)$
16	$0,00000011(0,0000)$	$0,00000837(0,0000)$	$0,00000035(0,0000)$
17	$0,00000007(0,0000)$	$0,00000859(0,0000)$	$0,00000045(0,0000)$
18	$0,00000004(0,0000)$	$0,00000926(0,0000)$	$0,00000045(0,0000)$
19	$0,00000007(0,0000)$	$0,00000712(0,0000)$	$0,00000045(0,0000)$
20	$0,00000012(0,0000)$	$0,00000691(0,0000)$	$0,00000047(0,0000)$
21	$0,00000006(0,0000)$	$0,00001002(0,0000)$	$0,00000040(0,0000)$
22	$0,00000006(0,0000)$	$0,00000909(0,0000)$	$0,00000045(0,0000)$
23	$0,00000015(0,0000)$	$0,00001367(0,0000)$	$0,00000043(0,0000)$
24	$0,00000003(0,0000)$	$0,00000550(0,0000)$	$0,00000039(0,0000)$
25	$0,00000010(0,0000)$	$0,00000678(0,0000)$	$0,00000048(0,0000)$
26	$0,00000006(0,0000)$	$0,00000828(0,0000)$	$0,00000039(0,0000)$
27	$0,00000009(0,0000)$	$0,00000959(0,0000)$	$0,00000039(0,0000)$
28	$0,00000004(0,0000)$	$0,00000667(0,0000)$	$0,00000045(0,0000)$
29	$0,00000008(0,0000)$	$0,00000965(0,0000)$	$0,00000040(0,0000)$
30	$0,00000009(0,0000)$	$0,00000747(0,0000)$	$0,00000052(0,0000)$
Média	$0,00000007(0,0000)$	$0,00000831(0,0000)$	$0,00000045(0,0000)$
Desv. Pad.	$0,00000003(0,0000)$	$0,00000220(0,0000)$	$0,00000007(0,0000)$

Tabela C.5 MSO

	RCDESIGN	*Busca RS*	*Busca TR*
1	0,00000000(0,0000)	0,00013337(0,0002)	0,00000081(0,0000)
2	0,00000000(0,0000)	0,00020255(0,0004)	0,00000054(0,0000)
3	0,00000000(0,0000)	0,00026781(0,0005)	0,00000035(0,0000)
4	0,00000000(0,0000)	0,00017381(0,0003)	0,00000082(0,0000)
5	0,00000000(0,0000)	0,00014984(0,0003)	0,00000029(0,0000)
6	0,00000000(0,0000)	0,00030083(0,0005)	0,00000140(0,0000)
7	0,00000000(0,0000)	0,00022611(0,0005)	0,00000037(0,0000)
8	0,00000000(0,0000)	0,00016161(0,0003)	0,00000032(0,0000)
9	0,00000000(0,0000)	0,00018167(0,0004)	0,00000059(0,0000)
10	0,00000000(0,0000)	0,00010679(0,0002)	0,00000036(0,0000)
11	0,00000000(0,0000)	0,00010289(0,0002)	0,00000038(0,0000)
12	0,00000000(0,0000)	0,00015079(0,0002)	0,00000033(0,0000)
13	0,00000000(0,0000)	0,00013091(0,0003)	0,00000042(0,0000)
14	0,00000000(0,0000)	0,00023950(0,0004)	0,00000082(0,0000)
15	0,00000000(0,0000)	0,00022155(0,0004)	0,00000036(0,0000)
16	0,00000000(0,0000)	0,00018183(0,0003)	0,00000030(0,0000)
17	0,00000000(0,0000)	0,00020944(0,0004)	0,00000040(0,0000)
18	0,00000000(0,0000)	0,00026862(0,0005)	0,00000041(0,0000)
19	0,00000000(0,0000)	0,00021086(0,0004)	0,00000024(0,0000)
20	0,00000000(0,0000)	0,00015494(0,0003)	0,00000055(0,0000)
21	0,00000000(0,0000)	0,00016650(0,0003)	0,00000041(0,0000)
22	0,00000000(0,0000)	0,00024024(0,0004)	0,00000043(0,0000)
23	0,00000000(0,0000)	0,00023850(0,0004)	0,00000050(0,0000)
24	0,00000000(0,0000)	0,00013584(0,0003)	0,00000038(0,0000)
25	0,00000000(0,0000)	0,00010659(0,0002)	0,00000031(0,0000)
26	0,00000000(0,0000)	0,00034276(0,0006)	0,00000031(0,0000)
27	0,00000000(0,0000)	0,00019123(0,0003)	0,00000104(0,0000)
28	0,00000000(0,0000)	0,00020529(0,0004)	0,00000026(0,0000)
29	0,00000000(0,0000)	0,00017242(0,0003)	0,00000046(0,0000)
30	0,00000000(0,0000)	0,00011520(0,0002)	0,00000105(0,0000)
Média	0,00000000(0,0000)	0,00018968(0,0003)	0,00000051(0,0000)
Desv. Pad.	0,00000000(0,0000)	0,00005926(0,0001)	0,00000028(0,0000)

Tabela C.6 *STAR*

	RCDESIGN	*Busca RS*	*Busca TR*
1	0,00012082(0,0002)	0,00036018(0,0004)	0,00023309(0,0003)
2	0,00013097(0,0002)	0,00034838(0,0004)	0,00020114(0,0003)
3	0,00012765(0,0001)	0,00036570(0,0004)	0,00022491(0,0003)
4	0,00014405(0,0002)	0,00035119(0,0004)	0,00020607(0,0003)
5	0,00016029(0,0002)	0,00040284(0,0004)	0,00023546(0,0003)
6	0,00014596(0,0002)	0,00040264(0,0005)	0,00022405(0,0003)
7	0,00015801(0,0002)	0,00030379(0,0003)	0,00023080(0,0004)
8	0,00017910(0,0002)	0,00042672(0,0005)	0,00020847(0,0003)
9	0,00020829(0,0003)	0,00044409(0,0005)	0,00020418(0,0003)
10	0,00030904(0,0003)	0,00040200(0,0004)	0,00021559(0,0003)
11	0,00013470(0,0002)	0,00036446(0,0004)	0,00021108(0,0003)
12	0,00013890(0,0002)	0,00038125(0,0005)	0,00022150(0,0002)
13	0,00018449(0,0002)	0,00046463(0,0005)	0,00019636(0,0002)
14	0,00017425(0,0002)	0,00038281(0,0005)	0,00020868(0,0003)
15	0,00050117(0,0005)	0,00041172(0,0005)	0,00025403(0,0004)
16	0,00012301(0,0001)	0,00052267(0,0006)	0,00020440(0,0003)
17	0,00014520(0,0002)	0,00040751(0,0005)	0,00026090(0,0005)
18	0,00013111(0,0002)	0,00031226(0,0004)	0,00021422(0,0003)
19	0,00013623(0,0002)	0,00041032(0,0004)	0,00023445(0,0003)
20	0,10871607(0,0531)	0,00034257(0,0004)	0,00020079(0,0003)
21	0,00012672(0,0002)	0,00044414(0,0005)	0,00021422(0,0003)
22	0,00013051(0,0002)	0,00037769(0,0004)	0,00021532(0,0003)
23	0,00038018(0,0005)	0,00038597(0,0004)	0,00022383(0,0003)
24	0,00032123(0,0005)	0,00035377(0,0004)	0,00019122(0,0003)
25	0,00012275(0,0002)	0,00033806(0,0004)	0,00022922(0,0003)
26	0,00013783(0,0002)	0,00039264(0,0004)	0,00021264(0,0003)
27	0,00012480(0,0001)	0,00037071(0,0005)	0,00019879(0,0003)
28	0,00014136(0,0002)	0,00035213(0,0004)	0,00022408(0,0003)
29	0,00012949(0,0002)	0,00029980(0,0004)	0,00023580(0,0003)
30	0,00029131(0,0003)	0,00040465(0,0006)	0,00022218(0,0003)
Média	0,00379918(0,0020)	0,00038424(0,0004)	0,00021858(0,0003)
Desv. Pad.	0,01981584(0,0097)	0,00004792(0,0001)	0,00001626(0,0001)

Tabela C.7 *Down-Jones*

	RCDESIGN	*Busca RS*	*Busca TR*
1	0,00071224(0,0011)	0,00123594(0,0024)	0,00070053(0,0011)
2	0,00068814(0,0010)	0,00115188(0,0019)	0,00068529(0,0010)
3	0,00071337(0,0012)	0,00113361(0,0018)	0,00068902(0,0011)
4	0,00093046(0,0014)	0,00124061(0,0021)	0,00068917(0,0010)
5	0,00071861(0,0011)	0,00117016(0,0019)	0,00067535(0,0010)
6	0,00069437(0,0011)	0,00118062(0,0021)	0,00070952(0,0011)
7	0,00069038(0,0011)	0,00113552(0,0019)	0,00068169(0,0010)
8	0,00069011(0,0011)	0,00119807(0,0020)	0,00068298(0,0010)
9	0,00068860(0,0011)	0,00114236(0,0016)	0,00069172(0,0010)
10	0,00068946(0,0011)	0,00123247(0,0021)	0,00069092(0,0011)
11	0,00068440(0,0010)	0,00117706(0,0020)	0,00071211(0,0011)
12	0,00079232(0,0011)	0,00119378(0,0020)	0,00068257(0,0010)
13	0,00071277(0,0011)	0,00113571(0,0016)	0,00068647(0,0010)
14	0,00070318(0,0011)	0,00120505(0,0021)	0,00068969(0,0011)
15	0,00069957(0,0010)	0,00115106(0,0019)	0,00069642(0,0011)
16	0,00072000(0,0011)	0,00117483(0,0019)	0,00068019(0,0010)
17	0,00068788(0,0011)	0,00115074(0,0019)	0,00066709(0,0010)
18	0,00068761(0,0011)	0,00120613(0,0022)	0,00072203(0,0011)
19	0,00069737(0,0011)	0,00114682(0,0018)	0,00068960(0,0010)
20	0,00072009(0,0012)	0,00113967(0,0018)	0,00069711(0,0011)
21	0,00069023(0,0011)	0,00118368(0,0023)	0,00069012(0,0010)
22	0,00068818(0,0011)	0,00113727(0,0018)	0,00068529(0,0011)
23	0,00072283(0,0011)	0,00121684(0,0020)	0,00069329(0,0010)
24	0,00070841(0,0011)	0,00118182(0,0021)	0,00067702(0,0010)
25	0,00068582(0,0011)	0,00124519(0,0025)	0,00070837(0,0011)
26	0,00068938(0,0011)	0,00117403(0,0019)	0,00071136(0,0011)
27	0,00071477(0,0011)	0,00118568(0,0019)	0,00067464(0,0010)
28	0,00123019(0,0014)	0,00113245(0,0017)	0,00068306(0,0010)
29	0,00067427(0,0011)	0,00120479(0,0021)	0,00068281(0,0010)
30	0,00068416(0,0011)	0,00120185(0,0019)	0,00067991(0,0010)
Média	0,00072697(0,0011)	0,00117886(0,0020)	0,00069018(0,0011)
Desv. Pad.	0,00010602(0,0001)	0,00003477(0,0002)	0,00001257(0,0000)

Tabela C.8 Belo Jardim

	RCDESIGN	*Busca RS*	*Busca TR*
1	0,00396170(0,0076)	0,01138219(0,0171)	0,00446990(0,0082)
2	0,00392963(0,0075)	0,01141648(0,0167)	0,00457121(0,0083)
3	0,00404432(0,0075)	0,01135179(0,0167)	0,00454072(0,0084)
4	0,00398440(0,0074)	0,01141752(0,0170)	0,00446536(0,0082)
5	0,00389121(0,0074)	0,01144610(0,0168)	0,00450660(0,0082)
6	0,00396594(0,0074)	0,01142879(0,0170)	0,00427910(0,0076)
7	0,00395984(0,0076)	0,01130677(0,0166)	0,00441606(0,0081)
8	0,00400584(0,0076)	0,01138494(0,0169)	0,00448348(0,0082)
9	0,00403750(0,0074)	0,01140729(0,0168)	0,00447268(0,0081)
10	0,00395477(0,0073)	0,01149137(0,0168)	0,00446144(0,0081)
11	0,00398760(0,0073)	0,01117401(0,0164)	0,00451352(0,0081)
12	0,00393084(0,0077)	0,01137999(0,0167)	0,00440024(0,0080)
13	0,00394043(0,0073)	0,01130594(0,0166)	0,00448011(0,0081)
14	0,00401335(0,0077)	0,01140433(0,0169)	0,00452445(0,0084)
15	0,00399251(0,0075)	0,01129096(0,0165)	0,00439074(0,0083)
16	0,00401266(0,0076)	0,01131851(0,0166)	0,00443851(0,0080)
17	0,00393003(0,0074)	0,01153772(0,0170)	0,00451025(0,0083)
18	0,00399615(0,0077)	0,01111884(0,0163)	0,00455684(0,0083)
19	0,00395408(0,0072)	0,01132665(0,0168)	0,00445938(0,0084)
20	0,00396411(0,0074)	0,01140780(0,0168)	0,00443277(0,0082)
21	0,00394873(0,0076)	0,01139455(0,0167)	0,00447657(0,0081)
22	0,00398595(0,0077)	0,01158971(0,0175)	0,00440865(0,0081)
23	0,00396240(0,0075)	0,01142260(0,0169)	0,00446618(0,0079)
24	0,00393681(0,0075)	0,01147799(0,0169)	0,00441295(0,0081)
25	0,00385956(0,0074)	0,01129511(0,0168)	0,00436971(0,0080)
26	0,00394475(0,0076)	0,01142211(0,0168)	0,00455001(0,0084)
27	0,00394558(0,0073)	0,01145638(0,0169)	0,00444232(0,0080)
28	0,00399993(0,0076)	0,01139961(0,0167)	0,00431818(0,0081)
29	0,00399264(0,0075)	0,01148460(0,0166)	0,00438340(0,0079)
30	0,00398602(0,0075)	0,01113371(0,0162)	0,00445310(0,0083)
Média	0,00396731(0,0075)	0,01137915(0,0168)	0,00445515(0,0081)
Desv. Pad.	0,00003986(0,0001)	0,00010637(0,0003)	0,00006759(0,0002)

Tabela C.9 São João do Cariri

	RCDESIGN	*Busca RS*	*Busca TR*
1	0,00319314(0,0056)	0,01145162(0,0171)	0,00399405(0,0073)
2	0,00320559(0,0056)	0,01173924(0,0176)	0,00385264(0,0066)
3	0,00322261(0,0056)	0,01157055(0,0175)	0,00391532(0,0071)
4	0,00308665(0,0054)	0,01148110(0,0173)	0,00382922(0,0067)
5	0,00314518(0,0056)	0,01154328(0,0176)	0,00399202(0,0076)
6	0,00314389(0,0057)	0,01166858(0,0173)	0,00391380(0,0069)
7	0,00316913(0,0056)	0,01151740(0,0172)	0,00390865(0,0074)
8	0,00315976(0,0057)	0,01177839(0,0177)	0,00380753(0,0071)
9	0,00315590(0,0055)	0,01132213(0,0170)	0,00403340(0,0073)
10	0,00310272(0,0055)	0,01180828(0,0180)	0,00391758(0,0072)
11	0,00322626(0,0057)	0,01144253(0,0173)	0,00391210(0,0070)
12	0,00319191(0,0057)	0,01177169(0,0175)	0,00399921(0,0071)
13	0,00302976(0,0053)	0,01163487(0,0175)	0,00406172(0,0072)
14	0,00306086(0,0054)	0,01159470(0,0177)	0,00394618(0,0067)
15	0,00316517(0,0057)	0,01148376(0,0172)	0,00400894(0,0073)
16	0,00321132(0,0056)	0,01146517(0,0171)	0,00404902(0,0072)
17	0,00312306(0,0055)	0,01118907(0,0165)	0,00403268(0,0073)
18	0,00314732(0,0056)	0,01148481(0,0171)	0,00388745(0,0071)
19	0,00319468(0,0055)	0,01164929(0,0175)	0,00387641(0,0072)
20	0,00317546(0,0058)	0,01172420(0,0177)	0,00398039(0,0071)
21	0,00314520(0,0055)	0,01125555(0,0170)	0,00399447(0,0072)
22	0,00316286(0,0055)	0,01164520(0,0173)	0,00370111(0,0069)
23	0,00313165(0,0056)	0,01151998(0,0172)	0,00391094(0,0073)
24	0,00320228(0,0057)	0,01158506(0,0173)	0,00397802(0,0072)
25	0,00312762(0,0055)	0,01148667(0,0172)	0,00412546(0,0075)
26	0,00312079(0,0056)	0,01156054(0,0173)	0,00399045(0,0073)
27	0,00312957(0,0056)	0,01144922(0,0170)	0,00372091(0,0070)
28	0,00317236(0,0056)	0,01135420(0,0169)	0,00372403(0,0067)
29	0,00320057(0,0056)	0,01146415(0,0171)	0,00406296(0,0073)
30	0,00317151(0,0056)	0,01171466(0,0176)	0,00406417(0,0072)
Média	0,00315583(0,0056)	0,01154520(0,0173)	0,00393969(0,0071)
Desv. Pad.	0,00004620(0,0001)	0,00015281(0,0003)	0,00010697(0,0002)

Tabela C.10 Triunfo

	RCDESIGN	*Busca RS*	*Busca TR*
1	0,00100315(0,0021)	0,00360368(0,0061)	0,00107313(0,0022)
2	0,00099158(0,0021)	0,00342192(0,0055)	0,00112629(0,0023)
3	0,00099588(0,0021)	0,00364885(0,0060)	0,00108482(0,0022)
4	0,00099256(0,0021)	0,00377972(0,0063)	0,00105826(0,0022)
5	0,00099278(0,0020)	0,00348244(0,0057)	0,00103545(0,0021)
6	0,00098609(0,0020)	0,00366424(0,0061)	0,00107876(0,0022)
7	0,00098872(0,0020)	0,00359355(0,0059)	0,00106192(0,0022)
8	0,00099816(0,0021)	0,00363174(0,0060)	0,00103918(0,0021)
9	0,00099504(0,0020)	0,00352951(0,0058)	0,00102371(0,0021)
10	0,00100162(0,0021)	0,00357639(0,0059)	0,00103763(0,0021)
11	0,00098783(0,0020)	0,00350084(0,0056)	0,00108025(0,0022)
12	0,00097189(0,0020)	0,00352029(0,0058)	0,00106933(0,0022)
13	0,00099516(0,0020)	0,00355357(0,0058)	0,00107239(0,0022)
14	0,00098469(0,0020)	0,00344017(0,0056)	0,00109154(0,0022)
15	0,00099118(0,0021)	0,00352349(0,0057)	0,00108543(0,0023)
16	0,00100023(0,0021)	0,00360868(0,0060)	0,00103546(0,0021)
17	0,00099228(0,0020)	0,00343745(0,0055)	0,00105035(0,0021)
18	0,00099417(0,0021)	0,00360612(0,0060)	0,00107909(0,0022)
19	0,00099488(0,0021)	0,00375174(0,0064)	0,00106389(0,0022)
20	0,00097896(0,0020)	0,00365629(0,0061)	0,00104468(0,0021)
21	0,00098855(0,0020)	0,00356552(0,0059)	0,00105879(0,0021)
22	0,00100426(0,0021)	0,00371448(0,0062)	0,00105787(0,0022)
23	0,00099328(0,0020)	0,00345860(0,0056)	0,00104627(0,0021)
24	0,00098404(0,0020)	0,00364782(0,0061)	0,00104065(0,0021)
25	0,00098940(0,0020)	0,00358904(0,0060)	0,00110124(0,0022)
26	0,00100336(0,0021)	0,00357683(0,0059)	0,00108283(0,0022)
27	0,00098407(0,0020)	0,00362958(0,0059)	0,00106428(0,0022)
28	0,00098956(0,0020)	0,00356920(0,0058)	0,00108089(0,0022)
29	0,00099552(0,0021)	0,00351084(0,0057)	0,00108142(0,0022)
30	0,00099068(0,0020)	0,00354349(0,0058)	0,00107443(0,0022)
Média	0,00099199(0,0000)	0,00357787(0,0000)	0,00106601(0,0000)
Desv. Pad.	0,00000721(0,0000)	0,00008885(0,0000)	0,00002261(0,0000)