COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

IDENTIFICAÇÃO DE MODELOS BASEADOS EM SISTEMAS FUZZY RECORRENTES COM DUPLO FEEDBACK USANDO EVOLUÇÃO DIFERENCIAL

Cristian Klen dos Santos

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Civil, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Civil.

Orientador: Alexandre Gonçalves Evsukoff

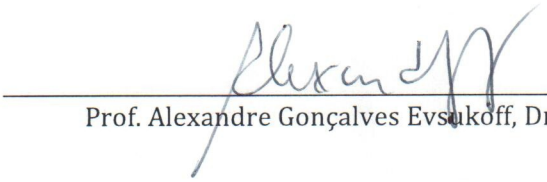
Rio de Janeiro
Julho de 2013

IDENTIFICAÇÃO DE MODELOS BASEADOS EM SISTEMAS FUZZY RECORRENTES COM DUPLO FEEDBACK USANDO EVOLUÇÃO DIFERENCIAL

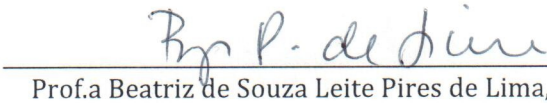
Cristian Klen dos Santos

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA CIVIL.

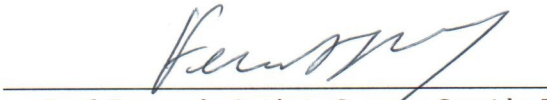
Examinada por:



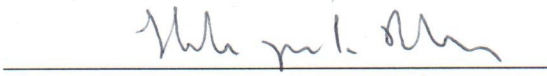
Prof. Alexandre Gonçalves Evsukoff, Dr.



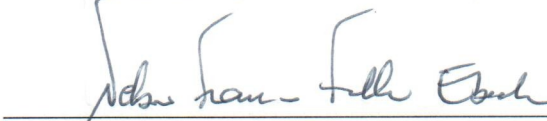
Prof.a Beatriz de Souza Leite Pires de Lima, D.Sc.



Prof. Fernando Antônio Campos Gomide, Ph.D.



Prof. Helio José Correa Barbosa, D.Sc.



Prof. Nelson Francisco Favilla Ebecken, D.Sc.

RIO DE JANEIRO, RJ, BRASIL

JULHO DE 2013

Santos, Cristian Klen dos

Identificação de Modelos Baseados em Sistemas Fuzzy Recorrentes com Duplo Feedback Usando Evolução Diferencial / Cristian Klen dos Santos. – Rio de Janeiro: UFRJ/COPPE, 2013.

XX, 150 p.: il.; 29,7 cm.

Orientador: Alexandre Gonçalves Evsukoff

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Civil, 2013.

Referências Bibliográficas: p. 105–119.

1. Identificação de Sistemas. 2. Sistemas Fuzzy Recorrentes. 3. Evolução Diferencial. I. Evsukoff, Alexandre Gonçalves. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Civil. III. Título.

Dedico este trabalho às mulheres que mais amo.
As quais são motivo de orgulho e regozijo em minha vida.

À minha mãe,
por cada uma das inúmeras dificuldades que superou para que eu pudesse chegar até aqui, demonstrando de maneira inquestionável seu inefável amor e resignação.

À minha avó, Iracy,
por ter sido um baluarte em minha vida
e participado ativamente na formação do meu caráter.

À minha irmã,
por sua amizade singular,
pelo companheirismo e compreensão ao longo da minha vida.

E à minha namorada, Alessandra,
por sua dedicação ao nosso relacionamento,
pelo cuidado diário e por seu imenso amor por mim.

Agradecimentos

Gostaria de agradecer a todas as pessoas que me proporcionaram a oportunidade de vivenciar cada uma das muitas experiências desafiadoras que me incitaram a galgar degraus sempre mais elevados. Gostaria, também, de externar a minha gratidão àquelas pessoas que, direta ou indiretamente, aguçaram minha curiosidade e minha paixão pela pesquisa científica. Outrossim, gostaria de agradecer:

Ao corpo docente, discente e aos funcionários do PEC, pelo convívio agradável no decorrer dos seis últimos anos, durante os quais concluí o mestrado e o doutorado, e à CAPES, pelo fomento a este trabalho de pesquisa.

Aos amigos da CEDAE, por terem torcido pelo meu sucesso. Aos amigos do NTT, pelo companheirismo, motivação e experiências compartilhadas. Aos amigos do IPqM, pelo incentivo e camaradagem, especialmente aos da DSI, pela compreensão e paciência, e por terem me apoiado irrestritamente nesta conturbada jornada.

Aos membros da banca examinadora desta tese, por todas as valiosas contribuições agregadas a este trabalho e, principalmente, ao meu orientador, pela amizade, confiança, inspiração, paciência e sábios conselhos ao longo desta trajetória.

À minha família, pelo apoio. À minha avó, pelo carinho e dedicação. À minha irmã, pela torcida, carinho e apoio em todos os momentos. Aos meus pais, por todo o apoio, pelo amor incondicional, dedicação e constante intercessão junto ao criador. À minha linda e adorável namorada, por seu grande amor e paciência, e por sua compreensão na privação da minha companhia durante o doutorado.

Acima de tudo, porém, quero agradecer a Deus, pelo fôlego de vida, por seu amor e bondade, e pela graça da realização de mais este sonho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

IDENTIFICAÇÃO DE MODELOS BASEADOS EM SISTEMAS FUZZY RECORRENTES COM DUPLO FEEDBACK USANDO EVOLUÇÃO DIFERENCIAL

Cristian Klen dos Santos

Julho/2013

Orientador: Alexandre Gonçalves Evsukoff

Programa: Engenharia Civil

Os sistemas fuzzy recorrentes são caracterizados por possuírem realimentação (feedback) e, por isso, são utilizados comumente para modelar sistemas dinâmicos complexos. Este trabalho apresenta o desenvolvimento de modelos baseados em sistemas fuzzy recorrentes que possuem uma conexão de feedback adicional e operadores de defasagem ajustáveis, permitindo que o modelo seja configurado para simulação e predição de sistemas dinâmicos de ordem desconhecida. Adicionalmente, uma abordagem de identificação simultânea da estrutura e dos parâmetros, baseada no algoritmo de evolução diferencial, foi desenvolvida. As abordagens propostas foram avaliadas em alguns problemas de benchmark. Os resultados obtidos mostraram o melhor desempenho da evolução diferencial sobre os algoritmos genéticos e a flexibilidade da evolução diferencial para identificar tanto os parâmetros quanto a estrutura de cada um dos modelos avaliados. Além disso, os resultados evidenciaram o potencial das novas estruturas, que proporcionaram um bom desempenho ao modelo. Embora a abordagem proposta seja caracterizada por uma formulação mais simples, ela apresentou desempenho equivalente quando comparada a alguns modelos do estado da arte nos problemas estudados.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

IDENTIFICATION OF MODELS BASED ON RECURRENT FUZZY SYSTEMS WITH DUAL FEEDBACK USING DIFFERENTIAL EVOLUTION

Cristian Klen dos Santos

July/2013

Advisor: Alexandre Gonçalves Evsukoff

Department: Civil Engineering

Recurrent fuzzy systems are characterized by feedback connections. Therefore, they are commonly used for modeling nonlinear dynamical systems. This work presents an approach to develop models based on recurrent fuzzy systems with an additional feedback connection and adjustable delay operators, allowing the model to be configured for both simulation and prediction of dynamic systems of unknown order. Additionally, an approach for unified structure and parameter identification based on differential evolution was also developed. The proposed approaches were evaluated in some benchmark problems. The obtained results show the best performance of differential evolution compared to genetic algorithms and the flexibility of differential evolution to identify both the parameters and the structure of each one of the considered models. Besides, the results demonstrated the potential of the new structures developed, yielding models with good performance. Although the proposed approach is characterized by a simpler formulation, it shows equivalent performance when compared with some state-of-the-art models considering the benchmark problems.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

SÍMBOLO	SIGNIFICADO
$\mathcal{D}$	Conjunto de dados
$\mathcal{R}$	Conjunto dos números reais
$\mathcal{V}$	Conjunto de soluções (população)
A	Símbolo linguístico associado aos conjuntos fuzzy dos regressores
B	Símbolo linguístico associado aos conjuntos fuzzy dos estados
C	Símbolo linguístico associado aos conjuntos fuzzy das entradas
CR	Constante de recombinação
D	Símbolo linguístico associado aos conjuntos fuzzy do feedback adicional
F	Fator de escala
J	Função de custo
K	Quantidade de elementos herdados do vetor mutante
M	Quantidade de modelos locais / regras
MAX_GER	Quantidade máxima de gerações
N	Quantidade de observações
NP	Tamanho da população
P	Parâmetro da equação de Hénon
Q	Parâmetro da equação de Hénon
R	Regra fuzzy
X	Função do sistema de equações de Lorenz que representa a dimensão X
Y	Função do sistema de equações de Lorenz que representa a dimensão Y
Z	Função do sistema de equações de Lorenz que representa a dimensão Z
a	Parâmetro da função X do sistema de equações de Lorenz
b	Parâmetro da função Z do sistema de equações de Lorenz
c	Parâmetro da função Y do sistema de equações de Lorenz
d	Quantidade de dimensões do vetor
e	Variável de erro
f	Função que representa o modelo do sistema

SÍMBOLO	SIGNIFICADO
g	Função de transição de estados
h	Função de saída
i	Índice sequencial
j	Índice sequencial
k	Índice sequencial
l	Vetor de restrição lateral
n	Quantidade de atrasos
p	Quantidade de protótipos
q	Operador de defasagem
r	Índice aleatório
rnd	Função geradora de números aleatórios
$rndn$	Função geradora de índices aleatórios
s	Vetor experimental
t	Instante de tempo
u	Variável de entrada
v	Vetor de projeto
w	Vetor mutante
x	Variável de estado
y	Variável de saída do sistema
$\hat{y}$	Variável de saída do modelo
z	Regressor
α	Coeficiente de ponderação do modelo local
β	Vetor base
γ	Parâmetro de posição da função de base
δ	Vetor diferencial
η	Vetor que representa a estrutura do modelo
θ	Vetor de parâmetros do modelo
λ	Vetor de parâmetros da combinação linear
μ	Função de pertinência
ξ	Variável de feedback adicional
ζ	Quantidade máxima de parâmetros permitida
ρ	Parâmetro de escala da função de base
σ	Desvio padrão
ϕ	Função de base
ψ	Função objetivo
ω	Valor de ativação da regra
Δ	Parâmetro de defasagem

Capítulo 1

1 Introdução

Desde a antiguidade procura-se compreender e representar problemas reais através de modelos. A modelagem de sistemas complexos e não bem compreendidos é sempre uma tarefa desafiadora. A criação de modelos derivados a partir dos primeiros princípios para tais sistemas é extremamente difícil ou mesmo impossível. Uma alternativa geralmente considerada neste cenário são os modelos derivados a partir de dados. Nesta abordagem, conhecida como identificação de sistemas ou modelagem empírica, apenas os dados experimentais são utilizados para gerar um modelo que represente o comportamento do sistema.

O desenvolvimento de modelos baseados em dados para sistemas dinâmicos não lineares tem despertado grande interesse e mantido ativa esta linha de pesquisa ao longo das últimas décadas (HONG et al., 2008; JUDITSKY et al., 1995; LJUNG, 1999; NELLES, 2001; SJÖBERG et al., 1995; TONG, 1980). Nos sistemas dinâmicos, a saída atual do sistema é função do seu comportamento em instantes passados, de modo que dele é dito possuir memória. Neste caso, o objetivo é predizer a saída do sistema em instantes futuros em função de um conjunto de observações do seu comportamento passado. Problemas relacionados a sistemas dinâmicos são encontrados em muitas áreas como, por exemplo, automação e controle (CHEN, T.-C.; REN; LOU, 2013), biotecnologia (BARUCH; HERNANDEZ, 2011), energia (HELL; COSTA; GOMIDE, 2007), ecologia (MON; LIN; YEH, 2013), hidrologia (EVSUKOFF; CATALDI; DE

LIMA, 2012), telecomunicações (MASTOROCOSTAS, P.; HILAS, 2012), só para citar algumas.

Como os sistemas dinâmicos são caracterizados pela dependência temporal do seu comportamento, os valores anteriores da entrada e da saída do sistema são geralmente utilizados para representar a sua memória. Muitas vezes, porém, uma grande quantidade de valores é necessária para representar adequadamente a dinâmica do sistema, causando o problema conhecido na literatura como "maldição da dimensionalidade" (BELLMAN, 1961) *apud* (NELLES, 2001). Para contornar este problema, estruturas recorrentes são frequentemente adotadas. Nestas estruturas, a memória do sistema é representada internamente, por meio dos seus estados e, portanto, proporcionam uma representação mais compacta ao modelo.

A crescente complexidade dos novos problemas que surgem à medida que a ciência e a tecnologia evoluem exige o desenvolvimento de modelos que precisam atender a critérios cada vez mais rígidos de desempenho. Neste caso, nem sempre as técnicas tradicionais conseguem produzir soluções com o nível desejado de desempenho, ou sequer serem utilizadas, quer seja pelo grau de complexidade do problema ou qualquer outra restrição encontrada. Neste contexto, técnicas baseadas em inteligência computacional surgem como alternativa às abordagens tradicionais.

Essas técnicas são inspiradas em aspectos do comportamento humano como, por exemplo, aprendizado, percepção, raciocínio, adaptação e evolução. Dentro desta classe de técnicas inteligentes encontram-se os sistemas fuzzy (KOSKO, B., 1994; MAMDANI, E.H., 1977; PEDRYCZ; GOMIDE, 2007; SUGENO; KANG, 1988; TAKAGI; SUGENO, 1985) e os algoritmos evolucionários (MA et al., em fase de elaboração; YU; GEN, 2010), que muitas das vezes são utilizados em conjunto para construir modelos híbridos (CORDÓN et al., 2004; HERRERA, 2008), que visam explorar as potencialidades de ambas as técnicas ao mesmo tempo em que tentam minimizar as limitações de cada uma delas.

Modelos baseados em sistemas fuzzy têm sido extensivamente utilizados como ferramenta para identificação de sistemas (ABONYI, 2003; HELLENDOORN; DRIANKOV, 1997; LILLY, 2011). O desenvolvimento desses modelos com estruturas recorrentes têm atraído grande interesse e se mostrado uma abordagem promissora para

problemas envolvendo sistemas dinâmicos não lineares, como será visto em detalhes nas seções 2.1.2 e 2.2.1.

A abordagem mais comum para a identificação dos parâmetros de sistemas fuzzy recorrentes é a utilização de métodos baseados no gradiente, como os algoritmos *BPTT* (Backpropagation-Through-Time) ou *RTRL* (Real Time-Recurrent-Learning) (NELLES, 2001). Porém, um dos principais problemas destes métodos é a possibilidade de facilmente ficarem estagnados em mínimos locais. Neste cenário, os algoritmos evolucionários aparecem como uma excelente alternativa (ALIEV, R. A. et al., 2009; EVSUKOFF; EBECKEN, 2004; FAZZOLARI et al., 2013; HERRERA, 2008; JUANG; CHANG, 2011; LEE, C.-H.; LEE, 2012; LIN, C.-J.; WU; LEE, 2011; LIN, H.-Y. et al., 2012; STAVRAKOUDIS; PAPASTAMOULIS; THEOCHARIS, 2008), pois são métodos de busca estocásticos e, portanto, menos prováveis de estagnarem em mínimos locais. Além disso, por serem métodos populacionais, proporcionam uma busca paralela por melhores soluções.

Outra vantagem da abordagem evolucionária sobre aquelas baseadas no gradiente, para a otimização não linear dos parâmetros de sistemas fuzzy recorrentes, é a flexibilidade de poder trabalhar com diversos tipos de estruturas diferentes sem a necessidade de derivar cada uma delas, visto que os algoritmos evolucionários não se baseiam na informação da derivada de funções evitando, assim, a manipulação de matrizes Hessianas/Jacobianas.

Se por um lado os algoritmos evolucionários apresentam algumas vantagens sobre os métodos baseados no gradiente, eles também possuem algumas limitações. A principal delas é devido a sua natureza heurística, que não pode garantir que a solução ótima seja encontrada em um número finito de iterações. Além disso, as abordagens evolucionárias demandam um alto custo computacional, devido principalmente à sua natureza populacional, e são dependentes de alguns parâmetros de controle, que interferem diretamente sobre o seu desempenho.

Um algoritmo evolucionário particularmente promissor para espaços de busca contínuos é a evolução diferencial (STORN, RAINER; PRICE, 1997), que tem recebido crescente interesse por sua capacidade de encontrar o mínimo global em diferentes tipos de problemas de otimização (DA SILVA; BARBOSA; LEMONGE, 2011; DAS;

SUGANTHAN, 2011; FEOKTISTOV, 2006; NERI; TIRRONEN, 2010; PLAGIANAKOS; TASOULIS; VRAHATIS, 2008; PRICE, K.; STORN; LAMPINEN, 2005). Estudos recentes no campo de identificação de parâmetros mostram que a evolução diferencial supera alguns dos algoritmos utilizados frequentemente para este fim como, por exemplo, os algoritmos genéticos (LIN, F. T., 2010; OH; KIM; PEDRYCZ, 2012). Adicionalmente, a evolução diferencial tem se destacado pela simplicidade de implementação, e por possuir poucos parâmetros de controle (DAS; SUGANTHAN, 2011; NERI; TIRRONEN, 2010). Todas essas qualidades têm motivado a utilização recente da evolução diferencial na otimização de sistemas fuzzy (LU; CHANG; TSAI, 2012; OH et al., 2012; SU, H.; YANG, 2011), e por isso, ela é utilizada neste trabalho.

1.1 Objetivos e Contribuições

Este trabalho apresenta a pesquisa que foi realizada com o objetivo de desenvolver novos modelos baseados no sistema fuzzy recorrente que foi apresentado em (GAMA et al., 2008), visando a melhoraria de desempenho. Este desenvolvimento teve como resultado a exploração de novas estruturas; a utilização de um novo algoritmo evolucionário; bem como a criação de uma nova abordagem de identificação.

O sistema fuzzy recorrente apresentado em (GAMA et al., 2008) é utilizado para modelar a equação de transição de estados em um modelo formulado pelas equações do espaço de estados, em que a equação de saída é uma combinação linear dos estados, que são calculados pelo sistema fuzzy. A estrutura do modelo é selecionada por meio de uma busca exaustiva, e seus parâmetros são identificados por um algoritmo genético canônico. A recorrência deste sistema fuzzy é definida através da relação de dependência temporal entre os estados.

Na extensão proposta, a criação das novas estruturas teve como base a utilização de três regressores que não foram considerados originalmente. As novas estruturas são caracterizadas pela utilização de mais uma conexão de feedback (além do estado), que pode vir do próprio modelo (saída prevista); do sistema dinâmico (saída observada); ou mesmo de ambos (erro de predição). Adicionalmente, dois

operadores de defasagem ajustáveis são utilizados a fim de melhorar o comportamento do modelo quando este for utilizado para processar sistemas dinâmicos com atraso. As novas estruturas permitem que o modelo continue operando em modo simulação, caso a saída prevista seja utilizada como feedback adicional, ou passe a operar em modo predição, caso o feedback adicional seja representado pela saída observada ou pelo erro de predição.

Diante dos recentes resultados em que o desempenho da evolução diferencial superou o desempenho dos algoritmos genéticos em alguns problemas (LIN, F. T., 2010; OH et al., 2012; SU, M.-T. et al., 2011), um dos objetivos específicos do trabalho foi analisar o desempenho do modelo desenvolvido por Gama et al. (2008) utilizando a evolução diferencial. Isto foi feito a fim de verificar qual das duas abordagens evolucionárias proporcionaria melhor desempenho ao modelo: a evolução diferencial ou os algoritmos genéticos.

Adicionalmente, uma nova abordagem é desenvolvida com o objetivo de permitir a identificação simultânea da estrutura e dos parâmetros do modelo. Esta nova abordagem está baseada na utilização hierárquica de duas instâncias do algoritmo de evolução diferencial. A instância do nível externo realiza a identificação da estrutura enquanto a do nível interno realiza a identificação dos parâmetros. Esta nova abordagem é completamente diferente da abordagem considerada originalmente em (GAMA et al., 2008), que utiliza uma busca exaustiva para selecionar a melhor estrutura e identifica os parâmetros do modelo através de um algoritmo genético.

1.2 Estrutura da Tese

O conteúdo da tese está organizado em seis capítulos, os quais são descritos a seguir:

- ✓ **Capítulo 1**

Este é o capítulo introdutório, que apresenta as motivações por trás deste trabalho, bem como os objetivos e as contribuições da tese. A estrutura do trabalho é apresentada nesta seção, que é a última do capítulo.

- ✓ **Capítulo 2**

Este capítulo aborda a revisão da literatura. Inicialmente, são apresentados os conceitos básicos que formam o pano de fundo a partir do qual este trabalho foi desenvolvido, de modo a permitir uma melhor compreensão a respeito dos assuntos que se seguem nos próximos capítulos. Posteriormente, o estado da arte é apresentado, de modo a contextualizar a abordagem proposta neste trabalho e posicioná-la diante dos trabalhos correlatos desenvolvidos recentemente. A primeira parte do capítulo começa discorrendo sobre a identificação de sistemas e prossegue com a sua formulação matemática. Em seguida, os sistemas fuzzy recorrentes são introduzidos e, finalmente, a apresentação dos conceitos básicos sobre a evolução diferencial encerra a primeira parte. A segunda parte também está subdividida em duas seções. Na primeira, as principais aplicações e modelos baseados em sistemas fuzzy recorrentes são apresentados, enquanto na última são apresentados os principais métodos de identificação baseados na evolução diferencial.

- ✓ **Capítulo 3**

Este capítulo está dividido em duas partes, e apresenta a metodologia que foi desenvolvida para alcançar os objetivos descritos na seção 1.1. A primeira parte introduz a formulação matemática dos novos modelos desenvolvidos, enquanto a segunda descreve o método de identificação simultânea da estrutura e dos parâmetros.

- ✓ **Capítulo 4**

Este capítulo apresenta os problemas que foram estudados e as simulações computacionais que foram realizados a fim de validar a abordagem proposta. Quatro problemas foram estudados. Os dois primeiros representam sistemas dinâmicos não lineares e os dois últimos representam séries temporais caóticas. Quanto às simulações computacionais, inicialmente, a evolução diferencial é comparada aos algoritmos genéticos no problema de identificação dos parâmetros. Em seguida, a nova estrutura e o novo método de identificação são avaliados. A última simulação avalia a influência de cada tipo de feedback no desempenho do modelo.

- ✓ **Capítulo 5**

Este capítulo é dedicado à apresentação e à análise dos resultados experimentais, que foram utilizados para verificar a validade das suposições que deram origem a este trabalho. Apenas os resultados consolidados são apresentados neste capítulo por uma questão de clareza. Os resultados detalhados são apresentados nos três apêndices, cada um dos quais está relacionado a uma simulação específica.

- ✓ **Capítulo 6**

Este capítulo apresenta as considerações finais da tese, e sugere as principais direções a serem seguidas na continuidade desta pesquisa.

Capítulo 2

2 Revisão da Literatura

2.1 Conceitos Básicos

2.1.1 Identificação de Sistemas

2.1.1.1 Introdução

A modelagem de sistemas é uma atividade de fundamental importância em, virtualmente, todas as áreas do conhecimento. O uso de modelos matemáticos é inerente a todos os campos da física e engenharia. Na verdade, a engenharia lida basicamente com o problema de como construir bons projetos baseadas em modelos matemáticos. Estes modelos são, em última instância, formalizações matemáticas das abstrações conceituais de sistemas físicos.

Os modelos matemáticos podem ser desenvolvidos ao longo de duas vias (ou pela combinação de ambas). Uma delas se baseia nas leis físicas que descrevem o sistema a ser modelado, sendo conhecida como modelagem pela física ou natureza do processo, ou ainda modelagem conceitual, que, por utilizar o equacionamento dos fenômenos envolvidos, nem sempre é possível ser realizada. Na outra, os modelos são desenvolvidos diretamente a partir de dados experimentais, representados pelos sinais de entrada e saída do sistema. Neste caso, pouco ou nenhum conhecimento prévio do sistema é necessário. Este tipo de abordagem é conhecido como

identificação de sistemas, modelagem empírica ou comportamental. Este trabalho baseia-se neste tipo de abordagem.

O termo identificação foi introduzido por (ZADEH, LOTFI A., 1956) para descrever o problema de se determinar o relacionamento entrada-saída de um sistema, de modo experimental, usando um modelo do tipo caixa-preta. Segundo ele, o termo identificação exprime o ponto crucial do problema com maior clareza do que as outras terminologias usadas com o mesmo propósito. A disciplina moderna de identificação de sistemas, como conhecida hoje, teve seu início em meados da década de 60, como parte da teoria de controle, influenciada, principalmente, pelo desenvolvimento do Filtro de Kalman (KALMAN, 1960).

No início da década de 70, Box e Jenkins; e Åström e Eykhoff tiveram um papel fundamental no desenvolvimento da área através da publicação dos seus trabalhos. O livro de Box e Jenkins (BOX; JENKINS, 1970) apresentou um arcabouço completo para a tarefa de identificação de sistemas lineares, desde a análise inicial dos dados até a construção do modelo e sua validação. Segundo (GEVERS, 2006), este livro continuou sendo uma importante referência na área de identificação de sistemas por mais de uma década. O artigo de Åström e Eykhoff (ÅSTRÖM; EYKHOFF, 1971), por sua vez, foi usado por muitos pesquisadores como inspiração para futuros trabalhos, pois apresentava o estado da arte e algumas questões importantes ainda em aberto naquela época.

Segundo Gevers (2006), em meados da década de 70, surgiram as primeiras tentativas de abordar o problema de identificação como um problema de aproximação, visando a busca pela melhor aproximação possível do sistema real em uma determinada classe de modelos. Foi então que, aos poucos, a busca pelo sistema real, que era o enfoque predominante na época, deu lugar à busca pela caracterização da melhor aproximação, tornando a caracterização do modelo de erros o principal foco da área. Durante a década de 80, os conceitos de erro de tendência e variância introduzidos por (LJUNG, 1985) levaram a comunidade científica a uma nova perspectiva do problema. A partir de então, o problema de identificação tornou-se um problema de projeto, em que o propósito do modelo passou a assumir um papel central. Esta nova abordagem de engenharia para a identificação de sistemas

distingue-se da abordagem estatística para o problema, segundo a qual o modelo deveria explicar os dados tão bem quanto possível.

O período compreendido entre meados da década de 70 e meados da década de 80 caracterizou-se pelo desenvolvimento desmedido de novas abordagens de identificação de sistemas pela comunidade de engenharia, como aponta Gevers (2006). Diante daquele cenário caótico, Ljung apresentou uma das suas maiores contribuições à área através da publicação do seu livro (LJUNG, 1999), e do desenvolvimento da *toolbox* de identificação de sistemas para o MATLAB (MATHWORKS, 2013).

Na década de 90, os sistemas não lineares começaram a ganhar força dentro da comunidade e o objetivo passou a ser a busca por classes universais de modelos caixa-preta para sistemas não lineares. Então, um esforço coletivo (GIANNAKIS; SERPEDIN, 2001; JUDITSKY et al., 1995; NELLES, 2001; SJÖBERG et al., 1995) foi empreendido para estabelecer um arcabouço comum que norteasse a identificação de sistemas não lineares a fim de se evitar o mesmo desenvolvimento caótico registrado na década passada, com relação aos sistemas lineares. De acordo com Gevers (2006), que apresenta um histórico muito mais detalhado sobre o desenvolvimento da área, o problema de identificação de sistemas não lineares é abrangente e difícil e, embora a área de identificação de sistemas seja madura, ela permanece em constante desenvolvimento.

A primeira etapa no processo de identificação experimental de sistemas é a coleta de dados. Estes dados são gerados, geralmente, através da excitação do sistema de interesse, utilizando-se algum tipo de sinal de entrada e observando-se a saída do sistema durante o tempo, de modo que, ao final, tenha-se um conjunto de dados que represente o comportamento do sistema. A partir de então, busca-se ajustar um modelo do processo que seja capaz de descrever adequadamente o comportamento do sistema. O modelo obtido é, então, testado e seu desempenho avaliado, a fim de se verificar se ele representa uma boa aproximação do sistema. Isso é geralmente feito comparando-se a saída do modelo com a saída do sistema, e esta informação é utilizada para se ajustar os parâmetros do modelo. Estas atividades são repetidas

iterativamente até que algum critério pré-estabelecido de adequação do resultado seja alcançado.

O problema de identificação de sistemas pode ser subdividido em quatro etapas (SJÖBERG et al., 1995). São elas: a escolha dos regressores; a definição do mapeamento funcional (que neste trabalho será formulado como uma expansão funcional); a estimação dos parâmetros e a validação do modelo. As duas primeiras etapas são geralmente consideradas como uma em (LJUNG, 1999), que as define como identificação da estrutura. A estrutura do modelo é considerada como o seu "esqueleto", de modo que, juntamente com os parâmetros, ela compõe o modelo.

2.1.1.2 Formulação do Problema

Seja $\mathcal{D} = \{\boldsymbol{u}(t), y(t)\}_{t=1}^{N}$ um conjunto de dados com N pares de observações entrada-saída que descrevem o comportamento de um sistema de interesse, em que $\boldsymbol{u}(t)$ representa as entradas e $y(t)$ a saída do sistema. O objetivo da identificação de sistemas é encontrar um mapeamento funcional $f: \mathcal{R}^d \rightarrow \mathcal{R}$ que seja capaz de descrever o comportamento futuro do sistema com base nessas observações passadas. Portanto, o modelo abstraído por $f(\cdot)$ pode ser formulado como:

$$\hat{y}(t|\boldsymbol{\theta}) = f(\boldsymbol{z}(t), \boldsymbol{\theta}) \tag{2.1}$$

em que $\hat{y}(t) \in \mathcal{R}$ é a saída estimada pelo modelo, cujos parâmetros livres são definidos em $\boldsymbol{\theta}$; $\boldsymbol{z}(t) \in \mathcal{R}^d$ é o vetor de regressores, que deve conter informação suficiente sobre o comportamento do sistema, e é definido como um subconjunto dos dados passados.

Estes dados representam as informações obtidas a partir de um ou mais dos sinais apresentados na Tabela 2.1, os quais são contextualizados na Figura 2.1. As setas, na figura, representam todos os sinais que podem ser utilizados para formar os regressores (descritos na Tabela 2.1). A cor azul indica que a informação é observada, enquanto a cor vermelha indica que a informação é estimada. $q-$ representa o operador de atraso, e o valor do seu expoente indica o número máximo de vezes em que o sinal é defasado no tempo.

Tabela 2.1. Possíveis sinais utilizados na definição dos regressores.

Tipo de Sinal	Descrição
$u(t-n_u)$	Entrada
$y(t-n_y)$	Saída Real
$\hat{y}(t-n_{\hat{y}}\|\theta)$	Saída Estimada
$e(t-n_e\|\theta)$	Erro

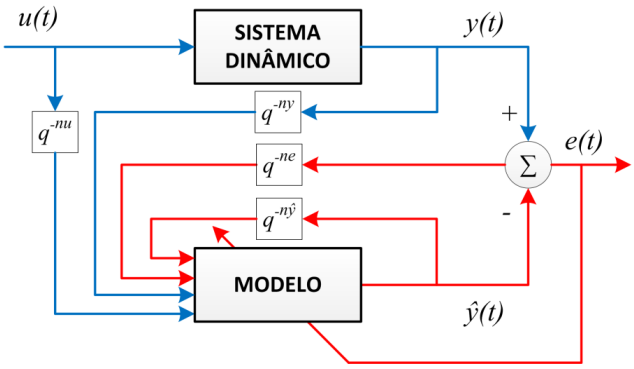


Figura 2.1. Identificação de Sistemas: um modelo é construído para representar o comportamento de um sistema dinâmico a partir de um conjunto de dados que caracterizam este sistema.

A maneira com que os sinais são combinados para formar os regressores se reflete diretamente na estrutura do modelo, e a sua definição é influenciada tanto pelas suposições acerca dos ruídos que perturbam o sistema quanto pelo modo de operação do modelo.

Segundo (LJUNG, 1999), um determinado modelo é dito ser um preditor se ele é uma função tanto das saídas observadas defasadas quanto das entradas defasadas. Por outro lado, um determinado modelo é dito ser um simulador caso ele seja uma função apenas das entradas defasadas. O primeiro tipo de configuração também é chamado de série-paralela – Figura 2.2 (a); e o último, de paralela – Figura 2.2 (b). O modelo de predição na Figura 2.2 (a) é um modelo alimentado adiante, enquanto o modelo de simulação na Figura 2.2 (b) é um modelo recorrente.

Modelos de simulação são requeridos sempre que a saída real do sistema não pode ser obervada durante a operação do modelo. Além disso, eles são particularmente úteis para emular sistemas reais e avaliar seu comportamento sobre várias condições em ambientes controlados. Modelos de predição, por sua vez, são utilizados com frequência para fazer prognósticos sobre o comportamento futuro do sistema, como, por exemplo, prever: condições climáticas, indicadores financeiros, etc.

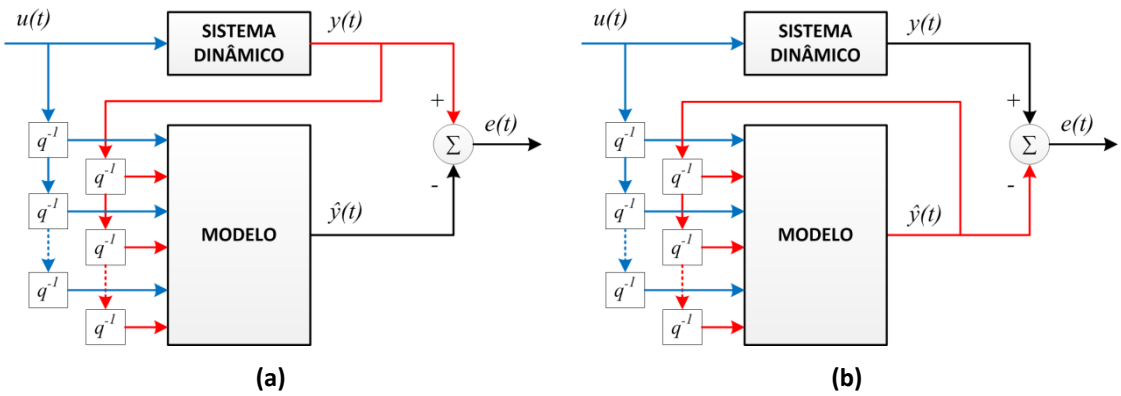


Figura 2.2. (a) Modelo de predição em configuração série-paralela (feedforward). (b) Modelo de simulação em configuração paralela (recorrente). Adaptação de (NELLES, 2001).

As estruturas usadas com maior frequência pelos modelos de sistemas dinâmicos não lineares são a NARX (Nonlinear Autoregressive with Exogenous Input):

$$\boldsymbol{z}(t) = \left[y(t-1), \dots y\left(t-n_y\right), u(t-1), \dots, u(t-n_u)\right] \tag{2.2}$$

a NARMAX (Nonlinear Autoregressive Moving Average with Exogenous Input):

$$\boldsymbol{z}(t) = \left[y(t-1), \dots y\left(t-n_y\right), u(t-1), \dots, u(t-n_u), e(t-1), \dots, e(t-n_e)\right] \tag{2.3}$$

e a NOE (Nonlinear Output Error):

$$\boldsymbol{z}(t) = \left[\hat{y}(t-1), \dots \hat{y}\left(t-n_{\hat{y}}\right), u(t-1), \dots, u(t-n_u)\right] \tag{2.4}$$

As duas últimas estruturas são recorrentes por natureza, uma vez que utilizam, respectivamente, o erro e a saída estimada como regressores. A realimentação torna o processo de identificação mais complexo (NELLES, 2001).

Uma abordagem comum para descrever o modelo formulado em (2.1) é utilizar uma expansão funcional. Nesta abordagem, o modelo é aproximado por uma série de funções de base ponderadas:

$$f(\boldsymbol{z}(t), \boldsymbol{\theta}) = \sum_{k=1}^{M} \alpha_k f_k\left(\boldsymbol{z}(t)\right) \tag{2.5}$$

em que α_k é o parâmetro que define o coeficiente de ponderação da k-ésima função de base local $f_k(\cdot)$. Essas funções de base são usualmente geradas a partir de uma mesma "função mãe" escalar $\phi(\cdot)$, e representam versões dilatadas e transladadas desta, sendo definidas como:

$$f_k\left(z_i(t)\right) = \phi\left(\rho_{k,i}\left(z_i(t) - \gamma_{k,i}\right)\right) \tag{2.6}$$

cujos parâmetros $\rho_{k,i}$ e $\gamma_{k,i}$ são o parâmetros de escala e posição, respectivamente.

A estrutura definida em (2.6) é bastante flexível e utilizada por diferentes tipos de modelos. Dependendo de como for configurada, ela pode representar redes neurais, sistemas fuzzy, etc. (LJUNG, 1999). Funções de base multidimensionais podem ser obtidas de diferentes maneiras, dependendo de como a equação (2.6) é interpretada. Uma das possibilidades é usar o produto tensorial das funções de base escalares:

$$f_k\big(\boldsymbol{z}(t)\big) = f_k(\boldsymbol{z}(t), \boldsymbol{\rho}_k, \boldsymbol{\gamma}_k) = \prod_{i=1}^{d} \phi_{k,i}\left(\rho_{k,i}\big(z_i(t) - \gamma_{k,i}\big)\right) \tag{2.7}$$

Esta abordagem é utilizada pelos sistemas fuzzy, cujas funções de base estão relacionadas às funções de pertinência. O mapeamento funcional formulado em (2.1) pode ser reescrito em termos da expansão funcional definida em (2.7) como:

$$\hat{y}(t|\boldsymbol{\theta}) = f(\boldsymbol{z}(t), \boldsymbol{\theta}) = \sum_{k=1}^{M}\left(\alpha_k \prod_{i=1}^{d} \phi_{k,i}\left(\rho_{k,i}\big(z_i(t) - \gamma_{k,i}\big)\right)\right) \tag{2.8}$$

de modo a evidenciar os subproblemas envolvidos no processo de identificação de sistemas, que são, de acordo com a formulação utilizada: (1) definir como o vetor de regressores $\boldsymbol{z}(t)$ deve ser formado; (2) definir a "função mãe" $\phi(\cdot)$ e o número M de funções de base utilizados na expansão funcional e; (3) ajustar os parâmetros livres do modelo, definidos em $\boldsymbol{\theta}$, que nesta formulação representa os parâmetros α, ρ e γ.

Este processo é geralmente guiado por uma medida de qualidade do modelo, representada por $J(\cdot)$, que define quão bem um modelo em particular explica ou se ajusta aos dados a partir dos quais ele foi identificado. Geralmente, $J(\cdot)$ representa uma função de erro, de modo que quanto menor o seu valor, melhor a qualidade do ajuste. O erro médio quadrático (MSE) é usualmente considerado para este fim:

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{t=1..N}^{N} (y(t) - \hat{y}(t))^2 \tag{2.9}$$

Para garantir que o modelo seja útil não apenas para explicar os dados de treino a partir dos quais ele foi identificado, mas também outros dados de interesse no mesmo domínio, isto é, que ele seja capaz de generalizar o comportamento do sistema dentro dos seus limites de operação, utiliza-se dados diferentes para avaliar seu desempenho, chamados dados de teste.

O mapeamento funcional, representado pela expansão funcional das funções de base, é representado neste trabalho por sistemas fuzzy recorrentes, cujas estruturas e parâmetros serão identificados pelo algoritmo de evolução diferencial. Ambos formam o núcleo básico a partir do qual este trabalho foi desenvolvido e, portanto, os conceitos fundamentais sobre eles são apresentados a seguir.

2.1.2 Sistemas Fuzzy Recorrentes

2.1.2.1 Introdução

Os sistemas fuzzy estão fundamentados na teoria nos conjuntos fuzzy. Esta teoria foi proposta por (ZADEH, LOTFI A., 1965) em meados da década de 60, e levou ao subsequente desenvolvimento da lógica fuzzy como base para a teoria do raciocínio aproximado, uma década após. Ambos os conceitos formalizaram um novo arcabouço que permitiu modelar níveis de imprecisão na representação e no processamento de informações através da ideia de raciocínio aproximado. A formalização deste arcabouço originou o desenvolvimento de modelos em que a descrição matemática é substituída por relações qualitativas expressas por termos linguísticos (ZADEH, LOTFI A., 1973). Uma classe importante de modelos é representada na forma de regras SE-ENTÃO, conhecidos como sistemas baseados em regras fuzzy, ou simplesmente sistemas fuzzy (MAMDANI, E. H., 1981; SUGENO; KANG, 1988; TAKAGI; SUGENO, 1985).

A teoria do raciocínio aproximado sugerida por Zadeh está diretamente relacionada com o processamento de informações (ZADEH, LOTFI ASKER, 1979). Ela considera expressões condicionais em que um conjunto de premissas imprecisas leva à dedução de conclusões potencialmente imprecisas. O relacionamento dessas expressões formam então regras SE-ENTÃO, em que a parte SE está relacionada às premissas e a parte ENTÃO às conclusões. Portanto, teoria de conjuntos fuzzy fornece, ao mesmo tempo, um arcabouço para lidar com informações imprecisas e um mecanismo de raciocínio diante de informações imprecisas. Segundo Zadeh, essas características conferem à teoria dos conjuntos fuzzy uma maneira aproximada, porém efetiva, de descrever o comportamento de sistemas que são complexos ou mal

definidos para admitirem uma formulação matemática precisa (ZADEH, LOTFI A., 1973).

Este raciocínio aproximado busca, de certa forma, emular a habilidade de raciocínio humano diante de cenários complexos. O grau de pertinência associado aos conjuntos fuzzy desempenha um papel no processo de raciocínio humano de várias formas distintas como, por exemplo, níveis de intensidade, graus de similaridade, níveis de incerteza, etc.

Na década de 70, começaram a surgir as primeiras aplicações de sistemas fuzzy no controle de processos mecânicos e industriais (KING; MAMDANI, 1977; MAMDANI, E.H., 1977; PAPPIS; MAMDANI, 1977; PROCYK; MAMDANI, 1979; ZADEH, LOTFI A., 1973). Inicialmente, o desenvolvimento dos sistemas fuzzy tinha como base a representação linguística do conhecimento extraído de especialistas. Nesta abordagem, o modelo é representado por um conjunto de descrições linguísticas do comportamento do sistema. O sucesso desta abordagem caracterizou-se em 1987 quando empresas japonesas como Hitachi, Omron e Fuji começaram a comercializar produtos que utilizavam a lógica fuzzy (GARRIDO, 2012).

Até o final da década de 70 os sistemas fuzzy eram modelos linguísticos gerados com base no conhecimento do especialista. Porém, a partir do início da década de 80 novas abordagens baseadas em dados passaram a ser consideradas na construção de sistemas fuzzy (SUGENO; KANG, 1988; TAKAGI; SUGENO, 1985; TONG, 1980; WANG, L. X.; MENDEL, 1992b). Neste período, Takagi e Sugeno apresentaram um novo tipo de sistema fuzzy, que atualmente é um dos mais utilizados, em que as conclusões das regras deixam de ser representadas por variáveis linguísticas (MAMDANI, E. H., 1981) e assumem uma representação funcional que descreve o comportamento local do sistema como uma função das variáveis consideradas nas premissas das regras (SUGENO; KANG, 1988; TAKAGI; SUGENO, 1985).

O interesse pelos sistemas fuzzy começou a crescer e a registrar um número cada vez maior de aplicações. Essa proliferação fez com que, na década de 90, a acurácia dos sistemas fuzzy assumisse um papel mais importante do que sua interpretabilidade (JANG; SUN, 1993; KANG; LEE; SUGENO, 1998; KOSKO, B., 1994; SETNES; BABUSKA; VERBRUGGEN, 1998; SJÖBERG et al., 1995; SUGENO, 1992;

WANG, L. X.; MENDEL, 1992a;1992b). Recentemente, porém, a interpretabilidade dos sistemas fuzzy voltou a ser considerada, de modo que se tem buscado um equilíbrio entre estes dois requisitos conflitantes (GACTO; ALCALÁ; HERRERA, 2011; ZHOU; GAN, 2008).

A busca por sistemas fuzzy tão precisos quanto possíveis, na década de 90, foi motivada, principalmente, pelos seguintes fatores: demonstração da sua capacidade de aproximação universal (KOSKO, B., 1994; WANG, L. X.; MENDEL, 1992a); demonstração da sua equivalência funcional com as redes neurais de base radial (HUNT; HAAS; MURRAY-SMITH, 1996; JANG; SUN, 1993); e o crescente desenvolvimento de modelos caixa-preta não lineares para identificação de sistemas (HELLENDOORN; DRIANKOV, 1997; JUDITSKY et al., 1995; SJÖBERG et al., 1995), especialmente aqueles baseados em redes neurais (CHEN, S.; BILLINGS, 1992; LEE, H. et al., 1991; NARENDRA, K. S.; PARTHASARATHY, 1990; PHAM; LIU, 1993), que também são aproximadores universais (HORNIK; STINCHCOMBE; WHITE, 1989).

A partir da década de 90, o grande interesse despertado pelos sistemas fuzzy e pelas redes neurais (KOSKO, BART, 1992), principalmente em problemas de identificação e controle, levou a utilização dessas duas tecnologias complementares em um arcabouço comum (denominado sistemas neuro-fuzzy ou fuzzy-neurais), evidenciando a capacidade de representação linguística dos sistemas fuzzy e o mecanismo de aprendizado das redes neurais (FIGUEIREDO; GOMIDE, 1999; JANG, 1993; JANG; SUN, 1995; LIN, C.-T.; LEE, 1991; NAUCK, D.; KRUSE, 1997;1999). Uma revisão histórica da hibridização dessas técnicas é apresentada em (NAUCK, D. D.; NÜRNBERGER, 2013). Segundo (LJUNG, 1999), quando modelos fuzzy possuem parâmetros a serem ajustados eles também são chamados modelos neuro-fuzzy.

O termo “sistema fuzzy recorrente” foi cunhado quase que ao mesmo tempo, e independentemente, durante o início da década de 90, relacionado aos conceitos de controladores fuzzy, autômatos finitos fuzzy e redes neurais recorrentes. O desenvolvimento de modelos fuzzy com estrutura recorrente ocorreu a partir de diferentes pontos de vista, porém com o propósito comum de prover memória aos sistemas fuzzy (estáticos), de modo a melhorar seu desempenho no processamento de sistemas dinâmicos.

Ramamoorthy, Huang e Hu apresentaram a ideia de sistemas fuzzy recorrentes através do desenvolvimento de sistemas de controle fuzzy com realimentação e garantia de estabilidade, motivados pelo bom desempenho e interpretabilidade dos controladores fuzzy e pelo potencial de arquiteturas com realimentação (HU, Y.; RAMAMOORTHY, 1994; RAMAMOORTHY; HUANG, 1993).

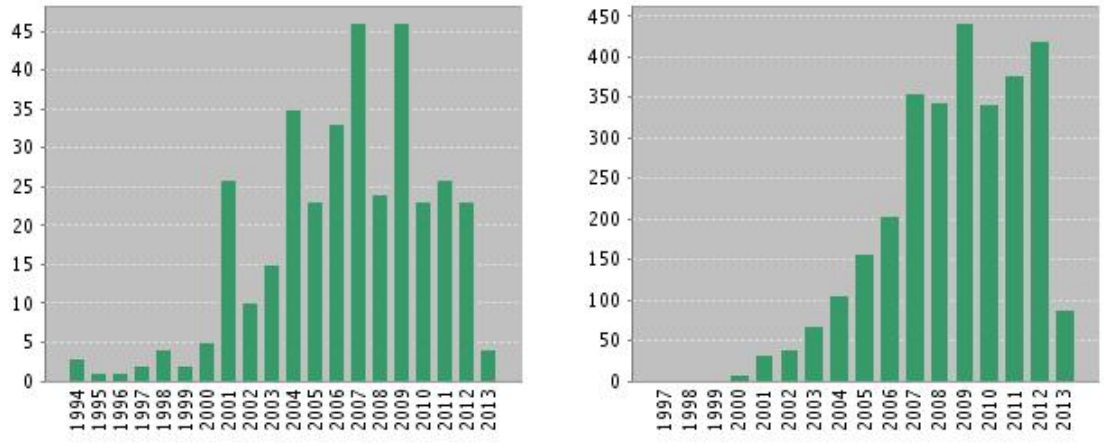
Unal e Khan sugeriram a ideia de sistemas fuzzy recorrentes baseados na capacidade de representação dinâmica das máquinas de estados finitos fuzzy. Eles mostraram que elas podem ser implementadas por sistemas neuro-fuzzy através da inclusão de uma conexão de realimentação global entre a camada de saída e a camada de entrada (UNAL; KHAN, 1994).

Gorrini e Bersini, por sua vez, propuseram um novo tipo de arquitetura que chamaram de sistema fuzzy recorrente, juntamente com um algoritmo de aprendizado para o ajuste das funções de pertinência, a fim de estender a capacidade de aproximação dos controladores fuzzy para processos dinâmicos de ordem desconhecida (GORRINI; BERSINI, 1994). Eles se basearam no desenvolvimento das redes neurais recorrentes (NARENDRA, K. S.; PARTHASARATHY, 1990;1991; NARENDRA, KUMPATI S.; PARTHASARATHY, 1992) e na demonstração da equivalência funcional entre sistemas fuzzy e redes neurais de base radial (JANG; SUN, 1993).

Desde então, diversos trabalhos apresentando o projeto e a aplicação de novos sistemas fuzzy recorrentes têm sido reportados na literatura. A Figura 2.3 ilustra o desenvolvimento desta linha de pesquisa em termos do número de (a) trabalhos publicados e (b) citações recebidas por ano, até abril/2013, segundo o repositório Web of Knowledge (Thomson Reuters), totalizando 352 documentos publicados e 2.982 citações recebidas. A pesquisa foi feita com base nos seguintes termos: ("*recurrent fuzzy*" OU "*recurrent neuro-fuzzy*") OU (*fuzzy* E ("*recurrent structure*" OU "*recurrent architecture*")) nos campos título OU assunto.

Utilizando-se os mesmos termos nos campos título, resumo e palavras-chave do repositório Scopus (Elsevier), a pesquisa retornou 445 publicações. As estatísticas desta pesquisa são apresentadas nos gráficos a seguir. A Figura 2.4 apresenta o gráfico de distribuição anual das publicações, segundo o qual fica claro que o efetivo crescimento do tema deu-se a partir do ano 2001. A Figura 2.5 mostra o gráfico do

número de publicações por nome do veículo, para aqueles com mais de 5 publicações. Este gráfico evidencia a importância dos eventos científicos e destaca a influência do IEEE no desenvolvimento dos sistemas fuzzy recorrentes. A Figura 2.6 apresenta o gráfico do número de publicações por país de filiação do autor (com mais de 10 publicações), segundo o qual fica evidente a relevância dos pesquisadores de Taiwan para o constante desenvolvimento desta linha de pesquisa. Taiwan é seguido de longe pela China. O Brasil, por sua vez, aparece com uma modesta, porém importante, contribuição.



(a) Documentos Publicados (b) Citações Recebidas

Figura 2.3 Distribuição anual do número de (a) publicações e (b) citações de documentos referentes a sistemas fuzzy recorrentes até Abril/2013 – Fonte: Web of Knowledge (Thomson Reuters)

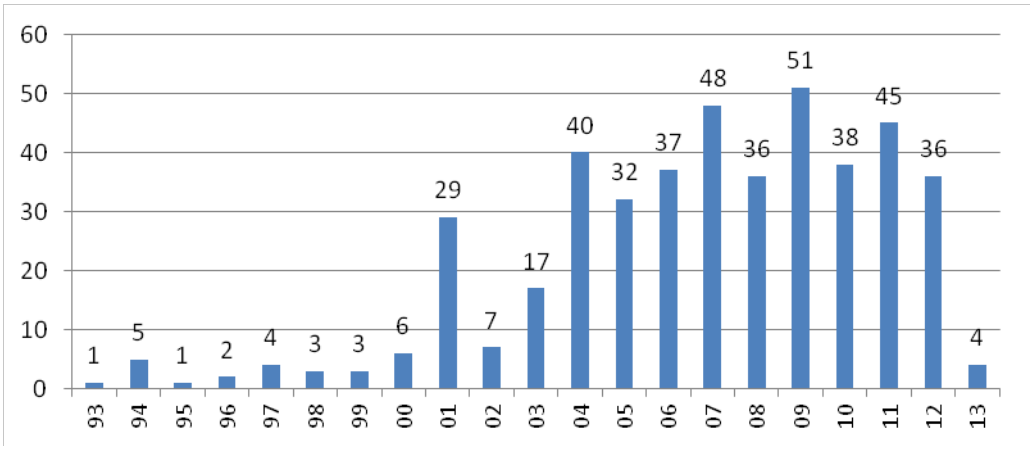


Figura 2.4 Distribuição anual do número de publicações referentes a sistemas fuzzy recorrentes até Abril/2013 – Fonte: Scopus (Elsevier)

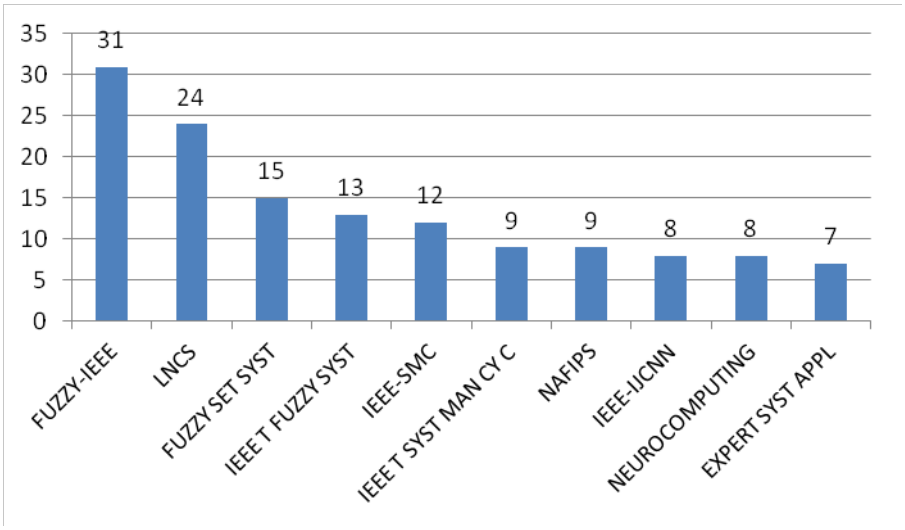


Figura 2.5 Número de publicações referentes a sistemas fuzzy recorrentes por nome do veículo (com mais de 5 publicações), até Abril/2013 – Fonte: Scopus (Elsevier)

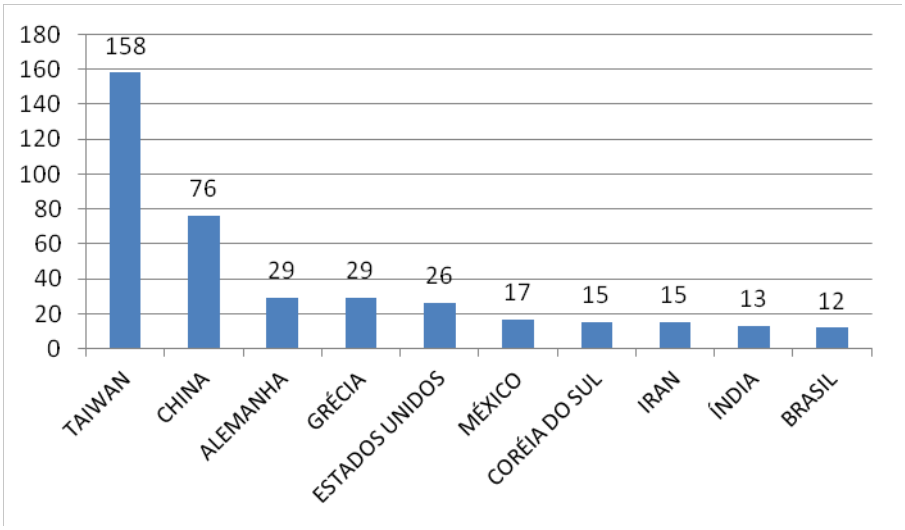


Figura 2.6 Número de publicações referentes a sistemas fuzzy recorrentes por país de filiação do autor (com mais de 10 publicações), até Abril/2013 – Fonte: Scopus (Elsevier)

2.1.2.2 Formulação Matemática

Modelos baseados em regras fuzzy são formados por (a) uma base de conhecimento, composta por uma base de dados e uma base de regras; (b) um processo de fuzzificação, responsável pelo mapeamento dos valores numéricos das variáveis para conjuntos fuzzy; (c) um mecanismo de inferência, que mapeia conjuntos fuzzy em conjuntos fuzzy, além de determinar como as regras são ativadas e combinadas; e (d) um processo de defuzzificação, que traduz o conjunto fuzzy de saída em um valor numérico. Este arcabouço é ilustrado pela Figura 2.7.

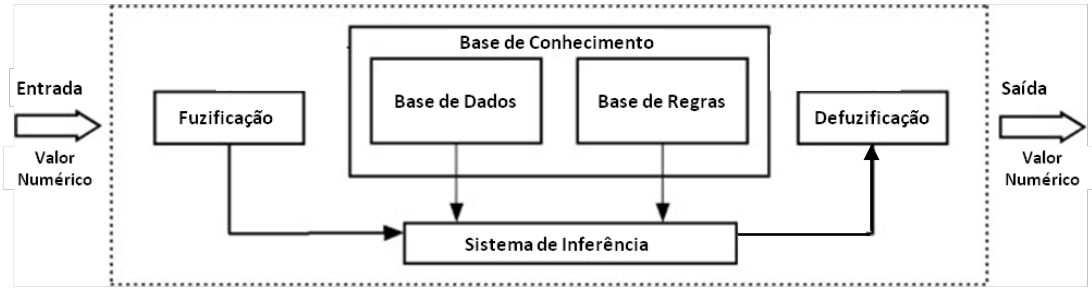


Figura 2.7. Estrutura de um Sistema Baseado em Regras Fuzzy

Nos sistemas fuzzy do tipo TSK, a etapa de defuzzificação é desnecessária, pois as conclusões das regras não são representadas por conjuntos fuzzy, e sim por valores numéricos, que podem ser definidos como constantes ou funções. Caso o valor seja uma constante, o sistema é dito ser do tipo TSK de ordem zero.

Uma componente essencial dos sistemas fuzzy são as regras linguísticas SE-ENTÃO, cuja estrutura é caracterizada por dois elementos: um conjunto de premissas e um conjunto de conclusões. Estes elementos são compostos por expressões fuzzy, e são relacionados entre si através do conceito de relação fuzzy, e processadas por mecanismos de inferência. Essas regras representam o relacionamento entre as variáveis do sistema, e possuem a seguinte forma geral: **SE** *premissas* **ENTÃO** *conclusões*, denotando que **SE** *um conjunto de premissas é satisfeito* **ENTÃO** *um conjunto de conclusões é inferido* a partir dele. A k-ésima regra de um sistema fuzzy funcional do tipo TSK pode ser representada como:

$$R_k: \textbf{SE } z_1(t) \text{ é } A_{k,1} \textbf{ E ... E } z_d(t) \text{ é } A_{k,d} \textbf{ ENTÃO } \hat{y}(t) \text{ é } \alpha_k \qquad \textbf{(2.10)}$$

em que as variáveis $z_1(t), \dots, z_d(t)$ e $\hat{y}(t)$ representam os regressores e a saída estimada; $A_{k,1}, \dots, A_{k,d}$ representam os símbolos linguísticos que definem a semântica dos respectivos conjuntos fuzzy; e $\alpha_k \in \mathcal{R}$ é o valor da conclusão da regra. A base de regras pode ser considerada, portanto, como uma forma de representação verbal do modelo, que relaciona os regressores à sua saída.

O valor que define o grau de ativação da regra, isto é, o valor que representa quão bem a premissa da regra é compatível com os dados de entrada, é calculado como:

$$\omega_k = \mu_{A_{k,1}}(z_1(t)) \wedge \dots \wedge \mu_{A_{k,d}}(z_d(t)) \qquad \textbf{(2.11)}$$

em que $\mu_{A_{k,i}}(z_i(t))$ é o grau de $z_i(t)$ no conjunto fuzzy $A_{k,1}$. O resultado da combinação entre as conjunções é considerado como uma medida de veracidade da regra, e o operador de conjunção ∧ é definido geralmente pela T-norma mínimo ou produto. Neste trabalho, a T-norma produto é considerada, de modo que o valor de ativação da regra passa a ser expresso pelo produto dos graus de pertinências:

$$\omega_k = \prod_{i=1}^{d} \mu_{A_{k,i}}(z_i(t)) \tag{2.12}$$

Como ω_k combina a informação das funções de pertinência unidimensionais $\mu_{A_{k,i}}(\cdot)$, ela pode ser vista como um tipo de função de pertinência multidimensional. À luz da formulação apresentada na seção 2.1.1.2, percebe-se que a equação (2.12) nada mais é do que uma função de base multidimensional, obtida pelo produto das funções de pertinência escalares, cuja função mãe é representada pela função de pertinência $\mu(\cdot)$.

O resultado do sistema é obtido através da agregação de cada uma das M regras, da seguinte forma:

$$\hat{y}(t) = \sum_{k=1}^{M} \left(\alpha_k \phi_k(\boldsymbol{z}(t))\right) \tag{2.13}$$

em que o valor de ϕ_k é calculado como:

$$\phi_k(\boldsymbol{z}(t)) = \omega_k / \sum_{k=1}^{M} \omega_k \tag{2.14}$$

cujo elemento do denominador é utilizado para normalizar o valor de ativação das regras, de modo que a soma de todos os fatores $\phi_k(\cdot)$ na soma ponderada definida em (2.13) some 1 para qualquer valor de $\boldsymbol{z}(t)$. Note que o valor do denominador será 1 caso os graus de pertinência somem 1 e a base de regras seja completa. Este fato pode ser observado, por exemplo, quando são utilizadas funções de pertinência triangulares* normalizadas, que satisfazem as seguintes condições:

* Muito embora a formulação considerada para representar os sistemas fuzzy geralmente utilize funções de pertinência Gaussianas, este trabalho restringe-se ao uso de funções de pertinência triangulares, a fim de manter a compatibilidade com o modelo original proposto por Gama et. at. (2008) na comparação dos resultados.

$$\sum_j \mu_{A_j}\big(z(t)\big) = 1, \quad \forall z(t)$$
$$\exists z(t) \ / \mu_{A_j}\big(z(t)\big) = 1, \ \ \forall j \tag{2.15}$$

Essas funções de pertinência geram uma partição fuzzy uniforme, cujos conjuntos são completamente definidos por um vetor de protótipos $\boldsymbol{\gamma} = [\gamma_i, \dots, \gamma_p]$, correspondentes aos vértices das funções de pertinência dos seus respectivos conjuntos fuzzy dentro do domínio no qual eles são definidos, conforme ilustrado pela Figura 2.8 abaixo.

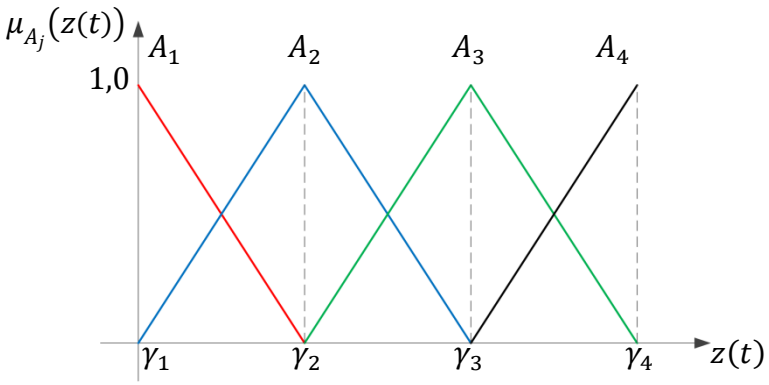


Figura 2.8. Representação de uma partição uniforme com 4 conjuntos fuzzy triangulares: A_1, A_2, A_3 e A_4, e seus respectivos valores modais γ_1, γ_2, γ_3 e γ_4.

Nestes casos, o valor de $\phi_k(\cdot)$ é dado pelo próprio grau de ativação da regra, de modo que a saída do sistema fuzzy passa ser calculada como:

$$\hat{y}(t) = \sum_{k=1}^{M} \left(\alpha_k \prod_{i=1}^{d} \mu_{A_{k,i}}\big(z_i(t)\big) \right) \tag{2.16}$$

em que as funções de pertinência são duplamente indexadas para evidenciar que são referentes à cada uma das regras (índice k) e a cada um dos regressores (índice i).

A equação (2.16) representa exatamente a formulação da expansão funcional apresentada em (2.8), em que a função mãe é representada pela função de pertinência triangular; os parâmetros de posição definem os valores modais dos conjuntos fuzzy triangulares, e os parâmetros de escala são a unidade, $\rho_{k,i} = 1, \forall k, i$.

O valor da saída do modelo definido na equação (2.16) acima pode ser reescrito de forma simplificada utilizando-se a representação vetorial:

$$\hat{y}(t) = \boldsymbol{\omega}(t)\boldsymbol{\alpha} \tag{2.17}$$

em que $\boldsymbol{\omega}(t)$ é o vetor linha formado pelos graus de ativação das regras, e $\boldsymbol{\alpha}$ é o vetor coluna formado pelos valores das conclusões das regras.

Os sistemas fuzzy recorrentes são uma extensão dos sistemas fuzzy tradicionais em que algum tipo de memória é incorporada à sua estrutura. Esta memória pode ser implementada de diversas formas, como será visto na seção 2.2.1.2, que apresenta o estado da arte dos modelos baseados em sistemas fuzzy recorrentes. De um modo geral, a memória é implementada através de conexões de realimentação, que podem se originar em diferentes elementos de um sistema fuzzy como, por exemplo, nas funções de pertinência, no consequente, na saída do sistema, etc.

No sistema fuzzy recorrente desenvolvido por Gama et al. (2008), as regras são caracterizadas pela presença de variáveis internas que representam os estados do sistema. A recorrência é obtida através da relação de dependência temporal entre essas variáveis, que aparecem tanto no antecedente quanto no consequente das regras. As regras possuem a seguinte forma:

$$R_k: \textbf{ SE } \boldsymbol{x}(t) \text{ é } B_k \textbf{ E } \boldsymbol{z}(t) \text{ é } A_k \textbf{ ENTÃO } x_1(t+1) \text{ é } \alpha_k \qquad \textbf{(2.18)}$$

em que $\boldsymbol{x}(t) = [x(t), \dots, x(t - d_x - 1)]$ representa o vetor das variáveis internas do sistema, composto por valores do estado no instante atual $x(t)$ e em instantes anteriores. Por isso, apenas a primeira componente do vetor de estados é considerada no consequente das regras. O vetor de regressores $\boldsymbol{z}(t) = [u_1(t), \dots, u_{d_u}(t)]$ é composto pelas variáveis de entrada do sistema; A_k e B_k são os rótulos linguísticos dos conjuntos fuzzy multidimensionais associados às variáveis de estado e aos regressores, respectivamente, cujos graus de pertinência são calculados de acordo com (2.12). Neste caso, o vetor com os graus de ativação das regras $\boldsymbol{\omega}(t)$ é dado pelo produto tensorial (através do operador de Kroneker) entre os vetores com os graus de pertinência multidimensionais dos estados e dos regressores, em todas M as regras:

$$\boldsymbol{\omega}(t) = \boldsymbol{\mu_B}\big(\boldsymbol{x}(t)\big) \otimes \boldsymbol{\mu_A}\big(\boldsymbol{z}(t)\big) \qquad \textbf{(2.19)}$$

em que os vetores $\boldsymbol{\mu_B}\big(\boldsymbol{x}(t)\big)$ e $\boldsymbol{\mu_A}\big(\boldsymbol{z}(t)\big)$ são formados, respectivamente, como:

$$\boldsymbol{\mu_B}\big(\boldsymbol{x}(t)\big) = \big[\mu_{B_1}\big(\boldsymbol{x}(t)\big), \dots, \mu_{B_M}\big(\boldsymbol{x}(t)\big)\big]$$

e $\qquad \textbf{(2.20)}$

$$\boldsymbol{\mu_A}\big(\boldsymbol{z}(t)\big) = \big[\mu_{A_1}\big(\boldsymbol{z}(t)\big), \dots, \mu_{A_M}\big(\boldsymbol{z}(t)\big)\big]$$

e as funções de pertinência multidimensionais $\mu_{B_i}\big(\boldsymbol{x}(t)\big)$ e $\mu_{A_i}\big(\boldsymbol{z}(t)\big)$ são calculadas como em (2.12). Finalmente, a saída do sistema é calculada usando (2.17).

2.1.3 Evolução Diferencial

2.1.3.1 Introdução

A Evolução diferencial é um método de otimização evolucionário baseado em direção, que foi desenvolvido por R. Storn e K. Price na década de 90 para lidar com problemas de otimização em espaços de busca contínuos (STORN, RAINER; PRICE, 1997). Este método de otimização é simples, eficiente, robusto, e capaz de lidar com funções-objetivo não diferenciáveis, não lineares e multimodais. A evolução diferencial pertence a uma classe de métodos de otimização genéricos, globais, estocásticos, populacionais, e que se baseiam num processo de evolução coletiva, cuja estrutura simula os mecanismos naturais de reprodução, mutação, recombinação e seleção.

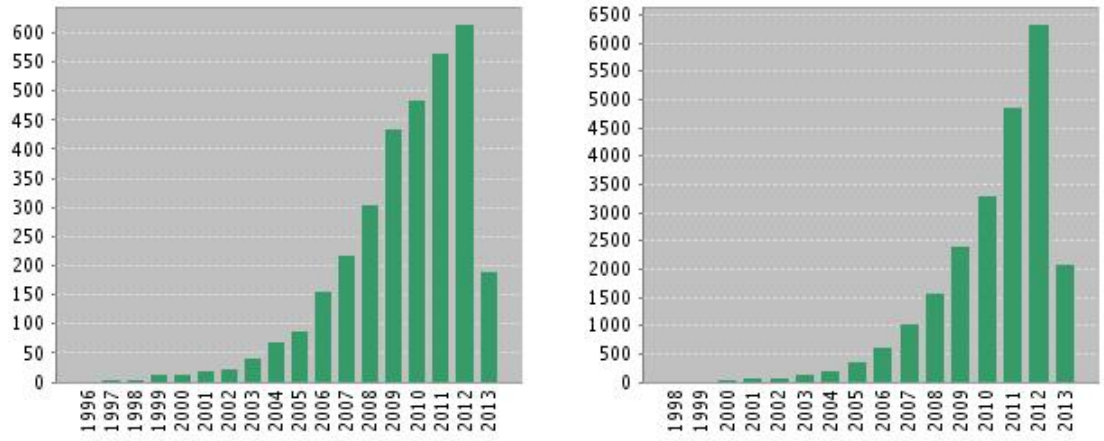
O desenvolvimento da evolução diferencial se deu a partir da necessidade de estender o método de otimização chamado de recozimento genético (PRICE, KENNETH V., 1994) para ajustar os coeficientes do polinômio de Chebychev. O recozimento genético foi apresentado por K. Price em outubro de 1994 como uma hibridização entre as técnicas de otimização conhecidas como recozimento simulado (KIRKPATRICK; GELATT; VECCHI, 1983) e algoritmos genéticos (GOLDBERG, 1989), para resolver problemas de otimização combinatória. R. Storn, porém, queria usá-lo para resolver problemas de otimização contínua e entrou em contato com K. Price para solicitar esta extensão (FEOKTISTOV, 2006).

Feoktistov (2006) relata que K. Price fez algumas alterações no algoritmo original, que passou a usar números de ponto flutuante ao invés de dígitos binários, e teve as operações lógicas substituídas por operações de aritmética vetorial. Essas alterações tornaram o recozimento genético em um método de otimização para espaços de busca contínuos, e deram origem ao operador de mutação diferencial, que é um dos elementos-chave no algoritmo de evolução diferencial. A partir da versão contínua do algoritmo, K. Price e R. Storn perceberam que se a mutação diferencial fosse combinada com mecanismos de recombinação discreta e seleção elitista por pares, o fator de recozimento do algoritmo não seria mais necessário.

Em vista dessas descobertas, novas alterações foram incorporadas ao algoritmo, dando origem a um novo método de otimização, que R. Storn e K. Price

passaram a chamar de evolução diferencial, em alusão ao operador de mutação. Este novo método foi descrito pela primeira vez em 1995, por meio de um relatório técnico do Instituto Internacional de Ciência da Computação de Berkeley, na Califórnia (EUA) (STORN, RAINER; PRICE, 1995). Um ano após, seu sucesso foi demonstrado internacionalmente, pela conquista do terceiro lugar na primeira competição internacional de otimização evolucionária, que aconteceu em maio de 1996 em Nagoya (Japão), durante a primeira conferência internacional de computação evolucionária, promovida pelo IEEE (STORN, R.; PRICE, 1996). No ano seguinte, a evolução diferencial voltou a figurar entre os melhores métodos evolucionários de otimização na segunda edição da competição (PRICE, K. V., 1997). O sucesso obtido nessas competições incentivou a divulgação do algoritmo para a comunidade científica internacional por meio de um artigo, publicado no *Journal of Global Optimization*, em 1997 (STORN, RAINER; PRICE, 1997).

Desde então, vários esforços têm sido empreendidos a fim de melhorar e estender o algoritmo, e diversos trabalhos têm reportado o sucesso da evolução diferencial na solução de um número cada vez maior de problemas reais em diferentes áreas de aplicação, como projetos de engenharia, tomada de decisão, escalonamento, controle, processamento de sinais, identificação de sistemas, reconhecimento de padrões, mineração de dados; biologia molecular, dentre outras (DAS; SUGANTHAN, 2011; FEOKTISTOV, 2006; NERI; TIRRONEN, 2010; PLAGIANAKOS et al., 2008; PRICE, K. et al., 2005). A Figura 2.9 a seguir mostra o crescimento exponencial do interesse pela evolução diferencial ao longo dos anos, em termos da quantidade de (a) trabalhos publicados (3.250) e (b) citações recebidas (23.144) até abril/2013, segundo o repositório Web of Knowledge (Thomson Reuters), utilizando-se os termos *optimization* E "*differential evolution*" no campo assunto.



(a) Documentos Publicados (b) Citações Recebidas

Figura 2.9 Distribuição anual da quantidade de (a) publicações e (b) citações de documentos referentes a evolução diferencial até Abril/2013 – Fonte: Web of Knowledge (Thomson Reuters).

2.1.3.2 Algoritmo

A ideia básica do processo de busca dos algoritmos de otimização evolucionários é a seguinte: dada uma população de indivíduos, a pressão causada pelo ambiente acarreta a seleção natural desses indivíduos, de maneira que apenas os mais adaptados ao meio sobrevivam e se reproduzam, aumentando assim a aptidão da população. Neste processo, a população de indivíduos representa um conjunto de soluções candidatas no espaço de busca das soluções potenciais. Cada uma dessas soluções é avaliada a fim de mensurar a adequação do indivíduo ao ambiente. Esses indivíduos sofrem transformações e são combinados entre si a fim de produzirem novas soluções. Se as novas soluções forem melhores do que as antigas, elas as substituem; caso contrário, elas são descartadas. Isso garante que a população evolua em direção à melhor solução.

Na evolução diferencial, os indivíduos são representados por vetores de números reais, e a transformação desses indivíduos é realizada através da diferença entre vetores, responsável por gerar vetores que perturbam a população, e cujas direções guiam a busca pela melhor solução. Esta natureza direcional-diferencial no processo de busca é a principal característica que distingue a evolução diferencial dos demais métodos de otimização evolucionários.

Considerando que todo problema de maximização pode ser convertido em um problema de minimização, pode-se generalizar a formulação do problema de otimização para espaços de busca contínuos como:

$$\boldsymbol{v}_* = \arg\min_{\boldsymbol{v}} \ \psi(\boldsymbol{v}) \tag{2.21}$$

em que $\psi(\cdot)$ é a função objetivo a ser minimizada; $\boldsymbol{v} \in \mathcal{R}^D$ é o um ponto no espaço de busca D-dimensional e cujas componentes $\boldsymbol{v}_i \in \mathcal{R}$ são chamadas de variáveis de decisão ou projeto, sujeitas a restrições laterais $\boldsymbol{l}_j^{\inf} \leq \boldsymbol{v}_j \leq \boldsymbol{l}_j^{\sup}$; e $\boldsymbol{v}_*$ representa a solução ótima. O objetivo da evolução diferencial é, segundo a formulação definida em (2.21), determinar os valores das variáveis de decisão $\boldsymbol{v}_i$ para os quais a função objetivo $\psi(\cdot)$ tem valor mínimo.

A evolução diferencial lida com a minimização da função objetivo definindo inicialmente um conjunto (chamado de população) $\mathcal{V} = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_{NP}\}$ com NP possíveis soluções geradas aleatoriamente, cada uma delas codificada como um vetor de pontos-flutuantes, representando um ponto no espaço de busca, cujo domínio é definido por restrições laterais previamente estabelecidas. A inicialização mais simples desses vetores é feita da seguinte forma:

$$\boldsymbol{v}_{i,j} = \mathbf{rnd}_j(0,1)\left(\boldsymbol{l}_j^{\sup} - \boldsymbol{l}_j^{\inf}\right) + \boldsymbol{l}_j^{\inf} \tag{2.22}$$

em que $\boldsymbol{v}_{i,j}$ representa a j-ésima variável da i-ésima solução inicial; $\mathbf{rnd}_j(0,1)$ é a função geradora de números aleatórios, distribuídos uniformemente, cujo índice j indica que novos valores são gerados para cada variável; $\boldsymbol{l}_j^{\inf}$ e $\boldsymbol{l}_j^{\sup}$ são valores definidos previamente que estabelecem, respectivamente, os limites inferiores e superiores da j-ésima variável. Existem, ainda, outros métodos de inicialização não convencionais, que buscam diminuir o número de iterações necessárias para encontrar a melhor solução (ALI; PANT; ABRAHAM, 2013; RAHNAMAYAN; TIZHOOSH; SALAMA, 2007).

Uma vez que as NP soluções iniciais foram geradas, a próxima etapa é avaliar cada uma delas. Caso o valor da função objetivo seja conhecido (o que raramente acontece), a avaliação das soluções candidatas permite identificar se alguma delas representa a solução ótima para o problema, através do valor do seu valor de aptidão. Contudo, além de verificar se a solução ótima foi encontrada, a avaliação das soluções

candidatas tem por objetivo atribuir um valor de aptidão aos vetores, para que seja possível comparar o desempenho entre eles e verificar se uma determinada solução é melhor do que outra.

O algoritmo de evolução diferencial implementa um método geracional de evolução que processa iterativamente o conjunto inicial de soluções em busca da solução ótima. Este método baseia-se na utilização de três mecanismos básicos de evolução: mutação, recombinação e seleção. O mecanismo de mutação confere diversidade ao conjunto de soluções, e permite que elas evitem ótimos locais. Além disso, a maneira com que este mecanismo é implementado afeta diretamente a forma com que o algoritmo explora o espaço de busca. O mecanismo de recombinação, por sua vez, confere preservação às soluções e assume um papel construtivo no processo evolutivo. Os mecanismos de mutação e recombinação são os responsáveis por criar NP novas soluções a cada geração. Finalmente, o mecanismo de seleção usa a função objetivo, e confere ao algoritmo um caráter elitista, garantindo que apenas as NP melhores soluções sejam mantidas na população a cada geração. Este mecanismo utiliza o valor de aptidão (valor da função objetivo) dos vetores para selecionar a melhor solução entre aquelas representadas pelos vetores candidatos e experimentais.

2.1.3.2.1 Mutação

A ideia básica por trás do mecanismo de mutação diferencial é gerar um vetor diferencial aleatório a partir do qual as informações de direção e magnitude são usadas para explorar novas regiões do espaço de busca. Este mecanismo pode ser generalizado da seguinte maneira:

$$\boldsymbol{w} = \boldsymbol{\beta} + F\boldsymbol{\delta} \tag{2.23}$$

em que $\boldsymbol{w}$ é o vetor mutante, $\boldsymbol{\beta}$ é o vetor base, F é a constante de diferenciação, também chamada de fator de escala, e $\boldsymbol{\delta}$ é o vetor diferencial. Esses são os elementos básicos do mecanismo de mutação diferencial, e podem ser implementados de diferentes maneiras, dependendo de como os vetores base e diferencial são definidos.

O fator de escala F é um parâmetro de controle responsável por regular a maneira com que o espaço de busca é explorado. Valores pequenos de F permitem

explorar a vizinhança local do vetor base, enquanto que valores maiores permitem explorar regiões mais distantes. A fim de se evitar a convergência prematura, o valor de F deve ser grande o suficiente para minimizar o efeito da pressão de seleção causada pela natureza elitista do mecanismo de seleção, que tende a minimizar a diversidade da população.

O valor de F pode ser constante durante todo o processo de busca ou assumir valores aleatórios diferentes a cada geração. Em seu artigo seminal, R. Storn e K. Prive definem $F \in [0,2]$ e sugerem $F = 0{,}5$ como uma boa escolha inicial (STORN, RAINER; PRICE, 1997). Posteriormente, porém, um novo limite superior $F < 1$ é sugerido (PRICE, K. et al., 2005). A existência de um limite inferior efetivo como $F \sim 0{,}3$ para garantir a convergência da população foi demonstrada em (ZAHARIE, 2002). Em (MEZURA-MONTES; VELÁZQUEZ-REYES; COELLO, 2006) os autores sugerem usar valores de $F \in [0{,}3; 0{,}9]$ definidos aleatoriamente a cada geração. A aleatoriedade de F é uma maneira simples, porém efetiva, de minimizar o risco de estagnação sem a necessidade de aumentar o tamanho da população (PRICE, K. et al., 2005).

Assumindo-se que o vetor base é um vetor aleatório e que apenas dois vetores são utilizados para criar o vetor diferencial, a geração do vetor mutante pode ser formulada como:

$$\boldsymbol{w} = \boldsymbol{v}_{r3} + F(\boldsymbol{v}_{r1} - \boldsymbol{v}_{r2}) \quad \textbf{(2.24)}$$

em que os índices $r1$, $r2$ e $r3$ são escolhidos aleatoriamente a cada mutação, e devem ser diferentes entre si. A Figura 2.10 na próxima página ilustra o processo de geração do vetor mutante.

De um modo geral, o mecanismo de mutação diferencial confere ao algoritmo a habilidade de auto adaptação espontânea à função objetivo durante o seu processo evolutivo. No início das gerações, a magnitude do deslocamento no espaço de busca tende a ser maior, pois os vetores estão aleatoriamente espalhados no espaço de busca. Porém, à medida que a população evolui, as soluções vão convergindo em direção ao ótimo global, e a distância entre os vetores se torna cada vez menor, reduzindo, portanto, a magnitude do deslocamento no espaço.

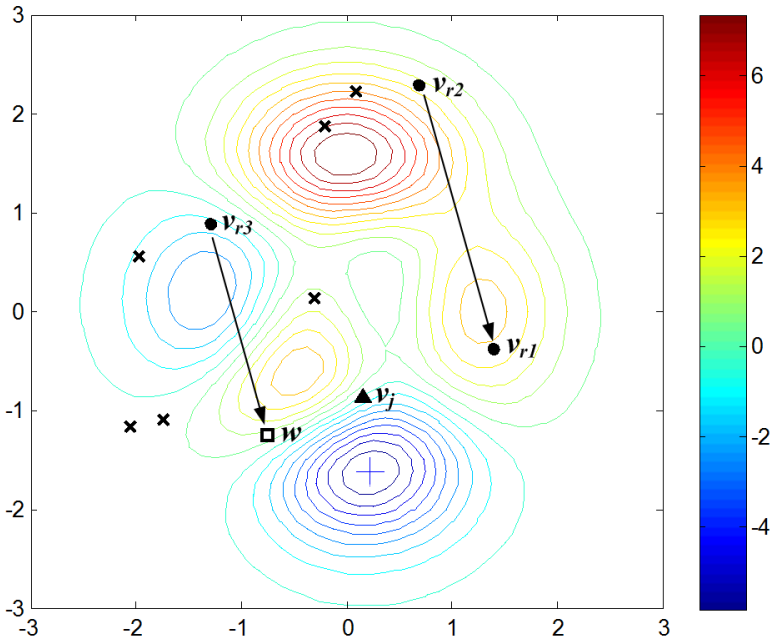


Figura 2.10. Processo de geração do vetor mutante w. Três vetores distintos da população são selecionados aleatoriamente: v_{r1}, v_{r2} e v_{r3}; a diferença ponderada entre os dois primeiros vetores $F(v_{r1} - v_{r2})$ é adicionada ao terceiro, gerando então o vetor mutante.

Em (TING; HUANG, 2009) os autores investigam o uso de mais de um vetor diferencial no mecanismo de mutação, e concluem que a utilização de mais de um vetor diferencial raramente proporciona melhorias em termos de qualidade da solução e velocidade de convergência. Eles sugerem, ainda, que o aumento do número de vetores diferenciais causa convergência prematura. Portanto, neste trabalho, um único vetor diferencial será considerado, tal que $\boldsymbol{\delta} = \boldsymbol{v}_{r1} - \boldsymbol{v}_{r2}$.

Duas outras estratégias são frequentemente consideradas na definição do mecanismo de mutação diferencial. Ao invés de utilizar uma solução aleatória para definir a região de busca, uma delas utiliza a melhor solução e a outra utiliza tanto a melhor solução quanto a solução atual que está sendo explorada. No primeiro caso, o vetor mutante é definido como:

$$\boldsymbol{w} = \boldsymbol{v}_{\#} + F\boldsymbol{\delta} \tag{2.25}$$

e no segundo, como:

$$\boldsymbol{w} = [\boldsymbol{v}_i + F(\boldsymbol{v}_{\#} - \boldsymbol{v}_i)] + F\boldsymbol{\delta} \tag{2.26}$$

em que $\boldsymbol{v}_{\#}$ representa a melhor solução da população na geração atual, e $\boldsymbol{v}_i$ a i-ésima solução que está sendo considerada no processo reprodutivo. A implementação formulada em (2.24) é conhecida como DE/*rand*/1, enquanto que as formulações em (2.25) e (2.26) são referidas como DE/*best*/1 e DE/*target-to-best*/1, respectivamente.

2.1.3.2.2 Recombinação

O mecanismo de recombinação complementa o mecanismo de mutação no processo reprodutivo. Ele é responsável por criar o vetor $\boldsymbol{s}$ que competirá por um lugar no conjunto de soluções candidatas. O "material genético" utilizado para construir a nova solução é herdado do vetor mutante $\boldsymbol{w}$ com probabilidade CR, e do vetor alvo $\boldsymbol{v}_i$ com probabilidade $1 - CR$, no tipo de recombinação binomial, formulado como:

$$s_j = \begin{cases} w_j & \text{se } \left(\mathbf{rnd}_j(0,1) \leq CR\right) \text{ ou } j = \mathbf{rndn}(1,d) \\ v_{i,j} & \text{caso contrário} \end{cases} \quad \textbf{(2.27)}$$

em que s_j, w_j e $v_{i,j}$ são, respectivamente, a j-ésima componente do vetor experimental, do vetor mutante e do vetor alvo; $\mathbf{rnd}_j(0,1)$ é a função que gera diferentes números aleatórios para cada componente j do vetor; $\mathbf{rndn}(1,d)$ é a função que seleciona um índice aleatório do vetor, garantindo que pelo menos um elemento do vetor experimental seja herdado do vetor mutante; finalmente, $CR \in [0,1]$ é o parâmetro de controle que reflete a probabilidade com que o vetor experimental herda as componentes do vetor mutante. Ele é chamado de constante de recombinação.

A recombinação binomial é similar à recombinação uniforme utilizada em outros algoritmos evolucionários. Seu nome está relacionado ao fato do número de componentes herdadas a partir do vetor mutante ter uma distribuição binomial (PRICE, K. et al., 2005). Outro tipo de recombinação considerada na evolução diferencial é conhecido como recombinação exponencial, que é similar às recombinações de um e dois pontos, usadas nos algoritmos genéticos. Na recombinação exponencial, o vetor experimental contém um número consecutivo de componentes herdadas do vetor mutante. Ela é formulada como:

$$s_j = \begin{cases} w_j & \text{se } j \in \{k, k+1, \dots, k+K-1\} \\ v_{i,j} & \text{caso contrário} \end{cases} \quad \textbf{(2.28)}$$

em que o índice inicial $k \in \{1, \dots, d\}$ é escolhido aleatoriamente e o inteiro $K \in \{1, \dots, d\}$, que representa o número de componentes a ser herdado do vetor mutante, é definido aleatoriamente. Esse tipo de recombinação foi utilizado na implementação original do algoritmo. Porém, atualmente, a maioria das implementações utiliza a recombinação binomial. A influência do mecanismo de recombinação no

comportamento da evolução diferencial é estudada em (ZAHARIE, 2009), que destaca que, para um mesmo valor de CR, a probabilidade de herdar "material genético" do vetor mutante é maior na recombinação binomial do que na exponencial. Esta conclusão ratifica o que já fora empiricamente sugerido em outros trabalhos quanto à definição do valor de CR para ambos os tipos de recombinação: que valores maiores devem ser utilizados na recombinação exponencial, e que valores menores são mais indicados para a recombinação binomial (GONG; TUSON, 2007; STORN, RAINER; PRICE, 1997).

A Figura 2.11, abaixo, ilustra o processo de geração do vetor $\boldsymbol{s}$. Nela, o vetor alvo $\boldsymbol{v}_i$ e o vetor mutante $\boldsymbol{w}$ são usados para gerar o vetor $\boldsymbol{s}$, cujas componentes são definidas como: $s_1 = v_{i,1}$ e $s_2 = w_2$.

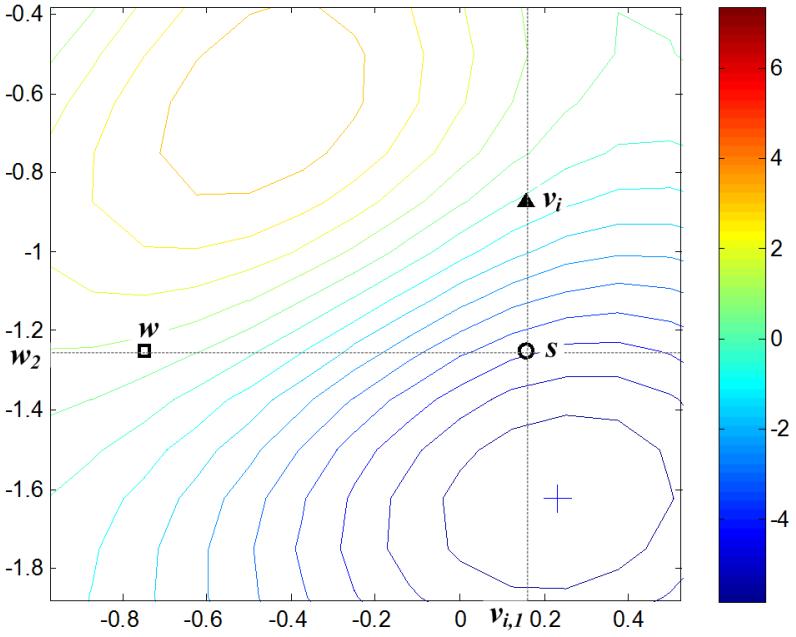


Figura 2.11. Geração do vetor experimental s, a partir do vetor alvo v_i e o do vetor mutante w.

2.1.3.2.3 Seleção

O mecanismo de seleção é ilustrado pela Figura 2.12 na página seguinte. Ele usa a função objetivo de modo a garantir que apenas as melhores soluções sejam mantidas na população. Caso o vetor $\boldsymbol{s}$ possua melhor aptidão que o vetor alvo, ele o substituirá na população; caso contrário, ele é descartado e o vetor v_i é mantido. Este mecanismo pode ser formulado como:

$$v_{i,j} = \begin{cases} s_j & \text{se } \left(\psi(s_j) < \psi(v_{i,j})\right) \\ v_{i,j} & \text{caso contrário} \end{cases} \quad \textbf{(2.29)}$$

em que $\psi(\cdot)$ é a função objetivo.

Após os operadores de mutação, recombinação e seleção terem sido aplicados a todos os vetores da população atual, a nova população de soluções candidatas assume o lugar da população atual na próxima geração, e este processo segue iterativamente até que algum critério de parada pré-estabelecido seja satisfeito. Geralmente, utiliza-se um número máximo de gerações, MAX_GER, como critério. Seu valor deve ser escolhido de modo a permitir que o espaço de busca seja bem explorado. Alternativamente, pode-se utilizar alguma estatística da aptidão populacional para se implementar algum critério mais sofisticado. A Tabela 2.2 apresenta uma visão geral do algoritmo de evolução diferencial.

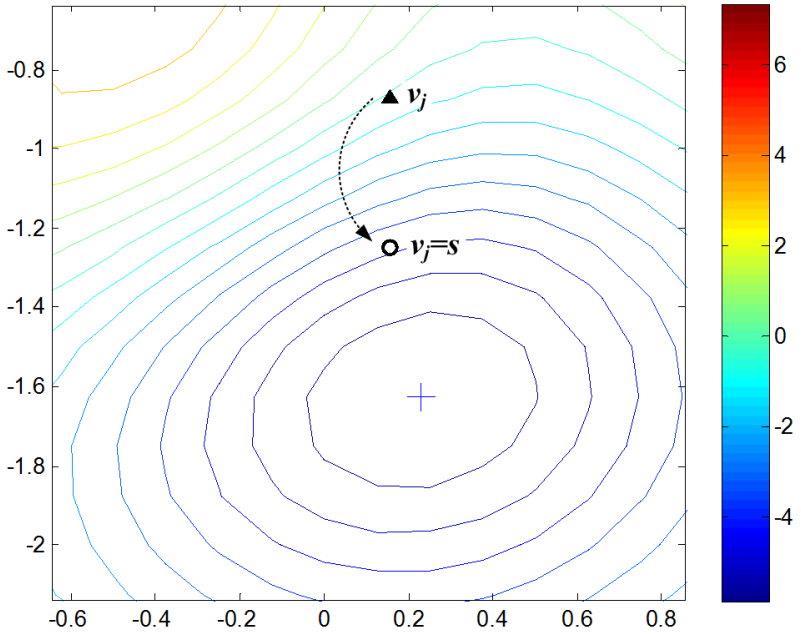


Figura 2.12. Mecanismo de seleção elitista da evolução diferencial. Se a solução representada pelo vetor s for melhor que a solução representada pelo vetor alvo v_i, então o vetor s assume o lugar do vetor v_i na população. Caso contrário, o vetor s é descartado e o vetor v_i continua na população.

Tabela 2.2. Algoritmo de Evolução Diferencial

Entrada:	Parâmetros F, CR, NP e MAX_GER
Saída:	Indivíduo que representa a melhor solução encontrada: $\boldsymbol{v}_{\#}$

```
01  Início
02  Configurar o algoritmo de acordo com os parâmetros
03  c ← 0
04  Inicializar a população V^(c)
05  Avaliar todos os vetores v_i^(c) ∈ V^(c) usando ψ(·)
06  Enquanto c < MAX_GER
07      c ← c + 1
08      Para cada vetor v_i^(c) ∈ V^(c) faça
09          Gerar o vetor mutante w
10          Gerar o vetor experimental s
11          Avaliar s usando ψ(·)
12          Definir v_i^(c+1)
13      Fim-Para
14  Fim-Enquanto
15  Retornar v_j^(c) ∈ V^(c): ψ(v_j^(c)) < ψ(v_i^(c)) ∀ i = 1, ..., NP  i ≠ j
16  Fim
```

A seguir, será apresentado o estado da arte relacionado aos sistemas fuzzy recorrentes; e logo depois, relacionado aos métodos de identificação baseados no algoritmo de evolução diferencial.

2.2 Estado da Arte

Nesta seção serão apresentados os principais modelos baseados em sistemas fuzzy recorrentes e as principais abordagens, baseadas na evolução diferencial, utilizadas para identificar a estrutura e os parâmetros de sistemas fuzzy. Como seria impraticável discorrer sobre todos os trabalhos publicados, apenas aqueles que tenham sido desenvolvidos ou utilizados recentemente serão considerados. A Tabela 2.3, na página 48, e a Tabela 2.4, na página 54, apresentam um resumo dos principais modelos e das principais abordagens de identificação, respectivamente.

2.2.1 Modelos Baseados em Sistemas Fuzzy Recorrentes

2.2.1.1 Aplicações

Modelos baseados em sistemas fuzzy recorrentes têm atraído grande interesse de pesquisadores graças aos resultados promissores alcançados no processamento de sistemas dinâmicos não lineares em diferentes tipos de aplicações, nas mais diversas áreas, como:

- **Automação**

Um sensor inferencial baseado em sistemas neuro-fuzzy recorrentes é utilizado em (JASSAR; LIAO; ZHAO, 2011) para projetar um controlador inferencial a fim de melhorar o funcionamento dos sistemas de aquecimento residencial em termos de eficiência energética e qualidade ambiental. Em (LIN, F.-J. et al., 2012) um modelo dinâmico com três graus de liberdade é proposto para um sistema de controle de posicionamento de uma gantry-stage para inspeção ótica, usando uma rede neural fuzzy recorrente com funções de pertinência intervalares assimétricas. O projeto de um sistema veicular anti-colisão que usa um controlador baseado em uma rede neural fuzzy recorrente é proposto em (MON, Y.-J.; LIN, C.-M., 2012). Em (CHEN, T.-C. et al., 2013) um modelo baseado em redes neurais fuzzy recorrentes é proposto para o controle em tempo real de motores ultrassônicos de ondas viajantes. Em (LIN, C. H.; LIN, 2013), os autores apresentam o desenvolvimento de um sistema de acionamento magnético de tração permanente para motor síncrono usando estimador de fluxo de rotor para motocicletas elétricas baseado em redes neurais híbridas fuzzy recorrentes.

- **Biotecnologia**

Os autores em (ADAMY; SCHWUNG, 2010) aplicam um sistema fuzzy recorrente de tempo contínuo na modelagem qualitativa de um complexo processo bioquímico sem a necessidade de fazer quaisquer suposições sobre as taxas de consumo de substratos envolvidas no processo. Em (BARUCH; HERNANDEZ, 2011) os autores propõem o uso de uma abordagem multi-modelo hierárquica, baseada em redes fuzzy-neurais recorrentes, para identificação dos parâmetros de um digestor anaeróbio para tratamento de esgoto de bioprocessos.

- **Ecologia**

Em (MON et al., 2013), os autores desenvolvem uma metodologia de controle robusto baseada em uma rede neuro-fuzzy recorrente para manter as multi-biomassas de sistemas ecológicos em estado de equilíbrio ótimo dentro de uma pequena vizinhança em um ecossistema sem perturbações.

- **Energia**

Os autores em (GE et al., 2010; WANG, L.; LUN; CAO, 2010) utilizam redes neurais fuzzy recorrentes dinâmicas para previsão de carga de energia elétrica em curto prazo. Em (LIN, W. M.; HONG; CHENG, 2011) os autores utilizam uma rede neural fuzzy recorrente para estimar a velocidade rotacional de um gerador de indução, sem a utilização de sensores, em um sistema de geração de energia eólica.

- **Genética**

O autor em (HAMED, 2010) descreve o uso conjunto de redes neuro-fuzzy recorrentes e redes Petri fuzzy para modelar as complexas relações causais entre genes a partir de dados experimentais. Segundo ele, a abordagem permite extrair informações sobre as interações genéticas, levando-se em consideração aspectos dinâmicos de regulação entre os genes, de forma altamente interpretável. Em (VINEETHA; CHANDRA SHEKARA BHAT; IDICULA, 2012) os autores aplicam redes neuro-fuzzy recorrentes do tipo TSK aos problemas de extração de relacionamentos regulatórios entre genes, e geração de redes regulatórias de genes, a partir de dados de microarray de pacientes portadores de câncer de cólon, e relatam quase 90% de acurácia na extração das relações segundo conhecimento biológico. Em outro trabalho (VINEETHA; CHANDRA SHEKARA BHAT; IDICULA, 2013), os referidos autores utilizam o mesmo sistema neuro-fuzzy recorrentes do tipo TSK para modelar redes de interação miRNA-mRNA a partir de perfis de expressão emparelhados de miRNA e mRNA, tanto para tumores de cólon quanto para tecidos normais, a fim de conseguir distingui-los.

- **Hidrologia**

Em (EVSUKOFF; DE LIMA; EBECKEN, 2011) os autores utilizam um sistema fuzzy recorrente para simular a vazão diária em longo prazo da bacia do rio Iguaçu, usando dados de precipitação. Em (EVSUKOFF et al., 2012) uma abordagem multi-modelo é

proposta para o mesmo problema, a fim de conferir maior robustez ao modelo em função das incertezas das previsões meteorológicas utilizadas.

- **Neurologia**

Em (ZHANG, Q.; LEE, 2013) é proposto um arcabouço baseado em redes neuro-fuzzy recorrentes para analisar a dinâmica temporal de estímulos emocionais utilizando dados obtidos a partir de eletroencefalogramas e informações visuais.

- **Petróleo**

A previsão do preço do petróleo é abordada em (HU, J. W. S.; HU; LIN, 2012), que destaca o maior desempenho das redes recorrentes neuro-fuzzy sobre as redes neurais com arquiteturas direta e recorrente.

- **Robótica**

Em (CHEN, C.; INOUE; SHIBATA, 2011) investiga-se a identificação de sistemas robóticos que simulam o movimento corporal extremamente rápido de jogadores de golfe no momento da tacada, e ressaltam que as redes neuro-fuzzy dinâmicas recorrentes apresentam bons resultados na identificação deste tipo de sistema. Os autores em (WAI; LIU; LIN, 2011) descrevem o projeto de um modelo para o controle robusto de acompanhamento de trajetória de um robô móvel usando uma rede neural dinâmica fuzzy recorrente, baseada em redes de Petri. Em (XU; SONG; LI, 2011), é apresentada a implementação de um sistema de controle baseado em redes neurais fuzzy dinâmicas, recorrentes e evolucionárias, para robôs usados na reabilitação de membros superiores. Os autores em (CHEN, C.; RICHARDSON, 2012) propõem um modelo para desvio de obstáculos em ambientes desconhecidos, utilizado por robôs móveis, baseado em sistemas neuro-fuzzy recorrentes dinâmicos com memória curta.

- **Telecomunicações**

Os autores em (MASTOROCOSTAS, P.; HILAS, 2012) apresentam uma abordagem para predição de despesas com ligações telefônicas em um campus universitário, utilizando uma rede neuro-fuzzy localmente recorrente.

- **Veículos autônomos**

Os autores em (ZHANG, L.-J. et al., 2011) projetam um observador baseado em uma rede neuro-fuzzy dinâmica recorrente, que é utilizado para estimar os estados da

dinâmica de mergulho no problema de controle de profundidade de veículos autônomos subaquáticos, em que não há informações sobre os estados. Em (MON, Y. J.; LIN, C. M., 2012), os autores propõem uma lei de guiamento baseada em um controlador neuro-fuzzy recorrente e um controlador supervisório para o sistema de guiamento de um veículo autônomo subaquático.

A Figura 2.13 abaixo apresenta a distribuição das principais áreas de aplicação que foram discutidas.

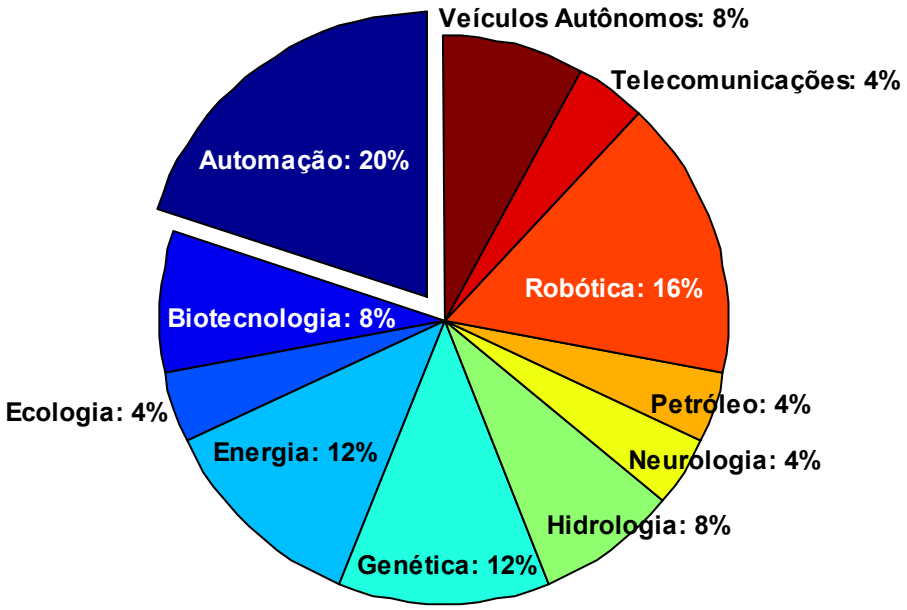


Figura 2.13. Distribuição das principais áreas de aplicação dos sistemas fuzzy recorrentes, de acordo com 25 pesquisas aplicadas, publicadas a partir de 2010.

2.2.1.2 Modelos

Os sistemas fuzzy recorrentes se destacam por apresentar um comportamento dinâmico baseado em memória, que é implementada por realimentação. São as conexões de realimentação que caracterizam a recorrência do modelo. Esta memória, geralmente, lida com a redução da dimensionalidade do problema, conferindo aos modelos recorrentes uma estrutura mais parcimoniosa. O tipo de estrutura e a maneira com que a recorrência é implementada são as características mais exploradas no desenvolvimento de novos modelos recorrentes. São, principalmente, essas características e a abordagem de identificação utilizada que distinguem os modelos entre si.

Os autores geralmente se valem de dois tipos diferentes de abordagem para descrever graficamente a estrutura dos sistemas fuzzy recorrentes. Uma delas se baseia em diagramas de bloco para representar as equações do espaço de estados, como ilustrado pela Figura 2.14. Neste tipo de abordagem, a recorrência do modelo é caracterizada pela realimentação da variável de estado. Em (GONZALEZ-OLVERA; TANG, 2007), os autores utilizam esse tipo de representação e propõem o modelo *CReNN* em que a função de transição de estados é representada por um sistema fuzzy dinâmico (recorrente), e a função de saída por um sistema fuzzy estático. Por sua vez, os autores em (GAMA et al., 2008) apresentam um modelo que utiliza apenas um único sistema fuzzy (recorrente), usado para modelar a função de transição de estados, e tratam a função de saída como uma combinação linear dos estados. Ambos os modelos são utilizados como simulador, no sentido de não utilizarem a saída observada do sistema na estrutura do modelo. O sistema fuzzy recorrente em (GONZALEZ-OLVERA; TANG, 2007) utiliza uma estrutura complicada no consequente das regras, cujos parâmetros são ajustados por um método baseado no gradiente, enquanto que em (GAMA et al., 2008) o consequente das regras é representado por um parâmetro constante, ajustado por um algoritmo genético simples.

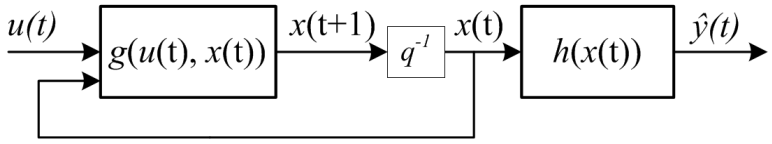


Figura 2.14. Diagrama de blocos de um modelo formulado por equações do espaço de estados, em que $\boldsymbol{u(t)}$, $\boldsymbol{x(t)}$ e $\boldsymbol{\hat{y}(t)}$ representam, respectivamente, a variável de entrada, de estado e de saída; e as funções $\boldsymbol{g(\cdot)}$ e $\boldsymbol{h(\cdot)}$ representam a função de transição de estado e de saída, respectivamente.

O outro tipo de abordagem baseia-se na representação da expansão funcional dos sistemas fuzzy na forma de rede. Neste caso, a recorrência do modelo é explorada em diversos níveis, ao contrário da abordagem baseada na formulação do espaço de estados, que geralmente explora a recorrência apenas através das variáveis de estado. Quando um sistema fuzzy é representado em forma de rede, esta rede é definida com pelo menos quatro camadas. A primeira delas é a camada de entrada; a segunda, de fuzzificação; a terceira, de ativação das regras e a quarta, de saída. Arquiteturas mais complexas podem ser definidas dependendo de como a estrutura do modelo é representada.

A recorrência neste tipo de abstração é geralmente explorada externa e/ou internamente. A recorrência externa pode ser obtida tanto pelas variáveis de saída quanto pelas variáveis de estado. Por sua vez, a recorrência interna é geralmente alcançada explorando-se as funções de pertinência e o valor de ativação das regras. Porém, embora raros, alguns modelos podem explorar tanto o consequente das regras, quanto variáveis internas. A recorrência interna pode ter um escopo local ou global. Caso o valor de um elemento seja realimentado apenas para ele mesmo, a recorrência é dita ser local; caso ele também seja realimentado para outros elementos, a recorrência é dita ser global.

A Figura 2.15 abaixo é um exemplo de modelo fuzzy recorrente em que a recorrência é do tipo externa, e obtida através da realimentação da variável de saída. Este tipo de abordagem é encontrado nos seguintes modelos fuzzy recorrentes: *RFNN* (LIN, W. M. et al., 2011); *ORFNN* (WANG, Y.-C.; CHIEN; LEE, 2008); e *GRFCMAC* (RODRIGUEZ; YU; MORENO-ARMENDARIZ, 2008). Nos modelos *RFNN* e *GRFCMAC* a camada de entrada da rede é definida como uma combinação linear entre as variáveis de entrada externas e a variável de saída que é realimentada, enquanto no modelo *ORFNN* a camada de entrada é definida diretamente pelas variáveis de entrada externas bem como pela saída realimentada.

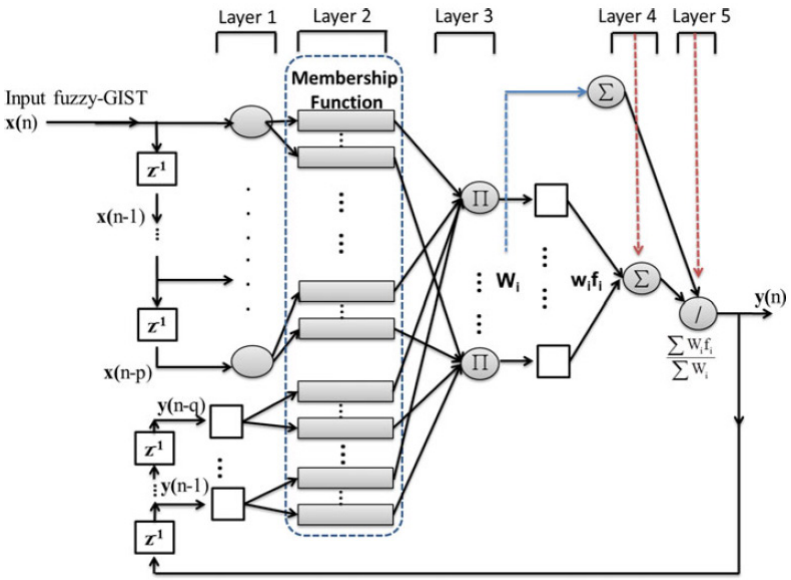


Figura 2.15. Exemplo de modelo fuzzy recorrente representado por uma rede em que a recorrência é caracterizada pela realimentação externa da saída do modelo (ZHANG, Q.; LEE, 2013).

Alguns modelos fuzzy recorrentes representados como rede possuem conexões de realimentação externas formadas a partir de elementos internos, como ilustrado na

Figura 2.16 a seguir. Exemplos de modelos fuzzy recorrentes com este tipo de estrutura são: *TRFN* (JUANG; CHANG, 2011); *SRNFN* (JUANG; LAI; TU, 2009); *EM-TRFN* (STAVRAKOUDIS et al., 2008); *RT2FNN-A* (LEE, C.-H. et al., 2008) e *CRFNN* (LIN, C.-J.; CHEN, 2006). Com exceção do modelo *RT2FNN-A,* que utiliza o mesmo tipo de função de pertinência tanto para os estados quanto para as variáveis de entrada, todos os demais modelos utilizam funções de pertinências diferentes para os estados e para as variáveis de entrada. Além disso, as funções de pertinência do modelo *RT2FNN-A* são intervalares, baseadas em conjuntos fuzzy tipo 2; enquanto os demais modelos utilizam funções de pertinência tradicionais, baseadas em conjunto fuzzy do tipo 1. Neste caso, para os estados, a função de pertinência utilizada é Sigmoidal, que possui uma abrangência global; enquanto que para as variáveis de entrada, a função de pertinência utilizada é Gaussiana, que possui um comportamento local.

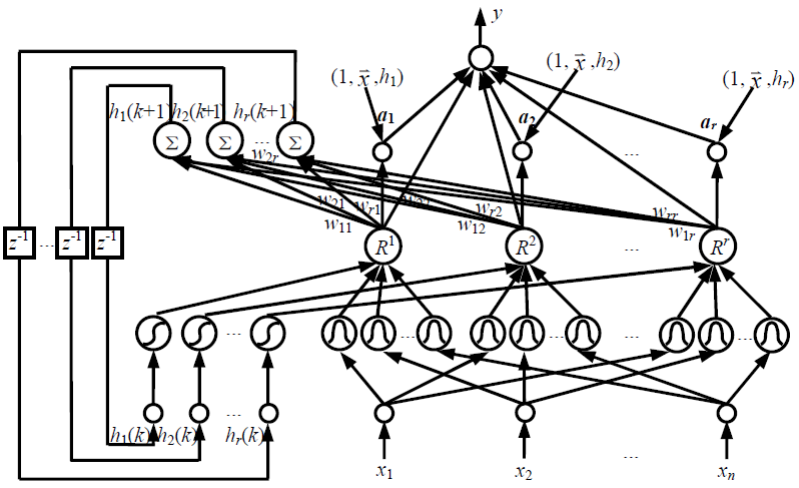


Figura 2.16. Exemplo de modelo fuzzy recorrente representado por uma rede em que a recorrência é caracterizada pela realimentação externa das variáveis de estado (TSAI; CHANG, 2012).

As regras desses modelos possuem duas conclusões no seu consequente. Uma delas define o valor da variável de saída, enquanto que a outra define o valor da variável de estado, que é utilizada para realimentar a rede. As variáveis de estado definem uma relação de dependência temporal entre o consequente e o antecedente das regras, e representam a memória do modelo. Embora cada uma das regras evolua ao mesmo tempo todos os estados, apenas o valor do estado associado à regra é utilizado no cálculo da sua saída.

Existem ainda situações em que os modelos fuzzy recorrentes possuem tanto conexões de realimentação externas quanto internas, como é o caso dos modelos

CRFCMAC (RODRIGUEZ et al., 2008) e *MRNFS* (KHANMIRZAEI; TESHNEHLAB; SHARIFI, 2010), cujas recorrências externa e interna são originadas, respectivamente, a partir da saída do modelo, e da função de pertinência. A recorrência desta última, definida apenas localmente.

A Figura 2.17 é um exemplo de modelo fuzzy com recorrência interna, caracterizada pela realimentação local da função de pertinência. Os seguintes modelos são representados por este tipo de estrutura: *IT2RFNN-AMF* (LIN, F.-J. et al., 2012) e *IT2RFNS-A* (LEE, C.-H.; CHANG, 2011), que utilizam funções de pertinência assimétricas do tipo 2. No primeiro, elas são Gaussianas e, no último, são triangulares; *NSRFNN* (LEE, C.-H.; LIN; YANG, 2012), que utiliza funções de pertinência baseadas em conjuntos fuzzy não estacionários Gaussianos, que lidam melhor com as incertezas presentes nos dados; e *CSRFNN* (CHANG, Y.-J.; HO, 2011), que utiliza funções de pertinências Gaussianas conjuntas, cujos centros e dispersões são valores complexos.

Os modelos *FLPRFNS* (LEE, C.-H.; LEE, 2012); *SRFNN* (MON, Y.-J.; LIN, C.-M., 2012); *RFBFNN* (CHIANG; CHU; JHOU, 2012); *EDRFNN* (XU et al., 2011); RFNFN (LIN, H.-Y. et al., 2012); *RFCMAC* (LIN, C.-J.; LEE, 2008) e *RCNFS* (LIN, C.-J.; CHEN, 2005) utilizam funções de pertinência Gaussianas tradicionais recorrentes. O modelo *FLPRFNS* possui uma camada baseada nas redes de Petri para eliminar regras redundantes, e o consequente das regras é uma rede neural de ligação funcional. O modelo *RCNFS* aplica operações compensatórias ao valor de ativação das regras a fim de otimizar o seu processo de inferência. O modelo RFCMAC utiliza uma abordagem baseada na memória associativa segundo o modelo cerebelar dos mamíferos, caracterizada por possuir rápida habilidade de aprendizado e boa capacidade de generalização local.

Enquanto o modelo *WRFNN* desenvolvido em (LIN, C.-J.; CHIN, 2004) é um sistema fuzzy TSK cujo consequente das regras é representado por uma rede neural Wavelet e a recorrência é obtida através de funções de pertinências Gaussianas recorrentes, o modelo *WRFNN* apresentado em (SONG; SHI, 2011) é um sistema fuzzy TSK de primeira ordem que utiliza funções de pertinência recorrentes do tipo Wavelet. Por sua vez, o modelo *DRFNS* (CHEN, C.; RICHARDSON, 2012) utiliza funções de pertinência Sigmoidais recorrentes.

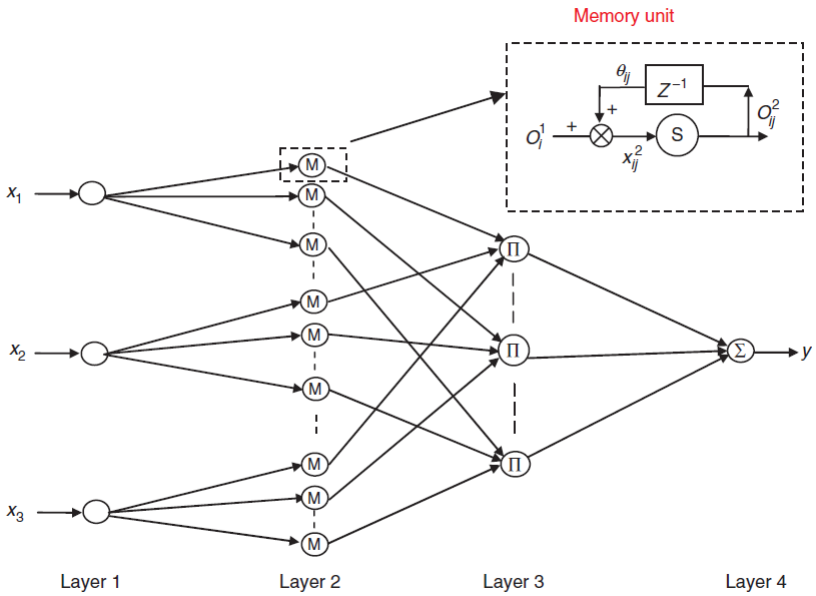


Figura 2.17. Exemplo de modelo fuzzy recorrente representado por uma rede em que a recorrência é caracterizada pela realimentação local das funções de pertinência (CHEN, C.; RICHARDSON, 2012).

A recorrência de todos os modelos que acabaram de ser apresentados, que é caracterizada pela realimentação local da função de pertinência, apresenta um escopo local. Isto é, o valor da saída da função de pertinência é realimentado diretamente para a sua entrada. Porém, no modelo *DRFNN* (HSU; CHENG, 2008), um escopo global é considerado. Ao invés de utilizar o valor da saída da função de pertinência para realimentá-la, o modelo utiliza o valor de ativação da regra para estabelecer a recorrência e calcular a função de pertinência. Portanto, a pertinência de uma determinada variável passa a ser influenciada não apenas pelo seu valor, mas também pelo grau de pertinência das outras variáveis, através do grau de ativação da regra.

Os modelos fuzzy recorrentes *MRFNN* (LIU; HUANG; JIA, 2007) e *MDRFN* (WU; ZHU; HUANG, 2011) possuem conexões de realimentação internas na sua estrutura, provenientes tanto da função de pertinência quanto do grau de ativação das regras.

Modelos fuzzy recorrentes que utilizam o valor de ativação das regras para estabelecer a conexão de realimentação podem apresentar escopo local ou global, dependendo de como o valor de ativação temporal das regras é calculado. Se ele utiliza informações apenas da própria regra, o seu escopo é local. Porém, se o cálculo envolver o valor de ativação das demais regras, seu escopo será global. Isto é, o valor de ativação da regra não é influenciado apenas por ela, mas também pelas outras regras. A Figura 2.18 representa um modelo fuzzy recorrente em que a recorrência é

estabelecida globalmente pelo valor de ativação das regras, enquanto que a Figura 2.19 representa um modelo em que a recorrência é estabelecida localmente pelo valor de ativação das regras. Alguns exemplos de modelos cuja estrutura é caracterizada pela realimentação global dos valores de ativação das regras são: *MRIT2NFS* (LIN, Y.-Y.; CHANG; PAL; et al., 2013); *IRSFNN* (LIN, Y.-Y.; CHANG; LIN, 2013) e *RNFN* (BALLINI; GOMIDE, 2010).

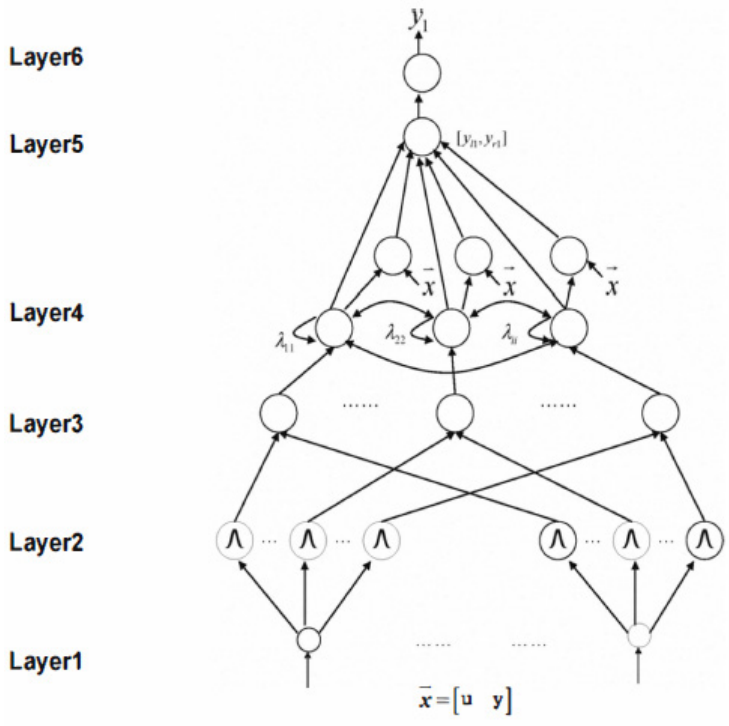


Figura 2.18. Exemplo de modelo fuzzy recorrente representado por uma rede em que a recorrência é caracterizada pela realimentação global dos valores de ativação das regras (LIN, Y.-Y. et al., 2010).

A recorrência dos modelos *MRIT2NFS* e *IRSFNN* é definida na camada de ativação temporal das regras, em que cada elemento corresponde a uma regra. A saída de cada um dos elementos nesta camada é calculada como uma combinação do próprio elemento (regra), dos demais elementos da mesma camada (recorrência global), bem como do elemento da camada anterior, que define o valor de ativação espacial da própria regra. Os coeficientes dos elementos desta combinação representam a taxa de compromisso entre os valores atuais e anteriores das variáveis de entrada para a saída do modelo. Enquanto o modelo *IRSFNN* utiliza funções de pertinência gaussianas tradicionais, o modelo *MRIT2NFS* utiliza funções de pertinência gaussianas intervalares do tipo 2.

Coeficientes intervalares também são utilizados no consequente das regras do modelo *MRIT2NFS*. Em ambos os modelos, o consequente é do tipo TSK. No modelo *IRSFNN,* porém, o consequente também pode ser representado por uma rede neural de ligação funcional. Neste caso, funções de base trigonométricas são utilizadas para criar uma região de decisão não linear em um espaço multidimensional, a fim de simplificar a aproximação do mapeamento não linear. Diferentemente dos modelos *MRIT2NFS* e *IRSFNN*, o modelo *RNFN* utiliza funções de pertinência triangulares ao invés de gaussianas. Além disso, a recorrência é definida de forma diferente, por elementos chamados de neurônios-T, cujas saídas são calculadas através de T-normas e S-normas. Outra característica do modelo *RNFN* é que o valor da sua saída é definido por uma rede neural de agregação com uma única camada que, ao contrário do modelo *IRSFNN*, não é representada no consequente das regras, e sim na camada de saída do modelo.

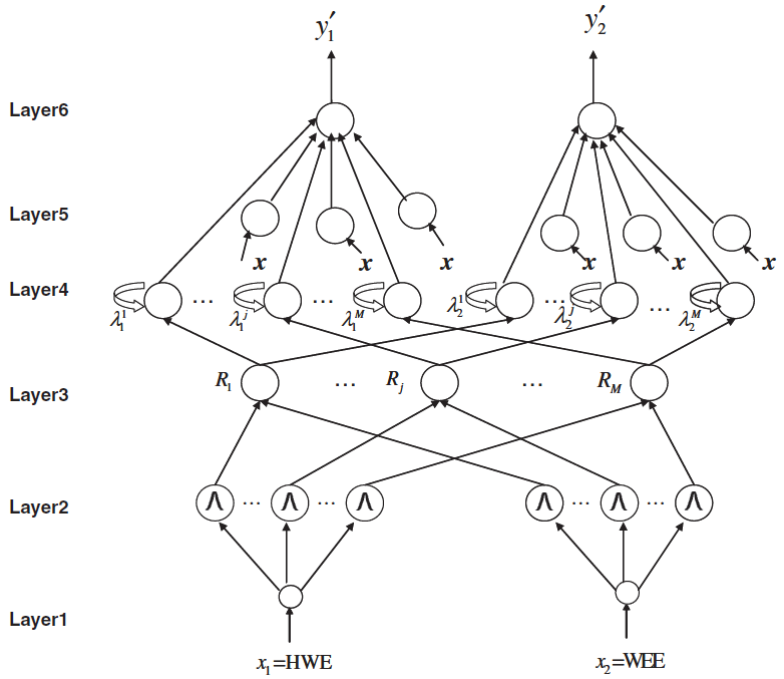


Figura 2.19. Exemplo de modelo fuzzy recorrente representado por uma rede em que a recorrência é caracterizada pela realimentação local dos valores de ativação das regras (TU; JUANG, 2012).

Alguns exemplos de modelos fuzzy recorrentes cuja estrutura é caracterizada pela realimentação dos valores de ativação das regras apenas localmente, como na Figura 2.19, são: *RSETI2FNN* (JUANG; HUANG; LIN, 2009; TU; JUANG, 2012); *RIFNN* (JUANG; LIN; HUANG, 2011); *LRFNN-SVR* (JUANG; HSIEH, 2010) e *RSEFNN-LF* (JUANG; LIN; TU, 2010). A recorrência desses modelos implementa uma memória local, definida

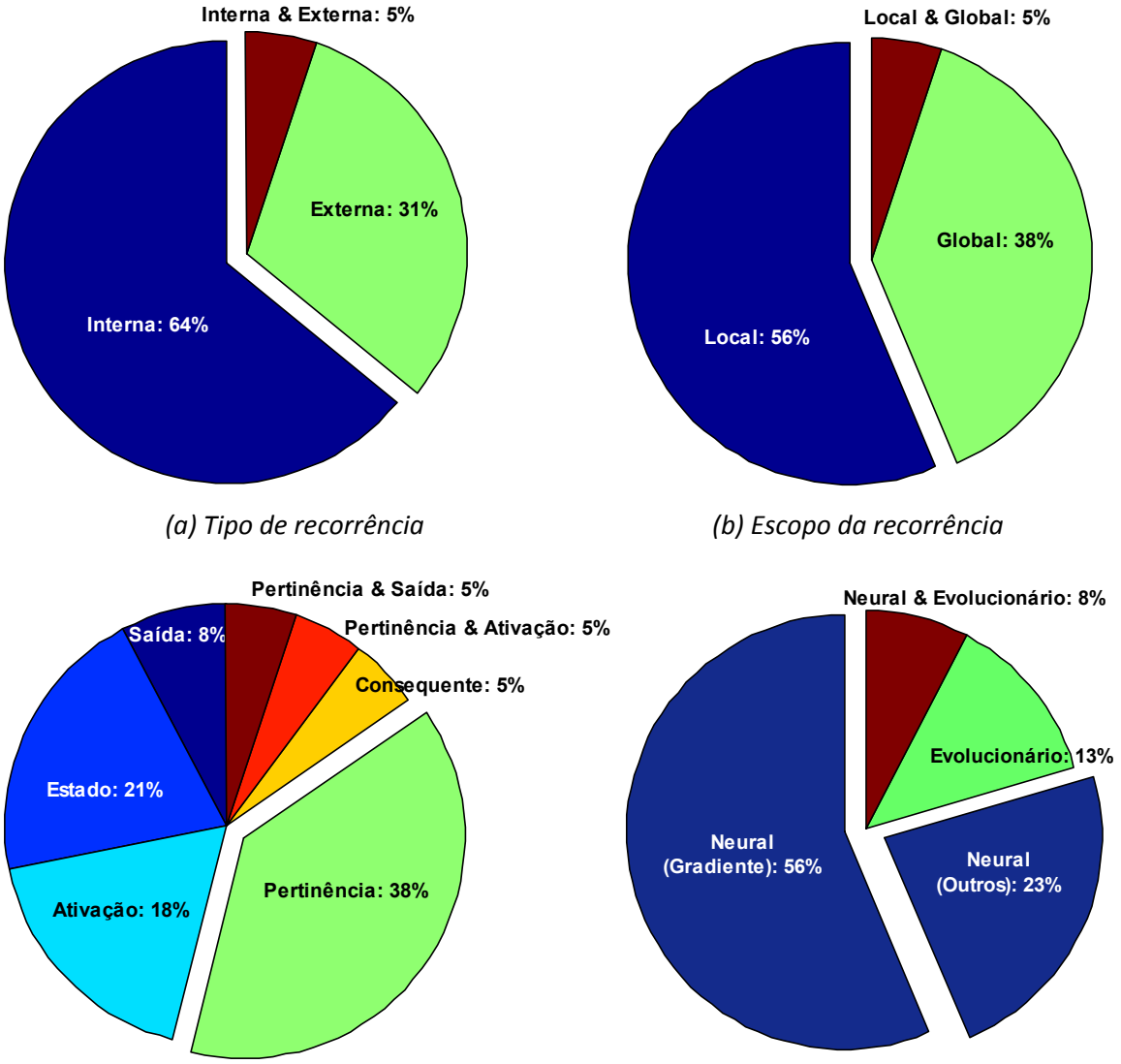
pelo grau de ativação temporal da regra, cujo cálculo utiliza tanto o grau atual de ativação espacial quanto o grau anterior de ativação temporal. Esses modelos utilizam apenas um parâmetro no cálculo do grau de ativação temporal da regra, que é definido no intervalo unitário e determina a taxa de contribuição entre os valores atuais e anteriores das variáveis de entrada para o cálculo da saída do modelo. Os modelos *RSETI2FNN* e *RIFNN* utilizam funções de pertinência gaussianas intervalares a fim de robustecer o modelo contra ruídos, enquanto os modelos *LRFNN-SVR* e *RSEFNN-LF* utilizam funções de pertinências gaussianas convencionais.

A recorrência dos modelos *LR-NFFS* (MASTOROCOSTAS, P.; HILAS, 2012) e *DBD-FNN* (MASTOROCOSTAS, P. A.; HILAS, 2009) é caracterizada pela presença de redes neurais recorrentes no consequente das regras. Neste caso, o antecedente das regras é considerado estático, e o seu consequente dinâmico. O modelo *T-SORNFN* (CHEN, C.-S., 2010) possui uma estrutura recorrente que é definida por elementos internos, assim como alguns dos modelos que possuem recorrência externa. Porém, esses elementos são localmente realimentados, ou seja, sua recorrência é interna, e não externa como aqueles.

Quanto à identificação dos modelos baseados em sistemas fuzzy recorrentes, dois métodos se destacam com relação ao ajuste dos seus parâmetros: métodos baseados no aprendizado neural e métodos baseados em algoritmos evolucionários. Existem, ainda, alguns métodos que combinam estas duas abordagens. A Tabela 2.3 apresenta um resumo das principais características dos modelos que representam o estado da arte, e que acabaram de ser descritos. Nesta tabela, os modelos são classificados tanto pelas características de sua recorrência quanto pelo tipo de aprendizado utilizado para ajustar seus parâmetros. As informações desta tabela são resumidas e quantificadas na Figura 2.20, que apresenta os gráficos de distribuição das principais características exploradas pelos 39 modelos considerados.

Tabela 2.3. Modelos fuzzy recorrentes que representam o estado da arte, classificados segundo a origem, o tipo e o escopo da recorrência, além do tipo de aprendizado. Os símbolos ☑ e ✓ indicam, respectivamente, se a recorrência é interna ou externa. Símbolos azuis indicam recorrência local, e símbolos vermelhos recorrência global. Os modelos sublinhados utilizam aprendizado evolucionário; aqueles em negrito utilizam aprendizado híbrido, enquanto os demais utilizam aprendizado neural.

Modelo	Referência	Origem da Recorrência				
		Saída	Estado	Ativação	Pertinência	Consequente
IRSFNN	(LIN, Y.-Y.; CHANG; LIN, 2013)			☑		
MRIT2NFS	(LIN, Y.-Y.; CHANG; PAL; et al., 2013)			☑		
DRFNS	(CHEN, C.; RICHARDSON, 2012)				☑	
FLPRFNS	(LEE, C.-H.; LEE, 2012)				☑	
IT2RFNN-AMF	(LIN, F.-J. et al., 2012)				☑	
LR-NFFS	(MASTOROCOSTAS, P.; HILAS, 2012)					✓
NSRFNN	(LEE, C.-H. et al., 2012)				☑	
RFBFNN	(CHIANG et al., 2012)				☑	
RFNFN	(LIN, H.-Y. et al., 2012)				☑	
SRFNN	(MON, Y.-J.; LIN, C.-M., 2012)				☑	
CSRFNN	(CHANG, Y.-J.; HO, 2011)				☑	
EDRFNN	(XU et al., 2011)				☑	
IT2RFNS-A	(LEE, C.-H.; CHANG, 2011)				☑	
MDRFN	(WU et al., 2011)			☑	☑	
RFNN	(LIN, W. M. et al., 2011)	✓				
RIFNN	(JUANG et al., 2011)			☑		
TRFN / TRFN	(JUANG, 2002) / (JUANG; CHANG, 2011)		✓			
WRFNN	(SONG; SHI, 2011)				☑	
LRFNN-SVR	(JUANG; HSIEH, 2010)			☑		
MRNFS	(KHANMIRZAEI et al., 2010)	✓			☑	
RNFN	(BALLINI; GOMIDE, 2010)			☑		
RSEFNN-LF	(JUANG et al., 2010)			☑		
T-SORNFN	(CHEN, C.-S., 2010)		☑			
DBD-FNN	(MASTOROCOSTAS, P. A.; HILAS, 2009)					✓
RSETI2FNN	(JUANG; HUANG; et al., 2009)			☑		
SRNFN	(JUANG; LAI; et al., 2009)		✓			
CRFCMAC	(RODRIGUEZ et al., 2008)	✓			☑	
DRFNN	(HSU; CHENG, 2008)				☑	
EM-TRFN	(STAVRAKOUDIS et al., 2008)		✓			
GRFCMAC	(RODRIGUEZ et al., 2008)	✓				
ORFNN	(WANG, Y.-C. et al., 2008)	✓				
RFCMAC	(LIN, C.-J.; LEE, 2008)				☑	
RFS-TSK	(GAMA et al., 2008)		✓			
RT2FNN-A	(LEE, C.-H. et al., 2008)		✓			
CReNN	(GONZALEZ-OLVERA; TANG, 2007)		✓			
MRFNN	(LIU et al., 2007)			☑	☑	
CRFNN	(LIN, C.-J.; CHEN, 2006)		✓			
RCNFS	(LIN, C.-J.; CHEN, 2005)				☑	
WRFNN	(LIN, C.-J.; CHIN, 2004)				☑	



(a) Tipo de recorrência *(b) Escopo da recorrência*

(c) Origem da recorrência *(d) Tipo de Aprendizado*

Figura 2.20. Distribuição dos 39 modelos apresentados na Tabela 2.3, de acordo com: (a) o tipo; (b) o escopo e (c) a origem da recorrência, bem como (d) o tipo de aprendizado, com destaque para as características mais exploradas.

Embora o tipo de aprendizado neural seja o mais utilizado, a grande maioria dos seus métodos baseia-se no gradiente (Figura 2.20-d), cuja derivação torna-se complexa, facilmente limitada a mínimos locais, e altamente dependente da estrutura do modelo. Para evitar esses potenciais problemas no desenvolvimento de modelos baseados em sistemas fuzzy recorrentes, alguns autores sugerem a utilização dos algoritmos evolucionários, visto que são métodos de otimização que não utilizam a informação de derivadas, são estocásticos e baseados em população. Exemplos de algoritmos evolucionários utilizados frequentemente no contexto de sistemas fuzzy recorrentes são os algoritmos genéticos (GA) (GAMA et al., 2008; STAVRAKOUDIS et

al., 2008; XU et al., 2011); enxame de partículas (PSO) (JUANG; CHANG, 2011; LEE, C.-H.; LEE, 2012; LIN, W. M. et al., 2011); colônia de formigas (ACO) (JUANG; CHANG, 2011); evolução diferencial (DE) (LIN, H.-Y. et al., 2012), enxame de abelhas (HBO) (KHANMIRZAEI et al., 2010), dentre outros. A próxima seção apresentará algumas abordagens evolucionárias desenvolvidas recentemente para identificação de sistemas fuzzy, baseadas exclusivamente no algoritmo de evolução diferencial.

2.2.2 Métodos de Identificação Baseados em Evolução Diferencial

A utilização de métodos evolucionários no processo de identificação de modelos fuzzy fornece aos sistemas fuzzy a capacidade de aprendizado e de adaptação que eles não possuem. Este tipo de abordagem híbrida, geralmente conhecida como sistemas fuzzy evolucionários, foi inicialmente utilizado no final da década de 80 a fim de solucionar o problema da seleção subjetiva das funções de pertinência de controladores fuzzy, que era baseada no conhecimento de especialistas (KARR; FREEMAN; MEREDITH, 1989). Segundo os autores deste trabalho, o controlador fuzzy obtido era mais eficiente quando as funções de pertinência eram ajustadas por algoritmos genéticos do que quando elas eram selecionadas manualmente.

O sucesso deste tipo de abordagem motivou o desenvolvimento de novas técnicas híbridas utilizando modelos fuzzy e métodos de otimização evolucionários, na maioria dos casos, os algoritmos genéticos (CORDÓN et al., 2004). Neste novo tipo de abordagem, os métodos de otimização evolucionários passaram a identificar não apenas os parâmetros como também a estrutura dos modelos fuzzy (FAZZOLARI et al., 2013; HERRERA, 2008).

A partir do final da década de 90, o sucesso demonstrado pelo algoritmo de evolução diferencial, em diversos problemas de otimização, motivou a sua utilização em conjunto com modelos fuzzy. Inicialmente, para otimizar as funções de pertinência de controladores fuzzy (CHANG, C. S.; XU; QUEK, 1999; CHEONG; LAI, 1999; SASTRY, K. K. N.; BEHERA; NAGRATH, 1999). Posteriormente, a evolução diferencial passou a ser utilizada para identificar tanto os parâmetros quanto a estrutura de modelos fuzzy, como será visto a seguir.

Geralmente, a identificação de sistemas baseados em regras fuzzy envolve os seguintes problemas (LJUNG, 1999; NELLES, 2001): escolha do tipo de modelo; determinação do número de regras; posicionamento dos conjuntos fuzzy no domínio de cada variável; e o ajuste dos parâmetros livres. Os três primeiros problemas estão associados à identificação da estrutura, enquanto que o último diz respeito a identificação dos parâmetros. Nos sistemas fuzzy evolucionários, esses problemas são tratados como um problema de otimização, e os métodos evolucionários passam a ser responsáveis por uma ou mais das seguintes tarefas, dependendo do tipo de abordagem utilizado (HERRERA, 2008): otimizar os parâmetros das funções de pertinência; otimizar os parâmetros livres do modelo; gerar uma base de regras ótima; selecionar um subconjunto ótimo de regras a partir de uma determinada base de regras.

Com relação ao tipo de identificação, a maioria das abordagens desenvolvidas recentemente tem utilizado a evolução diferencial para otimizar apenas os parâmetros do modelo. Em (EFTEKHARI et al., 2008), por exemplo, os autores utilizam o algoritmo de evolução diferencial apenas para refinar os parâmetros das funções de pertinência, definidos previamente por meio de um método de agrupamento subtrativo. Por outro lado, em (OH et al., 2012), os autores utilizam o algoritmo de evolução diferencial para otimizar unicamente os parâmetros livres de um controlador fuzzy. Em ambos os casos, a versão canônica DE/*rand*/1/bin é utilizada, e cada vetor representa uma base de regras completa.

Na maioria dos casos, porém, o algoritmo de evolução diferencial é utilizado para otimizar tanto os parâmetros das funções de pertinência quanto os parâmetros livres dos modelos fuzzy, simultaneamente (ALIEV, RAFIK A. et al., 2011; CHEN, C.-H., 2013; CHEN, C.-H.; LIN; LIN, 2009; HAN; LIN; CHANG, 2013; LIN, H.-Y. et al., 2012; LU et al., 2012; WANG, R. et al., 2012). Com exceção de (ALIEV, RAFIK A. et al., 2011), que utiliza a versão canônica DE/rand/1/bin, todos os demais trabalhos sugerem a utilização de versões com algum tipo de modificação na estrutura original do algoritmo de evolução diferencial, a fim de tentar melhorar alguma(s) de suas características. Em (CHEN, C.-H., 2013), (HAN et al., 2013), (LU et al., 2012) e (CHEN, C.-H. et al., 2009), as modificações estão relacionadas ao operador de mutação, enquanto que em (WANG,

R. et al., 2012) e (LIN, H.-Y. et al., 2012) elas são caracterizadas pela hibridização com outros métodos evolucionários.

Em (CHEN, C.-H., 2013), o autor propõe a utilização de uma versão modificada, denominada *DE-MAS*, baseada em um esquema de mutação adaptativo que utiliza um mecanismo de roleta para selecionar uma das variações: DE/*rand*/1; DE/*best*/1 ou DE/*target-to-best*/1, a fim de melhorar a habilidade de busca do algoritmo. Em (HAN et al., 2013), a versão *DELI*, baseada na variação DE/*target-to-best*/1, também sugere modificações no operador de mutação, que passa a explorar a vizinhança dos vetores, a fim de melhorar a capacidade de busca local do algoritmo. Neste caso, o fator de escala é configurado auto adaptativamente, de acordo com a estimativa da probabilidade de sucesso da população. Em (LU et al., 2012), os autores propõem a versão *MDE*, baseada na variação DE/*best*/1, em que um termo que considera o valor de aptidão do elemento alvo é adicionado ao vetor diferencial, buscando evitar a convergência prematura das soluções. Em (CHEN, C.-H. et al., 2009), a versão *MODE* é sugerida, cuja modificação também é implementada no operador de mutação. Porém, neste caso, utilizando um algoritmo de auto agrupamento, a fim de manter a diversidade da população e aumentar sua capacidade de busca.

Em (WANG, R. et al., 2012), a versão chamada *DEACS*, baseada na variação DE/rand/1, é hibridizada com o algoritmo de colônia de formigas para ajustar de forma auto adaptativa os parâmetros de mutação e recombinação, de acordo com o desempenho da população. Em (LIN, H.-Y. et al., 2012), a versão *CMDE* é caracterizada pela hibridização com um algoritmo cultural, a fim de acelerar a convergência e melhorar o desempenho do algoritmo.

Embora sejam um pouco menos frequentes, algumas abordagens adotam uma estratégia em que o algoritmo de evolução diferencial é empregado para identificar tanto a estrutura quanto os parâmetros de modelos fuzzy (CHEN, C.-H.; YANG, 2013; LAI, J. C. Y. et al., 2013; SINGH; KUMAR; PAUL, 2008; SU, H.; YANG, 2011; SU, M.-T. et al., 2011). Todas essas abordagens citadas também adotam algum tipo de modificação na versão original do algoritmo, a fim de melhorá-lo de alguma forma.

Em (SU, H.; YANG, 2011) e (SINGH et al., 2008) os autores sugerem um esquema de codificação híbrida binária-real para representar, em um só vetor, as

variáveis de projeto que representam os elementos da estrutura e dos parâmetros do modelo. Este esquema possibilita lidar com a otimização binária e contínua ao mesmo tempo. Enquanto a versão *DE/QDE* desenvolvida em (SU, H.; YANG, 2011) utiliza uma abordagem que seleciona um subconjunto ótimo de regras, a partir de uma base de regras criada previamente, a abordagem baseada na versão *vlx-DE*, desenvolvida em (SINGH et al., 2008), gera uma base de regras completa. Nesta abordagem, a evolução simultânea da estrutura e dos parâmetros é realizada utilizando-se vetores de tamanhos diferentes, e um novo operador de recombinação é proposto para lidar com esta característica particular. Inicialmente, o algoritmo evolui tanto a estrutura quanto os parâmetros. Após a estrutura convergir para uma região ótima do espaço de busca, o algoritmo continua com a otimização apenas dos parâmetros do modelo. A versão *vlx-DE* utiliza uma função objetivo composta, diferente das demais abordagens que foram citadas, que consideram basicamente o desempenho do modelo. Nesta versão, a função objetivo é composta por dois termos, que consideram tanto o desempenho quanto a complexidade do modelo, representada pelo número de regras.

Ao contrário das demais abordagens descritas, em que cada vetor representa uma base de regras completa, nas versões *KCoDE* (CHEN, C.-H.; YANG, 2013) e *RSMODE* (SU, M.-T. et al., 2011) cada vetor codifica uma única regra. Ambas as versões adotam uma estratégia multipopulação que evolui cada uma das regras separadamente, a fim de acelerar a busca e melhorar o desempenho do modelo. A versão *KCoDE* implementa um mecanismo de coevolução cooperativa baseada no espaço de crenças dos algoritmos culturais, através da decomposição do sistema fuzzy em subpopulações, cuja evolução emprega tanto os operadores da evolução diferencial quanto dos algoritmos culturais. Na versão *RSMODE*, a identificação da estrutura é feita apenas na primeira geração. Nas demais, os parâmetros são identificados.

A estratégia adotada pela versão *DWM-DE* (LAI, J. C. Y. et al., 2013) distingue-se das demais por utilizar não apenas dados de treinamento, mas também de validação, no processo de identificação simultânea da estrutura e dos parâmetros do modelo. Estes dois conjuntos de dados diferentes são utilizados a afim de se evitar o sobreajuste dos parâmetros do modelo. Outra característica desta versão é utilizar

dois objetivos simultaneamente no processo de otimização. Cada indivíduo é avaliado ao mesmo tempo com relação às métricas de sensibilidade e sensitividade do modelo. Neste caso, o operador de seleção só permite a sobrevivência das soluções que não sejam dominadas por nenhuma outra. Adicionalmente, esta versão implementa novos operadores de mutação e recombinação que utilizam funções wavelet. A Tabela 2.4 abaixo apresenta um resumo das principais características das abordagens que foram apresentadas nesta seção, as quais representam o estado da arte.

Tabela 2.4. Algumas abordagens recentes de identificação de sistemas fuzzy baseadas em evolução diferencial. Legenda: P = Parâmetros; E = Estrutura; R = Real; B = Binária; D = Desempenho; C = Complexidade; T = Treinamento e V = Validação.

Versão	Referência	Identificação	Codificação	Função Objetivo	Dados
DE-MAS	(CHEN, C.-H., 2013)	P	R	D	T
DELI	(HAN et al., 2013)	P	R	D	T
DWM-DE	(LAI, J. C. Y. et al., 2013)	E + P	R	D	T + V
KCoDE	(CHEN, C.-H.; YANG, 2013)	E + P	R	D	T
CMDE	(LIN, H.-Y. et al., 2012)	P	R	D	T
DE	(OH et al., 2012)	P	R	D	T
DEACS	(WANG, R. et al., 2012)	P	R	D	T
MDE	(LU et al., 2012)	P	R	D	T
DE	(ALIEV, RAFIK A. et al., 2011)	P	R	D	T
DE/QDE	(SU, H.; YANG, 2011)	E + P	B + R	D	T
RSMODE	(SU, M.-T. et al., 2011)	E + P	R	D	T
MODE	(CHEN, C.-H. et al., 2009)	P	R	D	T
DE	(EFTEKHARI et al., 2008)	P	R	D	T
vIX-DE	(SINGH et al., 2008)	E + P	B + R	D + C	T

2.3 Resumo

Este capítulo apresentou a revisão da literatura, cujo objetivo foi introduzir os conceitos básicos a partir dos quais o trabalho foi desenvolvido, bem como o estado da arte, de modo a contextualizar a abordagem proposta e posicioná-la diante dos trabalhos correlatos desenvolvidos recentemente.

Na primeira parte do capítulo, foram introduzidos os conceitos básicos e a formulação matemática referente: ao problema de identificação de sistemas, aos sistemas fuzzy recorrentes, e à evolução diferencial. Juntos, estes três conceitos compõem o alicerce sobre o qual a pesquisa foi realizada.

A segunda parte deste capítulo discorreu sobre o estado da arte dos modelos baseados em sistemas fuzzy recorrentes e dos métodos de identificação baseados no algoritmo de evolução diferencial. Quanto aos modelos baseados em sistemas fuzzy recorrentes, o objetivo foi apresentar as principais áreas de aplicação e os principais modelos utilizados recentemente, explorando principalmente as características estruturais relacionadas à sua recorrência. Quanto aos métodos de identificação, o objetivo foi apresentar as abordagens mais atuais, explorando as principais características relacionadas tanto ao algoritmo quanto à metodologia de identificação considerada.

No capítulo a seguir, a abordagem proposta será apresentada em detalhe. Inicialmente, o modelo de referência será introduzido, e sua formulação matemática apresentada. Em seguida, os modelos que foram desenvolvidos serão formulados. A principal característica desses modelos é a incorporação de uma conexão de feedback adicional à sua estrutura.

A fim de lidar com o aumento da complexidade do espaço de busca caracterizado pelas novas estruturas propostas, uma nova metodologia de identificação simultânea da estrutura e dos parâmetros dos novos modelos foram propostas e serão apresentadas na segunda parte do próximo capítulo. Nesta nova metodologia, duas instâncias do algoritmo de evolução diferencial serão utilizadas de forma hierárquica para selecionar a estrutura e estimar os parâmetros do modelo.

Capítulo 3

3 Modelos com Duplo Feedback e seu Método de Identificação

A característica fundamental dos sistemas fuzzy recorrentes são as conexões de realimentação na sua estrutura. São exatamente essas conexões que implementam a memória do modelo e garantem aos sistemas fuzzy recorrentes uma estrutura mais compacta, bem como a habilidade de processar sistemas dinâmicos de ordem desconhecida. Existem duas vias principais a serem exploradas no desenvolvimento de novos modelos baseados em sistemas fuzzy recorrentes. Em uma delas, a estrutura do modelo é explorada, principalmente com relação à sua memória. Na outra, explora-se o algoritmo de ajuste do modelo. Ambas as vias são consideradas neste trabalho, e exploradas nas duas próximas seções.

Como mencionado na seção 1.1, a abordagem proposta neste trabalho representa uma extensão do trabalho desenvolvido por Gama et al. (2008). Esta extensão explora o modelo desenvolvido por eles e sugere um novo método de identificação. Enquanto o modelo original considera apenas o feedback interno a partir dos estados, como pode ser observado na Figura 3.1, feedbacks externos também são considerados no desenvolvimento de novos modelos, cuja estrutura passa a ter duas conexões de feedback – uma interna ao modelo e outra externa a ele. Esses feedbacks externos são obtidos a partir da saída observada do sistema e/ou da saída estimada pelo modelo, explicita ou implicitamente.

A metodologia de identificação apresentada em (GAMA et al., 2008) utiliza um algoritmo genético simples para identificar apenas os parâmetros do modelo, e realiza uma busca exaustiva para identificar a estrutura ótima do modelo. O método de identificação proposto neste trabalho, por sua vez, utiliza o algoritmo de evolução diferencial para identificar simultaneamente a estrutura do modelo e seus respectivos parâmetros. Este método é apresentado em detalhes na seção 3.2. Antes disso, porém, a estrutura e a formulação dos novos modelos são discutidas.

3.1 Formulação dos Modelos com Duplo Feedback

3.1.1 Modelo de Referência

Sistemas dinâmicos não lineares, discretos no tempo, podem ser convenientemente representados através das equações do espaço de estados:

$$\begin{aligned} \boldsymbol{x}(t+1) &= g\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \\ y(t) &= h\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \end{aligned} \tag{3.1}$$

em que os vetores $\boldsymbol{x}(\mathrm{t})$ e $\boldsymbol{u}(\mathrm{t})$ representam, respectivamente, as variáveis de estado e de entrada; $g(\cdot)$ e $h(\cdot)$ são as funções de transição de estados e de saída, respectivamente; e $y(\mathrm{t})$ representa a saída do sistema. Como geralmente as entradas $\boldsymbol{u}(\mathrm{t})$ não interferem diretamente na saída $y(\mathrm{t})$, a função de saída torna-se dependente apenas dos seus estados $\boldsymbol{x}(\mathrm{t})$. Neste caso, o sistema passa a ser representado por:

$$\begin{aligned} \boldsymbol{x}(t+1) &= g\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \\ y(t) &= h\big(\boldsymbol{x}(t)\big) \end{aligned} \tag{3.2}$$

O modelo introduzido por Gama *et al.* (2008) utiliza este tipo de formulação, que é ilustrada pela Figura 3.1, em que a função $g(\cdot)$ é modelada por um sistema fuzzy recorrente, e a função $h(\cdot)$ linear nos estados. À luz da formulação que foi introduzida na seção 2.1.2.2, o resultado das equações definidas em **(3.2)** é dado por:

$$\begin{aligned} \boldsymbol{x}_1(t+1) &= \boldsymbol{\omega}(t)\boldsymbol{\alpha} \\ \hat{y}(t) &= \boldsymbol{x}(t)\boldsymbol{\lambda} \end{aligned} \tag{3.3}$$

em que $\boldsymbol{\lambda}$ é o vetor com os d_x coeficientes da combinação linear, e os vetores $\boldsymbol{\omega}(t)$ e $\boldsymbol{\alpha}$ são os mesmos definidos em (2.17). O primeiro representa o antecedente das M regras enquanto o último representa o consequente delas. Este modelo possui, portanto, $M + d_x$ graus de liberdade.

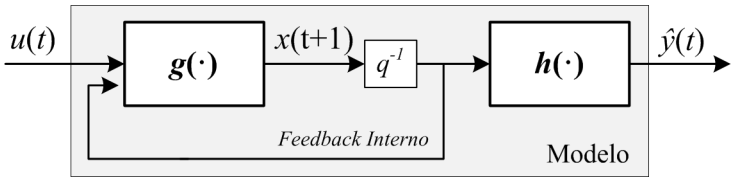


Figura 3.1. Estrutura bloco-esquemática do modelo de simulação *RFS-TSK,* desenvolvido por Gama et al. (2008), em que o sistema fuzzy recorrente modela apenas a função de transição de estados $\boldsymbol{g}(\cdot)$.

3.1.2 Modelos com Duplo Feedback

Embora os três modelos propostos neste trabalho sejam diferentes entre si, eles compartilham de certas semelhanças estruturais que lhes permitem ser representados por uma mesma formulação matemática. Basicamente, a diferença estrutural entre eles e o modelo desenvolvido por Gama et al. (2008) é a utilização de uma conexão de feedback adicional e de operadores de defasagem ajustáveis. Por outro lado, a distinção entre eles se dá apenas no tipo de feedback utilizado. Portanto, eles podem ser descritos utilizando-se a seguinte formulação:

$$\begin{aligned} \boldsymbol{x}(t+1) &= g\left(\boldsymbol{x}(t), \boldsymbol{u}(t-\boldsymbol{\Delta_u}), \xi\left(t-\Delta_\xi\right)\right) \\ \hat{y}(t) &= h\left(\boldsymbol{x}(t)\right) \end{aligned} \qquad \textbf{(3.4)}$$

em que $\boldsymbol{\Delta_u} = [\Delta_1, \ldots, \Delta_{d_u}]$ e Δ_ξ definem, respectivamente, o atraso de cada uma das d_u variáveis de entrada, e o atraso da variável $\xi \in \{y, \hat{y}, e\}$, que representa a conexão adicional de feedback considerada.

Os operadores de defasagem considerados na formulação acima são parametrizados e particularmente úteis quando o modelo está sendo utilizado para processar sistemas dinâmicos com atraso. Diferentemente dos modelos com dinâmica externa, que consideram uma grande quantidade de atrasos em cada variável, o operador de defasagem proposto considera apenas o instante temporal que mais influencia o comportamento do sistema, pois a estrutura do modelo é recorrente, e captura a dinâmica do sistema internamente, através das variáveis de estado.

A variável ξ definida na equação (3.4) representa uma das três possíveis conexões de feedback, como pode ser observado na Figura 3.2 abaixo, em que o operador $\underline{\vee}$ indica que apenas um dos sinais (setas azuis) pode ser utilizado como feedback para o modelo (setas vermelhas). A escolha do sinal é dependente do problema, e definida *a priori*. A seção 5.3 apresenta e compara o desempenho dos modelos em função de cada um desses sinais.

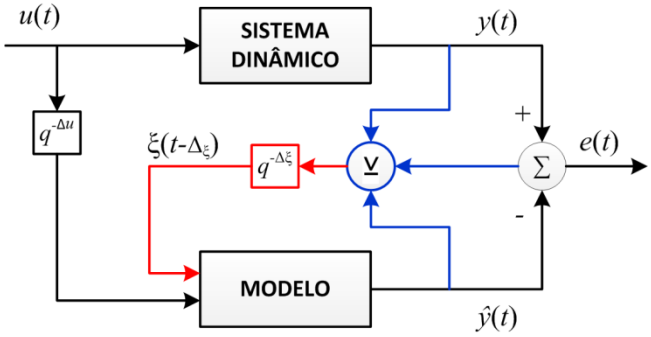


Figura 3.2. Diagrama de blocos da abordagem proposta para as novas estruturas.

As Figuras 3.3, 3.4 e 3.5, a seguir, apresentam a estrutura de cada um dos novos modelos propostos. Na Figura 3.3, o feedback externo é obtido explicitamente do próprio modelo, através de sua saída prevista. Este feedback adiciona uma nova camada de recorrência ao modelo. Na Figura 3.4, o feedback externo também é obtido explicitamente. Neste caso, porém, pela saída observada do sistema. Finalmente, a Figura 3.5 apresenta a estrutura cujo feedback externo é implicitamente obtido através do erro de predição, que depende tanto da saída observada do sistema quanto da saída estimada pelo modelo. Nestes dois últimos casos, o modelo deixa de ser um modelo de simulação pura e passa a operar em modo predição, visto que a saída observada do sistema passa a fazer parte da estrutura do modelo.

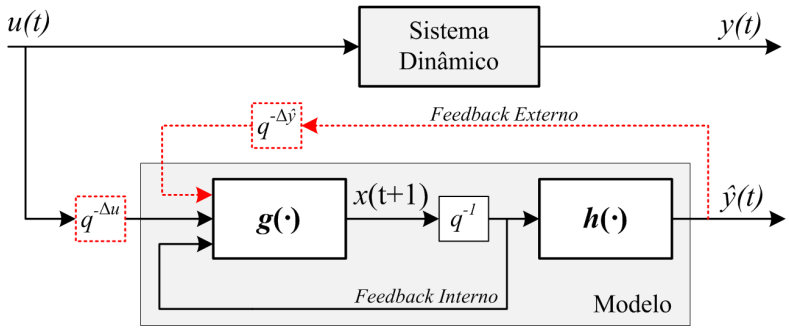


Figura 3.3. Diagrama de blocos do modelo de simulação proposto, cuja estrutura é caracterizada pela inclusão da saída estimada do modelo, que gera uma nova camada de recorrência, e de operadores de defasagem ajustáveis. Os elementos que representam extensões ao modelo original estão destacados pelas linhas pontilhadas em vermelho.

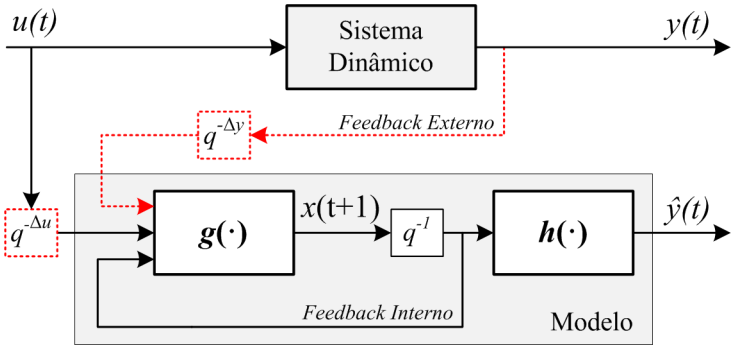


Figura 3.4. Diagrama de blocos do primeiro modelo de predição proposto, cuja estrutura é caracterizada pela inclusão da saída observada do sistema e de operadores de defasagem ajustáveis. Os elementos que representam extensões ao modelo original estão destacados pelas linhas pontilhadas em vermelho.

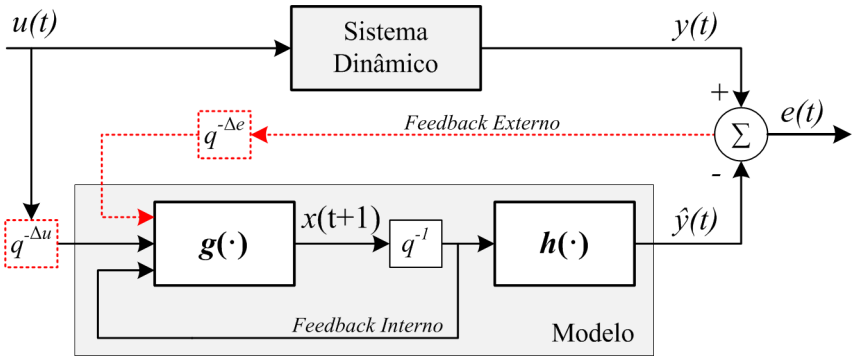


Figura 3.5. Diagrama de blocos do segundo modelo de predição proposto, cuja estrutura é caracterizada pela inclusão do erro de predição e de operadores de defasagem ajustáveis. Os elementos que representam extensões ao modelo original estão destacados pelas linhas pontilhadas em vermelho.

Se por um lado a utilização da conexão de feedback adiciona flexibilidade ao modelo, por outro ele aumenta a sua complexidade, visto que em uma base de regras completa e com particionamento regular todas as variáveis são consideradas nas premissas das regras, e o número de regras M cresce, em progressão geométrica, em função do número de variáveis e do número de protótipos definidos no domínio de cada uma delas.

As regras que descrevem o sistema fuzzy recorrente que representa a função de transição de estados do modelo formulado em (3.2) é escrita como:

$$R_k: \textbf{SE } \boldsymbol{x}(t) \text{ é } B_k \textbf{ E } \boldsymbol{u}(t-\boldsymbol{\Delta_u}) \text{ é } C_k \textbf{ E } \xi(t-\Delta_\xi) \text{ é } D_k \textbf{ ENTÃO } \boldsymbol{x}_1(t+1) \text{ é } \alpha_k \quad \textbf{(3.5)}$$

em que B_k, C_k e D_k são os rótulos linguísticos associados aos conjuntos fuzzy multidimensionais das variáveis de estado e entrada, e ao conjunto fuzzy unidimensional da variável de feedback, respectivamente. O valor de ativação das regras é calculado utilizando-se a mesma formulação em (2.19):

$$\boldsymbol{\omega}(t) = \boldsymbol{\mu_B}\big(\boldsymbol{x}(t)\big) \otimes \boldsymbol{\mu_C}\big(\boldsymbol{u}(t - \boldsymbol{\Delta_u})\big) \otimes \boldsymbol{\mu_D}\left(\xi\big(t - \Delta_\xi\big)\right) \quad \textbf{(3.6)}$$

em que $\boldsymbol{\mu_B}\big(\boldsymbol{x}(t)\big)$ e $\boldsymbol{\mu_C}\big(\boldsymbol{u}(t - \Delta_u)\big)$ são os vetores de pertinências das variáveis de estado e de entrada, respectivamente; e $\boldsymbol{\mu_D}\left(\xi\big(t - \Delta_\xi\big)\right)$ é o vetor de pertinências da variável de feedback. Portanto, todas as combinações dos conjuntos fuzzy das variáveis que aparecem nas premissas das regras são consideradas e, consequentemente, todos os dados são cobertos pela base de regras, cuja quantidade de regras é definida como:

$$M = p_x^{d_x} \times p_\xi \times \prod_{i=1..d_u} p_{u_i} \quad \textbf{(3.7)}$$

em que d_x e d_u são os números de estados e de variáveis de entrada; p_x é o número de conjuntos fuzzy no domínio das variáveis de estado; p_ξ é o número de conjuntos fuzzy no domínio da variável de feedback e p_{u_i} é o número de conjuntos fuzzy da i-ésima variável de entrada. Como todas as variáveis do vetor de estados representam uma mesma informação, elas podem utilizar o mesmo particionamento.

O resultado da aproximação feita pelo sistema fuzzy recorrente é calculado como em (2.17):

$$\boldsymbol{x}_1(t+1) = \boldsymbol{\omega}(t)\boldsymbol{\alpha} \quad \textbf{(3.8)}$$

tal que o modelo formulado em (3.4) também é obtido como em (3.3). Neste caso, porém, o modelo possui $M + d_x + d_u + 1$ graus de liberdade, em que os $n_u + 1$ parâmetros adicionais são referentes aos operadores de defasagem ajustáveis das variáveis de entrada e da variável de feedback.

A identificação do modelo formulado em (3.4) restringe-se a estabelecer os valores das componentes dos vetores $\boldsymbol{\eta}$ e $\boldsymbol{\theta}$ definidos em (3.9), que representam, respectivamente, a estrutura e os parâmetros do modelo. O valor de d_u é definido *a priori*, visto que é dependente do problema e, portanto, não faz parte do vetor $\boldsymbol{\eta}$. O vetor $\boldsymbol{\lambda}$ representa os coeficientes da combinação linear definida em (3.3).

$$\begin{aligned} \boldsymbol{\eta} &= \left[d_x; p_x; p_\xi; \boldsymbol{p_u}; \Delta_\xi; \boldsymbol{\Delta_u}\right] \\ \boldsymbol{\theta} &= [\boldsymbol{\alpha}; \boldsymbol{\lambda}] \end{aligned} \quad \textbf{(3.9)}$$

Esses vetores devem ser ajustados de modo a minimizar a diferença entre o valor observado da saída do sistema e o valor estimado da saída do modelo. Portanto, a identificação do modelo passa a ser tratada como um problema de otimização, cuja

proposta de solução, baseada no algoritmo de evolução diferencial, é apresentada a seguir.

3.2 Identificação Simultânea da Estrutura e dos Parâmetros

Como mencionado na seção anterior, os modelos com duplo feedback que foram desenvolvidos são completamente definidos pelos vetores $\boldsymbol{\eta}$ e $\boldsymbol{\theta}$, que representam, respectivamente, a estrutura e os parâmetros livres do modelo. O método proposto para identificar esses modelos baseia-se numa abordagem hierárquica em dois níveis (Figura 3.6), cada qual associado a uma instância do algoritmo de evolução diferencial. No primeiro nível, a estrutura do modelo é considerada e, portanto, as componentes do vetor $\boldsymbol{\eta}$ são otimizadas nele. As componentes do vetor $\boldsymbol{\theta}$ são otimizadas no segundo nível, que lida com os parâmetros do modelo. Para cada estrutura avaliada, um novo conjunto de parâmetros é ajustado a fim de tentar encontrar a melhor combinação estrutura–parâmetros para um determinado problema. Esta abordagem hierárquica é apresentada em maiores detalhes nas duas próximas seções.

Figura 3.6. Abordagem hierárquica em dois níveis para determinar simultaneamente a estrutura e os parâmetros do modelo.

Como foi discutido na seção 2.1.1.2, a identificação de sistemas é dependente do conjunto de dados representativo do comportamento do sistema, visto que, para sistemas invariantes no tempo (como é o caso dos sistemas tratados neste trabalho), o

modelo identificado deve ser capaz de descrever o comportamento futuro do sistema com base nestes dados. Afinal, o modelo identificado representará o comportamento do sistema, no máximo, tão bem quanto os dados utilizados. Além disso, os dados utilizados para avaliar o desempenho do modelo devem ser diferentes dos dados utilizados para ajustá-lo. Porém, eles precisam fazer parte do mesmo domínio.

A abordagem proposta considera dois conjuntos de dados distintos para identificar o modelo. O primeiro deles, chamado de dados de treino, é utilizado para otimizar os parâmetros livres do modelo. O outro é utilizado para avaliar o desempenho do modelo, a fim de verificar sua capacidade de generalização.

Antes de lidar com qualquer problema de otimização é preciso definir as variáveis de decisão e a função objetivo. Dois problemas de otimização distintos estão sendo considerados pela abordagem proposta e, portanto, estas definições devem ser feitas para cada um deles. O problema de otimização da estrutura, identificado como ***OE1***, é abordado pelo algoritmo ***DE1***, e será discutido a seguir; enquanto que o problema de otimização dos parâmetros, identificado como ***OP2***, é abordado pelo algoritmo ***DE2***, e será discutido logo depois.

Ambos os algoritmos utilizados na identificação do modelo consideram um critério de parada mais sofisticado do que aquele apresentado na seção 2.1.3.2.3. Ele baseia-se não apenas no número de gerações, mas também nas estatísticas da população. Isto é, o processo evolucionário termina quando uma das seguintes condições é satisfeita: (a) o número máximo de gerações é alcançado; (b) a aptidão média da população alcança 95% da aptidão da melhor solução e a aptidão da melhor solução se mantém inalterada por 50 gerações consecutivas.

3.2.1 Identificação da Estrutura

O problema de identificação da estrutura de um modelo é, sem dúvida, bem mais complexo do que o problema de identificação dos seus parâmetros. Ambos estão sendo tratados como um problema de otimização, em que a identificação da estrutura representa um problema de otimização combinatória, e a identificação dos parâmetros, um problema de otimização contínua. A abordagem proposta para identificação da estrutura baseia-se na comparação entre modelos com diferentes

complexidades. Esta complexidade está relacionada com a flexibilidade estrutural do modelo, que é caracterizada pelo número de graus de liberdade que ele possui.

Como os parâmetros estimados estão relacionados ao conjunto de dados de treino, e este é finito, quanto maior o número de parâmetros do modelo, menos dados serão utilizados para ajustar cada parâmetro. Para um conjunto de dados com poucas observações, por exemplo, quanto menor o número de graus de liberdade do modelo, melhor os parâmetros serão estimados a partir destes dados. Por outro lado, porém, quanto menor o número de graus de liberdade do modelo, menos flexível é sua estrutura. A solução deste dilema é o problema central a ser resolvido pela identificação da estrutura, que deve ser capaz de encontrar um modelo com complexidade ótima. Segundo o princípio da parcimônia, a partir de um conjunto de modelos que representem um sistema com desempenho similar, o mais simples deverá ser escolhido e considerado como sendo o melhor. Portanto, a função objetivo utilizada pelo ***DE1*** (retângulo azul da Figura 3.7), levará em consideração tanto o desempenho quanto a complexidade do modelo na avaliação das estruturas.

Diferentes critérios podem ser utilizados para este fim (HABER; UNBEHAUEN, 1990; HONG et al., 2008) como, por exemplo, o critério de informação de Akaike (AIC); o erro final de predição (FPE); o critério de informação de Bayes (BIC); a validação cruzada generalizada (GCV), dentre outros. Estes critérios proporcionam uma medida da qualidade do modelo simulando a situação em que ele é testado com um conjunto de dados diferentes. Eles são compostos pela soma de dois termos, um dos quais caracteriza o erro do modelo, enquanto o outro caracteriza a sua complexidade. A diferença entre estes critérios está basicamente na maneira com que eles ponderam as duas parcelas. A minimização de ambos os termos busca identificar uma estrutura que proporcione, ao mesmo tempo, um modelo que seja parcimonioso e que possua boa capacidade de generalização. A métrica AIC foi escolhida neste trabalho devido à consistência dos seus resultados em algumas simulações computacionais preliminares, que foram empreendidos a fim de avaliar o comportamento dos critérios descritos acima, e eleger aquele que seria utilizado. Esta métrica é definida como:

$$AIC = N \ln\big(J(\boldsymbol{\theta})\big) + 2|\boldsymbol{\theta}| \qquad \textbf{(3.10)}$$

em que N indica o número de observações no conjunto de dados utilizado para

os parâmetros do modelo, definidos em $\boldsymbol{\theta}$; $J(\cdot)$ é a função de custo (MSE) definida em (2.9), e $|\cdot|$ indica o número de componentes do vetor.

O processo de identificação da estrutura começa com a codificação das variáveis de projeto, que representam as componentes do vetor $\boldsymbol{\eta}$. Este vetor define o número de estados; a quantidade de conjuntos fuzzy no domínio de cada variável considerada no antecedente das regras, além dos valores de atraso das variáveis de entrada e de feedback. Neste caso, as variáveis de projeto representam valores inteiros que podem assumir apenas uma quantidade finita de valores discretos, definidos *a priori*, caracterizando a natureza combinatória da identificação da estrutura.

Visto que o algoritmo de evolução diferencial codifica todos os seus vetores internamente como números de ponto flutuante, independente do tipo das variáveis de projeto, a estratégia adotada foi arredondar os valores contínuos dos vetores para o inteiro mais próximo. Porém, para que não fosse necessária nenhuma mudança interna na estrutura do algoritmo, principalmente com relação aos seus operadores, este arredondamento só é realizado dentro na função objetivo, no momento em que os valores dos vetores são recuperados para definir a estrutura do modelo. Embora esta seja uma estratégia bastante simples, ela mostrou-se completamente satisfatória. Sua principal vantagem é poder utilizar a versão original do algoritmo, criada para problemas de otimização contínua, sem nenhum tipo de alteração na sua estrutura.

Por se tratar de um problema de otimização combinatória, o número máximo de possíveis estruturas pode ser muito grande, dependendo dos limites estabelecidos para os valores do vetor de projeto. Além disso, estruturas com uma complexidade muito elevada podem acabar sendo geradas no processo evolutivo. Para lidar com este problema, serão consideradas apenas estruturas com um número máximo de graus de liberdade, que será definido em função do número de observações dos dados utilizados no treinamento do modelo. Este limite é representado pelo parâmetro ζ. Portanto, apenas as estruturas em que $|\boldsymbol{\theta}| \leq \zeta$ serão consideradas válidas. As demais são consideradas inválidas.

Este critério é utilizado tanto na inicialização aleatória dos parâmetros quanto na função objetivo. Na inicialização, para garantir que apenas estruturas aleatórias válidas sejam geradas; na função objetivo, para evitar que os parâmetros de estruturas

inválidas sejam identificados desnecessariamente. Neste caso, a aptidão da solução que representa a estrutura inválida é penalizada com um valor bem alto, para evitar que ela seja selecionada para a próxima geração.

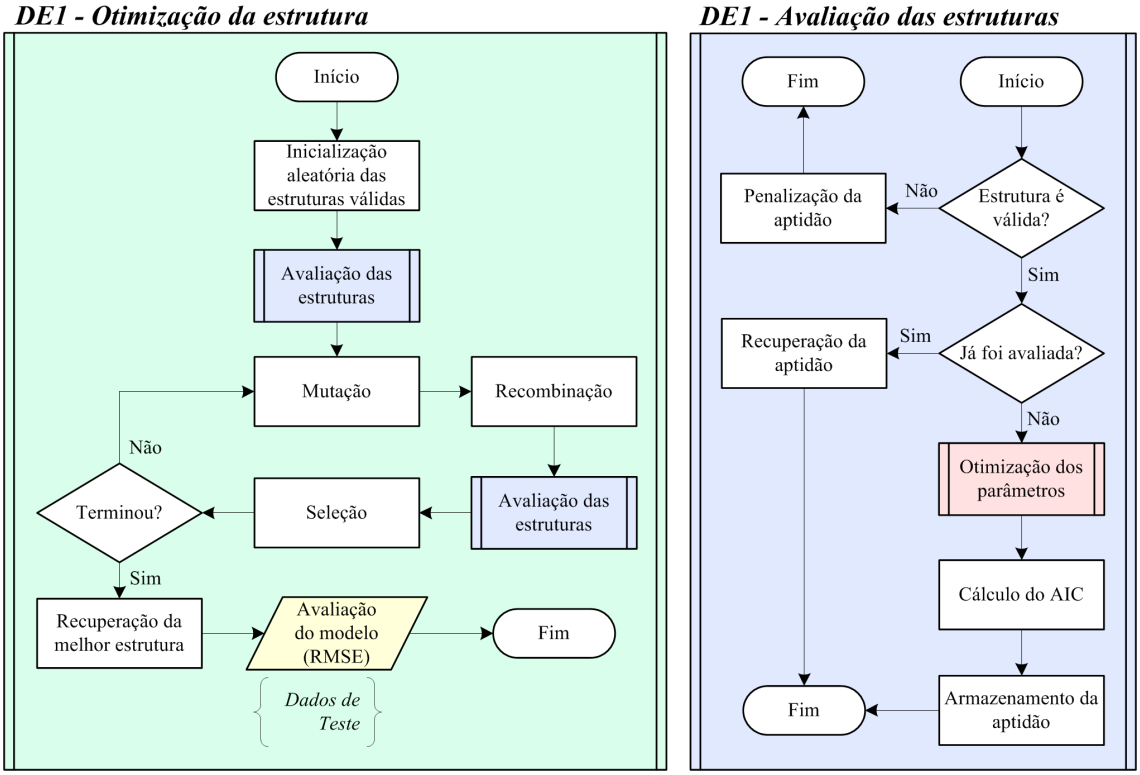


Figura 3.7. Fluxograma do algoritmo de evolução diferencial utilizado no primeiro nível do método hierárquico proposto. O retângulo verde à esquerda representa a instância do algoritmo responsável pela identificação da estrutura, cuja função objetivo é representada pelo retângulo azul à direita.

Adicionalmente, outra verificação é implementada na função objetivo. Neste caso, para garantir que apenas novas estruturas sejam avaliadas a cada geração. Deste modo, sempre que uma estrutura é identificada, o valor de aptidão do vetor que a representa é armazenado para evitar que a instância interna do algoritmo de evolução diferencial, responsável pela identificação dos parâmetros, tenha que ser executada novamente. Esta estratégia busca evitar o reprocessamento desnecessário das estruturas cujos parâmetros já foram identificados.

Ao terminar a identificação da estrutura do modelo, este é finalmente avaliado no conjunto de teste, a fim de verificar o seu desempenho. Esta avaliação é feita com base na raiz quadrada do erro médio quadrático ($RMSE$), definida como:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}(t))^2} \quad \textbf{(3.11)}$$

O processo de identificação dos parâmetros é discutido a seguir.

3.2.2 Identificação dos Parâmetros

No nível 2 da abordagem hierárquica proposta, o problema de identificação dos parâmetros é tratado como um problema de otimização contínua e, portanto, realizado de maneira bem mais simples e direta do que a identificação da estrutura (combinatória). Afinal, o algoritmo de evolução diferencial foi desenvolvido especificamente para este tipo de problema. Portanto, sempre que uma nova estrutura é avaliada no nível 1 (retângulo azul na Figura 3.7), seus respectivos parâmetros são identificados no nível 2. A identificação desses parâmetros é realizada pelo ***DE2*** dentro da função objetivo do ***DE1*** (retângulo vermelho na Figura 3.7), e suas etapas são descritas na Figura 3.8 abaixo.

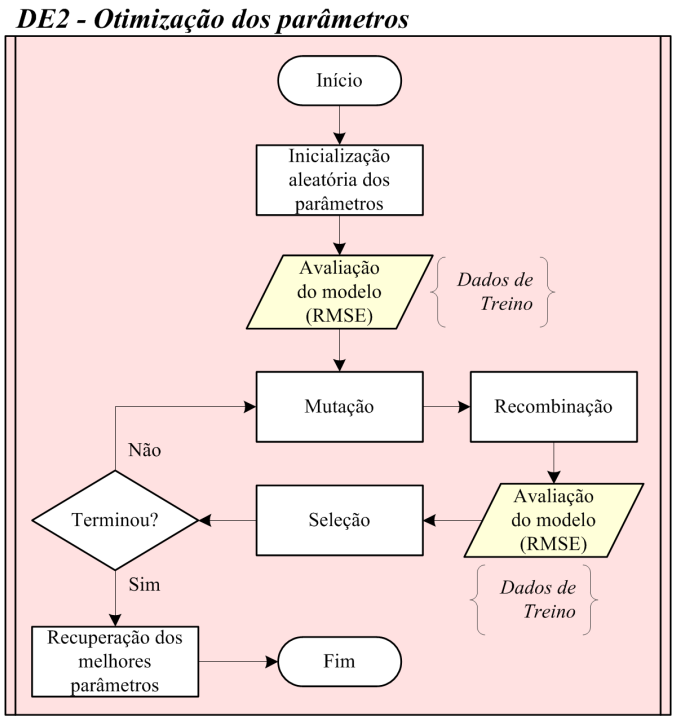


Figura 3.8. Fluxograma do algoritmo de evolução diferencial utilizado no segundo nível do método hierárquico proposto. Este retângulo vermelho representa a instância do algoritmo responsável pela identificação dos parâmetros, que é utilizada dentro da função objetivo da instância implementada no primeiro nível, como pode ser observado no retângulo azul da Figura 3.7.

As variáveis de projeto utilizadas pelo ***DE2*** representam as componentes do vetor $\boldsymbol{\theta}$, que define os parâmetros associados à conclusão das regras e os coeficientes da combinação linear referente à saída do modelo. A função objetivo do ***DE2*** realiza a avaliação do modelo nos dados de treino, utilizando a raiz quadrada do erro médio quadrático, definida em (3.11).

3.3 Resumo

Este capítulo apresentou em detalhes os novos modelos desenvolvidos e a metodologia proposta para a identificação simultânea da estrutura e dos parâmetros desses modelos baseados em sistemas fuzzy recorrentes com duplo feedback.

A primeira seção abordou a formulação matemática tanto do modelo de referência quanto dos modelos desenvolvidos, cujas principais características são a utilização de operadores de defasagem ajustáveis e uma nova conexão de feedback, que pode vir da saída observada do sistema, ou da saída estimada do modelo, ou do erro de predição.

Na segunda seção, o processo de identificação do modelo foi pormenorizado. Este processo é caracterizado por uma estrutura hierárquica em dois níveis. A seleção da estrutura é tratada como um problema de otimização combinatória no primeiro nível. No segundo nível, é realizada a estimação dos parâmetros do modelo. Neste caso, considerada como um problema de otimização contínua. Em ambos os problemas, a otimização é conduzida pela evolução diferencial. Uma instância do algoritmo é utilizada em cada nível.

O próximo capítulo apresentará em detalhes a metodologia experimental considerada nas simulações computacionais que foram empreendidas a fim de validar a abordagem proposta. Foram considerados problemas relacionados a sistemas dinâmicos não lineares e a séries temporais caóticas. Dois problemas diferentes de cada tipo foram considerados. Eles são apresentados no início do capítulo.

Três simulações computacionais diferentes foram realizadas. Cada qual com um propósito específico. A primeira delas teve por objetivo avaliar o desempenho dos métodos de otimização na estimação dos parâmetros. Na segunda simulação, a abordagem proposta foi avaliada, com relação à nova estrutura e ao novo método de identificação propostos. A última simulação teve por objetivo avaliar os modelos gerados com relação ao tipo de feedback adicional utilizado.

Capítulo 4

4 Simulações Computacionais

4.1 Problemas Estudados

Quatro problemas foram considerados a fim de avaliar a abordagem proposta neste trabalho. Os dois primeiros representam sistemas dinâmicos com comportamento não linear, enquanto os dois últimos são séries temporais caóticas. Os três primeiros problemas são frequentemente utilizados na literatura como benchmark, e foram selecionados a fim de permitir a comparação da abordagem proposta com o estado da arte. Embora o último problema também apareça na literatura, ele não é muito frequente e, por isso, só foi considerado na comparação entre a abordagem desenvolvida por Gama et al. (2008) e a abordagem proposta.

4.1.1 Sistemas Dinâmicos não Lineares

4.1.1.1 P1 – Planta 1

Esta planta é governada pela equação de diferenças abaixo, cuja saída depende dos valores das três últimas saídas e das duas últimas entradas (NARENDRA, K. S.; PARTHASARATHY, 1990;1991):

$$y(t+1) = \frac{y(t)y(t-1)y(t-2)u(t-1)(y(t-2)-1)+u(t)}{1+y^2(t-1)+y^2(t-2)} \quad \textbf{(4.1)}$$

Para o conjunto de treino, a entrada $u(t)$ foi gerada a partir de 400 observações aleatórias distribuídas uniformemente no intervalo $[-1,1]$, e mais 400 observações calculadas como $u(t) = 1{,}05\text{sen}(t\pi/45)$. No conjunto de teste, 1000 observações da entrada são geradas como (SASTRY, P. S.; SANTHARAM; UNNIKRISHNAN, 1994):

$$u(t) = \begin{cases} \text{sen}\left(t\dfrac{\pi}{25}\right) & t < 250 \\ 1 & 250 \leq t \leq 500 \\ -1 & 500 < t \leq 750 \\ 0{,}3\,\text{sen}\left(t\dfrac{\pi}{25}\right) + 0{,}1\,\text{sen}\left(t\dfrac{\pi}{32}\right) + 0{,}6\,\text{sen}\left(t\dfrac{\pi}{10}\right) & 750 < t \leq 1000 \end{cases} \quad \textbf{(4.2)}$$

A Figura 4.1, a seguir, ilustra os dados de entrada e saída desta planta nos conjuntos de treino e teste.

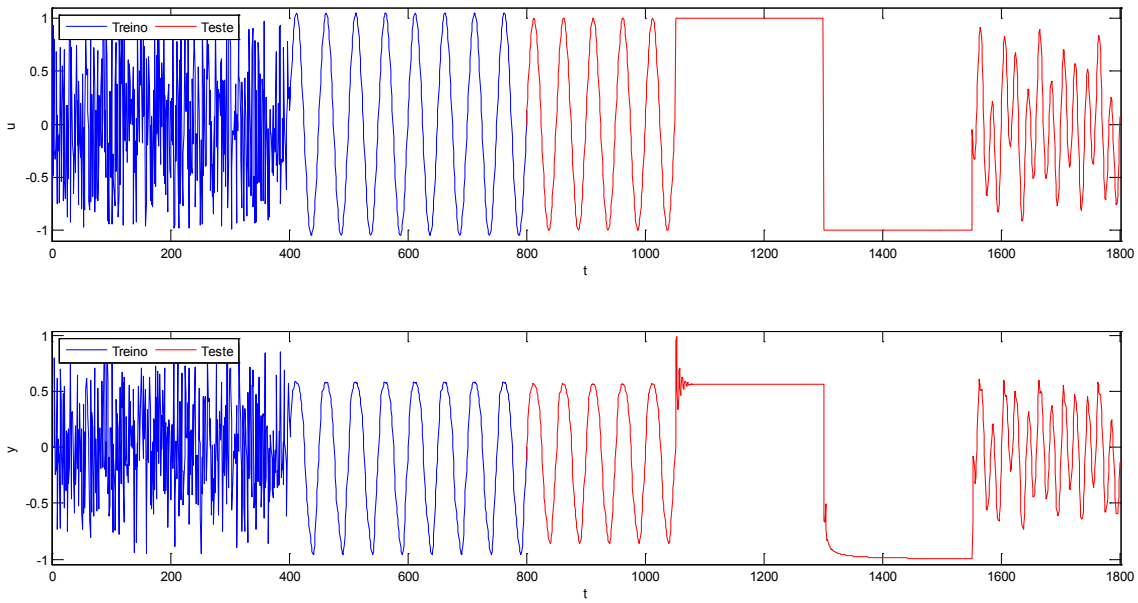


Figura 4.1. Dados de entrada e saída da Planta 1 nos conjuntos de treino e teste.

4.1.1.2 P2 – Planta 2

Esta é uma planta de segunda ordem governada pela equação de diferenças definida abaixo, cuja saída depende de valores da entrada com maior atraso (JIN-HWAN; UK-YOUL, 1998):

$$y(t+1) = 0{,}72y(t) + 0{,}025y(t-1)u(t-1) + 0{,}01u^2(t-2) + 0{,}2u(t-3) \quad \textbf{(4.3)}$$

Os mesmos dados de entrada gerados para a Planta 1 são utilizados para esta planta, como pode ser visto na Figura 4.2.

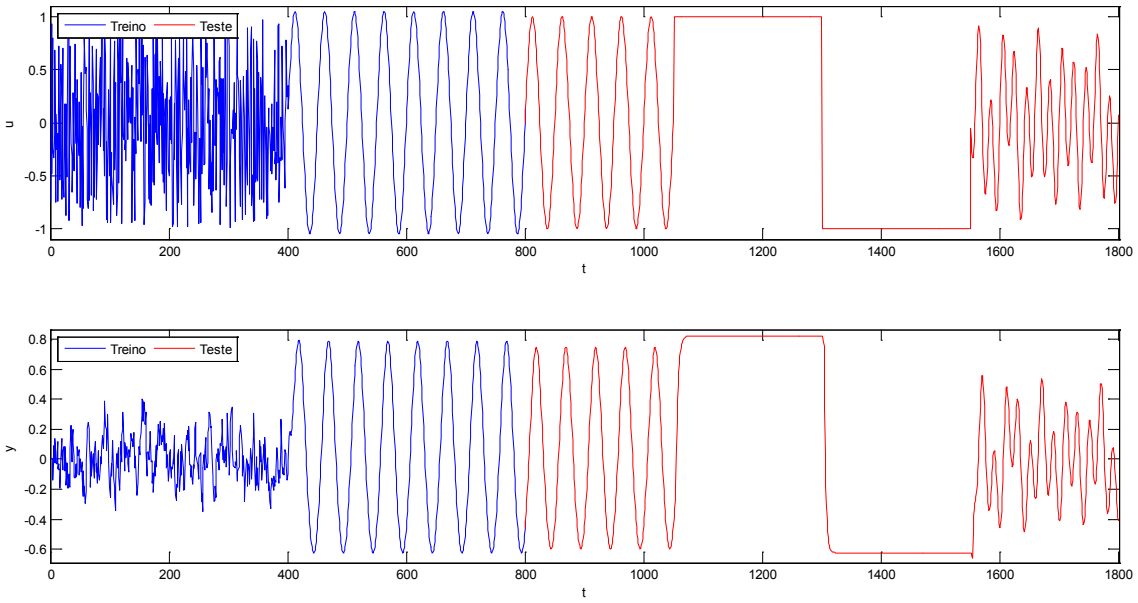


Figura 4.2. Dados de entrada e saída da Planta 2 nos conjuntos de treino e teste.

4.1.2 Séries Temporais Caóticas

A modelagem e o processamento de sistemas representados por séries caóticas têm sido investigada em diferentes domínios como física, química, biologia, medicina, finanças, etc. A previsão de séries caóticas tem grande significado prático nessas áreas, e ainda é um dos problemas mais desafiadores atualmente (BASHARAT; SHAH, 2009; CASDAGLI, 1989; FARMER; SIDOROWICH, 1987; LAI, Y.-C.; YE, 2003). Visto que sistemas caóticos dependem sensivelmente de condições iniciais, as trajetórias geradas a partir de dois estados semelhantes irão divergir exponencialmente, levando à aparente imprevisibilidade dos sistemas caóticos. Porém, diversos trabalhos já têm demonstrado a viabilidade da previsão desse tipo de sistema.

4.1.2.1 P3 – Série Caótica Univariada de Hénon

O sistema de Hénon é um exemplo de sistema dinâmico com comportamento caótico. Ele é um sistema discreto de segunda ordem regido pela seguinte equação (HÉNON, 1976):

$$y(t+1) = -P \times y^2(t) + Q \times y(t-1) + 1 \quad \textbf{(4.4)}$$

que produz um estranho atrator caótico quando $P = 1{,}4$ e $Q = 0{,}3$, como pode ser visto na Figura 4.4.

Duas mil observações são geradas a partir do estado inicial $[y(0); y(1)] = [0{,}4; 0{,}4]$. As primeiras 1000 são utilizadas no conjunto de treino, e as últimas 1000 no conjunto de teste, como mostra a Figura 4.3 abaixo.

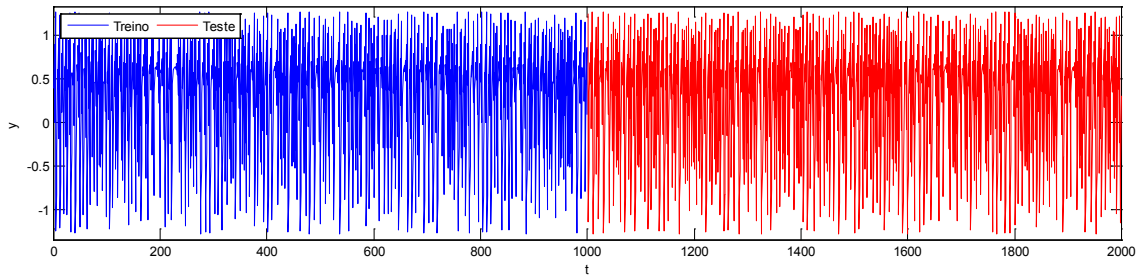


Figura 4.3. Série temporal caótica de Hénon.

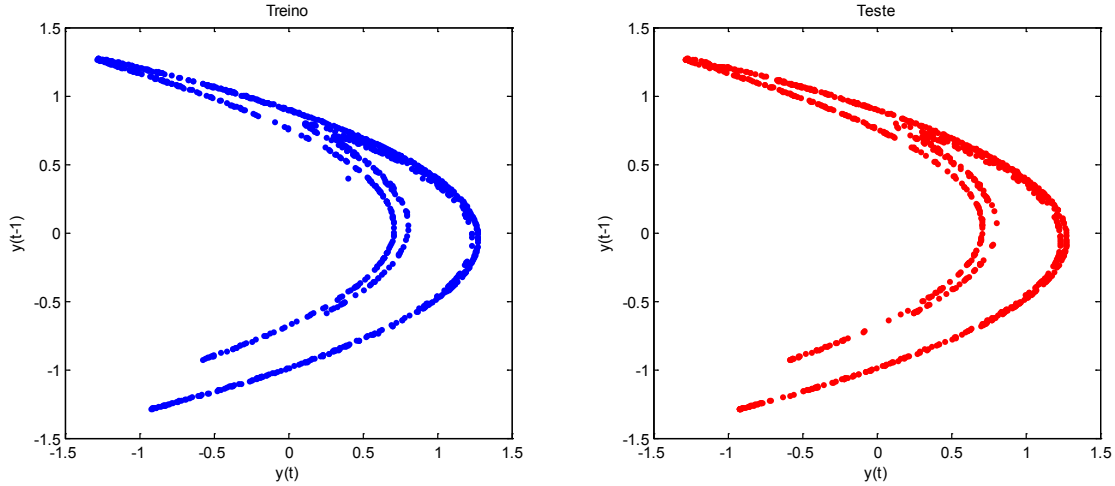


Figura 4.4. Espaço de fases do atrator caótico de Hénon nos conjuntos de treino e teste.

4.1.2.2 P4 – Série Caótica Multivariada de Lorenz

O sistema de Lorenz é um modelo tridimensional clássico de um sistema dinâmico não linear determinístico que apresenta atratores estranhos e comportamento caótico (LORENZ, 1963). Ele é descrito pelas seguintes equações diferenciais ordinárias:

$$\begin{cases} \dot{X} = -a(X - Y) \\ \dot{Y} = (c - Z)X - Y \\ \dot{Z} = XY - bZ \end{cases} \quad \textbf{(4.5)}$$

em que X, Y e Z são funções reais do tempo, e a, b e c são parâmetros reais positivos. A maioria das simulações numéricas tem definido os seguintes valores para os parâmetros: $a = 10$; $b = 8/3$ e $c = 28$.

Duas mil observações foram selecionadas para cada uma das três dimensões a partir dos dados gerados entre 0 e 80 seg., com intervalo de 0,01 seg, utilizando $x(0) = y(0) = z(0) = 10$. As primeiras 1.000 observações são utilizadas no conjunto de teste e as 1.000 seguintes no conjunto de treino. A Figura 4.5 ilustra o estranho atrator caótico tridimensional formado pelas 2.000 observações selecionadas; a Figura 4.6 apresenta a série temporal de cada uma das dimensões X, Y e Z; e a Figura 4.7 apresenta suas respectivas trajetórias bidimensionais.

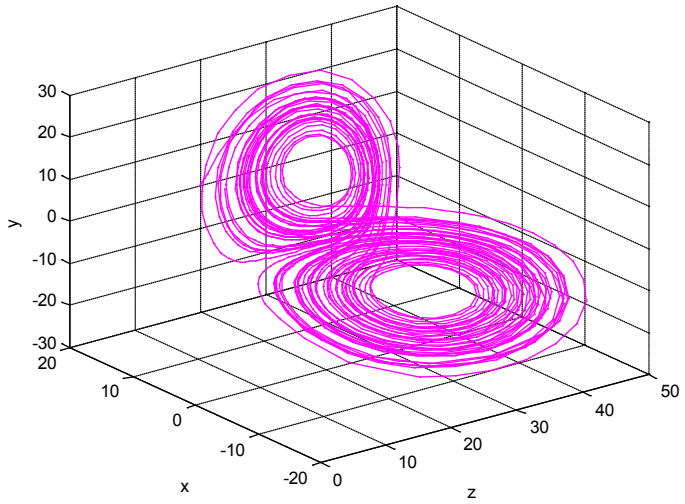


Figura 4.5. Estranho atrator caótico tridimensional de Lorenz

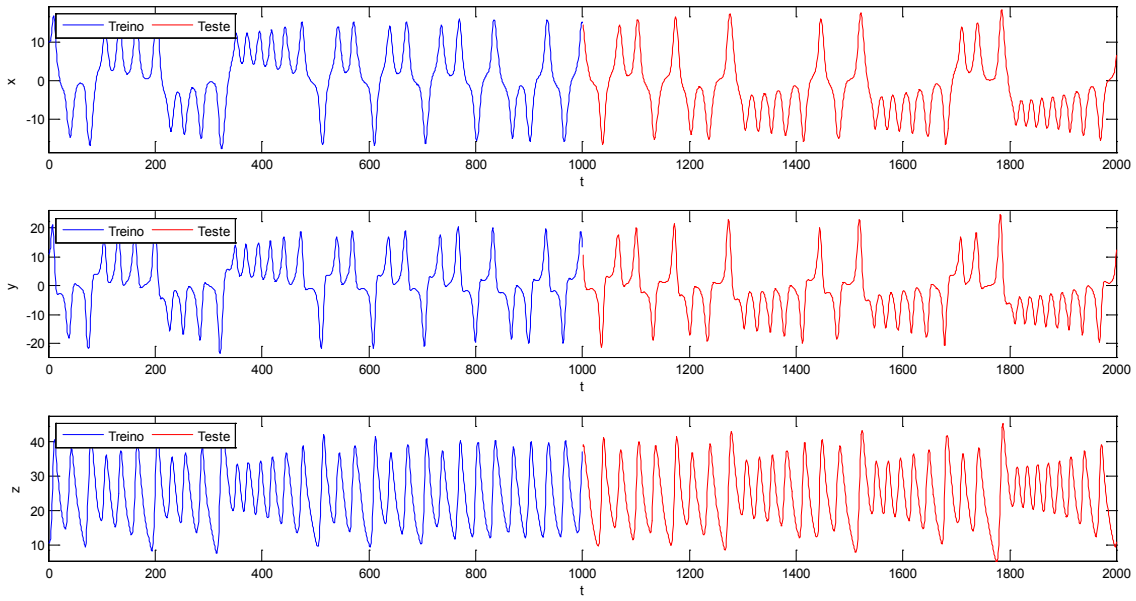


Figura 4.6. Séries temporais caóticas univariadas de Lorenz.

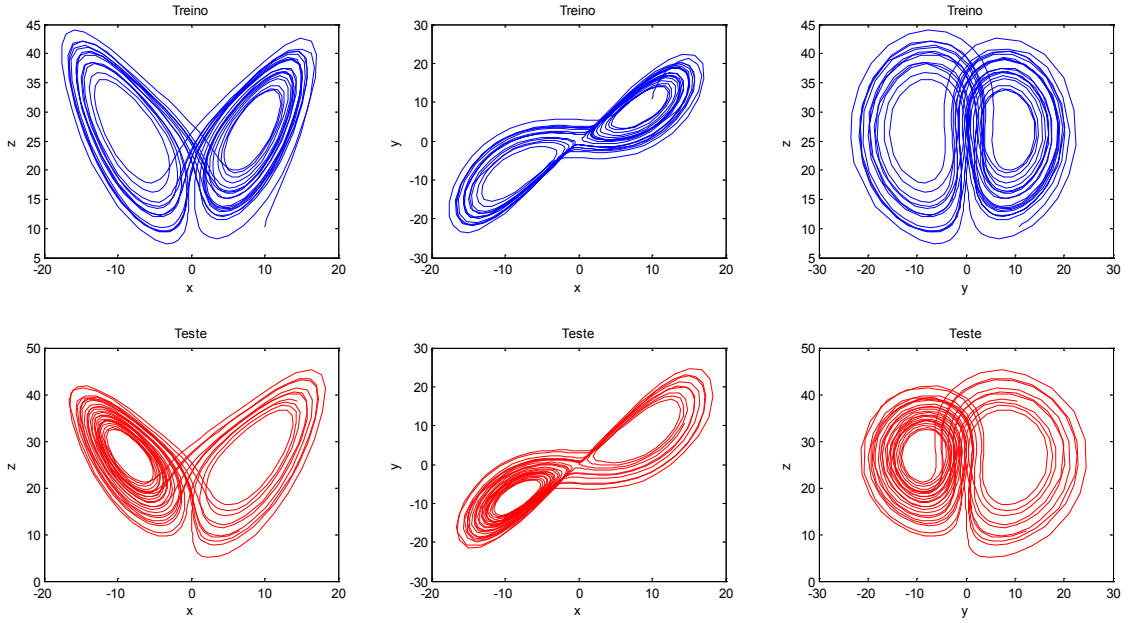


Figura 4.7. Trajetórias do atrator caótico de Lorenz nos conjuntos de treino e teste.

4.2 Descrição das Simulações

Três simulações computacionais foram realizadas nesta tese. Na primeira, o objetivo foi avaliar o desempenho entre o algoritmo genético e três variações do algoritmo de evolução diferencial, DE/*rand*/1/bin, DE/*best*/1/bin e DE/*target-to-best*/1/bin, na otimização dos parâmetros do modelo desenvolvido por Gama et al. (2008), apresentado na seção 3.1.1. Além de verificar se a evolução diferencial confirmaria sua superioridade sobre os algoritmos genéticos, como em (LIN, F. T., 2010; OH et al., 2012), desejava-se verificar qual das três variações da evolução diferencial obteria melhor desempenho e, consequentemente, também seria utilizada na otimização da estrutura dos modelos desenvolvidos.

O objetivo da segunda simulação foi avaliar o desempenho da nova abordagem proposta (estrutura com duplo feedback e parâmetros de defasagem ajustáveis; e identificação simultânea da estrutura e dos parâmetros usando a evolução diferencial) à luz da abordagem original (modelo com um feedback; e busca exaustiva da estrutura). A fim de avaliar tanto a nova estrutura quanto o novo procedimento de identificação, duas versões da abordagem originalmente proposta por Gama *et al.* (2008) foram consideradas. Em uma delas os parâmetros são otimizados por um algoritmo genético canônico, enquanto na outra utilizou-se a versão da evolução

diferencial que apresentou o melhor desempenho na primeira simulação computacional. A estrutura do modelo considerado pela abordagem proposta nesta simulação utilizou apenas a saída estimada do modelo como feedback adicional, visto que o modelo original opera em modo de simulação.

Na última simulação computacional, o objetivo foi verificar qual dentre os três tipos de feedback adicional proporcionaria melhor desempenho ao modelo. Ademais, comparou-se o desempenho dos novos modelos propostos com os principais modelos baseados em sistemas fuzzy recorrentes disponíveis na literatura.

Devido à natureza estocástica do algoritmo de evolução diferencial, resultados diferentes podem ser obtidos para um determinado modelo dependendo da semente de geração de números aleatórios utilizada. Por isso, todas as simulações computacionais foram repetidos por 30 vezes, como em (CHEN, C.-H., 2013; CHEN, C.-H.; YANG, 2013; SU, H.; YANG, 2011). A partir dos resultados obtidos nas 30 execuções, foram calculados os valores mínimo, máximo e médio do $RMSE$, bem como o desvio padrão e o intervalo de confiança do valor médio. Este último foi calculado usando a técnica de reamostragem do tipo BCa (DICICCIO; EFRON, 1996) com 1.000 amostras e 95% de confiança.

Adicionalmente, a avaliação dos resultados foi realizada com base em testes estatísticos não paramétricos tanto para múltiplas comparações quanto para comparações emparelhadas (DERRAC et al., 2011; GARCÍA et al., 2010). No primeiro caso, foram utilizados os testes estatísticos de Friedman com postos alinhados e de Finner (post-hoc). Para o último caso foi considerado o teste estatístico de Wilcoxon. Os valores dos testes foram obtidos pela ferramenta KEEL (ALCALÁ-FDEZ et al., 2009).

Nas simulações realizadas, a hipótese nula que foi testada considerou que todos os métodos possuíam desempenho equivalente, sem qualquer tipo de diferença significante entre eles. O resultado do teste de hipótese foi expresso pelo valor p, que representa a medida de significância estatística das diferenças observadas na avaliação da hipótese. Um grau de certeza de 95% foi considerado no teste ($\alpha = 0{,}05$), de modo que a hipótese nula foi rejeitada sempre que $p < 0{,}05$.

O teste de Wilcoxon é utilizado para comparar duas amostras e verificar se elas representam populações diferentes. O teste de Friedman, por sua vez, é utilizado para

comparar duas ou mais amostras e verificar se pelo menos duas delas representam populações diferentes. O teste de Friedman é particularmente recomendado para comparar até cinco métodos, pois realiza tanto a comparação individual dos métodos em diferentes amostras quanto à comparação conjunta dos métodos em uma mesma amostra (DERRAC et al., 2011). Portanto, ele leva em consideração tanto as diferenças intra-amostra, quanto inter-amostras.

Após aplicar o teste de Friedman, o teste *post-hoc* de Finner foi realizado, como sugerido em (GARCÍA et al., 2010). No teste *post-hoc*, o objetivo foi analisar as diferenças entre os valores dos postos obtidos pelo teste de Friedman e verificar se elas eram estatisticamente significantes. Este teste é útil devido a sua baixa probabilidade de aceitar erroneamente a hipótese nula (erro do tipo II), e utiliza valores ajustados de p a fim de minimizar o problema de rejeitar erroneamente a hipótese nula (erro do tipo I) (GARCÍA et al., 2010).

4.2.1 Avaliação dos Métodos de Otimização dos Parâmetros

O objetivo desta simulação computacional foi verificar qual dos algoritmos representava a melhor opção para o problema de otimização dos parâmetros, e definir qual variação da evolução diferencial seria utilizada na otimização da estrutura, *DE/rand/1/bin*, *DE/best/1/bin* ou *DE/target-to-best/1/bin*. Para tal, apenas o modelo desenvolvido por Gama et al. (2008) foi considerado, ou seja, sem duplo feedback e operador de defasagem (Figura 3.1). A estrutura deste modelo é representada por três números inteiros que indicam, respectivamente, o número de estados, o número de conjuntos fuzzy utilizados no domínio das variáveis de estado, e o número de conjuntos fuzzy utilizados no domínio da variável de entrada.

Deste modo, um modelo representado pelo número 235, por exemplo, indica uma estrutura com 2 variáveis de estado; 3 conjuntos fuzzy no domínio das variáveis de estado, e 5 conjuntos fuzzy no domínio da variável de entrada. São considerados modelos de 1ª, 2ª e 3ª ordem, ou seja, modelos com uma, duas ou três variáveis de estado; e o número de conjuntos fuzzy no domínio de cada variável varia entre 2 e 9.

Assim como em (GAMA et al., 2008), uma busca exaustiva é realizada para selecionar a melhor estrutura. Apenas estruturas com no máximo $\zeta = N \times 0{,}15$

parâmetros são consideradas, em que N indica o número de observações nos dados de treino. A Tabela 4.1 apresenta o número de estruturas consideradas para cada um dos problemas estudados. A estrutura selecionada foi aquela com a maior frequência de ocorrência nas 30 execuções. Quando mais de uma estrutura ocorreu com a mesma frequência, o princípio da parcimônia foi adotado, de modo que aquela com o menor número de parâmetros foi selecionada.

Como esta simulação tinha por objetivo avaliar particularmente os algoritmos de otimização, a comparação do desempenho entre os algoritmos genéticos e as três variações da evolução diferencial foi realizada apenas nos dados de treino. A parametrização do algoritmo genético foi realizada de acordo com (GAMA et al., 2008), e é apresentada na Tabela 4.2.

A parametrização das variações da evolução diferencial é apresentada na Tabela 4.3. Os valores de CR foram definidos experimentalmente considerando-se valores entre 0,1 e 0,9 com intervalo de 0,1. Cada um desses valores foi avaliado considerando-se o desempenho do modelo no problema P1. Dez execuções com diferentes inicializações dos números aleatórios foram realizadas para cada valor, e o valor de CR foi selecionado de acordo com o melhor desempenho médio.

Tabela 4.1. Número máximo de parâmetros e estruturas em cada problema.

Problema	N	ζ	**Estruturas**			
			1ª Ordem	*2ª Ordem*	*3ª Ordem*	*Total*
P1 P2	800	120	64	28	11	103
P3 P4(X) P4(Y) P4(Z)	1.000	150	64	34	13	111

Tabela 4.2. Parametrização do algoritmo genético (GA).

Parâmetro	**Valor**
Tamanho da População	20
Taxa de Recombinação	0,6
Taxa de Mutação	0,01
Gerações	1.750
Elitismo	1

Tabela 4.3. Parametrização da evolução diferencial (DE).

Parâmetro	Valor		
	DE/rand/1/bin	*DE/best/1/bin*	*DE/target-to-best/1/bin*
NP	20	20	20
CR	0,8	0,3	0,5
F	$^{*}\mathrm{rand}(0{,}3; 0{,}9)$	$^{*}\mathrm{rand}(0{,}3; 0{,}9)$	$^{*}\mathrm{rand}(0{,}3; 0{,}9)$
MAX_GER	1.750	1.750	1.750

* *Um número diferente é considerado a cada geração.*

4.2.2 Avaliação da Abordagem Proposta

O objetivo desta simulação computacional foi verificar o desempenho da abordagem proposta nos problemas estudados. Esta verificação considerou apenas uma das três novas estruturas desenvolvidas. Neste caso, a que utiliza a saída estimada do modelo como feedback adicional. A restrição deve-se ao fato do modelo original desenvolvido por Gama et al. (2008) não utilizar a saída real observada do sistema. Os parâmetros do modelo original foram identificados utilizando-se tanto o algoritmo genétic (RFS–GA) quanto a evolução diferencial (RFS–DE). Isso permitiu levar em consideração o impacto do novo mecanismo de identificação automática da estrutura na comparação dos resultados entre os modelos.

Embora o ideal fosse avaliar o impacto da nova estrutura e do novo mecanismo de identificação, separadamente, a fim de poder quantificar a contribuição individual de cada um para a melhoria do desempenho, isso teria um alto custo computacional devido à busca exaustiva da nova estrutura. Enquanto a estrutura do modelo original é caracterizada por 3 números inteiros, como visto na seção anterior, as novas estruturas propostas são caracterizadas por 6 números inteiros. Em se tratando de um problema de otimização combinatória, a busca exaustiva seria inviável para o novo tipo de estrutura. Na verdade, a motivação por trás do desenvolvimento do processo de identificação simultânea da estrutura e dos parâmetros é exatamente esta. Portanto, o desempenho de ambas as contribuições foi avaliado em conjunto.

Na nova estrutura, cada um dos 6 números que a descrevem indicam, respectivamente: o número de estados; o número de conjuntos fuzzy da variável de estado; o número de conjuntos fuzzy da variável de entrada; o número de conjuntos

fuzzy do feedback adicional; o valor do atraso da variável de entrada; e o valor do atraso do feedback adicional. Um modelo representado pelo número 135402, por exemplo, indica uma estrutura de primeira ordem com 3 conjuntos fuzzy na variável de estado; 5 conjuntos fuzzy na variável de entrada, que não possui atraso; e 4 conjuntos fuzzy na variável que representa o feedback adicional, cujo valor do atraso é 2.

Tanto nesta quanto na próxima simulação, a ordem do modelo varia de 1 a 3; o número de conjuntos fuzzy no domínio de cada variável varia entre 2 e 9; e os valores de atraso variam de 0 a 4 para a variável de entrada e de 1 a 4 para o feedback adicional. Assim como na simulação anterior, apenas as estruturas com no máximo $\zeta = N \times 0{,}15$ parâmetros são consideradas. Como pode ser visto na Tabela 4.4, o espaço de busca para as novas estruturas é bem maior do que o espaço de busca para a estrutura original. Além do mais, a estratégia de busca exaustiva considerada originalmente atua diretamente sobre o espaço de busca restrito, enquanto que o novo mecanismo de identificação automática da estrutura atua em todo o espaço de busca, penalizando as soluções que violam o espaço restrito. Nos dois primeiros problemas, a quantidade de estruturas válidas do novo modelo é mais de 60 vezes maior do que a quantidade de estruturas válidas do modelo original. Nas demais simulações, essa quantidade chega a ser superior a 70.

Tabela 4.4. Tamanho do espaço de busca para as estruturas em cada problema.

Problema	N	ζ	**Estrutura Original**		**Novas Estruturas**	
			Total	**Restrito**	**Total**	**Restrito**
P1 P2	800	120	192	103	30.720	6.340
P3 P4(X) P4(Y) P4(Z)	1.000	150	192	112	30.720	7.980

Como o método de identificação simultânea da estrutura e dos parâmetros que foi apresentado na seção 3.2 utiliza duas instâncias do algoritmo de evolução diferencial, dois conjuntos diferentes de parâmetros foram estabelecidos. A parametrização da instância do algoritmo responsável pela identificação dos parâmetros do modelo é a mesma definida na Tabela 4.3 para a versão DE/rand/1/bin,

que apresentou os melhores resultados na simulação anterior e passou a ser considerada nas demais simulações. A Tabela 4.5 abaixo apresenta a parametrização da instância do algoritmo que lida com o problema da identificação da estrutura. O valor do parâmetro CR foi definido experimentalmente de maneira análoga à que foi descrita na simulação anterior. Com relação ao algoritmo genético, foi utilizada a mesma parametrização definida na Tabela 4.2 para a simulação anterior.

Tabela 4.5. Parametrização da versão DE/rand/1/bin utilizada na identificação da estrutura.

Parâmetro	Valor
NP	12
CR	0,3
F	$^{*}\text{rand}(0{,}3; 0{,}9)$
MAX_GER	40

* *Um número diferente é considerado a cada geração.*

A inicialização utilizada pela instância do algoritmo que lida com a identificação da estrutura é aleatória para todos os elementos com exceção do primeiro, que representa a ordem do modelo. Este elemento é inicializado com o valor 1 nos quatro primeiros vetores da população; com o valor 2 nos quatro vetores seguintes; e com o valor 3 nos quatro últimos vetores da população. Deste modo, garante-se que a população inicial tenha a mesma quantidade de modelos de 1ª, 2ª e 3ª ordem, deixando a cargo do processo evolutivo decidir qual delas representa a melhor opção para o problema abordado.

Do mesmo modo que o critério AIC é utilizado pela abordagem proposta para selecionar as estruturas, ele também foi utilizado para selecionar a estrutura dos modelos RFS–GA e RFS–DE em cada uma das 30 execuções.

4.2.3 Avaliação do Tipo de Feedback Adicional

Esta foi a última simulação realizada, e teve por objetivo verificar qual dos três tipos de feedback adicional, saída estimada, saída observada, ou erro, seria a melhor opção para o modelo em termos de desempenho. Além disso, comparou-se o desempenho dos modelos baseados em cada um dos três tipos de feedbacks com os principais modelos baseados em sistemas fuzzy recorrentes apresentados na literatura, a fim de contextualizar o desempenho dos novos modelos.

As mesmas configurações utilizadas na simulação anterior foram consideradas nesta. Porém, uma nova métrica foi utilizada na avaliação dos resultados, tendo em vista que o $RMSE$ representa o erro na mesma escala da variável de saída. O $NDEI$ (índice de erro não dimensional) (LAPEDES; FARBER, 1987) *apud* (JANG, 1993):

$$NDEI = \frac{1}{\sigma}\sqrt{\frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}(t))^2} = \frac{RMSE}{\sigma} \quad \textbf{(4.6)}$$

foi considerado por ser uma medida não dimensional do erro, permitindo comparar o desempenho de um determinado modelo em diferentes problemas. Ele também é conhecido como $RSME$ normalizado, visto que representa a padronização desta medida por meio do desvio padrão da variável de saída, representado por σ na equação (4.6) acima.

No capítulo que se segue serão apresentados e discutidos os resultados obtidos nas simulações computacionais que foram detalhadamente descritas neste capítulo.

Capítulo 5

5 Resultados e Discussão

Este capítulo apresenta e discute os resultados das simulações computacionais que foram conduzidas a fim validar as abordagens propostas nesta tese segundo a metodologia descrita na seção 4.2.1. Os resultados apresentados nas três seções deste capítulo são considerados de forma consolidada por questões de clareza. No entanto, eles são detalhados nos apêndices. O Apêndice A apresenta os resultados detalhados referentes a seção 5.1; o Apêndice B apresenta os resultados detalhados referentes à seção 5.2; e, finalmente, o Apêndice C apresenta os resultados detalhados referentes à seção 5.3.

A seção 5.1 apresenta e discute os resultados da simulação computacional que foi conduzida a fim de avaliar o método de otimização. A seção 5.2 apresenta e discute os resultados da simulação computacional que foi realizada a fim de avaliar a abordagem proposta; enquanto a seção 5.3 apresenta e discute os resultados da simulação computacional que foi conduzida a fim de avaliar o tipo de feedback adicional.

Os melhores resultados apresentados nas tabelas estão destacados. Os valores em negrito representam os melhores resultados em termos de desempenho ($RMSE$) enquanto que os valores sublinhados representam os melhores resultados em termos de complexidade (quantidade de parâmetros).

5.1 Simulação 1: Método de Otimização

Nesta seção, são apresentados e discutidos os resultados obtidos na simulação computacional descrita na seção 4.2.1, que teve por objetivo verificar qual dos algoritmos representa a melhor opção para o problema de otimização dos parâmetros – o algoritmo genético, proposto por Gama et al. (2008), ou a evolução diferencial, proposta neste trabalho.

A Tabela 5.1 abaixo apresenta o resultado consolidado da primeira simulação em termos do desempenho e da complexidade dos modelos nos dados de treino, quantificados através do erro $RMSE$ e do número de parâmetros. Esses valores foram obtidos para cada um dos métodos de otimização considerados em cada um dos problemas estudados.

Tabela 5.1. Resultado comparativo entre os algoritmos – Simulação 1.

	GA	**DE/rand/1/bin**	**DE/best/1/bin**	**DE/target-to-best/1/bin**
P1				
Estrutura	234	334	334	334
Parâmetros	36	108	108	108
$RMSE$	0,0685 ± 0,0025	**0,0471 ± 0,0022**	0,0475 ± 0,0027	0,0511 ± 0,0044
P2				
Estrutura	236	322	322	322
Parâmetros	54	16	16	16
$RMSE$	0,0880 ± 0,0028	**0,0732 ± 0,0060**	0,0808 ± 0,0076	0,0795 ± 0,0073
P3				
Estrutura	148	335	335	335
Parâmetros	32	135	135	135
$RMSE$	0,1048 ± 0,0569	**0,0465 ± 0,0109**	0,0597 ± 0,0402	0,0609 ± 0,0183
P4(X)				
Estrutura	247	335	334	329
Parâmetros	112	135	108	72
$RMSE$	1,4575 ± 0,0913	**0,7937 ± 0,0378**	0,8036 ± 0,0539	0,8549 ± 0,0474
P4(Y)				
Estrutura	247	334	334	334
Parâmetros	112	108	108	108
$RMSE$	2,0471 ± 0,1331	**1,2174 ± 0,0372**	1,2726 ± 0,0822	1,3470 ± 0,1134
P4(Z)				
Estrutura	239	329	329	329
Parâmetros	81	72	72	72
$RMSE$	2,3871 ± 0,1768	1,3190 ± 0,0344	**1,3153 ± 0,0208**	1,3360 ± 0,0173

Os valores na Tabela 5.1 foram obtidos a partir dos valores apresentados na Tabela A.1 para o problema P1; Tabela A.2 para o problema P2; Tabela A.3 para o problema P3; Tabela A.4 para o problema P4(X); Tabela A.5 para o problema P4(Y); e Tabela A.6 para o problema P4(Z). Nestas tabelas, os resultados são apresentados para modelos de 1ª, 2ª e 3ª ordem. Enquanto os melhores resultados (Tabela 5.1) obtidos pelos algoritmos genéticos são referentes a modelos de 2ª ordem, exceto para o problema P3 cujo modelo é de 1ª ordem, os melhores resultados obtidos por todas as variações da evolução diferencial são referentes a modelos de 3ª ordem.

A evolução diferencial obteve melhor desempenho que os algoritmos genéticos em todos dos problemas. O desempenho da variação DE/rand/1/bin superou o desempenho das demais variações em todos os problemas, exceto o problema P4(Z), em que o melhor desempenho foi obtido pela variação DE/best/1/bin. Com relação à complexidade dos modelos, os algoritmos genéticos superaram a evolução diferencial em apenas dois dos seis problemas, a saber P1 e P3. A complexidade dos modelos gerados pelas variações da evolução diferencial foi a mesma para todos os problemas com a exceção do problema P4(X), em que o modelo gerado pela variação DE/target-to-best/1/bin foi mais parcimonioso.

A fim de verificar se as diferenças entre os métodos de otimização são estatisticamente significantes com relação ao desempenho e à complexidade dos modelos gerados, o teste estatístico de Friedman com posto alinhado foi realizado. A Tabela 5.2 apresenta os valores do posto médio e de p para este teste, e o resultado sugere que a hipótese nula não pode ser rejeitada para nenhum dos dois critérios testados, indicando que nenhum método em particular é ao mesmo tempo melhor do que todos os outros.

Porém, a diferença entre os postos médios do desempenho sugere que o algoritmo genético é consideravelmente pior do que a evolução diferencial. Esta diferença é então adicionalmente analisada usando o teste *post-hoc* de Finner para confirmar se as diferenças são estatisticamente significantes entre o melhor método e os demais, evitando assim incorrer no erro do tipo II. Este teste adicional leva em consideração as múltiplas comparações emparelhadas e utiliza para isso um valor

ajustado de p. A Tabela 5.3 apresenta os valores ajustados de p segundo o teste de Finner associados ao teste de Friedman que é mostrado na Tabela 5.2.

Tabela 5.2. Valor do posto médio obtido pelo teste de Friedman – Simulação 1.

Algoritmo	Desempenho	Complexidade
GA	21,5	12
DE/rand/1/bin	**7,8333**	14,3333
DE/best/1/bin	9,5	12,6667
DE/target-to-best/1/bin	11,1667	11
Valor p	0,2332	0,1894

Tabela 5.3. Valor ajustado de p obtido pelo teste post-hoc de Finner – Simulação 1.

Algoritmo	Desempenho	Complexidade
GA	0,0024	0,8216
DE/rand/1/bin	–	0,7989
DE/best/1/bin	0,5517	0,8216
DE/target-to-best/1/bin	0,6831	–

De acordo com o resultado do teste mostrado na Tabela 5.3, a hipótese nula para o desempenho da versão DE/*rand*/1/bin pode ser rejeitada apenas para o algoritmo genético. Neste caso, com um nível de significância considerável. Para as demais variações, a hipótese nula é aceita, corroborando o resultado obtido no teste de Friedman. Portanto, enquanto o desempenho da variação DE/*rand*/1/bin pode ser considerado estatisticamente melhor do que o desempenho do algoritmo genético, ele não pode ser considerado estatisticamente melhor que o desempenho das demais variações. Com relação à complexidade dos modelos, o resultado do teste de Finner ratifica o resultado obtido pelo teste de Friedman, e aceita a hipótese nula com um alto nível de significância. Embora a variação DE/*target-to-best*/1/bin tenha apresentado modelos com menor número de parâmetros, não há evidências estatísticas de que os modelos gerados por esta técnica sejam mais parcimoniosos do que os modelos gerados pelas demais técnicas.

5.2 Simulação 2: Abordagem Proposta

Nesta seção, são apresentados e discutidos os resultados obtidos na simulação computacional descrita na seção 4.2.2, que teve por objetivo analisar o desempenho da abordagem proposta nos problemas estudados à luz da abordagem original, desenvolvida por Gama et al. (2008). Neste caso, a abordagem proposta foi avaliada em função do modelo cujo feedback adicional é representado pela saída estimada pelo próprio modelo. Este modelo é referido nesta seção por *DFRFS–DE*, enquanto que os modelos baseados na abordagem original são referidos por *RFS–AG* e *RFS–DE*, dependendo do tipo de algoritmo utilizado.

A Tabela 5.4 apresenta os resultados consolidados da segunda simulação, cujos valores foram retirados das tabelas do Apêndice B, que apresentam os resultados detalhados desta simulação. Ao contrário da simulação anterior em que o desempenho foi avaliado apenas nos dados de treino, a Tabela 5.4 apresenta o desempenho tanto nos dados de treino quanto nos dados de teste. Pois, nesta simulação, a habilidade de generalização dos modelos também está sendo levada em consideração, sendo o principal objeto de análise do desempenho.

Como pode ser visto na Tabela 5.4, a estrutura mais frequente dos modelos gerados pela abordagem proposta são de 1ª ordem, exceto para o problema P1 em que o modelo é 2ª ordem. Neste problema, a estrutura mais frequente de ambos os modelos, *RFS–AG* e *DFRFS–DE*, possuem o mesmo número tanto de estados quanto de conjuntos fuzzy nas variáveis de estado e entrada. Porém, com uma estrutura mais complexa, utilizando a saída estimada atrasada em dois instantes, o modelo *DFRFS–DE* obteve um desempenho médio nos dados de teste quase 3,5 vezes melhor do que o modelo *RFS–AG*, embora com um número de parâmetros também quase 3,5 vezes maior que este modelo.

Uma das possíveis razões para a geração de modelos de 1ª ordem pela abordagem proposta é o termo de penalização da complexidade do modelo, que é definido pelo critério de informação de Akaike, utilizado na função objetivo do algoritmo responsável pela identificação da estrutura. Isto acontece em função do aumento do número de parâmetros devido à inclusão das variáveis de estado defasadas nos modelos de ordem superior.

Tabela 5.4. Resultado comparativo entre os modelos – Simulação 2.

	RFS-GA	RFS-DE	DFRFS-DE
P1			
Estrutura mais frequente	232	332	232402
Parâmetros	28 ± 10	71 ± 25	92 ± 16
$RMSE$ (Treino)	0,0677 ± 0,0013	0,0456 ± 0,0040	**0,0265 ± 0,0014**
$RMSE$ (Teste)	0,0989 ± 0,0656	0,0478 ± 0,0019	**0,0286 ± 0,0081**
P2			
Estrutura mais frequente	223	322	134331
Parâmetros	13 ± 4	36 ± 23	32 ± 6
$RMSE$ (Treino)	$0{,}0865 \pm 7{,}5539 \times 10^{-4}$	0,0667 ± 0,0047	$\mathbf{0{,}0055 \pm 1{,}8041 \times 10^{-4}}$
$RMSE$ (Teste)	0,1505 ± 0,0359	0,0573 ± 0,0204	**0,0220 ± 0,0124**
P3			
Estrutura mais frequente	148	335	139302
Parâmetros	42 ± 22	126 ± 23	97 ± 25
$RMSE$ (Treino)	0,0907 ± 0,0050	0,0445 ± 0,0062	$\mathbf{0{,}0368 \pm 6{,}9978 \times 10^{-4}}$
$RMSE$ (Teste)	0,1280 ± 0,0060	0,0509 ± 0,0098	**0,0236 ± 0,0012**
P4(X)			
Estrutura mais frequente	247	334	124704
Parâmetros	89 ± 28	101 ± 26	75 ± 19
$RMSE$ (Treino)	1,3825 ± 0,0602	**0,7810 ± 0,0273**	0,8785 ± 0,0067
$RMSE$ (Teste)	1,8427 ± 0,9263	0,6553 ± 0,0442	**0,6508 ± 0,0467**
P4(Y)			
Estrutura mais frequente	244	334	127704 / 129404
Parâmetros	82 ± 26	96 ± 20	81 ± 17
$RMSE$ (Treino)	1,9819 ± 0,0781	1,2255 ± 0,0456	**1,1969 ± 0,0136**
$RMSE$ (Teste)	2,4162 ± 0,5115	1,1868 ± 0,0667	**0,9722 ± 0,0545**
P4(Z)			
Estrutura mais frequente	237	329	126604
Parâmetros	57 ± 22	71 ± 4	99 ± 25
$RMSE$ (Treino)	2,2570 ± 0,0766	**1,2676 ± 0,0235**	1,3614 ± 0,0420
$RMSE$ (Teste)	2,1401 ± 0,0828	1,3155 ± 0,0183	**1,3067 ± 0,3461**

O modelo DFRFS–DE obteve melhor desempenho que os modelos RFS–AG e RFS–DE em todos os problemas estudados, tanto nos dados de treino quanto nos dados de teste, com a exceção dos dados de treino nos problemas P4(X) e P4(Z), em que o modelo RFS–DE obteve melhore desempenho. Excetuando-se o problema P4(Z), os resultados do modelo DFRFS–DE em todos os problemas foram bastante consistentes, como pode ser observado nos pequenos valores do desvio padrão. Quanto à complexidade, o modelo RFS–AG foi geralmente mais parcimonioso, porém, com um desempenho bem inferior aos demais.

A fim de verificar se a diferença de desempenho entre o modelo DFRFS–DE e os modelos RFS–AG e RFS–DE é estatisticamente significante, o teste estatístico de Wilcoxon foi realizado. A Tabela 5.5 abaixo apresenta os valores de p para este teste. O resultado sugere que a hipótese nula deve ser rejeitada para o modelo RFS–AG em todos os problemas estudados com um alto nível de significância. Quanto ao modelo RFS–DE, no entanto, o resultado sugere a rejeição da hipótese nula nos problemas P1, P2, P3 e P4(Y). Em todos estes casos, com um alto nível de significância. Nos problemas P4(X) e P4(Z), contudo, o resultado sugere que a hipótese nula deve ser aceita com um bom nível de significância. Nestes dois problemas, o desempenho nos dados de treino foram melhores para o modelo RFS–DE que para o modelo DFRFS–DE.

Tabela 5.5. Valor de p obtido pelo teste de Wilcoxon.

Problema	DFRFS-DE vs. RFS-GA	DFRFS-DE vs. RFS-DE
P1	$1{,}7344 \times 10^{-6}$	$3{,}5152 \times 10^{-6}$
P2	$1{,}7344 \times 10^{-6}$	$2{,}3534 \times 10^{-6}$
P3	$1{,}7344 \times 10^{-6}$	$1{,}7344 \times 10^{-6}$
P4(X)	$1{,}7344 \times 10^{-6}$	0,8290
P4(Y)	$1{,}7344 \times 10^{-6}$	$1{,}7344 \times 10^{-6}$
P4(Z)	$2{,}1266 \times 10^{-6}$	0,5716

Pode-se afirmar, portanto, com significância estatística, que a abordagem proposta gera modelos com melhor desempenho que a abordagem original em todos os problemas estudados. Porém, quando a abordagem original utiliza o algoritmo de evolução diferencial ao invés do algoritmo genético, não se pode fazer este tipo de afirmação, em que pese o desempenho médio obtido pelo modelo DFRFS–DE ser melhor do que o desempenho médio obtido pelo modelo RFS–DE. Este resultado ratifica a superioridade da evolução diferencial sobre os algoritmos genéticos quando ambos são utilizados para otimizar os parâmetros do modelo desenvolvido por Gama *et al.* (2008).

Com relação ao processo de identificação proposto, a Figura 5.1 apresenta a distribuição das estruturas geradas na exploração do espaço de busca, durante o processo evolutivo. Ela evidencia tanto a capacidade de exploração do algoritmo de evolução diferencial utilizado para realizar a identificação da estrutura, quanto a importância dos dois mecanismos de verificação implementados na sua função

objetivo, ilustrada na Figura 3.7. A capacidade de exploração fica evidente se for levado em conta: o desempenho do modelo (Tabela 5.4) e a razão entre a quantidade de estruturas candidatas (Tabela B.7) e o número máximo de estruturas possíveis (Tabela 4.4).

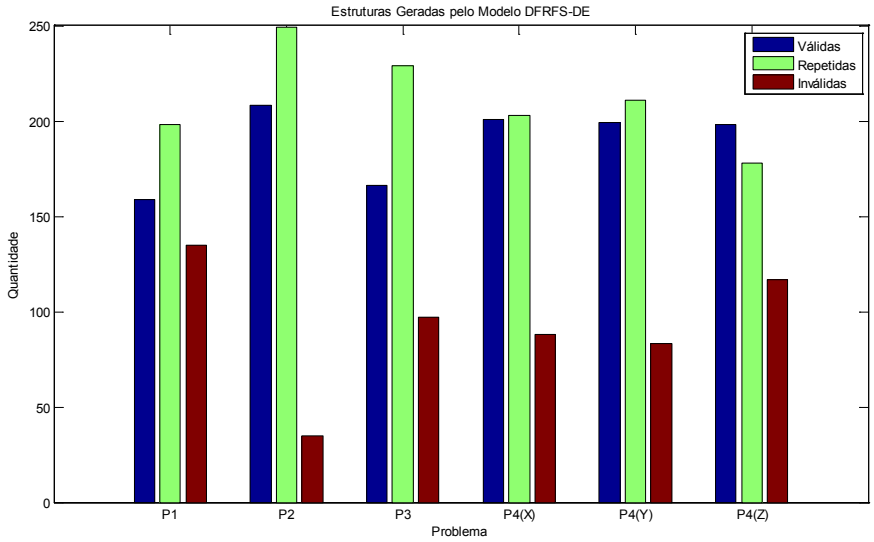


Figura 5.1. Quantidade de estruturas geradas pela abordagem proposta na exploração do espaço de busca.

Em um espaço de busca caracterizado por 6.340 estruturas viáveis para os problemas P1 e P2, e por 7.980 estruturas viáveis para os problemas P3, P4(X), P4(Y) e P4(Z), a taxa média de estruturas válidas geradas pela abordagem proposta para o primeiro caso representa um valor entre 0,5% e 4% do número máximo de estruturas viáveis, enquanto que, no segundo caso, essa taxa varia entre 1% e 3%. Estes valores associados ao desempenho final do modelo evidenciam a habilidade da evolução diferencial em selecionar regiões promissoras do espaço de busca.

Com relação aos dois mecanismos de verificação implementados na função objetivo, a Figura 5.1 mostra a importância de ambos no processo evolutivo. Das quase 25.000 possíveis estruturas inválidas do espaço de busca irrestrito para todos os problemas, menos de 1% delas foram geradas pelo algoritmo de evolução diferencial. Este valor destaca a habilidade do algoritmo em explorar de forma eficiente o espaço de busca.

Por fim, a quantidade de soluções válidas repetidas mostra a importância do mecanismo de armazenamento e recuperação da aptidão no tempo de execução do algoritmo, visto que evita a utilização de recurso computacional desnecessário uma

vez que as estruturas que já foram processadas não são reprocessadas. Com a exceção do problema P4(Z), a quantidade de estruturas repetidas supera a quantidade de estruturas únicas geradas durante a execução do algoritmo, mostrando a diferença que a implementação deste simples mecanismo faz na quantidade de avaliações da função objetivo.

5.3 Simulação 3: Tipo de Feedback Adicional

Os resultados referentes à simulação computacional descrita na seção 4.2.3, que teve por objetivo avaliar o desempenho dos modelos em termos do tipo de feedback adicional, são apresentados e discutidos nesta seção. A Tabela 5.6 na próxima página apresenta os resultados consolidados desta última simulação, cujos valores foram obtidos a partir das tabelas do Apêndice C, que apresentam os resultados detalhados da simulação. Os valores são referentes aos dados de teste.

Os melhores resultados foram obtidos pelos modelos que utilizam a saída observada como feedback adicional. Este resultado já era esperado, visto que a informação adicional utilizada por estes modelos é independente do seu comportamento. No caso dos outros dois tipos de feedback adicional, a informação utilizada pelo modelo é essencialmente dependente do seu comportamento e, portanto, contaminada com o erro do modelo. Os modelos obtidos com a utilização tanto da saída estimada quanto da saída observada são todos de 1ª ordem, exceto para o problema P1. Neste caso, eles são de 2ª ordem. Por outro lado, os modelos baseados no erro de predição são de 3ª ordem, exceto para os problemas P2 e P3, em que eles são de 2ª e 1ª ordem, respectivamente.

Geralmente, valores de $RMSE$ em torno de 5% do valor máximo da saída indicam um bom nível de desempenho do modelo, embora esse limite varie de acordo com o tipo de aplicação (EVSUKOFF et al., 2011). Nos problemas estudados, essa taxa foi de até 4% quando o feedback adicional foi representado pela saída estimada ou pela saída observada. Nos casos em que o erro foi usado como feedback adicional, no entanto, essa taxa variou dependendo do problema. Os melhores valores foram

obtidos nos problemas P2 e P4(Z), com uma taxa menor que 4%, enquanto que o pior foi obtido no problema P1. Neste caso, com uma taxa média em torno de 11%.

Tabela 5.6. Resultado comparativo entre os tipos de feedback nos dados de teste.

	Saída Estimada	**Saída Observada**	**Erro**
P1			
Estrutura mais frequente	232402	232402 / 232502	332203
Parâmetros	92 ± 16	92 ± 14	104 ± 14
$RMSE$	0,0286 ± 0,0081	**0,0256 ± 0,0063**	0,1114 ± 0,0821
$RMSE\%$	[2,67; 3,28]	[2,40; 2,84]	[9,07; 15,15]
P2			
Estrutura mais frequente	134331	133331	223331
Parâmetros	32 ± 6	29 ± 6	38 ± 8
$RMSE$	0,0220 ± 0,0124	**0,0217 ± 0,0098**	0,0220 ± 0,0405
$RMSE\%$	[2,31; 3,57]	[2,31; 3,21]	[1,41; 5,17]
P3			
Estrutura mais frequente	139302	159302	178202
Parâmetros	97 ± 25	114 ± 28	108 ± 24
$RMSE$	0,0236 ± 0,0012	**0,0225 ± 0,0042**	0,0740 ± 0,0156
$RMSE\%$	[1,82; 1,89]	[1,68; 1,93]	[5,40; 6,27]
P4(X)			
Estrutura mais frequente	124704	125921 / 126731 / 129503 / 129603	323331
Parâmetros	75 ± 19	91 ± 20	85 ± 20
$RMSE$	0,6508 ± 0,0467	**0,5486 ± 0,0328**	0,8235 ± 0,6023
$RMSE\%$	[3,49; 3,66]	[2,94; 3,07]	[3,86; 7,47]
P4(Y)			
Estrutura mais frequente	127704 / 129404	127721 / 127703	324321 / 324221
Parâmetros	81 ± 17	100 ± 17	99 ± 22
$RMSE$	0,9722 ± 0,0545	**0,9019 ± 0,0470**	1,5417 ± 1,6189
$RMSE\%$	[3,87; 4,03]	[3,60; 3,74]	[4,66; 8,53]
P4(Z)			
Estrutura mais frequente	126604	127704	326201
Parâmetros	99 ± 25	104 ± 22	100 ± 20
$RMSE$	1,3067 ± 0,3461	**0,8547 ± 0,0205**	1,3126 ± 0,2450
$RMSE\%$	[2,69; 3,23]	[1,87; 1,90]	[2,74; 3,12]

Com relação à complexidade dos modelos gerados, a utilização da saída estimada proporcionou a geração de modelos mais parcimoniosos.

O teste de Friedman com posto alinhado foi realizado a fim de verificar se a diferença de desempenho e de complexidade entre os modelos gerados com os diferentes tipos de feedback adicional possui significância estatística, a exemplo do que foi feito na análise dos resultados da primeira simulação. A Tabela 5.7 a seguir

apresenta os valores do posto médio e de p para este teste. O resultado sugere que a hipótese nula não pode ser rejeitada para nenhum dos critérios considerados.

Tabela 5.7. Valor do posto médio obtido pelo teste de Friedman – Simulação 3

Feedback Adicional	Desempenho	Complexidade
Saída estimada	8,5833	4,4167
Saída observada	5	11,6667
Erro	14,9167	12,4167
Valor p	0,1267	0,1138

O teste de hipótese indicou que nenhum método em particular é, ao mesmo tempo, melhor do que todos os outros. Porém, a diferença entre os valores do posto médio (Tabela 5.7) sugere que a saída observada proporciona melhor desempenho, e a saída estimada a menor complexidade. Portanto, o teste *post-hoc* de Finner foi realizado a fim de verificar se esta diferença possui significância estatística. O resultado deste teste é o valor ajustado de p (Tabela 5.8).

Tabela 5.8. Valor ajustado de p obtido pelo teste post-hoc de Finner – Simulação 3

Feedback Adicional	Desempenho	Complexidade
Saída estimada	0,2445	–
Saída observada	–	0,0188
Erro	0,0026	0,0188

O resultado do teste de Finner apresentado na tabela acima sugere que a hipótese nula com relação ao desempenho deve ser rejeitada para o erro e aceita para a saída estimada, indicando a superioridade dos modelos baseados na saída observada apenas sobre aqueles baseados no erro de predição. Quanto à complexidade, porém, a hipótese nula deve ser rejeitada tanto para o erro quanto para a saída observada, ratificando que os modelos baseados na saída estimada são mais parcimoniosos que aqueles gerados com base na saída observada ou no erro.

Na Figura 5.2 apresentada na próxima página, o desempenho dos modelos é apresentado segundo o tipo de feedback em função do índice de erro não dimensional (NDEI), que facilita a comparação do desempenho dos modelos em todos os problemas estudados ao mesmo tempo.

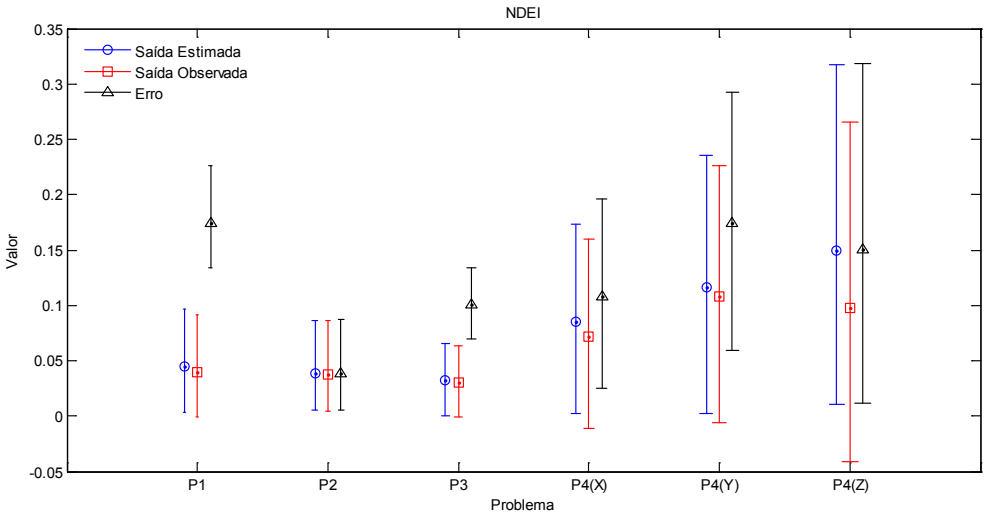


Figura 5.2. Desempenho dos modelos nos dados de teste segundo o índice de erro não dimensional (NDEI). Valores médios e seus respectivos intervalos de confiança (95%)

Como pode ser observado no gráfico, os modelos baseados tanto na saída estimada quanto na saída observada lidam melhor com os problemas P1, P2 e P3 do que com os problemas P4(X), P4(Y) e P4(Z). Além do melhor desempenho nos três primeiros problemas, os resultados ao longo das 30 execuções são mais consistentes nestes problemas, com um intervalo de confiança mais restrito do que nos três últimos problemas, em que o intervalo de confiança é mais abrangente.

Os modelos baseados no erro, por sua vez, tiveram o melhor resultado apenas no problema P2. Neste caso, o desempenho foi semelhante ao dos modelos baseados tanto na saída estimada quanto na observada. De um modo geral, o erro de predição não representa uma boa opção para o feedback adicional, pois o desempenho dos modelos baseados nele é sempre inferior ao desempenho dos modelos baseados na saída prevista. Por outro lado, o desempenho dos modelos baseados na saída estimada foi sempre próximo ao desempenho dos modelos baseados na saída observada, exceto para o problema P4(Z). Neste caso particular, seu desempenho foi pior, assemelhando-se ao desempenho dos modelos baseados no erro.

Este resultado sugere, ao menos para os problemas estudados, que sempre que a saída observada estiver disponível ela deve ser utilizada, pois proporciona a geração de modelos com melhor desempenho. Por outro lado, caso não esteja disponível, como é o caso nos problemas de simulação, a saída estimada é uma boa opção como feedback adicional, pois os modelos baseados nela apresentam um desempenho semelhante ao dos modelos que se valem da saída observada.

Uma possível explicação para o pior desempenho dos modelos baseados no erro de predição seria a inflexibilidade estrutural do sistema fuzzy recorrente, responsável por modelar o comportamento não linear do próprio erro do modelo e relacioná-lo com a saída. Na verdade, esta tarefa está longe de ser simples até mesmo para modelos mais robustos estruturalmente, devido ao alto grau de não linearidade intrínseco ao problema.

As figuras a seguir apresentam os gráficos da saída média dos modelos contra a saída real do sistema e o gráfico de erro dos modelos em cada um dos problemas estudados. A Figura 5.3 e a Figura 5.4 são referentes aos sistemas dinâmicos não lineares, e representam, respectivamente, o problema P1 e P2. A Figura 5.5 refere-se a série temporal caótica univariada de Hénon. A Figura 5.6, a Figura 5.7 e a Figura 5.8, por sua vez, são todas referentes à série temporal caótica multivariada de Lorenz, e representam, respectivamente, os problemas P4(X), P4(Y) e P4(Z).

Como pode ser observado na Figura 5.3 abaixo, o desempenho obtido com a saída estimada e a saída prevista é bem semelhante. O desempenho dos modelos baseados no erro de predição foram piores apenas em parte dos dados, especificamente entre os instantes 1300 e 1550, em que o comportamento do sistema se manteve constante, com o valor da saída caindo abruptamente até chegar próximo a -1, e subindo também abruptamente ao final deste período.

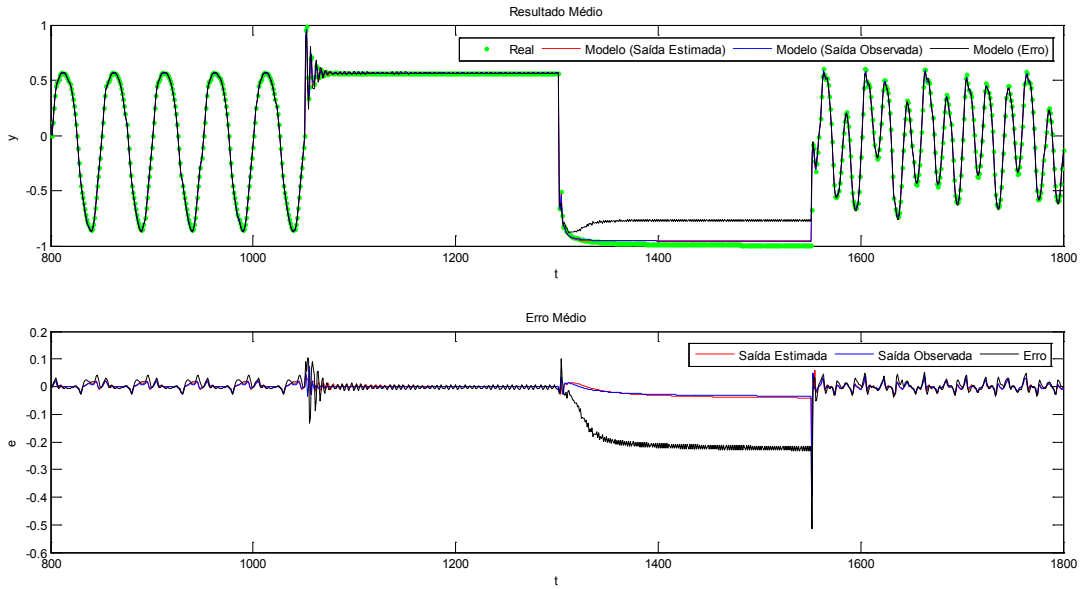


Figura 5.3. Desempenho médio dos modelos nos dados de teste – Problema P1

Os gráficos da Figura 5.4 abaixo mostram um desempenho equivalente entre os modelos, independente do tipo de feedback utilizado. Porém, o gráfico de erro evidencia o pior desempenho dos modelos baseados no erro quando o comportamento do sistema se mantém constante, ocasião em que a sua saída possui valores extremos.

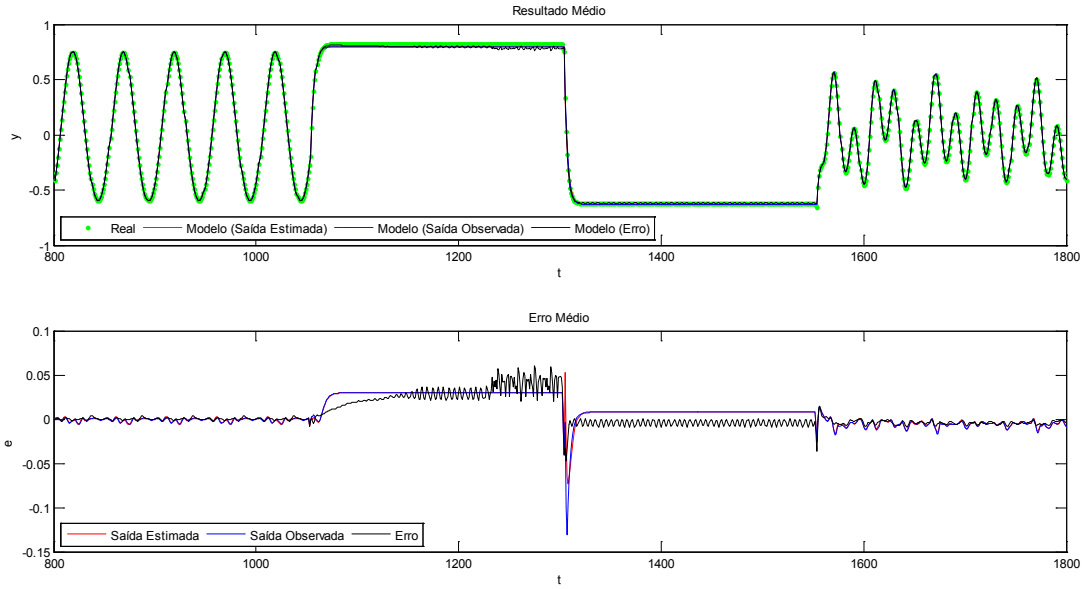


Figura 5.4. Desempenho médio dos modelos nos dados de teste – Problema P2

Os gráficos do espaço de fases e o gráfico de erro na Figura 5.5 abaixo mostram o bom desempenho dos modelos baseados tanto na saída estimada quanto na saída observada. A trajetória caótica do sistema reconstruída por esses modelos é bem próxima da trajetória real do sistema. Por outro lado, a trajetória reconstruída com base no erro de predição não foi tão boa quanto as demais.

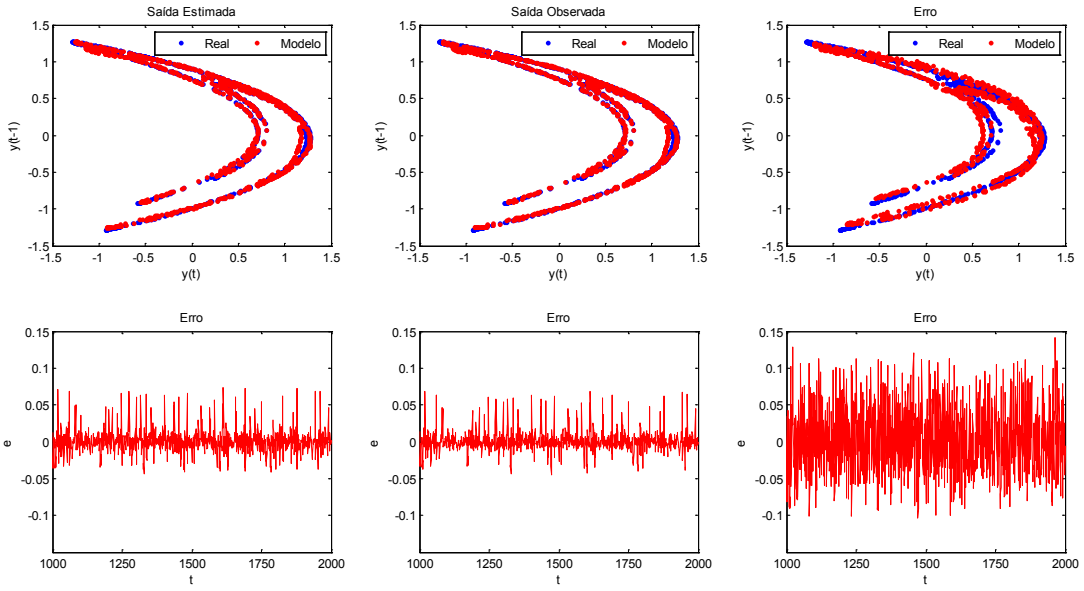


Figura 5.5. Desempenho médio dos modelos nos dados de teste – Problema P3

O desempenho dos modelos na série temporal caótica multivariada de Lorenz é mostrado nos gráficos das três figuras que se seguem. O desempenho dos modelos em todas as três dimensões do problema é qualitativamente equivalente independente do tipo de feedback utilizado, embora a Figura 5.2 e a Tabela 5.6 tenham evidenciado a diferença quantitativa no desempenho dos modelos dependendo do tipo de feedback utilizado. O desempenho dos modelos na dimensão X do sistema é mostrado nos gráficos da Figura 5.6, enquanto as dimensões Y e Z são tratadas, respectivamente, pela Figura 5.7 e pela Figura 5.8.

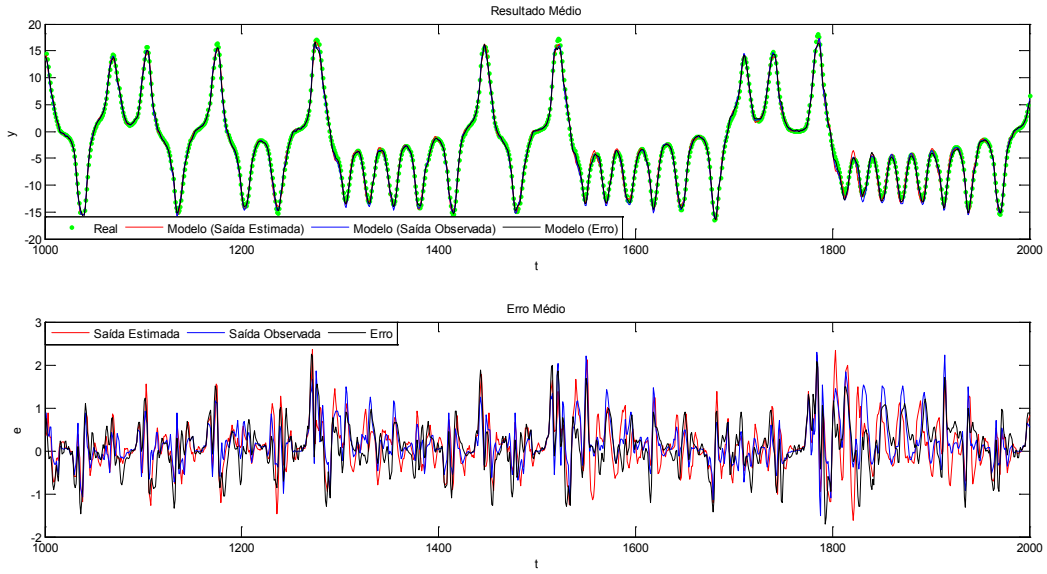


Figura 5.6. Desempenho médio dos modelos nos dados de teste – Problema P4(X)

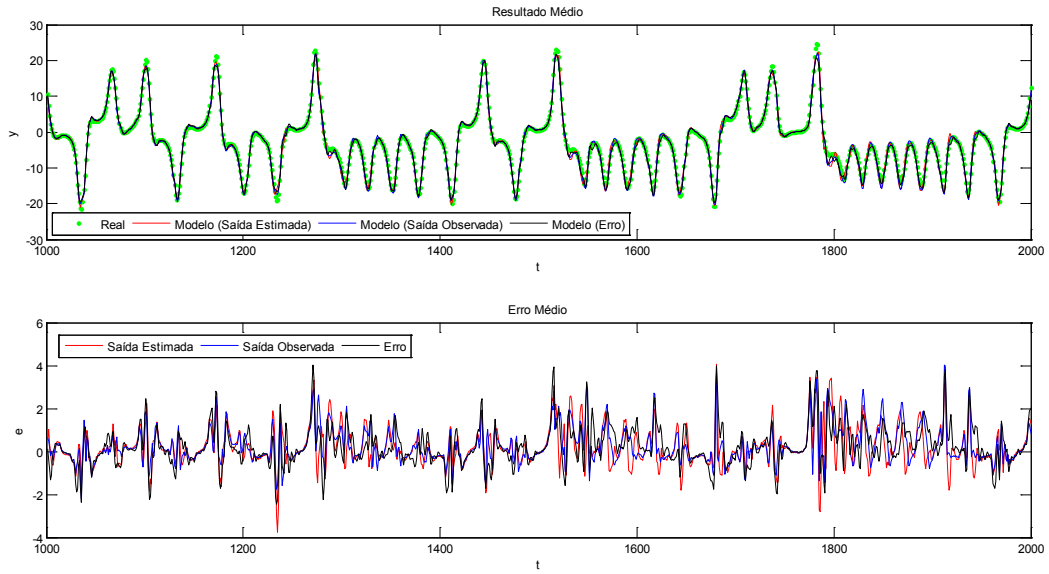


Figura 5.7. Desempenho médio dos modelos nos dados de teste – Problema P4(Y)

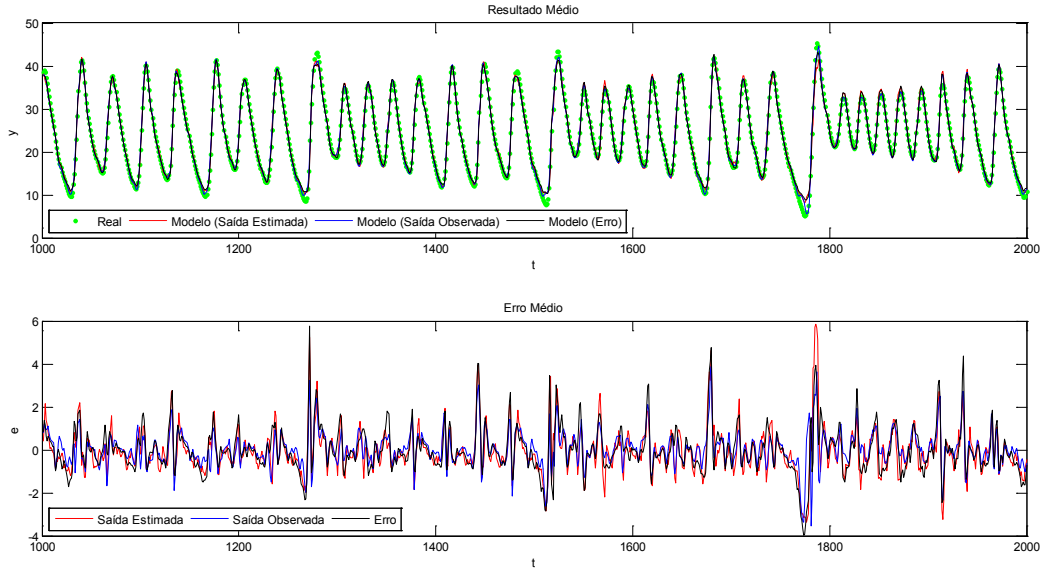


Figura 5.8. Desempenho médio dos modelos nos dados de teste – Problema P4(Z)

A Tabela 5.9 a seguir apresenta a comparação do número de parâmetros e do valor do $RMSE$ nos dados de teste dos problemas P1, P2 e P3, entre os modelos desenvolvidos e alguns dos modelos que foram apresentados na seção 2.2.1.2, a fim de contextualizar a abordagem proposta neste trabalho. Os modelos desenvolvidos estão representados como *DFRFS-DE*($\hat{y}$), *DFRFS-DE*(y) e *DFRFS-DE*(e), de acordo com o tipo de feedback utilizado na sua estrutura.

Com relação à complexidade, percebe-se que a abordagem proposta tende a gerar modelos menos parcimoniosos que os modelos considerados na comparação, principalmente nos problemas P1 e P3. No problema P2, no entanto, a diferença do número de parâmetros entre os modelos não é tão marcante. Dentre os modelos analisados, aquele que apresentou a menor complexidade é o *RFCMAC*, cujo desempenho também foi o melhor entre todos os modelos.

Os modelos gerados segundo a abordagem proposta são menos parcimoniosos que os demais modelos analisados. Isto se dá porque a abordagem proposta considera uma base de regras completa, com particionamento regular. Por outro lado, a abordagem utilizada pelos outros modelos, excetuando-se o modelo *RFS–TSK*, considera a geração iterativa de uma base de regras mínima, que leva em consideração a distribuição dos valores das variáveis utilizadas pelo modelo. Embora o modelo *RFS–TSK* também utilize a mesma abordagem de geração das regras, o número de variáveis utilizadas é menor e, portanto, o número de regras tende a ser reduzido.

Tabela 5.9. Resultado comparativo dos modelos desenvolvidos com a literatura

Modelo	**Problema P1**		**Problema P2**		**Problema P3**	
	Parâmetros	*RMSE*	*Parâmetros*	*RMSE*	*Parâmetros*	*RMSE*
DFRFS-DE ($\hat{y}$)	92	0,0286	32	0,0220	97	0,0236
DFRFS-DE (y)	92	0,0256	29	0,0217	114	0,0225
DFRFS-DE (e)	104	0,1114	38	0,0220	108	0,0740
IRSFNN (LIN, Y.-Y.; CHANG; LIN, 2013)	42	0,0310	26	0,0220	40	0,0140
MRIT2NFS (LIN, Y.-Y.; CHANG; PAL; et al., 2013)	40	0,0078	40	0,0028	57	**0,0023**
RIFNN (JUANG et al., 2011)	–	–	36	0,0288	54	0,0510
LRFNN–SVR (JUANG; HSIEH, 2010)	29	0,0296	29	0,0306	31	0,0155
RSEFNN–LF (JUANG et al., 2010)	34	0,0383	30	0,0279	94	0,0031
RSETI2FNN (JUANG; HUANG; et al., 2009)	–	–	42	0,0058	60	0,0047
RFCMAC (LIN, C.-J.; LEE, 2008)	24	**0,0013**	24	**0,0012**	–	–
RFS-TSK (GAMA et al., 2008)	36	0,0685	54	0,0880	32	0,1048
RCNFS (LIN, C.-J.; CHEN, 2005)	–	–	27	0,0221	–	0,0035
TRFN (JUANG, 2002)	33	0,0346	33	0,0313	36	0,0206
RFNN (LEE, C.-H.; TENG, 2000)	112	0,0114	–	–	60	0,0469

A Figura 5.9 na próxima página apresenta uma comparação qualitativa do desempenho entre os modelos apresentados na Tabela 5.9. As linhas horizontais que cruzam as barras verticais no gráfico destacam o valor do desempenho do modelo *DFRFS-DE*(y), que obteve o melhor desempenho dentre os modelos desenvolvidos, facilitando a comparação visual com os demais modelos nos respectivos problemas.

Dos modelos desenvolvidos, apenas os modelos *DFRFS-DE*($\hat{y}$) e *DFRFS-DE*(y) conseguiram um desempenho competitivo com os modelos da literatura. De um modo geral, pode-se dizer que o desempenho deles foi consideravelmente pior apenas do que o desempenho dos modelos *MRIT2NFS, RFCMAC* e *RSETI2FNN.* A taxa entre o $RMSE$ desses modelos e o valor máximo da saída real não chega sequer a 1% em qualquer dos problemas estudados. Por outro lado, a taxa dos modelos *DFRFS-DE*($\hat{y}$) e *DFRFS-DE*(y) é de quase 3% no problema P1 e P2, e quase 2% no problema P3.

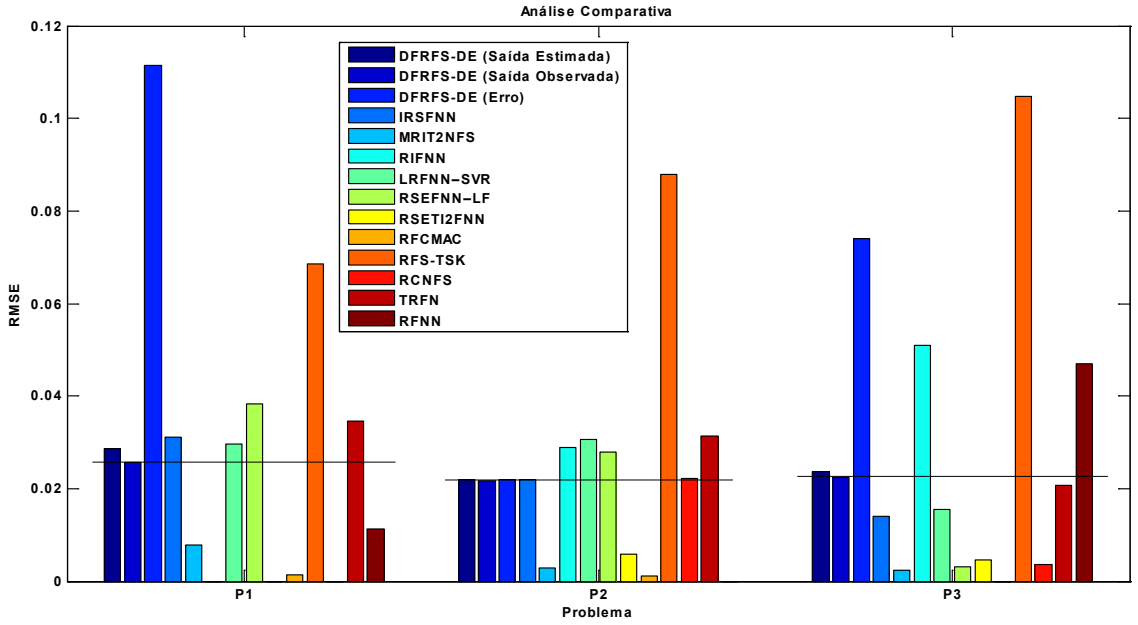


Figura 5.9. Gráfico comparativo do desempenho dos modelos desenvolvidos com a literatura

A Tabela 5.10 a seguir apresenta o resumo comparativo entre o desempenho de todos os modelos nos três problemas considerados a fim de facilitar a análise dos resultados. De acordo com ela, o modelo *DFRFS-DE*(e) pode ser considerado um pouco melhor do que o modelo *RFS-TSK*, visto que seu desempenho é melhor em dois dos três problemas estudados, enquanto ele pode ser considerado equivalente ao modelo *RIFNN*, uma vez que o modelo desenvolvido foi melhor do que ele no problema P2 e pior no P3. Em comparação aos demais modelos, o modelo *DFRFS-DE*(e) teve o desempenho pior em pelo menos dois dos três problemas, sendo, portando, considerado pior do que eles.

O resultado da análise comparativa entre os modelos é bem semelhante para os modelos *DFRFS-DE*($\hat{y}$) e *DFRFS-DE*(y) nos problemas estudados. Eles podem ser considerados melhores do que os seguintes modelos: *RIFNN, RSEFNN–LF, RFS-TSK* e *TRFN*, pois obtiveram melhor desempenho em pelo menos dois dos três problemas. Com relação ao modelo *LRFNN–SVR*, pode-se dizer que o modelo DFRFS-DE(y) é melhor do que ele pela mesma razão, enquanto ele é considerado equivalente ao modelo *DFRFS-DE*($\hat{y}$), pois ambos foram melhores em apenas um problema e tiverem desempenho equivalente no outro. O modelo *IRSFNN* é considerado equivalente ao modelo *DFRFS-DE*(y) e melhor que o modelo *DFRFS-DE*($\hat{y}$), enquanto os modelos *RFNN* e *RCNFS* podem ser considerados, respectivamente, equivalentes a e melhores que ambos os modelos desenvolvidos.

Tabela 5.10. Resumo comparativo entre o desempenho dos modelos. Os símbolos +, – e ≅ indicam que os modelos propostos possuem desempenho, respectivamente, melhor, pior e equivalente aos modelos da literatura, nos respectivos problemas.

Modelos	DFRFS-DE($\hat{y}$)			DFRFS-DE(y)			DFRFS-DE(e)		
	P1	***P2***	***P3***	***P1***	***P2***	***P3***	***P1***	***P2***	***P3***
IRSFNN	≅	≅	–	+	≅	–	–	≅	–
MRIT2NFS	–	–	–	–	–	–	–	–	–
RIFNN		+	+		+	+		+	–
LRFNN–SVR	≅	+	–	+	+	–	–	+	–
RSEFNN–LF	+	+	–	+	+	–	–	+	–
RSETI2FNN		–	–		–	–		–	–
RFCMAC	–	–		–	–		–	–	
RFS-TSK	+	+	+	+	+	+	–	+	+
RCNFS		≅	–		≅	–		≅	–
TRFN	+	+	≅	+	+	≅	–	+	–
RFNN	–		+	–		+	–		–

Algumas observações importantes precisam ser feitas com relação ao resultado da análise comparativa entre os modelos. Em primeiro lugar, não foi possível realizar o teste de significância estatística para a diferença de desempenho entre os modelos comparados por não haver dados suficientes.

Todos os modelos analisados utilizam a saída observada do sistema como regressor, com exceção do modelo *RFS-TSK*, que é um modelo puramente de simulação, assim como o modelo *DFRFS-DE*($\hat{y}$) desenvolvido. Portanto, à luz desta particularidade, pode-se dizer que o modelo *DFRFS-DE*($\hat{y}$) é ainda mais competitivo que os modelos *IRSFNN, RIFNN, LRFNN-SVR, RSEFNN-LF, TRFN e RCNFS*, por não se valer da informação da saída observada do sistema e ainda assim obter bom desempenho.

Excluindo-se o modelo *RFS-TSK*, cujo aprendizado também é evolucionário, todos os demais modelos considerados na comparação utilizam o aprendizado neural. Destes, apenas o modelo *LRFNN-SVR* não é baseado no gradiente. Dentre os modelos apresentados na Tabela 5.9 e na Tabela 5.10, apenas os modelos *RFS-TSK* e *TRFN* possuem recorrência externa, caracterizada pela realimentação global do estado. Todos os demais utilizam uma estrutura com recorrência interna. Dentre estes, os modelos *IRSFNN* e *MRIT2NFS* são globalmente recorrentes e, como os modelos *RIFNN, LRFNN-SVR* e *RSETI2FNN*, implementam a recorrência através da realimentação do

valor de ativação das regras. Os demais modelos são localmente recorrentes. A recorrência dos modelos *RFCMAC* e *RCNFS* é implementada nas funções de pertinência.

Com a exceção do modelo *RFS–TSK* que foi estendido, o qual, assim como os modelos propostos, possui uma estrutura simples e utiliza regras fuzzy do tipo TSK de ordem zero, todos os demais modelos geram regras fuzzy cuja interpretação é prejudicada em função da estrutura do modelo. Na verdade, a própria recorrência já minimiza a legibilidade das regras, mesmo para o modelo *RFS–TSK*. Porém, a abordagem apresentada em (GAMA et al., 2008) propõe uma metodologia que possibilita a representação de sistemas fuzzy recorrentes do tipo TSK de ordem zero como autômatos finitos. Esta mesma metodologia pode ser aplicada aos modelos desenvolvidos, melhorando assim a sua interpretabilidade que, diferentemente dos outros modelos, não foi tão comprometida pela estrutura do modelo.

A seguir, serão apresentadas as considerações finais a respeito deste trabalho de pesquisa, que teve por objetivo desenvolver novos modelos baseados no sistema fuzzy recorrente que foi apresentado em (GAMA et al., 2008), visando a melhoraria de desempenho. Este objetivo foi satisfatoriamente alcançado, como foi evidenciado e discutido neste capítulo, através dos resultados obtidos nas simulações computacionais.

Capítulo 6

6 Conclusão

Este trabalho apresentou o desenvolvimento de novas estruturas para o sistema fuzzy recorrente proposto por Gama *et al.* (2008), e de uma nova metodologia para identificação simultânea da estrutura e dos parâmetros, baseada no algoritmo de evolução diferencial. Assim como na abordagem original, os novos modelos desenvolvidos são formulados através das equações do espaço de estados, em que a função de transição de estados é representada por um sistema fuzzy recorrente e a função de saída é uma função linear dos estados.

As novas estruturas são caracterizadas pela utilização de duas conexões de feedback. Uma delas, herdada do modelo original, é representada pelos estados. A outra pode vir do próprio modelo (saída prevista); do sistema dinâmico (saída observada); bem como indiretamente a partir de ambos (erro de predição). Além disso, operadores de defasagem ajustáveis são utilizados a fim de melhorar o desempenho do modelo quando este estiver representando sistemas dinâmicos com atraso.

Outrossim, uma nova abordagem de identificação foi desenvolvida, utilizando-se duas instâncias do algoritmo de evolução diferencial de maneira hierárquica. Uma delas responsável pela identificação da estrutura do modelo e a outra pela identificação dos seus parâmetros. Esta abordagem permitiu lidar com ambos os problemas de identificação simultaneamente. Além disso, a evolução diferencial mostrou ser um método de otimização bastante flexível, pois foi utilizada para lidar

tanto com um problema de otimização combinatória (identificação da estrutura), quanto um problema de otimização contínua (identificação dos parâmetros), sem nenhum tipo de alteração na sua estrutura interna. Como já era esperado, a evolução diferencial superou o desempenho dos algoritmos genéticos nos problemas estudados, confirmando as afirmações de superioridade recentemente apresentadas na literatura.

Os resultados obtidos nas simulações evidenciaram a capacidade preditiva dos modelos desenvolvidos, ratificando a eficácia de estruturas recorrentes na modelagem de sistemas dinâmicos. Adicionalmente, a abordagem proposta se mostrou tão boa quanto aquelas utilizadas pelos modelos considerados na comparação dos resultados, visto que o desempenho dos novos modelos foi equivalente ao desempenho daqueles modelos, de um modo geral. Os únicos modelos com desempenho realmente superior foram os modelos *MRIT2NFS* (LIN, Y.-Y.; CHANG; PAL; et al., 2013), *RSETI2FNN* (JUANG; HUANG; et al., 2009), e *RFCMAC* (LIN, C.-J.; LEE, 2008). Os dois primeiros utilizam funções de pertinência intervalares do tipo 2, que lidam melhor com incertezas; enquanto o último utiliza uma arquitetura baseada no funcionamento do cerebelo humano, que é conhecida por sua notável capacidade de aprendizado.

Em que pese a pesquisa bibliográfica apresentada nesta tese ter descortinado novos horizontes a serem explorados, principalmente com relação à forma com que a recorrência é implementada, este trabalho se restringiu a avaliar a utilização de outros regressores que não haviam sido considerados na estrutura original do modelo desenvolvido por Gama et al. (2008), levando à criação de novas estruturas. Duas, das três novas estruturas, permitiram estender a utilização do modelo para problemas de predição; enquanto a outra, que ganhou uma nova camada de recorrência, melhorou o desempenho do modelo quando configurado para simulação.

Embora a abordagem proposta neste trabalho tenha apresentado bons resultados nos problemas de benchmark, que foram consideradas nas simulações computacionais, é preciso avaliar o seu potencial em outros tipos de problemas, sobretudo problemas reais. Na próxima página, são apresentadas as características mais relevantes que devem ser priorizadas na continuação deste trabalho.

Uma das maiores limitações dos modelos desenvolvidos, se não a maior delas, que foi herdada da abordagem originalmente proposta por Gama et al. (2008), é a

utilização de uma base de regras completa, gerada em função do particionamento regular das variáveis utilizadas pelo modelo, pois o número de parâmetros cresce com o número de regras, que por sua vez também cresce com o número de variáveis. Esta característica limita a utilização da abordagem proposta em sistemas com muitas entradas. Na verdade, sistemas MISO com mais de 2 ou 3 entradas já inviabilizaria a geração de modelos mais estruturalmente flexíveis, principalmente se um pequeno conjunto de dados for utilizado no treinamento do modelo.

Portanto, a primeira e mais importante direção futura para este trabalho seria a utilização de uma base de regras reduzida, a exemplo do que acontece com a grande maioria dos sistemas fuzzy recorrentes que foram apresentados na Tabela 2.3. Uma das abordagens consideradas por esses sistemas é a geração iterativa das regras de acordo com os valores das variáveis, em que apenas as regras ativadas com um alto grau são consideradas. Outras abordagens utilizam funções de pertinência multidimensionais, baseadas em algoritmos de agrupamento, que levam em consideração a dispersão dos dados, a fim de gerar regras apenas nas regiões de grande densidade. Outra possibilidade, ainda, é realizar a poda das regras após a geração da base de regras.

Outro aspecto a ser explorado é a recorrência da estrutura. Neste caso, tanto a origem quanto o escopo devem ser investigados, a fim de levantar as potencialidades e limitações de cada uma delas e tentar definir qual seria a configuração mais indicada. Uma das limitações do tipo de recorrência utilizado neste trabalho é o aumento da dimensionalidade do problema. De acordo com a Figura 2.20, a maioria dos modelos que representam o estado da arte utiliza a recorrência definida nas funções de pertinência, e a Tabela 2.3 mostra que em todos esses modelos, excetuando-se o modelo *DRFNN* (HSU; CHENG, 2008), a recorrência tem um escopo local, de modo que um número menor de parâmetros precisa ser ajustado se comparado a modelos com recorrência global.

Referências

ABONYI, J. **Fuzzy model identification for control**. Boston: Birkhäuser, 2003.

ADAMY, J.; SCHWUNG, A. Qualitative modeling of dynamical systems employing continuous-time recurrent fuzzy systems. **Fuzzy Sets and Systems,** v. 161, n. 23, p. 3026-3043, 2010.

ALCALÁ-FDEZ, J. et al. KEEL: a software tool to assess evolutionary algorithms for data mining problems. **Soft Computing,** v. 13, n. 3, p. 307-318, 2009.

ALI, M.; PANT, M.; ABRAHAM, A. Unconventional initialization methods for differential evolution. **Applied Mathematics and Computation,** v. 219, n. 9, p. 4474-4494, 2013.

ALIEV, R. A. et al. Evolutionary algorithm-based learning of fuzzy neural networks. Part 2: Recurrent fuzzy neural networks. **Fuzzy Sets and Systems,** v. 160, n. 17, p. 2553-2566, 2009.

ALIEV, R. A. et al. Type-2 fuzzy neural networks with fuzzy clustering and differential evolution optimization. **Information Sciences,** v. 181, n. 9, p. 1591-1608, 2011.

ÅSTRÖM, K. J.; EYKHOFF, P. System identification - A survey. **Automatica,** v. 7, n. 2, p. 123-162, 1971.

BALLINI, R.; GOMIDE, F. Recurrent fuzzy neural computation: modeling, learning and application. In: **2010 IEEE International Conference on Fuzzy Systems**, Barcelona, Spain. 2010.

BARUCH, I. S.; HERNANDEZ, S. M. Decentralized direct I-term fuzzy-neural control of an anaerobic digestion bioprocess plant. In: **2011 IEEE Symposium on Computational Intelligence in Control and Automation**, Paris, France. 2011.

BASHARAT, A.; SHAH, M. Time series prediction by chaotic modeling of nonlinear dynamical systems. In: **2009 IEEE International Conference on Computer Vision**, Kyoto, Japan. 2009.

BELLMAN, R. E. **Adaptive control processes: a guided tour**. New Jersey: Princeton University Press, 1961.

BOX, G. E. P.; JENKINS, G. M. **Time series analysis, forecasting and control**. San Francisco, USA: Holden-Day, 1970.

CASDAGLI, M. Nonlinear prediction of chaotic time series. **Physica D: Nonlinear Phenomena,** v. 35, n. 3, p. 335-356, 1989.

CHANG, C. S.; XU, D. Y.; QUEK, H. B. Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system. **IEE Proceedings - Electric Power Applications,** v. 146, n. 5, p. 577-583, 1999.

CHANG, Y.-J.; HO, C.-L. Compact self-constructing recurrent fuzzy neural network with decision feedback for quadrature amplitude modulation signaling systems. **International Journal of Adaptive Control and Signal Processing,** v. 25, n. 12, p. 1087-1099, 2011.

CHEN, C.-H. Design of TSK-type fuzzy controllers using differential evolution with adaptive mutation strategy for nonlinear system control. **Applied Mathematics and Computation,** v. 219, n. 15, p. 8277-8294, 2013.

CHEN, C.-H.; LIN, C.-J.; LIN, C.-T. Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution. **IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,** v. 39, n. 4, p. 459-473, 2009.

CHEN, C.-H.; YANG, S.-Y. A knowledge-based cooperative differential evolution for neural fuzzy inference systems. **Soft Computing,** v. 17, n. 5, p. 883-895, 2013.

CHEN, C.-S. TSK-type self-organizing recurrent-neural-fuzzy control of linear microstepping motor drives. **IEEE Transactions on Power Electronics,** v. 25, n. 9, p. 2253-2265, 2010.

CHEN, C.; INOUE, Y.; SHIBATA, K. Identification of a golf swing robot using soft computing approach. **Neural Computing and Applications,** v. 20, n. 5, p. 729-740, 2011.

CHEN, C.; RICHARDSON, P. Mobile robot obstacle avoidance using short memory: A dynamic recurrent neuro-fuzzy approach. **Transactions of the Institute of Measurement and Control,** v. 34, n. 2-3, p. 148-164, 2012.

CHEN, S.; BILLINGS, S. A. Neural networks for nonlinear dynamic system modelling and identification. **International Journal of Control,** v. 56, p. 319-346, 1992.

CHEN, T.-C.; REN, T.-J.; LOU, Y.-W. Ultrasonic motor control based on recurrent fuzzy neural network controller and general regression neural network controller. In: MADANI, K.;DOURADO, A.*, et al* (Ed.). **Computational Intelligence**, v.465, 2013. p.291-305. (Studies in Computational Intelligence).

CHEONG, F.; LAI, R. Designing a hierarchical fuzzy logic controller using differential evolution. In: **1999 IEEE International Conference on Fuzzy Systems**, Seoul, South Korea. 1999.

CHIANG, H.-K.; CHU, C.-T.; JHOU, Y.-T. Fuzzy control with fuzzy basis function neural network in magnetic bearing system. In: **2012 IEEE International Symposium on Industrial Electronics**, Hangzhou, China. 2012.

CORDÓN, O. et al. Ten years of genetic fuzzy systems: current framework and new trends. **Fuzzy Sets and Systems,** v. 141, n. 1, p. 5-31, 2004.

DA SILVA, E. K.; BARBOSA, H. J. C.; LEMONGE, A. C. C. An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization. **Optimization and Engineering,** v. 12, n. 1-2, p. 31-54, 2011.

DAS, S.; SUGANTHAN, P. N. Differential evolution: a survey of the state-of-the-art. **IEEE Transactions on Evolutionary Computation,** v. 15, n. 1, p. 4-31, 2011.

DERRAC, J. et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. **Swarm and Evolutionary Computation,** v. 1, n. 1, p. 3-18, 2011.

DICICCIO, T. J.; EFRON, B. Bootstrap confidence intervals. **Statistical Science,** v. 11, n. 3, p. 189-212, 1996.

EFTEKHARI, M. et al. Eliciting transparent fuzzy model using differential evolution. **Applied Soft Computing,** v. 8, n. 1, p. 466-476, 2008.

EVSUKOFF, A. G.; CATALDI, M.; DE LIMA, B. S. L. P. A multi-model approach for long-term runoff modeling using rainfall forecasts. **Expert Systems with Applications,** v. 39, n. 5, p. 4938-4946, 2012.

EVSUKOFF, A. G.; DE LIMA, B. S. L. P.; EBECKEN, N. F. F. Long-term runoff modeling using rainfall forecasts with application to the Iguaçu river basin. **Water Resources Management,** v. 25, n. 3, p. 963-985, 2011.

EVSUKOFF, A. G.; EBECKEN, N. F. F. Identification of recurrent fuzzy systems with genetic algorithms. In: **2004 IEEE International Conference on Fuzzy Systems**, Budapest, Hungary. 2004.

FARMER, J. D.; SIDOROWICH, J. J. Predicting chaotic time series. **Physical Review Letters,** v. 59, n. 8, p. 845-848, 1987.

FAZZOLARI, M. et al. A Review of the application of multiobjective evolutionary fuzzy systems: current status and further directions. **IEEE Transactions on Fuzzy Systems,** v. 21, n. 1, p. 45-65, 2013.

FEOKTISTOV, V. **Differential evolution: in search of solutions**. Springer-Verlag New York, Inc., 2006.

FIGUEIREDO, M.; GOMIDE, F. Design of fuzzy systems using neurofuzzy networks. **IEEE Transactions on Neural Networks,** v. 10, n. 4, p. 815-827, 1999.

GACTO, M. J.; ALCALÁ, R.; HERRERA, F. Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. **Information Sciences,** v. 181, n. 20, p. 4340-4360, 2011.

GAMA, C. A. et al. Parameter identification of recurrent fuzzy systems with fuzzy finite-state automata representation. **IEEE Transactions on Fuzzy Systems,** v. 16, n. 1, p. 213-224, 2008.

GARCÍA, S. et al. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. **Information Sciences,** v. 180, n. 10, p. 2044-2064, 2010.

GARRIDO, A. A brief history of fuzzy logic. **Broad Research in Artificial Intelligence and Neuroscience,** v. 3, n. 1, p. 71-77, 2012.

GE, C. et al. Application of dynamic recurrent neural network in power system short-term load forecasting. In: **2010 International Conference on Computing Control and Industrial Engineering**, Wuhan, China. 2010.

GEVERS, M. A personal view of the development of system identification: A 30-year journey through an exciting field. **IEEE Control Systems Magazine,** v. 26, n. 6, p. 93-105, 2006.

GIANNAKIS, G. B.; SERPEDIN, E. A bibliography on nonlinear system identification. **Signal Processing,** v. 81, n. 3, p. 533-580, 2001.

GOLDBERG, D. E. **Genetic algorithms in search, optimization and machine learning**. Addison-Wesley Longman Publishing Co., Inc., 1989. 372

GONG, T.; TUSON, A. L. Differential evolution for binary encoding. In: SAAD, A.;DAHAL, K.*, et al* (Ed.). **Soft Computing in Industrial Applications**: Springer Berlin Heidelberg, v.39, 2007. cap. 24, p.251-262. (Advances in Soft Computing).

GONZALEZ-OLVERA, M. A.; TANG, Y. A new recurrent neurofuzzy network for identification of dynamic systems. **Fuzzy Sets and Systems,** v. 158, n. 10, p. 1023-1035, 2007.

GORRINI, V.; BERSINI, H. Recurrent fuzzy systems. In: **1994 IEEE International Conference on Fuzzy Systems**, Orlando, USA. 1994.

HABER, R.; UNBEHAUEN, H. Structure identification of nonlinear dynamic systems—A survey on input/output approaches. **Automatica,** v. 26, n. 4, p. 651-677, 1990.

HAMED, R. I. Inferring gene interactions from microarray gene expression data using fuzzy Petri net. In: **2010 IEEE International Conference on Communication Control and Computing Technologies**, Ramanathapuram, India. 2010.

HAN, M.-F.; LIN, C.-T.; CHANG, J.-Y. Differential evolution with local information for neuro-fuzzy systems optimisation. **Knowledge-Based Systems,** v. 44, n. 0, p. 78-89, 2013.

HELL, M.; COSTA, P.; GOMIDE, F. Recurrent Neurofuzzy Network in Thermal Modeling of Power Transformers. **IEEE Transactions on Power Delivery,** v. 22, n. 2, p. 904-910, 2007.

HELLENDOORN, H.; DRIANKOV, D. **Fuzzy model identification: selected approaches**. Springer, 1997. ISBN 9783540627210.

HÉNON, M. A two-dimensional mapping with a strange attractor. **Communications in Mathematical Physics,** v. 50, n. 1, p. 69-77, 1976.

HERRERA, F. Genetic fuzzy systems: taxonomy, current research trends and prospects. **Evolutionary Intelligence,** v. 1, n. 1, p. 27-46, 2008.

HONG, X. et al. Model selection approaches for non-linear system identification: a review. **International Journal of Systems Science,** v. 39, n. 10, p. 925-946, 2008.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks,** v. 2, n. 5, p. 359-366, 1989.

HSU, C.-F.; CHENG, K.-H. Recurrent fuzzy-neural approach for nonlinear control using dynamic structure learning scheme. **Neurocomputing,** v. 71, n. 16-18, p. 3447-3459, 2008.

HU, J. W. S.; HU, Y. C.; LIN, R. R. W. Applying neural networks to prices prediction of crude oil futures. **Mathematical Problems in Engineering,** v. 2012, n. Article ID 959040, 2012.

HU, Y.; RAMAMOORTHY, P. A. Design of stable recurrent fuzzy-logic-controllers. In: **1994 IEEE International Conference on Fuzzy Systems**, Orlando, USA. 1994.

HUNT, K. J.; HAAS, R.; MURRAY-SMITH, R. Extending the functional equivalence of radial basis function networks and fuzzy inference systems. **IEEE Transactions on Neural Networks,** v. 7, n. 3, p. 776-781, 1996.

JANG, J. S. R. ANFIS: adaptive-network-based fuzzy inference system. **IEEE Transactions on Systems, Man and Cybernetics,** v. 23, n. 3, p. 665-685, 1993.

JANG, J. S. R.; SUN, C.-T. Functional equivalence between radial basis function networks and fuzzy inference systems. **IEEE Transactions on Neural Networks,** v. 4, n. 1, p. 156-159, 1993.

_____. Neuro-fuzzy modeling and control. **Proceedings of the IEEE,** v. 83, n. 3, p. 378-406, 1995.

JASSAR, S.; LIAO, Z.; ZHAO, L. A recurrent neuro-fuzzy system and its application in inferential sensing. **Applied Soft Computing,** v. 11, n. 3, p. 2935-2945, Apr 2011.

JIN-HWAN, K.; UK-YOUL, H. Fuzzy model based predictive control. In: **1998 IEEE International Conference on Fuzzy Systems**, 1998, Anchorage, AK, USA. 1998.

JUANG, C.-F. A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. **IEEE Transactions on Fuzzy Systems,** v. 10, n. 2, p. 155-170, 2002.

JUANG, C.-F.; CHANG, P.-H. Recurrent fuzzy system design using elite-guided continuous ant colony optimization. **Applied Soft Computing,** v. 11, n. 2, p. 2687-2697, Mar 2011.

JUANG, C.-F.; HSIEH, C.-D. A locally recurrent fuzzy neural network with support vector regression for dynamic-system modeling. **IEEE Transactions on Fuzzy Systems,** v. 18, n. 2, p. 261-273, 2010.

JUANG, C.-F.; HUANG, R.-B.; LIN, Y.-Y. A recurrent self-evolving interval type-2 fuzzy neural network for dynamic system processing. **IEEE Transactions on Fuzzy Systems,** v. 17, n. 5, p. 1092-1105, 2009.

JUANG, C.-F.; LAI, C.-L.; TU, C.-C. Dynamic programming prediction errors of recurrent neural fuzzy networks for speech recognition. **Expert Systems with Applications,** v. 36, n. 3 PART 2, p. 6368-6374, 2009.

JUANG, C.-F.; LIN, Y.-Y.; HUANG, R.-B. Dynamic system modeling using a recurrent interval-valued fuzzy neural network and its hardware implementation. **Fuzzy Sets and Systems,** v. 179, n. 1, p. 83-99, 2011.

JUANG, C.-F.; LIN, Y.-Y.; TU, C.-C. A recurrent self-evolving fuzzy neural network with local feedbacks and its application to dynamic system processing. **Fuzzy Sets and Systems,** v. 161, n. 19, p. 2552-2568, 2010.

JUDITSKY, A. et al. Nonlinear black-box models in system identification: mathematical foundations. **Automatica,** v. 31, n. 12, p. 1725-1750, 1995.

KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. **Transactions of the ASME - Journal of Basic Engineering,** v. 82, n. Series D, p. 35-45, 1960.

KANG, G.; LEE, W.; SUGENO, M. Stability analysis of TSK fuzzy systems. In: **1998 IEEE International Conference on Fuzzy Systems**, Alaska, USA. 1998.

KARR, C. L.; FREEMAN, L. M.; MEREDITH, D. L. Improved Fuzzy Process Control of Spacecraft Autonomous Rendezvous Using a Genetic Algorithm. In: **1989 Intelligent Control and Adaptive Systems**, Philadelphia, USA. 1989.

KHANMIRZAEI, Z.; TESHNEHLAB, M.; SHARIFI, A. Modified honey bee optimization for recurrent neuro-fuzzy system model. In: **2010 International Conference on Computer and Automation Engineering**, Singapore. 2010.

KING, P. J.; MAMDANI, E. H. The application of fuzzy control systems to industrial processes. **Automatica,** v. 13, n. 3, p. 235-242, 1977.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by Simulated Annealing. **Science,** v. 220, n. 4598, p. 671-680, 1983.

KOSKO, B. **Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence**. Englewood Cliffs, New Jersey: Prentice-Hall International, 1992. ISBN 9780136114352.

KOSKO, B. Fuzzy systems as universal approximators. **IEEE Transactions on Computers,** v. 43, n. 11, p. 1329-1333, 1994.

LAI, J. C. Y. et al. Hypoglycaemia detection using fuzzy inference system with multi-objective double wavelet mutation Differential Evolution. **Applied Soft Computing,** v. 13, n. 5, p. 2803-2811, 2013.

LAI, Y.-C.; YE, N. Recent developments in chaotic time series analysis. **International Journal of Bifurcation and Chaos,** v. 13, n. 06, p. 1383-1422, 2003.

LAPEDES, A.; FARBER, R. **Nonlinear signal processing using neural networks-prediction and system modeling**. Los Alamos Nat. Lab. Los Alamos, NM. 1987. (Tech. Rep. LA-UR-87-2662)

LEE, C.-H.; CHANG, F.-Y. Interval type-2 recurrent fuzzy neural system for nonlinear systems control using stable simultaneous perturbation stochastic approximation algorithm. **Mathematical Problems in Engineering,** v. 2011, 2011.

LEE, C.-H. et al. A recurrent interval type-2 fuzzy neural network with asymmetric membership functions for nonlinear system identification. In: **2008 IEEE International Conference on Fuzzy Systems**, Hong Kong, China. 2008.

LEE, C.-H.; LEE, Y.-C. Nonlinear systems design by a novel fuzzy neural system via hybridization of electromagnetism-like mechanism and particle swarm optimisation algorithms. **Information Sciences,** v. 186, n. 1, p. 59-72, 2012.

LEE, C.-H.; LIN, C.-M.; YANG, M.-S. Control of nonlinear systems using non-stationary embedded recurrent fuzzy neural networks. In: **2012 International Conference on Machine Learning and Cybernetics**, Xian, Chiba. 2012.

LEE, C.-H.; TENG, C.-C. Identification and control of dynamic systems using recurrent fuzzy neural networks. **IEEE Transactions on Fuzzy Systems,** v. 8, n. 4, p. 349-366, 2000.

LEE, H. et al. Nonlinear system identification using recurrent networks. In: **1991 IEEE International Joint Conference on Neural Networks**, Singapore. 1991.

LILLY, J. H. **Fuzzy control and identification**. John Wiley & Sons, 2011.

LIN, C.-J.; CHEN, C.-H. Identification and prediction using recurrent compensatory neuro-fuzzy systems. **Fuzzy Sets and Systems,** v. 150, n. 2, p. 307-330, 2005.

_____. A compensation-based recurrent fuzzy neural network for dynamic system identification. **European Journal of Operational Research,** v. 172, n. 2, p. 696-715, 2006.

LIN, C.-J.; CHIN, C.-C. Prediction and identification using wavelet-based recurrent fuzzy neural networks. **IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,** v. 34, n. 5, p. 2144-2154, 2004.

LIN, C.-J.; LEE, C.-Y. A self-organizing recurrent fuzzy CMAC model for dynamic system identification. **International Journal of Intelligent Systems,** v. 23, n. 3, p. 384-396, 2008.

LIN, C.-J.; WU, C.-F.; LEE, C.-Y. Design of a recurrent functional neural fuzzy network using modified differential evolution. **International Journal of Innovative Computing, Information and Control,** v. 7, n. 2, p. 669-683, 2011.

LIN, C.-T.; LEE, C. S. G. Neural-network-based fuzzy logic control and decision system. **IEEE Transactions on Computers,** v. 40, n. 12, p. 1320-1336, 1991.

LIN, C. H.; LIN, C. P. The hybrid RFNN control for a PMSM drive electric scooter using rotor flux estimator. **International Journal of Electrical Power and Energy Systems,** v. 51, p. 213-223, 2013.

LIN, F.-J. et al. Three-degree-of-freedom dynamic model-based intelligent nonsingular terminal sliding mode control for a gantry position stage. **IEEE Transactions on Fuzzy Systems,** v. 20, n. 5, p. 971-985, 2012.

LIN, F. T. Performance comparison of Differential Evolution and Genetic Algorithms for the fuzzy transportation problem. **ICIC Express Letters,** v. 4, n. 6 B, p. 2469-2474, 2010.

LIN, H.-Y. et al. A hybrid of differential evolution and cultural algorithm for recurrent functional neural fuzzy networks and Its applications. **International Journal of Fuzzy Systems,** v. 14, n. 4, p. 519-529, 2012.

LIN, W. M.; HONG, C. M.; CHENG, F. S. Design of intelligent controllers for wind generation system with sensorless maximum wind energy control. **Energy Conversion and Management,** v. 52, n. 2, p. 1086-1096, 2011.

LIN, Y.-Y.; CHANG, J.-Y.; LIN, C.-T. Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network. **IEEE Transactions on Neural Networks and Learning Systems,** v. 24, n. 2, p. 310-321, Feb 2013.

LIN, Y.-Y. et al. An internal/interconnection recurrent type-2 fuzzy neural network (IRT2FNN) for dynamic system identification. In: **2010 IEEE International Conference on Systems, Man and Cybernetics**, Istanbul, Turkey. 2010.

LIN, Y.-Y. et al. A Mutually Recurrent Interval Type-2 Fuzzy Neural Network (MRIT2FNN) with Self-evolving Structure and Parameters. **IEEE Transactions on Fuzzy Systems,** v. PP, n. 99, p. 1-1, 2013.

LIU, H.; HUANG, D.; JIA, L. Dynamic system modeling with multilayer recurrent fuzzy neural network. In: **2007 International Conference on Computational Intelligence and Security**, Harbin, China. 2007.

LJUNG, L. Asymptotic variance expressions for identified black-box transfer function models. **IEEE Transactions on Automatic Control,** v. 30, n. 9, p. 834-844, 1985.

_____. **System identification: theory for the user**. Second Edition. Prentice Hall, 1999.

LORENZ, E. N. Deterministic nonperiodic flow. **Journal of the Atmospheric Sciences,** v. 20, n. 2, p. 130-141, 1963.

LU, H.-C.; CHANG, M.-H.; TSAI, C.-H. Parameter estimation of fuzzy neural network controller based on a modified differential evolution. **Neurocomputing,** v. 89, n. 0, p. 178-192, 2012.

MA, H. et al. On the equivalences and differences of evolutionary algorithms. **Engineering Applications of Artificial Intelligence**, doi: 10.1016/j.engappai.2013.05.002 em fase de elaboração.

MAMDANI, E. H. Application of fuzzy logic to approximate reasoning using linguistic systems. **Fuzzy Sets and Systems**, n. 26, p. 1182–1191, 1977.

MAMDANI, E. H. A fuzzy rule-based method of controlling dynamic processes. In: **1981 IEEE Conference on Decision and Control**, San Diego, USA. 1981.

MASTOROCOSTAS, P.; HILAS, C. A computational intelligence-based forecasting system for telecommunications time series. **Engineering Applications of Artificial Intelligence,** v. 25, n. 1, p. 200-206, 2012.

MASTOROCOSTAS, P. A.; HILAS, C. S. A block-diagonal recurrent fuzzy neural network for system identification. **Neural Computing & Applications,** v. 18, n. 7, p. 707-717, 2009.

MATHWORKS. **MATLAB**. Natick, Massachusetts: The MathWorks Inc. 2013.

MEZURA-MONTES, E.; VELÁZQUEZ-REYES, J.; COELLO, C. A. C. A comparative study of differential evolution variants for global optimization. In: **Anual Conference on Genetic and Evolutionary Computation**, Seattle, USA. 2006.

MON, Y.-J.; LIN, C.-M. Supervisory recurrent fuzzy neural network control for vehicle collision avoidance system design. **Neural Computing and Applications,** v. 21, n. 8, p. 2163-2169, 2012.

MON, Y. J.; LIN, C. M. Supervisory recurrent fuzzy neural network guidance law design for autonomous underwater vehicle. **International Journal of Fuzzy Systems,** v. 14, n. 1, p. 54-64, 2012.

MON, Y. J.; LIN, C. M.; YEH, R. G. Intelligent control for long-term ecological systems. **Journal of Intelligent and Fuzzy Systems,** v. 24, n. 4, p. 905-913, 2013.

NARENDRA, K. S.; PARTHASARATHY, K. Identification and control of dynamical systems using neural networks. **IEEE Transactions on Neural Networks,** v. 1, n. 1, p. 4-27, 1990.

_____. Gradient methods for the optimization of dynamical systems containing neural networks. **IEEE Transactions on Neural Networks,** v. 2, n. 2, p. 252-262, 1991.

NARENDRA, K. S.; PARTHASARATHY, K. Neural networks and dynamical systems. **International Journal of Approximate Reasoning,** v. 6, n. 2, p. 109-131, 1992.

NAUCK, D.; KRUSE, R. A neuro-fuzzy method to learn fuzzy classification rules from data. **Fuzzy Sets and Systems,** v. 89, n. 3, p. 277-288, 1997.

_____. Neuro-fuzzy systems for function approximation. **Fuzzy Sets and Systems,** v. 101, n. 2, p. 261-271, 1999.

NAUCK, D. D.; NÜRNBERGER, A. Neuro-fuzzy systems: a short historical review. In: MOEWES, C. e NÜRNBERGER, A. (Ed.). **Computational Intelligence in Intelligent Data Analysis**: Springer Berlin Heidelberg, v.445, 2013. cap. 7, p.91-109. (Studies in Computational Intelligence).

NELLES, O. **Nonlinear system identification: from classical approaches to neural networks and fuzzy models**. Springer-Verlag, 2001. 785

NERI, F.; TIRRONEN, V. Recent advances in differential evolution: a survey and experimental analysis. **Artificial Intelligence Review,** v. 33, n. 1, p. 61-106, 2010.

OH, S.-K.; KIM, W.-D.; PEDRYCZ, W. Design of optimized cascade fuzzy controller based on differential evolution: Simulation studies and practical insights. **Engineering Applications of Artificial Intelligence,** v. 25, n. 3, p. 520-532, 2012.

PAPPIS, C. P.; MAMDANI, E. H. A Fuzzy Logic Controller for a Trafc Junction. **IEEE Transactions on Systems, Man and Cybernetics,** v. 7, n. 10, p. 707-717, 1977.

PEDRYCZ, W.; GOMIDE, F. **Fuzzy systems engineering: toward human-centric computing**. Michigan: John Wiley, 2007.

PHAM, D. T.; LIU, X. Identification of linear and nonlinear dynamic systems using recurrent neural networks. **Artificial Intelligence in Engineering,** v. 8, n. 1, p. 67-75, 1993.

PLAGIANAKOS, V. P.; TASOULIS, D. K.; VRAHATIS, M. N. A review of major application areas of differential evolution. In: CHAKRABORTY, U. (Ed.). **Advances in Differential**

Evolution: Springer Berlin Heidelberg, v.143, 2008. cap. 8, p.197-238. (Studies in Computational Intelligence).

PRICE, K.; STORN, R. M.; LAMPINEN, J. A. **Differential Evolution: A Practical Approach to Global Optimization**. Springer-Verlag New York, Inc., 2005. ISBN 3540209506.

PRICE, K. V. Genetic annealing. **Dr. Dobb's Journal**, p. 127-132, 1994.

PRICE, K. V. Differential evolution vs. the functions of the 2nd ICEO. In: **1997 IEEE International Conference on Evolutionary Computation**, Indianapolis, USA. 1997.

PROCYK, T. J.; MAMDANI, E. H. A linguistic self-organizing process controller. **Automatica,** v. 15, n. 1, p. 15-30, 1979.

RAHNAMAYAN, S.; TIZHOOSH, H. R.; SALAMA, M. M. A. A novel population initialization method for accelerating evolutionary algorithms. **Computers & Mathematics with Applications,** v. 53, n. 10, p. 1605-1614, 2007.

RAMAMOORTHY, P. A.; HUANG, S. Design of stable fuzzy-logic-controlled feedback systems. In: **1993 IEEE International Conference on Fuzzy Systems**, San Francisco, USA. 1993.

RODRIGUEZ, F. O.; YU, W.; MORENO-ARMENDARIZ, M. A. Nonlinear systems identification via two types of recurrent fuzzy CMAC. **Neural Processing Letters,** v. 28, n. 1, p. 49-62, 2008.

SASTRY, K. K. N.; BEHERA, L.; NAGRATH, I. J. Differential evolution based fuzzy logic controller for nonlinear process control. **Fundamenta Informaticae,** v. 37, n. 1, p. 121-136, 1999.

SASTRY, P. S.; SANTHARAM, G.; UNNIKRISHNAN, K. P. Memory neuron networks for identification and control of dynamical systems. **IEEE Transactions on Neural Networks,** v. 5, n. 2, p. 306-319, 1994.

SETNES, M.; BABUSKA, R.; VERBRUGGEN, H. B. Transparent fuzzy modelling. **International Journal of Human-Computer Studies,** v. 49, n. 2, p. 159-179, 1998.

SINGH, L.; KUMAR, S.; PAUL, S. Automatic simultaneous architecture and parameter search in fuzzy neural network learning using novel variable length crossover differential evolution. In: **2008 IEEE International Conference on Fuzzy Systems**, Hong Kong, China. 2008.

SJÖBERG, J. et al. Nonlinear black-box modeling in system identification: a unified overview. **Automatica,** v. 31, n. 12, p. 1691-1724, 1995.

SONG, J.-R.; SHI, H.-B. Dynamic system modeling based on wavelet recurrent fuzzy neural network. In: **2011 International Conference on Natural Computation**, Shanghai, China. 2011.

STAVRAKOUDIS, D. G.; PAPASTAMOULIS, A. K.; THEOCHARIS, J. B. Evolutionary identification of a recurrent fuzzy neural network with enhanced memory capabilities. In: **2008 International Workshop on Genetic and Evolving Fuzzy Systems**, Witten-Bommerholz, Alemanha. 2008.

STORN, R.; PRICE, K. **Differential evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces**. International Computer Science Institute. Berkeley, CA. 1995

STORN, R.; PRICE, K. Minimizing the real functions of the ICEC'96 contest by differential evolution. In: **1996 IEEE International Conference on Evolutionary Computation**, Nagoya, Japan. 1996.

STORN, R.; PRICE, K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. **Journal of Global Optimization,** v. 11, n. 4, p. 341-359, 1997.

SU, H.; YANG, Y. Differential evolution and quantum-inquired differential evolution for evolving Takagi-Sugeno fuzzy models. **Expert Systems with Applications,** v. 38, n. 6, p. 6447-6451, 2011.

SU, M.-T. et al. A rule-based symbiotic modified differential evolution for self-organizing neuro-fuzzy systems. **Applied Soft Computing,** v. 11, n. 8, p. 4847-4858, 2011.

SUGENO, M. Fuzzy control: Principles, practice and perspectives. In: **1992 IEEE International Conference on Fuzzy Systems**, San Diego, USA. 1992.

SUGENO, M.; KANG, G. T. Structure identification of fuzzy model. **Fuzzy Sets and Systems,** v. 28, n. 1, p. 15-33, 1988.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. **IEEE Transactions on Systems, Man and Cybernetics,** v. 15, n. 1, p. 116-132, 1985.

TING, C.-K.; HUANG, C.-H. Varying number of difference vectors in differential evolution. In: **2009 IEEE Congress on Evolutionary Computation**, Trondheim, Norway. 2009.

TONG, R. M. The evaluation of fuzzy models derived from experimental data. **Fuzzy Sets and Systems,** v. 4, n. 1, p. 1-12, 1980.

TSAI, C. C.; CHANG, Y. L. Two-degree-of-freedom control using recurrent fuzzy neural networks for a class of nonlinear discrete-time time-delay systems. In: **2012 International Conference on System Science and Engineering**, Dalian, China. 2012.

TU, C.-C.; JUANG, C.-F. Recurrent type-2 fuzzy neural network using Haar wavelet energy and entropy features for speech detection in noisy environments. **Expert Systems with Applications,** v. 39, n. 3, p. 2479-2488, 2012.

UNAL, F. A.; KHAN, E. A fuzzy finite state machine implementation based on a neural fuzzy system. In: **1994 IEEE International Conference on Fuzzy Systems**, Orlando, USA. 1994.

VINEETHA, S.; CHANDRA SHEKARA BHAT, C.; IDICULA, S. M. Gene regulatory network from microarray data of colon cancer patients using TSK-type recurrent neural fuzzy network. **Gene,** v. 506, n. 2, p. 408-416, 2012.

_____. MicroRNA–mRNA interaction network using TSK-type recurrent neural fuzzy network. **Gene,** v. 515, n. 2, p. 385-390, 2013.

WAI, R. J.; LIU, C. M.; LIN, Y. W. Robust path tracking control of mobile robot via dynamic petri recurrent fuzzy neural network. **Soft Computing,** v. 15, n. 4, p. 743-767, 2011.

WANG, L.; LUN, Z.; CAO, X. Based on Dynamic Recursion of Fuzzy Neural Network in the Short-term Load Forecasting of Power System. In: XU, B. Y. e SHEN, J. (Ed.). **2010 International Conference on Information, Electronic and Computer Science, Vols 1-3**, 2010. p.336-339.

WANG, L. X.; MENDEL, J. M. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. **IEEE Transactions on Neural Networks,** v. 3, n. 5, p. 807-814, 1992a.

_____. Generating fuzzy rules by learning from examples. **IEEE Transactions on Systems, Man and Cybernetics,** v. 22, n. 6, p. 1414-1427, 1992b.

WANG, R. et al. Assessment of human operator functional state using a novel differential evolution optimization based adaptive fuzzy model. **Biomedical Signal Processing and Control,** v. 7, n. 5, p. 490-498, 2012.

WANG, Y.-C.; CHIEN, C.-J.; LEE, D.-T. An output recurrent fuzzy neural network based iterative learning control for nonlinear systems. In: **2008 IEEE International Conference on Fuzzy Systems**, Hong Kong, China. 2008.

WU, G.-D.; ZHU, Z.-W.; HUANG, P.-H. A TS-type maximizing-discriminability-based recurrent fuzzy network for classification problems. **IEEE Transactions on Fuzzy Systems,** v. 19, n. 2, p. 339-352, 2011.

XU, G.; SONG, A.; LI, H. Adaptive impedance control for upper-limb rehabilitation robot using evolutionary dynamic recurrent fuzzy neural network. **Journal of Intelligent & Robotic Systems,** v. 62, n. 3-4, p. 501-525, Jun 2011.

YU, X.; GEN, M. **Introduction to evolutionary algorithms**. London: Springer-Verlag 2010.

ZADEH, L. A. On the Identification Problem. **Circuit Theory, IRE Transactions on,** v. 3, n. 4, p. 277-281, 1956.

_____. Fuzzy sets. **Information and Control,** v. 8, n. 3, p. 338-353, 1965.

_____. Outline of a new approach to the analysis of complex systems and decision processes. **IEEE Transactions on Systems, Man and Cybernetics,** v. SMC-3, n. 1, p. 28-44, 1973.

ZADEH, L. A. A theory of approximate reasoning. **Machine intelligence,** v. 9, p. 149-194, 1979.

ZAHARIE, D. Critical values for the control parameters of differential evolution algorithms. In: **2002 International Conference on Soft Computing**, Brno, Czech Republic. 2002.

_____. Influence of crossover on the behavior of differential evolution algorithms. **Applied Soft Computing,** v. 9, n. 3, p. 1126-1138, 2009.

ZHANG, L.-J. et al. DRFNN-Adaptive Output Feedback Controller for Depth Tracking of AUV. In: **2011 Chinese Control Conference**, Yantai, China. 2011.

ZHANG, Q.; LEE, M. Analyzing the dynamics of emotional scene sequence using recurrent neuro-fuzzy network. **Cognitive Neurodynamics,** v. 7, n. 1, p. 47-57, Feb 2013.

ZHOU, S.-M.; GAN, J. Q. Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling. **Fuzzy Sets and Systems,** v. 159, n. 23, p. 3091-3131, 2008.

Apêndice A.

Resultados Detalhados da Simulação 1

▪ Problema P1 – Planta 1

Tabela A.1. Desempenho dos algoritmos – Problema P1

	GA	DE/rand/1/bin	DE/best/1/bin	DE/target-to-best/1/bin
Estrutura – 1ª Ordem	*175*	*179*	*179*	*179*
Melhor	0,0747	0,0687	0,0688	0,0691
Pior	0,0815	0,0757	0,0935	0,0749
Médio ± Desvio	0,0762 ± 0,0016	**0,0701 ± 0,0018**	0,0721 ± 0,0046	0,0715 ± 0,0015
Interv. Conf.	[0,0758; 0,0770]	[0,0697; 0,0709]	[0,0710; 0,0753]	[0,0710; 0,0720]
Estrutura – 2ª Ordem	*234*	*254*	*254*	*253*
Melhor	0,0658	0,0530	0,0530	0,0578
Pior	0,0782	0,0678	0,0682	0,0899
Médio ± Desvio	0,0685 ± 0,0025	**0,0558 ± 0,0029**	0,0577 ± 0,0035	0,0617 ± 0,0058
Interv. Conf.	[0,0678; 0,0698]	[0,0551; 0,0574]	[0,0566; 0,0590]	[0,0604; 0,0653]
Estrutura – 3ª Ordem	*325*	*334*	*334*	*334*
Melhor	0,2284	0,0448	0,0438	0,0464
Pior	0,2930	0,0524	0,0541	0,0628
Médio ± Desvio	0,2647 ± 0,0311	**0,0471 ± 0,0022**	0,0475 ± 0,0027	0,0511 ± 0,0044
Interv. Conf.	[0,2541; 0,2750]	[0,0465; 0,0480]	[0,0467; 0,0485]	[0,0499; 0,0530]

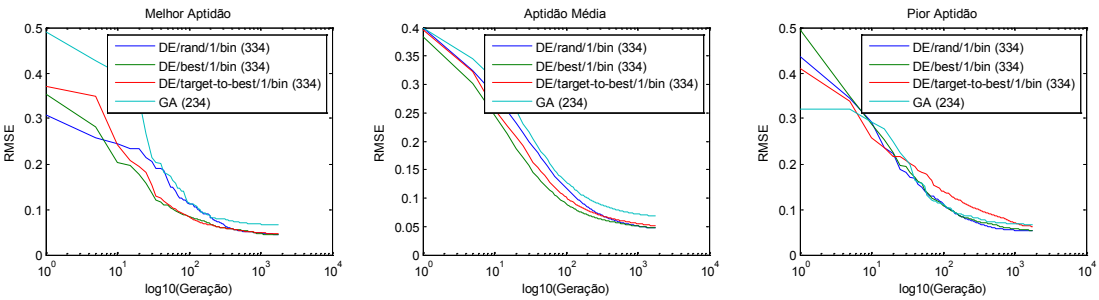
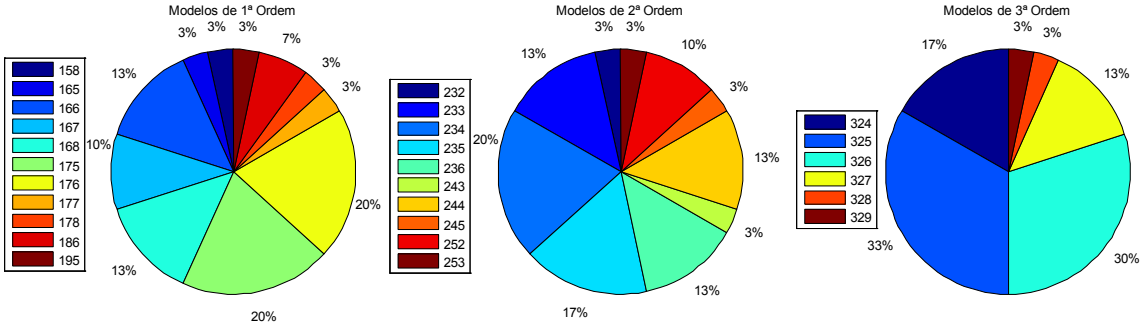
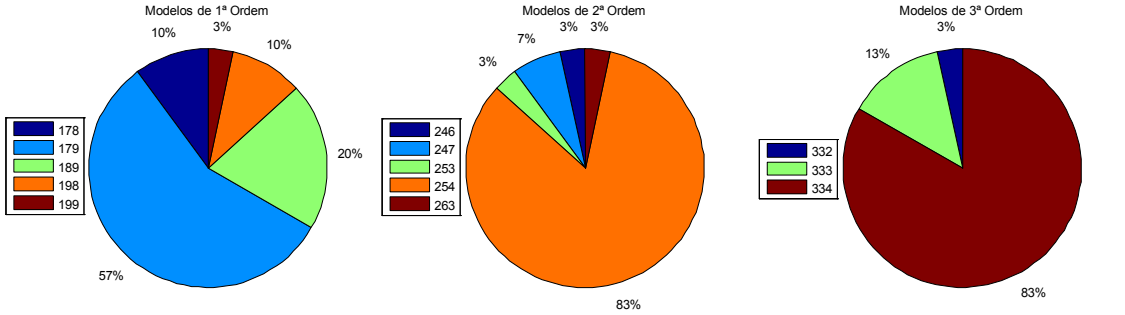


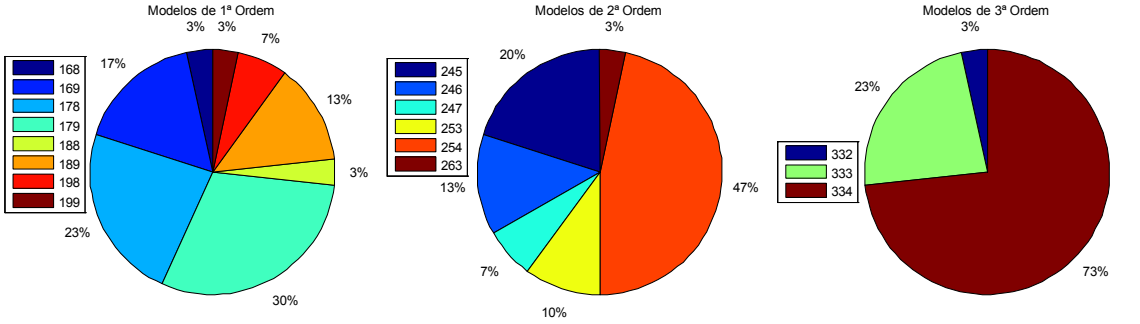
Figura A.1. Evolução da Melhor Solução nas 30 execuções – Problema P1



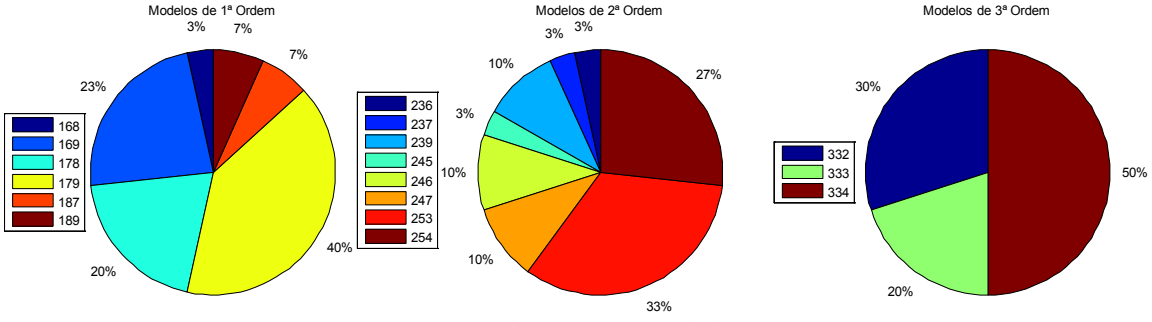
(a) GA



(b) DE/rand/1/bin



(c) DE/best/1/bin



(d) DE/target-to-best/1/bin

Figura A.2. Frequência de ocorrência das estruturas nas 30 execuções – Problema P1

▪ Problema P2 – Planta 2

Tabela A.2. Desempenho dos algoritmos no conjunto de treino – Problema P2

	GA	DE/rand/1/bin	DE/best/1/bin	DE/target-to-best/1/bin
Estrutura – 1ª Ordem	*145*	*159*	*157*	*159*
Melhor	0,0874	0,0845	0,0853	0,0847
Pior	0,1259	0,1112	0,1623	0,1542
Médio ± Desvio	0,0967 ± 0,0111	**0,0879 ± 0,0062**	0,0948 ± 0,0181	0,0953 ± 0,0149
Interv. Conf.	[0.0935; 0.1015]	[0.0863; 0.0914]	[0.0896; 0.1031]	[0.0913; 0.1023]
Estrutura – 2ª Ordem	*236*	*239*	*237*	*236*
Melhor	0,0828	0,0791	0,0799	0,0801
Pior	0,0949	0,0845	0,0998	0,0852
Médio ± Desvio	0,0880 ± 0,0028	**0,0805 ± 0,0013**	0,0824 ± 0,0037	0,0818 ± 0,0010
Interv. Conf.	[0,0871; 0,0890]	[0,0802; 0,0811]	[0,0815; 0,0849]	[0,0815; 0,0822]
Estrutura – 3ª Ordem	*324*	*322*	*322*	*322*
Melhor	0,2105	0,0665	0,0667	0,0668
Pior	0,2140	0,0869	0,0917	0,0905
Médio ± Desvio	0,2117 ± 0,0008	**0,0732 ± 0,0060**	0,0808 ± 0,0076	0,0795 ± 0,0073
Interv. Conf.	[0.2114; 0.2120]	[0.0714; 0.0758]	[0.0780; 0.0833]	[0.0770; 0.0820]

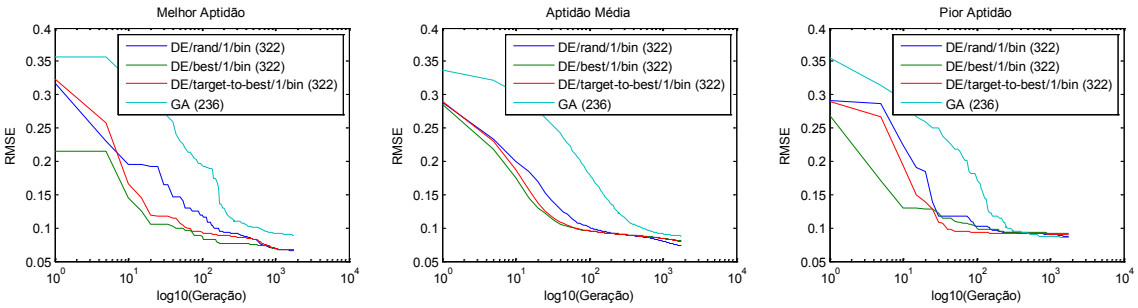
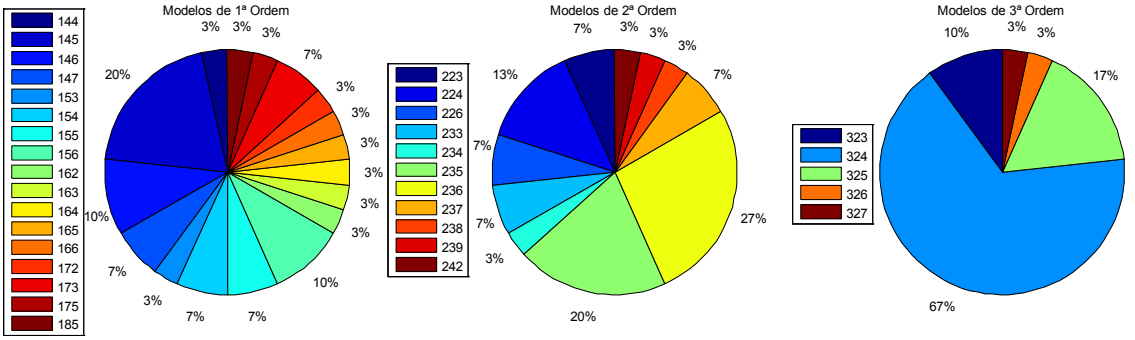
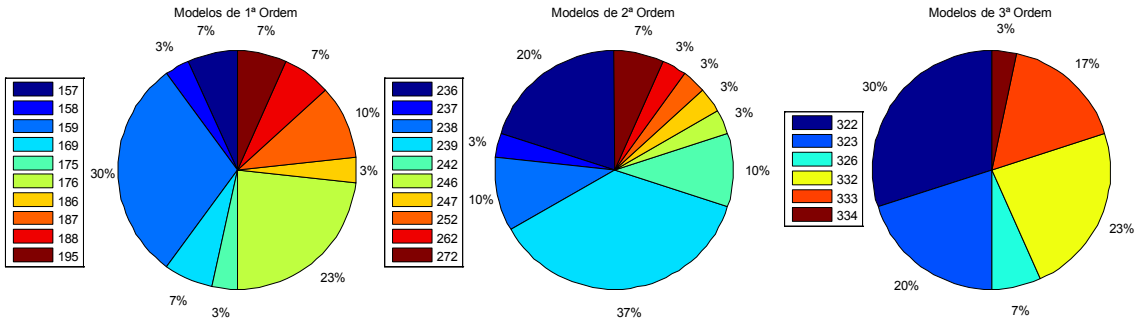


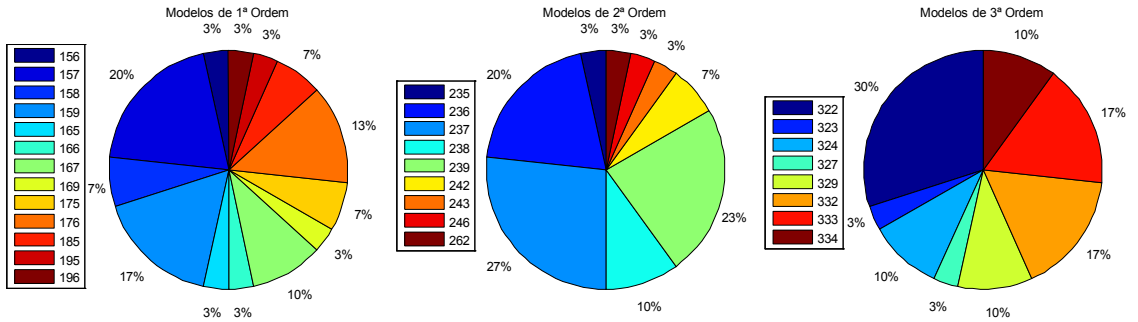
Figura A.3. Evolução da Melhor Solução nas 30 execuções – Problema P2



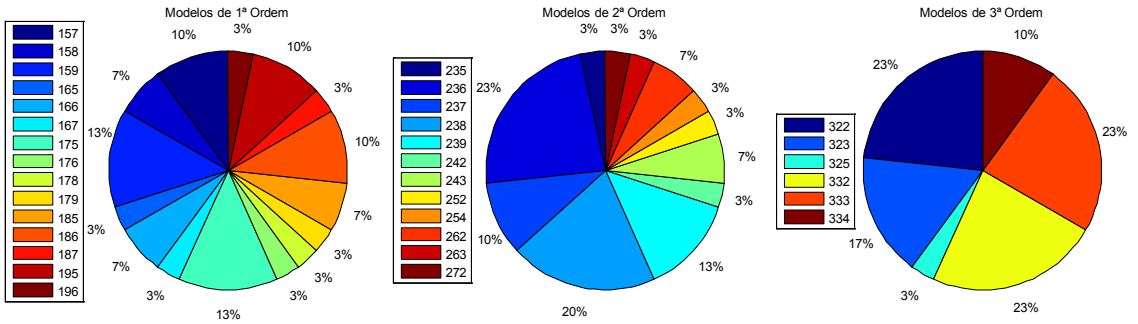
(a) GA



(b) DE/rand/1/bin



(c) DE/best/1/bin



(d) DE/target-to-best/1/bin

Figura A.4. Frequência de ocorrência das estruturas nas 30 execuções – Problema P2

▪ Problema P3 – Série Caótica Univariada de Hénon

Tabela A.3. Desempenho dos algoritmos no conjunto de treino – Problema P3

	GA	DE/rand/1/bin	DE/best/1/bin	DE/target-to-best/1/bin
Estrutura – 1ª Ordem	*148*	*189*	*198*	*187*
Melhor	0,0858	0,0504	0,0537	0,0593
Pior	0,4004	0,1052	0,3264	0,2697
Médio ± Desvio	0,1048 ± 0,0569	**0,0641 ± 0,0135**	0,1170 ± 0,0755	0,0937 ± 0,0389
Interv. Conf.	[0,0938; 0,1577]	[0,0596; 0,0695]	[0,0941; 0,1487]	[0,0842; 0,1204]
Estrutura – 2ª Ordem	*249*	*249*	*249*	*249*
Melhor	0,0840	0,0496	0,0501	0,0524
Pior	0,3349	0,1523	0,2233	0,1923
Médio ± Desvio	0,1340 ± 0,0564	**0,0703 ± 0,0262**	0,0846 ± 0,0493	0,0942 ± 0,0384
Interv. Conf.	[0,1184; 0,1587]	[0,0634; 0,0833]	[0,0714; 0,1093]	[0,0824; 0,1109]
Estrutura – 3ª Ordem	*329*	*335*	*335*	*335*
Melhor	0,4326	0,0346	0,0360	0,0391
Pior	0,5423	0,0866	0,2617	0,0997
Médio ± Desvio	0,4869 ± 0,0408	**0,0465 ± 0,0109**	0,0597 ± 0,0402	0,0609 ± 0,0183
Interv. Conf.	[0,4736; 0,5012]	[0,0434; 0,0517]	[0,0511; 0,0840]	[0,0545; 0,0677]

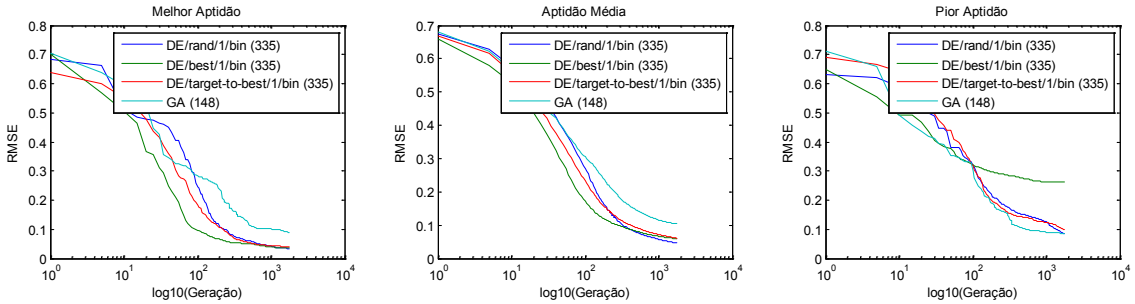
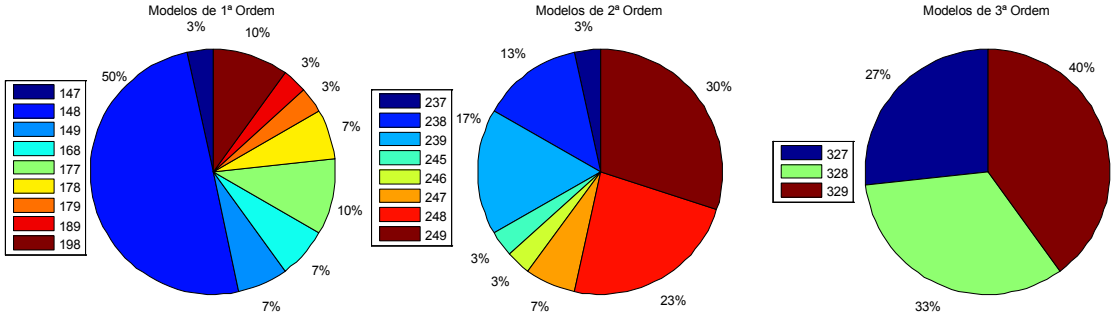
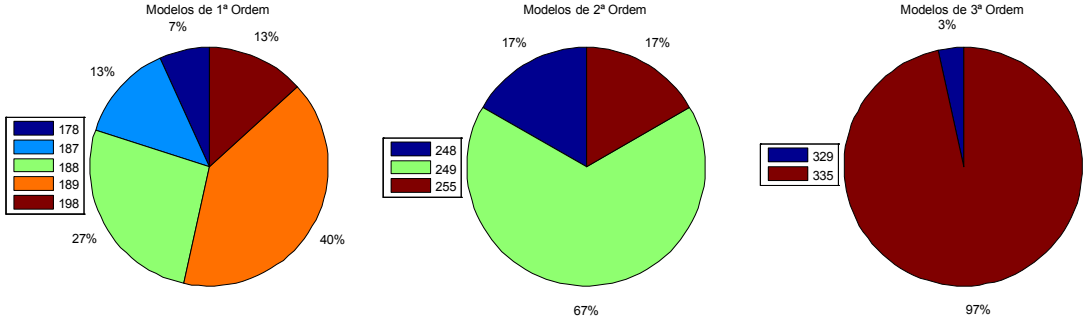


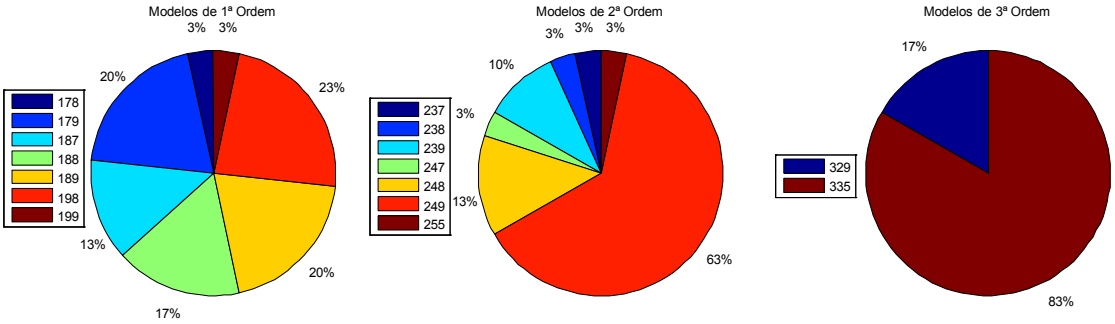
Figura A.5. Evolução da Melhor Solução nas 30 execuções – Problema P3



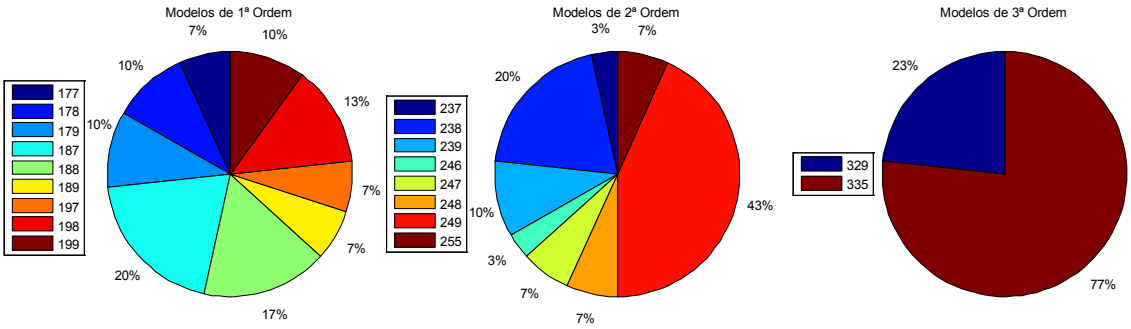
(a) GA



(b) DE/rand/1/bin



(c) DE/best/1/bin



(d) DE/target-to-best/1/bin

Figura A.6. Frequência de ocorrência das estruturas nas 30 execuções – Problema P3

▪ Problema P4 – Série Caótica Multivariada de Lorenz

• Dimensão X

Tabela A.4. Desempenho dos algoritmos no conjunto de treino – Problema P4(X)

	GA	DE/rand/1/bin	DE/best/1/bin	DE/target-to-best/1/bin
Estrutura – 1ª Ordem	*199*	*199*	*199*	*189*
Melhor	1,6413	1,6224	1,6245	1,6308
Pior	2,0656	1,7126	1,9300	1,7465
Médio ± Desvio	1,6941 ± 0,0841	**1,6405 ± 0,0173**	1,6681 ± 0,0627	1,6432 ± 0,0229
Interv. Conf.	[1.6733; 1.7452]	[1.6362; 1.6511]	[1.6530; 1.7058]	[1.6375; 1.6564]
Estrutura – 2ª Ordem	*247*	*246*	*246*	*247*
Melhor	1,2877	1,2240	1,2215	1,1968
Pior	1,6167	1,5839	1,5367	1,5431
Médio ± Desvio	1,4575 ± 0,0913	**1,2664 ± 0,0851**	1,2945 ± 0,1024	1,3475 ± 0,0886
Interv. Conf.	[1.4235; 1.4901]	[1.2419; 1.3108]	[1.2622; 1.3343]	[1.3146; 1.3788]
Estrutura – 3ª Ordem	*325*	*335*	*334*	*329*
Melhor	2,9748	0,7541	0,7247	0,8000
Pior	3,1446	0,8766	1,0054	1,0140
Médio ± Desvio	3,0445 ± 0,0386	**0,7937 ± 0,0378**	0,8036 ± 0,0539	0,8549 ± 0,0474
Interv. Conf.	[3.0323; 3.0597]	[0.7814; 0.8073]	[0.7898; 0.8300]	[0.8403; 0.8756]

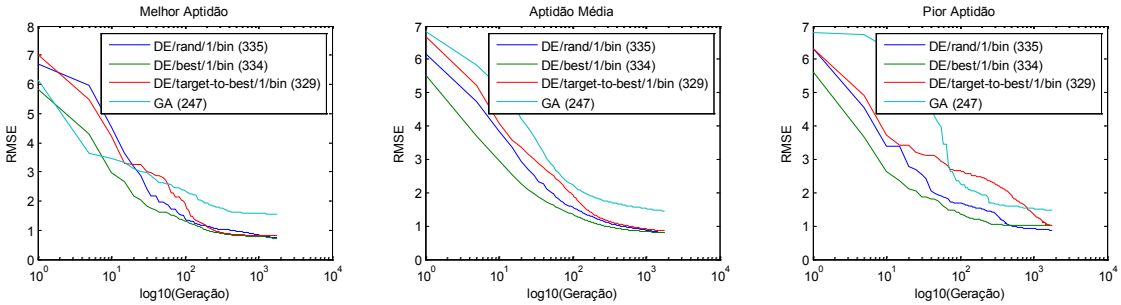
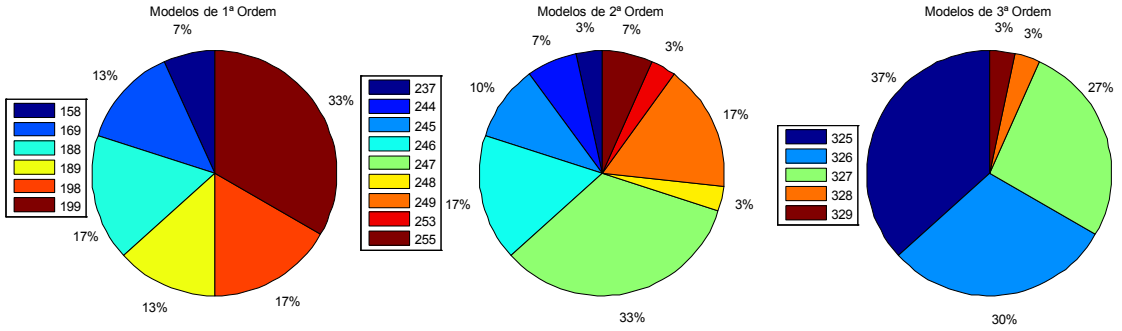
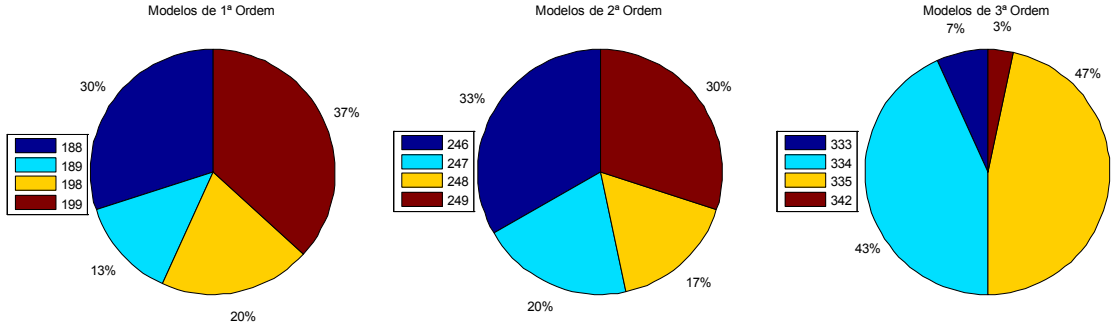


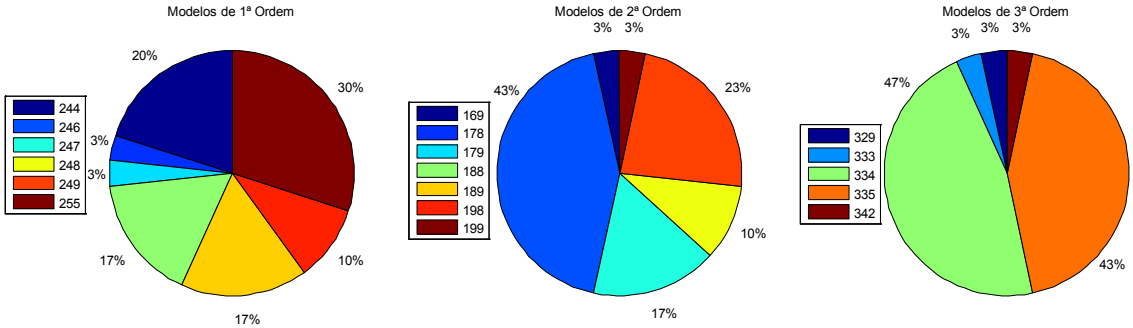
Figura A.7. Evolução da melhor solução nas 30 execuções – Problema P4(X)



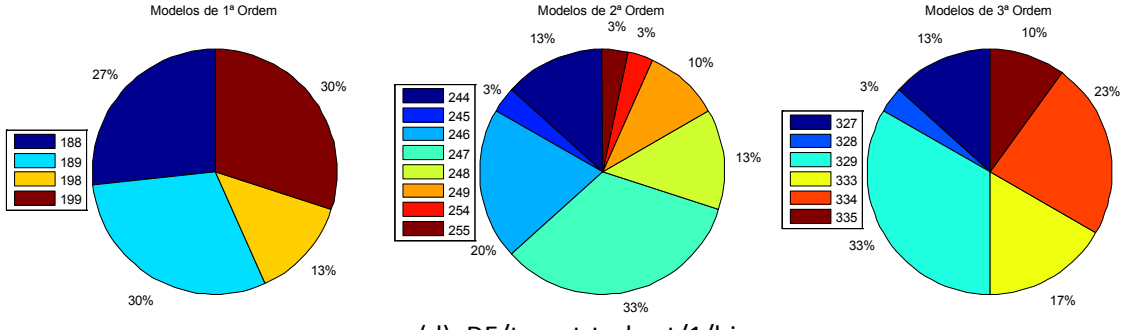
(a) GA



(b) DE/rand/1/bin



(c) DE/best/1/bin



(d) DE/target-to-best/1/bin

Figura A.8. Frequência de ocorrência das estruturas nas 30 execuções – Problema P4X

- **Dimensão Y**

Tabela A.5. Desempenho dos algoritmos no conjunto de treino – Problema P4(Y)

	GA	DE/rand/1/bin	DE/best/1/bin	DE/target-to-best/1/bin
Estrutura – 1ª Ordem	*178*	*188*	*188*	*188*
Melhor	2,4333	2,3981	2,4031	2,4002
Pior	2,6109	2,5730	2,5834	2,5836
Médio ± Desvio	2,4757 ± 0,0402	**2,4252 ± 0,0329**	2,4443 ± 0,0408	2,4365 ± 0,0430
Interv. Conf.	[2.4647; 2.4931]	[2.4175; 2.4480]	[2.4330; 2.4624]	[2.4259; 2.4587]
Estrutura – 2ª Ordem	*247*	*249*	*249*	*247*
Melhor	1,8561	1,6407	1,6409	1,7437
Pior	2,3889	1,9672	2,3323	2,2948
Médio ± Desvio	2,0471 ± 0,1331	**1,7501 ± 0,0821**	1,8255 ± 0,1984	1,8735 ± 0,1457
Interv. Conf.	[2.0070; 2.1029]	[1.7242; 1.7804]	[1.7653; 1.9083]	[1.8318; 1.9435]
Estrutura – 3ª Ordem	*325*	*334*	*334*	*334*
Melhor	4,0605	1,1376	1,1659	1,1670
Pior	4,2138	1,3192	1,5586	1,7378
Médio ± Desvio	4,1200 ± 0,0348	**1,2174 ± 0,0372**	1,2726 ± 0,0822	1,3470 ± 0,1134
Interv. Conf.	[4.1088; 4.1323]	[1.2023; 1.2281]	[1.2508; 1.3129]	[1.3155; 1.3972]

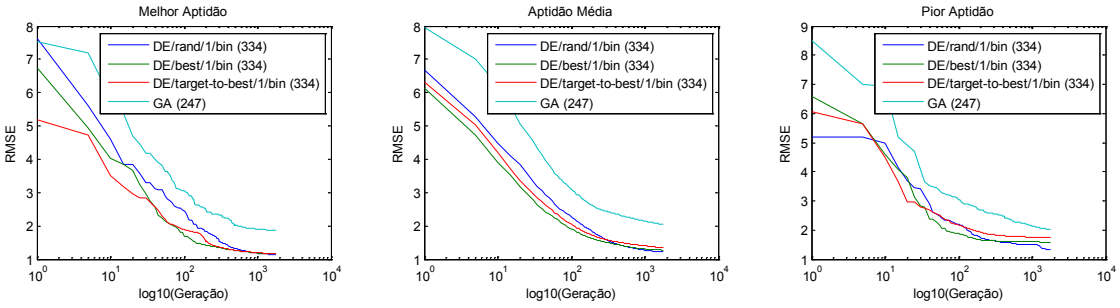
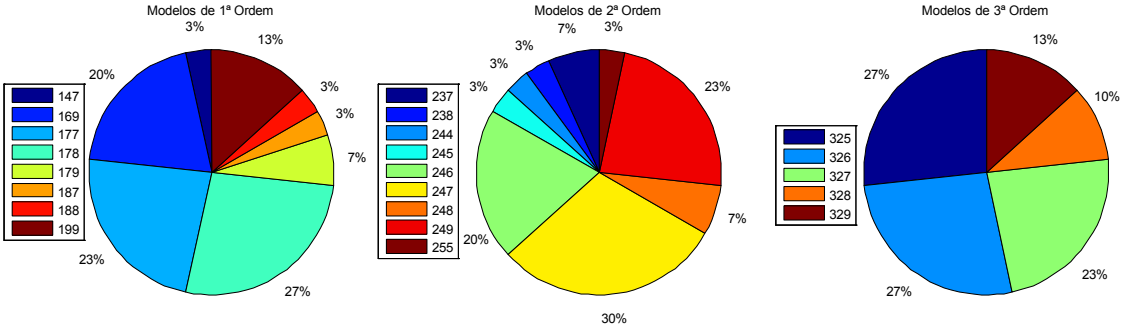
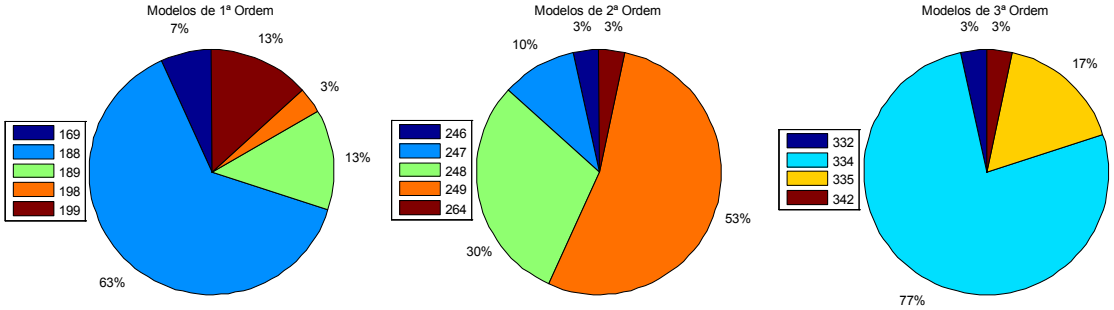


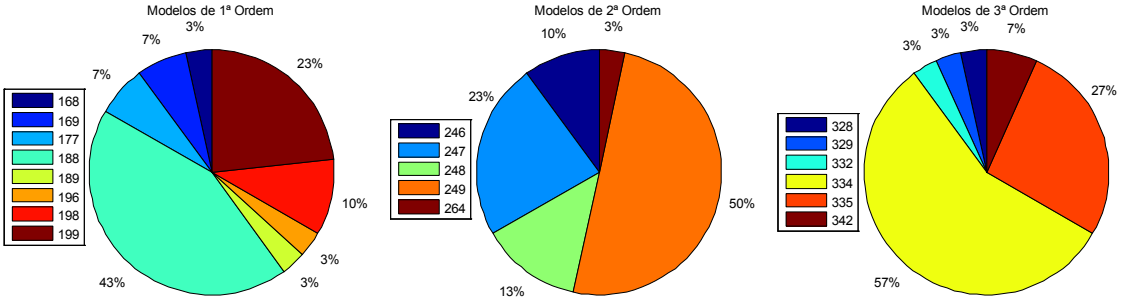
Figura A.9. Evolução da melhor solução nas 30 execuções – Problema P4(Y)



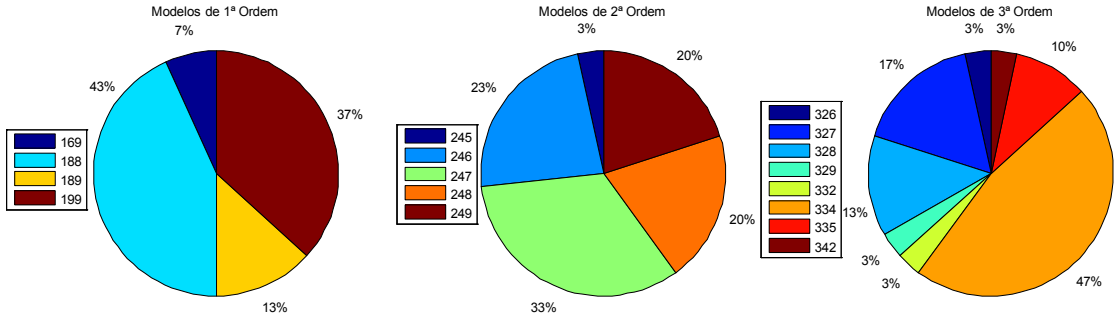
(a) GA



(b) DE/rand/1/bin



(c) DE/best/1/bin



(d) DE/target-to-best/1/bin

Figura A.10. Frequência de ocorrência das estruturas nas 30 execuções – Problema P4Y

- **Dimensão Z**

Tabela A.6. Desempenho dos algoritmos no conjunto de treino – Problema P4(Z)

	GA	DE/rand/1/bin	DE/best/1/bin	DE/target-to-best/1/bin
Estrutura – 1ª Ordem	*189*	*199*	*199*	*179*
Melhor	2,9383	2,7020	2,9814	3,0014
Pior	3,2982	3,0753	3,8922	3,0625
Médio ± Desvio	3,0774 ± 0,0662	**2,9993 ± 0,0598**	3,0788 ± 0,1695	3,0197 ± 0,0145
Interv. Conf.	[3,0575; 3,1037]	[2,9599; 3,0122]	[3,0420; 3,1873]	[3,0155; 3,0255]
Estrutura – 2ª Ordem	*239*	*248*	*249*	*239*
Melhor	2,1266	1,7952	1,7941	1,8845
Pior	2,8685	2,0850	2,4451	2,1342
Médio ± Desvio	2,3871 ± 0,1768	**1,8859 ± 0,0860**	1,9459 ± 0,1639	1,9238 ± 0,0565
Interv. Conf.	[2,3314; 2,4563]	[1,8587; 1,9211]	[1,8957; 2,0157]	[1,9092; 1,9502]
Estrutura – 3ª Ordem	*325*	*329*	*329*	*329*
Melhor	4,2784	1,2944	1,2945	1,3054
Pior	5,0653	1,4840	1,3785	1,3699
Médio ± Desvio	4,6775 ± 0,3241	1,3190 ± 0,0344	**1,3153 ± 0,0208**	1,3360 ± 0,0173
Interv. Conf.	[4,5713; 4,7927]	[1,3109; 1,3409]	[1,3093; 1,3252]	[1,3300; 1,3427]

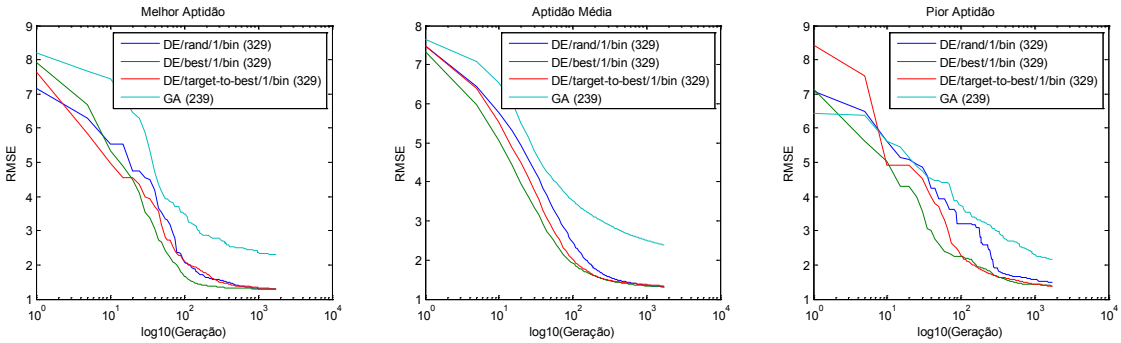
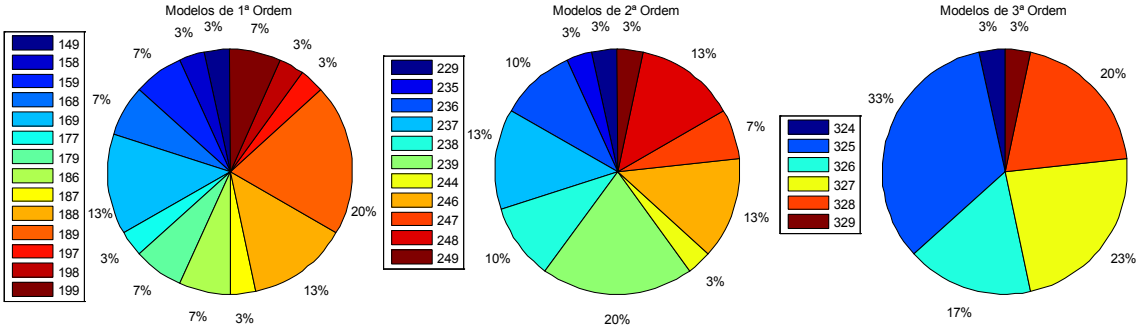
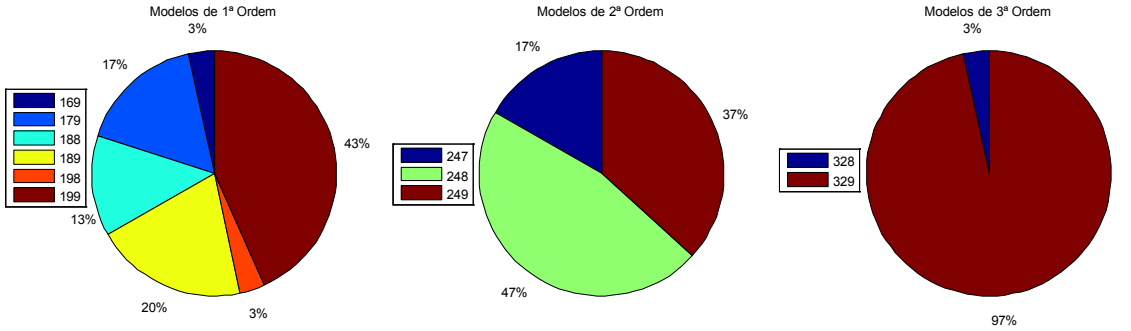


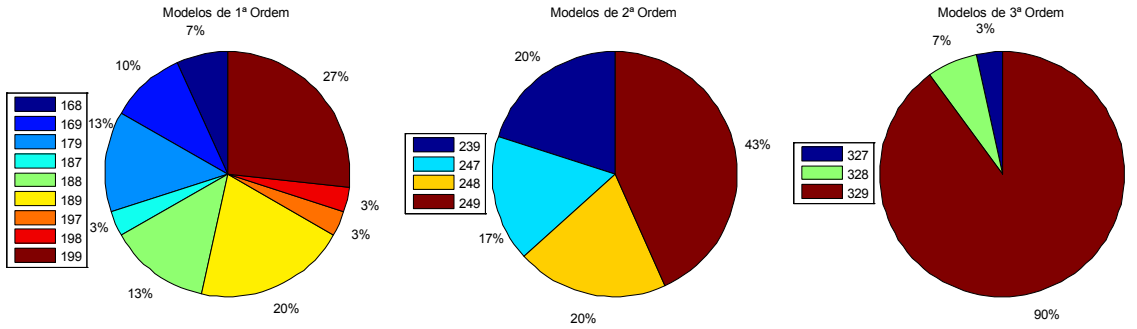
Figura A.11. Evolução da melhor solução nas 30 execuções – Problema P4(Z)



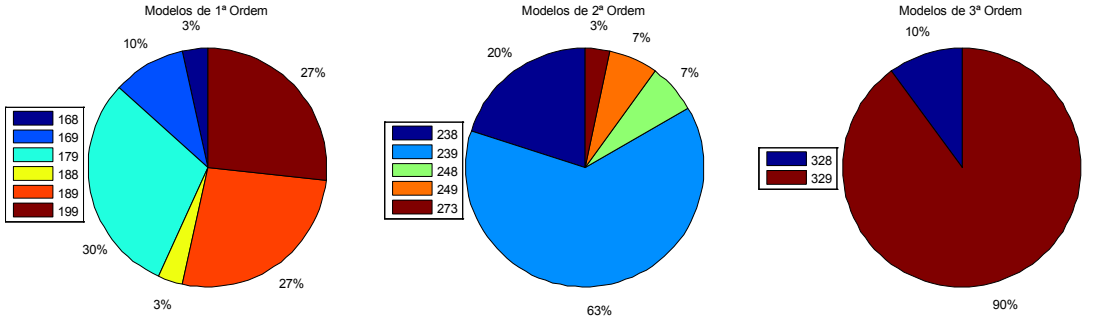
(a) GA



(b) DE/rand/1/bin



(c) DE/best/1/bin



(d) DE/target-to-best/1/bin

Figura A.12. Frequência de ocorrência das estruturas nas 30 execuções – Problema P4Z

Apêndice B.

Resultados Detalhados da Simulação 2

▪ Problema P1 – Planta 1

Tabela B.1. Resultado comparativo entre os modelos – Problema P1

		RFS–GA	RFS–DE	DFRFS–DE
Desempenho				
Treino	Melhor	0,0645	0,0448	0,0248
	Pior	0,0704	0,0498	0,0308
	Médio ± Desvio	0,0677 ± 0,0013	0,0456 ± 0,0040	**0,0265 ± 0,0014**
	Interv. Conf.	[0,0672; 0,0681]	[0,0440; 0,0469]	[0,0262; 0,0272]
Teste	Melhor	0,0566	0,0372	0,0186
	Pior	0,2372	0,0524	0,0590
	Médio ± Desvio	0,0989 ± 0,0656	0,0478 ± 0,0019	**0,0286 ± 0,0081**
	Interv. Conf.	[0,0777; 0,1261]	[0,0471; 0,0484]	[0,0264; 0,0324]
Complexidade				
	Menor	18	54	72
	Maior	54	108	108
	Média ± Desvio	**28 ± 10**	71 ± 25	92 ± 16
	Interv. Conf.	[24; 32]	[62; 81]	[86;97]

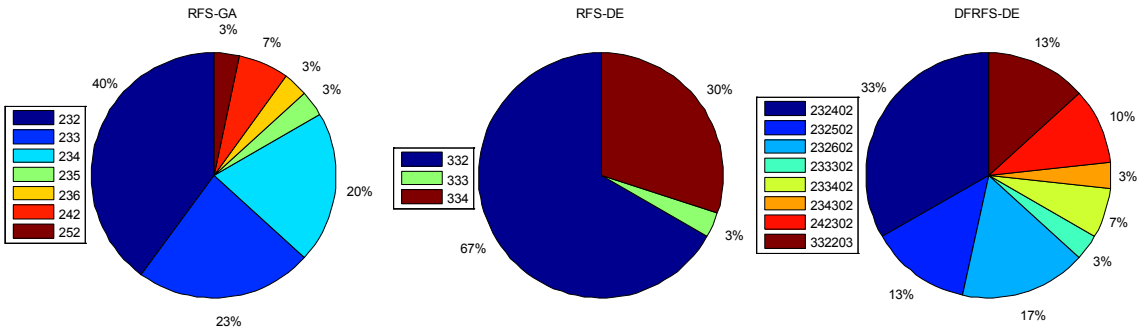


Figura B.1. Distribuição das estruturas selecionadas nas 30 execuções, segundo o critério AIC – Problema P1

▪ Problema P2 – Planta 2

Tabela B.2. Resultado comparativo entre os modelos – Problema P2

		RFS–GA	RFS–DE	DFRFS–DE
Desempenho				
Treino	Melhor	0,0852	0,0581	0,0052
	Pior	0,0878	0,0787	0,0060
	Médio ± Desvio	$0{,}0865 \pm 7{,}5539 \times 10^{-4}$	0,0667 ± 0,0047	$\mathbf{0{,}0055 \pm 1{,}8041 \times 10^{-4}}$
	Interv. Conf.	[0,0862; 0,0867]	[0,0650; 0,0683]	[0,0054; 0,0055]
Teste	Melhor	0,0892	0,0200	0,0100
	Pior	0,2309	0,1139	0,0766
	Médio ± Desvio	0,1505 ± 0,0359	0,0573 ± 0,0204	**0,0220 ± 0,0124**
	Interv. Conf.	[0,1389; 0,1642]	[0,0511; 0,0656]	[0,0190; 0,0294]
Complexidade				
	Menor	8	16	16
	Maior	27	81	45
	Média ± Desvio	**13 ± 4**	36 ± 23	32 ± 6
	Interv. Conf.	[12; 15]	[29; 44]	[30; 34]

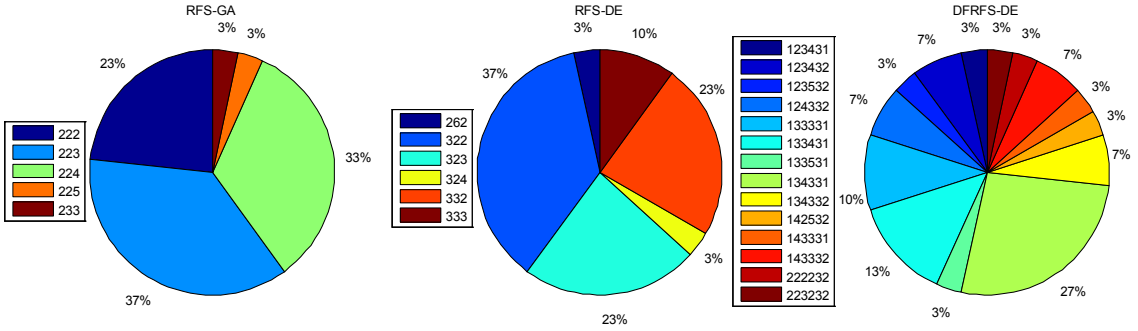


Figura B.2. Distribuição das estruturas selecionadas nas 30 execuções, segundo o critério AIC – Problema P2

▪ Problema P3 – Série Caótica Univariada de Hénon

Tabela B.3. Resultado comparativo entre o modelo original e o proposto – Problema P3

		RFS–GA	RFS–DE	DFRFS–DE
Desempenho				
Treino	Melhor	0,0766	0,0346	0,0356
	Pior	0,1033	0,0578	0,0382
	Médio ± Desvio	0,0907 ± 0,0050	0,0445 ± 0,0062	**0,0368 ± 6,9978** $\times$ **10^{-4}**
	Interv. Conf.	[0,0889; 0,0924]	[0,0425; 0,0467]	[0,0366; 0,0371]
Teste	Melhor	0,1093	0,0392	0,0211
	Pior	0,1428	0,0748	0,0263
	Médio ± Desvio	0,1280 ± 0,0060	0,0509 ± 0,0098	**0,0236 ± 0,0012**
	Interv. Conf.	[0,1258; 0,1300]	[0,0480; 0,0550]	[0,0232; 0,0241]
Complexidade				
	Menor	32	56	72
	Maior	144	144	135
	Média ± Desvio	**42 ± 22**	126 ± 23	97 ± 25
	Interv. Conf.	[26; 55]	[115; 133]	[89; 107]

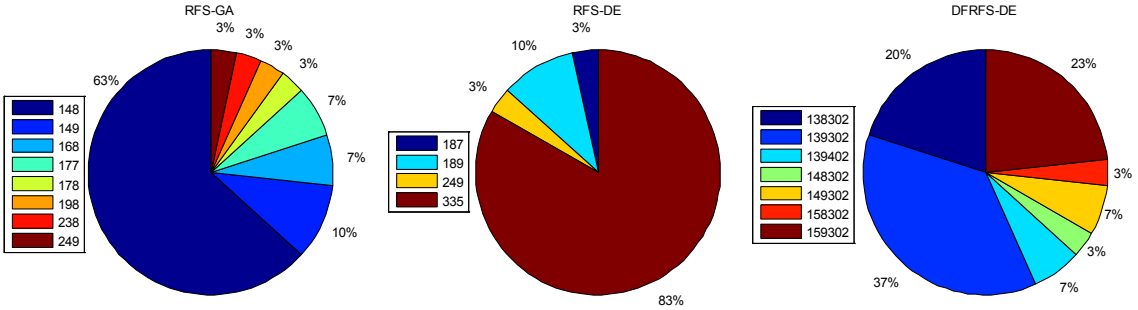


Figura B.3. Distribuição das estruturas selecionadas nas 30 execuções, segundo o critério AIC – Problema P3

▪ Problema P4 – Série Caótica Multivariada de Lorenz

• Dimensão X

Tabela B.4. Resultado comparativo entre o modelo original e o proposto – Problema P4(X)

		RFS–GA	RFS–DE	DFRFS–DE
Desempenho				
Treino	Melhor	1,2883	0,7395	0,8628
	Pior	1,5535	0,8349	0,8864
	Médio ± Desvio	1,3825 ± 0,0602	**0,7810 ± 0,0273**	0,8785 ± 0,0067
	Interv. Conf.	[1,3610; 1,4034]	[0,7726; 0,7914]	[0,8759; 0,8806]
Teste	Melhor	1,2481	0,6205	0,5767
	Pior	5,1671	0,7719	0,8022
	Médio ± Desvio	1,8427 ± 0,9263	0,6553 ± 0,0442	**0,6508 ± 0,0467**
	Interv. Conf.	[1,5918; 2,2929]	[0,6407; 0,6724]	[0,6365; 0,6691]
Complexidade				
	Menor	27	56	56
	Maior	144	135	112
	Média ± Desvio	89 ± 28	101 ± 26	**75 ± 19**
	Interv. Conf.	[77; 98]	[92; 110]	[69; 84]

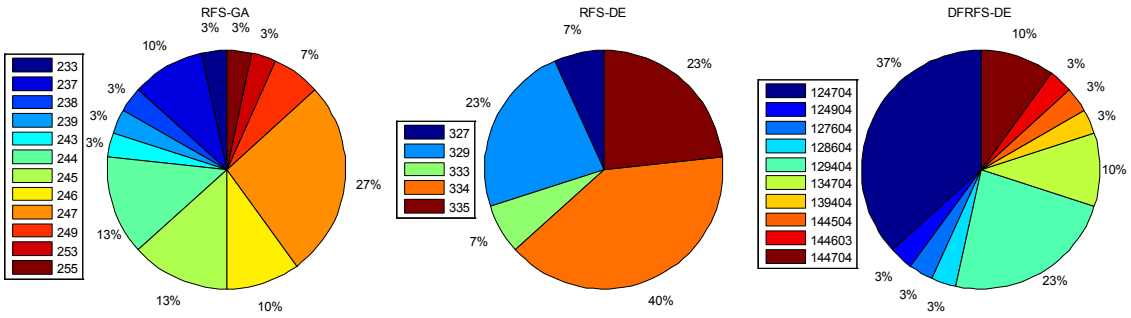


Figura B.4. Distribuição das estruturas selecionadas nas 30 execuções, segundo o critério AIC – Problema P4(X)

- **Dimensão Y**

Tabela B.5. Resultado comparativo entre o modelo original e o proposto – Problema P4(Y)

		RFS–GA	RFS–DE	DFRFS–DE
Desempenho				
Treino	Melhor	1,7992	1,1376	1,1709
	Pior	2,1444	1,3022	1,2203
	Médio ± Desvio	1,9819 ± 0,0781	1,2255 ± 0,0456	**1,1969 ± 0,0136**
	Interv. Conf.	[1,9523; 2,0061]	[1,2092; 1,2403]	[1,1918; 1,2016]
Teste	Melhor	1,8695	1,1030	0,8619
	Pior	3,7737	1,3403	1,1092
	Médio ± Desvio	2,4162 ± 0,5115	1,1868 ± 0,0667	**0,9722 ± 0,0545**
	Interv. Conf.	[2,2650; 2,6352]	[1,1653; 1,2125]	[0,9518; 0,9915]
Complexidade				
	Menor	48	54	56
	Maior	144	108	126
	Média ± Desvio	82 ± 26	96 ± 20	**81 ± 17**
	Interv. Conf.	[73; 92]	[88; 103]	[75; 87]

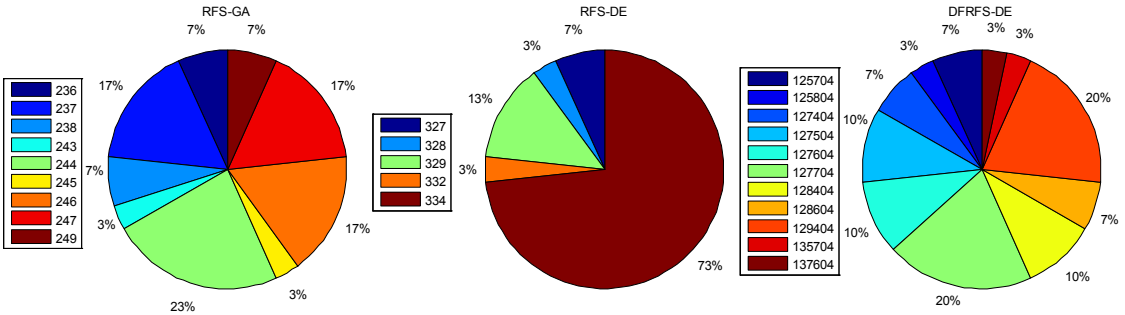


Figura B.5. Distribuição das estruturas selecionadas nas 30 execuções, segundo o critério AIC – Problema P4(Y)

- **Dimensão Z**

Tabela B.6. Resultado comparativo entre o modelo original e o proposto – Problema P4(Z)

		RFS–GA	RFS–DE	DFRFS–DE
Desempenho				
Treino	Melhor	2,1243	1,2257	1,2869
	Pior	2,3772	1,3231	1,4746
	Médio ± Desvio	2,2570 ± 0,0766	**1,2676 ± 0,0235**	1,3614 ± 0,0420
	Interv. Conf.	[2,2299; 2,2829]	[1,2605; 1,2769]	[1,3474; 1,3783]
Teste	Melhor	1,9897	1,2944	1,0428
	Pior	2,3519	1,3692	2,6358
	Médio ± Desvio	2,1401 ± 0,0828	1,3155 ± 0,0183	**1,3067 ± 0,3461**
	Interv. Conf.	[2,1152; 2,1712]	[1,3096; 1,3232]	[1,2210; 1,4634]
Complexidade				
	Menor	28	56	56
	Maior	112	72	144
	Média ± Desvio	**57 ± 22**	71 ± 4	99 ± 25
	Interv. Conf.	[50; 65]	[69; 71]	[91; 109]

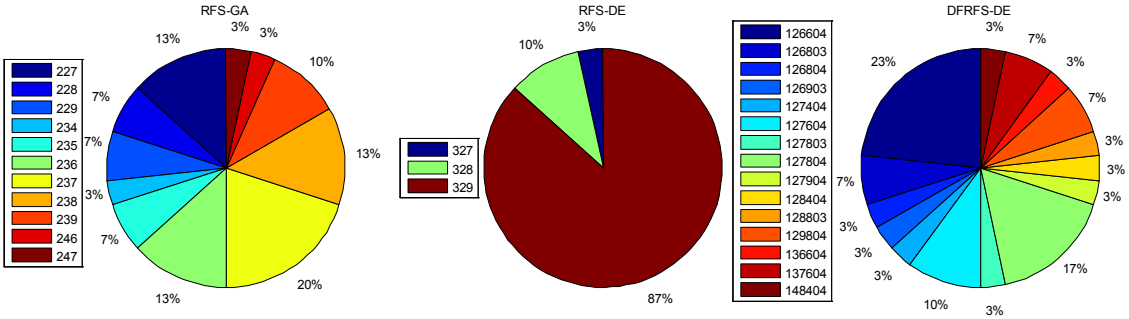


Figura B.6. Distribuição das estruturas selecionadas nas 30 execuções, segundo o critério AIC – Problema P4(Z)

▪ Exploração do Espaço de Busca pela Abordagem Proposta

Tabela B.7. Exploração do espaço de busca (quantidade de estruturas geradas) – Simulação 2.

	Válidas	Repetidas	Inválidas
P1			
Mínima	111	129	75
Máxima	223	288	172
Média ± Desvio	159 ± 28	198 ± 44	135 ± 23
Interv. Conf.	[150; 169]	[184; 214]	[126; 142]
P2			
Mínima	131	172	12
Máxima	253	335	69
Média ± Desvio	208 ± 32	249 ± 38	35 ± 12
Interv. Conf.	[197; 219]	[236; 263]	[31; 40]
P3			
Mínima	106	140	41
Máxima	233	334	141
Média ± Desvio	166 ± 29	229 ± 48	97 ± 28
Interv. Conf.	[156; 177]	[212; 245]	[87; 107]
P4(X)			
Mínima	126	107	29
Máxima	248	337	145
Média ± Desvio	201 ± 25	203 ± 49	88 ± 30
Interv. Conf.	[190; 208]	[187; 220]	[79; 99]
P4(Y)			
Mínima	148	124	42
Máxima	253	294	140
Média ± Desvio	199 ± 25	211 ± 46	83 ± 26
Interv. Conf.	[190; 208]	[195; 227]	[75; 93]
P4(Z)			
Mínima	162	117	57
Máxima	223	255	162
Média ± Desvio	198 ± 19	178 ± 38	117 ± 26
Interv. Conf.	[190; 204]	[165; 192]	[107; 126]

Apêndice C.

Resultados Detalhados da Simulação 3

▪ Problema P1 – Planta 1

Tabela C.1. Resultado comparativo entre os tipos de feedback – Problema P1

		Saída Estimada – $\hat{y}$	Saída Observada – y	Erro – e
Desempenho				
Treino	Melhor	0,0248	0,0245	0,0259
	Pior	0,0308	0,0433	0,0475
	Médio ± Desvio	**0,0265 ± 0,0014**	0,0267 ± 0,0033	0,0324 ± 0,0078
	Interv. Conf.	[0,0262; 0,0272]	[0,0260; 0,0294]	[0,0301; 0,0356]
Teste	Melhor	0,0186	0,0165	0,0332
	Pior	0,0590	0,0429	0,4380
	Médio ± Desvio	0,0286 ± 0,0081	**0,0256 ± 0,0063**	0,1114 ± 0,0821
	Interv. Conf.	[0,0264; 0,0324]	[0,0237; 0,0281]	[0,0896; 0,1497]
Complexidade				
	Menor	72	72	54
	Maior	108	108	108
	Média ± Desvio	92 ± 16	**92 ± 14**	104 ± 14
	Interv. Conf.	[86;97]	[87; 97]	[96; 107]

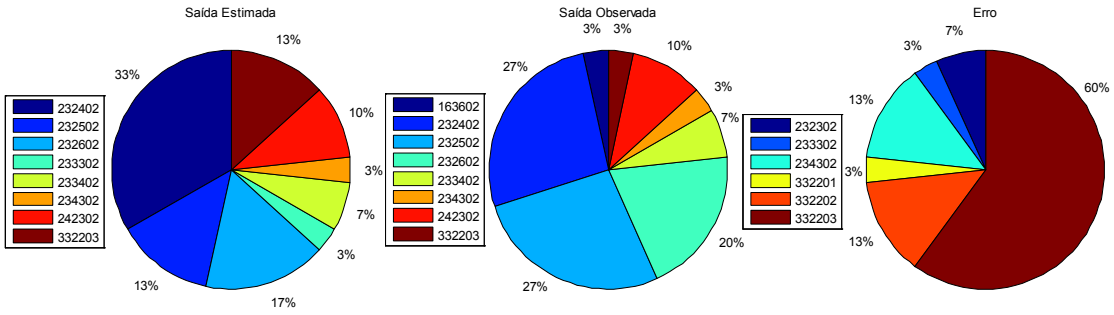
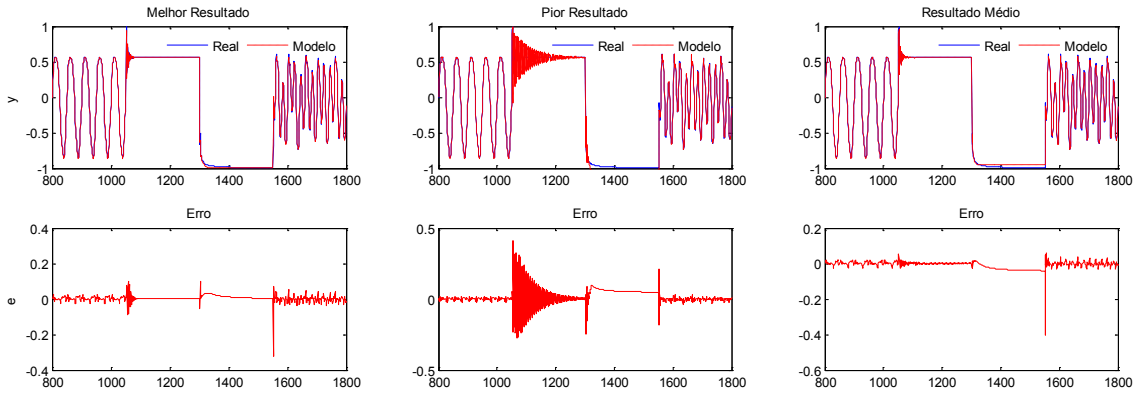
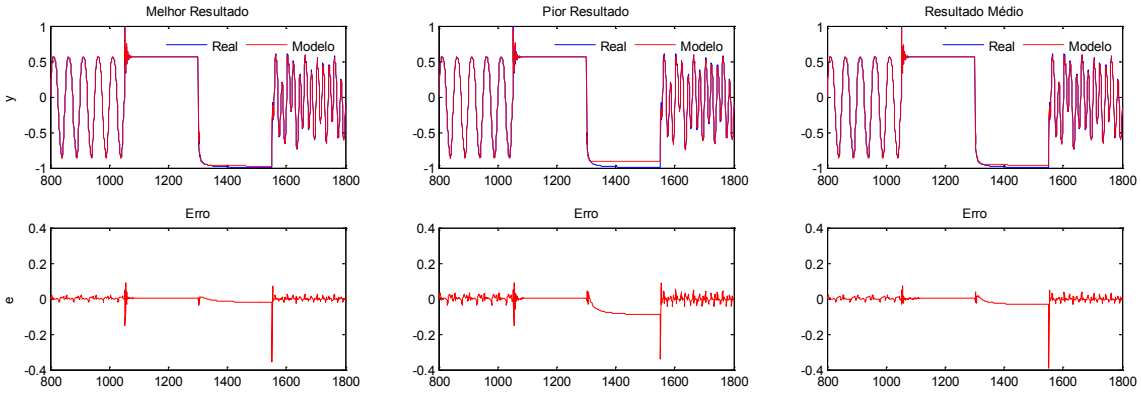


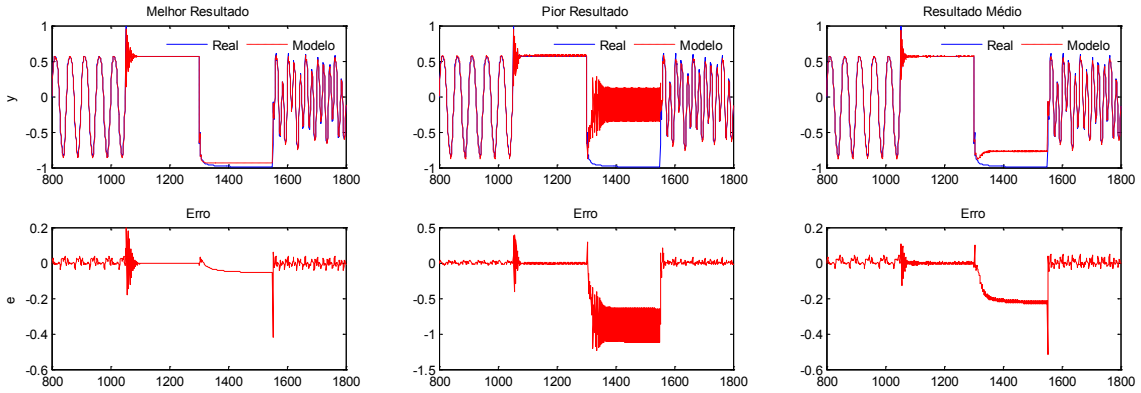
Figura C.1. Distribuição das estruturas selecionadas nas 30 execuções, segundo o critério AIC – Problema P1



(a) Saída Estimada



(b) Saída Observada



(c) Erro

Figura C.2. Resultados dos modelos nas 30 execuções segundo o tipo de feedback – Problema P1

- **Problema P2 – Planta 2**

Tabela C.2. Resultado comparativo entre os tipos de feedback – Problema P2

		Saída Estimada – $\hat{y}$	Saída Observada – y	Erro – e
Desempenho				
Treino	Melhor	0,0052	0,0053	0,0049
	Pior	0,0060	0,0061	0,0051
	Médio ± Desvio	$0{,}0055 \pm 1{,}8041 \times 10^{-4}$	$0{,}0055 \pm 1{,}4644 \times 10^{-4}$	$\mathbf{0{,}0050 \pm 4.5229 \times 10^{-5}}$
	Interv. Conf.	[0,0054; 0,0055]	[0,0054; 0,0055]	[0,0050; 0,0050]
Teste	Melhor	0,0100	0,0112	0,0042
	Pior	0,0766	0,0474	0,1915
	Médio ± Desvio	0,0220 ± 0,0124	**0,0217 ± 0,0098**	0,0220 ± 0,0405
	Interv. Conf.	[0,0190; 0,0294]	[0,0190; 0,0264]	[0,0116; 0,0426]
Complexidade				
	Menor	16	16	36
	Maior	45	36	81
	Média ± Desvio	32 ± 6	**29 ± 6**	38 ± 8
	Interv. Conf.	[30; 34]	[27; 31]	[36; 44]

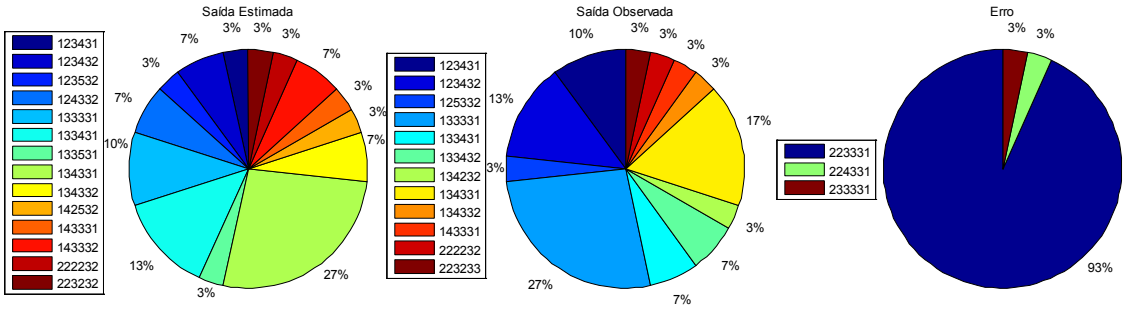
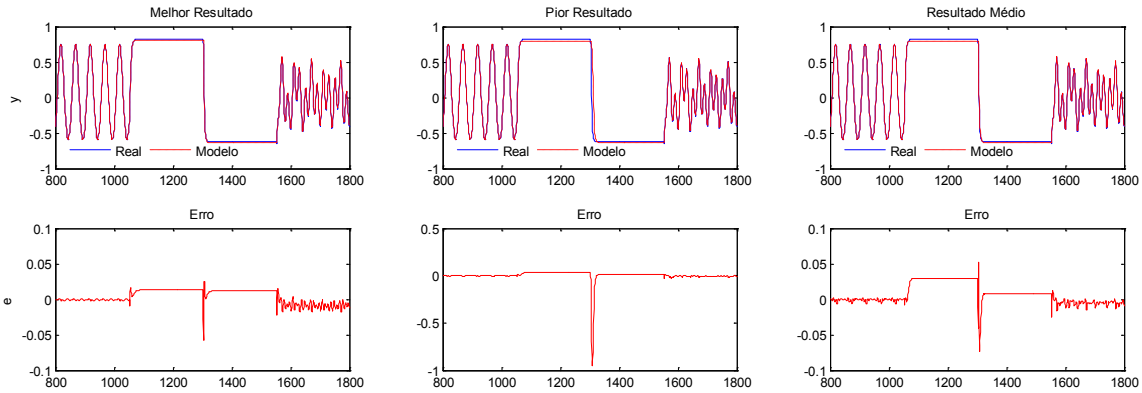
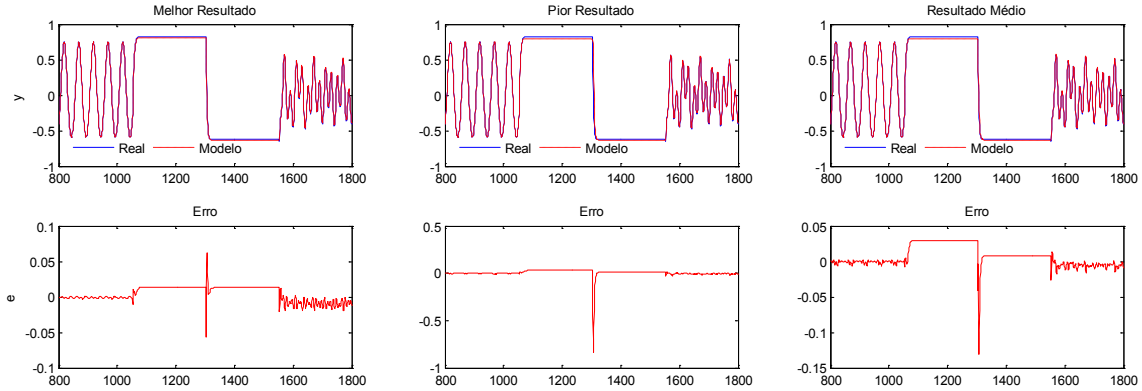


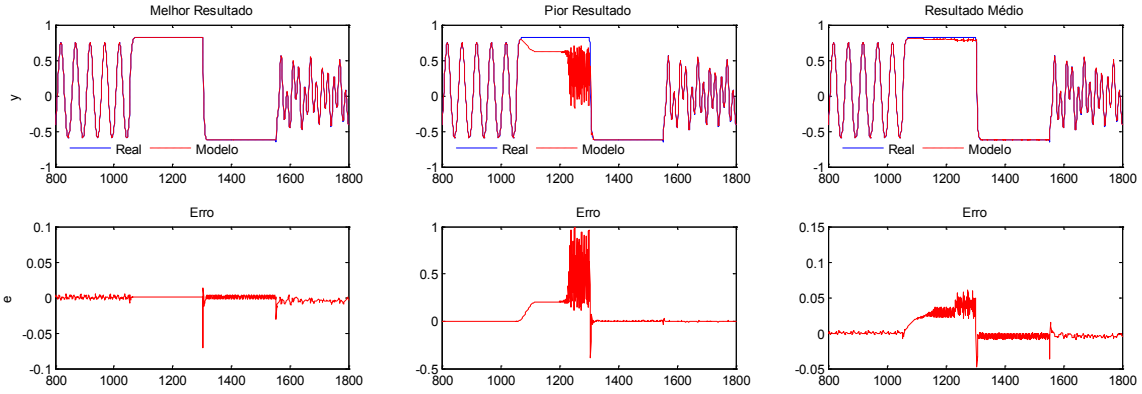
Figura C.3. Distribuição das estruturas selecionadas nas 30 execuções, segundo o critério AIC – Problema P2



(a) Saída Estimada



(b) Saída Observada



(c) Erro

Figura C.4. Resultados dos modelos nas 30 execuções segundo o tipo de feedback – Problema P2

▪ Problema P3 – Série Caótica Univariada de Hénon

Tabela C.3. Resultado comparativo entre os tipos de feedback – Problema P3

		Saída Estimada – $\hat{y}$	Saída Observada – y	Erro – e
Desempenho				
Treino	Melhor	0,0356	0,0345	0,0539
	Pior	0,0382	0,0444	0,0797
	Médio ± Desvio	$0{,}0368 \pm 6{,}9978 \times 10^{-4}$	**0,0364 ± 0,0020**	0,0662 ± 0,0078
	Interv. Conf.	[0,0366; 0,0371]	[0,0359; 0,0376]	[0,0636; 0,0688]
Teste	Melhor	0,0211	0,0186	0,0504
	Pior	0,0263	0,0374	0,1113
	Médio ± Desvio	0,0236 ± 0,0012	**0,0225 ± 0,0042**	0,0740 ± 0,0156
	Interv. Conf.	[0,0232; 0,0241]	[0,0214; 0,0246]	[0,0688; 0,0798]
Complexidade				
	Menor	72	72	64
	Maior	135	144	144
	Média ± Desvio	**97 ± 25**	114 ± 28	108 ± 24
	Interv. Conf.	[89; 107]	[102; 123]	[99; 116]

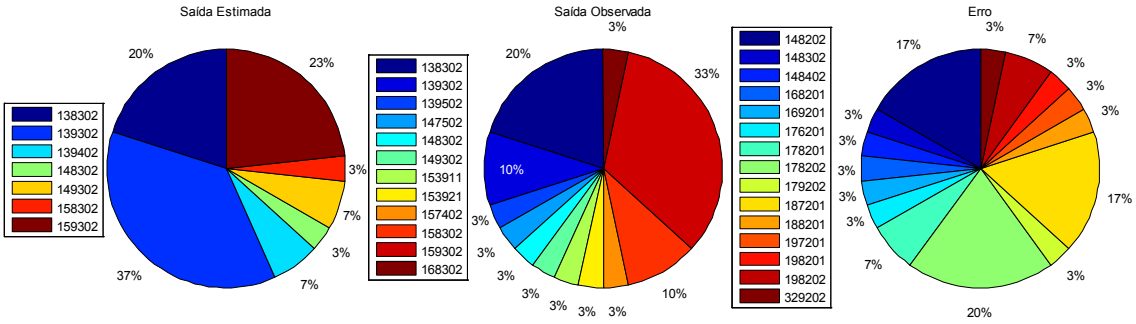
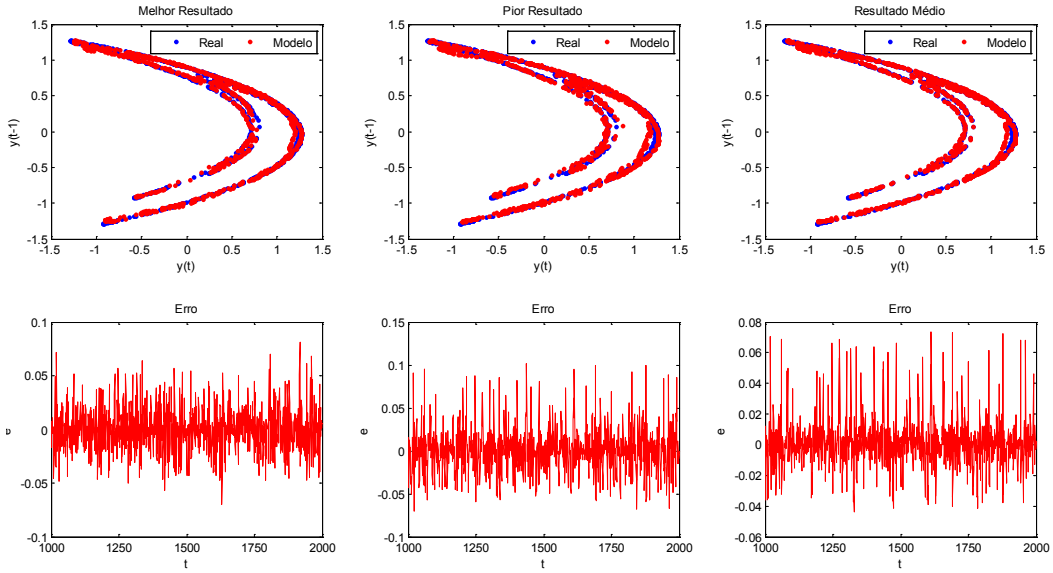
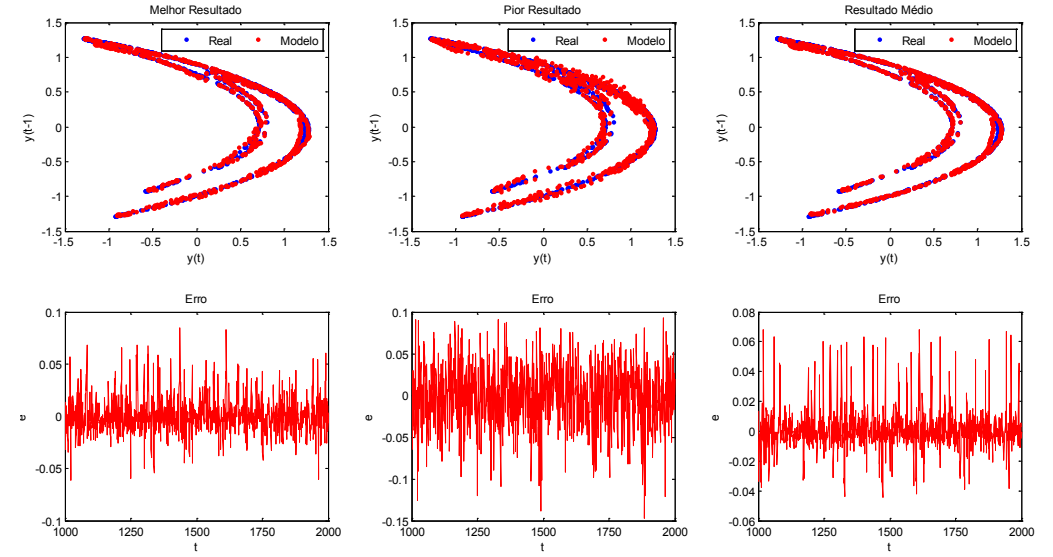


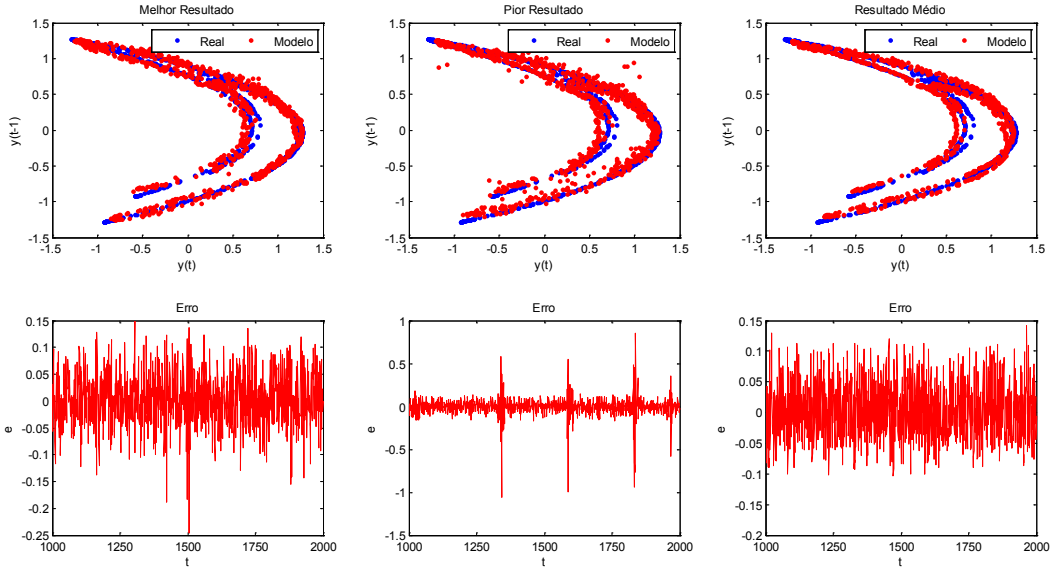
Figura C.5. Distribuição das estruturas selecionadas nas 30 execuções, segundo o critério AIC – Problema P3



(a) Saída Estimada



(b) Saída Observada



(c) Erro

Figura C.6. Resultados dos modelos nas 30 execuções segundo o tipo de feedback – Problema P3

▪ Problema P4 – Série Caótica Multivariada de Lorenz

• Dimensão X

Tabela C.4. Resultado comparativo entre os tipos de feedback – Problema P4(X)

		Saída Estimada – $\widehat{y}$	Saída Observada – y	Erro – e
Desempenho				
Treino	Melhor	0,8628	0,7010	0,8381
	Pior	0,8864	0,8662	1,0542
	Médio ± Desvio	0,8785 ± 0,0067	**0,7697 ± 0,0393**	0,9770 ± 0,0569
	Interv. Conf.	[0,8759; 0,8806]	[0,7554; 0,7840]	[0,0965; 0,9959]
Teste	Melhor	0,5767	0,4859	0,5923
	Pior	0,8022	0,5968	3,9616
	Médio ± Desvio	0,6508 ± 0,0467	**0,5486 ± 0,0328**	0,8235 ± 0,6023
	Interv. Conf.	[0,6365; 0,6691]	[0,5372; 0,5601]	[0,7049; 1,3640]
Complexidade				
	Menor	56	50	48
	Maior	112	144	128
	Média ± Desvio	**75 ± 19**	91 ± 20	85 ± 20
	Interv. Conf.	[69; 84]	[84; 98]	[78; 92]

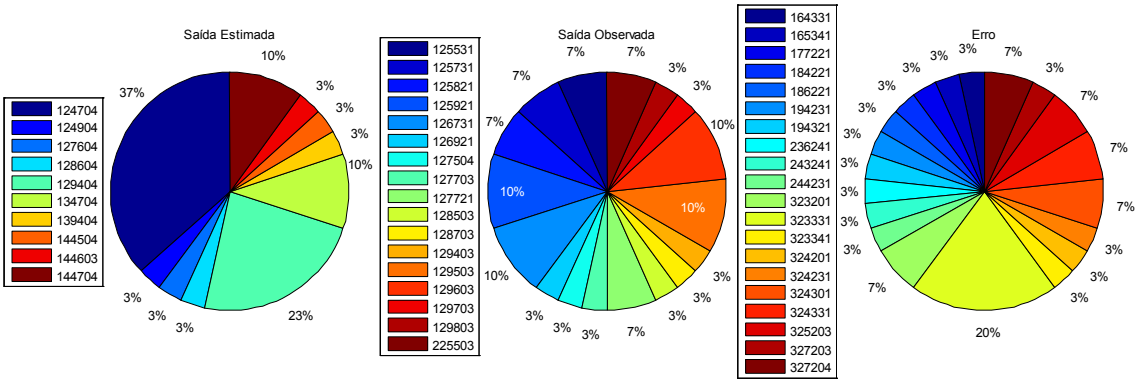
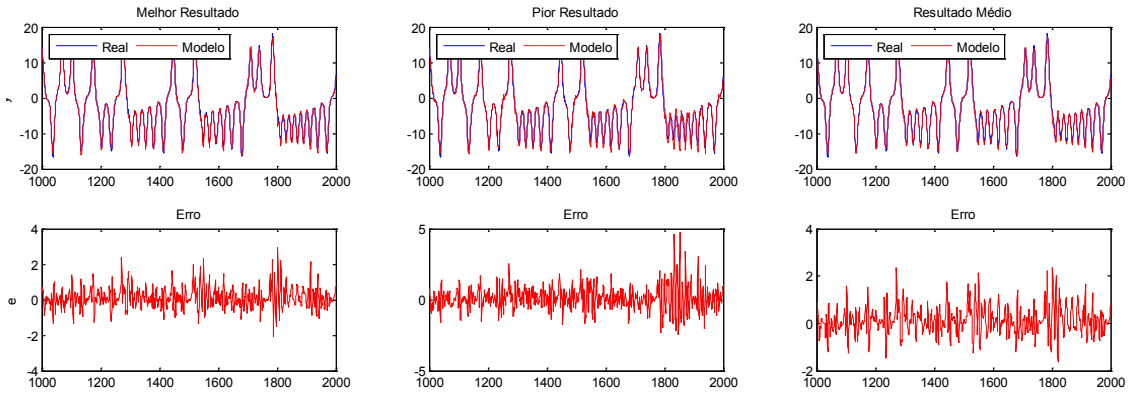
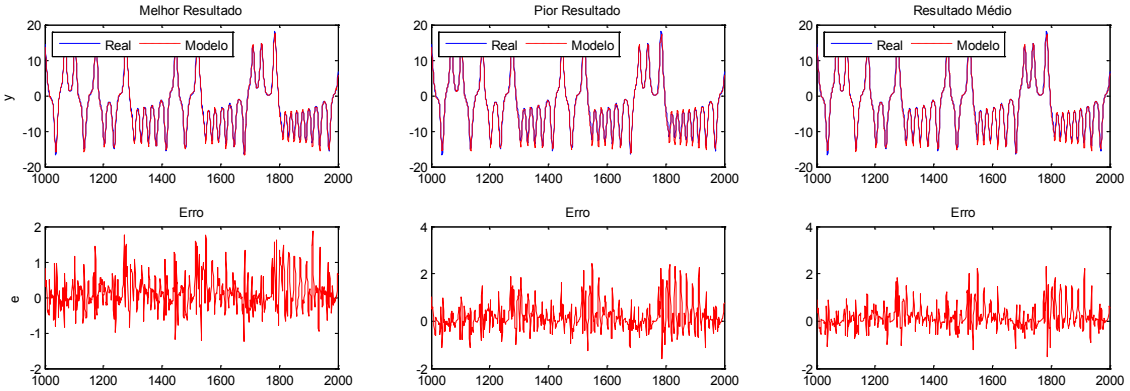


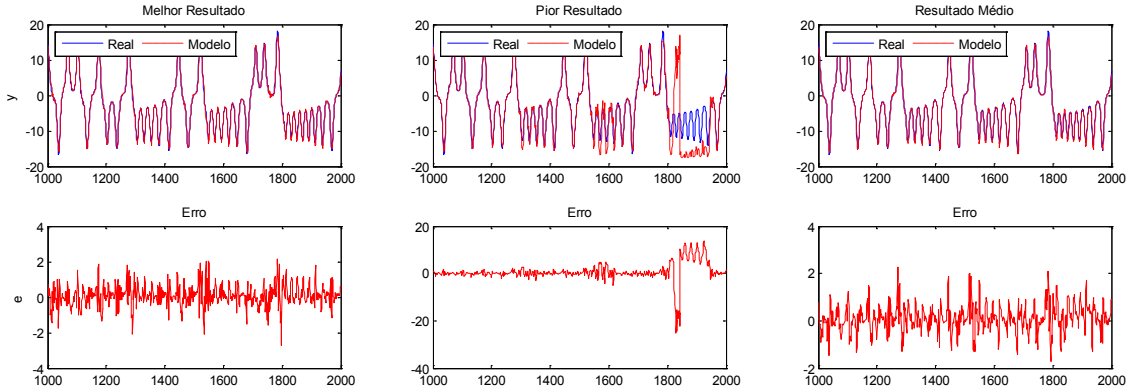
Figura C.7. Distribuição das estruturas selecionadas nas 30 execuções, segundo o critério AIC – Problema P4(X)



(a) Saída Estimada



(b) Saída Observada



(c) Erro

Figura C.8. Resultados dos modelos nas 30 execuções segundo o tipo de feedback – Problema P4(X)

- **Dimensão Y**

Tabela C.5. Resultado comparativo entre os tipos de feedback – Problema P4(Y)

		Saída Estimada – $\hat{y}$	Saída Observada – y	Erro – e
Desempenho				
Treino	Melhor	1,1709	0,8785	1,2401
	Pior	1,2203	1,0872	1,4418
	Médio ± Desvio	1,1969 ± 0,0136	**1,0157 ± 0,0492**	1,3603 ± 0,0502
	Interv. Conf.	[1,1918; 1,2016]	[0,9979; 1,0322]	[1,3427; 1,3773]
Teste	Melhor	0,8619	0,8345	0,9637
	Pior	1,1092	0,9968	6,6512
	Médio ± Desvio	0,9722 ± 0,0545	**0,9019 ± 0,0470**	1,4533 ± 1,1888
	Interv. Conf.	[0,9518; 0,9915]	[0,8866; 0,9203]	[1,1477; 2,0996]
Complexidade				
	Menor	56	80	63
	Maior	126	147	144
	Média ± Desvio	**81 ± 17**	100 ± 17	99 ± 22
	Interv. Conf.	[75; 87]	[95; 107]	[92; 106]

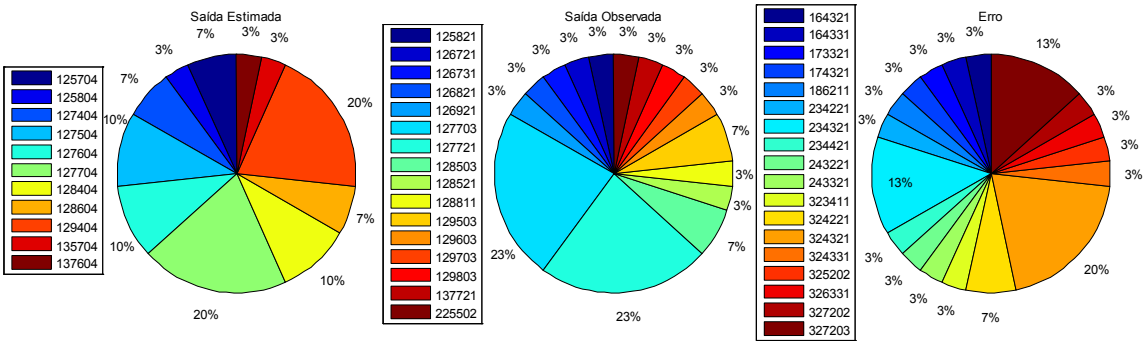
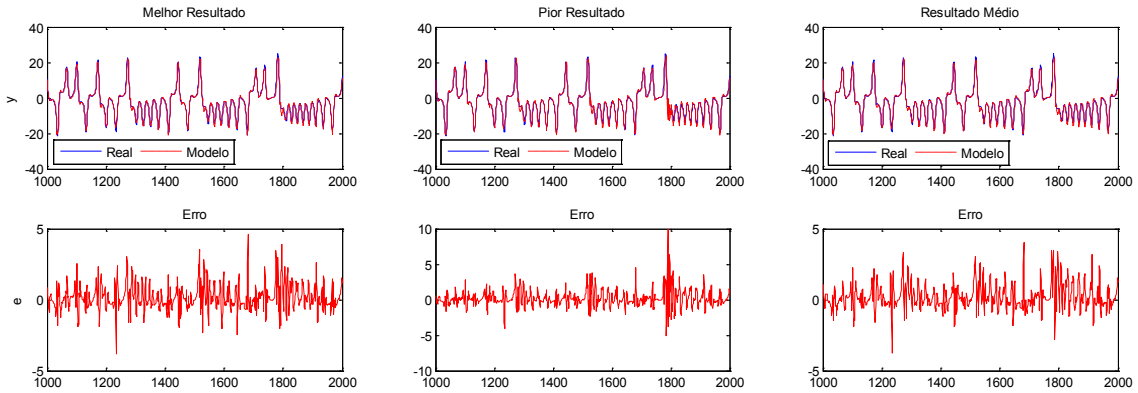
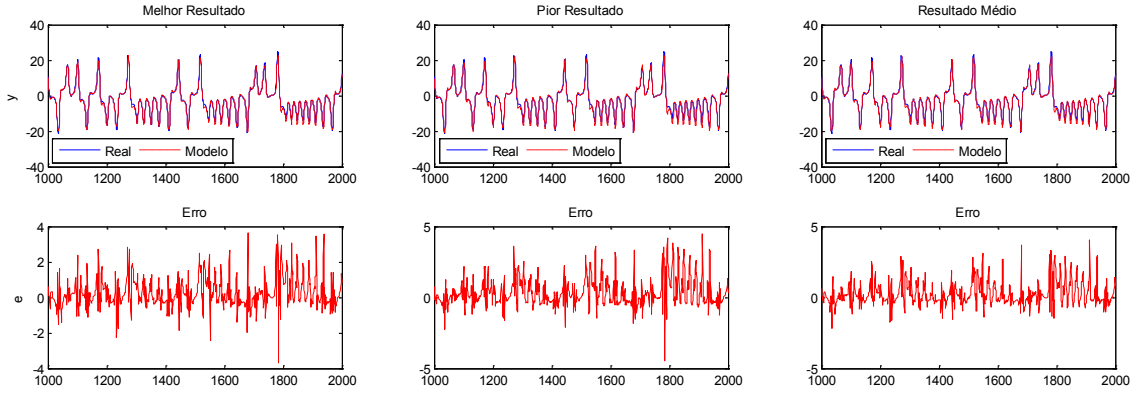


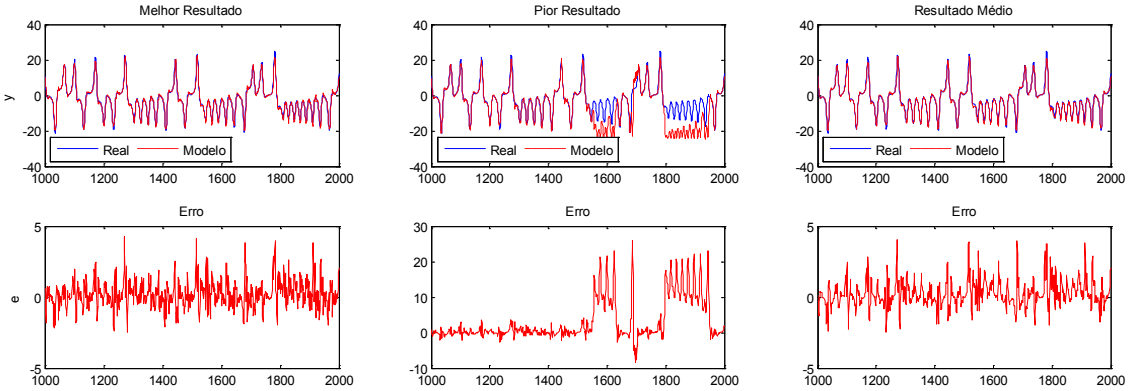
Figura C.9. Distribuição das estruturas selecionadas nas 30 execuções, segundo o critério AIC – Problema P4(Y)



(a) Saída Estimada



(b) Saída Observada



(c) Erro

Figura C.10. Resultados dos modelos nas 30 execuções segundo o tipo de feedback – Problema P4(Y)

- **Dimensão Z**

Tabela C.6. Resultado comparativo entre os tipos de feedback – Problema P4(Z)

		Saída Estimada – $\hat{y}$	Saída Observada – y	Erro – e
Desempenho				
Treino	Melhor	1,2869	1,0737	1,3118
	Pior	1,4746	1,1569	1,4766
	Médio ± Desvio	1,3614 ± 0,0420	**1,1027 ± 0,0211**	1,3844 ± 0,0458
	Interv. Conf.	[1,3474; 1,3783]	[1,0960; 1,1107]	[1,3687; 1,4018]
Teste	Melhor	1,0428	0,8132	1,0652
	Pior	2,6358	0,8992	2,0682
	Médio ± Desvio	1,3067 ± 0,3461	**0,8547 ± 0,0205**	1,3126 ± 0,2450
	Interv. Conf.	[1,2210; 1,4634]	[0,8472; 0,8615]	[1,2430; 1,4137]
Complexidade				
	Menor	56	56	60
	Maior	144	147	144
	Média ± Desvio	**99 ± 25**	104 ± 22	100 ± 20
	Interv. Conf.	[91; 109]	[97; 112]	[93; 107]

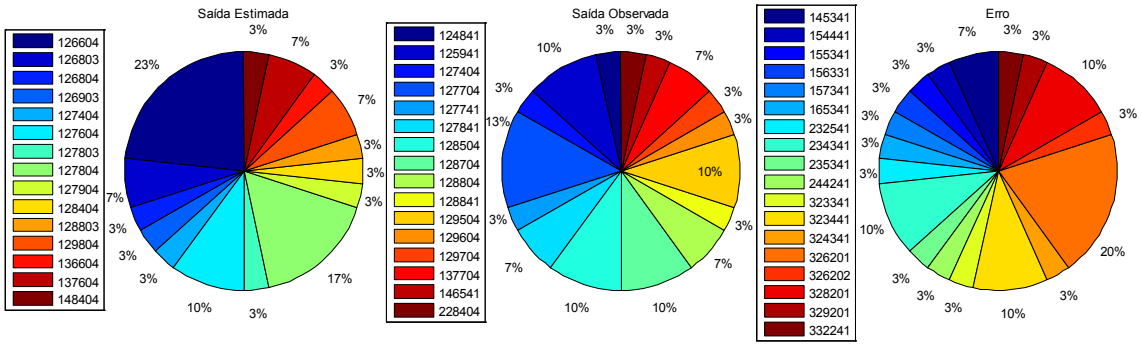
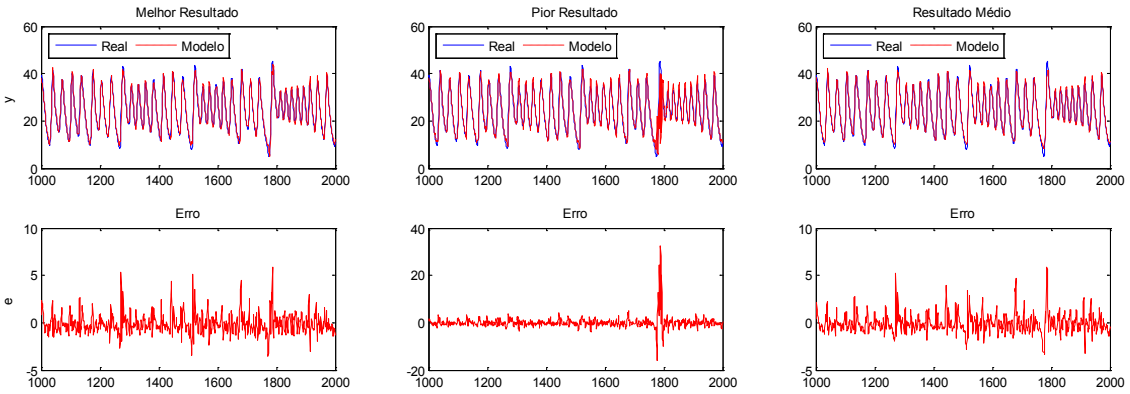
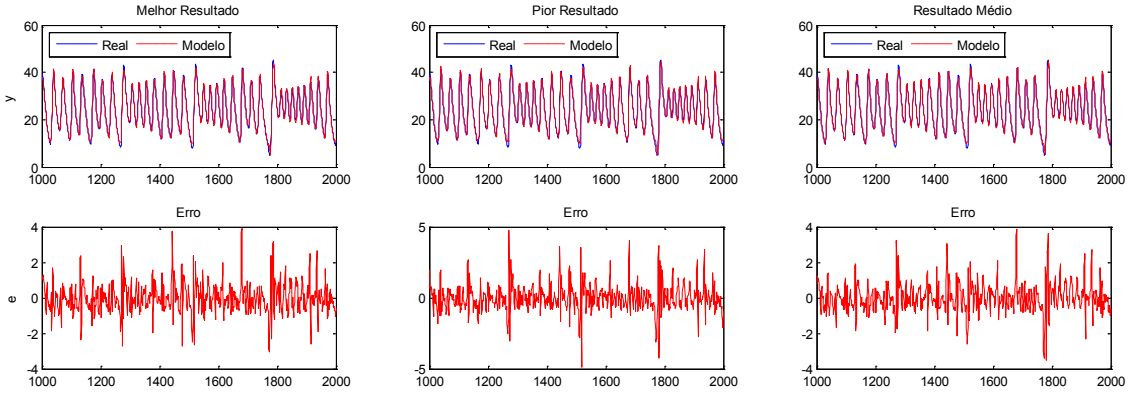


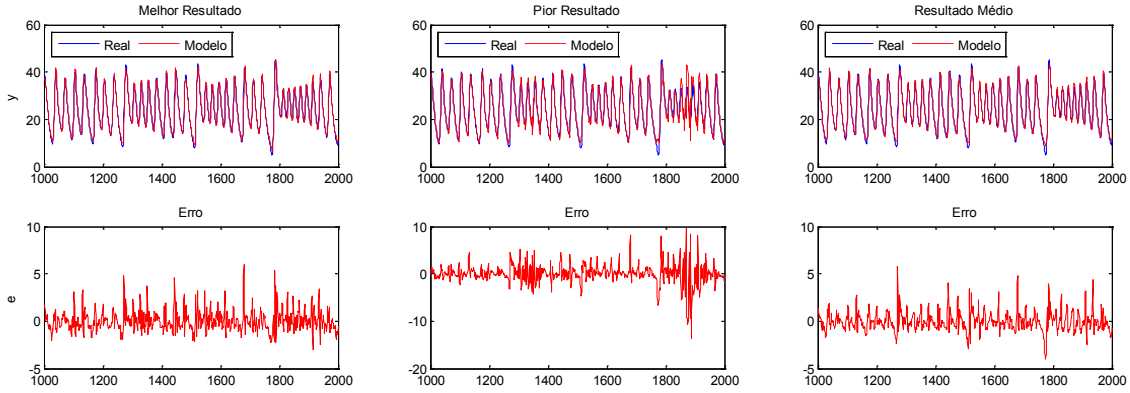
Figura C.11. Distribuição das estruturas selecionadas nas 30 execuções, segundo o critério AIC – Problema P4(Z)



(a) Saída Estimada



(b) Saída Observada



(c) Erro

Figura C.12. Resultados dos modelos nas 30 execuções segundo o tipo de feedback – Problema P4(Z)