PROGRAMAÇÃO GENÉTICA APLICADA À IDENTIFICAÇÃO DE ACIDENTES DE UMA USINA NUCLEAR PWR

Victor Henrique Cabral Pinheiro

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Nuclear, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Nuclear.

Orientador: Roberto Schirru

Rio de Janeiro

Fevereiro de 2018

PROGRAMAÇÃO GENÉTICA APLICADA À IDENTIFICAÇÃO DE ACIDENTES DE UMA USINA NUCLEAR PWR

Victor Henrique Cabral Pinheiro

DISSERTAÇÃO 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 MESTRE EM CIÊNCIAS EM ENGENHARIA NUCLEAR.

Examinada por:

_____

Prof. Roberto Schirru, D.Sc.

_____

Prof. Fernando Carvalho da Silva, D.Sc.

_____

Prof. Cláudio Márcio do Nascimento Abreu Pereira, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

FEVEREIRO DE 2018

Pinheiro, Victor Henrique Cabral

Programação Genética Aplicada à Identificação de Acidentes de uma Usina Nuclear PWR/ Victor Henrique Cabral Pinheiro. – Rio de Janeiro: UFRJ/COPPE, 2018.

XI, 91 p.: il.; 29,7 cm.

Orientador: Roberto Schirru

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Nuclear, 2018.

Referências Bibliográficas: p. 70-87.

1. Sistemas de diagnóstico. 2. Programação genética. 3. Reconhecimento de padrões. I. Schirru, Roberto. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Nuclear. III. Título.

À minha família.

AGRADECIMENTOS

Agradeço à minha família querida – principalmente minha mamãe, meu pai e meu irmão – pelo apoio, pelos momentos de alegria e por terem proporcionado que eu estudasse e pudesse chegar até aqui.

Agradeço ao meu orientador, professor Roberto Schirru, por toda a orientação (o que pode parecer redundante, mas não é!). Devo agradecimentos pelo excelente suporte e mentoria, pela paciência, pela disponibilidade, pelos conselhos, pela paciência e ainda mais uma vez pela paciência. Foi uma experiência excelente ser seu aluno de mestrado, e através dela acho que amadureci e cresci, portanto, agradeço.

Agradeço ao pessoal do LMP pelo auxílio, por estarem sempre prontos a me tirar dúvidas e orientarem com boa vontade quando necessário. Destaco aqui (dentre muitos!) os pesquisadores Andressa Nicolau e Alan de Lima.

Agradeço aos amigos de dentro e de fora da universidade. Aos de dentro, pelo companheirismo durante o mestrado e por terem tornado essa jornada mais agradável, e aos de fora, pelo encorajamento e companhia.

Por fim, quero agradecer de maneira geral ao restante do corpo docente e técnico-administrativo do PEN/COPPE. Tive uma grata surpresa pela maneira atenciosa e agradável como acolheram a mim e aos demais alunos, e com como trabalham com diligência e dedicação para alcançar e manter o padrão de qualidade apresentado pelo programa.

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

PROGRAMAÇÃO GENÉTICA APLICADA À IDENTIFICAÇÃO DE ACIDENTES DE UMA USINA NUCLEAR PWR

Victor Henrique Cabral Pinheiro

Fevereiro/2018

Orientador: Roberto Schirru

Programa: Engenharia Nuclear

Neste trabalho são apresentados os resultados do estudo que avaliou a performance do algoritmo de computação evolucionária programação genética como ferramenta de otimização e geração de atributos em um sistema de reconhecimento de padrões para identificação e diagnóstico de acidentes de uma usina nuclear com reator de água pressurizada. São apresentados ainda as bases de um sistema de reconhecimento de padrões, o estado da arte da programação genética e de sistemas similares de diagnóstico de acidentes e transientes de usinas nucleares. Dentro do conjunto da evolução temporal de 17 variáveis operacionais dos três acidentes/transientes considerado, além da condição normal, a função da programação genética foi evoluir regressores não lineares de combinações dessas variáveis que fornecessem o máximo de informação discriminatória para cada um dos eventos. Após testes exaustivos com diversas associações de variáveis, a programação genética se mostrou uma metodologia capaz de fornecer taxas de acerto de, ou muito próximas de, 100%, com parametrizações do algoritmo relativamente simples e em tempo de treinamento bastante razoável, mostrando ser capaz de fornecer resultados compatíveis e até superiores a outros sistemas disponíveis na literatura, com a vantagem adicional de requerer pouco (e muitas vezes nenhum) pré-tratamento nos dados.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

GENETIC PROGRAMMING APPLIED TO THE IDENTIFICATION OF ACCIDENTS OF A PWR NUCLEAR POWER PLANT

Victor Henrique Cabral Pinheiro

February/2018

Advisor: Roberto Schirru

Department: Nuclear Engineering

This work presentes the results of the study that evaluated the efficiency of the evolutionary computation algorithm genetic programming as a technique for the optimization and feature generation at a pattern recognition system for the diagnostic of accidents in a pressurized water reactor nuclear power plant. The foundations of a typical pattern recognition system, the state of the art of genetic programming and of similar accident/transient diagnosis systems at nuclear power plants are also presented. Considering the set of the time evolution of seventeen operational variables for the three accident scenarios approached, plus normal condition, the task of genetic programming was to evolve non-linear regressors with combination of those variables that would provide the most discriminatory information for each of the events. After exhaustive tests with plenty of variable associations, genetic programming was proven to be a methodology capable of attaining success rates of, or very close to, 100%, with quite simple parametrization of the algorithm and at very reasonable time, putting itself in levels of performance similar or even superior as other similar systems available in the scientific literature, while also having the additional advantage of requiring very little pretreatment (sometimes none at all) of the data.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

ACRÔNIMOS E ABREVIAÇÕES

AG – Algoritmo Genético.

CE – Computação Evolucionária.

DDF – Detecção e Diagnóstico de Falhas.

EFH – Engenharia de Fatores Humanos.

GV – Gerador de Vapor.

IA – Inteligência Artificial.

LOCA – Loss of Coolant Accident.

NRC – Nuclear Regulatory Commission.

NUREG – United States Nuclear Regulatory Commission technical report designation.

PG – Programação Genética.

PWR – Pressurized Water Reactor.

RNA – Rede Neural Artificial.

RP – Reconhecimento de Padrões.

SGTR – Steam Generator Tube Rupture.

UN – Usina Nuclear.

CAPÍTULO 1

INTRODUÇÃO

1.1 – Contextualização do Problema

As usinas nucleares (UNs) são sistemas altamente complexos operados e monitorados por humanos. Quando confrontados com um transiente não planejado, como um cenário de acidente, falha de equipamento, distúrbio externo ao sistema, entre outros, o operador precisa realizar procedimentos de diagnóstico e ações corretivas baseados nas leituras fornecidas pelos instrumentos de medida dos processos. Dependendo da severidade e das características do evento anômalo, principalmente em seus estágios incipientes, essas leituras podem prover informações conflitantes, confusas e, por conseguinte, conducentes à identificação errada do evento e posteriores ações equivocadas, as quais ocasionam uma piora das condições de operação da usina, podendo levar, em últimas instâncias, a acidentes graves.

A capacidade dos operadores de identificarem correta e rapidamente os eventos depende de fatores como treinamento, sistemas de controle adequadamente construídos e procedimentos de emergência corretamente descritos. De fato, o trabalho humano, em situações de alto risco, é tradicionalmente baseado em tarefas sistemáticas e metódicas que prescrevem a forma mais correta e segura de realizar os serviços [1]. Levando em consideração o exposto até aqui, fica claro que é fundamental que, para realizar as tarefas e procedimentos exatamente como descrito, os operadores necessitam, tanto quanto possível, de informações corretas e sem ambiguidade no que concerne à situação da planta.

Dada a gravidade das consequências de acidentes em usinas nucleares (e especialmente após o acidente de Three Mile Island, em março de 1979), muita atenção tem sido dada, por parte da Comissão de Regulamentação Nuclear (NRC), à área de Engenharia de Fatores Humanos (EFH), no tocante à importância de melhorar a interface homem-máquina em projetos de salas de controle, e uma série de documentos normativos (como o NUREG 711 e o NUREG 800) tem sido estabelecida desde então para este fim. De particular interesse para este trabalho é o tópico referente à interface operador-sistema, presente na NUREG 711, o qual recomenda que a atividade do operador seja focada nas tarefas de monitoração, tomada de decisão e controle da usina. Sendo assim, sistemas que auxiliem objetivamente o operador no rápido entendimento do estado operacional são úteis para melhorar seu desempenho durante o trabalho.

Sob este paradigma de segurança, espera-se que o erro humano – isto é, qualquer desvio de performance considerando a sequência de ações especificadas nos procedimentos – seja evitado. Considerando que as centrais nucleares são instalações de segurança críticas, e dado o grande número de variáveis monitoradas simultaneamente (que levam a padrões complexos, dificultando o diagnóstico por parte do usuário), nos últimos anos muitos trabalhos relevantes que utilizam técnicas de inteligência artificial tem sido publicados com o objetivo de auxiliar o operador com ferramentas de diagnóstico de transientes, diminuindo sua carga cognitiva e oferecendo uma identificação automática robusta e confiável do estado da usina. Tal desafio se enquadra na área da inteligência artificial (IA) de reconhecimento de padrões.

O reconhecimento de padrões (RP) [2] é uma disciplina científica que busca classificar os dados ou objetos em diferentes classes usando alguma informação, seja ela um conhecimento *à priori* ou uma distribuição estatística dos dados. Essa disciplina tem se mostrado cada vez mais relevante e é utilizada em um vasto espectro de aplicações, incluindo reconhecimento de fala e de faces, processamento de imagem, diagnóstico médico, etc.

A formulação de um sistema para reconhecimento de padrões segue normalmente três etapas. Primeiro, um sensor (transdutor) extrai dos inputs as medidas, que são os "dados brutos". Então, esses dados brutos são processados de maneira a obter-se a informação relevante, num procedimento conhecido como *extração de atributos*. Como os atributos relevantes a um determinado sistema costumam ser um subespaço muito pequeno do total de atributos que podem ser extraídos analisando os dados brutos, as características[1] extraídas passam, em geral, por um processo de *redução dimensional*. Finalmente, um ou mais classificadores são empregados para dividir as amostras de dados em diferentes categorias, e o faz utilizando as informações discriminatórias das características extraídas. Tal classificador pode ser tão simples quanto um processo de decisão binária ou tão complexo quanto uma rede neural.

Fica claro da descrição acima que o processo de extração de atributos dos dados brutos é de vital importância para o reconhecimento de padrões, haja vista que ele é a ponte entre os dados brutos e o classificador. De fato, idealmente, as características (atributos) extraídas deveriam conter informação discriminatória suficiente para que o trabalho do classificador seja facilitado (o que, em geral, equivale a dizer mais "barato" computacionalmente). Infelizmente,

[1] No contexto de reconhecimento de padrões, e inclusive nesta dissertação, os termos "atributos" e "características" são utilizados de maneira intercambiável.

nem todas as características extraídas contém informação útil, sendo muitas delas redundantes, não correlacionadas ou ruidosas. Tais características precisam ser filtradas para que o classificador realize o seu trabalho de maneira mais eficiente.

A filtragem das características ruidosas é usualmente realizada através do processo de *seleção de atributos* e da *geração de atributos*, que compõem a redução dimensional mencionada anteriormente. Normalmente, a distribuição estatística dos dados auxilia no processo de seleção de atributos, no qual os atributos úteis são mantidos e os ruidosos são descartados. Já o processo de geração de atributos é mais complexo; nele, os atributos extraídos são transformados numa menor quantidade de novos atributos, de maneira tal que o máximo de informação discriminatória é mantido. Muitos métodos de geração de atributos – como análise do componente principal (Principal Component Analysis – PCA [3]), análise do componente independente (Independent Component Analysis - ICA) e sua generalização para o caso não linear (Maximum Variance Unfolding - MVU) – foram desenvolvidos e são aplicáveis à várias classes de problemas.

Tradicionalmente, o problema de identificação de acidentes em usinas nucleares tem sido abordado através de métodos estatísticos de RP e técnicas de inteligência artificial, como sistemas especialistas (SE), lógica nebulosa (LN), redes neurais artificiais (RNA), algoritmos genéticos (GA), algoritmos evolucionários com inspiração quântica, além de técnicas de inteligência de enxame, como otimização por enxame de partículas (particle swarm optimization - PSO) clássico e com inspiração quântica, entre outras, obtendo diferentes graus de sucesso (o Capítulo 4 oferece um estudo bibliográfico aprofundado).

A programação genética (PG) [4], em particular, foi o método aplicado neste estudo no problema de identificação de acidentes em usinas nucleares – pela primeira vez, de acordo com as pesquisas deste autor – como método de seleção e geração de atributos. A PG pertence a uma classe de algoritmos evolucionários, e foi proposta e padronizada em 1992 por John Koza como uma generalização do algoritmo genético (AG). De fato, ambos possuem sua inspiração na teoria da evolução biológica, mas a PG difere do AG e de outros métodos de aprendizado de máquina (“machine learning”) tradicionais ao considerar, como indivíduos a serem evoluídos, programas de computador, que são soluções em potencial para o problema em questão. Durante a evolução, a PG busca otimizar a população de indivíduos através de uma métrica de “fitness” (capacidade do indivíduo de resolver o problema) definida previamente.

A PG tem recebido muita atenção por parte da comunidade acadêmica devido a sua aplicabilidade em certos tipos de problemas que não são bem resolvidos por outras técnicas de aprendizado de máquina. Realmente, a programação genética, embora computacionalmente intensiva, costuma apresentar resultados bastante competitivos ou melhores que outras técnicas quando aplicada à problemas complexos ou não lineares. Algumas de suas principais vantagens são: é extremamente geral e aplicável à várias classes de problemas, não requer informação *à priori* com respeito à distribuição das séries de dados (ou mesmo pré-processamento dos mesmos), e além disso permite que sejam alterados os parâmetros ou a estrutura das soluções durante a evolução.

A motivação para tamanho interesse em PG para a redução dimensional se deve, ainda, ao fato de que ela tem a capacidade inerente de selecionar atributos úteis (como parte da evolução) e ignorar outros [5]. A transformação de atributos é realizada fazendo combinações não lineares, através de funções matemáticas, dos atributos extraídos.

Com o avanço da capacidade de processamento e tecnologia computacional, mesmo um computador pessoal[2] atualmente se mostra capaz de arcar com a (muito criticada) complexidade computacional da PG, tornando-a menos impeditiva. Isso levou a um aumento muito substancial nas publicações acadêmicas e aplicações da PG, principalmente a partir de 2002.

1.2 - Objetivos

Com base nestes fatores citados, esta pesquisa foi conduzida com o objetivo de aplicar a PG ao problema de identificação de acidentes em centrais nucleares, a partir de séries temporais de variáveis físicas obtidas com dados simulados para a usina de Angra 2. Nesta aplicação, diversos parâmetros e a combinação de variáveis ótimas para a PG na tarefa de classificação serão avaliados, de maneira a obter soluções eficientes computacionalmente.

1.3 - Organização

No **Capítulo 1** foi feita uma introdução sobre o problema de identificação de acidentes em centrais nucleares, a motivação do trabalho e foi introduzida qual será a abordagem aplicada ao problema: reconhecimento de padrões através da programação genética.

[2] De fato, um computador pessoal foi utilizado nesta pesquisa, fornecendo resultados em cerca de 6 horas para uma execução completa do algoritmo. Mais detalhes na Seção 5.3.1.

No **Capítulo 2** é exibida uma explanação mais detalhada sobre a área de reconhecimento de padrões. São desenvolvidas as etapas de um sistema de RP típico e apresentados alguns métodos de classificação.

O **Capítulo 3** tratará da técnica empregada para o reconhecimento de padrões neste trabalho, a Programação Genética. Um breve histórico da técnica, seu funcionamento básico geral, técnicas mais avançadas e atuais e trabalhos relevantes disponíveis na literatura encontram-se lá.

O **Capítulo 4** aprofunda o estudo bibliográfico das contribuições já realizadas na área de sistemas de diagnóstico para a identificação de transientes em usinas nucleares.

O **Capítulo 5** descreve mais minuciosamente o problema e parametriza e descreve os resultados obtidos com a aplicação da programação genética ao conjunto de acidentes escolhidos para teste.

O **Capítulo 6** apresenta conclusões e as sugestões para futuros trabalhos.

CAPÍTULO 2

PANORAMA DE RECONHECIMENTO DE PADRÕES

O reconhecimento de padrões é uma área científica que tem sido definida de várias maneiras na literatura [2,6]. Em particular, como:

- O ato de tomar dados primários e realizar uma ação baseada na categoria do(s) padrão(ões);
- A disciplina científica cujo objetivo é a classificação de objetos em categorias.

De maneira geral, o objetivo de qualquer sistema de RP é, efetivamente, agrupar objetos em categorias. Os objetos podem ser sinais ondulatórios, imagens ou outro tipo de medidas. O agrupamento de objetos é uma tarefa importante, que pode ser realizada utilizando ordenamentos estatísticos, inteligência artificial e muitos outros métodos, alguns dos quais serão apresentados neste capítulo. Devido ao aumento da automação na produção industrial, recentemente o RP tem recebido muita atenção e demanda industrial e acadêmica. Neste estudo, o RP foi utilizado na tarefa de classificação, embora seja frequentemente aplicado à tarefas de regressão, agrupamento (clusterização), descrição, entre outras.

Um sistema de RP típico está ilustrado na Figura 2.1. Um ou mais sensores ou transdutores (que podem ser termômetros, câmeras, microfones, etc.) fornecem medidas que serão posteriormente utilizadas para a extração de atributos. Um atributo ideal é aquele cujos valores são bem similares para objetos na mesma categoria e diferentes para objetos em categorias distintas. Caso o número de atributos extraídos seja muito grande, um método de redução dimensional pode ser aplicado para remover aqueles redundantes ou irrelevantes. Após isso, os atributos são transferidos a um classificador, que tem como função utilizá-los para estabelecer limites e "rótulos" (labels) a cada padrão, de acordo com intervalos limitantes no espaço de atributos. Finalmente, a eficiência do classificador treinado é avaliada utilizando dados de teste.

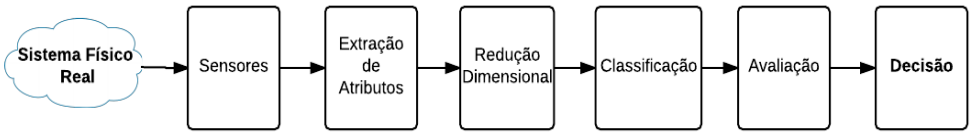


Figura 2.1 – Versão básica de um sistema de reconhecimento de padrões.

2.1 – Coleta de Dados

Em geral, há três tipos básicos de dados: categóricos, ordinais e numéricos. Dados categóricos tratam-se de informações não-numéricas sem ordenamento de classes, como a descrição da profissão de uma pessoa (engenheiro, cientista, economista, etc.). Quando há ordenamento de classes em dados categóricos, estes são chamados de dados ordinais, por exemplo: pequeno, médio e grande. Dados numéricos, como o nome indica, contém números reais, sejam informações sobre pressão sanguínea, peso, variáveis físicas, etc. Como o objetivo do reconhecimento de padrões envolve realizar tratamento computacional nos dados, caso sejam categóricos ou ordinais, eles devem ser "transformados" em dados numéricos.

Como já mencionado, a coleta de dados é usualmente realizada utilizando algum tipo de transdutor ou sensor. Esses dados contém o sinal original, com algum grau de ruído.

2.2 – Extração de Características

Qualquer objeto classificável possui particularidades que o diferenciam de outros objetos. Essas peculiaridades discriminatórias, em problemas de reconhecimento de padrões, são chamadas de atributos. Em certos problemas de RP, os dados crus ("raw data") coletados já representam os atributos em si, e não há a necessidade de realizar o processo de extração explicitamente. Atributos podem ser divididos em três categoriais principais: físicos, estruturais e matemáticos. Atributos físicos incluem: a cor de um objeto, seu odor, o material que o compõe, etc. Atributos estruturais representam as propriedades espaciais de um objeto, como altura, peso, comprimento, etc. Já as características matemáticas – as mais importantes do ponto de vista de RP – representam, como o nome anuncia, as suas propriedades matemáticas: média, variância, componentes principais, autovetores, covariância, etc.

A escolha de atributos é um passo crítico no projeto do sistema de RP. Idealmente, um atributo é simples de extrair, invariante à transformações irrelevantes, insensível a ruído e útil para a discriminação entre dois objetos de categorias diferentes. Já que as características extraídas serão posteriormente utilizadas pelo classificador para a tarefa de classificação, a característica ideal também é aquela que facilita o trabalho do classificador, fornecendo valores característicos similares para exemplares da mesma classe e valores consideravelmente diferentes para integrantes de classes distintas. Algumas ilustrações espaciais dessas qualidades desejáveis aos atributos estão na Figura 2.2.

Figura 2.2 – Exemplos de atributos ideais [5].

Vale ressaltar, ainda, que em algumas literaturas o processo de extração de características é considerado parte integrante da geração de características, mas em outros trabalhos (inclusive neste), a geração de características é um processo diferenciado que envolve a transformação dimensional das características extraídas inicialmente.

2.3 – Redução Dimensional

No contexto de RP, redução dimensional é o ato de reduzir o número de atributos sendo considerados em um determinado problema. Esse conceito é derivado do domínio da estatística, onde grandes quantidades de dados levam, muitas vezes, a um fenômeno conhecido na literatura como "maldição da dimensionalidade" [7-9], ilustrado na Figura 2.3, em que há um número ótimo de atributos, além do qual a performance do classificador piora. Algoritmos de machine learning e RP estão susceptíveis a esse problema e métodos de redução dimensional são frequentemente aplicados para abordá-lo [10].

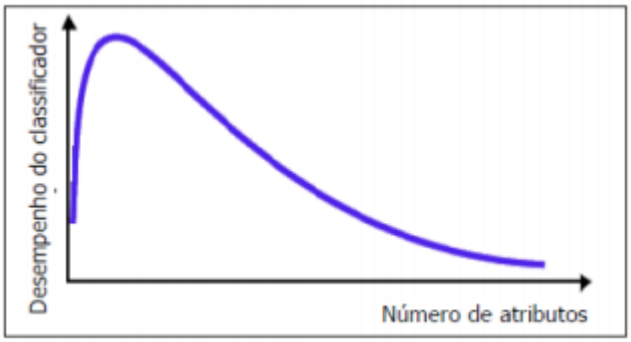


Figura 2.3 – Maldição da dimensionalidade [11].

Espaços de busca contínuos e discretos de alta dimensão [12] – como é o caso do problema tratado neste trabalho – trazem muitos desafios à análise de dados. Um dos principais é que nem todos os atributos são realmente importantes para o tratamento do problema em questão. Alguns dos atributos podem ser correlacionados, levando à redundâncias, ou irrelevantes, podendo trazer ruído à análise e aos atributos relevantes. Esses atributos irrelevantes ou correlacionados contribuem negativamente para a performance e precisão do

classificador. Além disso, o tempo de treinamento (que já é considerável para conjuntos de dados de alta dimensão) pode aumentar linearmente (ou, dependendo do algoritmo, exponencialmente) com o aumento no número de atributos.

Matematicamente, o problema de redução dimensional pode ser descrito da seguinte forma: dado o conjunto de dados $x = (x_1, x_2, \dots, x_m)^T$, encontrar uma representação de menor dimensão desse conjunto de dados, $y = (y_1, y_2, \dots, y_n)^T$, $n < m$, tal que essa última representação consiga, de acordo com algum critério, capturar as características importantes do conjunto original.

A redução dimensional pode ser subdividida em duas categorias principais. Na primeira, chamada de seleção de atributos, um subconjunto do conjunto de dados existente é selecionado, levando em consideração algumas restrições, com o objetivo de maximizar a informação discriminatória entre as categorias. Já na segunda categoria (geração de atributos), os atributos são transformados em um conjunto reduzido, de maneira a capturar o máximo de informação contido nos atributos iniciais.

2.4 – Seleção de Atributos

O objetivo da seleção de atributos, como já mencionado, é obter um subconjunto de atributos mais relevante, do ponto de vista de informação discriminatória, entre as categorias (classes). A função principal desse procedimento é identificar a importância dos diferentes atributos em um dado problema. Em termos de classificação, características importantes (ricas) são aquelas que possuem substancial variância entre classes (inter-classes) e pouca variância dentro de cada mesma classe (intra-classe). A escolha de características importantes simplifica o trabalho do classificador [13].

Existem duas abordagens principais para a seleção de atributos: a abordagem wrapper (“empacotador”), assim chamada porque o processo de escolha do subconjunto de atributos está “empacotado” com o algoritmo de aprendizado sendo utilizado [14]; e a abordagem de filtro, onde um atributo é avaliado utilizando as características intrínsecas dos dados, como correlação, variância, etc. A abordagem de filtro é mais simples e menos “cara” computacionalmente que a wrapper, além de fornecer soluções mais gerais, que apresentam boa performance em uma gama mais vasta de classificadores.

2.5 – Geração de Atributos

A geração de atributos é um termo abrangente e poderoso no domínio do RP, que em algumas literaturas abarca também o conceito de seleção de características. Consiste em realizar combinações dos atributos disponíveis de maneira a obter um subconjunto reduzido de atributos tal que as classes subjacentes (implícitas) sejam mais separáveis do que eram no conjunto original[3]. Vale salientar aqui a diferença entre este processo e o descrito na Seção 2.4, no qual são feitas meras seleções dos atributos mais relevantes.

Em todas as aplicações de RP o objetivo da geração de atributos é o mesmo: reduzir a quantidade de recursos necessários para alcançar o objetivo, buscando diminuir ou evitar a "maldição da dimensionalidade" (Seção 2.3). Assim, não é apenas uma questão de diminuir o custo computacional, mas também de aprimorar a performance do classificador, algo que claramente não pode ser negligenciado.

As técnicas de geração de características disponíveis na literatura são bem diversas. Elas podem ser classificadas em lineares ou não lineares, dependendo da função de classificação que é utilizada; supervisionadas ou não-supervisionadas, levando em consideração a disponibilidade de informações sobre as classes; etc. Dentre as técnicas lineares, a análise de componente principal (PCA) é uma das mais simples e largamente utilizadas, fornecendo resultados satisfatórios para uma vasta gama de aplicações em que modelagens lineares são aplicáveis. A PG, como já foi dito, está se tornando uma das principais técnicas de geração de características para os casos não lineares e, sendo a técnica adotada nesta dissertação, terá o capítulo 3 dedicado a sua descrição mais detalhada.

2.6 – Métodos de Classificação

Classificação é o processo de atribuir um rótulo de classe a um grupo de objetos descrito por um conjunto de atributos. Dependendo da metodologia, os classificadores podem ser divididos em três categorias principais [15]:

- Classificadores baseados em similaridade: os padrões similares entre si são atribuídos à mesma classe. Ex.: casamento (template matching), regra dos vizinhos mais próximos (nearest neighbour rule), etc;

[3] Cabe esclarecer que, no contexto de processamento de sinais e imagens, a geração de atributos é frequentemente considerada um processo um pouco diferente, e descreve o ato de extrair atributos do sinal ou imagem originais, mas utilizando representações reduzidas das entradas (inputs) que, espera-se, contenham informação relevante suficiente para realizar a tarefa buscada.

- Classificadores baseados em probabilidade: Ex.: regra de decisão bayesiana, classificador ML (maximum likelihood), classificador de Parzen, etc;
- Classificadores baseados em intervalos de decisão (decision boundaries): Ex.: análise discriminante de Fisher, árvores de decisão, SVMs (support vector machines), etc.

Os classificadores buscam discriminar classes encontrando diferenças e similaridades entre padrões. Classificadores que necessitam de treinamento (ou aprendizado) tem seus parâmetros ajustados (através das características das diferentes classes) durante a fase de treino, para facilitar a discriminação entre as classes. Duas das principais características consideradas para ajustar os parâmetros dos classificadores são a variância entre classes e dentro da mesma classe (intra-class e inter-class variance).

Os inputs que alimentam o classificador influenciam bastante o seu desempenho. Em problemas de classificação, ou o processo de geração de atributos maximiza a informação discriminatória apresentada ao classificador, ou o atributo é diretamente fornecido ao classificador para que este busque os limites (decision boundaries) ótimos entre as classes. Pelo que já foi exposto, fica claro que o processo de geração de atributos facilita o trabalho do classificador. Como regra geral, tem-se que, quanto mais informação discriminatória está presente nos atributos, menos eficiente precisa ser o classificador, e vice-versa.

CAPÍTULO 3

PROGRAMAÇÃO GENÉTICA

No capítulo anterior, as partes básicas de um sistema de reconhecimento de padrões foram discutidas. Uma das suas componentes-chave é a geração de atributos, utilizada para a redução dimensional. Neste trabalho, a programação genética (PG), uma técnica de aprendizado de máquina, foi aplicada para este fim. A PG oferece a possibilidade de realizar a seleção de atributos e a transformação dos mesmos (geração de atributos) concomitantemente, o que a torna, nesse aspecto, superior às demais metodologias. Este capítulo apresenta um panorama geral do algoritmo de programação genética.

3.1 – Computação Evolucionária

Computação evolucionária (CE) é um termo generalista atribuído a uma vasta quantidade de técnicas baseadas, em maior ou menor grau, no modelo Darwiniano de evolução natural e sobrevivência do mais apto. O termo tem suas raízes filosóficas em trabalhos de Alan Turing, e se tornou real com o advento do algoritmo genético (AG), das estratégias evolucionárias e programação evolucionária. Todos os métodos de computação evolucionária funcionam, basicamente, através da atribuição de uma "meta" no início da evolução, a qual é aplicada para avaliar e comparar a "qualidade" de potenciais soluções para o problema em questão durante a evolução. Em situações ideais, as técnicas de CE levam a resultados ótimos ou próximos do ótimo após um certo número de iterações (chamadas, usualmente, de gerações).

O elo em comum entre todos os métodos de computação evolucionária é a idéia da evolução de potenciais soluções para problemas, que podem ser tão simples quanto vencer um jogo de entretenimento ou tão complexo quanto resolver um problema de engenharia. A CE pode ser aplicada a uma diversa e extensa gama de domínios do conhecimento.

3.2 – Programação Genética

A PG pertence a uma classe de metodologias de computação evolucionária na qual programas de computador (no sentido de instruções capazes de serem interpretadas pelo computador [16]) são evoluídos num paradigma inspirado no modelo darwiniano de evolução natural. Em 1992, John Koza padronizou, introduziu e lançou as bases para a técnica que, segundo o próprio autor, começara a ser testada na década de 80.

A PG é uma técnica de CE para resolver problemas automaticamente sem requerer que o usuário especifique *a priori* a forma ou a estrutura da solução [17], sendo um método sistemático e independente do domínio [18] para obter soluções de problemas automaticamente, fornecendo instruções de alto nível [19] ao computador a respeito do que precisa ser feito.

3.2.1 – Breve histórico da técnica

Em seus ensaios em 1948 [20] e 1950 [21], Intelligent Machinery e Computing Machinery and Intelligence, Alan Turing corretamente percebeu que, no futuro, os computadores poderiam, um dia, resolver problemas de maneira inteligente através de um processo evolucionário no qual o programa de computador (material genético) estivesse sujeito a várias modificações progressivas (mutações) guiadas por um processo análogo ao da seleção natural (pressão de busca: a fitness).

Em 1975 [22], John Holland formalizou o conceito de recombinação de genes, introduziu o operador de crossover entre strings binárias e desenvolveu a metodologia do algoritmo genético.

KOZA [4], tomando como base uma série de trabalhos feitos pós-1975 [23-32] sobre as possibilidades de evoluir algo que não fossem strings binárias, trouxe à tona e sistematizou a técnica de programação genética. Desde então, a PG tem sido alvo de muitos estudos, publicações e críticas que são reportados regularmente em publicações e conferências [33-37].

3.2.2 – Funcionamento básico geral

A PG, então, trabalha evoluindo programas escritos em uma linguagem de alto nível (como C, Pascal ou Lisp). Geração após geração a PG estocasticamente transforma uma população de programas buscando encontrar aquele que melhor resolve o(s) problema(s) em questão (Figura 3.1).

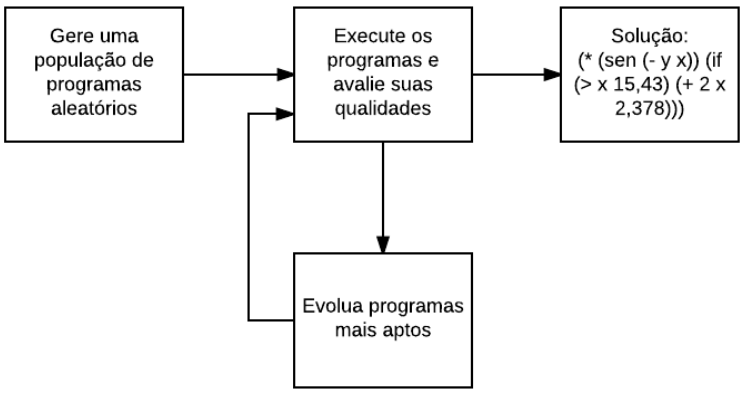


Figura 3.1 – Rotina básica de um sistema PG.

Assim, a analogia com o algoritmo genético é imediata, sendo a principal diferença entre eles a representação dos genótipos [38]. De fato, ambos utilizam os mesmos operadores genéticos primários (cruzamento e mutação) e possuem o conceito de métrica de fitness, mas divergem no fato de que o AG em geral trabalha com cromossomos formados por strings binárias de tamanho fixo, enquanto que, na PG, os cromossomos são as instruções do programa e variam em tamanho e forma.

3.2.2.1 – Árvores de sintaxe

Os programas a serem evoluídos são expressos como árvores de sintaxe [39], que consistem de funções e terminais.

Ex.: $\max(x + x, x + 3 * y)$

neste exemplo, x, y e 3 são as "folhas" (extremidades) da árvore e chamadas de terminais, já max, + e * são os nós internos, chamados de funções ou operadores[4].

O set (conjunto) de funções, F, e o set de terminais, T, constituem o set de "primitivas" para essa população, variando de aplicação para aplicação da técnica.

É comum representar os programas em PG com a notação de prefixo típica do Lisp (fato que, aliás, torna esta linguagem e suas variantes, como clojure, especialmente adequadas para escrever códigos para a PG), pois assim é fácil estabelecer a correspondência entre o programa

[4] Embora esta última terminologia seja evitada para não confundir com os operadores genéticos, que serão explicados posteriormente.

e sua representação em árvore. Assim, $\max(x + x, x + 3 * y)$ será, em notação de prefixo, (max (+ x x) (+ x (* 3 y))), e sua representação em árvore está dada na Figura 3.2:

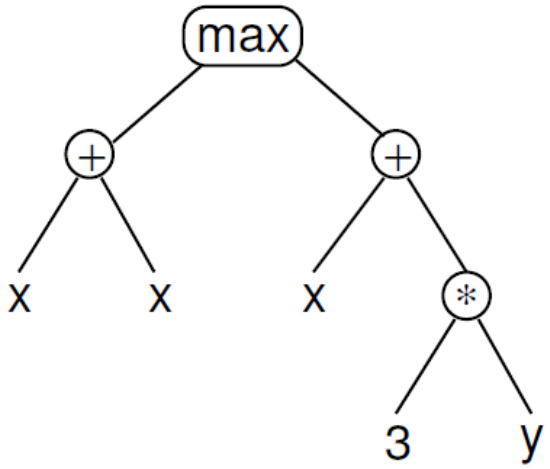


Figura 3.2 – Exemplo de um indivíduo típico na PG [17].

O primeiro nó da árvore é chamado raiz, e os nós subsequentes são chamados subárvores, galhos ou folhas, dependendo da sua posição relativa e do número de elementos. O número máximo vertical de nós (a contar pela raiz, considerada nó zero) é chamado "depth" (profundidade), D, da árvore. Já a quantidade de nós da árvore é chamada de "size" (tamanho).

Embora não seja obrigatório, é comum especificar de antemão o "arity"[5], que significa o número de argumentos que as funções podem conter (os terminais têm arity 0, por definição). Assim, os parênteses se tornam redundantes, facilitando a análise do programa.

3.2.2.2 – Inicialização da população

No que diz respeito à geração aleatória da população inicial (Figura 3.1), existem três métodos principais (embora mais métodos estejam se tornando conhecidos na literatura, conforme a PG tem sido explorada). São eles:

- Método de inicialização "full" (Figura 3.3): de 0 até D-1 (onde D é a profundidade da árvore, conforme Seção 3.2.2.1), monta-se cada indivíduo da população escolhendo-se aleatoriamente funções do set de funções F. Quando atinge-se o depth, escolhe-se um terminal (Figura 3.3). A

[5] De acordo com as pesquisas deste autor, não existe um correspondente direto na língua portuguesa ao termo "arity". Alguns autores optam por traduzi-lo como "aridade", o que é um neologismo, embora a maioria mantenha a terminologia em inglês ou prefiram denotar como "grau".

variabilidade de formas e tamanhos de árvore neste método é baixa, embora ele seja considerado adequado em certas aplicações.

- Método de inicialização "grow" (Figura 3.4): de 0 até D-1 escolhe-se aleatoriamente funções ou terminais. Quando atinge-se o depth, escolhe-se um terminal.
- Ramped half-and-half: distribui-se igualmente uma quantidade de indivíduos para cada profundidade inicial de árvore, indo de 1 até D. Metade dos indivíduos em cada profundidade é construído segundo o método "full", e a outra metade conforme o método "grow". Este método contorna as desvantagens do método "grow", de possivelmente montar árvores muito pequenas, e do método "full" de fornecer pouca diversidade. Devido a isso, é o método de inicialização da população mais amplamente utilizado.

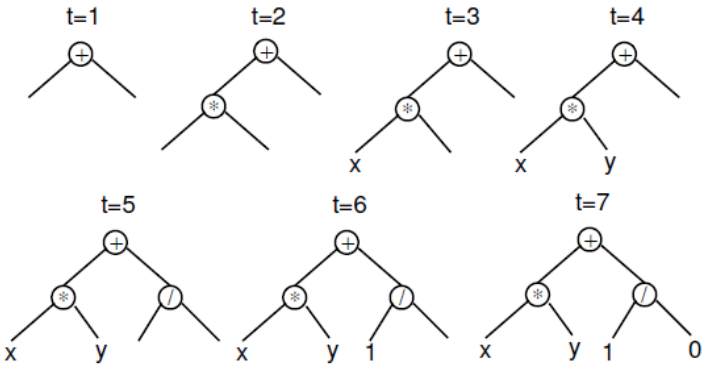


Figura 3.3 – Método de inicialização "full" [17].

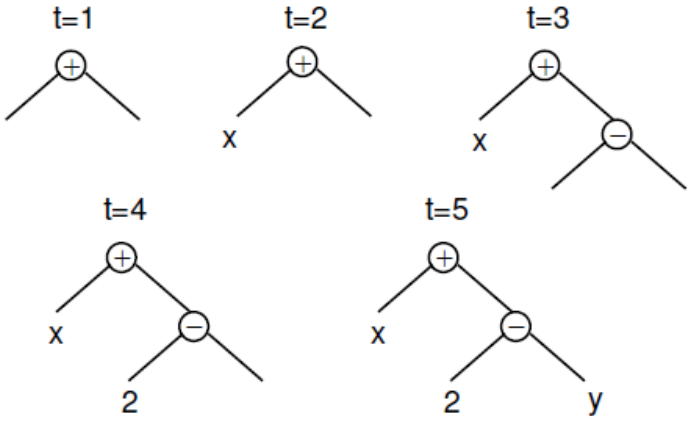


Figura 3.4 – Método de inicialização "grow" [17].

3.2.2.3 – Seleção

Os métodos de seleção na PG são bastante similares aos métodos mais famosos de seleção dos algoritmos genéticos.

O mecanismo de seleção mais utilizado em PG é a seleção por torneio (“tournament selection”), na qual toma-se aleatoriamente uma porção de indivíduos e aquele com fitness mais alta do grupo é selecionado. Como não se leva em consideração, na seleção do indivíduo mais apto do grupo, a diferença relativa entre as fitness dos indivíduos, mas tão somente aquele que possui melhor fitness, este método tende a manter a pressão de busca constante.

Também é comum a seleção proporcional à fitness, dada pela equação (3.1):

$$\frac{f\left(s_i(t)\right)}{\sum_{j=1}^{M} f\left(s_j(t)\right)} \tag{3.1}$$

onde s é um indivíduo da população M e $f\left(s_i(t)\right)$ é sua fitness na geração t. Este método atribui probabilidades de seleção maiores aos indivíduos mais aptos, assim como no AG.

3.2.2.4 – Cruzamento

Existe uma diversidade enorme de operadores de cruzamento na PG [40]. Os principais e mais utilizados são os operadores de trocas de subárvores (subtree-swapping crossovers, como o da Figura 3.5, onde é gerado apenas um programa-filho) que, como o nome sugere, produzem descendentes trocando subárvores entre os programas-pais. Dentro desta classe estão:

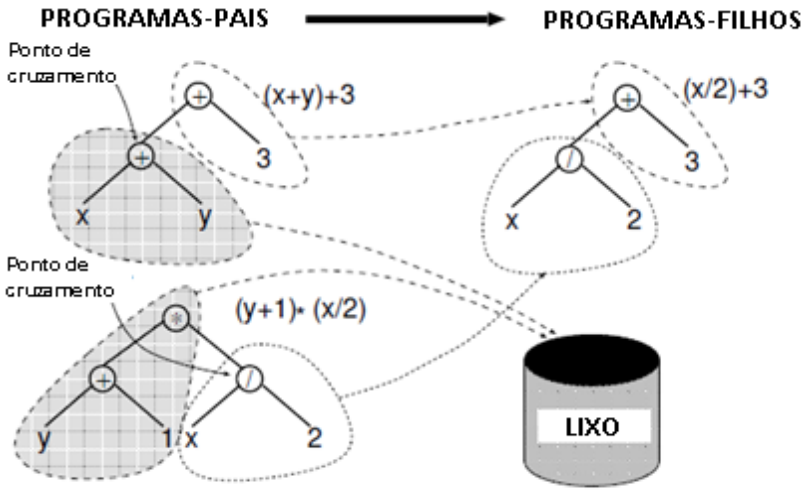


Figura 3.5 – Mecanismo de um cruzamento de subárvores genérico [17].

- standard crossover;
- one-point crossover;
- size-fair crossover;

- strongly-typed crossover;

É comum especificar que o ponto de cross seja uma função em 90% das vezes. Faz-se isso para evitar o cruzamento em nós de terminais, trocando pouco material genético, haja vista que há mais terminais (folhas), numa árvore típica, do que ramos.

3.2.2.5 – Mutação

Embora inicialmente tenha sido sentenciada como sem muita relevância para a PG [4], atualmente é considerada importante para manter a exploração do espaço de busca, com aplicações em que os percentuais de mutação vão de 1% à 50% da população.

Tal como o cruzamento, a mutação também possui diversas variantes na PG, embora a mais comum seja a subtree mutation (Figura 3.6), na qual um nó de um indivíduo da população é escolhido aleatoriamente e sua subárvore correspondente é substituída por uma subárvore formada também aleatoriamente, respeitando os limites de tamanho de árvore.

Outra opção razoavelmente comum é a point mutation, bastante similar ao "bit flip" do AG, na qual escolhe-se aleatoriamente um nó de um indivíduo e substitui-se o mesmo por uma primitiva aleatória de mesma aridade.

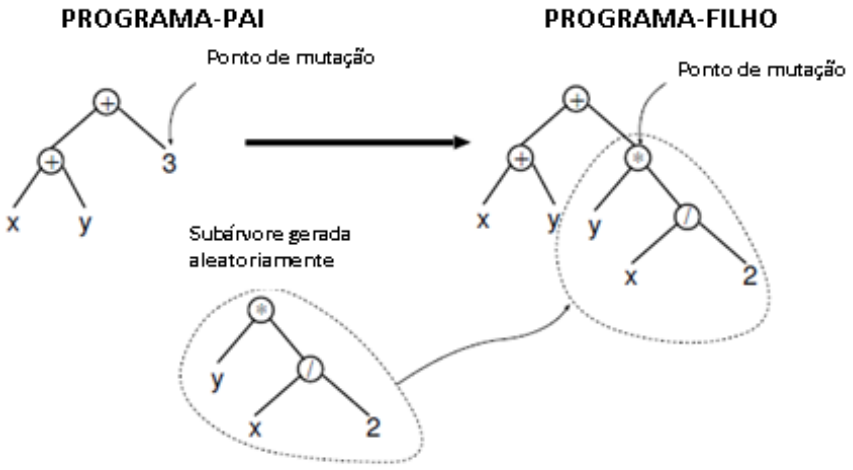


Figura 3.6 – Esquema do mecanismo de uma mutação de subárvore [17].

3.2.2.6 – Fitness

Existem duas propriedades que o set de primitivas deve satisfazer em qualquer aplicação da PG:

- Fechamento (closure): determina que qualquer subárvore deve poder ser utilizada como argumento de quaisquer das funções do set de funções. Assim, é necessário, por exemplo, utilizar versões "protegidas" de funções cujo domínio não esteja definido em algum ponto. Como exemplo imediato tem-se a função divisão "÷", que não está definida quando o denominador é igual a zero. Neste caso, utiliza-se a versão protegida dessa função, usualmente denotada por "%", na qual define-se a restrição de que, quando o terminal referente ao denominador for zero, a função retornará o valor 1 (ou 0, ou qualquer outro tratamento[6] definido pelo programador).
- Suficiência (sufficiency): condição que afirma que o set de primitivas deve ser capaz de, combinando-se, expressar uma solução para o problema em questão. Por exemplo: o set de primitivas {AND, OR, NOT, x_1, x_2, ..., x_n} é sempre suficiente para problemas de indução booleana, já que é capaz de produzir todas as funções booleanas das variáveis (terminais) x_1, x_2, ..., x_n. Conjuntos insuficientes de primitivas, na melhor das hipóteses, apenas aproximarão a solução obtida pela PG, embora, dependendo do grau de precisão requerido, essa aproximação seja boa o suficiente. Em problemas reais, muitas vezes não se sabe qual é o conjunto suficiente, e adicionar e fazer testes com algumas primitivas, embora torne o algoritmo mais lento, são alternativas, muito embora possam enviesar a PG de maneira inesperada.

Dado que o set de primitivas foi escolhido de maneira a satisfazer a essas propriedades, cada programa da população expressa uma solução para o problema. O potencial dessa solução, isto é, o quão boa ela é para resolver o problema em questão, é dado ao se comparar o resultado da avaliação do programa com a medida de fitness, definida e ajustada previamente.

A medida de fitness é o parâmetro mais complexo e menos axiomático a ser definido num sistema PG. Ele pode ser, entre outros:

- O *erro* entre o output, obtido através dos inputs, e o valor conhecido (chamado de target), como em problemas de regressão simbólica;
- O *tempo* que ele demora para levar o sistema a um estado determinado, utilizado em problemas de scheduling;
- A *precisão* do programa em reconhecer padrões ou identificar objetos, parâmetro considerado em treinamentos de redes neurais ou data mining.

[6] Embora a escolha da constante 1 como resultado deste caso particular seja mais razoável, pois fornece à PG uma maneira simples de obter este valor. Combinado com um mecanismo similar de obter a constante 0 através de (- x x), tem-se a garantia de que a PG poderá obter facilmente essas duas constantes importantes.

Em problemas como regressão simbólica, séries temporais e data mining, a melhor solução é aquela que mais se aproxima de todos os pontos, contribuindo cada um deles de forma incremental para a fitness. Nesses casos, cada ponto é chamado de "fitness case".

3.2.2.7 – Critério de parada

O critério de parada em PG pode consistir de um número máximo de gerações a ser atingido ou um indicativo de sucesso específico para o problema, como a obtenção de um erro abaixo de um valor limite para o somatório dos fitness cases em problemas de regressão simbólica, por exemplo.

Tipicamente, o indivíduo com melhor fitness é então extraído do sistema e designado como o resultado da rodagem do algoritmo, embora mais indivíduos possam ser registrados ao longo das gerações e consultados, conforme for adequado à classe do problema e a necessidade do usuário.

3.2.2.8 – Parâmetros

Baseado no que foi até aqui exposto, conclui-se que os parâmetros a serem ajustados na execução de um programa de PG são vários, principalmente:

1. A profundidade máxima das árvores iniciais (D_i);
2. A profundidade máxima das árvores durante o funcionamento da PG (D);
3. O tamanho da população (M);
4. O número máximo de gerações (G);
5. O conjunto de terminais (T). Ex.: 1, 2, rand, *e*, pi, etc;
6. O set de funções (F): +, -, sen, cos, log, etc;
7. O método de inicialização;
8. A medida de fitness (*f*);
9. O método de seleção;
10. Operadores genéticos: tipos e probabilidades;
11. O critério de parada.

Fazendo referência ao fluxograma da Figura 3.1, os parâmetros de 1 à 7 referem-se à etapa de geração aleatória de programas, o parâmetro 8 ao bloco de avaliação das qualidades dos programas, os blocos 9 e 10 à evolução de programas mais aptos e, por fim o bloco 11 ao fim da recursividade do algoritmo e fornecimento da solução.

3.3 – Técnicas Avançadas em PG

A PG, desde a sua padronização em 1992, tem evoluído rapidamente, à despeito de uma certa falta de solidez teórica (Seção 3.4), e muitas técnicas e conceitos mais avançados foram elaborados sobre as bases do exposto na Seção 3.2.

3.3.1 – Evolução de estruturas modulares e hierárquicas

Até aqui, a PG tem sido descrita como capaz de evoluir expressões mais simples que, embora adequadas para a resolução de uma vasta gama de problemas, raramente exibirão características modulares em larga escala, típicas de objetos e indivíduos mais complexos, como uma árvore, um animal ou um avião.

Na tentativa de emular tais estruturas em PG, muito esforço acadêmico tem sido realizado desde 1992, sendo as funções definidas automaticamente (automatically defined functions - ADFs) e ramificações produtoras de resultado (result-producing branches - RPBs), que possibilitam as operações de alterações de arquitetura (architecture altering operations), aquelas que tiveram maior êxito nessa empreitada.

As ADFs são, conceitualmente, muito similares a blocos de código típicos de um programa de computador que são chamados repetidamente para realizar alguma tarefa, como normalizar um conjunto de valores ou traçar um gráfico, ao passo que as RPBs são análogas ao programa principal, que "chama" essas funções.

A intenção dessas subárvores que são, em si, sub-rotinas é, então, possibilitar que a PG evolua estruturas modulares e hierárquicas, como exemplificado na Figura 3.7, onde tem-se duas ADFs e uma RPB. Neste exemplo simples, se cada ADF simboliza a subrotina de levar um número ao quadrado, a RPB representa a operação ADF(ADF(x)), dando como resultado x^4. Dada uma população de estruturas como essas, a PG pode recombiná-las e mutá-las, de acordo com os operadores já descritos e algumas restrições adicionais necessárias.

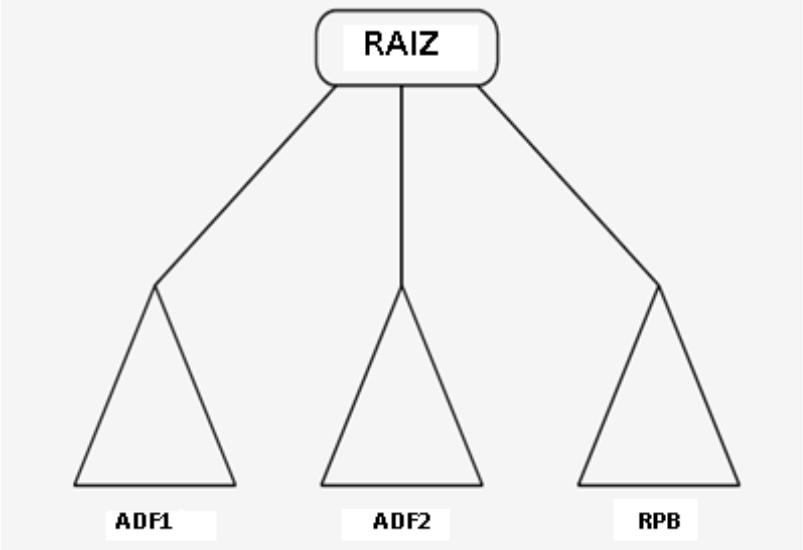


Figura 3.7 – Esquema do mecanismo de uma mutação de subárvore [17].

Indo além desse conceito estão as operações de alteração de arquitetura, onde a PG pode, inclusive, alterar, excluir e incluir ADFs durante a execução do algoritmo. Esta idéia de arquitetura modificável levou ao "solucionador geral de problemas por programação genética" (genetic programming problem solver - GPPS) [41], o qual, em teoria, seria capaz de resolver um conjunto extremamente vasto de problemas de diferentes áreas apenas com a definição prévia pelo usuário da medida de fitness. Entretanto, a ideia apresentou mais relevância conceitual do que prática, devido à necessidade de construir muito especificamente e cautelosamente a fitness para tornar viável o resultado, além de requerer um esforço computacional muito grande.

3.3.2 – Estruturas com restrições

Admitindo que a propriedade closure (Seção 3.2.2.6) está satisfeita, uma condição implícita é que todas as combinações de estruturas tem a mesma probabilidade de serem "úteis" para um dado indivíduo. Mesmo assim, quando tem-se de antemão alguma suspeita quanto à forma das melhores soluções, ferramentas de tipificação e gramática podem ser aplicadas para enviesar ou restringir a busca do algoritmo, com o objetivo primário de aumentar a chance de encontrar o programa ideal.

Existem três abordagens principais no tocante à restrição sintática ou semântica na PG: reforço estrutural simples (Seção 3.3.2.1), PG fortemente tipificada (Seção 3.3.2.2) e restrições baseadas em gramática (Seção 3.3.2.3).

Ainda quanto à estruturas com restrições, é importante salientar que muitas críticas tem sido feitas à técnica, especialmente endereçadas ao fato de requerer maior processamento e esforço computacional (nesse aspecto, restrições sintáticas, como sistemas baseados em gramáticas, trazem menor peso computacional adicional que restrições semânticas, como sistemas tipificados). Além disso, se mal aplicada, pode direcionar o sistema para partes do espaço de busca onde não está a solução ótima. Contudo, sistemas com restrições em PG têm sido corretamente aplicados com sucesso na literatura, limitando o espaço de busca de maneira valiosa [42] e melhorando a performance do algoritmo em vários problemas interessantes [43].

3.3.2.1 – Reforço estrutural simples

Esta abordagem envolve a modificação do sistema PG para forçar todos os indivíduos a terem uma determinada estrutura. Se, por exemplo, sabe-se que a solução do problema deve ter comportamento periódico, pode-se considerar restringir a busca à soluções da forma $a * sen(b * t)$, permitindo que a e b sejam argumentos evoluídos livremente, mas mantendo o restante da estrutura fixa.

A PG permite considerável liberdade na aplicação de técnicas como essa: pode-se forçar os indivíduos a terem a estrutura de interesse e restringir os operadores genéticos para que não alterem nenhuma das regiões fixas da árvore; pode-se evoluir os componentes separadamente ou ainda trabalhar com duas (ou mais) populações diferentes, uma que evolui candidatos para a e a outra para b.

3.3.2.2 – PG fortemente tipificada

Nesta opção, cada terminal possui um "tipo", e cada função possui tipos para cada um dos seus argumentos e um tipo para o valor que ela retorna. Todas as árvores iniciais e os operadores genéticos são implementados de maneira a não violar as restrições desses tipos.

Como exemplo simples, considerando a função if, a qual recebe três argumentos (um teste booleano, o valor a retornar se o teste for verdadeiro e o valor a retornar se o teste for falso). Assumindo que os valores a retornar são numéricos, a função if retornará também um valor numérico. Neste caso, fica claro que, se o primeiro argumento (o teste) for escolhido como ponto de cross num dos pais, a subárvore extraída do segundo parente deverá retornar um valor booleano. Inversamente, se o segundo ou terceiro argumentos forem o ponto de cross, a subárvore inserida deverá ser numérica.

3.3.2.3 – Restrições baseadas em gramáticas

Outra técnica que está sendo bastante explorada na PG é a ideia de expressar restrições como gramáticas. Este é um tópico bastante abrangente que vai desde maneiras mais "mecânicas" de aplicar restrições como as discutidas na Seção 3.3.2.1 até métodos que envolvem conceitos de linguística computacional, como as *tree adjoining grammars* (TAGs), as quais possibilitam novos tipos de modificação estrutural, e a *grammatical evolution* (GE), que considera os indivíduos como sequências de números inteiros a serem evoluídos no contexto da gramática previamente definida.

Por exemplo, para restringir soluções à forma $a * sen(b * t)$ poderia ser utilizada a gramática sequencial simples:

arvore = E x sin(E x t);

E = var | (E op E);

op = + | - | x | ÷;

var = x | y | z.

de maneira que um indivíduo possível seria a expressão $y * sen\big((x + z) * t\big)$, representado pela árvore da Figura 3.8:

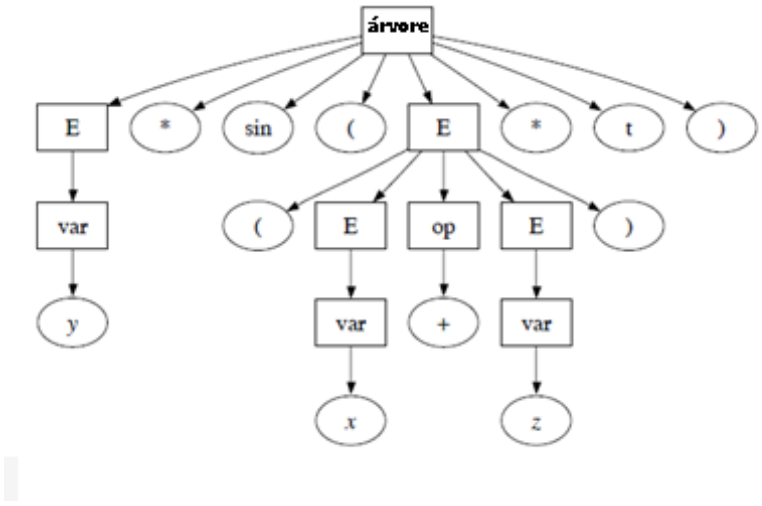


Figura 3.8 – Exemplo de indivíduo com restrição gramatical.

3.3.3 – Programação genética linear e gráfica

Embora a aplicação original e mais difundida da PG tenha sido a evolução de programas expressos como árvores interpretáveis, existem outros tipos de PG nos quais os programas são representados de maneira diferente, seja ela linear ou gráfica (paralela).

3.3.3.1 – Programação genética linear

Exceto por Lisp e algumas outras, a maioria das linguagens de programação funciona através da representação de comandos de maneira linear que são executados em passos consecutivos (salvo o fato sabido de que estruturas de controle, loops e direcionadores podem mudar a ordem de interpretação). Além disso, computadores não executam programas em estruturas hierárquicas, como árvores, naturalmente, de modo que interpretadores e compiladores normalmente fazem parte de sistemas de PG baseados nesse tipo de estruturas, exigindo mais processamento.

Levando esses dois fatores em consideração, a busca por mais clareza e velocidade levou ao desenvolvimento da programação genética linear, na qual os programas são sequências lineares de instruções, de número fixo ou variável (Figura 3.9).

Figura 3.9 – Representação típica de programas na PG linear.

A semântica da PG linear é consideravelmente diferente. Não há mais raízes, subárvores e as demais estruturas inspiradas em árvores. Nem mesmo há funções e terminais, mas sim instruções lidas de registros de memória, calculadas e armazenadas em outros registros, de maneira muito similar à códigos de programação em linguagens de alto nível.

Quanto aos operadores genéticos, foi desenvolvida uma variedade considerável de formas de realizar o cruzamento e a mutação [44]. Seus mecanismos de ação não diferem muito dos mencionados na PG tradicional, entretanto. A mutação consiste em substituir uma parte do código de um programa por outra formada aleatoriamente. Já as operações de cruzamento, que em suas formas mais comuns são chamadas de cruzamento normal e cruzamento homólogo (e respectivamente representadas nas Figuras 3.10 e 3.11), funcionam da seguinte maneira: uma

parte do material genético de um programa pai é trocado com o de outro, de maneira a variar (cruzamento normal) ou não (cruzamento homólogo) o tamanho dos programas filhos.

Figura 3.10 – Cruzamento normal na PG linear.

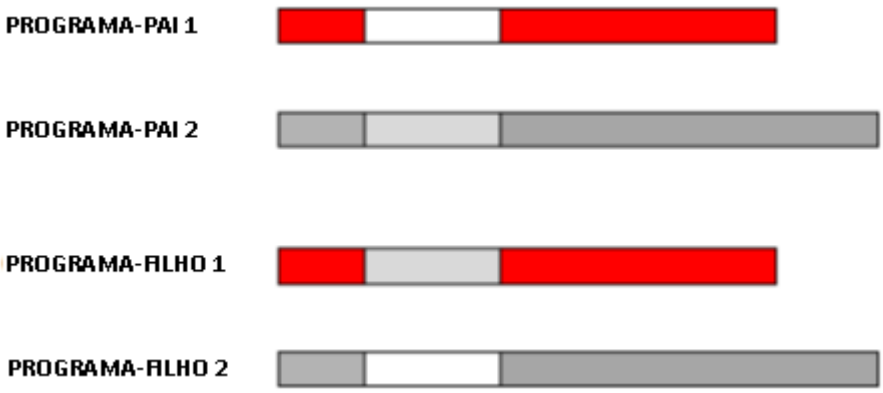


Figura 3.11 – Cruzamento homólogo na PG linear.

3.3.3.2 – Programação genética gráfica

Diferentemente da PG linear, vista como unidimensional, a idéia básica da programação genética gráfica é a evolução de programas paralelamente (fazendo uma analogia com evolução bidimensional), seja no sentido de generalizar a interpretação das árvores para reutilizar resultados parciais visando obter ganhos de eficiência (parallel distributed GP – PDGP [45]), fazer uma fusão da PG com discriminação linear para obter a classificação paralela de sinais e imagens (parallel algorithm discovery and orchestration – PADO [46]) ou representar programas como cromossomos lineares indexados por números inteiros correspondentes a pontos de um vetor bidimensional (Cartesian GP [47]).

Todavia, com exceção da PG cartesiana, que está sendo efetivamente explorada e aplicada até os dias atuais [48-50], as duas outras técnicas citadas entraram em relativo desuso.

3.3.4 – Programação genética de multiobjetivo

A PG de multiobjetivo, assim como ocorre com os demais algoritmos evolucionários, envolve a otimização de soluções com respeito à múltiplas metas ou funções de fitness, $f_1, f_2,...$, simultaneamente.

Os diferentes objetivos podem ser combinados de tal forma a produzir uma, assim chamada, "função de fitness agregada" (Seção 3.3.4.1) ou podem ser mantidos separados de fato, envolvendo conceitos de eficiência de Pareto (Seção 3.3.4.2).

3.3.4.1 – Função de fitness agregada

Desde a fundação da PG, os conceitos de multiobjetivo estão sendo levados em consideração, ambicionando combinar o alvo padrão, de encontrar programas que apresentam melhor fitness, com o objetivo de obter programas menores, mais parcimoniosos e tratáveis pelo ser humano [51-53]. De fato, a possibilidade inerente à PG de variar tamanho e forma das soluções durante a evolução leva àquele que deve ser um dos seus principais obstáculos (que serão melhor descritos na Seção 3.4): o fenômeno de "bloating" (inchaço), no qual o tamanho dos programas pode crescer descontroladamente, sem proporcional melhora na fitness.

Posteriormente, implementações da PG de multiobjetivo com outras combinações de metas foram realizadas com considerável sucesso: ZHANG & BHOWAN [54] aplicaram a técnica ao problema de detecção de objetos, sendo a função agregada uma combinação linear simples da taxa de detecção, taxa de falsos positivos e área dos falsos positivos; O'REILLY & HEMBERG [55] combinaram 6 objetivos numa aplicação na área de sistemas de Lindenmayer, ao passo que KOZA *et al*. [56] Consideraram 16 diferentes objetivos para o projeto de circuitos elétricos analógicos.

3.3.4.2 – Mantendo os objetivos separados

Aqui, é comum a introdução dos conceitos de dominância de Pareto: dado um grupo de objetivos, diz-se que uma solução domina a outra se esta não é inferior à outra em nenhum objetivo e, adicionalmente, é melhor que ela em pelo menos um dos objetivos. Um exemplo bidimensional da noção de ótimos de Pareto está ilustrado na Figura 3.12: o indivíduo A domina o indivíduo B ao longo do eixo y, mas B domina A nos pontos do eixo x, de maneira que não há ordenamento entre eles. O indivíduo 2, entretanto, domina B nos dois eixos e portanto seria considerado estritamente melhor que B.

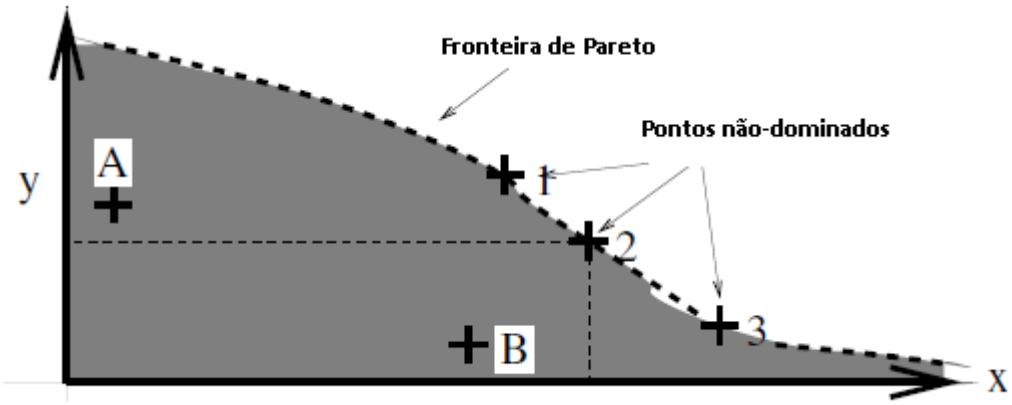


Figura 3.12 – Exemplo bidimensional da fronteira de Pareto [17].

Assim, a meta do algoritmo de busca torna-se a identificação do grupo de soluções que não são dominadas por nenhuma outra. Idealmente, busca-se encontrar a fronteira de Pareto, isto é, o grupo de todas as soluções no espaço de busca que não são dominadas por nenhuma outra. Realisticamente, entretanto, a fronteira de Pareto está limitada pela precisão da representação do problema: se x e y não forem enumeráveis (forem, por exemplo, números reais), a fronteira de Pareto também não o será.

Na literatura, há exemplos de aplicações da PG de multiobjetivos separados em LANGDON [57], que utilizou seleção de Pareto com dois objetivos (fitness e velocidade) para melhorar a performance da PG no clássico problema de benchmark da trilha de formigas de Santa Fé; ROSS [58] considerou diferentes variações de seleção de Pareto numa PG que evolui texturas 2D e na identificação de sistemas caóticos, com os objetivos indo desde a variância das séries temporais caóticas até complexidade e performance dos modelos, entre outras várias aplicações [59-61].

3.3.5 - Programação genética desenvolvimentista

Trata-se de uma das subáreas de pesquisa mais prolíficas da PG, com praticamente metade dos resultados mais relevantes na área alcançados até 2010 tendo sido obtidos através dela [62].

A PG desenvolvimentista (developmental genetic programming) [41,63] consiste na atribuição de estruturas mais gerais para o set de primitivas, tais como “integrador”, “ganho” e “lag” para o set de funções de um controlador, ou “capacitor”, “indutor” e “resistor” no set de funções para a evolução de um circuito elétrico (Figura 3.13). É importante que as primitivas

sejam capazes de evoluir estruturas apropriadas ao domínio do problema, seja inserindo e dimensionando componentes, modificando topologicamente as estruturas, etc.

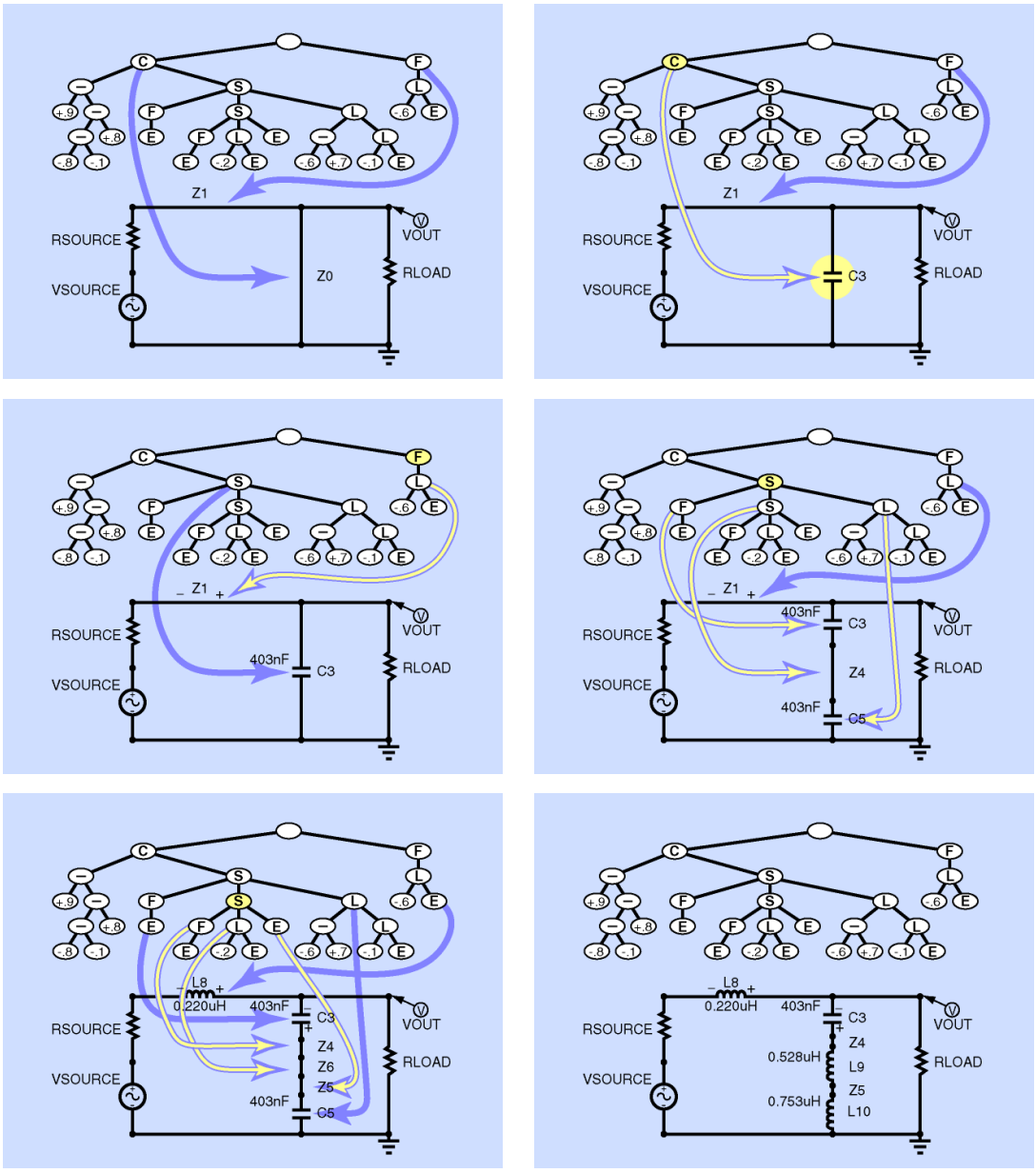


Figura 3.13 – Diagrama representando um circuito elétrico na PG desenvolvimentista [63].

Através dessa generalização, a intenção é que a PG seja capaz de ir além da produção de programas de computador (ainda que indiretamente). A PG desenvolvimentista já foi explorada na evolução de redes neurais [64,65], circuitos eletrônicos [66,67] e numerosas outras áreas.

3.3.6 – Outras áreas de pesquisa

Existem, ainda, outras técnicas avançadas em PG menos exploradas que as citadas nos últimos tópicos, ou que caíram em desuso com os anos. São elas, entre outras:

- Programação Genética Probabilística [68];
- Programação Genética Distribuída Geograficamente [69,70];

3.4 – Fundamentação Teórica da PG

Ainda há muitas questões em aberto, do ponto de vista teórico, para a PG [71]. Dentre as seguintes, as principais serão tratadas separadamente nesta seção:

- Teoria de esquemas (Seção 3.4.1);
- Aplicação de cadeias de Markov à PG (Seção 3.4.2);
- Análise do espaço de busca (Seção 3.4.3);
- Complexidade e convergência (Seção 3.4.4);
- Bloating (Seção 3.4.5);
- Otimização de parâmetros;
- Existência (ou não) de um "almoço grátis" para a PG [71];
- etc.

Conquanto haja essa falta de resultados quanto a sua fundamentação teórica, a PG tem se mostrado empiricamente eficiente para resolver uma vasta gama de problemas (alguns dos quais foram citados nas seções anteriores). Além disso, progressos têm sido feitos no desenvolvimento de técnicas anti-bloating que favorecem a evolução de programas mais parcimoniosos.

3.4.1 – Teoria de esquemas

Esquemas, no contexto de algoritmos evolucionários, são grupos de pontos do espaço de busca que compartilham alguma característica sintática em comum. A teoria dos esquemas nas técnicas de computação evolucionária AG e PG consiste basicamente em descrever de que maneira o número (ou fração) dos indivíduos de uma população variam com o tempo e a aplicação dos operadores genéticos (de uma geração para outra).

Do ponto de vista teórico, uma teoria de esquemas é muito importante para a PG, pois permite entender melhor a evolução dos indivíduos ao longo das gerações, fazer comparações

entre os operadores genéticos, estudar a evolução dos tamanhos dos programas, investigar o fenômeno de bloating, convergência, tamanho da população, deception, entre outras aplicações.

De uma forma geral, a teoria dos esquemas para o AG já é bastante geral e consolidada [72,73]. Porém, até 2000, para a PG apenas havia sido desenvolvida a teoria exata para um tipo de crossover (o one-point crossover), além de vários outros estudos mais superficiais que forneciam "worst-case-scenarios" e limites inferiores, isto é, o mínimo de indivíduos de um determinado esquema que, em média, iriam para a próxima geração.

POLI [40] através da definição de hiperesquemas, desenvolveu o teorema exato para vários outros tipos de crossover para a PG. De fato, a teoria apresentada, embora não tão geral quanto a teoria de esquemas para o AG, é cabível para todos os crossovers que têm como mecanismo a troca de subárvores, o que inclui o standard crossover, considerado o crossover mais importante utilizado na PG.

Apesar de mais geral, a definição de hiperesquemas não contorna a dificuldade básica da fundamentação de uma teoria de esquemas para a PG. A característica inerente à técnica de variar tamanho e forma dos programas durante a evolução (Seção 3.2.2), diametralmente oposta ao tamanho fixo de *bits* binários do AG, faz com que a teoria de esquemas da PG seja muito menos direta e muito mais complexa matematicamente. Os hiperesquemas trouxeram um resultado acadêmico aguardado há vários anos dentro da área da programação genética, pois pode ser aplicado no estudo da probabilidade de transmissão dos esquemas do standard crossover (equação 3.2), mas está longe de solucionar completamente o problema teórico e trazer os benefícios já ressaltados nesta seção.

$$\alpha(H,t) = (1-p_{xo})p(H,t) + p_{xo}\sum_{k,l}\frac{1}{N(G_k)N(G_l)}\sum_{i\in H\cap G_k}\sum_{j\in G_l}p(U(H,i)\cap$$

$$G_k,t)p(L(H,i,j)\cap G_l,t) \qquad (3.2)$$

3.4.2 – Cadeias de Markov

As cadeias de Markov são relevantes na análise teórica de algoritmos evolucionários operando em espaços de busca discretos [74]. Em verdade, o AG e a PG são estocásticos com comportamento Markoviano, já que a população na geração seguinte tipicamente depende diretamente apenas da população da geração atual. Assim, em tese, para estudar o comportamento do algoritmo basta computar a probabilidade de que este mova-se de um determinado estado (ou, neste caso, de uma determinada população) para outro. Estas

probabilidades são usualmente ordenadas em uma matriz estocástica *M*. Isto posto, *M* guarda as probabilidades de todas as possíveis transições que o algoritmo pode fazer em uma geração. Elevada ao quadrado, ela fornecerá as probabilidades de transição em duas gerações, e assim sucessivamente.

Assim, tomando potências da matriz *M* e multiplicando o resultado pela distribuição de probabilidades inicial (obtida na inicialização do algoritmo), em princípio é possível calcular a probabilidade de que um problema seja resolvido após *n* gerações, qual será a melhor fitness e a fitness média após *n* gerações, etc.

Infelizmente, apesar de alguns avanços [75,76], os modelos markovianos aplicados à PG ainda estão pouco desenvolvidos, fornecendo ainda poucos resultados concretos (ver Seção 3.4.4).

3.4.3 – Distribuição da fitness pelo espaço de busca

A caracterização da distribuição da fitness ao longo do espaço de busca explorado pela PG é outro tópico de pesquisa teórica importante [77].

Nesta área, o principal resultado é a descoberta de que a distribuição da funcionalidade[7] de programas não universais[8] converge para um limite à medida em que o tamanho do programa aumenta (embora o número de programas possíveis para um dado comprimento aumente exponencialmente). A prova matemática para sistemas Lisp e códigos sem loops, além de estudos subsequentes, podem ser encontrados em [78-82].

Recentemente, um certo esforço foi feito no sentido de demonstrar a limitação da funcionalidade de programas Turing-completos [83], mas pouco foi alcançado de mais significativo a respeito de como analisar e melhorar a busca no espaço de programas. Um mecanismo para estudar o impacto de operadores de crossovers em termos semânticos [84] e a possibilidade de não repetir programas com o mesmo grau de funcionalidade durante a busca [85] são dois dos exemplos mais proeminentes.

[7] No contexto da PG, as árvores de sintaxe são análogas aos genótipos e suas funcionalidades (ou semânticas) aos fenótipos. Em outras palavras, a funcionalidade de um programa é simplesmente a sua interpretação pelo sistema.

[8] Ou, mais formalmente, não Turing-completos.

3.4.4 – Complexidade e convergência

Por conta da escassez de resultados teóricos importantes, ilustrada nas seções anteriores, a ergodicidade[9] de um sistema PG ainda é uma questão em aberto. Ainda assim, algumas propriedades teóricas das matrizes de transição de Markov podem ser estabelecidas para alguns casos.

Na prática, afirmar que um sistema PG pode encontrar soluções para um problema (transcorridas gerações suficientes) significa estabelecer que existe um número inteiro k tal que todos os elementos da matriz de transição M^k são não-nulos.

Em RUDOLPH [86] e SCHMITT & DROSTE [87] encontram-se provas matematicamente complexas de que a PG apresenta convergência garantida em uma quantidade generalizada de espaços de busca, dado que a probabilidade de mutação seja não-nula. Baseado nesses resultados, [88] apresenta uma prova da convergência de um sistema PG teórico onde as probabilidades de mutação e cruzamento são progressivamente levadas à zero.

Apesar de esse ser naturalmente o próximo passo, a aplicação desses resultados, juntamente com técnicas de complexidade computacional, na análise da velocidade e número de gerações em que a PG irá até a solução ainda é muito incipiente. É razoável supor, entretanto, que esses estudos figurem na literatura em breve, já que são uma tendência em outros tipos de algoritmos evolucionários.

3.4.5 – Bloat

Talvez o problema mais inconveniente de ordem prática durante a execução de um código de programação genética seja o fenômeno de bloating, definido como o aumento repentino e desenfreado no tamanho dos programas após algumas gerações, sem correspondente melhora relevante na fitness. Extremamente comum, tal "inchaço" no tamanho dos programas traz consequências graves diretas à performance do algoritmo, devido à inclusão de introns[10] nos descendentes das gerações. Entre seus efeitos estão: lentidão na avaliação dos

[9] Termo que originalmente está relacionado a sistemas nos quais a evolução futura pode ser prevista através de cálculos probabilísticos. Aqui, denota a capacidade de um algoritmo alcançar todas as possíveis soluções para um problema, independentemente do tempo.

[10] Definidos como os fragmentos "inúteis" de código acumulados devido ao fenômeno de bloating, isto é, código que não afeta o resultado da avaliação do indivíduo da PG. O termo também tem analogia direta com a biologia, onde introns são seções de DNA de um gene que não codificam qualquer parte da proteína produzida pelo gene.

indivíduos, consumo de mais recursos computacionais e aumento na complexidade das soluções, dificultando sua interpretação.

Embora alguns autores [89] defendam a posição de que o código intron pode também ter um papel benéfico num sistema PG – na medida em que fornece uma proteção estrutural que tende a preservar os blocos mais aptos de um indivíduo, evitando que estes sejam degenerados quando da aplicação dos operadores genéticos – é consenso na comunidade acadêmica que o fenômeno de bloating deve ser, se não evitado, pelo menos controlado.

O bloating foi um dos primeiros assuntos a ser tratado teoricamente na PG, há mais de duas décadas, e desde então tem sido alvo constante de estudos acadêmicos. Dessa forma, um sem-número de pesquisas já foram feitas contemplando desde o seu mecanismo até ferramentas para combatê-lo. Mesmo assim, ainda é uma questão teórica em aberto até mesmo os fatores que levam ao bloating, e como ele progride e se propaga pelas gerações.

3.4.5.1 – Modelos teóricos sobre o bloating

Existem explicações qualitativas clássicas sobre o fenômeno de bloating, como as teorias da "tendência de remoção" [90] (removal bias) e a de "replicação precisa" [91], que "culpam" a operação de crossover pelo inchaço dos programas, e a teoria da natureza do espaço de busca [92] que, baseado no descrito na Seção 3.4.3, conjectura que a PG tende a selecionar e produzir indivíduos cada vez maiores simplesmente porque há mais deles, visto que após um certo tamanho de programas, a distribuição das fitness não varia com o tamanho.

Também existem trabalhos que visam formalizar quantitativamente o fenômeno de bloating [93,94]. O mais rigoroso deles se apoia no resultado da teoria de esquemas discutido na Seção 3.4.1 e estabelece a equação (3.3), uma expressão incremental para a variação no tamanho médio dos programas ao longo das gerações, válida para sistemas PG com qualquer operador de crossover simétrico[11]:

[11] Operadores simétricos são aqueles em que a probabilidade de selecionar um ponto de cruzamento em particular nos programas-pais não depende da ordem em que estes são selecionados na população

$$E[\mu(t+1) - \mu(t)] = \sum_l l \times \big(p(l,t) - \Phi(l,t)\big) \quad (3.3)$$

onde $\Phi(l,t)$ é a proporção de programas de tamanho l na geração atual, E é o operador esperança matemática, $\mu(t)$ representa o tamanho médio dos programas na geração t e $p(l,t)$ é a probabilidade de que sejam selecionados programas de tamanho l na geração t[12].

De fato, existe uma explicação mais recente para o bloating que está baseada e é consistente com a equação (3.3) e a teoria de esquemas. Chamada de "crossover bias theory" [96], afirma que o cruzamento tende a levar a população a uma distribuição de tamanhos de programas (conhecidas como distribuição de Lagrange de segundo e terceiro tipos) em que programas pequenos são muito mais comuns que programas grandes. Como os programas pequenos usualmente não tem muitas chances de resolver o problema (apresentam fitness baixa), eles tenderão a ser ignorados pelos mecanismos de seleção, aumentando o tamanho médio dos programas.

3.4.5.2 – Controlando o bloating

Diversas técnicas tem sido propostas para controlar o bloating, incluindo a definição prévia de tamanhos e profundidades limites; utilização de operadores genéticos com viés anti-bloating embutido [97]; as modelagens de multiobjetivo (já mencionadas na Seção 3.3.4, que otimizam a fitness enquanto minimizam o tamanho dos programas) e aquele que curiosamente foi um dos primeiros métodos propostos, ainda por Koza em 1992, mas que até hoje é um dos mais eficientes: o método de pressão parcimoniosa (parsimony pressure method – PPM).

Como ficou claro através dos estudos da teoria de esquemas e evolução de tamanhos de programas (equação 3.3), em sistemas com operadores simétricos o bloating só pode acontecer se houver um desequilíbrio entre as probabilidades de seleção e as frequências dos programas, isto é, caso programas maiores que a média apresentem fitness melhor que a fitness média, caso programas menores que a média possuam fitness pior que a fitness média, ou ambos. Por conta disso, fica claro que, para controlar o bloating, é necessário modular as probabilidades de seleção levando em consideração os seus tamanhos.

O método de pressão parcimoniosa faz exatamente isso: modifica as probabilidades de seleção dinamicamente ao atribuir uma penalidade à fitness dos programas de acordo com seu tamanho. Assim, a nova função de fitness (considerada apenas para guiar a evolução, e não para

[12] É interessante notar que, para seleção proporcional à fitness, esta expressão matemática é equivalente à equação de Price em seu teorema sobre seleção e covariância [95].

efeitos de apresentação de soluções e critérios de parada) é dada por um $f(x) = c * l(x)$, com a constante c sendo conhecida como "coeficiente de parcimônia". A determinação deste parâmetro varia com o problema e a aplicação: defini-la muito pequena fará o bloating ocorrer livremente da mesma forma, e muito grande fará a PG dar valor demais à minimização de tamanhos, podendo convergir para programas pouco úteis. Devido a isso, existem trabalhos [52] que apontam como boa prática o ajuste adaptativo do coeficiente a cada geração.

3.5 – Resultados Práticos da PG

Não obstante o que foi exibido até aqui, desde a sua concepção, em 1992, a PG vem sendo utilizada para a resolução de muitos problemas em uma vasta gama de áreas [17,98-106], como circuitos de computação quântica, circuitos elétricos analógicos, antenas, sistemas mecânicos, controladores, jogos, álgebra finita, sistemas de fotônica, reconhecimento de imagem, redes neurais, sistemas de lentes ópticas, séries temporais, data mining, bioinformática, algoritmos de ordenação, geração de código, robótica, planejamento (scheduling), regressão simbólica, engenharia reversa, entre muitos outros.

Até 2008, havia mais de 5000 aplicações da PG disponíveis na literatura. Dessas, 76 são consideradas "human-competitive" de acordo com os critérios estabelecidos em KOZA [107], sendo 21 tão bons quanto resultados obtidos e patenteados no século 20, 7 tão bons quanto patentes do século 21 e 2 que são invenções patenteáveis: dois controladores (um PID – proporcional integral derivativo – e um não-PID) com parâmetros que fornecem performances melhores que os disponíveis até então para algumas aplicações [108]. Essas últimas invenções, em particular, foram obtidas rodando um sistema PG por 29 dias, com uma população de 100000 indivíduos e tamanho máximo de árvore (depth) de 500.

Tudo indica, portanto, que a melhora nos resultados obtidos pela PG está em correlação direta com o aumento da capacidade de processamento e armazenamento dos computadores (lei de Moore), o que sugere que ela será cada vez mais utilizada para obter resultados que sejam "human-competitive".

CAPÍTULO 4

SISTEMAS DE DIAGNÓSTICO DE TRANSIENTES EM USINAS NUCLEARES: ESTADO DA ARTE

4.1 – Introdução

Como já dito no Capítulo 1, uma usina nuclear (UN) é um sistema crítico do ponto de vista da segurança, sendo este o principal fator a ser levado em consideração na sua concepção e operação. Ademais, há uma demanda cada vez maior para que as usinas sejam operadas de maneira rentável, estando o mais próximo possível de sua capacidade máxima. Melhorar, concomitantemente, os fatores de segurança e de capacidade praticados nas UNs implica em adotar procedimentos preventivos eficientes para lidar com os possíveis contratempos que aparecem durante a operação.

Uma falha pode ser definida como um desvio da condição desejada de funcionamento, capaz de impedir o sistema de realizar uma determinada ação com um nível de desempenho especificado [109]. A grande quantidade de variáveis monitoradas em uma instalação nuclear leva a um sem-número de falhas possíveis nos equipamentos, instrumentos ou processos, com consequências que podem ir desde pequenas variações na potência de saída do reator, desligamento rápido do reator (trip) ou exposição de pessoas à radiação, em casos mais graves onde ocorre uma sucessão de falhas em cadeia.

Devido às razões citadas aqui e no Capítulo 1, sistemas inteligentes de detecção e diagnóstico de falhas (DDFs) têm sido elaborados ao longo das últimas quatro décadas com o objetivo de detectar, isolar e identificar as falhas de um sistema [110]. Esses três processos significam, respectivamente, determinar se há alguma falha, encontrar a sua localização e, por último, sua extensão e características.

Os DDFs podem ser aplicados no monitoramento contínuo de um sistema operante, caso em que são chamados de sistemas online. São ainda classificados entre sistemas baseados em modelos ou não. Os sistemas baseados em modelos utilizam, como o nome sugere, modelos matemáticos para representar o funcionamento normal de um sistema. Falhas neste sistema são detectadas e diagnosticadas checando-se a consistência entre o comportamento medido e o previsto pelo modelo. Dada a dificuldade em obter-se um modelo prático e preciso de muitas situações e sistemas complexos reais, essa alternativa tem encontrado aplicação bastante limitada.

Já os sistemas livres de modelos são posteriormente classificados em métodos baseados em dados e baseados em sinais. Os DDFs baseados em dados consistem de técnicas estatísticas (técnicas de regressão, modelos autorregressivos e de médias móveis, etc.) ou técnicas de inteligência artificial (redes neurais, sistemas de lógica fuzzy, computação evolucionária, entre outros.), e também tem sua base de funcionamento na correlação entre as medidas do sistema. Entretanto, as relações são formuladas de maneira implícita, ao treinar padrões empíricos através da análise de dados de funcionamento normal (sem falhas) do sistema. Tais padrões são então usados para estimar os valores verdadeiros estatísticos das novas medidas, e as falhas são detectadas e diagnosticadas através de análise estatística de resíduos. Já os modelos baseados em sinais tomam as decisões de diagnóstico e detecção comparando atributos extraídos de um sinal com valores de linha de base esperados.

Um esquema com os principais sistemas de DDF encontra-se na Tabela 1. A Tabela 2, por sua vez, mostra como, ao longo das décadas, os modelos baseados em sinais e dados foram aplicados com diferentes graus de sucesso para sistemas de diagnóstico em usinas nucleares, trazendo grandes benefícios para a segurança e eficiência dessas centrais. De fato, do ponto de vista da segurança e economia, podem ser citados como alguns desses benefícios:

- Redução na exposição à radiação dos funcionários, já que os sistemas DDF permitem otimizar o agendamento das atividades de reparo e manutenção dos componentes;
- Aumento da confiabilidade dos equipamentos, através dos sistemas de monitoramento online que permitem detectar falhas incipientes e realizar atividades corretivas com maior antecedência;
- Aumento na capacidade de pronta resposta após uma falha, pois o rápido diagnóstico permite ganhar tempo para realizar as contramedidas na ocasião de uma falha, evitando inclusive que evoluam para situações mais graves;
- Otimização do tempo útil da central nuclear, ao diminuir paradas desnecessárias para inspeção de equipamentos ou por desligamento rápido do reator em situações adversas;
- Extensão da vida útil dos equipamentos e da usina como um todo, através de melhor monitoramento da performance da usina e da gestão do seu ciclo de vida.

Tabela 1 – Classificação dos Sistemas DDF.

Sistemas baseados em modelos	**Sistemas livres de modelos**	
	Métodos Baseados em Dados	**Métodos Baseados em Sinais**
Equações de Paridade	Redes Neurais Artificiais (RNAs)	Análise Espectral (SA)
Observador de Diagnóstico	Técnica de Estimação de Estado Multivariada (MSET)	Análise Tempo-Frequência (TFA)
Filtros de Kalman	Análise de Componentes Principais (PCA)	Transformada Wavelet (WT)
Estimação de Parâmetros	Mínimos Quadrados Parciais (PLS)	Modelo de Sinais Autorregressivo (ARSM)
	Computação Evolucionária (CE)	Cartas de Controle

Tabela 2 – Aplicação de métodos de DDF nas UNs.

Aplicação	**Métodos Baseados em Dados**	**Métodos Baseados em Sinais**
Monitoração da calibração de instrumentos	Estimação de saída dos sensores (RNA, MSET, PCA, AAKR)	_____
Monitoração da performance dinâmica da instrumentação		Análise de ruído (ARSM, SA)
Monitoração de equipamentos	Estimação de dados sensoriais (RNA, MSET)	Análise de vibração e emissão acústica (SA, WT, TFA)
Monitoração estrutural	_____	Análise de sinais acústicos das estruturas (SA, TFA)
Monitoração do núcleo do reator	Estimação de parâmetros do reator (RNA)	Análise de “neutron noises” (SA, WT, estimação de taxa de decaimento)
Identificação de transientes	Reconhecimento de padrões (RNA, LF, CE)	_____

4.2 – Sistemas de Diagnóstico de Transientes

Transientes em uma usina nuclear podem ser iniciados pela falha de um equipamento ou perturbações externas[13], e devem ser corretamente identificados o mais rapidamente possível para que ações adequadas sejam tomadas para evitar ou aplacar as consequências negativas. Um sistema de identificação de transientes automático, robusto e rápido é, portanto, de grande valia para salvaguardar o operador e a usina como um todo dos impactos desses eventos indesejados.

Durante um transiente, alguns outputs da instrumentação da usina apresentarão padrões diferentes daqueles observados durante a operação normal. Tais padrões podem ser diferentes dos provocados por outros transientes, podem variar com a severidade do problema e até mesmo com suas condições iniciais. Assim, o problema de identificação de transientes é, essencialmente, um problema de reconhecimento de padrões, mas que possui complexidade profunda, dado o enredamento dos sistemas de controle das centrais nucleares.

Ao longo das últimas duas décadas, a elaboração de sistemas de identificação automática de transientes em usinas nucleares como ferramenta de suporte ao operador tem sido realizada através de métodos estatísticos clássicos de reconhecimento de padrões (Seção 4.2.1) e principalmente através de três grandes técnicas de inteligência artificial, combinadas ou não, e suas subvariações: as redes neurais artificiais (Seção 4.2.2), a lógica fuzzy (Seção 4.2.3), e os métodos heurísticos de computação evolucionária (Seção 4.2.4).

4.2.1 – Métodos estatísticos clássicos de reconhecimento de padrões

Nos estudos de GALBALLY & GALBALLY [111], é apresentado um sistema de reconhecimento de padrões automático para a identificação de transientes em uma usina nuclear através da técnica “dynamic time warping” (DTW). Considerado bastante simples, ele se baseia na correspondência direta entre as variáveis temporais durante o transiente, dispensa etapas de pré-processamento e não requer grandes quantidades de dados de treino. O sistema foi testado considerando bases de dados operacionais reais da usina nuclear de Cofrentes, na Espanha, obtendo taxas de precisão de mais de 90%.

Utilizando a seleção sequencial de atributos (SSA) como método de redução dimensional em um sistema de reconhecimento de padrões clássico, LIN [112] elaborou

[13] Mais detalhes e definições sobre transientes serão dadas no Capítulo 5.

recentemente um esquema complexo de dois estágios capaz de facilitar a identificação de eventos iniciadores[14] de possíveis acidentes severos. A taxa de identificação correta nos dados de teste (que foram obtidos no simulador da usina nuclear de Maanshan) foi comparada à obtida em um sistema similar no qual o método de redução dimensional fora o algoritmo genético, ambos empatando em cerca de 95% de identificação correta em um cenário de 12 categorias de eventos com um total de 112 eventos. Entretanto, o sistema proposto com a técnica SFS apresentou velocidade cerca de 10 vezes maior no tempo computacional requerido para encontrar o conjunto ótimo de sensores para a identificação.

A modelagem funcional multinível (MFM) como ferramenta de diagnóstico nas usinas nucleares foi objeto de estudo de LIND & ZHANG [113]. Argumentando que a "gestão de falhas" (fault management) inclui diversas subtarefas, como a detecção, diagnóstico e recuperação, as decisões em situações com consequências potencialmente sérias à usina ou ao meio ambiente frequentemente ainda recaem nas mãos dos operadores, de maneira que a automatização nessas situações só pode ser feita até um certo limite, dependendo do risco envolvido. A modelagem funcional, segundo os autores, seria adequada para a tarefa de diagnóstico por ser capaz de abarcar os níveis complexos de abstração (referentes à análise de objetivos, causa e consequência e relevância relativa) que advém do conhecimento acerca do funcionamento da planta. O sistema foi aplicado na análise de operação simulada de um reator PWR e um reator FBR ("fast breeder reactor"), gerando árvores de falha que foram avaliadas como mais completas do que as obtidas através das metodologias tradicionais.

Em um trabalho bastante recente, WU *et al*. [114] apresentaram um sistema de RP para detecção de eventos iniciadores que incorpora o teste T2 de Hotelling, estimação de assinatura espacial (spatial signature) e identificação de outliers para contornar as limitações que, segundo os autores, estão presentes na maioria dos demais sistemas de identificação – quais sejam, respectivamente, não levar em conta a simultaneidade na computação dos dados advindos dos sensores, desconsiderar a informação espacial (interrelação entre os sensores), que pode ser útil na diferenciação de eventos e, por último, o diagnóstico incorreto de eventos que não estão presentes no conjunto de treino do sistema – presentes na literatura científica. A performance do sistema proposto foi avaliada aplicando-o a um conjunto de dados obtidos no simulador da usina PWR de Maanshan, obtendo uma taxa máxima de identificação correta de 96,16%, tempo de detecção entre 1 a 10 segundos e isolamento de falhas fora do conjunto de treino de 92,8%.

[14] Definidos como eventos que causam um distúrbio nas UNs que podem levar à danos ao núcleo.

Demais trabalhos relevantes aplicando técnicas clássicas de RP incluem: LIN & CHANG [115] com a técnica de "casamento" (template matching); MA & JIANG [116] que estudaram um sistema de diagnóstico baseado em classificação semi-supervisionada (SSC) e GOFUKU [117] que propôs um sistema híbrido de diagnóstico composto por um módulo de estimação de variáveis relevantes através de análise do sistema físico, SVM para a identificação de estados, transformada wavelet (WT) para detecção de variações nos sinais medidos e diagnóstico baseado em medidas de similaridade.

4.2.2 – RNA

A técnica de IA tradicionalmente mais investigada e amplamente utilizada para a identificação de transientes nas UNs é a rede neural artificial. Revisões bibliográficas bastante satisfatórias dos trabalhos feitos até 2007, principalmente com as RNAs, podem ser encontradas em MÓL [118] e SANTOSH [119]. Inicialmente, a maioria dos estudos consistia em utilizar dados de simuladores para treinar as redes de maneira a tomar as informações de uma pequena quantidade de sensores como dados de entrada por toda a duração do transiente. Embora seja uma técnica fácil de implementar, dificilmente era capaz de identificar o transiente em tempo hábil.

Alternativamente, trabalhos como [120] e [121] sugerem uma abordagem mais complexa, na qual as RNAs são treinadas para, tomando medidas instantâneas de uma quantidade maior de sensores como inputs, diagnosticarem o transiente enquanto ele se desenvolve. A importante lacuna nos sistemas de redes iniciais de identificar os transientes não pertencentes ao conjunto de testes como "não sei", ao invés de tentar classificá-los por similaridade, foi preenchida nas pesquisas de BARTAL [122], MÓL *et al*. [123] e EMBRECHTS [124] utilizando técnicas como as redes neurais probabilísticas e as redes neurais com "saltos" (esta última esquematizada na Figura 4.1), as quais aumentam a sensibilidade das primeiras camadas da rede para os erros apresentados em seu output.

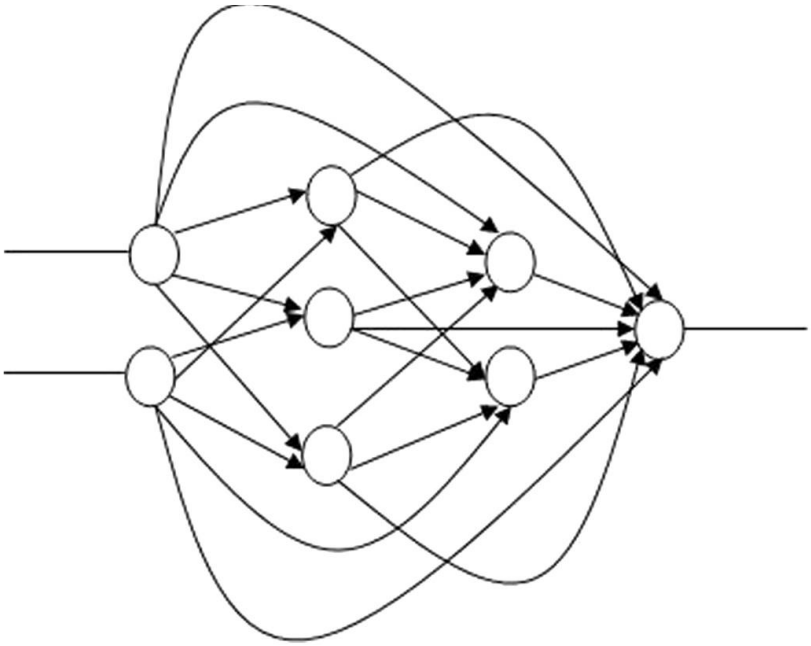


Figura 4.1 – Vista esquemática de uma rede neural com saltos [125].

Mais recentemente, redes neurais recorrentes, como a rede de Elman (Figura 4.2) e de Jordan, tem sido usadas como ferramentas para lidar com sinais de inputs temporais, principalmente quando o transiente é tão rápido que pode ser tratado como um estado de quase-equilíbrio, ou quando a identificação imediata do transiente é particularmente importante [126].

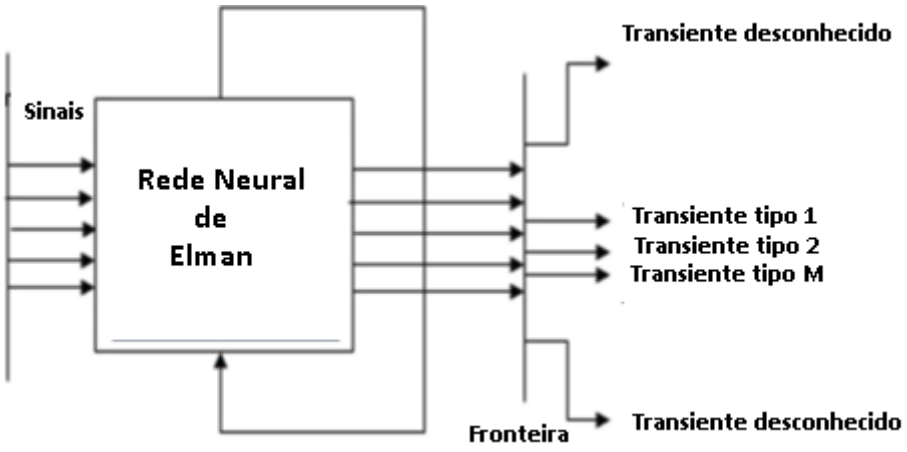


Figura 4.2 – Classificador de Elman [125].

4.2.3 – Lógica fuzzy

Cada vez mais aparecem na literatura aplicações de técnicas de IA alternativas às RNAs (embora estas ainda sejam relativamente comuns e importantes) para sistemas de diagnóstico

de transientes, como lógica fuzzy e computação evolucionária, principalmente devido ao ceticismo quanto à confiabilidade das redes em aplicações em sistemas críticos de segurança, dada a sua característica "caixa-preta", que limitam a verificabilidade e a compreensão de sua performance [127].

De fato, talvez a segunda metodologia mais explorada ao longo dos anos no problema de identificação de transientes em usinas nucleares tenha sido a lógica fuzzy (esquematizada na Figura 4.3), seja de maneira solitária ou combinada com as redes neurais, dando origem aos chamados sistemas híbridos "neuro-fuzzy".

Posto isso, ZIO *et al.* [128] aplicam análise de similaridade de dados baseada em lógica fuzzy para propor um sistema de identificação de modos de falha (FM) e de estimação de tempo de recuperação (RT) que consiste em considerar um grupo de padrões multidimensionais advindos de diversos cenários de acidentes e aplicar a análise de similaridade numa série de dados de um acidente em desenvolvimento. A metodologia proposta foi aplicada em um conjunto de cenários de acidentes disponível, obtendo resultados satisfatórios tanto do ponto de vista de precisão quanto de velocidade de diagnóstico.

Outro trabalho utilizando lógica fuzzy encontra-se em PRUSTY *et al*. [129]. Nele, alguns dos aspectos delicados dos sistemas baseados na fuzificação e defuzificação de dados são avaliados, com vistas à otimização dos mesmos para que a identificação dos transientes possa ser realizada na janela de tempo típica de recebimento de informações em uma usina nuclear – da ordem de milissegundos. Para tanto, diversos sistemas de classificação baseados em lógica fuzzy, com diferentes parâmetros, são examinados quanto à performance em grupos de dados de um simulador de uma usina com reator FBR. São abordados fatores como a quantidade ótima de variáveis de entrada, a interpretabilidade dos resultados fornecidos e sua precisão.

Sistemas neuro-fuzzy foram implementados para representação qualitativa dos transientes em EVSUKOFF & GENTIL [130] e, em COSTA *et al.*[131], como uma combinação de redes neurais perceptron multi-camadas (MLP) modular e sistema fuzzy, para treinamento de um identificador de uma série de transientes.

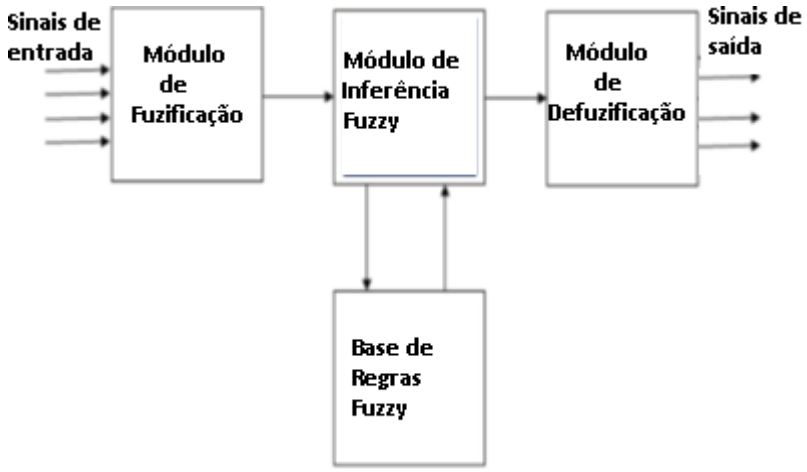


Figura 4.3 – representação esquemática de um sistema de lógica fuzzy.

4.2.4 – Computação evolucionária

Ainda que a maioria dos trabalhos presentes na literatura sobre a identificação de transientes nas usinas nucleares esteja baseada em técnicas como as RNAs e sistemas fuzzy, essas duas técnicas apresentam limitações. A dificuldade na modelagem de dados temporais e no reconhecimento de transientes fora do conjunto de treinamento são dois grandes problemas das redes neurais perceptron de múltiplas camadas (MLPs). Tentar contorná-los utilizando redes neurais mais robustas e avançadas, como as redes neurais recorrentes, usualmente traz consigo a desvantagem destas apresentarem um processo de treinamento longo e difícil, que tende a ficar preso em mínimos locais. Os sistemas fuzzy, por sua vez, embora não sofram tanto com esses fatores e sejam mais robustos diante de dados ruidosos, apresentam as restrições de ser necessária a interpretação dos seus resultados e de normalmente não cobrirem uma quantidade suficiente de transientes para serem aplicados à operação real de uma usina nuclear.

Embora os sistemas combinados neuro-fuzzy venham aparecendo como formas elaboradas de compensar esses pontos fracos, técnicas heurísticas de computação evolucionária também têm aparecido na literatura como alternativas modernas capazes de contornar algumas dessas fraquezas, possuindo grande capacidade de generalização e maior eficiência na busca dos mínimos globais durante a modelagem. Além disso, utilizar técnicas heurísticas como método de seleção de atributos permite otimizar o próprio processo de classificação, identificando quais variáveis são mais relevantes para cada tipo de transiente e restringindo a monitoração a elas, por exemplo. De fato, algumas técnicas heurísticas consideradas recentemente têm frequentemente superado as redes neurais em desempenho (embora estas últimas também estejam melhorando seus resultados).

Um dos primeiros trabalhos que aplicou técnicas de computação evolucionária como a principal ferramenta na construção de um sistema de diagnóstico para as UNs data do ano 2000, com YANGPING *et al.* [132]. Utilizando o algoritmo genético para a busca de "indivíduos" adequados à representação das condições da planta, os autores realizaram experimentos com dados de um simulador da central nuclear de Pequim, conseguindo bons resultados na identificação de algumas situações de falhas. Não considerar o problema da atribuição de um rótulo "não sei" para eventos fora do grupo de treinamento, a intrincada adaptação à função de fitness elaborada (ilustrada na equação 4.1) e a quantidade limitada de falhas testadas são algumas das desvantagens do sistema.

$$F_j = \left(\sum_{i=1}^{n}\left(\left(\sum_{k=1}^{m} I_{ik}\right)\delta_i + (1-\delta_i)|lgP_i|\right)\right)\sum_{i=1}^{n}\left(\delta_i \sum_{k=1}^{m} S_k\right)\left(\sum_l \sum_p \left(\left(\delta_l \delta_p\right)N_{lp}\right)\right)A_j \qquad (4.1)$$

Dois anos mais tarde, ALMEIDA *et al.* [133] apresentaram uma abordagem possibilística, baseada na teoria dos conjuntos nebulosos e no método do conjunto mínimo de centroides (CMC) otimizado por algoritmo genético, para a identificação de transientes com geração de respostas "não sei" para transientes desconhecidos pelo sistema, o que é fundamental para não induzir o operador à ações erradas. A abordagem foi testada na identificação de três transientes considerando 15 variáveis, e funcionou de maneira robusta, precisa e segura, mesmo em situações em que foi introduzido ruído ao conjunto de dados, na qual o sistema tendeu a fornecer a resposta "não sei", mais conservativa.

O algoritmo evolucionário com inspiração quântica (QEA) foi testado para identificação automática de condições anormais e acidentes em NICOLAU *et al.*[134]. O sistema consiste em comparar as distâncias euclidianas entre grupos de variáveis temporais do evento anômalo e os vetores pseudo-centróides (que, idealmente, seriam os vetores de Voronoi) de cada um dos transientes que constavam no conjunto de treinamento (na ocasião foram considerados três transientes); aquele que apresentasse a menor distância indicaria a classe do transiente ao qual o evento anômalo pertence. A função do QEA é trabalhar como otimizador para encontrar as melhores "coordenadas" desses vetores-protótipo. De um conjunto contendo a evolução temporal, durante 60 segundos, de 16 variáveis, várias combinações foram feitas buscando aquelas que forneceriam a maior capacidade de identificação e discriminação para os transientes envolvidos (praticamente um processo "fio de Ariadne" de redução dimensional). Por fim, os resultados obtidos, inclusive com a adição de ruído aos sinais das variáveis, mostraram a robustez e eficiência do sistema, chegando a 100% de classificação correta em alguns casos com apenas quatro variáveis. Uma versão mais robusta e ampla desse sistema otimizado pelo

QEA foi publicada há pouco tempo em NICOLAU & SCHIRRU [135] abarcando a identificação de transientes desconhecidos como “não sei” através da identificação de áreas de influência em torno dos vetores representativos de cada evento.

Juntamente com o QEA, talvez a técnica de computação evolucionária que mais vem sendo explorada ao longo dos últimos anos em sistemas de diagnóstico de transientes seja a otimização por enxame de partículas (PSO) e sua variação com inspiração quântica, a QDPSO (quantum delta potential swarm optimization). Como alguns dos exemplos mais relevantes, em 2008, com MEDEIROS & SCHIRRU [136], o PSO foi utilizado como ferramenta de otimização de um identificador baseado em distâncias, obtendo classificações corretas de até 100%, dependendo do número de partições do espaço de busca, considerando três transientes. Posteriormente, em 2014 (consultar NICOLAU & SCHIRRU [137]) a técnica QDPSO (Quantum Delta Potential Swarm Optimization) foi testada como otimizador em um sistema similar, mas que considera agora distâncias de Minkowski, e não mais euclidianas, como métrica para a avaliação de similaridades.

Até onde foram as buscas e pesquisas deste autor, este é o primeiro trabalho disponível na literatura em que a técnica de computação evolucionária programação genética é utilizada como ferramenta de otimização e geração de atributos em um sistema de identificação de transientes em usinas nucleares.

CAPÍTULO 5

APLICAÇÃO DA PROGRAMAÇÃO GENÉTICA A UM CONJUNTO DE ACIDENTES POSTULADOS PARA UMA USINA NUCLEAR PWR

5.1 - Introdução

Neste capítulo será descrito o conjunto de acidentes considerado para os testes experimentais do sistema proposto de identificação de transientes através da programação genética, bem como os pormenores referentes à implementação do referido sistema. Serão também relatados os resultados obtidos com a técnica, apresentando as comparações com resultados similares disponíveis na literatura.

As situações de acidentes abordadas nesta análise experimental fazem parte do conjunto de acidentes postulados para a usina nuclear de Angra 2 (Tabela 3, onde encontram-se destacados os acidentes considerados, além da condição de operação normal). Cada condição de operação está caracterizada por uma assinatura temporal de 17 variáveis de processo ao longo de 61 segundos de operação (Tabela 4), e foi simulada por ALVARENGA [138].

Tabela 3 – Situações de operação da usina.

Situação	Descrição simplificada
BLACKOUT	Perda de alimentação elétrica externa
BLACKSEM	Perda de alimentação elétrica sem desligamento do reator
LOCA	Perda de refrigerante do sistema primário
NORMAL	Condição normal de potência
MEFWISO	Isolamento da alimentação principal e auxiliar
MEFWISEM	Isolamento da alimentação principal e auxiliar sem desligamento do reator
MFWBR	Ruptura da alimentação principal
MFWBRSEM	Ruptura da alimentação principal sem desligamento do reator
MFWISEM	Isolamento da alimentação principal sem desligamento do reator
MFWISO	Isolamento da alimentação principal
MSTMISO	Isolamento da linha de vapor principal
SGTR	Ruptura de tubos do gerador de vapor
STMLIBR	Ruptura da linha de vapor principal
TRIPTUR	Desligamento da turbina
TRIPTURSEM	Desligamento da turbina sem desligamento do reator

Tabela 4 – Variáveis de estado.

Numeração	Variável	Unidade
1	Vazão no núcleo	%
2	Temperatura na perna quente	ºC
3	Temperatura na perna fria	ºC
4	Vazão no núcleo	kg/s
5	Nível no gerador de vapor – faixa larga	%
6	Nível no gerador de vapor – faixa estreita	%
7	Pressão no gerador de vapor	MPa
8	Vazão de água de alimentação	kg/s
9	Vazão de vapor	kg/s
10	Vazão na ruptura	kg/s
11	Vazão no circuito primário	kg/s
12	Pressão no sistema primário	MPa
13	Potência térmica	%
14	Potência nuclear	%
15	Margem de subresfriamento	ºC
16	Nível do pressurizador	%
17	Temperatura média no primário	ºC

No conjunto de dados de assinatura temporal, o primeiro segundo corresponde à situação normal de potência e o segundo segundo ao início do trip do reator [12]. Um exemplo do conjunto de dados, aqui exemplificada para a situação de operação BLACKOUT, encontra-se na Figura 5.1. Nota-se que, devido à presença de variáveis com ordens de grandeza distintas, um pré-tratamento necessário ao conjunto de dados é realizar a sua normalização, entre zero e um, da maneira descrita na Equação 5.1:

$$var_{norm} = \frac{var_{nãonorm} - \min(var)}{\max(var) - \min(var)} \tag{5.1}$$

	A	B	C	D	E	F	G	H	I	J	K	
1	vazao %	temp q (C)	temp f (C)	vazao (kg/s)	nivel gv L %	nivel gv E %	P do GV (Mpa)	Vazao Agua (kg/s)	Vazao Vapor (kg/s)	Vazao Rupt. (kg/s)	Vcp (kg/s)	Tempo
2	105,2230	324,4650	291,4560	50418.4	50,0000	62,5153	6,8948	527,7670	527,7670	-	105,2230	
3	105,3900	324,4630	291,4570	50418.3	50,0078	62,5169	6,8941	527,8900	528,3960	-	105,3900	
4	99,4508	325,0090	291,5000	50044.8	49,0423	62,2777	6,9473	34,5845	354,4610	-	99,4663	
5	92,9815	324,3020	291,9100	49535.0	45,4870	61,7591	7,1857	34,0993	152,9940	-	93,0863	
6	87,1439	320,7590	292,3370	49044.1	39,7598	61,2484	7,4117	33,5098	210,3680	-	87,3292	
7	81,9561	316,3150	292,6920	48593.8	34,8187	60,8078	7,5294	33,2219	120,2060	-	82,1691	
8	77,3490	312,0960	292,9300	48191.6	29,3472	60,3384	7,6311	32,9721	98,3528	-	77,5427	
9	73,2299	308,5080	293,1010	47844.3	23,8022	59,8971	7,7236	32,7810	52,4708	-	73,3858	
10	69,5197	305,5930	293,2560	47552.9	18,4795	59,5180	7,7740	32,6529	27,3939	-	69,6409	
11	66,1596	303,2760	293,4140	47310.5	13,5314	59,1576	7,8118	32,5522	68,2616	-	66,2537	
12	63,1044	301,4490	293,5650	47189.2	8,9743	58,7984	7,8267	32,5302	15,6450	-	63,1782	
13	60,3190	300,0600	293,6890	47157.4	4,0336	58,3093	7,8421	32,4943	(9,4187)	-	60,3758	
14	57,7686	298,9920	293,7890	47125.8	0,5529	57,8674	7,8692	32,4322	25,3094	-	57,8116	
15	55,4223	298,1790	293,8800	47094.9	0,7015	57,4781	7,8690	32,4357	1,4646	-	55,4568	
16	53,2591	297,6070	293,9590	47064.6	0,7056	52,1000	7,8694	32,4263	0,7594	-	53,2871	
17	51,2592	297,2250	294,0220	47035.0	0,7136	47,5297	7,8882	32,3966	8,9994	-	51,2812	
18	49,4037	296,9790	294,0770	47006.3	0,7183	46,8936	7,8880	32,3954	(7,6856)	-	49,4212	
19	47,6774	296,8310	294,1260	46978.5	0,7231	47,3437	7,8883	32,3838	9,2459	-	47,6918	
20	46,0676	296,7540	294,1680	46951.6	0,7261	47,7631	7,9021	32,3700	2,5007	-	46,0792	
21	44,5629	296,7270	294,2060	46925.8	0,7313	48,2433	7,9029	32,3641	(10,4109)	-	44,5722	
22	43,1528	296,7380	294,2420	46901.2	0,7392	48,7094	7,9075	32,3450	13,3258	-	43,1605	
23	41,8288	296,7760	294,2770	46877.8	0,7404	49,1304	7,9189	32,3369	(0,6345)	-	41,8351	
24	40,5835	296,8330	294,3100	46855.6	0,7469	49,5968	7,9211	32,3264	(9,4321)	-	40,5886	
25	39,4096	296,9050	294,3430	46834.8	0,7540	50,0367	7,9298	32,3013	14,6993	-	39,4140	
26	38,3012	296,9860	294,3790	46815.5	0,7583	50,4380	7,9399	32,2948	(2,0570)	-	38,3047	
27	37,2532	297,0740	294,4170	46797.8	0,7663	50,8783	7,9439	32,2799	(7,8273)	-	37,2560	
28	36.2603	297.1660	294.4560	46781.7	0.7747	51.2939	7.9558	32.2501	14.5234	-	36.2625	

Figura 5.1 – Exemplo do conjunto de dados para o BLACKOUT.

De maneira sucinta, as etapas realizadas nesta pesquisa experimental, e que serão explicadas separadamente e com mais detalhes mais à frente (Seção 5.3), foram:

1. Foi realizado um pré-tratamento (a normalização, definida anteriormente), devido ao de as variáveis apresentarem ordens de grandeza diferentes entre si;
2. Para cada uma das situações de operação, combinações das assinaturas temporais das 17 variáveis correspondentes da Tabela 4 foram fornecidas ao sistema PG para que este realizasse um processo de regressão simbólica não-linear de múltiplos inputs, onde o target é o tempo correspondente a cada um dos fitness cases. Instrumentado o sistema dessa maneira, o resultado dessa regressão é, com efeito, o classificador para aquela situação de operação considerada;
3. Simultaneamente foi feita uma busca pela parametrização ótima do sistema PG, buscando a melhor relação entre custo computacional e desempenho dos classificadores obtidos pelo sistema;
4. A performance dos classificadores obtidos foi testada através de uma métrica de acerto (ver Seção 5.3.3).

5.2 – Acidentes e Variáveis de Estado Selecionadas Para os Testes Experimentais

Dentre os acidentes listados na Tabela 3, BLACKOUT, LOCA e SGTR foram os selecionados para os testes experimentais, além da condição de operação NORMAL, para avaliação da capacidade do sistema de identificar desvios da situação de funcionamento

corriqueira da usina. A motivação principal para a seleção desses acidentes, que serão descritos mais detalhadamente a seguir[15], foi a existência de trabalhos anteriores similares, com os quais os resultados aqui obtidos poderão ser comparados mais diretamente, mas não há nenhum tipo de limitação a eles no que concerne à metodologia de identificação em si: *a priori*, classificadores poderiam ser obtidos para quaisquer das situações de operação presentes na tabela.

5.2.1 – BLACKOUT

Bastante relevante, este evento refere-se à perda de alimentação externa para operação de equipamentos auxiliares, como as bombas de refrigeração do reator, bombas de água de alimentação e de circulação, diminuindo rapidamente a potência das mesmas e podendo levar a um perigoso superaquecimento do reator. De fato, estima-se que eventos de perda de energia respondam por 51,4% das situações de dano potencial ao núcleo do reator [139].

Usualmente, na ocorrência de um blackout, a usina é levada ao modo de resfriamento de emergência do reator, com potência fornecida por geradores diesel. Neste caso, as bombas de circulação principal são desligadas, o que leva a um aumento na pressão do circuito secundário e consequente atuação de válvulas de descarga de acionamento rápido, além de válvulas de alívio do gerador de vapor (GV), liberando vapor à atmosfera, até que o suprimento de potência auxiliar seja restaurado.

5.2.2 – LOCA

A sigla significa “loss of coolant accident”, e consiste, como o nome sugere, na perda de refrigeração vital para o controle da temperatura do reator, podendo ocasionar um aquecimento excessivo do núcleo e, em situações extremas, derretimento do mesmo. Tal perda de agentes refrigerantes pode ocorrer por uma ruptura nas tubulações de refrigerante do reator ou linhas de conexão, e é classificada, de acordo com o tamanho da ruptura, como LOCA pequeno (SBLOCA), médio (MBLOCA) ou grande (LBLOCA).

Os dados disponíveis e utilizados para os testes experimentais referem-se à simulação de um pequeno LOCA, definido como uma ruptura de aproximadamente 5 cm nas tubulações por onde circulam os agentes refrigerantes do reator, ou linhas conectoras. A principal diferença entre um LOCA pequeno ou grande está nas taxas de descarga de agentes refrigerantes, nas

[15] Para um detalhamento maior das condições de operação da usina e seus possíveis transientes e acidentes, consultar MÓL [118].

variações de pressão ao longo do tempo e na influência predominante de efeitos gravitacionais (pequeno LOCA) ou inerciais (grande LOCA) [140].

Em geral, um pequeno LOCA é caracterizado por um período relativamente longo (de dezenas de minutos à várias horas após o rompimento) durante o qual o sistema primário mantém-se a uma pressão relativamente alta. Tão logo as bombas sejam desativadas, manual ou automaticamente, a força gravitacional domina a vazão e a distribuição do agente refrigerante no sistema primário. Os eventos subsequentes dependem da localização, tamanho e forma do rompimento, além dos comportamentos dos sistemas primário e secundário, que por sua vez são fortemente influenciados por medidas mitigadoras automáticas ou iniciadas pelos operadores.

Normalmente, a resposta do reator em uma situação de pequeno LOCA é consideravelmente mais lenta, quando comparada com os eventos de um grande LOCA. Isso permite dispor de mais tempo e maiores possibilidades de intervenção humana.

5.2.3 – SGTR

Considerado um dos acidentes mais factíveis, a ruptura de tubos dos geradores de vapor (“steam generator tube rupture”) é um evento potencialmente bastante grave que, entre outras coisas, leva à contaminação do circuito secundário através do vazamento de refrigerante radioativo do sistema de resfriamento do reator (SRR) através do tubo do GV rompido. Isso forma um caminho direto para a liberação de conteúdo radioativo à atmosfera, através das válvulas de alívio do sistema secundário.

O SGTR causa a perda de refrigerante primário além da capacidade dos sistemas de controle químicos e de volume (SCQV), e leva a uma despressurização do SRR, com consequente trip (desligamento automático) do reator, por baixa pressão no pressurizador ou sobretemperatura, e atuação do sistema de injeção de segurança (IS). Ao contrário de outros eventos de perda de refrigerante (como o próprio LOCA descrito anteriormente), em geral faz-se necessária a intervenção rápida do operador, com vistas à prevenção da liberação de material radioativo à atmosfera.

Após a atuação do sistema IS, a pressão no SRR tenderá a se estabilizar num valor tal que a vazão no sistema IS se iguala à vazão através do tubo rompido. O operador deve determinar que o acidente ocorreu observando, principalmente, a diferença na vazão de vapor e de água de alimentação (para detecção antes do trip do reator) e o aumento no nível de

radiação no gerador de vapor afetado, e deve atuar no procedimento de recuperação focado em isolar a seção com o tubo rompido e cortar a vazão antes que o nível de água no gerador de vapor suba até o tubo de vapor principal. Deve ser realizado o resfriamento e despressurização do sistema até desativar o sistema IS e o vazamento do sistema primário ao secundário, ao mesmo tempo em que se mantém a segurança da planta, principalmente estabelecendo uma margem suficiente de subresfriamento no SRR.

5.2.4 – Variáveis de estado

Como já descrito na Tabela 4, cada uma das situações de operação considerada (BLACKOUT, LOCA, NORMAL e SGTR) está representada pela evolução temporal (chamada de assinatura) de 17 variáveis, correspondentes aos 61 segundos após o início do evento. As Figuras 5.2, 5.3, 5.4, 5.5 e 5.6 mostram, graficamente, como se comportam essas variáveis em cada uma das 4 circunstâncias, estando em preto o BLACKOUT, em azul o LOCA, em verde o NORMAL e em vermelho o SGTR.

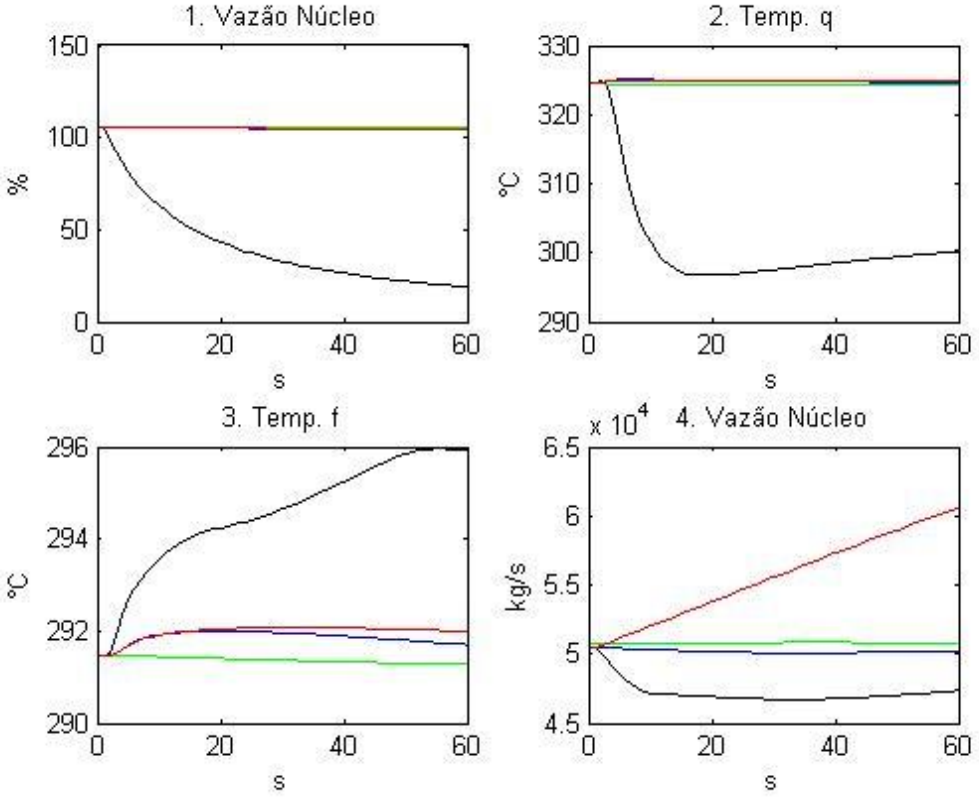


Figura 5.2 – Assinatura das variáveis 1, 2, 3 e 4 para cada uma das situações de operação.

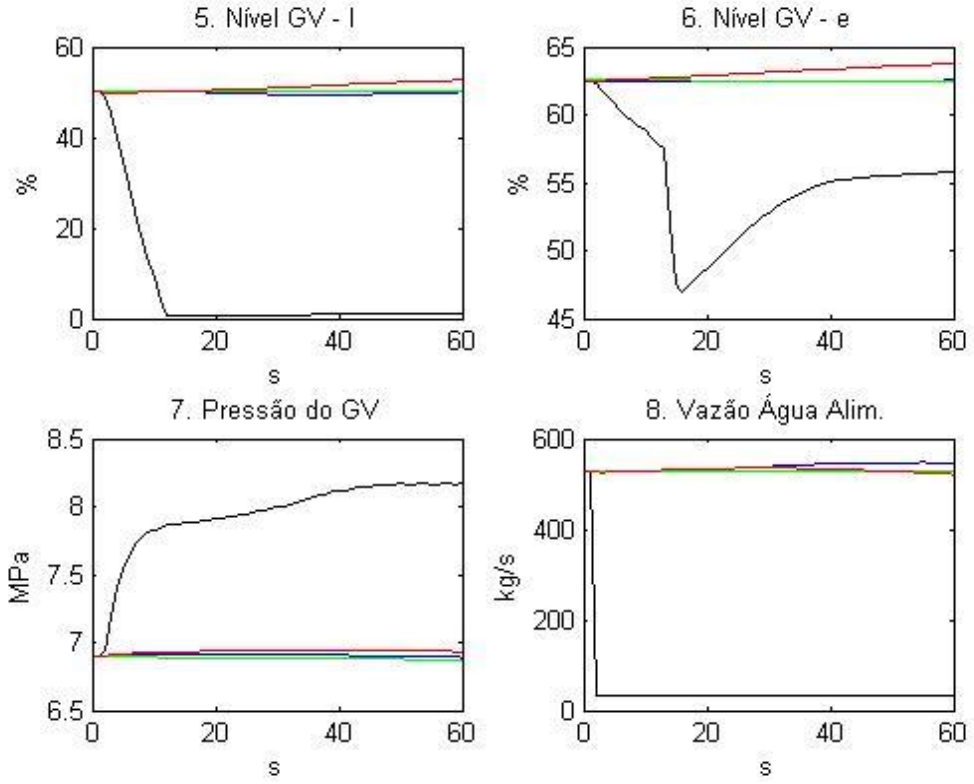


Figura 5.3 – Assinatura das variáveis 5, 6, 7 e 8 para cada uma das situações de operação.

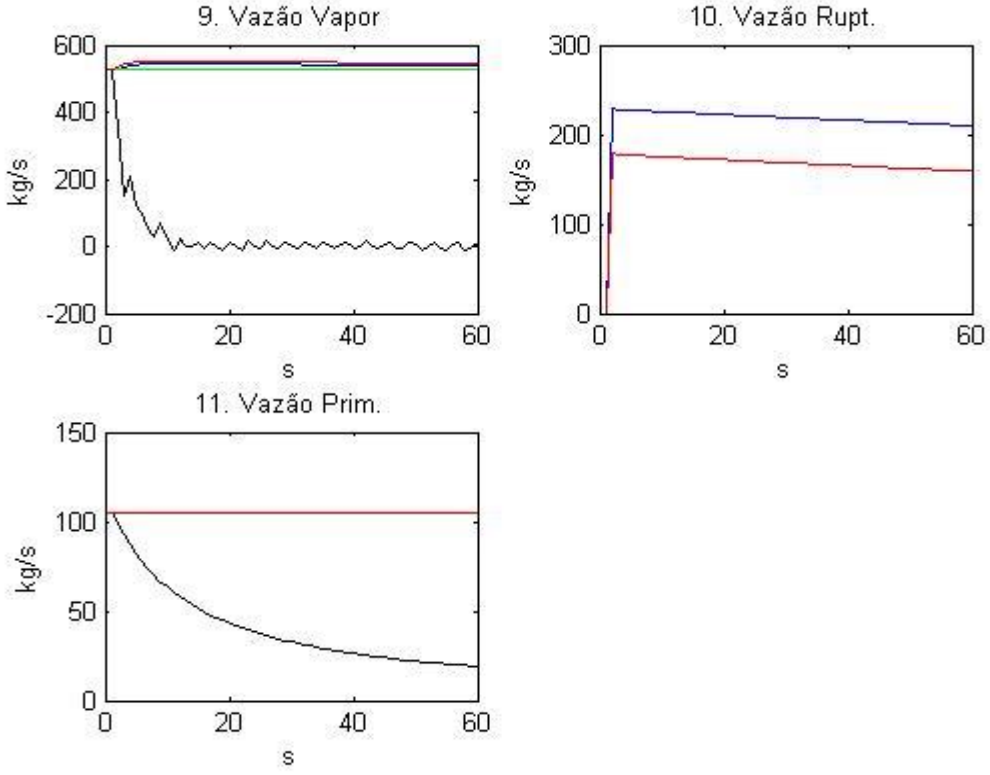


Figura 5.4 – Assinatura das variáveis 9, 10 e 11 para cada uma das situações de operação.

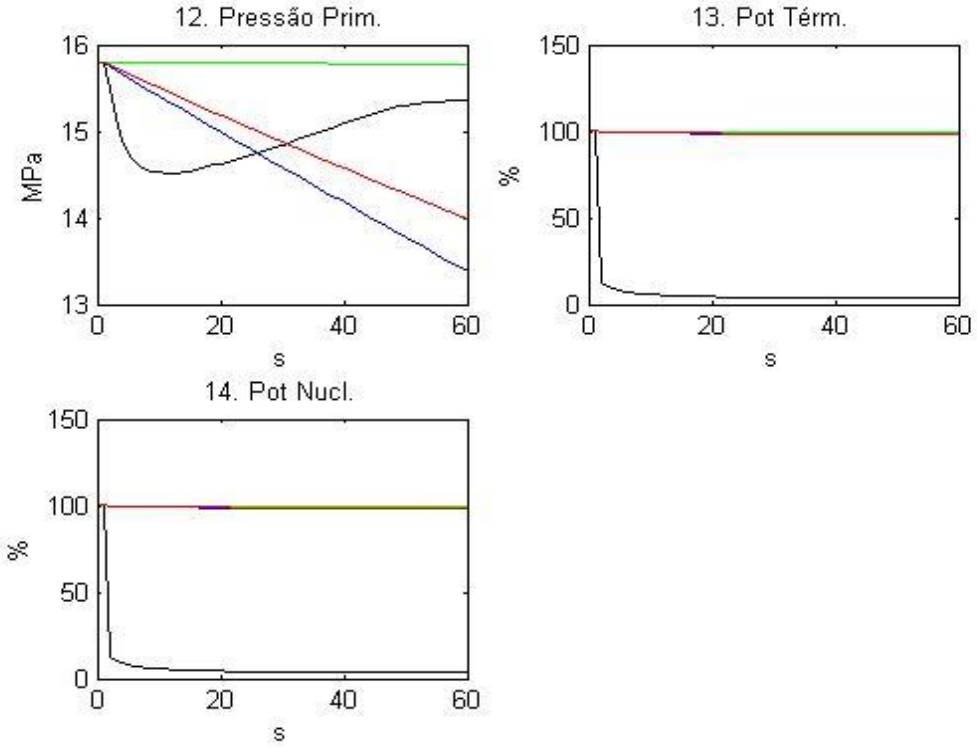


Figura 5.5 – Assinatura das variáveis 12, 13 e 14 para cada uma das situações de operação.

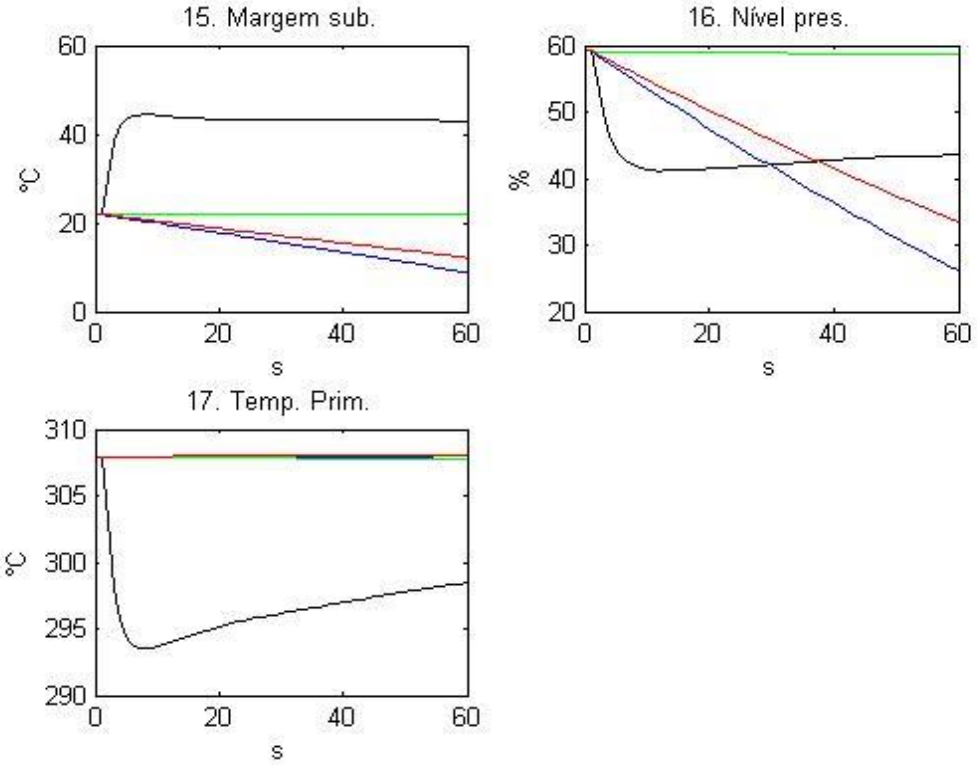


Figura 5.6 – Assinatura das variáveis 15, 16 e 17 para cada uma das situações de operação.

5.3 – Modelagem Experimental e Variáveis Consideradas

5.3.1 – Parâmetros do sistema PG

O sistema PG utilizado nos testes experimentais foi, inicialmente, adaptado da implementação "Little Lisp – Computer Code for Genetic Programming", disponibilizada gratuitamente na internet[16] por John Koza como um complemento ao seu livro de 1992, e executado no programa Allegro CL 10.

Entendendo que o "Little Lisp" continha uma versão ainda muito "primitiva" da programação genética, sem vários dos recursos citados no capítulo 3 e, portanto, bastante limitado, foi utilizada a toolbox para o MATLAB "GPLAB" versão 4, de autoria da pesquisadora Sara Silva, também disponível gratuitamente online[17]. A mesma foi executada no programa MATLAB R2012b e adaptada para este problema, até ser capaz de fornecer os resultados que serão listados no restante deste trabalho.

Após exaustivos testes com diferentes configurações, a combinação ótima de parâmetros (conforme a Seção 3.2.2.8) encontrada – a qual forneceu resultados melhores ou tão bons quanto outras parametrizações mais complexas e mais caras computacionalmente – e aplicada nas análises para a obtenção dos classificadores finais para cada um dos acidentes foi[18]:

- Método de inicialização: ramped half-and-half (Seção 3.2.2.2);

- Set de terminais: "rand"[19] e 0;

- Set de funções: +, -, *, sen, cos;

- Medida de fitness: soma dos erros absolutos entre o output esperado e o valor retornado pelo indivíduo em todos os fitness cases (quanto menor a fitness, mais apto o indivíduo);

- Tamanho da população: 100;

- Número máximo de gerações: 1000;

[16] http://www.genetic-programming.org/gplittlelisp.html
[17] http://gplab.sourceforge.net/
[18] É evidente que a parametrização no GPLAB é um processo bem mais complexo que esse, e a quantidade de critérios a estabelecer é muito maior. Para efeito de verificação e reprodutibilidade dos resultados, a lista completa com os parâmetros fixados nas análises finais deste trabalho encontram-se em anexo.
[19] Número aleatório entre 0 e 1.

- Profundidade máxima das árvores iniciais: 6;

- Profundidade máxima das árvores durante o funcionamento da PG: 17;

- Método de seleção: tournament selection (Seção 3.2.2.3) com pressão parcimoniosa lexicográfica[20] e tamanho de sampling 1% da população;

- Operadores genéticos: cruzamento de troca de subárvore (50%) e mutação de subárvore (50%);

- Critério de parada: número máximo de gerações atingido ou exatidão em todos os fitness cases.

Cabe ressaltar aqui que as execuções do sistema PG foram realizadas em um notebook Lenovo modelo 80UF0000BR, com processador Intel core i3-6100, onde cada corrida do algoritmo tomava cerca de uma hora e meia para alcançar o número máximo de gerações (em nenhum dos casos o critério de parada de exatidão em todos os fitness cases foi atingido).

5.3.2 – Variáveis consideradas para a regressão simbólica

Diferentes combinações das variáveis físicas da Tabela 4 foram avaliadas, na definição de quais seriam escolhidas para a obtenção dos classificadores pelo sistema PG. Os testes foram sendo feitos baseados em trabalhos anteriores, a saber, [133], [134], [135] e [136], além de expertise e conhecimento do comportamento das variáveis ao longo dos acidentes.

Dessa forma, embora outras combinações tenham fornecido resultados satisfatórios, as variáveis consideradas para a obtenção dos regressores que forneceram os resultados que serão descritos na Seção 5.4 foram as variáveis 4, 7, 14 e 16, destacadas na Figura 5.7:

[20] Trata-se de uma adaptação mais moderna da seleção por torneio, que incorpora um controle de bloating ativo: assim como na seleção por torneio, uma quantidade de indivíduos escolhidos aleatoriamente é tomada e aquele com melhor fitness do grupo é selecionado. Porém, se dois indivíduos forem igualmente aptos, aquele com a árvore mais curta é escolhido.

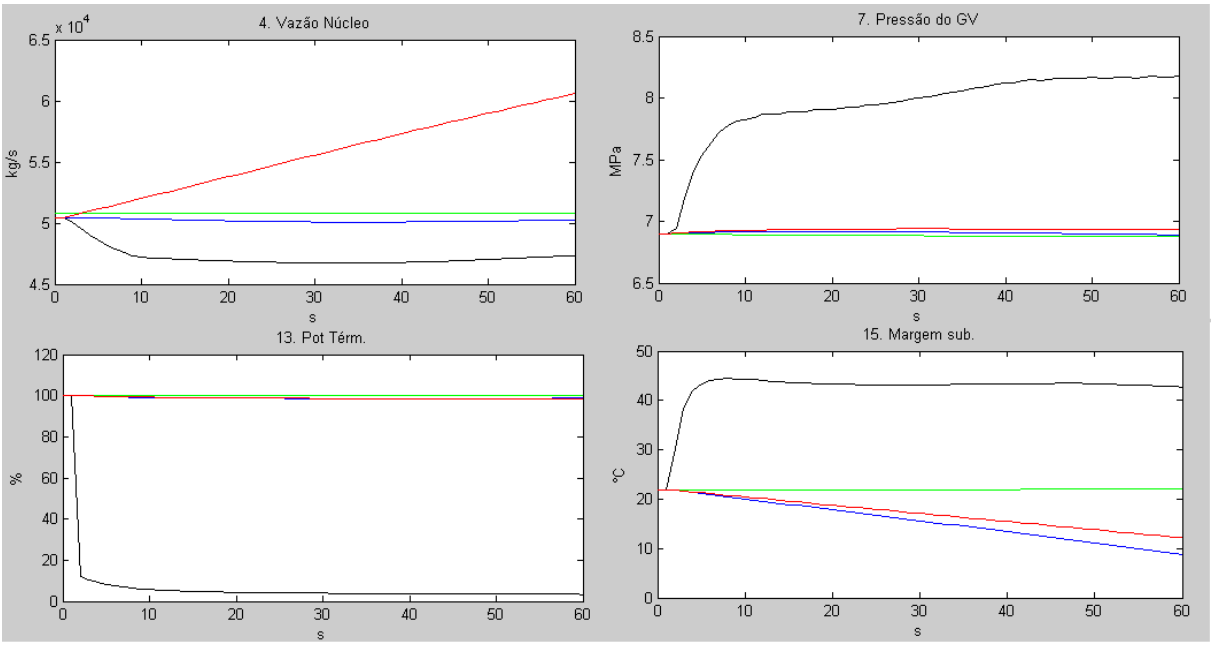


Figura 5.7 – Variáveis consideradas para a obtenção dos classificadores.

5.3.3 – Método experimental

A função do sistema PG, como já descrito, é obter classificadores através do reconhecimento dos padrões de evolução temporal das variáveis, específicos para cada situação de operação. Através de uma análise de regressão simbólica (técnica comumente aplicada em sistemas de RP) onde os inputs são os valores das quatro variáveis consideradas para uma dada situação de operação, e o target o tempo correspondente, a PG obtém automaticamente combinações não lineares das funções e dos terminais (descritos na Seção 5.3.1) e aplica os operadores genéticos ao longo das gerações, de maneira a obter o indivíduo que apresente o menor somatório de erro absoluto entre os outputs (o valor obtido quando da avaliação dos indivíduos) e o target. Este indivíduo mais apto tem sua árvore guardada e é o resultado fornecido pelo sistema PG: o classificador para aquela situação de operação específica.

Obtidos os indivíduos mais aptos para cada acidente e a situação de operação normal, uma métrica da eficácia de cada indivíduo (chamada de "acerto") foi obtida da seguinte maneira:

1. Para obter-se a taxa de acerto do regressor correspondente à situação de operação LOCA, por exemplo, insere-se os dados das variáveis para o LOCA nos regressores de cada uma das condições de operação, obtendo outputs para cada um deles correspondentes a cada target (instante de tempo);

2. Em seguida, compara-se a distância (isto é, diferença absoluta) entre cada um dos outputs obtidos e o target;
3. Caso a distância obtida para a situação de operação considerada (neste caso, o LOCA) com o regressor correspondente a ela seja a menor de todas, considera-se um acerto para o classificador do LOCA, caso contrário, um erro;
4. Repete-se o procedimento para todas as outras situações de operação.

5.4 – Resultados e Análise dos Resultados

Diversos testes e ensaios com diferentes parametrizações e configurações do algoritmo foram realizados, sempre em triplicata. Embora muitas das configurações tenham fornecido resultados satisfatórios, serão listados aqui aqueles obtidos com os parâmetros especificados na Seção 5.3.1, pois foram os que forneceram melhor relação entre desempenho com menor custo computacional.

Aliás, é importante frisar que, como será visto mais adiante, embora as árvores obtidas tenham um número de nós e complexidade considerável (refletindo a complexidade do problema), o cálculo das taxas de acerto, após obtidos os classificadores, é fornecido em frações de segundos, mesmo nos casos dos regressores de maior tamanho (ou seja, com mais nós).

5.4.1 – Resultados

Nas Tabelas 5 (BLACKOUT), 6 (LOCA), 7 (NORMAL) e 8 (SGTR) estão alocados, para cada uma das três análises da triplicata, o indivíduo mais apto obtido pelo sistema PG, a profundidade e tamanho da sua árvore (Seção 3.2.2.1), sua fitness, a taxa de acerto obtida pelo indivíduo (cujo máximo corresponde a acertos em todos os 61 segundos) e em quais segundos ele errou, sendo aquele que obteve a menor distância neste instante representado por “B”, “L”, “N” ou “S”, dependendo da inicial de cada situação de operação.

Tabela 5 – Resultados experimentais para o BLACKOUT.

Análise	Profundidade	Tamanho	Fitness	Acerto (%)	Erros (s)
1	17	308	49,1748	96,7213 (59)	5 (L), 6 (L)
2	17	426	58,0024	91,8033 (56)	6 (L), 7 (L), 8 (L), 9 (S), 10 (S)
3	17	346	47,6051	95,0820 (58)	9 (L), 12 (L), 17 (N)

Tabela 6 – Resultados experimentais para o LOCA.

Análise	Profundidade	Tamanho	Fitness	Acerto (%)	Erros (s)
1	17	253	17,0461	96,7213 (59)	0 (B), 4 (B)
2	17	512	9,5587	98,3607 (60)	0 (B)
3	17	203	17,8854	100 (61)	_____

Tabela 7 – Resultados experimentais para o NORMAL.

Análise	Profundidade	Tamanho	Fitness	Acerto (%)	Erros (s)
1	17	536	21,5085	100 (61)	_____
2	16	116	16,9780	100 (61)	_____
3	17	292	16,1007	100 (61)	_____

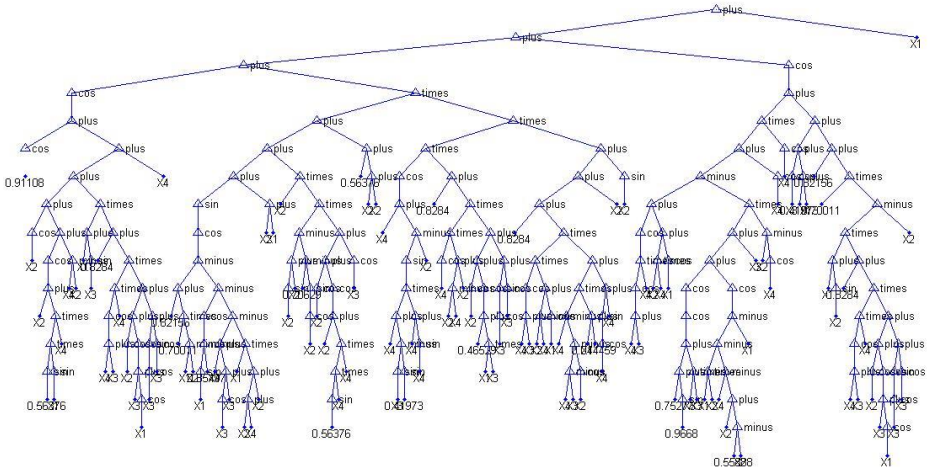
Tabela 8 – Resultados experimentais para o SGTR.

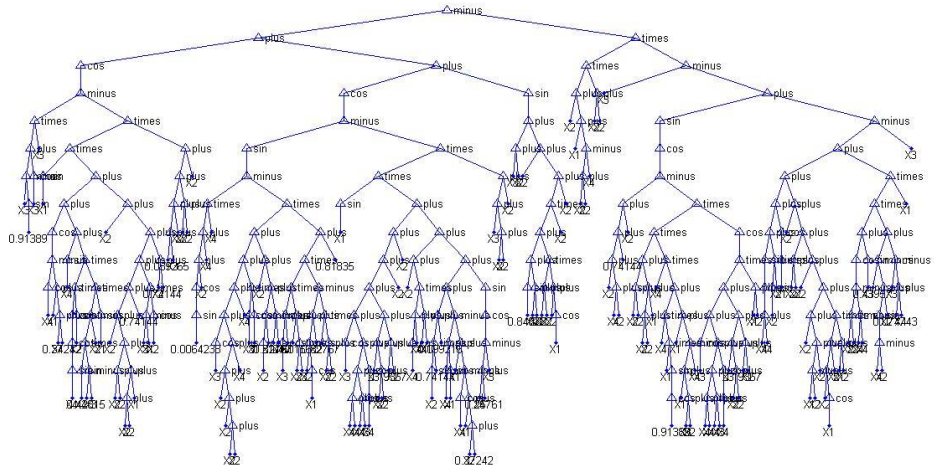
Análise	Profundidade	Tamanho	Fitness	Acerto (%)	Erros (s)
1	17	284	6,4135	93,4426 (57)	24 (L), 25 (L), 26 (L), 27 (L)
2	15	213	6,3361	98,3607 (60)	22 (L)
3	17	277	4,2317	98,3607 (60)	13 (N)

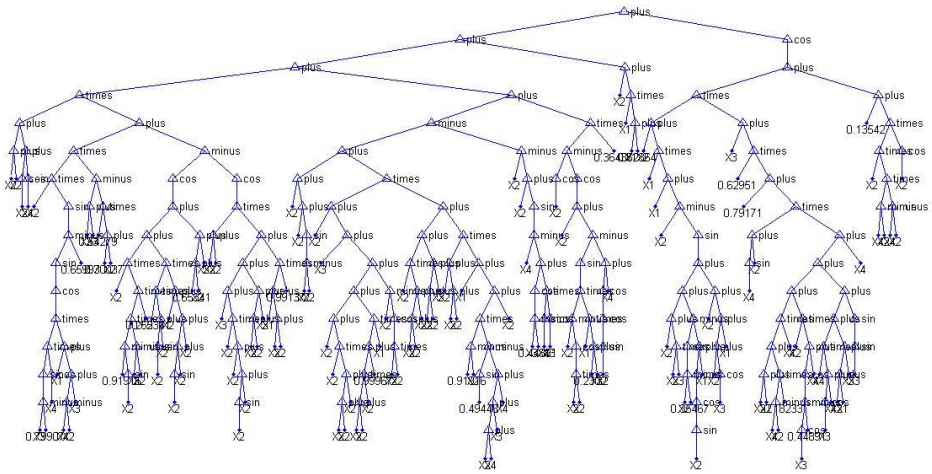
Os classificadores correspondentes às análises da Tabela 5, 6, 7 e 8 encontram-se como árvores de sintaxe respectivamente, na Figura 5.8. A seguir, como ilustração, está a "string" gerada pelo sistema PG como indivíduo mais apto na análise 2 para a condição NORMAL. Na string, X1, X2, X3 e X4 referem-se, respectivamente, às variáveis 4, 7, 13 e 15, e "times", "plus", "minus", "cos" e "sin" às funções já relatadas na Seção 5.3.1. A evolução da fitness do indivíduo mais apto ao longo da execução do sistema PG, para esta análise, encontra-se na Figura 5.9.

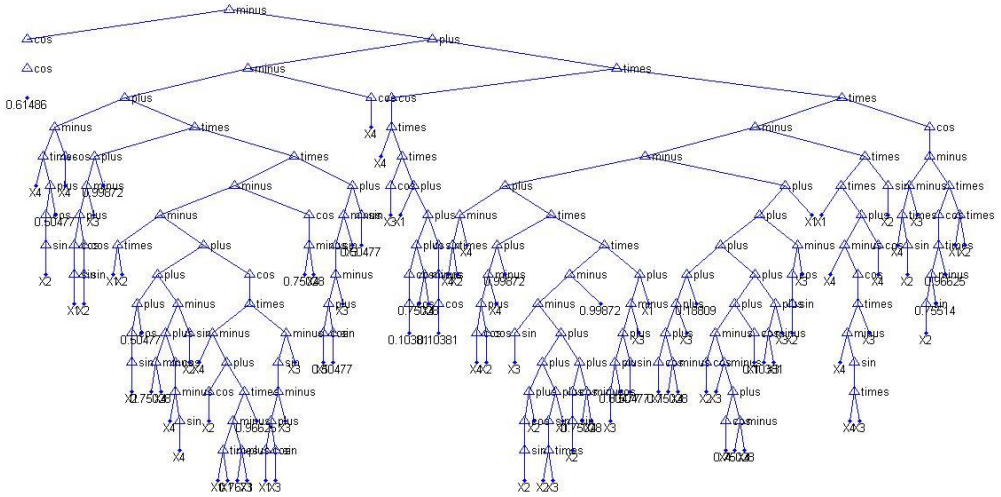
"(times(plus(cos(cos(minus(cos(sin(plus(times(cos(0.47801),X4),sin(plus(X4,X3))))),0.14715
))),cos(X2)),plus(times(plus(cos(times(times(X4,plus(X2,0.87305)),X3)),times(cos(minus(mi
nus(X1,X2),X1)),plus(times(plus(X3,cos(sin(X2))),plus(plus(cos(plus(X4,0.10385)),X4),time
s(0.93224,plus(times(plus(X4,X4),plus(plus(X4,X4),X4)),sin(plus(X2,times(cos(X4),X2))))))

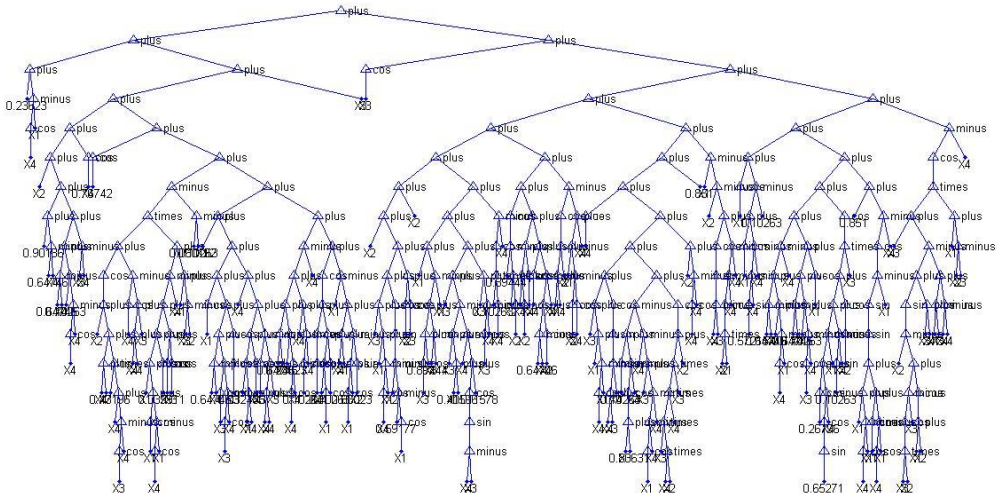
),plus(plus(X4,plus(X3,plus(times(cos(0.47801),X4),minus(plus(cos(sin(times(X3,X2))),sin(times(cos(X4),X4))),X2)))),plus(X4,0.66431))))),plus(X4,cos(plus(plus(minus(X3,minus(X4,sin(X3))),X2),X2)))),plus(X4,plus(X3,plus(0.72483,X3))))))".

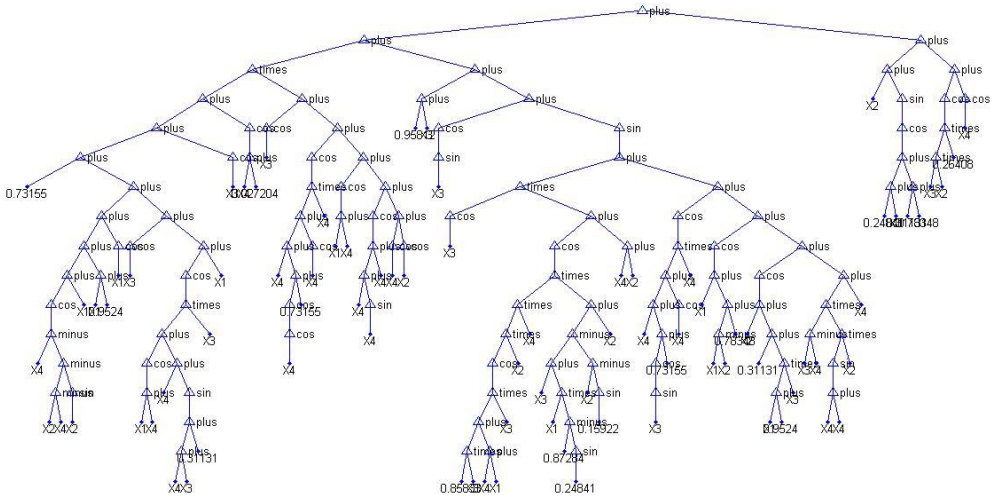


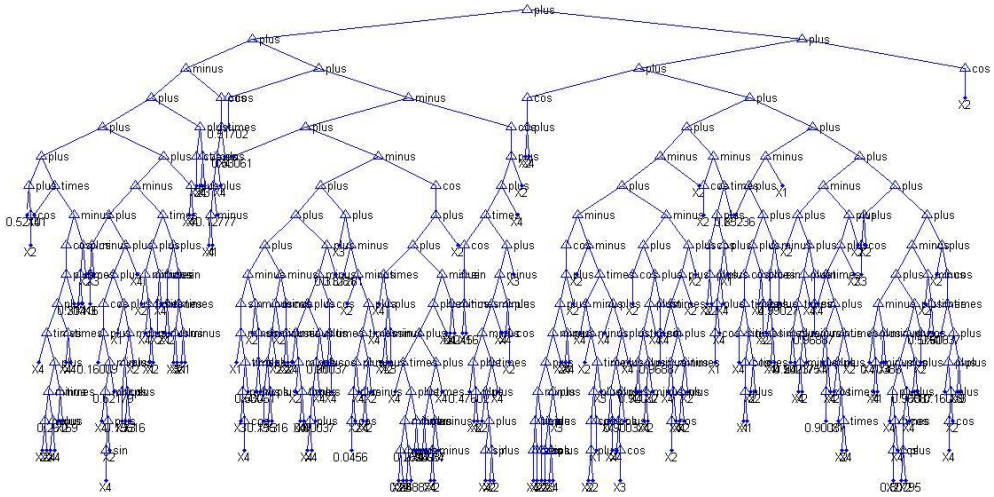


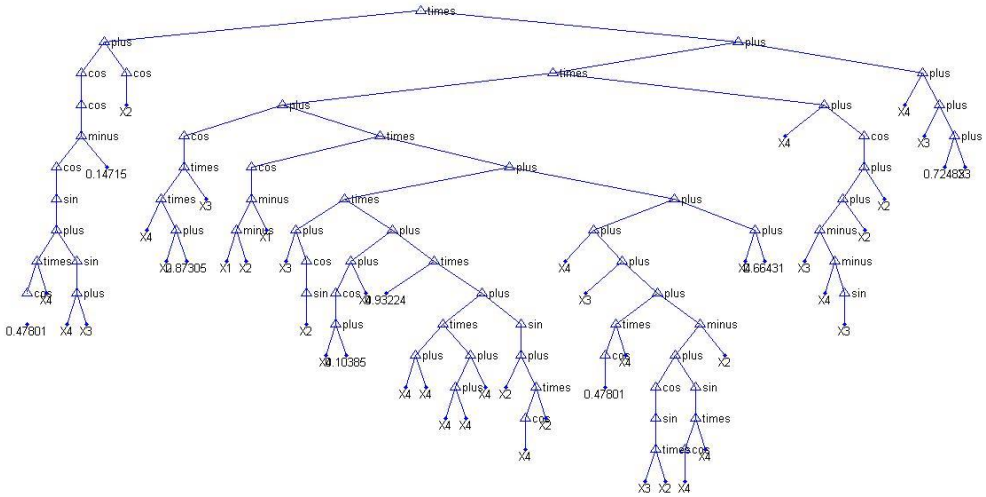


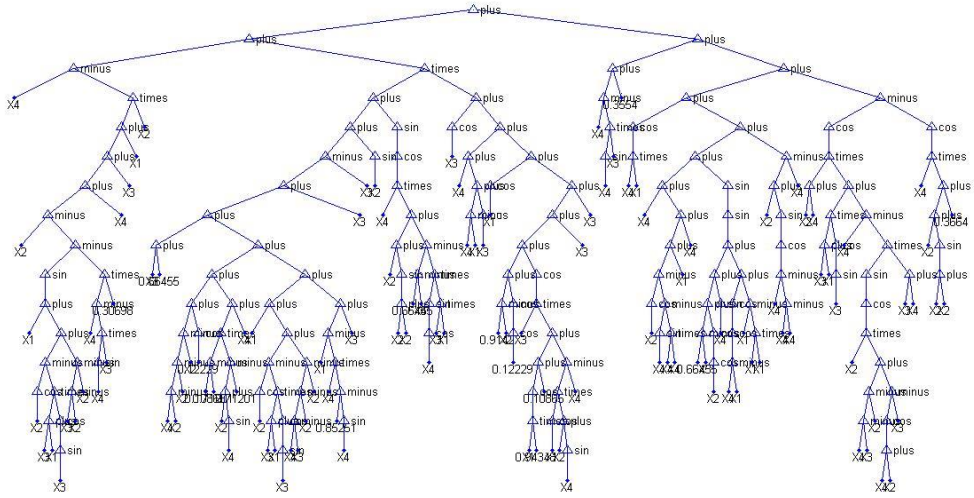












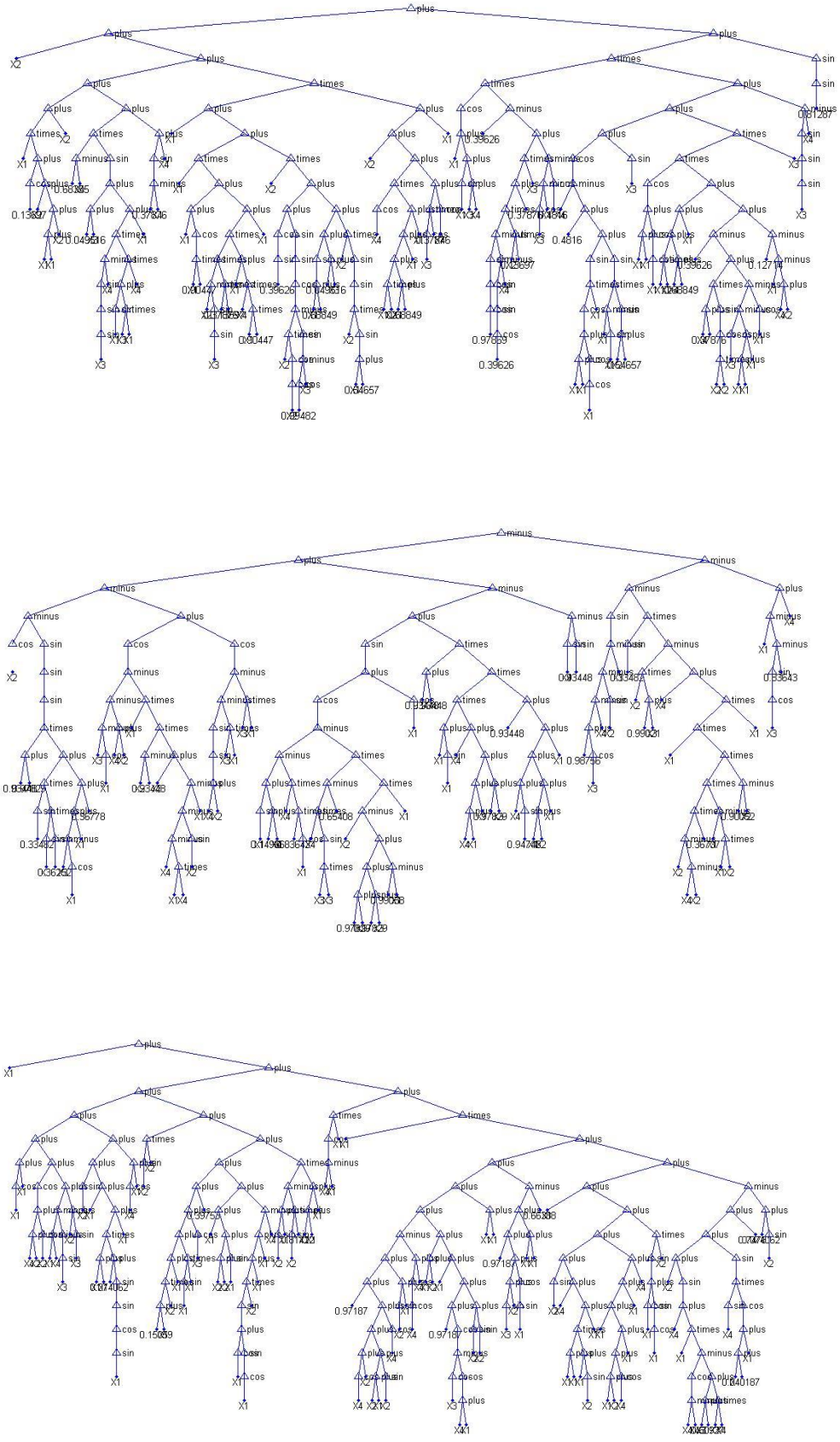


Figura 5.8 – Árvores de sintaxe de cada um dos resultados fornecidos.

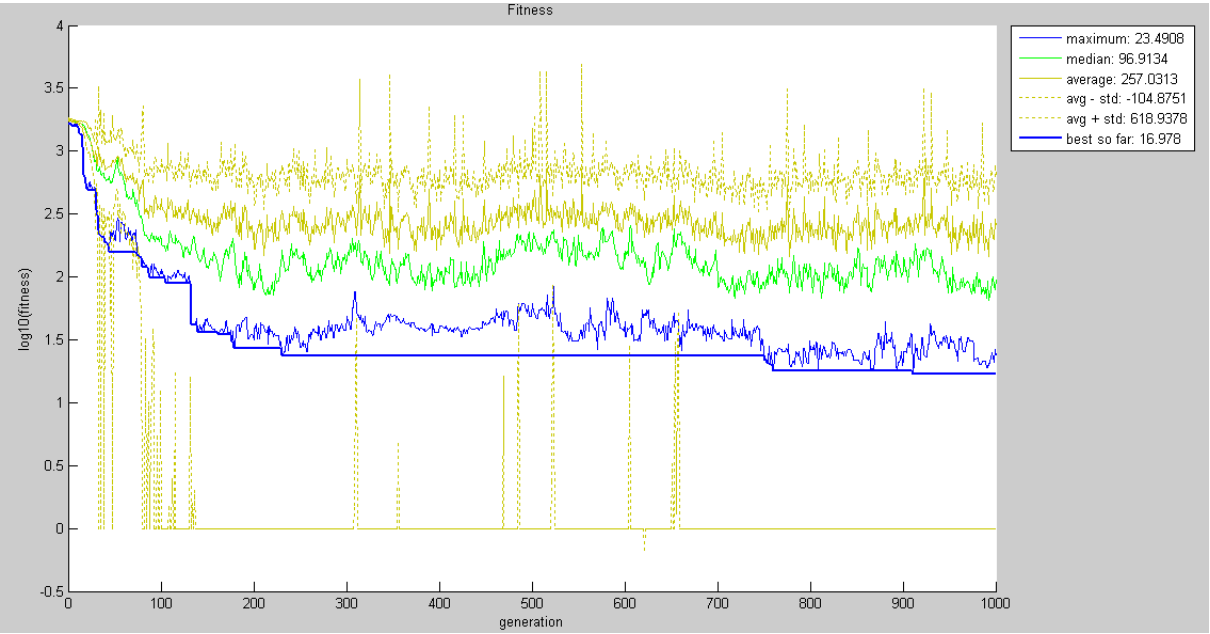


Figura 5.9 – Evolução da fitness na análise 2 para a situação NORMAL.

Ao passo que a string correspondente ao resultado 1 para a condição de operação NORMAL é:

"(plus(plus(minus(plus(plus(plus(plus(0.52101,cos(X2)),times(X4,minus(cos(plus(plus(times(X4,plus(minus(minus(X2,X2),plus(X4,X4)),times(0.26169,X2))),times(X4,X4)),X1)),plus(times(0.30416,X1),X2)))),plus(minus(plus(minus(cos(X3),X3),plus(plus(cos(plus(0.16009,minus(minus(X4,plus(sin(X4),X2)),0.62179))),plus(X1,minus(plus(cos(X3),plus(0.15516,X3)),X2))),X4)),times(plus(minus(X2,times(X4,minus(X1,X2))),times(sin(X2),X4)),plus(cos(times(X1,X2)),sin(times(plus(X3,X2),minus(X1,X1)))))),plus(X4,X4))),plus(cos(X2),times(X4,X3))),cos(times(plus(plus(minus(X4,X1),0.12777),X4),X4))),plus(cos(0.51702),minus(plus(0.53061,minus(plus(plus(plus(minus(sin(minus(X1,times(minus(X3,cos(X4)),X4))),minus(X2,sin(plus(0.53061,X4)))),minus(minus(minus(plus(cos(X3),plus(0.15516,X3)),X2),X2),cos(times(X2,X2)))),plus(minus(plus(minus(X4,minus(X2,times(X4,X4))),plus(plus(plus(plus(X4,X4),0.90037),X4),X1)),cos(times(0.90037,cos(X4)))),0.31281)),plus(X3,minus(0.31281,minus(X2,plus(times(plus(plus(cos(0.0456),X2),X4),minus(minus(X4,X2),X2)),X4)))))),cos(plus(plus(times(X4,plus(minus(X1,X2),sin(X3))),minus(plus(plus(times(X4,plus(minus(minus(X2,X2),plus(X4,0.96887)),times(0.26169,X2))),plus(times(plus(X4,0.90037),minus(minus(X4,X2),X4)),X4)),X2),minus(X4,X2))),X2)))),cos(plus(plus(times(cos(sin(times(0.0456,X1))),plus(minus(minus(minus(plus(0.47602,plus(X1,X2)),times(plus(X2,plus(sin(X4),plus(X2,X2))),X4)),X4),plus(X4,cos(X2))),X3)),X4),X2))))),plus(plus(cos(plus(X2,X4)),plus(plus(minus(plus(min

us(cos(plus(plus(minus(plus(minus(times(cos(X4),cos(X2)),sin(sin(X2))),plus(plus(cos(X2),plus(X3,X4)),X3)),X2),plus(X4,X4)),X2)),minus(X2,times(minus(X4,minus(X2,times(plus(cos(plus(X2,X2)),cos(X1)),X3))),plus(X4,plus(plus(plus(X4,cos(X4)),plus(plus(X4,cos(X3)),0.90037)),X4))))),minus(plus(cos(X2),plus(plus(times(plus(X4,0.90037),minus(minus(X4,X2),X2)),X4),sin(sin(0.96887)))),plus(plus(times(X4,plus(minus(X4,plus(cos(X2),X4)),times(plus(X4,X2),X2))),X2),X2))),X2),minus(cos(X2),times(plus(cos(plus(X2,X2)),plus(plus(plus(cos(X1),X4),X4),X1)),X3))),plus(minus(plus(0.65236,plus(plus(cos(times(sin(X4),times(cos(plus(X1,X1)),times(X2,X2)))),plus(X4,cos(X2))),minus(cos(plus(X2,times(X1,X4))),sin(0.99027)))),X1),plus(minus(plus(X2,plus(plus(times(plus(X4,0.90037),minus(minus(X4,X2),0.21751)),X4),sin(sin(0.96887)))),plus(plus(times(X4,plus(minus(X4,plus(cos(0.90037),X4)),times(plus(X2,times(X1,times(X2,X4))),X2))),X2),X2)),plus(plus(cos(X3),X2),plus(minus(plus(minus(plus(X4,minus(X4,X1)),minus(0.70386,X4)),plus(plus(plus(plus(X4,times(cos(X2),X4)),plus(cos(plus(0.80795,X2)),X4)),plus(0.96887,X1)),cos(X2))),minus(X2,times(0.57516,X4))),plus(X2,cos(times(0.90037,plus(plus(0.16009,plus(cos(X4),X2)),plus(X3,X3)))))))))),cos(X2))))".

5.4.2 – Análise dos resultados

Observando os resultados, há muitos pontos a discutir. Entre os principais:

- Pode-se perceber claramente que não há uma relação direta entre a fitness do indivíduo e a sua performance em taxa de acerto. Isso era de se esperar, tratando-se de um processo estocástico como a PG com elevado grau de não linearidade nos dados e combinações de primitivas;
- A PG parece ter mais facilidade para discriminar algumas situações (como o LOCA e o NORMAL) do que outras. Isso também não é surpresa, já que alguns acidentes apresentam maior grau de similaridade entre si do que outros;
- Analisando os resultados para o BLACKOUT, o sistema apresentou uma certa dificuldade para encontrar um classificador que permitisse discernir este acidente do LOCA entre 5 e 15 segundos. Observando os gráficos da Figura 5.7, nota-se que o BLACKOUT é o evento mais dissimilar aos demais (embora essa diferença de padrão se acentue apenas após os 5 segundos). Portanto, esse resultado não era esperado. Entretanto, vale lembrar que a PG pode ter automaticamente "sacrificado" esses segundos iniciais em favor de um classificador mais eficiente ao longo do restante da evolução temporal, já que devido à grande não linearidade dos regressores obtidos (ver

Seção 5.4.1), não deve-se esperar que, necessariamente, pontos menos similares tenham relação direta com maiores taxas de acerto;

- Ponderações similares podem ser feitas quanto à performance dos classificadores obtidos para o SGTR. Não observa-se, nos gráficos, grande similaridade entre essa situação de operação e o LOCA na região dos 25 segundos, embora o sistema tenha repetidamente fornecido classificadores que os confudem nessa região de tempo, possivelmente pelas mesmas razões discorridas quanto ao BLACKOUT;
- Em todos os acidentes foram obtidas taxas de acerto de, no mínimo, 92%. Para uma técnica que, num computador de uso pessoal de características básicas para intermediárias, leva 1 hora e meia para fornecer cada um dos classificadores em sua fase de treinamento e, uma vez obtido, segundos para identificar a taxa de acerto (e, portanto, fazer um possível diagnóstico ou simplesmente auxiliar o operador na direção de que evento é passível de estar ocorrendo), considera-se que os resultados são compatíveis com os obtidos em estudos anteriores, como [12], [134] e [136].

CAPÍTULO 6

CONCLUSÕES E TRABALHOS FUTUROS

6.1 - Conclusões

Resumidamente, nesta dissertação foi apresentado, pela primeira vez na literatura científica, um estudo da performance do algoritmo de computação evolucionária programação genética como ferramenta de reconhecimento de padrões para diagnóstico de eventos anômalos em usinas nucleares.

A metodologia utilizada neste trabalho consistiu em fornecer ao sistema PG dados de assinaturas temporais de diferentes variáveis correspondentes a certos acidentes especificados e a condição de operação normal. O sistema se encarregou de automaticamente evoluir regressores não lineares de combinações de funções e terminais na procura do classificador que fornecesse a maior quantidade de informação discriminatória entre os eventos operacionais. Agindo, assim, concomitantemente como um seletor e gerador de atributos, usando a terminologia clássica de sistemas de reconhecimento de padrões.

No conjunto de acidentes considerado, o sistema PG foi capaz de fornecer classificadores com taxas de acerto de, no mínimo, 92% e, no máximo 100%. Utilizando os equipamentos e programas citados na Seção 5.3.1 e nos testes com quatro situações de operação (BLACKOUT, LOCA, NORMAL, SGTR) e quatro variáveis realizados, o sistema leva cerca de 6 horas para fazer uma corrida completa, obtendo os quatro classificadores, um para cada evento operacional, concluindo sua fase de treinamento. Isso deixa claro que, em verdade, como dito na Seção 1.1, a complexidade computacional da programação genética já deixou, há tempos, de ser um impedimento à implementação do algoritmo.

Observa-se, avaliando as strings obtidas na Seção 5.4.1, bem como suas árvores correspondentes, que inteligibilidade dos resultados não é um ponto forte do sistema PG. Entretanto, ele compensa isto ao fornecer resultados com grau de confiabilidade elevada (foram realizados cerca de 30 outros testes preliminares em que, consistentemente, a taxa de acerto ficou acima de 90%), em tempo hábil que pode ser ajustado aumentando ou diminuindo o grau de exigência dos parâmetros do algoritmo em si, relativa simplicidade em sua configuração e o fato de não necessitar de pré-tratamento quase nenhum nos dados (tendo sido necessário apenas normalizá-los).

Pelas razões aqui expostas, a programação genética pode ser considerada uma técnica eficaz e competitiva para ser aplicada à obtenção de sistemas de diagnósticos como ferramentas de suporte ao operador em usinas nucleares, tendo fornecido resultados similares, e até superiores, a alguns dos estudos já mencionados anteriormente no capítulo 4, com o adendo de possuir menor complexidade e tratamento nos dados.

6.2 – Possíveis Trabalhos Futuros

O comportamento do sistema PG quando aplicado ao problema tratado nesta dissertação parece sugerir que o algoritmo talvez tenha uma capacidade inerente de "selecionar automaticamente" não só as combinações não lineares de valores das variáveis, como também quais variáveis são mais relevantes (por possuírem maior informação discriminatória) e quais podem ser descartadas na obtenção de classificadores para um conjunto qualquer de acidentes. Isto eliminaria a necessidade do processo exaustivo que consiste em buscar (muitas vezes por tentativa e erro), o conjunto ótimo de variáveis, o qual normalmente não é encontrado, apenas aproximado. Até onde foram as pesquisas deste autor, não foram encontrados trabalhos tratando dessa possibilidade, e provavelmente muitas técnicas mais modernas da programação genética terão que ser incorporadas para abordá-la eficientemente.

Muito embora a quantidade de testes realizada nesta dissertação faça crer que chegou-se a uma combinação de "custo-benefício" muito boa no tocante à configuração do sistema PG, a parametrização ótima do sistema continua sendo um problema em aberto que pode ser abordado em trabalhos futuros. Efetivamente, embora o sistema PG possua bom grau de robustez, com resultados que não são muito dependentes das parametrizações do algoritmo (uma vez bem escolhidos os conjuntos de variáveis), o custo computacional da implementação do sistema mostra-se bastante sensível à variações nos seus parâmetros, de maneira que esta é um ponto onde estudos mais aprofundados seriam de grande valia.

Seria interessante aplicar o sistema proposto a diferentes conjuntos de acidentes e transientes, já que, à princípio, com maiores ou menores ajustes e maior ou menor grau de complexidade, a programação genética pode ser aplicada à quaisquer situações de operação, contanto que se disponha de poder computacional suficiente.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] CARVALHO, P. V. R. et al. A neuro-fuzzy system to support the attention and direction of nuclear power plant operators. **Brazilian Journal of Operations & Production Management**, v.12, p. 2-14, Jun. 2015.

[2] DUDA, R. O. et al. **Pattern Recognition**. Nova Iorque: John Wiley & Sons, 2000.

[3] SHLENS, J. A tutorial on principal component analysis: technical report. **Cornell University Repository**. v. 51, n. 52, p. 65-77, 2002.

[4] KOZA, J. **On the programming of computers by means of natural selection**. Cambridge: MIT Press, 1992.

[5] ASLAM, M. W. **Pattern recognition using genetic programming for classification of diabetes and modulation data**. 2013. 220 f. Tese (Doutorado em Ciências) – Departamento de Engenharia Elétrica e Eletrônica, Universidade de Liverpool, Liverpool, Fev. 2013.

[6] THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern Recognition**. Cambridge: Academic Press, 2003.

[7] BELLMAN, R. E. **Adaptive control processes**: a guided tour. Princeton: Princeton university press, 2015.]

[8] AGGARWAL, C. C. On randomization, public information and the curse of dimensionality. In: ICDE DATA ENGINEERING CONFERENCE, 23., 2007. Istanbul. **Proceedings**... Turquia, 2007. p. 136-145.

[9] KUO, F. Y.; SLOAN, I. H. Lifting the curse of dimensionality. **Notices of the American Mathematical Society**, v. 52, n. 11, p. 1320-1328, 2005.

[10] BACKES, A. **Extração e seleção de atributos** [on-line]. Faculdade de Computação. Universidade Federal de Uberlândia. Out. 2015 [citado em 4 out. 2017]. Disponível em: <http://www.facom.ufu.br/~backes/pgc204/Aula10-SelecaoAtributos.pdf>.

[11] FODOR, I. A survey of dimension reduction techniques: technical report. **Lawrence Livermore National Laboratory**. Maio 2002.

[12] NICOLAU, A. S. **Computação quântica e inteligência de enxames aplicados na identificação de acidentes de uma usina nuclear PWR**. 2010. 91 f. Dissertação (Mestrado em Ciências) – Programa de Engenharia Nuclear, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Fev. 2010.

[13] GUO, H. **Feature generation and dimensionality reduction using genetic programming**. 2009. 198 f. Tese (Doutorado em Ciências) – Departamento de Engenharia Elétrica e Eletrônica, Universidade de Liverpool, 2009.

[14] PRATI, R. C. **Seleção de atributos** [on-line]. Faculdade de Ciência da Computação. Universidade Federal do ABC. Abr. 2010 [citado em 4 out. 2017]. Disponível em: <http://professor.ufabc.edu.br/~ronaldo.prati/DataMining/selecao-de-atributos.pdf>.

[15] JAIN, A. K.; DUIN, R. P.. W.. ; MAO, J. Statistical pattern recognition: a review. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 22, n. 1, p. 4-37, Nov. 1999.

[16] COMPUTER program. In: Wikipédia: a enciclopédia livre. Disponível em: <https://en.wikipedia.org/wiki/Computer_program>. Acesso em 13 fev. 2017.

[17] POLI, R.; LANGDON, W. B.; MCPHEE, N. F. **A field guide to genetic programming**. Lulu. 2008. Disponível em: < http://www.gp-field-guide.org.uk>.

[18] BENTON, J. et al. **Heuristics and search for domain-independent planning** [on-line]. International Conference on Automated Planning and Scheduling, 26., jun. 2016 [citado em 13 fev. 2017]. Disponível em: <http://icaps16.icaps-conference.org/hsdip.html>.

[19] STATEMENT (computer science). In: Wikipédia: a enciclopédia livre. Disponível em: <https://en.wikipedia.org/wiki/Statement_(computer_science)>. Acesso em 13 fev. 2017.

[20] TURING, A. M. Intelligent machinery, a heretical theory. In: COPELAND, B. J. (Ed.). **The essential Turing: seminal writings in computing, logic, philosophy, artificial intelligence, and artificial life, plus the secrets of enigma**. Oxford: Oxford University Press, 2004. p. 395-432.

[21] TURING, A. M. Computing machinery and intelligence . In: INCE, D. C. (Ed.). **Mechanical intelligence: collected works of A.M. Turing**. Amsterdã: North-Holland, 1992. p. 133-160.

[22] HOLLAND, J. H. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. Ann Arbor: University of Michigan press, 1992.

[23] HOLLAND, J. H.; REITMAN, J. S. Cognitive systems based on adaptive algorithms. **Acm Sigart Bulletin**, n. 63, p. 49-61, jun. 1977.

[24] HOLLAND, John H. **Induction: Processes of inference, learning, and discovery**. Cambridge: MIT Press, 1989.

[25] SMITH, S. F. **A learning system based on genetic adaptive algorithms**. 1980. Tese (Doutorado em Ciências) – Universidade de Pittsburgh, Pittsburgh, 1980.

[26] FORSYTH, R. BEAGLE - a darwinian approach to pattern recognition. **Kybernetes**, v. 10, n. 3, p. 159-166, 1981.

[27] CRAMER, N. L.. A representation for the adaptive generation of simple sequential programs. In: FIRST INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 1., 1985, Pittsburgh. **Proceedings**... p. 183-187. 1985.

[28] HICKLIN, J. F. **Application of the genetic algorithm to automatic program generation**. 1986. Tese (Doutorado em Ciências) Universidade de Idaho, Idaho, 1986.

[29] FUJIKI, C. **An evaluation of Holland's genetic algorithm applied to a program generator**. Dissertação (Mestrado em Ciências) – Departamento de Ciência da Computação, Universidade de Idaho, Idaho, l986.

[30] FUJIKI, C. Using the genetic algorithm to generate Lisp source code to solve the prisoner's dilemma. In: SECOND INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 2., 1987, Cambridge. **Proceedings**... 1987.

[31] ANTONISSE, H. J.; KELLER, K. S. Genetic operators for high-level knowledge representations. In: SECOND INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 2., 1987, Cambridge. **Proceedings**... 1987.

[32] BICKEL, A. S.; BICKEL, R. W. Tree structured rules in genetic algorithms. In: SECOND INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 2., 1987, Cambridge. **Proceedings**... 1987.

[33] DEB, K. et al. (Eds.). Lecture notes in computer science (LNCS) 3102-3103. In: GENETIC AND EVOLUTIONARY COMPUTATION – GECCO, 6., 2004, Seattle. **Proceedings**... São Francisco: Morgan Kaufmann, 230 f, 2004.

[34] FIRST ANNUAL CONFERENCE IN GENETIC PROGRAMMING, 1., 1996, Stanford. **Proceedings**... Cambridge: MIT Press, 1996.

[35] GENETIC PROGRAMMING: EUROPEAN CONFERENCE – EUROGP , 8., 2005, Lausanne. **Proceedings**... 2005.

[36] YU, G.; WORZEL, W.; RIOLO, R. (eds.) Genetic programming theory and practice III. In: WORKSHOP FOR THE STUDY OF COMPLEX SYSTEMS, 3., 2005, Michigan. **Proceedings**... Nova Iorque: Springer, 2005.

[37] SECOND ASIAN-PACIFIC WORKSHOP ON GENETIC PROGRAMMING, 1., 2003, Cairns. **Proceedings**... Cairns: CQUniversity Press, 2003.

[38] ABRAHAM, A.; NEDJAH, N.; MOURELLE, L. Evolutionary computation: from genetic algorithms to genetic programming. **Genetic Systems Programming**, v. 13, p. 1-20, 2006.

[39] ABSTRACT syntax tree. In: Wikipédia: a enciclopédia livre. Disponível em: <https://en.wikipedia.org/wiki/Abstract_syntax_tree>. Acesso em 13 fev. 2017.

[40] POLI, R.; MCPHEE, N. F.. General schema theory for genetic programming with subtree-swapping crossover: part I. **Evolutionary Computation**, v. 11, n. 1, p. 53-66, 2003.

[41] KOZA, J. R. et al. **Genetic programming III**: darwinian invention and problem solving. São Francisco: Morgan Kaufmann, 1999.

[42] RATLE, A.; SEBAG, M.. Genetic programming and domain knowledge: beyond the limitations of grammar-guided machine discovery. In: INTERNATIONAL CONFERENCE ON PARALLEL PROBLEM SOLVING FROM NATURE, 6., 2000, Paris. **Proceedings**…Berlin: Springer, 2000. p. 211-220.

[43] HOAI, N. X.; MCKAY, R. I.; ESSAM, D. Representation and structural difficulty in genetic programming. **IEEE Transactions on Evolutionary Computation**, v. 10, n. 2, p. 157-166, 2006.

[44] LANGDON, W. B.; BANZHAF, W. Repeated sequences in linear genetic programming genomes. **Complex Systems**, v. 15, n. 4, p. 285-306, 2005.

[45] POLI, R. Discovery of symbolic, neuro-symbolic and neural networks with parallel distributed genetic programming: technical report. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETS AND GENETIC ALGORITHMS, 3., 1998. Norwich. **Proceedings**... Viena: Springer, 1998. p. 419-423.

[46] TELLER, A. Evolving programmers: the co-evolution of intelligent recombination operators. In: ANGELINE, P. J.; KINNEAR, K. E. (Eds.). **Advances in Genetic Programming**. Cambridge: MIT Press, 1996. p. 45-68.

[47] MILLER, J. F. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In: FIRST ANNUAL CONFERENCE ON GENETIC AND EVOLUTIONARY COMPUTATION, 1., 1999, Orlando. **Proceedings**... São Francisco: Morgan Kaufmann, 1999, v. 2. p. 1135-1142.

[48] VASICEK, Z. Cartesian gp in optimization of combinational circuits with hundreds of inputs and thousands of gates. In: EUROPEAN CONFERENCE ON GENETIC PROGRAMMING, 18., 2015, Copenhagem. **Proceedings**... Cham: Springer, 2015. p. 139-150.

[49] SIKULOVA, Michaela; SEKANINA, Lukas. Acceleration of evolutionary image filter design using coevolution in cartesian gp. In: PARALLEL PROBLEM SOLVING FROM NATURE, 12., 2012, Taormina. **Proceedings**... Berlin: Springer, 2012. p. 163-172.

[50] MILLER, J. F.; MOHID, M. Function optimization using cartesian genetic programming. In: ANNUAL CONFERENCE COMPANION ON GENETIC AND EVOLUTIONARY COMPUTATION, 15., 2013, Amsterdã. **Proceedings**... Nova Iorque: ACM Press 2013. p. 147-148.

[51] ZHANG, B. T.; MUHLENBEIN, H. Evolving optimal neural networks using genetic algorithms with Occam's razor. **Complex Systems**, v. 7, n. 3, p. 199-220, 1993.

[52] ZHANG, B. T.; MÜHLENBEIN, H. Balancing accuracy and parsimony in genetic programming. **Evolutionary Computation**, v. 3, n. 1, p. 17-38, 1995.

[53] ZHANG, B. T.; OHM, P.; MÜHLENBEIN, H. Evolutionary induction of sparse neural trees. **Evolutionary Computation**, v. 5, n. 2, p. 213-236, 1997.

[54] ZHANG, M.; BHOWAN, U. Pixel statistics and program size in genetic programming for object detection: technical report. **Victoria University of Wellington Repository**. Fev. 2004. Disponível em: <http://www.mcs.vuw.ac.nz/comp/Publications/archive/CS-TR-04/CS-TR-04-3.pdf>. 15 p.

[55] O'REILLY, U. M.; HEMBERG, M.. Integrating generative growth and evolutionary computation for form exploration. **Genetic Programming and Evolvable Machines**, v. 8, n. 2, p. 163-186, jun. 2007.

[56] KOZA, J. R. et al. Towards industrial strength automated design of analog electrical circuits by means of genetic programming. In: O'REILLY, U. M. et al. (Eds.). **Genetic**

programming theory and practice II. Nova Iorque: Springer Science & Business Media, 2006. p. 121-138.

[57] LANGDON, W. B.; POLI, R. Better trained ants for genetic programming: technical report. **School of Computer Science Research Reports**. Universidade de Birmingham, Birmingham, abr. 1998.

[58] ROSS, B. J.; ZHU, H. Procedural texture evolution using multi-objective optimization. **New Generation Computing**, v. 22, n. 3, p. 271-293, 2004.

[59] BARLOW, G. J. Design of autonomous navigation controllers for unmanned aerial vehicles using multi-objective genetic programming. **Electrical Engineering Reports Repository**. Universidade do estado da Carolina do Norte, Raleigh, mar. 2004.

[60] ARAUJO, Lourdes. Multiobjective genetic programming for natural language parsing and tagging. In: PARALLEL PROBLEM SOLVING FROM NATURE, 9., 2006, Reiquiavique. **Proceedings**... Verlag: Springer, 2006. p. 433 - 442.

[61] KOTANCHEK, M.; SMITS, G.; VLADISLAVLEVA, E. Pursuing the pareto paradigm tournaments, algorithm variations & ordinal optimization. In RIOLO, R. L. et al. (Eds.). **Genetic Programming Theory and Practice IV**, v. 5. Ann Arbor: Springer, 2006. p. 167-185.

[62] KOZA, J. R. Human-competitive results produced by genetic programming. **Genetic Programming and Evolvable Machines**, v. 11, n. 3-4, p. 251-284, 2010.

[63] KOZA, J. R. et al. **Genetic programming IV:** routine human-competitive machine intelligence. Nova Iorque: Springer Science & Business Media, 2006.

[64] GRUAU, F. **Neural network synthesis using cellular encoding and the genetic algorithm**. 1994. 159 f. Tese (Doutorado em Ciências) – Departamento de Ciência da Computação, Escola Normal Superior de Lion, Lion, França, jan. 1994.

[65] GRUAU, F.; WHITLEY, D. Adding learning to the cellular development of neural networks: evolution and the baldwin effect. **Evolutionary Computation**, v. 1, n. 3, p. 213-233, 1993.

[66] KOZA, J. R. et al. Use of automatically defined functions and architecture-altering operations in automated circuit synthesis with genetic programming. In: ANNUAL CONFERENCE ON GENETIC PROGRAMMING, 1., 1996, Stanford. **Proceedings**... Cambridge: MIT Press, 1996. p. 132-140.

[67] KOZA, J. R. et al. Automated wywiwyg design of both the topology and component values of electrical circuits using genetic programming. In: ANNUAL CONFERENCE ON GENETIC PROGRAMMING, 1., 1996, Stanford. **Proceedings**... Cambridge: MIT Press, 1996. p. 123-131.

[68] SALUSTOWICZ, R; SCHMIDHUBER, J. Probabilistic incremental program evolution. **Evolutionary Computation**, v. 5, n. 2, p. 123-141, 1997.

[69] LANGDON, W. B. **Genetic programming and data structures**. 1996. 350 f. Tese (Doutorado em Ciências) – Departamento de Ciência da Computação, Universidade de Londres, Londres, 1996.

[70] ANDRE, D.; KOZA, J. R. A parallel implementation of genetic programming that achieves super-linear performance. **Information Sciences**, v. 106, n. 3-4, p. 201-218, 1998.

[71] POLI, R. et al. Theoretical results in genetic programming: the next ten years?. **Genetic Programming and Evolvable Machines**, v. 11, n. 3-4, p. 285-320, 2010.

[72] STEPHENS, C.R.; WAELBROECK, H. Effective degrees of freedom in genetic algorithms. **Physical Review E**, v. 57, n. 3, p. 3251, 1998.

[73] STEPHENS, C.; WAELBROECK, H. Schemata evolution and building blocks. **Evolutionary Computation**, v. 7, n. 2, p. 109-124, 1999.

[74] DAVIS, T. E.; PRINCIPE, J. C. A Markov chain framework for the simple genetic algorithm. **Evolutionary Computation**, v. 1, n. 3, p. 269-288, 1993.

[75] POLI, R.; MCPHEE, N. F.; ROWE, J. E. Exact schema theory and markov chain models for genetic programming and variable-length genetic algorithms with homologous crossover. **Genetic Programming and Evolvable Machines**, v. 5, n. 1, p. 31-70, 2004.

[76] MITAVSKIY, B.; ROWE, J. Some results about the Markov chains associated to GPs and general EAs. **Theoretical Computer Science**, v. 361, n. 1, p. 72-110, 2006.

[77] LANGDON, W. B.; POLI, R. **Foundations of genetic programming**. Nova Iorque: Springer Science & Business Media, 2013.

[78] LANGDON, William B. Convergence rates for the distribution of program outputs. In: GENETIC AND EVOLUTIONARY COMPUTATION – GECCO, 4., 2002, Nova Iorque. **Proceedings**... São Francisco: Morgan Kaufmann, p. 812-819, 2002.

[79] LANGDON, W. B. How many good programs are there? How long are they?. In: FOUNDATIONS OF GENETIC PROGRAMMING, 7., 2003, Cidade do México. **Proceedings**... Nova Iorque: Morgan Kaufmann, p. 183-202, 2003.

[80] LANGDON, W. B. The distribution of reversible functions is Normal. **Genetic Programming Series**, v. 6, p. 173-188, 2003.

[81] LANGDON, W. B. Convergence of program fitness landscapes. In: GENETIC AND EVOLUTIONARY COMPUTATION – GECCO, 5., 2003, Chicago. **Proceedings**... Berlin: Springer, p. 1702-1714, 2003.

[82] LANGDON, W. B. The distribution of amorphous computer outputs. In: THE GRAND CHALLENGE IN NON-CLASSICAL COMPUTATION: INTERNATIONAL WORKSHOP, 1., 2005, Iorque. **Proceedings**... disponível em: < http://discovery.ucl.ac.uk/id/eprint/483>. 2003.

[83] WOODWARD, J. R. Evolving turing complete representations. In: CONGRESS ON EVOLUTIONARY COMPUTATION, 2003, Canberra. **Proceedings**... p. 830-837, 2003.

[84] MCPHEE, N.; OHS, B.; HUTCHISON, T. Semantic building blocks in genetic programming. In: EUROPEAN CONFERENCE ON GENETIC PROGRAMMING, 11., 2008, Nápoles. **Proceedings**... Berlin: Springer, p. 134-145, 2008.

[85] WOODWARD, J. R.; BAI, R. Canonical representation genetic programming. In: FIRST ACM/SIGEVO SUMMIT ON GENETIC AND EVOLUTIONARY COMPUTATION, 1., 2009, Xangai. **Proceedings**... Nova Iorque: ACM Press, p. 585-592, 2009.

[86] RUDOLPH, G. Convergence analysis of canonical genetic algorithms. **IEEE transactions on neural networks**, v. 5, n. 1, p. 96-101, 1994.

[87] RUDOLPH, G. Convergence of evolutionary algorithms in general search spaces. In: INTERNATIONAL CONFERENCE ON EVOLUTIONARY COMPUTATION, 1996. **Proceedings**... 1996.

[88] SCHMITT, L. M.; DROSTE, S. Convergence to global optima for genetic programming systems with dynamically scaled operators. In: GENETIC AND EVOLUTIONARY

COMPUTATION – GECCO, 8., 2006, Seattle. **Proceedings**... Nova Iorque: ACM Press, p. 879-886, 2006.

[89] NORDIN, P.; FRANCONE, F.; BANZHAF, W. Explicitly defined introns and destructive crossover in genetic programming. In: WORKSHOP ON GENETIC PROGRAMMING: FROM THEORY TO REAL-WORLD APPLICATIONS, 1., .1995, Nova Iorque. **Proceedings**... Nova Iorque: University of Rochester Press, p. 6-22, 1995.

[90] SOULE, T.; FOSTER, J. A. Removal bias: a new cause of code growth in tree based evolutionary programming. In: INTERNATIONAL CONFERENCE ON EVOLUTIONARY COMPUTATION, 1998, Anchorage. **Proceedings**... Nova Jersei: IEEE Press, p. 781-786, 1998.

[91] MCPHEE, Nicholas Freitag; MILLER, Justin Darwin. Accurate replication in genetic programming. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 6., 1995, Pittsburgh. **Proceedings**... Nova Iorque: Kaufmann, p. 303-309, 1995.

[92] LANGDON, W. B.; POLI, R. Fitness causes bloat. In: CHAWDRY, P. K.; ROY, R.; PANT, R. K. (Eds.). **Soft computing in engineering design and manufacturing**. Londres: Springer, 1998. p. 13-22.

[93] BANZHAF, W.; LANGDON, W. B. Some considerations on the reason for bloat. **Genetic Programming and Evolvable Machines**, v. 3, n. 1, p. 81-91, 2002.

[94] ROSCA, J. A probabilistic model of size drift. **Genetic Programming Series,** v. 6, p. 119-136, 2003.

[95] PRICE, G. R. Selection and covariance. **Nature**, v. 227, p. 520-521, 1970.

[96] POLI, R.; LANGDON, W. B.; DIGNUM, S. On the limiting distribution of program sizes in tree-based genetic programming. In: EUROPEAN CONFERENCE ON GENETIC PROGRAMMING, 10., 2007, Valência. **Proceedings**... Berlin: Springer, 2007. p. 193-204.

[97] LANGDON, W. B. Size fair and homologous tree crossovers for tree genetic programming. **Genetic Programming and Evolvable Machines**, v. 1, n. 1-2, p. 95-119, 2000.

[98] LEE, Y. S.; TONG, L. I. Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming. **Knowledge-Based Systems**, v. 24, n. 1, p. 66-72, 2011.

[99] BOJARCZUK, C. C. **Programação genética com restrições para descoberta de conhecimento em base de dados médicos**. 2001. 102 f. Dissertação (Mestrado em Ciências) – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Centro Federal de Educação Tecnológica do Paraná, Paraná, 2001.

[100] BRAMEIER, M.; BANZHAF, W. A comparison of linear genetic programming and neural networks in medical data mining. **IEEE Transactions on Evolutionary Computation**, v. 5, n. 1, p. 17-26, 2001.

[101] NEELY, C.; WELLER, P.; DITTMAR, R. Is technical analysis in the foreign exchange market profitable? A genetic programming approach. **Journal of Financial and Quantitative Analysis**, v. 32, n. 4, p. 405-426, 1997.

[102] CHEN, S. H.; YEH, C. H. Using genetic programming to model volatility in financial time series: the cases of nikkei 225 and s&p 500. In: ANNUAL CONFERENCE ON GENETIC PROGRAMMING, 2., 1997, Stanford. **Proceedings**... São Francisco: Morgan Kaufmann, p. 58-63, 1997.

[103] ABOUDAN, M. A. Genetic programming prediction of stock prices. **Computational Economics**, v. 16, n. 3, p. 207-236, 2000.

[104] GUVEN, A. Linear genetic programming for time-series modelling of daily flow rate. **Journal of Earth System Science**, v. 118, n. 2, p. 137-146, 2009.

[105] TAKAC, A. Cellular genetic programming algorithm applied to classification task. Neural Network World, v. 14, p. 435-452, 2004.

[106] SOUZA, L. V. **Programação genética e combinação de preditores para previsão de séries temporais**. Tese (Doutorado em Ciências) — Departamento de Informática, Universidade Federal do Paraná, Paran2006.

[107] KOZA, J. R. Human-competitive results produced by genetic programming. **Genetic Programming and Evolvable Machines**, v. 11, n. 3-4, p. 251-284, 2010.

[108] KEANE, M. A.; KOZA, J. R.; STREETER, M. J. Automatic synthesis using genetic programming of an improved general-purpose controller for industrially representative plants. In: NASA / DOD WORKSHOP ON EVOLVABLE HARDWARE, 4., 2004, Alexandria. **Proceedings**... Califórnia: IEEE, 113-122, 2005.

[109] ISERMANN, R.; BALLE, P. Trends in the application of model-based fault detection and diagnosis of technical processes. **Control Engineering Practice**, v. 5, n. 5, p. 709-719, 1997.

[110] MA, J.; JIANG, J. Applications of fault detection and diagnosis methods in nuclear power plants: a review. **Progress in Nuclear Energy**, v. 53, n. 3, p. 255-266, 2011.

[111] GALBALLY, J.; GALBALLY, D. A pattern recognition approach based on DTW for automatic transient identification in nuclear power plants. **Annals of Nuclear Energy**, v. 81, p. 287-300, 2015.

[112] LIN, T. H. et al. Feature extraction and sensor selection for NPP initiating event identification. **Annals of Nuclear Energy**, v. 103, p. 384-392, 2017.

[113] LIND, M.; ZHANG, X. Functional modelling for fault diagnosis and its application for npp. **Nuclear Engineering and Technology**, v. 46, n. 6, p. 753-772, 2014.

[114] WU, S. C. et al. Multivariate algorithms for initiating event detection and identification in nuclear power plants. **Annals of Nuclear Energy**, v. 111, p. 127-135, 2018.

[115] LIN, C.; CHANG, H. J.. Identification of pressurized water reactor transient using template matching. **Annals of Nuclear Energy**, v. 38, n. 7, p. 1662-1666, 2011.

[116] MA, J.; JIANG, J. Semisupervised classification for fault diagnosis in nuclear power plants. **Nuclear Engineering and Technology**, v. 47, n. 2, p. 176-186, 2015.

[117] GOFUKU, A. Integrated diagnostic technique for nuclear power plants. **Nuclear Engineering and Technology**, v. 46, n. 6, p. 725-736, 2014.

[118] MÓL, A. C. A. **Um sistema de identificação de transientes com inclusão de ruídos e indicação de eventos desconhecidos**. 2002. 115 f. Tese (Doutorado em Ciências) – Programa de Engenharia Nuclear, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Dez. 2002.

[119] SANTOSH, T. V. et al. Application of artificial neural networks to nuclear power plant transient diagnosis. **Reliability Engineering & System Safety**, v. 92, n. 10, p. 1468-1472, 2007.

[120] UHRIG, R. E.; TSOUKALAS, L. H. Soft computing technologies in nuclear engineering applications. **Progress in Nuclear Energy**, v. 34, n. 1, p. 13-75, 1999.

[121] UHRIG, R. E.; HINES, J. Computational intelligence in nuclear engineering. **Nuclear Engineering and Technology**, v. 37, n. 2, p. 127-138, 2005.

[122] BARTAL, Y.; LIN, J.; UHRIG, R. E. Nuclear power plant transient diagnostics using artificial neural networks that allow “don’t-know” classifications. **Nuclear Technology**, v. 110, n. 3, p. 436-449, 1995.

[123] MÓL, A.C. A.; MARTINEZ, A. S.; SCHIRRU, R. A. New approach for transient identification with "Don’t Know" response using neural networks. In: RUAN, D. (Ed.). **Power plant surveillance and diagnostics: applied research and diagnostics**. Berlin: Springer, 2010. p. 253-272.

[124] EMBRECHTS, M. J.; BENEDEK, S. Hybrid identification of nuclear power plant transients with artificial neural networks. **IEEE Transactions on Industrial Electronics**, v. 51, n. 3, p. 686-693, 2004.

[125] MOSHKBAR-BAKHSHAYESH, K.; GHOFRANI, M. B. Transient identification in nuclear power plants: a review. **Progress in Nuclear Energy**, v. 67, p. 23-32, 2013.

[126] ROVERSO, D. On-line early fault detection and diagnosis with the alladin transient classifier. In: AMERICAN NUCLEAR SOCIETY INTERNATIONAL TOPICAL MEETING ON NUCLEAR PLANT INSTRUMENTATION, CONTROL AND HUMAN-MACHINE INTERFACE TECHNOLOGIES, 4., 2004, Colúmbia. **Proceedings**... Illinóis: ANS, p. 19-22, 2004.

[127] WANG, T. et al. A similarity based prognostic approach for remaining useful life estimation of engineered systems. In: INTERNATIONAL CONFERENCE ON

PROGNOSTICS AND HEALTH MANAGEMENT, 1., 2008, Denver. **Proceedings**... Califórnia: IEEE, p. 1-6, 2008.

[128] ZIO, E.; MAIO, F. D.; STASI, M. A data-driven approach for predicting failure scenarios in nuclear systems. **Annals of Nuclear Energy**, v. 37, n. 4, p. 482-491, 2010.

[129] PRUSTY, M. R. et al. Performance analysis of fuzzy rule based classification system for transient identification in nuclear power plant. **Annals of Nuclear Energy**, v. 76, p. 63-74, 2015.

[130] EVSUKOFF, A.; GENTIL, S. Recurrent neuro-fuzzy system for fault detection and isolation in nuclear reactors. **Advanced Engineering Informatics**, v. 19, n. 1, p. 55-66, 2005.

[131] COSTA, R. G. et al. An efficient neuro-fuzzy approach to nuclear power plant transient identification. **Annals of Nuclear Energy**, v. 38, n. 6, p. 1418-1426, 2011.

[132] YANGPING, Z.; BINGQUAN, Z.; DONGXIN, W. Application of genetic algorithms to fault diagnosis in nuclear power plants. **Reliability Engineering & System Safety**, v. 67, n. 2, p. 153-160, 2000.

[133] ALMEIDA, J C. S.; SCHIRRU, R.; PEREIRA, C. M. N. A. A possibilistic approach for transient identification with'don't know'response capability optimized by genetic algorithm. In: INTERNATIONAL NUCLEAR ATLANTIC CONFERENCE, 1., 2002, Rio de Janeiro. **Proceedings**... Rio de Janeiro: ABEN, 2002.

[134] NICOLAU, A. S.; SCHIRRU, R.; LIMA, A. M. M. Accident identification system with automatic detection of abnormal condition using quantum computation. In: INTERNATIONAL CONFERENCE ON MATHEMATICS AND COMPUTATIONAL METHODS APPLIED TO NUCLEAR SCIENCE AND ENGINEERING, Rio de Janeiro, 2011. **Proceedings**... Rio de Janeiro: LAS/ANS, 2011.

[135] NICOLAU, A. S.; SCHIRRU, R. A new methodology for diagnosis system with “Don’t Know” response for Nuclear Power Plants. **Annals of Nuclear Energy**, v. 100, p. 91-97, 2017.

[136] MEDEIROS, J. A. C. C.; SCHIRRU, R. Identification of nuclear power plant transients using the Particle Swarm Optimization algorithm. **Annals of Nuclear Energy**, v. 35, n. 4, p. 576-582, 2008.

[137] NICOLAU, A. S.; SCHIRRU, R. QDPSO and minkowski distance applied to transient diagnosis system. In: INTERNATIONAL CONFERENCE ON AGENTES AND ARTIFICIAL INTELLIGENCE, 6., 2014, Angers. **Proceedings**... Springer International Publishing, p. 611-616, 2014.

[138] ALVARENGA, M. A. B. **Diagnóstico do desligamento de um reator nuclear através de técnicas avançadas de inteligência artificial**. 1997. Tese (Doutorado em Ciências) – Programa de Engenharia Nuclear, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1997.

[139] AMINOV, R. Z.; YURIN, V. E. Nuclear power plant safety improvement based on hydrogen technologies. **Nuclear Energy and Technology**, v. 1, n. 1, p. 77-81, 2015.

[140] LEWIS, M. J. et al. Thermohydraulics of emergency core cooling in light water reactors-a state of the art report. **CSNI Report**, n. 161, 1989.

ANEXO

PARÂMETROS DO GPLAB

maxresources: []

functions: {‘plus’ ‘minus’ ‘times’ ‘sin’ ‘cos’ }

terminals: {'rand' [0]}

numvars: 4

autovars: 1

initpoptype: 'rampedinit'

expected: 'rank85'

elitism: 'replace'

survival: 'fixedpopsize'

dynamicresources: '0'

resourcespopsize: 'steady'

resourcesfitness: 'normal_accept'

periode: 1

ajout: 'M1'

sampling: 'lexictour'

drawperspin: 2147483647

calcfitness: 'regfitness'

adjustfitness: []

precision: 12

lowerisbetter: 1

gengap: 100

savetofile: 'never'

operatorprobstype: 'fixed'

initialprobstype: 'fixed'

reproduction: 0.1000

operatornames: {'crossover' 'mutation'}

operatornparents: [2 1]

operatornchildren: [2 1]

initialfixedprobs: [0.5000 0.5000]

initialvarprobs: [0.5000 0.5000]

homocrosstype: 'homocross'

pointmutationtype: '1point'

broodpairs: 5

gde: 0

gdeF: 0.8000

gdeCR: 0.9000

shd: 0

savedir: []

savename: []

datafilex: [1x84 char]

datafiley: [1x83 char]

usetestdata: 0

testdatafilex: []

testdatafiley: []

files2data: 'xy2inout'

numbackgen: 3

percentback: 0.2500

percentchange: 0.2500

adaptinterval: []

adaptwindowsize: []

smalldifference: 0.0125

minprob: 0.1000

tournamentsize: 0.0100

depthnodes: '1'

fixedlevel: 1

dynamiclevel: '1'

inicmaxlevel: 6

inicdynlevel: 6

realmaxlevel: 17

veryheavy: 0

calccomplexity: 0

calcdiversity: {}

hits: [100 0]

output: 'normal'

graphics: {'plotfitness'}

keepevalssize: 100

opeq: 0

opeqtype: 'DynOpEq'

opeq_binsize: 1

opeq_target: []

rst: 0

rst_rss: 1

rst_rsr: 1

rst_ri: 0.5000

M3GP: 0

filters: {'strictdepth' 'dyndepth'}