Aplicação de uma Programação Genética Gramatical Coevolutiva no Apoio à Inferência da Máxima Deformação Longitudinal de Dutos com Amassamento

João Marcos de Freitas

JUIZ DE FORA

DEZEMBRO, 2016

Aplicação de uma Programação Genética Gramatical Coevolutiva no Apoio à Inferência da Máxima Deformação Longitudinal de Dutos com Amassamento

João Marcos de Freitas

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Heder Soares Bernardino

Co-orientador: Helio José Corrêa Barbosa

JUIZ DE FORA

DEZEMBRO, 2016

Aplicação de uma Programação Genética Gramatical Coevolutiva no Apoio à Inferência da Máxima Deformação Longitudinal de Dutos com Amassamento

João Marcos de Freitas

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Heder Soares Bernardino
Doutor

Douglas Adriano Augusto
Doutor

Itamar Leite de Oliveira
Doutor

JUIZ DE FORA

9 DE DEZEMBRO, 2016

Aos meus pais, amigos e professores.

Resumo

Combustíveis fluidos extraídos de fontes subterrâneas são um recurso importante no atual cenário de produção de energia. Todavia, existem vários fatores que dificultam essa prática, como os danos sofridos pelo duto que realiza a extração dos combustíveis, causando amassamentos. O objetivo desse trabalho é determinar meios de relacionar características observadas no duto e no fluido com a máxima deformação longitudinal do duto a partir de dados obtidos através de análises via o Método dos Elementos Finitos. É proposto então automatizar esse processo de descoberta utilizando um sistema inteligente que consiga evoluir modelos em forma simbólica. Métodos de Programação Genética (PG) têm se mostrado adequados a esse tipo de aplicação e uma PG Gramatical é adotada aqui, em que os modelos (programas) são inferidos por meio de uma Gramática Formal. Dentre as vantagens da adoção de uma gramática formal, pode-se destacar a garantia de geração de programas/modelos válidos e a possibilidade de limitar o espaço de busca pela introdução de algum viés (por exemplo, considerando o conhecimento prévio do especialista). Além disso, é usada uma abordagem coevolutiva visando priorizar a avaliação de dados que são mais difíceis de serem avaliados. Experimentos computacionais preliminares foram conduzidos para resolver o problema de inferir um modelo para a máxima deformação longitudinal de dutos com amassamento e os resultados encontrados mostram que a aplicação da PG Gramatical Coevolutiva é capaz de resolver o problema com boa acurácia e menor custo computacional.

Palavras-chave: Programação Genética Gramatical, Coevolução, Inteligência Computacional, Engenharia de dutos.

Abstract

The extraction of underground fluid fuels is an important resource in order to produce energy. However, there are several factors that make this practice hard, such as damage that causes deformations on the pipe that extracts the fuels. The objective of this work is to determine relationships between characteristics observed on the pipe and fluid with the maximum logitudinal deformation of the pipe from data obtained through analyses using the Finite Element Method. An automatic process for knowledge discovery using an intelligent system that can evolve models in symbolic form is proposed here. Genetic Programming methods presented good results to this type of application and a grammatical approach is adopted here, where the models (programs) are inferred by means of a Formal Grammar. A grammar brings to the GP technique the benefit of generating only valid programs/models and the possibility of limiting the search space by introducing bias. Also, a co-evolutionary approach is used to focus the search process on data which are harder to evaluate. Preliminary computational experiments were conducted to solve the problem of inferring a model of the maximum longitudinal deformation of pipes and the results indicate that the application of a Co-evolutionary Grammar based Genetic Programming can solve this problem with good accuracy and less computational cost.

Keywords: Grammar based Genetic Programming, Co-evolution, Computational Intelligence, Pipe Engineering.

Agradecimentos

Agradeço a Deus, pela oportunidade de estar aqui e também a minha família, em especial aos meus pais Marta e João, pelo apoio, suporte e amor durante esses anos. Aos meus amigos, que me ajudaram de diversas maneiras durante minha graduação. Aos meus orientadores, pela paciência, didática e pela oportunidade de desenvolver estes trabalhos. Aos meus professores, que me deram capacidade e conhecimento para chegar até aqui. Muito obrigado.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
AG	Algoritmo Genético
PG	Programação Genética
PGG	Programação Genética Gramatical
PGGM	Programação Genética Gramatical Multiobjetiva
PGGC	Programação Genética Gramatical Coevolutiva
PIG	Programação Imunológica Gramatical
MMQ	Método dos Mínimos Quadrados
GLC	Gramática Livre de Contexto

1 Introdução

1.1 Programação Genética

A automatização de processos tem sido feita pelo homem por séculos e ganhou grande avanço com o apoio de métodos computacionais. Com o objetivo de auxiliar a descoberta de conhecimento, principalmente de cunho científico, tem sido desenvolvidas técnicas como a Programação Genética (PG), onde seu principal objetivo é, através de um programa de computador, aprender a resolver um problema sem ser explicitamente programado para isso.

Em "A Origem das Espécies" (DARWIN, 1859), sobre a Teoria da Evolução, Darwin apresenta a origem da diversidade biológica onde, através de adaptações graduais via seleção natural, as espécies se especializam em relação ao ambiente onde vivem. A teoria da seleção natural mostra a tendência de indivíduos mais aptos terem maior chance de sobrevivência e acesso à reprodução. Dessa forma, esses indivíduos têm maior chance de propagar seu material genético. Esse processo permite que características favoráveis sejam propagadas e que as desfavoráveis tendam a desaparecer. As características são passadas através do cruzamento de indivíduos da população e, em certos casos, podem ocorrer mutações onde seu material genético é alterado. Na natureza, casos onde uma espécie afeta o processo de evolução de outra também podem ser observados. Esse processo conjunto é chamado de coevolução. Quando existe uma dependência entre diferentes organismos e ocorrem alterações em um deles, o outro tende a se adaptar a essa alteração, portanto, eles coevoluem.

A PG é uma meta-heurística estocástica bio-inspirada, baseada no princípio darwiniano da Seleção Natural. Ela gera programas candidatos para resolver um problema com base apenas no que é esperado como resposta, sem a necessidade de descrever o passo a passo de sua resolução ou utilizar informações inerentes ao problema. Uma solução candidata é representada como um indivíduo que possui um genótipo que pode ser traduzido em um fenótipo que é um programa. Ela evolui uma população de pro-

gramas candidatos à solução aplicando operadores genéticos e selecionando os mais aptos para continuar o processo a cada iteração. A adaptação de um programa ao problema está definida pela sua proximidade da resposta esperada, dessa forma programas mais adaptados possuem características favoráveis e permanecem mais tempo na população.

No entanto, a PG não difere programas consistentes de inconsistentes e possui um espaço de busca muito amplo, onde programas indesejados podem ser gerados. Em exemplo seria utilizar argumentos de tipos diferentes dos aceitos por uma função, como operações booleanas sobre argumentos reais. Com a necessidade de limitar o espaço de busca, podem ser adotadas gramáticas formais livres de contexto (GLC). Assim, os programas são gerados através de uma linguagem definida pelo usuário (AUGUSTO, 2004; FREITAS et al., 2014). Através de uma GLC é possível criar regras que definem a sintaxe dos programas e, assim, o espaço de busca é restringido às soluções consistentes.

Outra característica da PG é a geração de soluções simbólicas. Sendo assim, os programas gerados podem ser interpretados, tornando possível identificar características da solução, que esta escrita numa linguagem definida pela GLC.

A PG utiliza dados do fenômeno que se deseja modelar e tenta encontrar relações entre eles para resolver o problema de modelagem. No entanto, em alguns casos, alguns dados do problema são mais difíceis de serem avaliados de forma correta do que outros. É desejável uma maneira de explorar esses dados de forma mais eficiente a fim de melhorar as soluções em direção a esses dados. Neste contexto, diversas pesquisas mostram que a coevolução é uma boa estratégia (PAREDIS, 1994; AUGUSTO, 2004, 2009).

1.2 Trabalhos Relacionados

Várias técnicas de aprendizagem de máquina foram desenvolvidas nos últimos tempos a fim de inferir modelos a partir de dados. Dentre essas técnicas pode-se destacar por exemplo: Máquinas de Vetor de Suporte (MVS), Redes Neurais Artificiais (RNA) e Algoritmos Genéticos (AG). Técnicas como MVS são baseadas no aprendizado supervisionado utilizando dados de entrada para reconhecer padrões e, assim, realizar a regressão ou classificação dos dados. RNA é uma técnica também de aprendizado de máquina que visa criar modelos matemáticos que simulem o funcionamento de neurônios de organismos

complexos que aprendem através de experiência e reforço.

A Programação Imunológica Gramatical (PIG) é uma variante de Programação Genética que utiliza conceitos de sistemas imunológicos naturais para realizar a busca. Os resultados alcançados em (BERNARDINO et al., 2011) foram superiores do que aqueles desenvolvidos por especialistas e usados em códigos internacionais que definem regras padrões para esse tipo de problema. Além disso, os autores apresentaram (i) modelos mais complexos mas acurados e (ii) modelos mais simples mas com erros maiores em relação aos valores esperados.

A aplicação de uma Programação Genética Gramatical (PGG) capaz de resolver o problema dos dutos e reforçar informações previamente encontradas em (BERNARDINO et al., 2011) foi apresentada em (FREITAS et al., 2014). A PGG evolui modelos candidatos gerados a partir de gramáticas formais, representando os programas em estruturas de árvores de derivação, utilizando operações de seleção, recombinação e mutação, e usando elitismo.

Observando os resultados obtidos pela PGG, verificou-se que os modelos encontrados são complexos. Por isso, a PGG foi modificada em (FREITAS et al., 2015) para realizar uma busca multiobjetivo a fim de minimizar o erro e a complexidade dos modelos.

A coevolução também é utilizada em diversos trabalhos na literatura. Em (PAREDIS, 1994), é desenvolvido um Algoritmo Genético Coevolucionário para ajustar pesos de RNAs para a classificação de dados. Em seu trabalho, ele descreve o processo de coevolução promovendo competições entre as populações de programas e dados de entrada. A coevolução ocorre selecionando um representante de cada população e realizando uma competição. Se uma RNA classifica um dado corretamente, sua aptidão é acrescida enquanto a do dado é decrescida. Porém, se ela não classifica corretamente, sua aptidão é decrescida e a do dado é acrescida. O histórico desses confrontos é armazenado em um registro. É então introduzido o conceito de aptidão cumulativa, onde a aptidão de um indivíduo depende das últimas competições em que participou.

Uma aplicação de Programação Genética Gramatical realizando coevolução amostra-classificador pode ser vista em (AUGUSTO, 2004). Com a tarefa de inferir modelos para a classificação de dados, o algoritmo aplica a coevolução promovendo competições entre

a população de soluções candidatas e os dados de treinamento. Assim como em (PAREDIS, 1994), uma memória cumulativa que armazena os sucessos e derrotas nos indivíduos é utilizada.

Posteriormente, Augusto (2009) apresenta ainda uma aplicação da Programação Genética Multipopulacional e Coevolucionária para classificar dados. Em seu trabalho, é proposto o uso dos conceitos de *bagging* (BREIMAN, 1996) e *boosting* (SCHAPIRE, 1990), além de aplicar a coevolução em dois âmbitos: o primeiro faz a coevolução entre populações diferentes que cooperam em um regime semi-isolado num modelo de ilhas e o segundo dentro de uma mesma população.

Neste trabalho pretende-se aplicar a coevolução entre a população de programas e a de dados de treinamento em um problema de regressão simbólica, utilizando também a aptidão cumulativa.

1.3 Problema

A inferência de modelos a partir de dados é uma boa alternativa para encontrar soluções de problemas de modelagem. Pode-se destacar para esse tipo de situação as técnicas de Programação Genética, por sua capacidade de encontrar fórmulas aproximadas e acuradas para representar o fenômeno que está sendo modelado. Ainda é possível limitar o uso de variáveis nas soluções e controlar a complexidade dos modelos.

Ao ser detectado um amassamento em um duto, definido pela deformação sem perda de material, uma análise rápida e precisa sobre suas condições deve ser realizada para definir seu estado e, então, tomar decisões sobre sua manutenção. Códigos internacionais, como o da ASME B31.8 (2007), indicam que a decisão pode ser tomada de acordo com limites baseados na deformação equivalente de von Mises, que define o estado de deformação de sólidos. Uma alternativa para estimar seu estado é utilizar o Método dos Elementos Finitos. Entretanto, o método é inviabilizado por ser computacionalmente custoso e, desta forma, dificilmente pode ser adotado nesse tipo de ambiente. Surge então a necessidade de definir um modelo que seja capaz de auxiliar a tomada de decisão de forma acurada, com baixa complexidade e com custo computacional reduzido.

A ASME B31.8 (2007) indica que a deformação do duto se baseia nas deformações

de flexão, referentes à curvatura do duto, e na maior deformação longitudinal da membrana. São apresentados em (NORONHA et al., 2008) resultados similares àqueles encontrados pelo Método de Elementos Finitos para as componentes de flexão utilizando um modelo simples. (BERNARDINO et al., 2011) apresentam uma Programação Imunológica Gramatical (PIG) aplicada a esse problema. Os resultados obtidos mostram que o os modelos encontrados são mais acurados que os anteriormente definidos. Em (FREITAS et al., 2014, 2015), são apresentadas soluções de melhor acurácia em relação aos encontrados por (BERNARDINO et al., 2011) utilizando a Programação Genética Gramatical e sua versão multiobjetivo. No entanto, é verificado que dentre os registros utilizados, alguns são mais difíceis de serem representados. Surge então a necessidade de utilizar alguma técnica que trate esse fenômeno encontrado nos dados, na tentativa de melhorar a acurácia das soluções. A ideia de coevolução é então usada aqui.

O banco de dados utilizado aqui é o mesmo usado em (BERNARDINO et al., 2011; FREITAS et al., 2014). Este banco é composto por 554 registros contendo a deformação longitudinal máxima de dutos com diferentes valores de diâmetro externo (D), espessura da parede (t), profundidade de identação (d) e o diâmetro do identador (Φ). Os valores de deformação foram obtidos através de simulações computacionais utilizando o Método de Elementos Finitos (BERNARDINO et al., 2011). A Figura 1.1 ilustra um duto e alguns dos atributos utilizados.

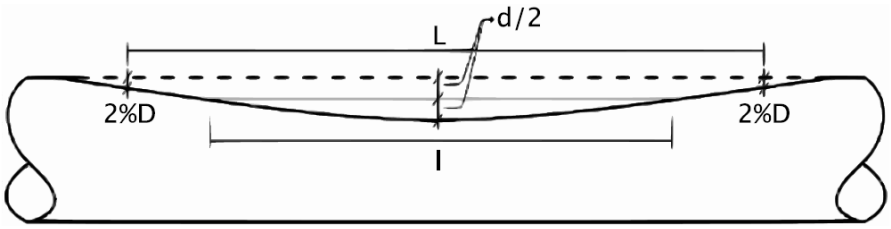

Figura 1.1: Ilustração do duto com amassamento (BERNARDINO et al., 2011).

Junto dos parâmetros descritos anteriormente, são utilizados no modelo: a tensão circunferencial (σ_θ), induzida pela pressão interna, e a tensão de escoamento do material (σ_y). O conjunto de variáveis adimensionadas que seguem foram apresentadas por (BERNARDINO et al., 2011):

(i) $x_1 = t/D$

(ii) $x_2 = \sigma_\theta / \sigma_y$

(iii) $x_3 = d/D$

(iv) $x_4 = \Phi/D$

(v) $x_5 = v$, onde v é o coeficiente de Poisson

(vi) $x_6 = E_t/E$, onde E é o módulo de Young e E_t um módulo tangente

(vii) $x_7 = d/l$, onde l é o tamanho da deformação em 0.5 d

(viii) $x_8 = d/L$, onde L é o tamanho da deformação em 2% D

Como um mesmo material foi adotado nas simulações, então as variáveis x_5 e x_6 não são usadas aqui. Portanto, assim como em (BERNARDINO et al., 2011; FREITAS et al., 2014), as variáveis x_1, x_2, x_3, x_4, x_7 e x_8 serão adotadas aqui para encontrar uma forma simbólica para $\epsilon : \mathbb{R}^6 \to \mathbb{R}$ tal que

$$\epsilon = f(x_1, x_2, x_3, x_4, x_7, x_8)$$

representa um modelo para máxima deformação longitudinal de dutos com amassamento.

1.4 Hipótese

Trabalhos anteriores (FREITAS et al., 2014, 2015) mostram que a PG é capaz de encontrar bons resultados para o problema de inferir um modelo para a máxima deformação longitudinal de dutos. No entanto, verifica-se que alguns dados do problema são difíceis de serem avaliados e, assim, diminuem a acurácia das soluções. Dessa forma, é desejável introduzir um viés na busca para tratar essas entradas e melhorar a acurácia. Os trabalhos (PAREDIS, 1994; AUGUSTO, 2004, 2009) apresentam a aplicação de técnicas de coevolução para focar a busca em entradas que acumulam mais erro e os bons resultados mostram que a técnica é promissora. Por apresentar circunstâncias similares no problema abordado aqui, a PGG pode ser modificada para aplicar coevolução entre uma população de modelos de regressão e uma de padrões de entrada. Assim, a hipótese considerada

aqui é a de que se a busca por modelos para a máxima deformação longitudinal de dutos com amassamentos focar nos dados que são mais difíceis de serem previstos, resultados melhores serão encontrados, já que menos processamento será despendido com entradas mais fáceis.

1.5 Justificativa

Este trabalho é justificado dada importância de se definir modelos acurados para o problema descrito, que é essencial para soluções energéticas. Outro ponto se refere aos bons resultados encontrados em (AUGUSTO, 2009) aplicando a técnica de coevolução em um ambiente de classificação de dados. Apesar de se tratar de uma área diferente de problemas, a técnica tende a ser apropriada também em um ambiente de regressão e, assim, trazer dois ganhos à PGG: (i) melhorar acurácia dos modelos encontrados e (ii) realizar a busca com menor custo computacional.

1.6 Objetivos

O objetivo geral deste trabalho é propor e analisar uma PGG coevolutiva quando aplicada ao problema da inferência de um modelo para a máxima deformação longitudinal de dutos com amassamentos. Os objetivos deste trabalho são:

- Definir formas de como tratar problemas de regressão simbólica num ambiente de coevolução;
- Avaliar se a Programação Genética Gramatical Coevolutiva (PGGC) é capaz de encontrar boas soluções para o problema da máxima deformação longitudinal de dutos com amassamento;
- Reforçar informações encontradas em trabalhos anteriores sobre a Programação Genética Gramatical (PGG).

2 Programação Genética Gramatical

2.1 Algoritmo Genético

Na década de 1960 surgiram os Algoritmos Genéticos (AG), métodos capazes de inferir, num processo evolutivo simulando a evolução natural, uma solução de um problema. Esses métodos ganharam visibilidade e interesse de muitos pesquisadores como (HOLLAND, 1975) que em seu trabalho mostra como o processo de seleção natural que ocorre na natureza também pode ser aplicado em sistemas artificiais. Neste mesmo trabalho, são apresentados os primeiros vislumbres de coevolução. Ao longo dos anos, diversas aplicações de AG foram vistas na tentativa de desenvolver programas de computador iniciando os primeiros passos da PG. O AG melhora uma população de indivíduos em um processo evolutivo baseado em algum critério representado pela aptidão.

No processo evolutivo são criados novos indivíduos através de operações genéticas sobre a população, como cruzamento (troca de material genético) e mutação (alteração de material genético), gerando uma nova população a cada geração. A nova população gerada deve substituir a antiga, no entanto existem algumas formas de fazer essa substituição, como uma abordagem geracional onde os novos indivíduos substituem os mais velhos, porém os piores indivíduos da nova população dão lugar aos melhores da população anterior, chamados de elite. Outra abordagem é o *steadystate* que ao longo de uma geração substitui os indivíduos menos aptos pelos recém gerados através da recombinação em tempo de execução. Após satisfazer o critério de parada do processo evolutivo a PG retorna o melhor indivíduo encontrado. O fluxograma 2.1 exemplifica o funcionamento de um AG genérico.

2.2 Programação Genética

Sendo uma especialização dos Algoritmos Genéticos, a PG foi formalizada em (KOZA, 1992) como a evolução de populações de soluções a um problema arbitrário na forma de

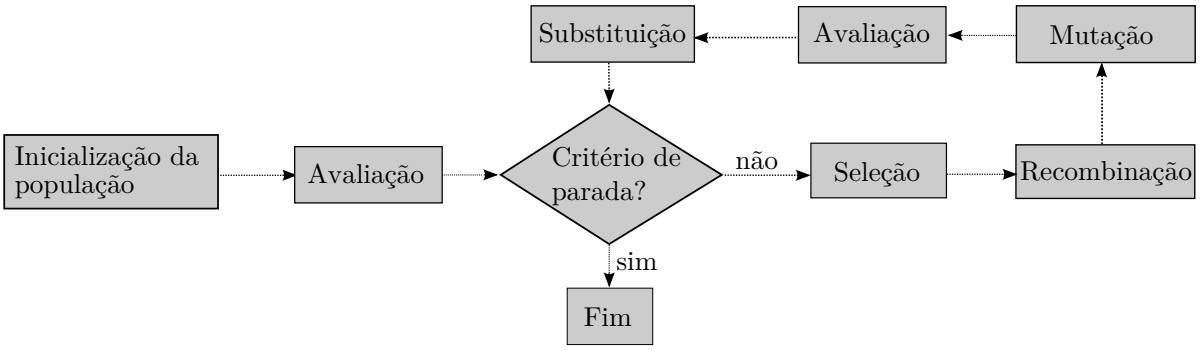

Figura 2.1: Fluxograma AG genérico.

programas de computador em um processo que simula a evolução natural. A evolução se dá em um processo iterativo, aplicando uma série de operadores genéticos sobre uma estrutura que representa um programa. Diversas estruturas são utilizadas para representar os programas, mas três tipos são mais utilizados (AUGUSTO, 2009): linear, onde um programa é representado por uma palavra, árvore e grafo. Cada modelo representa simbolicamente um programa que pode ser executado e, assim, avaliado. A população é comumente formada por um conjunto de tamanho fixo de indivíduos.

Assim como nos Algoritmos Genéticos, as operações de cruzamento e mutação fazem parte da PG operando sobre a população corrente e gerando novos indivíduos. No cruzamento, dois indivíduos selecionados na população geram dois filhos que são formados pela carga genética dos pais. A mutação adiciona variabilidade genética na população alterando o genótipo de um indivíduo recém gerado. Essas operações são detalhadas na Seção 2.4.

O processo de evolução faz uma seleção de indivíduos da população para participar do processo reprodutivo que realiza recombinações e mutações. Os indivíduos gerados são classificados e os mais aptos são mantidos para a próxima geração. Esse processo é repetido até que um critério de parada seja atendido. Diversos critérios de parada podem ser aplicados, tais como: (i) número de gerações, (ii) número de cálculos da função objetivo e (iii) erro dentro de um limite aceitável.

saiNa Programação Genética utilizada em (FREITAS et al., 2015), o critério de parada adotado é baseado no número de cálculos de função objetivo, totalizando 100000 cálculos. O tamanho da população e o número de gerações é ajustado para que seja realizado esse total de operações. DOUGLAS

Este processo evolutivo é como apresentado no Algoritmo 1.

Algoritmo 1: Pseudo-código da Programação Genética.

```
1  Criar aleatoriamente a população inicial P;
2  Inicializar a memória dos registros;
3  Avaliar todos os indivíduos de P;
4  enquanto critério de parada não for satisfeito faça
5  |  enquanto |P_tmp| < |P| faça
6  |  |  Selecionar indivíduos p_1 e p_2 por torneio;
7  |  |  Copiar p_1 em p'_1 e p_2 em p'_2;
8  |  |  se aleatório ≤ p_cruzamento então
9  |  |  |  Cruzar p'_1 com p'_2;
10 |  |  fim se
11 |  |  se aleatório ≤ p_mutação então
12 |  |  |  Aplicar operadores de mutação em p'_1 e p'_2;
13 |  |  fim se
14 |  |  Avaliar p'_1 e p'_2;
15 |  |  Inserir p'_1 e p'_2 em P_tmp;
16 |  fim enqto
17 |  P ← elite de P ∪ elite de P_tmp;
18 |  descartar P_tmp;
19 fim enqto
20 retorna melhor indivíduo encontrado;
```

2.3 Gramáticas Formais

Na PG, mesmo tendo uma boa representação dos modelos, ainda é possível que soluções inválidas sejam geradas. As Gramáticas Formais Livres de Contexto (GLCs) podem ser utilizadas para restringir o espaço de busca para que apenas programas válidos sejam criados. Sua aplicação no campo da PG incide diretamente sobre a estrutura dos programas, definindo os componentes que podem ser utilizados (tais como funções e variáveis) e suas regras de combinação (AUGUSTO, 2004). Uma PG utilizando uma GLC para a inferência da máxima deformação longitudinal de dutos com amassamento é apresentada em (FREITAS et al., 2014, 2015). Através da GLC, pode-se estabelecer regras nas relações entre os operadores e operandos. Por exemplo, pode-se permitir que operadores lógicos utilizem apenas operandos lógicos ou que um operador de raiz quadrada nunca atue sobre uma variável particular.

A GLC atua como uma geradora de sentenças obedecendo as regras definidas

nela (AUGUSTO, 2004) e pode ser definida por uma quádrupla $G = (N, \Sigma, S, P)$, onde:

N : alfabeto de não terminais (elementos auxiliares da gramática)

Σ : alfabeto de terminais (aqueles que aparecem nas palavras da linguagem)

S : símbolo inicial; $S \in N$

P : conjunto de produções

Pode-se definir regras para cada elemento de N que são chamadas de produções. Uma ilustração de regra de uma GLC pode ser como:

$$<\text{NT}> \ ::= prod1 \mid prod2 \mid prod3$$

onde <NT>é um não terminal, e $prod1$, $prod2$ e $prod3$ são produções de <NT>. Segue um exemplo de um conjunto de regras de produção para expressões aritméticas posfixadas, onde os argumentos são dispostos antes de uma função:

$$< expr > ::= < expr >< expr >< op > \mid < expr >< uop > \mid < var > \mid < num >$$

$$< op > ::= + \mid - \mid * \mid / \mid pow$$

$$< uop > ::= log \mid exp \mid sqrt \mid sen \mid cos \mid tg$$

$$< var > ::= x \mid y \mid z$$

$$< const > ::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid \pi$$

Ao adotar uma representação em estrutura de árvore combinada com GLC, os programas passam a ser árvores de derivação, onde os nós internos são não terminais e as folhas terminais da gramática. Devido a essas características, esta representação foi escolhida aqui, trazendo ganho e estrutura para as operações genéticas. A derivação de um modelo a partir de uma GLC é ilustrada na Figura **??**. A partir do não terminal inicial ($< expr >$), uma de suas produções é escolhida e tornam-se os nós filhos da raiz da árvore de derivação. Em seguida, para cada não terminal gerado é feito uma derivação

seguindo o mesmo procedimento. Esses passos se repetem até que seja gerado um nó terminal (folha). Neste exemplo são necessários 8 passos para obter um modelo. Dois indivíduos sob esta perspectiva são apresentados nas Figuras 2.3 e 2.4. Esses modelos serão utilizados na Seção para ilustrar as operações genéticas.

2.4 Operações

Operações genéticas são necessárias para que o processo de evolução aconteça. Na natureza é possível observar esse fenômeno, principalmente, no reino animal, onde ocorre o cruzamento entre indivíduos de uma mesma espécie para gerar descendentes. Pode acontecer também o fenômeno da mutação, que é um processo inerente apenas ao próprio indivíduo.

2.4.1 Cruzamento

O cruzamento na PGG deve respeitar as regras da GLC adotada. Esse processo é realizado entre dois modelos candidatos A (Figura 2.3) e B (Figura 2.4) e é ilustrada na Figura 2.5. No exemplo de cruzamento entre os indivíduos A e B, uma subárvore de A é selecionada aleatoriamente. Em seguida, é sorteada uma subárvore em B onde o nó raiz dessa segunda subárvore contém a mesma raiz da primeira subárvore. Estas subárvores são então trocadas entre A e B dando origem a dois (novos) modelos A' e B'. Realizando a recombinação dessa forma, garante-se que os novos modelos gerados são sempre válidos, logo que, as subárvores trocadas foram derivadas a partir do mesmo não terminal da gramática utilizada. A ocorrência da recombinação depende de uma probabilidade definida pelo usuário e que normalmente assume um valor alto.

2.4.2 Mutação

A Mutação (ilustrada na Figura 2.6) também deve ser realizada de modo que o modelo resultante atenda às regras da GLC. Um nó não terminal do indivíduo é sorteado e, a partir dele, é derivada uma nova subárvore baseada na mesma regra na GLC, garantindo assim, que o modelo é continua factível.

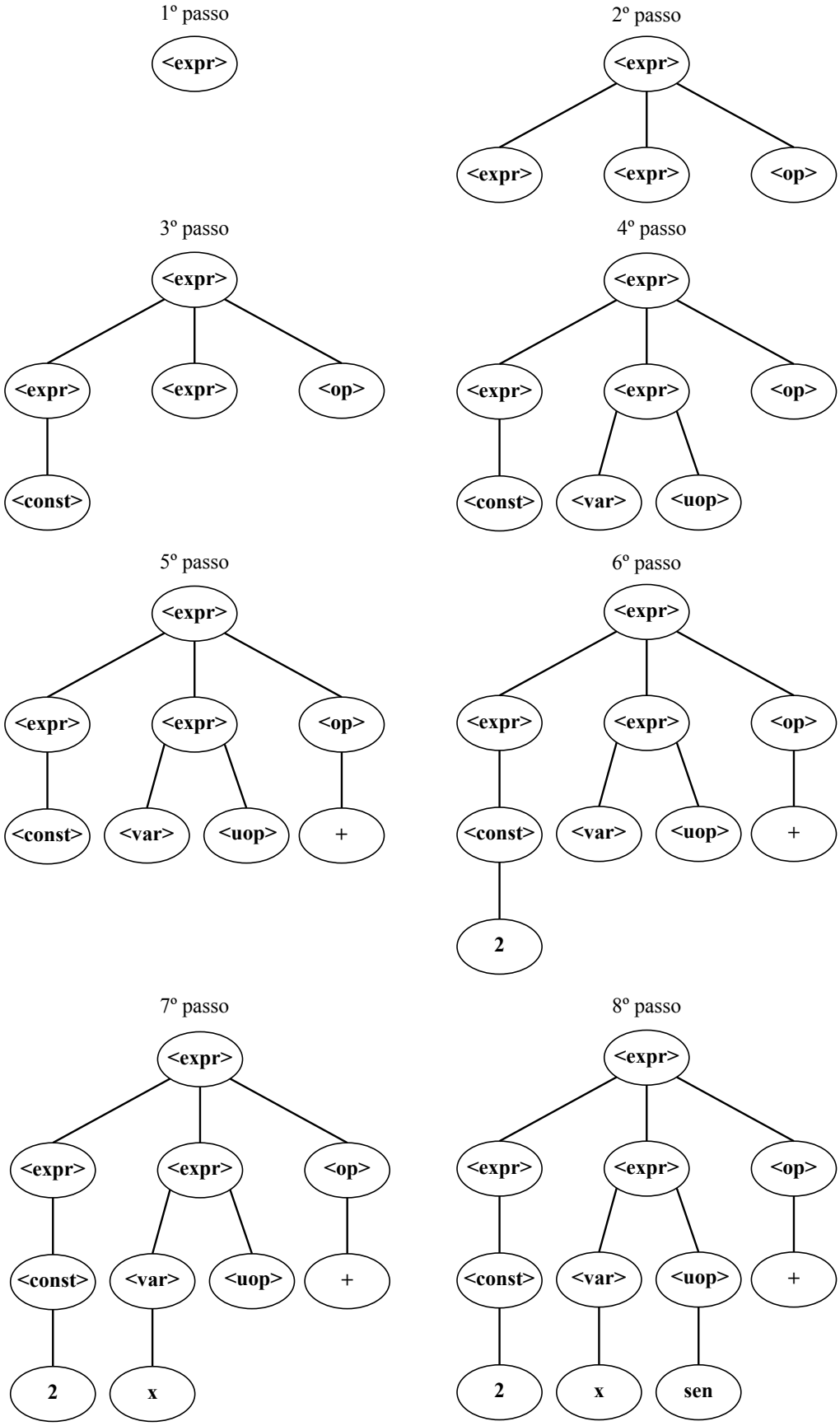

Figura 2.2: Exemplo da geração de um modelo ao acaso usando uma GLC.

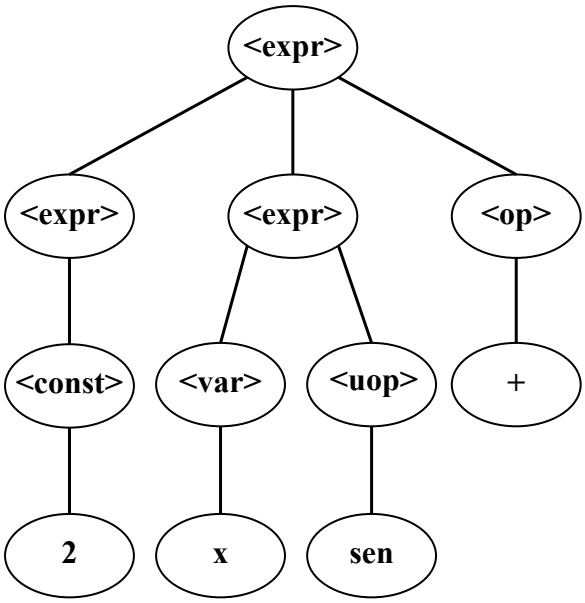

Figura 2.3: Exemplo de representação para o modelo 2 x sen + que representa o programa "$2 + sen(x)$".

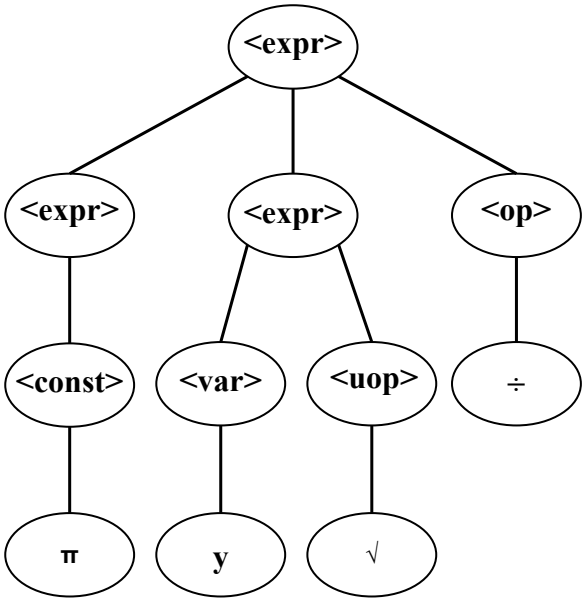

Figura 2.4: Exemplo de representação para o modelo π y $\sqrt{}$ $\div$ que representa o programa "$\frac{\pi}{\sqrt{y}}$".

Com o objetivo de limitar a complexidade dos modelos gerados e evitar problemas como orçamento computacional, é comum incorporar uma restrição de profundidade máxima nas árvores da PGG. Desta forma, os procedimentos de geração da população inicial, cruzamento e mutação estão sujeitos a essa restrição. Ao realizar a troca entre subárvores, é necessário que ela mantenha a profundidade das árvores abaixo do limite, assim como quando é derivado uma nova subárvore na mutação.

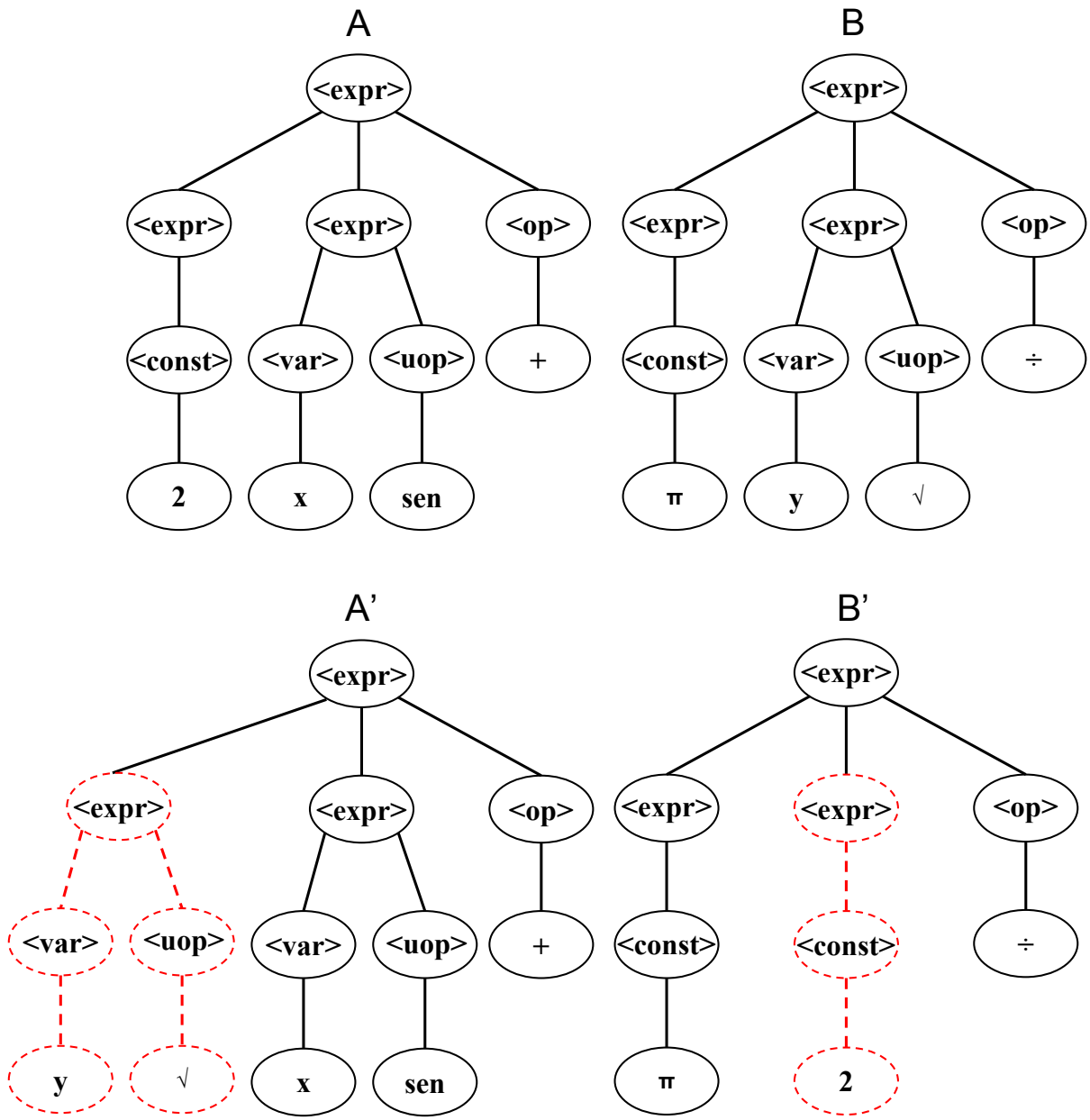

Figura 2.5: Exemplo de Cruzamento.

2.5 Método dos Mínimos Quadrados

É comum utilizar técnicas para ajustar melhor os modelos a seus dados. Uma alternativa é utilizar o Método dos Mínimos Quadrados (MMQ), um caso específico de Identificação Paramétrica que minimiza o erro de um modelo ajustando alguns parâmetros. Neste caso, uma estrutura é definida para o modelo de forma a adicionar coeficientes a_0, a_1, a_2, ..., a_n que serão ajustados para resolver o problema com mais acurácia. É definido $f(x) = g(x, a)$, onde $f(x)$ é conhecida, x são os valores a serem avaliados e a é o conjunto de coeficientes a serem ajustados a fim de minimizar a soma dos quadrados dos erros. Isso

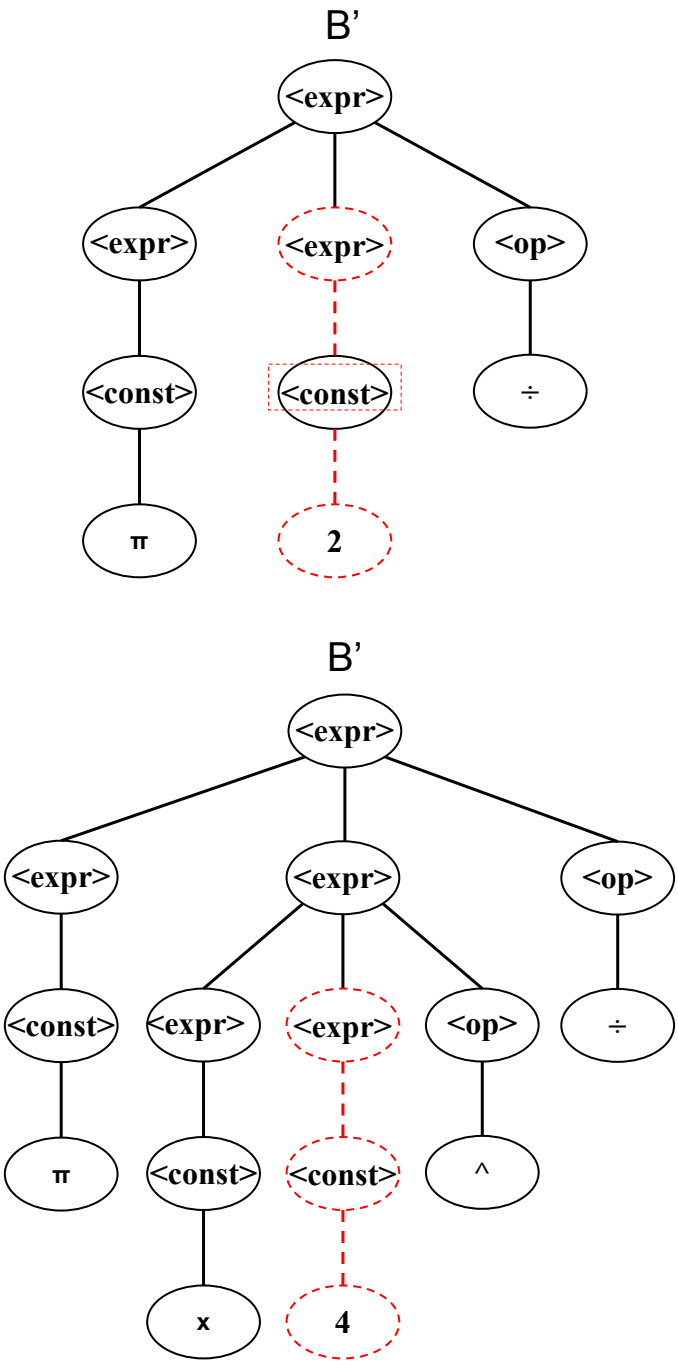

Figura 2.6: Exemplo de Mutação.

pode ser feito minimizando a função

$$\Psi(a) = \left(\sum_{i=1}^{M} |y_i - g(x_i, a_0, ..., a_n)|^p \right)^{1/p}$$

onde y_i é o valor esperado, $g(x_i, a_0, ..., a_n)$ o valor obtido para o i-ésimo dado e p é um coeficiente. No caso onde $p = 2$, $\Psi(a)$ é diferenciável e se a função f é descrita como

uma combinação linear de funções de base ϕ_i (ou seja, $f(x) = g(x, a) = \sum_{i=1}^{n} a_i \phi_i(x)$), o problema recai em resolver um sistema linear com n equações. Os coeficientes que minimizam a função $\Psi(a)$ são obtidos ao resolver o sistema linear. Este método é chamado de Mínimos Quadrados. Na forma matricial, o Método dos Mínimos Quadrados pode ser denotado como

$$\left(X^T X\right) a = X^T y$$

onde X são os dados do problema, y é o vetor de respostas esperadas para cada registro e T indica a transposta de uma matriz.

Na PGG as funções de base são identificadas pelo operador "!" que simboliza a soma de funções de base. A Figura 2.7 mostra um modelo nessa representação. As funções de base para os coeficientes $a_0, a_1 e a_2$ são respectivamente $\phi_0(x) = x$, $\phi_1(x) = z$ e $\phi_2(x) = y^2$ montando um sistema de 3 equações.

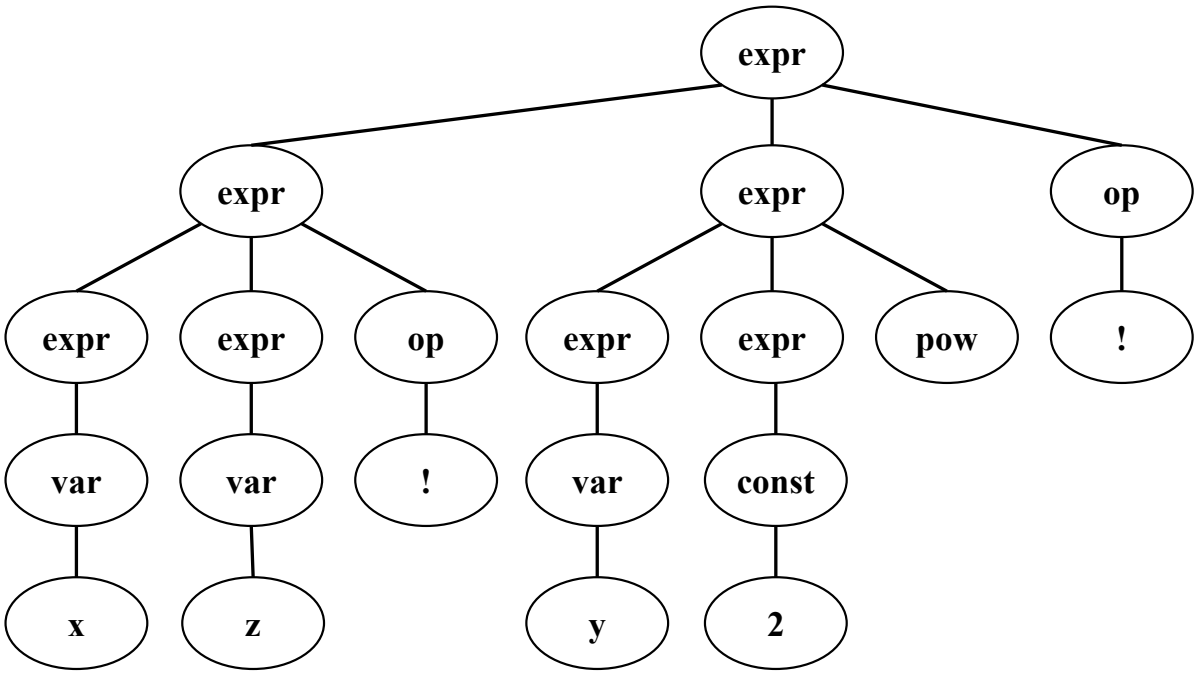

Figura 2.7: Exemplo de modelo definindo funções de base.

Utilizando essas ideias, a PGG é capaz de gerar modelos sempre válidos e compreensíveis ao usuário, facilitando assim a descoberta de conhecimento, além de ser possível minimizar o erro das soluções via MMQ. Apesar das muitas vantagens, existem limitações computacionais para a aplicação da PG em problemas reais, como aqueles em que os modelos são demasiadamente extensos ou requerem simulações computacionalmente muito

custosas. No entanto, com a evolução da tecnologia e disponibilidade de computadores com cada vez mais memória e poder de processamento, a PG se torna uma alternativa apropriada. Além disso, é possível paralelizar facilmente a PGG, obtendo alto ganho em termos de tempo de processamento. Uma outra grande vantagem da PGG é sua fácil adaptação e integração com outras técnicas, sendo possível criar algoritmos híbridos como pode ser visto em (AUGUSTO, 2009) onde a PG é hibridizada com as técnicas de *boosting* e *bagging*.

3 Coevolução

3.1 Introdução

A coevolução pode ser definida pelo processo onde ocorre uma mudança evolucionária em uma população em resposta a interações de uma segunda população que exerce uma pressão sobre a primeira; finalmente, a segunda população é também modificada em resposta às alterações da primeira. Diversos tipos de coevolução podem ser encontrados na natureza, sendo de natureza mutualística ou competitiva. No mutualismo, ambas espécies se beneficiam das mudanças. Quando a coevolução é competitiva ela pode aparecer em dois cenários: no caso de predatismo, onde uma espécie desenvolve vantagem para se alimentar de outra ou no parasitismo onde uma se beneficia de outra mas sem matá-la. Na natureza, um exemplo claro pode ser visto na polinização, onde uma planta depende de um animal como uma abelha para captar seu pólen e fecundar outra planta. Já o animal depende do néctar para se alimentar. A evolução das plantas as torna mais atrativas aos animais enquanto os animais desenvolvem características propícias a polinizar mais plantas. Este processo afeta inclusive outras populações que dependem do sucesso da polinização para sobreviver, gerando uma grande rede coevolucionária.

3.2 O Algoritmo de Coevolução

A coevolução no âmbito da PG altera a maneira como a busca é realizada a fim de focar mais esforço computacional em registros de difícil avaliação. Assim como nos trabalhos (PAREDIS, 1994; AUGUSTO, 2009), é desejado coevoluir duas populações: a de indivíduos e a de dados de entrada. A PG é aplicada em (PAREDIS, 1994) e (AUGUSTO, 2009) a um ambiente de classificação de dados.

Para realizar a coevolução, é necessário atribuir um valor que indica a qualidade dos indivíduos e dos dados. Essa perspectiva acerca dos modelos candidatos e dos dados varia de acordo com o processo de busca. Essa qualidade representa a aptidão do modelo

e, por outro lado, a dificuldade de uma instância dos dados de ser corretamente avaliada pelos indivíduos da população. Essa qualidade é definida através de um histórico que armazena os sucessos e derrotas de suas últimas avaliações. As posições da memória são atualizadas através de competições entre um classificador e um dado. A posição mais antiga da memória de ambos dá lugar ao novo resultado, operando num esquema *first in first out*. Enquanto a população de classificadores se especializa nos dados, eles exercem uma pressão contrária utilizando registros mais difíceis nas competições, introduzindo um viés na busca na direção de tentar ajustar esses dados e, assim, representar melhor o problema.

A versão coevolutiva da PG altera alguns elementos do processo de busca, como a substituição da população. Comumente a PG é geracional, ou seja, a população mais antiga é substituída pela mais nova ao fim de cada geração; pode-se adicionalmente aplicar elitismo por exemplo. Já com a coevolução, dado o dinamismo do processo, a PG opera com melhor desempenho ao adotar um esquema de substituição $steady - state$, onde um par de novos indivíduos gerados já compete imediatamente para entrar na população. Aqui, adota-se o esquema de substituição dos dois piores indivíduos da população pelos filhos recém criados.

Alguns parâmetros são inseridos pela coevolução, como o tamanho da memória (m) e a quantidade de competições promovidas a cada geração do processo de busca (c). Esses parâmetros causam um importante reflexo no mecanismo de busca. Uma memória curta pode não caracterizar um registro devidamente, além de diminuir a diferenciação na população. Por outro lado, um valor de m muito grande diminui a capacidade de reação da busca às mudanças que ocorrem nos modelos (AUGUSTO, 2009). A quantidade de competições realizadas, chamada de ciclos, também precisa ser ajustada. Um valor pequeno de c não adapta os registros rapidamente, mas um grande valor pode atenuar o efeito de forma demasiada além de descaracterizar a vantagem da coevolução de realizar menos cálculos.

Em problemas de classificação, a qualidade de uma solução pode ser descrita pela sua taxa de acertos. Além disso, a dificuldade em se aprender a relação presente num registro pode ser descrito pela taxa de erros que ele gera. Num ambiente de regressão,

o erro observado ao avaliar um registro num modelo de regressão é calculado através da Equação 3.1 e esse valor é armazenado na memória de ambos. A aptidão de um indivíduo pode ser descrita através de seu erro em relação a um conjunto de dados utilizando a Equação 3.2, onde M é o número de registros utilizados no cálculo.

$$\text{erro} = (y_i - f(x_i))^2 \tag{3.1}$$

$$\text{erro quadrático médio} = \frac{1}{M}\sum_{i=1}^{M}(y_i - f(x_i))^2 = \frac{erro}{M} \tag{3.2}$$

Para que os registros possam ser comparados, um indicador é calculado sobre sua memória, a média dos erros acumulados, via Equação 3.3, onde m é o tamanho da memória de competições e *memoria* o vetor de m que armazena o histórico. Assim, um modelo de regressão bem adaptado acumula menos erro enquanto um menos adaptado tem uma média maior. Por outro lado, um registro com mais erro acumulado é identificado como um registro difícil de ser resolvido, enquanto um com menor acúmulo médio é considerado mais fácil.

$$\text{erro acumulado médio} = \frac{1}{m}\sum_{i=1}^{m}\text{memoria}\,[i] \tag{3.3}$$

O processo de coevolução aplicado aqui pode ser visto no Algoritmo 8. A cada ciclo é selecionado um modelo de regressão na população e em seguida um registro dos dados; ambos através de torneios. O torneio recebe dois indivíduos e compara a aptidão de cada um. No caso de um indivíduo, aquele com menor acúmulo de erro na memória pela equação 3.3 é selecionado. Por outro lado, ao se tratar dos registros, aquele que acumula mais erro é preferido. É realizada a competição avaliando o modelo de regressão com o registro selecionados (calculando $P_i(D_j)$).

Para aplicar a coevolução, o algoritmo 1 deve ser modificado, trocando a substituição da população pelo passo $steady - state$. Um pseudo-código da PGG coevolutiva é apresentado no Algoritmo 3.

A avaliação de um novo indivíduo consiste em preencher sua memória realizando m competições com registros selecionados via torneio. O processo dinâmico da coevolução traz também uma característica muito desejável: a avaliação de um indivíduo não requer

Algoritmo 2: COEVOLUIR

Entrada: População de regressores P, população de registros D

```
1 início
2     para cada ciclo faça
3         P_i ← regressor bem adaptado de P
4         D_j ← registro de dados de D via torneio
5         Avaliar regressor P_i com o registro D_j
6         Substitui memória mais antiga de D_i e P_i pela avaliação
7     fim para
8 fim
```

Algoritmo 3: Pseudo-código da Programação Genética Gramatical Coevolucionária.

```
1  Criar aleatoriamente a população inicial P;
2  Inicializar a memória dos registros;
3  Avaliar todos os indivíduos de P;
4  Avaliar todos os registros de D;
5  enquanto critério de parada não for satisfeito faça
6      enquanto i < |P| faça
7          Coevoluir(P, D);
8          Selecionar indivíduos p_1 e p_2 por torneio;
9          Copiar p_1 em p'_1 e p_2 em p'_2;
10         se aleatório ≤ p_cruzamento então
11             Cruzar p'_1 com p'_2;
12         fim se
13         se aleatório ≤ p_mutação então
14             Aplicar operadores de mutação em p'_1 e p'_2;
15         fim se
16         Avaliar p'_1 e p'_2;
17         Inserir p'_1 e p'_2 em P via steady state;
18     fim enqto
19 fim enqto
20 retorna melhor indivíduo encontrado;
```

a utilização de todos os registros. Isso significa que as avaliações demandam menos tempo de processamento, que é o maior custo envolvido na PG. Isso acontece pois para avaliar um indivíduo são feitas somente avaliações de registros suficientes para preencher sua memória. Como o tamanho da memória é menor que o conjunto total de dados, seu tempo de processamento é então reduzido.

4 Experimentos Computacionais

Como descrito nos objetivos desse trabalho, pretende-se resolver aqui o problema de obter um modelo em forma simbólica para máxima deformação longitudinal de dutos com amassamento utilizando a técnica da PGGC.

Os códigos da PGG, PGGM e PGGC foram desenvolvidos em linguagem C++ utilizando somente bibliotecas nativas da própria linguagem de programação. Uma estrutura de dados orientada a objetos é utilizada para definir os componentes da PG. O ambiente computacional utilizado nos experimento deste trabalho é formado por uma CPU Intel Core i7-3770, 3.40Ghz e 8Gb de memória RAM com sistema operacional Debian GNU/Linux 8 64bits. O código fonte é disponibilizado no repositório aberto no link bitbucket.org/ciml-ufjf/ciml-lib .

Esta sessão apresenta a divisão dos experimentos e comparações com outras técnicas aplicadas ao mesmo problema.

4.1 Métodos

Os dados descritos na Seção 1.3 foram utilizados (i) durante a evolução e (ii) para avaliar a capacidade de generalização dos modelos encontrados. Para isso, os dados foram separados em um conjunto de treinamento, validação e teste. A separação dos grupos é feita de forma aleatória sorteando 70% dos registros para treinamento, 20% como validação e 10% para teste. Vale ressaltar que os conjuntos de treinamento e validação são sorteados aleatoriamente em cada execução independente, já o de teste é mantido fixo. Ao final do processo evolutivo a população tem sua aptidão calculada pela equação 3.1 utilizando os dados de validação, e o melhor indivíduo é avaliado sobre os dados de teste e selecionado para comparação entre execuções independentes.

Os parâmetros da PGG utilizados aqui são os mesmos definidos em (FREITAS et al., 2015). A população é formada por 50 indivíduos e seu tamanho é fixo durante a evolução. É adotado como critério de parada o cálculo de 100000 avaliações de função

objetivo (cálculos de previsão sobre todo o conjunto de dados de treinamento). A profundidade máxima da árvore foi limitada em 30; vale ressalvar que não existe nenhuma penalidade referente à complexidade dos modelos.

Outros parâmetros importantes são as taxas dos operadores genéticos. (FREITAS et al., 2014) apresenta um estudo de parâmetros ideais para este problema, sugerindo as taxas de recombinação e mutação iguais a 100%. Geralmente a taxa de mutação é baixa, mas devido à necessidade de diversificar a população, uma taxa mais alta neste problema obteve melhores resultados.

A gramática livre de contexto utilizada neste trabalho possui as seguintes regras de derivação:

$$
\begin{aligned}
\text{<expr>} &::= \text{<expr> <expr> !} \mid \text{<base>} \\
\text{<base>} &::= \text{<base> <base> <op>} \mid \text{<base> <uop>} \mid \text{<var>} \mid \text{<const>} \\
\text{<op>} &::= + \mid - \mid * \mid / \mid \mathit{pow} \\
\text{<uop>} &::= \mathit{log} \mid \mathit{exp} \mid \mathit{sqrt} \\
\text{<var>} &::= x_1 \mid x_2 \mid x_3 \mid x_4 \mid x_7 \mid x_8 \\
\text{<const>} &::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Nota-se pela gramática adotada que as expressões assumem representação pós-fixa e são resolvidas utilizando um autômato de pilha. O operador "!" simboliza o somatório das funções de base para o ajuste dos coeficientes lineares via o Método dos Mínimos Quadrados (MMQ). Para os casos sem o MMQ, as gramáticas apenas não incluem a geração de uma combinação linear de funções de base utilizando o operador "!". Apesar de uma gramática com restrições quanto à forma dos modelos poder ser aplicada, a literatura (BERNARDINO et al., 2011; FREITAS et al., 2014, 2015) mostra que a gramática simples obtém melhores resultados.

Os parâmetros incluídos pela coevolução seguem os sugeridos em (AUGUSTO, 2009), sendo então utilizados 50 competições por ciclo (c) e 20 competições na memória (m).

Foram considerados 3 métodos envolvendo a PGGC neste trabalho. O primeiro

(PGGC$^{(1)}$) faz a busca de forma simples, como descrita no Algoritmo 3. As outras duas abordagens adicionam um passo de otimização utilizando o MMQ para o ajuste dos coeficientes lineares. Uma variante que utiliza o MMQ é rotulada aqui como PGGC$^{(2)}$ e realiza o ajuste dos coeficientes lineares durante o passo de avaliação utilizando os dados selecionados como amostragem. A última técnica de PGG proposta aqui (PGGC$^{(3)}$) faz os mesmos passos de avaliação da PGGC$^{(2)}$, porém um novo ajuste dos coeficientes lineares é feito ao fim da busca sobre o melhor modelo encontrado mas agora utilizando todo o conjunto de dados de treinamento.

4.2 Resultados

As tabelas 4.1, 4.2 e 4.3 mostram os resultados dos experimentos propostos na Seção 4.1, respectivamente, para a PGGC$^{(1)}$, PGGC$^{(2)}$ e PGGC$^{(3)}$.

São exibidos os valores de erro (equação 3.1) do melhor e pior resultados de todas as execuções em relação ao erro de validação, além do valor médio, mediana e desvios padrões. Os valores são apresentados em relação aos conjuntos de treino, validação e teste.

Tabela 4.1: Resultados da PGGC$^{(1)}$

	Menor	Maior	Média	Mediana	Desvio Padrão
Treino	$5,42E-7$	$2,79E-5$	$1,18E-5$	$1,87E-6$	$4,97E-5$
Validação	$4,76E-7$	$1,81E-5$	$7,75E-6$	$1,38E-6$	$3,22E-5$
Teste	$4,89E-7$	$1,72E-5$	$8,43E-6$	$1,78E-6$	$3,06E-5$

Tabela 4.2: Resultados da PGGC$^{(2)}$

	Menor	Maior	Média	Mediana	Desvio Padrão
Treino	$4,47E-8$	$1,19E-7$	$8,60E-8$	$9,00E-8$	$1,91E-8$
Validação	$5,19E-8$	$1,17E-7$	$8,49E-8$	$8,70E-8$	$1,83E-8$
Teste	$5,31E-8$	$2,46E-7$	$1,27E-7$	$1,17E-7$	$4,84E-8$

Tabela 4.3: Resultados da PGGC$^{(3)}$

	Menor	Maior	Média	Mediana	Desvio Padrão
Treino	$4,47E-8$	$1,58E-7$	$8,71E-8$	$9,01E-8$	$2,22E-8$
Validação	$5,19E-8$	$1,45E-7$	$8,58E-8$	$8,70E-8$	$2,04E-8$
Teste	$5,31E-8$	$2,37E-7$	$1,24E-7$	$1,17E-7$	$4,37E-8$

Comparando os resultados obtidos nas tabelas, pode-se ver que a utilização de MMQ traz ganho à acurácia das soluções. Entre as propostas PGGC$^{(2)}$ e PGGC$^{(3)}$, as qualidades dos resultados são muito próximas. Desta forma, pode-se concluir que não há vantagem em realizar o cálculo do MMQ para a população final em PGGC$^{(3)}$. Em todos os 3 casos, pode-se observar que os resultados de treino, validação e teste são muito próximos.

A Tabela 4.4 apresenta os melhores resultados encontrados na literatura para o problema dos dutos. Os algoritmos comparados com a PGGC são:

- Programação Imunológica Gramatical (PIG) (BERNARDINO et al., 2011), baseado em Sistemas Imunes Artificiais e Evolução Gramatical. Sistemas Imunes Artificiais são compostos de metodologias inteligentes inspiradas no sistema imune biológico. O PIG utiliza diferentes gramáticas e o MMQ para no problema dos dutos.

- Programação Genética Gramatical (FREITAS et al., 2014) implementa uma PG utilizando árvores de derivação e GLC. Para o problema dos dutos foram utilizadas diferentes GLC.

- Programação Genética Gramatical Multiobjetivo (FREITAS et al., 2015) expande a PGG em ambiente multiobjetivo na tentativa de minimar erro e complexidade utilizando fronteiras de Pareto. Aplicando diferentes GLC, o MMQ e usando como segundo objetivo a complexidade do modelo medida através da profundidade da árvore esse método obtém os melhores resultados encontrados na literatura.

A Tabela 4.4 é dividida em duas partes, onde a segunda aplica ajuste via MMQ. Na primeira parte pode-se ver que o PGGC obtém resultados compatíveis aos da literatura. Sem utilizar o MMQ, a PGGC se equipara ao PIG e PGG, porém ainda é inferior ao PGGM. Por outro lado, ambas as abordagens que utilizam MMQ têm resultados muito próximos. Esses resultados são equivalentes aos das outras técnicas, obtendo valores melhores do que os alcançados pela PIG, porém inferiores ao PGGM que utiliza gramática simples e o MMQ.

Os melhores modelos encontrados nas 3 abordagens propostas aqui são apresentados nos Apêndices. Para a abordagem PGGC$^{(3)}$ o melhor modelo encontrado é seme-

Tabela 4.4: Resultados obtidos em relação aos dados de teste para $PGG^{(1)}$, $PIG^{(1)}$ e $PGGM^{(1)}$ com Gramática Simples, $PGG^{(2)}$, $PIG^{(2)}$ e $PGGM^{(2)}$ com Gramática Constrita, $PGGM^{(3)}$ com Gramática Simples e $PGGM^{(4)}$ com Gramática Constrita e Mínimos Quadrados e $PIG^{(3)}$ com o melhor resultado com Mínimos Quadrados, $PGGC^{(1)}$, $PGGC^{(2)}$ e $PGGC^{(3)}$.

Algoritmo	**Menor**	**Maior**	**Média**	**Mediana**	**Desvio Padrão**
$PGG^{(1)}$	$3,62E-7$	$5,77E-6$	$1,24E-6$	$1,05E-6$	$7,22E-7$
$PIG^{(1)}$	$6,59E-7$	$1,54E-6$	$1,24E-6$	$1,32E-6$	$2,70E-7$
$PGGM^{(1)}$	$4,63E-7$	$7,71E-7$	$2,71E-7$	$2,37E-7$	$1,34E-7$
$PGG^{(2)}$	$3,66E-7$	$8,25E-6$	$2,03E-6$	$1,76E-6$	$1,27E-6$
$PIG^{(2)}$	$5,80E-7$	$1,89E-6$	$1,13E-6$	$1,08E-6$	$3,24E-7$
$PGGM^{(2)}$	$9,29E-8$	$2,37E-7$	$1,95E-7$	$2,10E-7$	$4,62E-8$
$PGGC^{(1)}$	$4,89E-7$	$1,72E-5$	$8,43E-6$	$1,78E-6$	$3,06E-5$
$PIG^{(3)}$	$1,86E-7$	$2,60E-7$	$2,21E-7$	$2,20E-7$	$1,49E-8$
$PGGM^{(3)}$	$\mathbf{1,33E-8}$	$\mathbf{6,03E-8}$	$\mathbf{3,56E-8}$	$\mathbf{3,59E-8}$	$\mathbf{1,17E-8}$
$PGGM^{(4)}$	$2,33E-8$	$9,92E-5$	$3,42E-6$	$1,03E-7$	$1,80E-5$
$PGGC^{(2)}$	$5,31E-8$	$2,46E-7$	$1,27E-7$	$1,17E-7$	$4,84E-8$
$PGGC^{(3)}$	$5,31E-8$	$2,37E-7$	$1,24E-7$	$1,17E-7$	$4,37E-8$

lhante ao encontrado em $PGGC^{(2)}$; apenas os coeficientes lineares calculados via MMQ são diferentes. Isso se dá pela $PGGC^{(3)}$ utilizar todo o conjunto de dados de treino para os cálculos. No entanto, é importante destacar que isso não gerou uma diferença significativa na qualidade das soluções.

A complexidade dos modelos encontrados é um outro fator importante de ser analisado. A PGGC não impõe penalidades quanto à complexidade dos modelos, assim como foi feito com a PGG. Portanto, não ocorre nenhuma pressão de seleção no sentido de gerar modelos mais simples. Como consequência, verifica-se que o melhor modelo encontrado pela PGGC tem complexidade maior que o melhor modelo encontrado pela PGGM, um algoritmo de busca multiobjetivo que trabalha sobre a acurácia e a complexidade das soluções. Além disso, os modelos alcançados pela PGGM são os melhores entre todas as técnicas utilizadas aqui nas comparações. No entanto, é importante ressaltar que a PGGC possui uma vantagem sobre os demais métodos usados aqui nas comparações: o custo computacional ao se considerar o mesmo número de cálculos da função objetivo. Na PGG são gerados 50 indivíduos por geração e o processo de evolução perdura por 2000 gerações, totalizando 100000 cálculos de função objetivo. A PGGC também gera a mesma quantidade de indivíduos por iteração, e ainda realiza o mesmo número de avaliações de função objetivo, porém um número menor de registros é avaliado. O processo de busca da

PGG envolve o cálculo de valor predito de todos os registros utilizados para treino (neste caso 70% dos dados), totalizando aqui 38900000 avaliações de pontos (2000 gerações x 50 indivíduos x 388 registros). Já a PGGC requer aqui 4500000 avaliações de pontos (2000 iterações x 25 pares de indivíduos gerados x {50 ciclos + 2 avaliações}), cerca de $11,56\%$ do orçamento total da PGGM. Dessa forma o PGGC alcança resultados similares aos da literatura, mesmo com um custo computacional muito menor.

5 Conclusões

Este trabalho segue uma linha de pesquisa sobre a Programação Genética Gramatical com o objetivo de buscar um modelo simbólico para inferir a máxima deformação longitudinal de dutos com amassamento. Verificou-se que técnicas de Programação Genética Gramatical são capazes de (i) representar boas soluções, (ii) obter melhores resultados ao serem combinadas a algum método de ajuste de coeficientes (como o Método dos Mínimos Quadrados), (iii) alcançar modelos generalizáveis e pouco complexos via uma abordagem multiobjetivo e, neste trabalho, (iv) encontrar soluções próximas do estado da arte com orçamento computacional reduzido.

A abordagem coevolutiva da PGG é apresentada aqui e os experimentos computacionais realizados mostram que seus resultados são similares aos da literatura. No entanto, existe uma redução em relação ao número de pontos a serem avaliados quando a ideia de coevolução é adotada, de modo a diminuir o custo computacional. Nos experimentos realizados aqui, observou-se que a PGG foi capaz de obter resultados similares aos disponíveis na literatura com apenas $11,56\%$ das avaliações de modelo.

Finalmente, pretende-se explorar outras frentes envolvendo a PGG, tais como (i) a realização de ajustes de coeficientes não lineares; (ii) a alteração do cálculo da complexidade dos modelos candidatos utilizado na abordagem multiobjetivo; (iii) ajustar o número de avaliações de função objetivo utilizados pela PGGC a fim de utilizar o mesmo custo computacional da PGGM e comparar os resultados obtidos; e (iv) realizar novos testes para ajustar melhor os parâmetros da PGG e suas variações aqui apresentadas.

Bibliografia

ASME B31.8. *Gas transmission and distribution piping systems*. 2007.

AUGUSTO, D. A. *Co-evolução Amostra-Classificador integrada à Programação Genética Gramatical para Classificação de Dados*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, COPPE, 2004.

AUGUSTO, D. A. *Programação Genética Multi-Populacional e Co-Evolucionária para Classificação de Dados*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, COPPE, 2009.

BERNARDINO, H. et al. Inferring strains on a locally deformed pipe via grammar-based immune programming. In: *Proc. of the Iberian Latin American Congress on Computational Methods in Engineering (CILAMCE)*. [S.l.: s.n.], 2011.

BREIMAN, L. Bagging predictors. In: *Machine Learning*. [S.l.: s.n.], 1996.

DARWIN, C. *On the Origin of Species by Means of Natural Selection*. [S.l.]: Murray, 1859. Or the Preservation of Favored Races in the Struggle for Life.

FREITAS, J. de et al. Aplicação de uma programação genética gramatical na inferência da máxima deformação longitudinal de dutos com amassamento. In: *Proc. of the Iberian Latin American Congress on Computational Methods in Engineering (CILAMCE)*. [S.l.: s.n.], 2014.

FREITAS, J. de et al. Aplicação de uma programação genética gramatical multiobjetivo na inferência da máxima deformação longitudinal de dutos com amassamento. In: *Proc. of the Iberian Latin American Congress on Computational Methods in Engineering (CILAMCE)*. [S.l.: s.n.], 2015.

HOLLAND, J. *Adaptation in Natural and Artificial Systems*. [S.l.]: University of Michigan Press, 1975.

KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. [S.l.]: The MIT Press, 1992.

NORONHA, D. et al. Some remarks on the strain based assessment of pipeline dents. In: *Proc. of the Intl. Pipeline Conference*. [S.l.: s.n.], 2008.

PAREDIS, J. Steps towards co-evolutionary classification neural networks. In: *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*. [S.l.: s.n.], 1994.

SCHAPIRE, R. E. The strngth of weak learnbility. In: *Mach. Learn.* [S.l.: s.n.], 1990.

A

Melhor Modelo da PGGC$^{(1)}$

exp(exp(((((2 * x8) - (x4 - (6 - (x7 - 6)))) pow ((sqrt((x4 - sqrt(log(3)))) + x7) - sqrt(x7))) - (8 - x1)))) + (x7 - sqrt(((1 + log(sqrt(sqrt(8)))) + exp((x3 - (x2 * x7)))))) + (((log(x8) + x7) - ((6 / (log(log((3 pow ((x8 - 3) + (5 / (8 + 6)))))) * x1)) * x2)) * (x2 / (log((4 + (x2 pow 5))) - sqrt((3 pow ((x8 - 3) pow (6 / (8 + 6)))))))) + sqrt((x7 - 8)) + sqrt(6) + 8 + exp(log(1)) + ((exp(((4 * (5 pow (sqrt(x3) + sqrt(x7)))) - ((exp(sqrt(1)) / 7) - 6))) - sqrt(1)) - (x7 - (log((9 + (sqrt((log(7) / 3)) + exp(log(9))))) pow sqrt(((4 * (5 pow (sqrt(x3) + sqrt(x7)))) - ((exp(sqrt(1)) / 7) - 6)))))) + sqrt((x2 * sqrt((log(3) pow (((x3 / sqrt(x3)) + sqrt(sqrt(x1))) - (x7 - log(log(log(sqrt(3))))))))))) + log((9 - exp((((((3 - x7) pow x2) - (log(((log(x3) / exp(((6 / (log(log((sqrt(x7) pow ((x8 - 3) + (5 / (8 + 6)))))) * x1)) * x2))) * 9)) - (x2 pow 5))) - exp(exp(((((sqrt(((x7 pow log(x7)) pow 9)) - x7) - 9) / log(sqrt((4 pow 3)))) - (8 pow x1))))) * (x7 - (((exp((sqrt(8) / ((3 - x1) pow x2))) * log((9 - sqrt(log(8))))) + x1) / 5)))))) + sqrt(sqrt(sqrt(x1))) + sqrt(x3) + exp(4) + x7 + (exp(exp(((((exp(9) - x7) - 9) / log(sqrt((4 pow 3)))) - (8 pow x1)))) + (sqrt(((5 + 3) - ((sqrt(x4) pow sqrt(((x2 - (exp(9) + exp((x3 + (log(x1) - (sqrt(exp(log(x7))) pow 1)))))) + (x1 pow sqrt(8))))) + log(((9 - 4) * 5))))) / (x3 * 2))) + ((x3 / (sqrt((((exp(x4) * x4) - 6) / 6)) - x7)) / (4 / (x3 * 4)))

B

Melhor Modelo da PGGC$^{(2)}$ e da PGGC$^{(3)}$

0.171327*3 + -0.0167831*(6 * 8) + -0.105887*sqrt(log((x3 + 6))) + 0.000329687*4 + 1.19886e+11*log(x1) + 0.269585*(exp(((x7 / x2) / x3)) pow sqrt((exp((sqrt(x7) + (x2 / (9 / exp(7))))) + 4))) + 0.00332051*((sqrt((exp((log(((x3 / ((log(6) + (x3 - 3)) - (sqrt(x2) + x2))) - log(sqrt((x3 * exp(x1)))))) - (3 / (log(x4) + (x1 * 7))))) - x3)) * x2) * x3) + -0.00139068*(x8 + (sqrt((exp(x1) / (x2 - (x4 * 4)))) * log(x3))) + -496.852*log(((exp(sqrt(x3)) * 3) * (sqrt(x3) + (exp(7) + x2)))) + -3.86669*(6 / x7) + -0.00140407*x7 + -0.000320674*x3 + 41.1484*(x1 / ((exp(((x3 - log(exp(x2))) pow (x7 + (x7 + 4)))) + x7) / (2 / (7 + 2)))) + -4.74492e-06*((x3 * x7) * x3) + 0.0222432*(x2 / (x8 / x7)) + -1.30044*(log(x8) * exp(((log(x1) + log(log(exp((exp((log(x7) + (log(x2) / (6 / log(8))))) - 8))))) - ((((x3 * exp(log(x8))) + x8) * (sqrt((((x3 * 4) / (x2 + 5)) - ((x1 * x3) / (6 / (sqrt(sqrt(4)) + 6))))) + 3)) / 4)))) + -1.12224e-06*exp((x7 * (((sqrt(6) * 4) / (((log(1) * ((sqrt(1) / ((sqrt(log(sqrt(x3))) * sqrt((1 / x3))) / x3)) + exp(((3 / log(2)) / x7)))) - (((3 * (x8 - (x3 * ((sqrt((x4 + (sqrt(log(9)) + sqrt(7)))) * ((x4 * 4) + 4)) / x3)))) * 4) / x3)) + 2)) + sqrt(log((exp(log(x4)) - exp(x4)))))) + -1.99809e+10*exp(exp(log(sqrt((sqrt((3 - (((exp((log(log(x2)) - exp(7))) / log(4)) + (x3 + 3)) / (6 * (log(log(exp(x7))) + 2))))) + 6))))) + 4963.21*exp(((log(x7) + sqrt(log(sqrt((sqrt((log((x1 / ((sqrt(sqrt(exp(sqrt(2)))) / sqrt(x7)) / x8))) - ((x4 + ((x3 + x1) + 6)) / (3 * (log(log(exp(x7))) + 2))))) + 6))))) - ((x1 * (3 + (x4 * 4))) / x7))) + -1234.99*exp(((x4 * sqrt(sqrt(sqrt(sqrt(sqrt((log((exp (exp((((x8 * 4) / ((x1 + ((sqrt((x8 + (x1 + (x4 * 4)))) * log(1)) - x3)) + 5)) + (2 * 4)))) / 4)) + x3)))))))) - ((sqrt((log(log((x1 - x1))) + x1)) * (x8 + (x4 * 4))) / x3))) + 0.00172059*(exp(log(exp((log((exp((sqrt((exp(((exp ((x4 + log(exp(x8)))) - log(x3)) pow (x7 * 6))) - x7)) pow 5)) - (7 + 9))) + x8)))) / (x4 * log(x4))) + 1612.98*exp(x7)