UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Flávio Andrade Amaral Motta

Hibridização de Programação Genética Gramatical com Estratégia Evolutiva e Evolução Diferencial Aplicada a Problemas de Regressão Simbólica e de Classificação

Juiz de Fora

2018

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Flávio Andrade Amaral Motta

Hibridização de Programação Genética Gramatical com Estratégia Evolutiva e Evolução Diferencial Aplicada a Problemas de Regressão Simbólica e de Classificação

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Heder Soares Bernardino
Coorientador: Itamar Leite de Oliveira

Juiz de Fora
2018

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Motta, Flávio Andrade Amaral.
Hibridização de Programação Genética Gramatical com Estratégia Evolutiva e Evolução Diferencial Aplicada a Problemas de Regressão Simbólica e de Classificação / Flávio Andrade Amaral Motta. -- 2018.
87 f. : il.

Orientador: Heder Soares Bernardino
Coorientador: Itamar Leite de Oliveira
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós Graduação em Ciência da Computação, 2018.

1. Programação Genética Gramatical. 2. Evolução Diferencial. 3. Estratégia Evolutiva. 4. Hibridismo. I. Bernardino, Heder Soares, orient. II. Oliveira, Itamar Leite de, coorient. III. Título.

Flávio Andrade Amaral Motta

Hibridização de Programação Genética Gramatical com Estratégia Evolutiva e Evolução Diferencial Aplicada a Problemas de Regressão Simbólica e de Classificação

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovada em 5 de Julho de 2018.

BANCA EXAMINADORA

Prof. Dr. Heder Soares Bernardino - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Itamar Leite de Oliveira - Coorientador
Universidade Federal de Juiz de Fora

Prof. Dr. Carlos Cristiano Hasenclever Borges
Universidade Federal de Juiz de Fora

Prof. Dr. Luismar Marques Porto
Universidade Federal de Santa Catarina

Prof.ª Dr.ª Jaqueline da Silva Angelo
Laboratório Nacional de Computação Científica

Ao meu anjo Katarina

Sempre presente

AGRADECIMENTOS

Primeiramente agradeço a Deus por me dar força e coragem em todos os momentos.

A minha noiva Ana por todo companheirismo, sabendo me incentivar e motivar todos os dias desse percurso.

Aos meus familiares e amigos por estarem sempre ao meu lado, incentivando e acreditando em meu potencial em momentos que até mesmo eu não acreditava.

Ao meu orientador Prof. Heder Soares Bernardino e ao meu coorientador Prof. Itamar Leite de Oliveira pelo apoio, paciência, dedicação e incentivo ao decorrer deste mestrado.

Aos colegas João Marcos de Freitas e Felipe de Souza pelo apoio no desenvolvimento do trabalho.

Agradeço à CAPES, pelo apoio financeiro, que possibilitou minha estadia em Juiz de Fora para a execução do curso de mestrado.

Aos membros da Banca Examinadora pela avaliação deste trabalho.

E a todos que contribuíram, direta ou indiretamente, na realização deste trabalho.

"And when at last the work is
done Don't sit down, it's time to
dig another one"
Pink Floyd

RESUMO

A Regressão simbólica consiste na manipulação de expressões matemáticas para encontrar uma função que melhor representa um conjunto de dados. Já problemas de classificação podem ser entendidos como o efeito de dispor por classes um conjunto de elementos. Técnicas computacionais foram desenvolvidas para resolver esses tipos de problema, a Programação Genética (PG) é uma delas. Uma vantagem dessa técnica é o fato de produzir modelos simbólicos que são possíveis de serem interpretados. A Programação Genética Gramatical (PGG) surgiu com o uso de gramáticas formais para auxiliar a busca desses modelos. Um problema da PGG é o ajuste de coeficientes, já que apenas valores gerados pela gramática podem aparecer dentro de um modelo. Diferentes formas de hibridização de PGG são utilizadas neste trabalho para ajuste de coeficientes. Evolução Diferencial (ED) e Estratégia Evolutiva (EE) são técnicas de otimização contínua e o objetivo deste trabalho é de gerar melhores soluções do que uma PGG padrão, ao utilizar essas técnicas para realizar o ajuste dos coeficientes. Foram testados 23 problemas de regressão simbólica e 7 de classificação para comparar o desempenho das técnicas propostas com a PGG padrão. Foram encontrados resultados promissores quando comparados a PGG na sua forma padrão.

Palavras-chave: Programação Genética Gramatical. Evolução Diferencial. Estratégia Evolutiva. Hibridismo.

ABSTRACT

Symbolic regression is the manipulation of mathematical expressions to find a function that best represents a dataset. Classification problems can be understood as the effect of arranging by classes a set of elements. Computational techniques were developed to solve these types of problems, Genetic Programming (GP) is one of them. One advantage of this technique is that it produces symbolic models that are easy to interpret. Grammar-based Genetic Programming (GGP) came up with the use of formal grammars to aid in the search for these models. One GGP problem is the coefficient adjustment as only values from the grammar can appear within a model. Different hybridization forms of GGP are used in this work to solve this problem. Differential Evolution (DE) and Evolution Strategy (ES) are continuous optimization techniques, the objective here is to generate better solutions than a standard GGP, by the usage of these techniques to perform the adjustment of the coefficients. We tested 23 symbolic regression and 7 classification problems to compare the performance of the proposed techniques with the standard GGP. Promising results compared to standard GGP were found.

Keywords: Grammar-based Genetic Programming. Differential Evolution. Evolution Strategies. Hibrydsm.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

AsC Área sob a Curva

ED Evolução Diferencial

EE Estratégia Evolutiva

NAN *Not a number*

OEP Otimização por Enxame de Partículas

PD Perfis de Desempenho

PG Programação Genética

PGC Programação Genética Cartesiana

PGGEE Programação Genética Gramatical com Estratégia Evolutiva

PGGED Programação Genética Gramatical com Evolução Diferencial

PGG Programação Genética Gramatical

PGL Programação Genética Linear

SUMÁRIO

1 INTRODUÇÃO

Problemas de otimização são resolvidos diariamente por qualquer tipo de pessoa. Encontrar uma forma de solucionar esses problemas tendo o menor custo possível é desejável. Um problema de otimização de minimização pode ser descrito matematicamente como

Otimizar

$$f(x) \tag{1.1}$$

Sujeito a

$$g(x_i) \leq 0 \tag{1.2}$$

$$h(x_i) = 0 \tag{1.3}$$

$$x_{iMin} \leq x_i \leq x_{iMax} \tag{1.4}$$

onde deseja-se minimizar ou maximizar a função $f(x)$, geralmente é utilizada a minimização. O vetor de x representa o conjunto de soluções, sendo ele considerado ótimo quando a melhor resposta para um x válido é encontrado. O índice i representa as dimensões do problema, sendo ele limitado ao número de variáveis presente.

Diversos problemas podem ser interpretados como sendo de otimização. Como exemplo:

- Escolher a ordem de execução de tarefas de compromissos em um dia;
- Otimizar os gastos em um supermercado levando em consideração a relação custo x benefício;
- Conseguir fazer uma relação para qual estrada usar levando em conta qualidade da estrada e tempo de uma viagem;
- Escolher a melhor forma de distribuir objetos dentro de um caminhão; e
- Planejar uma construção civil com menor uso de materiais.

Modelar esses problemas para serem resolvidos por um computador agiliza esse processo, principalmente pela quantidade de possibilidades que podem ser avaliadas em um curto período de tempo. Diversas formas de resolver problemas de otimização são estudadas na computação. Várias formas de resolver problemas podem ser obtidas quando

observados alguns comportamentos na natureza. A natureza é uma forte fonte de inspiração para a construção de soluções em sistemas computacionais (LEWIS *et al.*, 2016). Tem-se como exemplo algumas técnicas de solução de problemas baseadas na natureza:

- Algoritmo genético;
- Programação genética;
- Colônia de formigas;
- Enxame de partículas;
- Redes perceptron.

Programação Genética (PG) é uma proposta baseada na teoria Darwiniana da evolução das espécies (DARWIN, 1859) e utiliza o conceito de indivíduos que ao longo do tempo evoluem, procriam e geram proles mais aptas a resolver problemas. A PG foi desenvolvida inicialmente por Koza (1992) para realizar a construção automática de programas em uma certa linguagem computacional. Um ponto a se destacar é que para realizar essa construção não é necessário conhecimento prévio do problema.

Com diversos estudos sendo realizados utilizando PG, algumas técnicas foram surgindo. A Programação Genética Gramatical (PGG) foi implementada quase simultaneamente por três grupos de pesquisa independentes. Whigham (1995) propos o (*Context-free Grammar Genetic Programming* CFG-GP), o qual utiliza uma gramática livre de contexto para restringir diversas operações da CFG-GP. Geyer-Shulz (1997) sugeriu um sistema muito similar ao anterior, em que a maior diferença é o algoritmo utilizado para inicializar a população. Wong e Leung (1997) propuseram o LOGENPRO, que utiliza PROLOG (*Definite Clause Grammars* DCG) para gerar programas. Gramáticas são utilizadas para restringir sintaticamente o uso de termos em uma linguagem (MüLLER, 2016). Por meio de uma gramática, é possível direcionar a busca feita pela Programação Genética.

Duas utilizações comuns de PG e suas vertentes são a classificação e a regressão simbólica. O uso da técnica se deve à possibilidade de extrair conhecimento do modelo obtido, sendo útil para um especialista entender o problema a ser resolvido. A classificação é um processo de identificação de uma instância para determinar a qual grupo este pertence (UMADEVI; MARSELINE, 2017). Utilizando um conjunto de dados observados,

deve-se aprender como atribuir uma instância a um grupo com base em suas características.

Em estatística, a análise de regressão é um conjunto de processos utilizados para descobrir o relacionamento entre variáveis (DRAPER; SMITH, 2014). Regressão simbólica é um tipo de análise de regressão a qual realiza uma busca no espaço das expressões matemáticas em forma simbólica para encontrar um modelo que melhor se adequa aos dados disponíveis (KEIJZER, 2008).

As aplicações para ambos tipos de problemas podem ser das mais diversas. Modelos encontrados em classificação podem ser utilizados para traçar perfis sócio-econômicos de cidadãos, ou até mesmo indicar se uma compra de um usuário pode ser uma fraude. Na regressão simbólica, descobrir o comportamento de uma função a partir dos dados disponíveis pode ser utilizado para qualquer aplicação que seja modelada matematicamente.

A tarefa de encontrar coeficientes numéricos de um problema de regressão simbólica ou de classificação pode ser difícil, principalmente quando o problema não envolve apenas coeficientes inteiros (EVETT; FERN, 1998). Realizar diversas operações aritméticas é uma forma de superar essa limitação. Outra forma de alcançar valores não inteiros é incluindo-os na gramática que rege o algoritmo. Devido ao alto número de problemas compostos por valores inteiros e reais, outra forma de determinar uma aproximação adequada é desejada.

Portanto, o uso da Programação Genética com outras técnicas é muito comum, como pode ser visto em (EL-MIHOUB *et al.*, 2006). Nesse trabalho foi realizado um levantamento sobre diferentes formas de hibridização da Programação Genética. Diversas formas de integração com outros algoritmos de busca e otimização foram estudadas. Também são apresentados os problemas utilizados. Foi concluído que a hibridização é uma boa abordagem para solução de problemas usando Programação Genética.

Evolução Diferencial (ED) foi introduzida por Storn e Price (1995) e foi desenvolvida para resolver um problema de ajuste polinomial de Chebychev. Em 1996, participou da primeira competição internacional em computação evolutiva, conquistando o terceiro lugar geral (BERSINI M. DORIGO, 1996). É uma técnica poderosa, apesar de sua simplicidade, principalmente para realização de otimização contínua, pois opera naturalmente sobre os números reais (STORN; PRICE, 1997). Plagianakos *et al.* (2008) fizeram um levantamento sobre o desempenho do algoritmo nas principais áreas de aplicação, mostrando sua

eficiência.

A Estratégia Evolutiva (EE) foi desenvolvida por Eigen (1973). Desenvolvida inicialmente para otimização em mecânica de fluidos, recebeu então um foco maior para funções reais em geral (FOGEL, 2005).

Com o objetivo de realizar ajuste de coeficientes em problemas de regressão simbólica, Motta *et al.* (2017) utilizaram um hibridismo entre PGG e ED. A ED foi utilizada apenas durante a evolução da PGG. A pesquisa realizada resultou no trabalho (MOTTA *et al.*, 2018) em que atacou-se o mesmo conjunto de problemas, porém utilizando também a EE para realizar esse ajuste de coeficientes.

O objetivo é gerar melhores soluções com uma PGG via hibridização, principalmente para problemas que utilizam coeficientes contínuos. Para isso, abordagens de hibridização de PGG com ED e EE são propostas para melhorar os coeficientes numéricos dos modelos obtidos pela PGG.

Para a regressão simbólica, uma análise do desempenho nas funções sugeridas por Nicolau *et al.* (2015) foi feita. Essas expressões abrangem diversos tipos de comportamentos de função, sendo um conjunto de testes bem diversificado. Também foram utilizadas as funções propostas por Motta *et al.* (2017), que sugeriram diversas funções de apenas uma dimensão. Para aplicação em problemas de classificação, foram utilizadas algumas das bases mais conhecidas da literatura (BUSCEMA *et al.*, 2010; CHARYTANOWICZ *et al.*, 2010; FISHER, 1936; SIGILLITO *et al.*, 1989; MANGASARIAN *et al.*, 1990). São conjuntos que representam dados dentre diversas aplicações, que vão de conjunto com características de flores até descrição de pacientes diagnosticados com câncer de mama. Foi então realizada uma comparação das técnicas propostas com a PGG em sua forma padrão.

2 REVISÃO DA LITERATURA

Alguns trabalhos para realizar busca de coeficientes usando Programação Genética (PG) podem ser encontrados na literatura.

Howard e D'Angelo (1995) exploraram um hibridismo entre PG e Algoritmos Genéticos (AG) para realizar regressão simbólica, utilizando PG para evoluir as expressões enquanto de forma simultânea o AG foi utilizado para evoluir os coeficientes utilizados na expressão. Com este trabalho, os autores perceberam a necessidade de um alto número de dados para treinamento, já que os indivíduos estavam apenas super ajustando os dados e no momento de testá-los estavam com baixo desempenho. Com maior número de dados para treinamento foi percebida uma melhora em relação a uma PG tradicional.

Funie *et al.* (2014) abordaram o problema de estratégias para alta frequência de movimentação de ações. Os indivíduos da PG foram modelados como uma árvore binária, sendo verdadeiro (1) (compra) e falso (2) (venda). Logo após a etapa de avaliação dos indivíduos em uma PG, o algoritmo proposto naquele trabalho executa uma Otimização por Enxame de Partículas(OEP) nos melhores indivíduos da população. No próximo passo, todos os indivíduos que foram alterados pelo OEP são submetidos aos operadores genéticos (cruzamento e mutação) e depois formam a nova população junto aos que não foram alterados. Foi concluído que o algoritmo ajuda a identificar e analisar o comportamento do mercado assim como identificar regras de movimentação que podem causar mudanças significativas.

Alonso *et al.* (2009) sugeriram o uso da Estratégia Evolutiva (EE) para atacar o problema de geração de constantes para uma PG. Assim que finalizada a evolução da PG, a EE é utilizada para determinar os melhores valores das constantes encontradas para problemas de regressão simbólica. Para análise da eficiência da técnica, foram utilizados seis problemas de regressão simbólica. O principal parâmetro avaliado pelo trabalho foi a porcentagem do processamento computacional que cada uma das técnicas deve ocupar, foram utilizados a PG comum, assim como o uso da EE com 25%, 50%, 75% do orçamento. Com o algoritmo hibrido desenvolvido, foi possível perceber que a proposta que designa 25% do orçamento para a EE foi a de melhor resultado, por ser superior em 4 das seis funções testadas.

Hibridização da Evolução Diferencial (ED) com outras técnicas pode ser facilmente encontrada na literatura, porém apenas alguns trabalhos explorando a combinação com PG foram feitos. Roy *et al.* (2012) sugerem melhorar a inicialização do vetor da ED com o uso de uma PG. Para isso a PG utiliza um método de aprendizado para gerar diversas estratégias de inicialização para a ED. É concluído que a utilização dessa técnica é útil principalmente pela fácil adaptação a diferentes problemas. Os resultados apresentados mostram uma melhoria considerável se comparado a uma ED comum.

Neste trabalho é proposta a utilização de uma PGG hibridizada com outra técnica de otimização para realizar o ajuste dos coeficientes. Como opções foram testadas tanto ED como EE.

O método proposto aqui se difere dos demais por (i) adaptar a gramática durante a evolução, adicionando as constantes encontradas a ela (ii) utilizar uma PGG ao invés de uma PG tradicional (iii) realizar um comparativo tanto em regressão simbólica como em classificação (iv) utilizar tanto Estratégias Evolutivas como Evolução Diferencial em conjunto com a PGG

3 FUNDAMENTOS

Para que todos os conceitos do desenvolvimento deste trabalho fiquem claros, este capítulo faz uma introdução do conteúdo utilizado. Através de exemplos e explicações mais aprofundadas do conteúdo, os conceitos a serem introduzidos aqui, são: Programação Genética (PG), Programação Genética Gramatical (PGG), Evolução Diferencial (ED), Estratégia Evolutiva (EE), regressão simbólica e classificação.

3.1 PROGRAMAÇÃO GENÉTICA

A evolução das espécies é uma teoria sugerida por Darwin (1859) que propõe explicar como uma população evolui ao longo do tempo. Segundo Darwin, a evolução de uma espécie ocorre por meio de processos que agem sobre a população. Esses processos são denominados como reprodução, mutação, competição e seleção. Mutação e reprodução são processos genéticos que afetam diretamente a forma como cada indivíduo é composto, também descritos como os operadores responsáveis pela variabilidade dos indivíduos de uma população. Na competição e na seleção, os organismos mais bem adaptados ao meio têm maiores chances de sobrevivência do que os menos adaptados, deixando um número maior de descendentes.

Com base nessa teoria e baseado no trabalho de Holland (1975), foi implementada uma técnica denominada Programação Genética (PG) (KOZA, 1992). A PG é uma técnica de computação evolutiva que visa automatizar a construção de programas de computador. Utilizando uma população de programas de computador, a técnica busca otimizar os indivíduos avaliando-os através de uma função objetivo.

Problemas de aprendizado de máquina supervisionado são uma das mais comuns aplicações dessa técnica, o qual programas representam modelos que mapeiam entradas em saídas. É possível encontrar aplicações de Programação Genética em diversos problemas de otimização, como indicado por (LANGDON *et al.*, 2008).

O Algoritmo 1 representa um pseudo-código da PG, do qual uma população inicial é criada aleatoriamente, no primeiro laço ocorre a substituição de indivíduos, no laço mais interno os filhos são gerados.

Na Programação Genética alguns parâmetros devem ser definidos antes da execução,

Algoritmo 1: Pseudo-código da Programação Genética (BROWNLEE, 2012)

```
Programação Genética (TP, MUT, CR, P)
    Entradas: Número máximo de avaliações de função objetivo P, Tamanho da
              população TP, Fator de cruzamento CR, Fator de mutação MUT
    Saída   : Indivíduo best
    início
        Pop ← CriarPopulaçãoInicialAleatória(TP);
        Pop_tmp ← ∅;
        Avaliar Pop;
        while P não é atingido do
            while |Pop_tmp| < TP do
                Selecionar p_1 e p_2 pertencentes a Pop por torneio;
                Copiar p_1 para p'_1 e p_2 para p'_2;
                aleatorio ← numeroAleatorio(0, 1);
                if aleatorio < CR then
                    Aplicar cruzamento em p'_1 e p'_2;
                end if
                if aleatorio < MUT then
                    Aplicar mutação em p'_1 e p'_2;
                end if
                Avaliar p'_1 e p'_2;
                Inserir p'_1 e p'_2 em Pop_tmp;
            end while
            Pop ← elite(Pop) ∪ elite(Pop_tmp);
            Esvaziar Pop_tmp;
        end while
        best ← melhorInd(Pop);
        return best
    fim
```

como, operadores e operandos, inicialização da população, cálculo de função objetivo, critério de parada, seleção, mutação, cruzamento e substituição da população.

3.1.1 DEFINIÇÃO DOS OPERADORES E OPERANDOS

A definição dos operadores e operandos é crucial pois é a partir deles que todo indivíduo da população será construído. Os elementos mais comuns encontrados são: estruturas condicionais (IF - ELSE, SWITCH), estruturas de repetição (FOR, WHILE, DO WHILE), constantes (4.3, π, -2), variáveis (a, b, x) , operadores relacionais ($\leq, \geq$) e operadores aritméticos (+, -, *, $\sqrt{}$).

3.1.2 REPRESENTAÇÃO DOS INDIVÍDUOS

Em PG os indivíduos da população são comumente representados por árvores (POLI *et al.*, 2008). Por exemplo, a Figura 3.1 representa a expressão $2 * (y - x)$. As variáveis e constantes dos programas são as folhas das árvores, na PG esses elementos são denominados terminais. As operações que podem aparecer em um indivíduo ficam nos nós internos e são chamadas de funções.

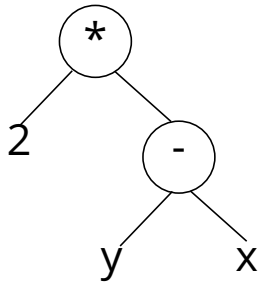


Figura 3.1: Exemplo de árvore que representa a expressão $2 * (y - x)$. Adaptado de (POLI *et al.*, 2008).

Comumente na PG as expressões são representadas na forma prefixada, de forma similar a utilizada em LISP (LANGDON *et al.*, 2008). Por exemplo, a expressão $2 * (y - x)$ ficaria $(* \ 2 \ (- \ y \ x))$. Essa notação facilita a visualização da relação entre as sub-expressões e suas sub-árvores correspondentes.

3.1.3 INICIALIZAÇÃO DOS INDIVÍDUOS

O principal desafio da inicialização de população em uma PG é prover o máximo de diversidade possível. Para lidar com esse conceito algumas técnicas são sugeridas e as

mais populares são: *Full*, *Grow* e *Ramped half-and-half* (POLI *et al.*, 2008).

Na inicialização utilizando a técnica *full* um número é escolhido aleatoriamente e então a árvore é gerada cheia até esta altura escolhida, como na Figura 3.2. Esse processo é repetido para cada indivíduo da população

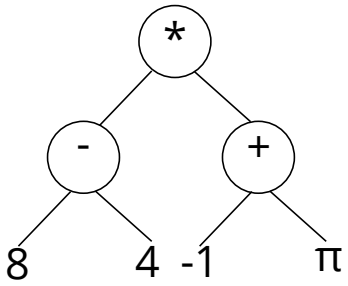


Figura 3.2: Árvore gerada através do método *full* com altura = 3

Para a técnica *grow* (ver Figura 3.3) os nós da árvore são derivados de forma aleatória. Com isso um indivíduo pode tomar diferentes formas.

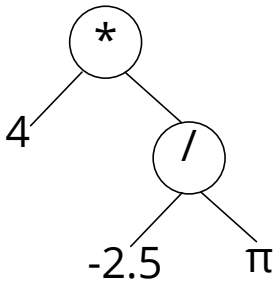


Figura 3.3: Árvore gerada através do método *grow*

Na *Ramped half-and-half* a cada indivíduo que for gerado, um sorteio entre as duas técnicas anteriores é realizado, dessa forma cada indivíduo pode ser gerado de qualquer uma das duas formas, o pseudo-código para esse método é apresentado no Algoritmo 2.

O cálculo da aptidão de um indivíduo varia de acordo com o problema a ser tratado. Nas Seções 3.5 e 3.6 esses cálculos são explicados com mais detalhes.

3.1.4 CRITÉRIO DE PARADA

O critério de parada então não pode sempre se basear na busca da solução exata, com isso alguns outros critérios de parada são usualmente utilizados, como: aptidão em uma faixa aceitável, número máximo de gerações, número máximo de avaliações da função objetivo, tempo de execução e estagnação do processo evolutivo (POLI *et al.*, 2008).

Algoritmo 2: Pseudo-código para *Ramped half-and-half* que gera a população inicial da PG

```
RampedHH (TP)
    Entradas: Tamanho da população TP
    Saída   : População inicial Pop
    início
        Pop ← ∅;
        for i ← 0 to k do
            aleatorio ← numeroInteiroAleatorio(1, 2);
            if aleatorio == 1 then
                Pop_i ← GerarFull(i);
            end if
            else
                Pop_i ← GerarGrow(i);
            end if
        end for
    fim
    return Pop
```

No caso de solução exata, a busca deve ser interrompida apenas quando um indivíduo da população representa a solução exata para o problema. A aptidão dentro de uma faixa aceitável é utilizada quando se sabe o quão preciso o modelo deve ser para atender a necessidade da aplicação. Sua desvantagem é o fato de que o processo é encerrado assim que atinge essa faixa, porém há a possibilidade da aptidão melhorar se algumas gerações a mais forem usadas. A estagnação do processo evolutivo finaliza a execução quando não encontra melhorias por algumas gerações. Seu problema é que o processo evolutivo pode demorar para realmente estagnar fazendo evoluções pequenas, ou até mesmo estagnar por apenas algumas gerações mas sairia desse mínimo local com mais algumas gerações.

A parada através de um número máximo de gerações é muito utilizada quando duas técnicas distintas são comparadas. O problema é que algumas gerações podem fazer diferença entre duas técnicas. Assim como a parada pelo número máximo de gerações, a parada pelo número máximo de avaliações da função objetivo é utilizada quando duas técnicas são comparadas. Tendo como vantagem ser mais adequada para técnicas híbridas que consomem mais avaliações durante suas gerações. A utilização do tempo de execução como critério de parada é útil quando se tem um tempo predeterminado para um sistema. Como a execução é encerrada assim que o tempo é atingido então pode acontecer do processo estar num momento crítico da evolução. Também é afetada pelas configurações do computador, assim como pela habilidade do programador.

3.1.5 SELEÇÃO

Na seleção natural Darwiniana, os melhores indivíduos tendem a procriar com maior frequência. Algumas técnicas para reproduzir esse comportamento são: proporcional a aptidão (roleta), *ranking* e torneio.

Para realizar a seleção proporcional a aptidão, deve-se realizar a soma total da aptidão da população e então aplicar uma divisão da aptidão individual pelo total. Esse valor obtido é proporcional a probabilidade de ele ser selecionado.

A seleção por *ranking* surgiu como alternativa para a seleção por roleta (GREFENSTETTE; BAKER, 1989). A principal diferença entre elas, é que ao invés de utilizar a aptidão do indivíduo para calcular sua probabilidade de ser selecionado, utiliza-se a sua posição em uma lista ordenada do pior para o melhor em função da aptidão.

A seleção por torneio consiste em realizar repetidos torneios entre alguns indivíduos escolhidos aleatoriamente. O vencedor de cada uma dessas seleções é selecionado para o cruzamento. De acordo com o tamanho do torneio, a pressão pode ser alterada. O Algoritmo 3 representa uma seleção por torneio.

Algoritmo 3: Pseudo-código de uma seleção por torneio

```
Torneio (Pop, TP, n, k)
    Entradas: População Pop, Tamanho da População TP, Tamanho do torneio n,
              Número de indivíduos a ser selecionado k
    Saída   : População depois da seleção Pop'
    início
        PopTemporaria ← ∅;
        Pop' ← ∅;
        for i ← 0 to k do
            PopTemporaria ← n indivíduos aleatórios da população Pop;
            Pop'∪ melhorIndividuo(PopTemporaria);
        end for
    fim
```

3.1.6 CRUZAMENTO

Ao menos dois pais são necessários para realizar o cruzamento. Logo após a seleção desses indivíduos, um ou mais filhos são gerados. Para gerar dois indivíduos o cruzamento seleciona aleatoriamente uma sub-árvore de um pai e realiza a troca com uma sub-árvore do outro pai. Sempre respeitando a construção da árvore, ou seja os novos indivíduos não

podem ser infactíveis. Essa operação pode ser vista na Figura 3.4.

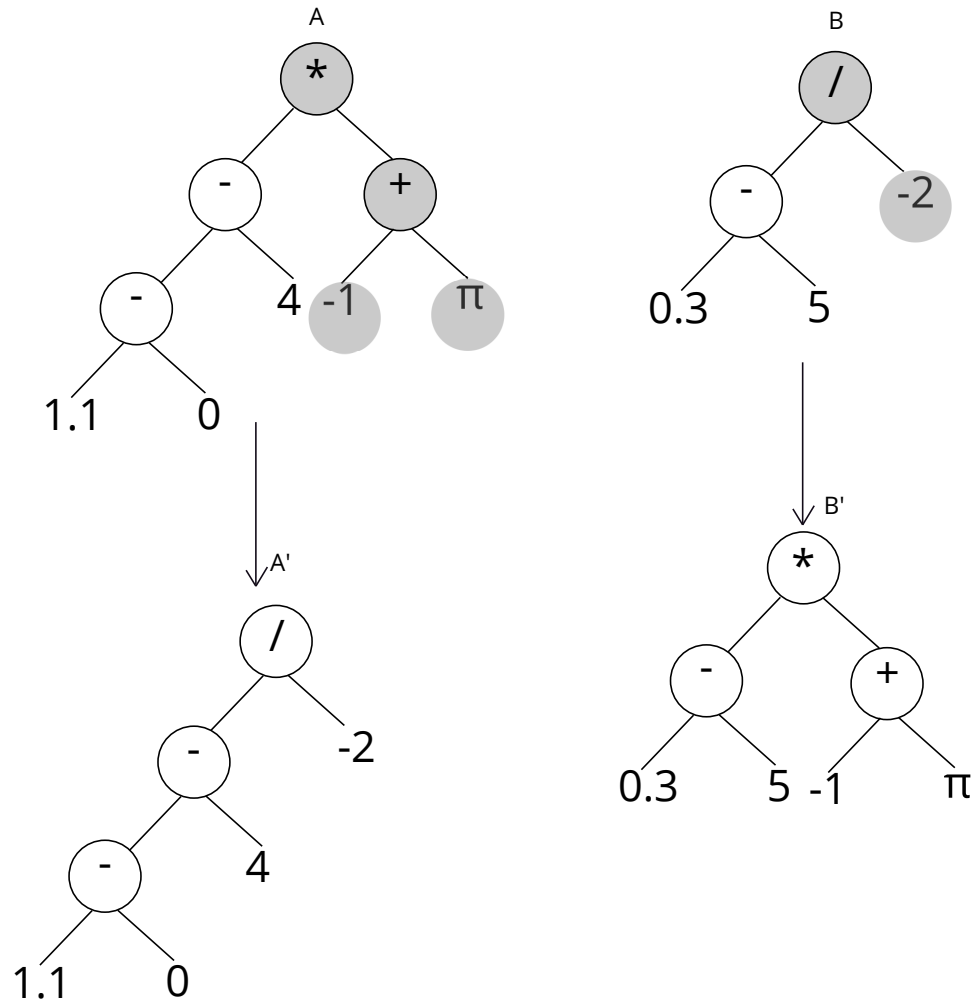


Figura 3.4: Exemplo de operador de cruzamento de uma PG gerando dois filhos. A e B são os pais e A' e B' correspondem aos filhos.

3.1.7 MUTAÇÃO

A mutação é o operador genético responsável por introduzir e manter diversidade da população de uma geração para outra. É o operador responsável pela parte de exploração do espaço de busca. Existem diversas formas de realizar a mutação, a mutação por encolhimento reduz o tamanho do indivíduo por trocar um nó pai por um nó filho. Pode-se também substituir um nó aleatório e somente ele por um novo, chamada de mutação do alelo. Também existe a mutação por permutação, o que ocorre é a troca de duas sub-árvores selecionadas aleatoriamente dentro de um mesmo indivíduo.

A mutação canônica como representada na Figura 3.5 substitui uma sub-árvore aleatória por uma completamente nova.

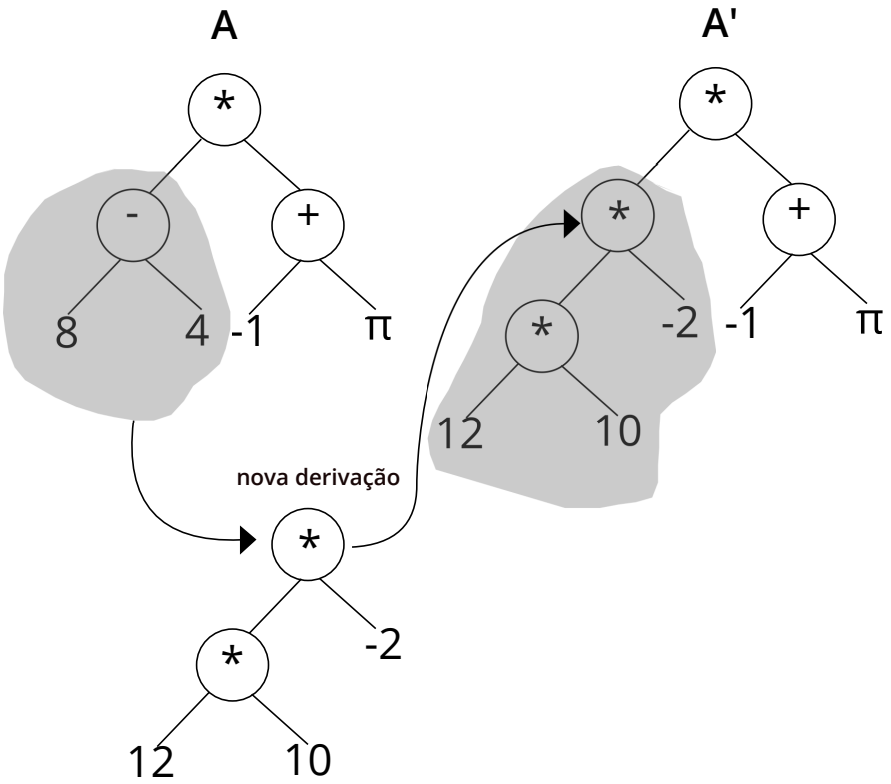


Figura 3.5: Exemplo de mutação canônica. A é um indivíduo da população e A' é obtido pela substituição da nova derivação no nó indicado.

Nesse momento do algoritmo diversos novos indivíduos foram gerados. Como a população tem de respeitar um limite máximo, deve ser realizada a substituição de alguns elementos da população.

3.1.8 CRIAÇÃO DE CONSTANTES

Koza (1992) relata o problema da Programação Genética em conseguir gerar constantes para um modelo. Uma alternativa comum para atacar esse problema é a geração aleatória, da qual uma constante é gerada aleatoriamente dentro de uma faixa predefinida de valores. Poli *et al.* (2008) relata que essa alternativa pode fazer com que o comportamento de um indivíduo varie sempre que chamado, mesmo utilizando parâmetros iguais de entrada.

Alguns trabalhos propõem métodos matemáticos como busca local gradiente para realizar essa busca (TOPCHY; PUNCH, 2001). Enquanto outros utilizam técnicas evolutivas para gerar e otimizar esses valores (ALONSO *et al.*, 2009).

3.1.9 ALGUMAS VERTENTES DA PROGRAMAÇÃO GENÉTICA

Com a percepção do poder da PG, diversas vertentes surgiram ao longo do tempo. A Programação Genética Cartesiana (CGP) surgiu para realizar a evolução de circuitos digitais desenvolvido por Miller *et al.* (1997). O termo "Programação genética cartesiana" apareceu primeiro em (MILLER, 1999) e foi posposto como uma forma de Programação Genética por Miller e Thomson (2000). Denominado como CGP pois representa um programa utilizando uma grade bi-dimensional de nós.

Na Programação Genética Linear (PGL) os programas são representados como uma sequencia de instruções de linguagem de programação imperativa. Banzhaf (1993) realizou uma abordagem linear e Nordin (1994) utilizou esses conceitos, operando diretamente em representação imperativa.

Gramáticas são utilizadas como formas de representação de estruturas na ciência da computação. Com todo seu poder de restringir estruturas, se tornaram uma ferramenta importante também na Programação Genética. Os primeiros sistemas de Programação Genética baseados em gramatica surgiram no meio dos anos 1990 (WONG; LEUNG, 1995b,a; WHIGHAM, 1995; GEYER-SHULZ, 1997; WONG; LEUNG, 1997) e rapidamente seu uso cresceu.

3.2 PROGRAMAÇÃO GENÉTICA GRAMATICAL

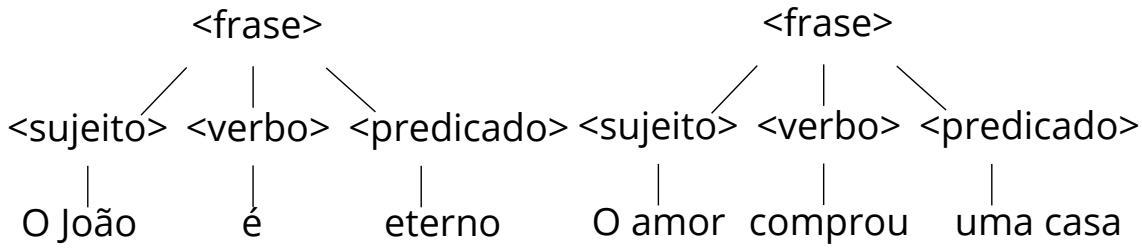
Na formalização de gramáticas proposta por (CHOMSKY, 1956, 1957), uma gramática G consiste dos seguintes componentes: Um conjunto finito N de símbolos não terminais, um conjunto finito Σ de símbolos terminais, um conjunto finito P de regras de produção sendo cada uma da forma: $(\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*$ onde * é o símbolo para o operador fecho de Kleene e $\cup$ denota o conjunto união. Isto é, cada regra de produção mapeia de um conjunto de símbolos para outro. Uma gramática é formalizada então como a tupla (N, Σ, P, S).

Na PGG todo indivíduo é gerado por meio de uma gramática formal. Uma de suas principais utilidades é a de limitar expressões simbólicas sintaticamente. Isso pode ser tanto para determinar expressões válidas como para reforçar restrições de tipo (HOPCROFT J.E.; MOTWANI, 2014).

Na língua portuguesa há restrição pela gramática regente. Por exemplo, a estrutura

básica de uma frase é constituída por sujeito, verbo e então predicado, assim, tem-se um exemplo utilizando gramática formal.

$$< frase > ::= < sujeito >< verbo >< predicado >$$
$$< sujeito > ::= O\ João \mid A\ Maria \mid O\ amor$$
$$< verbo > ::= caiu \mid comprou \mid é$$
$$< predicado > ::= uma\ casa \mid eterno(a)$$



(a) Derivação 1 (b) Derivação 2

Figura 3.6: Exemplo de duas derivações diferentes para a mesma gramática

Por mais que a frase "O amor comprou uma casa" não seja semanticamente correta por não haver sentido, sua estrutura sintática respeita as regras gramaticais regentes sob a língua portuguesa, portanto é considerada sintaticamente correta.

A junção da PG com gramática recebe o nome de Programação Genética Gramatical (PGG). Whigham (1995) utilizou gramática livre de contexto como a estrutura regente para um indivíduo de uma PG.

PGG é uma técnica de PG que utiliza gramática formal para limitar as expressões geradas durante a busca. Assim sendo, tem algumas vantagens, como, (i) aplicação mais limpa de restrições semânticas, (ii) espaço de busca reduzido, (iii) introdução de viés, (iv) possibilidade de extensão como operadores genéticos adicionais e (v) evolução da própria gramática (MCKAY *et al.*, 2010). Por outro lado, seu conceito é mais complexo e sua implementação mais trabalhosa.

A maioria dos elementos de uma PG são preservados em uma PGG. Por exemplo, o procedimento de seleção não depende do tipo de representação e pode ser o mesmo utilizado na versão padrão da PG.

Algoritmo 4: Pseudo-código de uma Programação Genética Gramatical

```
ProgramaçãoGenéticaGramatical
    Entradas: Número limite de avaliações de função objetivo P, Tamanho da
              população TP, Fator de cruzamento CR, Fator de mutação MUT,
    início
        Criar população inicial Pop de tamanho TP;
        Inicializar Pop_tmp vazio;
        Avaliar (Pop);
        while P não atingido do
            while |Pop_tmp| < TP do
                Selecionar p_1 e p_2 por torneio;
                Copiar p_1 para p'_1 e p_2 para p'_2;
                if aleatorio < CR then
                    Aplicar cruzamento em p'_1 e p'_2;
                end if
                if aleatorio < MUT then
                    Aplicar mutação em p'_1 e p'_2;
                end if
                Avaliar p'_1 e p'_2;
                Inserir p'_1 e p'_2 em Pop_tmp;
            end while
            pop ← elite de Pop ∪ elite de Pop_tmp;
            Esvaziar Pop_tmp;
        end while
    fim
```

Cada indivíduo continua sendo representado por uma árvore. No entanto na PGG eles são árvores de derivação. As árvores de derivação são compostas por sentença e símbolos não terminais derivados de uma gramática formal regente como visto nas Figuras 3.7 e 3.8. Em uma árvore de derivação, o indivíduo gerado é sintaticamente correto no que diz respeito às regras adotadas pela gramática. Uma expressão candidata pode ser lida pelas suas folhas na notação de pós ordem. Portanto, pode-se perceber que todas as folhas são símbolos terminais da gramática.

Para realizar o cruzamento na PGG os conceitos da PG são aproveitados. Assim como na forma original, os nós selecionados em ambos indivíduos devem respeitar a composição sintática da gramática, ou seja, ao escolher um nó interno de um pai, o cruzamento só pode ser realizado com outra árvore que contenha um nó não terminal de derivação coincidente.

Dois indivíduos (pais) geram dois filhos. Para realizar essa operação uma sub-árvore é escolhida de cada pai e a prole é criada com características de ambos pais, como mostrado na Figura 3.7.

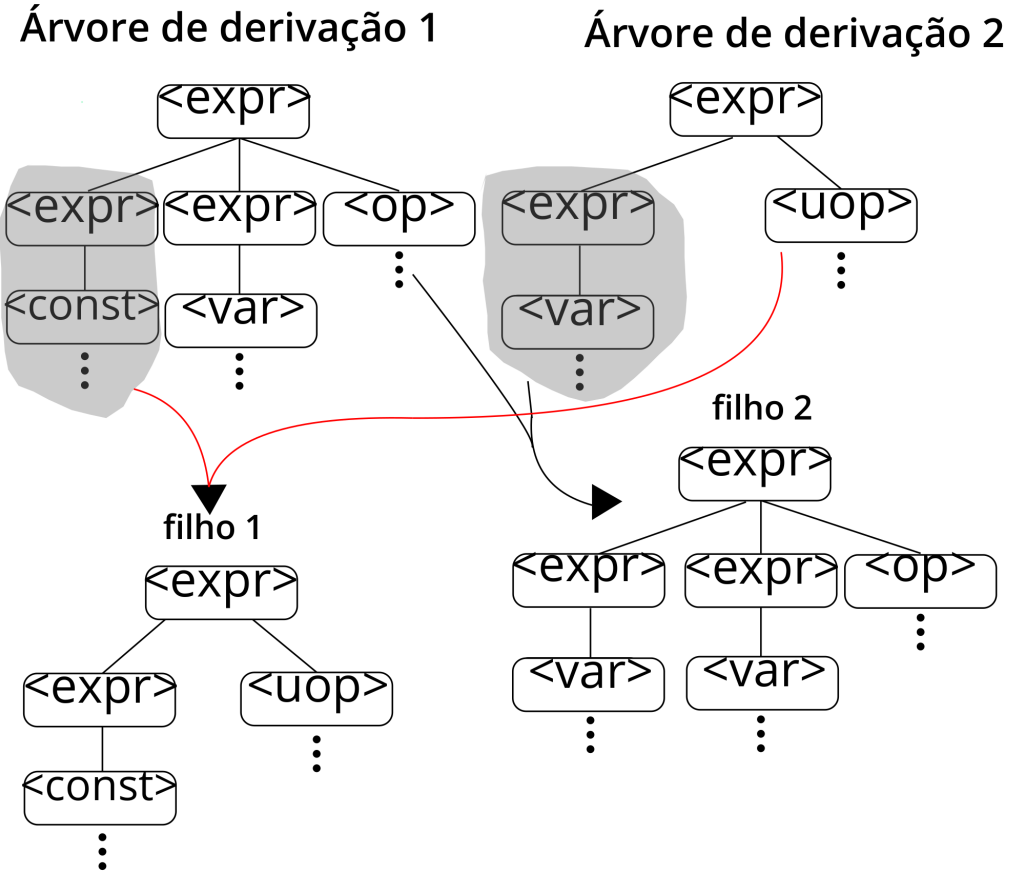


Figura 3.7: Exemplo de operador de cruzamento de uma PGG

A mutação é uma perturbação ocorrida em um indivíduo. Primeiro um nó não terminal

é escolhido aleatoriamente. Então uma sub-árvore é criada realizando outra derivação a partir da gramática começando do símbolo não terminal correspondente, como mostrado na Figura 3.8.

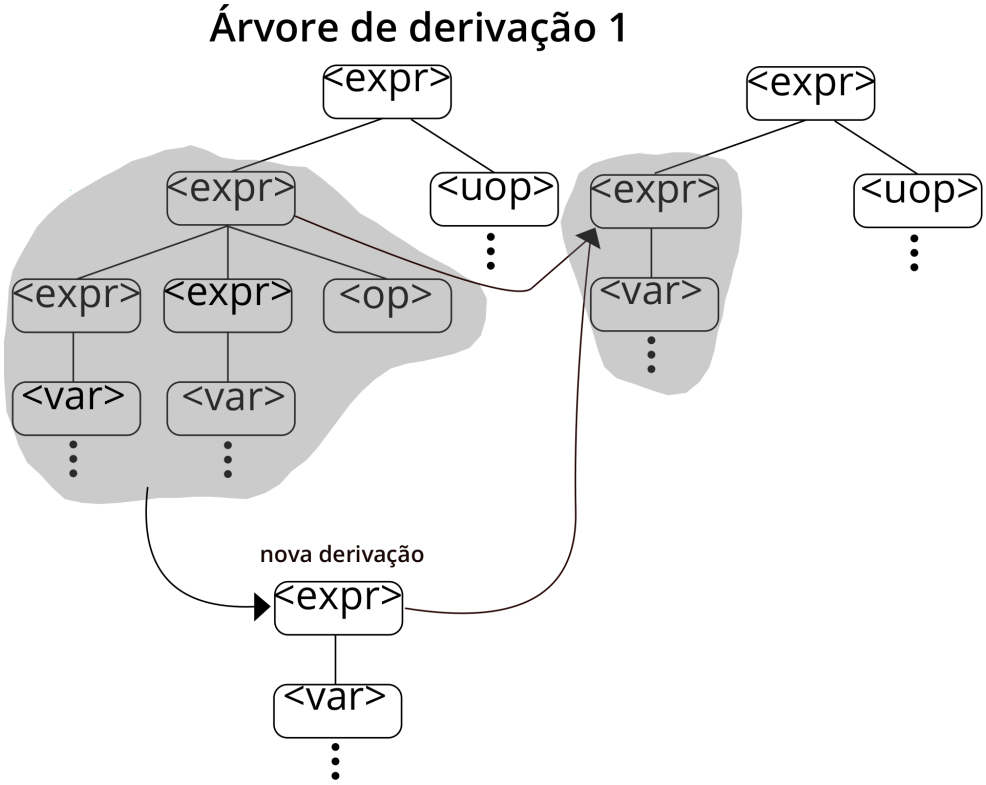


Figura 3.8: Exemplo de operador de mutação de uma PGG

3.3 EVOLUÇÃO DIFERENCIAL

Considerada como uma técnica evolutiva, utiliza um método estocástico de busca para realizar a otimização. São aplicadas mutação e cruzamento para gerar novos indivíduos na população, logo após é feita a seleção e assim uma nova geração é criada.

(CHENG; HWANG, 2001) relatam características pertinentes ao algoritmo, descrevendo-as:

- É um algoritmo de busca estocástica, originalmente motivado pelos mecanismos da seleção natural
- Não é propício a estagnar em ótimos locais, pois como opera em uma população de soluções, realiza a busca em diferentes regiões do espaço de busca
- Recomendado para funções descontínuas, pois não utiliza suas derivadas

- Geralmente não há a necessidade de utilizar uma grande população (acima de 10 indivíduos)
- Funciona também com otimização local, pois os diferenciais gerados eventualmente tornam-se infinitesimais

3.3.1 OPERADORES

Os operadores utilizados na Evolução Diferencial são baseados no princípio da evolução natural, dessa forma são utilizados para manter a diversidade da população, evitar convergências prematuras e obter a melhor solução possível.

Na variante mais básica, três indivíduos são selecionados aleatoriamente para que ocorra uma mutação. O algoritmo realiza uma diferença vetorial ponderada de dois desses indivíduos. Depois é realizada a soma ao outro vetor escolhido, gerando assim um novo indivíduo. Esse processo de mutação é o mais comumente utilizado, chamado de "DE/rand/1" e é calculado como:

$$V_{i,G+1} = X_{\alpha,G} + F * (X_{\beta,G} - X_{\gamma,G}) \tag{3.1}$$

onde $\alpha, \beta, \gamma \in 1, 2, ..., TP \mid \alpha \neq \beta \neq \gamma \neq i$ representam índices da população, sendo TP o tamanho da população. $V_{i,G+1}$ é o indivíduo a ser gerado, $X_{\alpha,G}$ é o vetor base, F > 0 é uma constante real que representa a ponderação da diferença entre os vetores $X_{\beta,G}X_{\gamma,G}$.

Existem outras variantes para mutação em uma ED (DAS; SUGANTHAN, 2011), são algumas delas:

- DE/best/1: $V_{i,G+1} = X(best) + F * (X_{\alpha,G} - X_{\beta,G})$
- DE/best/2: $V_{i,G+1} = X(best) + F * (X_{\alpha,G} - (X_{\beta,G}) + F * (X_{\gamma,G} - X_{\delta,G})$
- DE/rand/2: $V_{i,G+1} = X_{\alpha,G} + F * (X_{\beta,G} - X_{\gamma,G}) + F * (X_{\delta,G} - X_{\eta,G})$

Assim como na Equação 3.1, $\alpha, \beta, \gamma, \delta, \eta \in 1, 2, ..., TP \mid \alpha \neq \beta \neq \gamma \neq \delta \neq \eta$ representam índices da população. Um exemplo bidimensional de mutação na ED pode ser visto na Figura 3.9.

Para aprimorar a diversidade da população, o cruzamento ocorre após gerar o vetor base através da mutação.

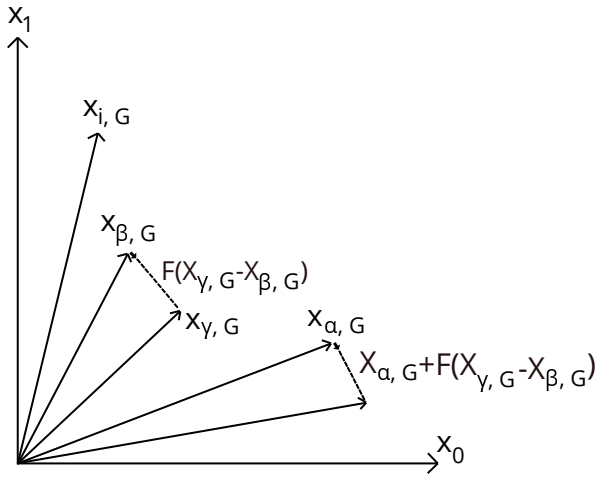


Figura 3.9: Exemplo bidimensional da mutação em uma ED. Adaptado de (BOCCATO *et al.*, 2009)

Para o cruzamento binomial, inicialmente é escolhido um número aleatório, sendo este menor que o número de variáveis D (dimensão do vetor), o número escolhido funciona como ponto inicial do vetor base (x). Também é determinado um número L, que é utilizado para determinar o número de componentes que o vetor doador (v) contribuirá para o vetor base. Com esses valores determinados, é possível obter o vetor resultante (u) através de:

$$u_{i,j} = \begin{cases} v_{i,j} \text{ se } (rand[0,1] \leq TC \text{ ou } j = j_{rand}) \\ x_{i,j} \text{ senão} \end{cases}$$

3.3.2 SELEÇÃO

A seleção é realizada para manter constante o tamanho da população, para isso o algoritmo determina quem deve sobreviver para a próxima geração. O operador de seleção pode ser descrito como na Equação:

$$x_{i,G+1} = \begin{cases} u_{i,G} \text{ se } (f(u_{i,G}) \leq f(x_{i,G}) \\ x_{i,G} \text{ se } (f(u_{i,G}) > f(x_{i,G})) \end{cases} \tag{3.2}$$

Um pseudo-código para a ED pode ser visto no Algoritmo 5.

Algoritmo 5: Pseudo-código da Evolução Diferencial (BROWNLEE, 2012)

```
Evolução Diferencial (TP, MUT, CR, GEN)
    Entradas: Tamanho da população TP, escala de mutação MUT, taxa de
              cruzamento TC, numero de gerações GEN
    início
        G ← 0;
        CriarPopulaçãoInicialAleatória(TP);
        for i = 0 até TP do
            Avaliar f(x⃗_{i,G}) ;            // x⃗_{i,G} é um indivíduo na população
        end for
        for G ← 1 até GEN do
            for i ← 1 até NP do
                SelecionarAleatoriamente(r1, r2, r3) ;        // r1 ≠ r2 ≠ r3 ≠ i
                jRand ← RandInt(1, N) ;        // N é o número de variáveis
                for j ← 1 até N do
                    if Rand(0, 1) < CR or j = jRand then
                        u_{i,j,G+1} = x_{r3,j,G} + F.(x_{r1,j,G} − x_{r2,j,G});
                    end if
                    else
                        u_{i,j,G+1} = x_{i,j,G};
                    end if
                end for
                if f(u⃗_{i,G+1}) ≤ f(x⃗_{i,G}) then
                    x⃗_{i,G+1} = u⃗_{i,G+1};
                end if
                else
                    x⃗_{i,G+1} = x⃗_{i,G};
                end if
            end for
        end for
    fim
```

3.4 ESTRATÉGIA EVOLUTIVA

As Estratégias Evolutivas constituem uma classe de algoritmos evolutivos utilizados como técnica de busca para problemas de otimização de funções reais (RAGGETT, 1982; BäCK *et al.*, 1991). Ela utiliza repetidas variações estocásticas (mutação) seguida por seleção, da qual uma nova prole é gerada, avaliada e os melhores serão os pais da próxima geração.

Uma ideia importante introduzida na EE é a adaptação dos parâmetros de forma *online*. Ou seja, enquanto ocorre o processo evolutivo, os parâmetros da estratégia são adaptados através da introdução dos mesmos na representação genética dos indivíduos. Esses parâmetros são, taxa de mutação, desvio padrão da mutação, e probabilidade de recombinação.

A EE é considerada uma técnica com bons resultados para otimização contínua (HANSEN *et al.*, 2015). Seus indivíduos normalmente são compostos por vetor de parâmetros reais de dimensão n.

A técnica e seus parâmetros são normalmente definidos pelo número de filhos (λ), número de pais (μ) e a estratégia de substituição.

3.4.1 ESTRATÉGIAS DE SUBSTITUIÇÃO

Existem alguns tipos de estratégias de substituição em uma EE (HANSEN *et al.*, 2015), são elas: $(1+1)$-EE, $(\mu+\lambda)$-EE, $(\mu+1)$-EE, (μ,λ)-EE

$(1+1)$-EE é composta por apenas um indivíduo, nesse tipo de abordagem apenas o operador genético de mutação é utilizado. Uma representação interessante para essa estratégia é a do indivíduo ser um par de vetor real do tipo (x,σ). Onde x é um ponto no espaço de busca, e σ representa o vetor de desvios padrões utilizado para atualizar x. Para otimizar x, a mutação segue da seguinte forma:

$$x^{t+1} = x^t + G(0,1) \tag{3.3}$$

Onde $G(0,\sigma)$ é um vetor de números gerados por uma função gaussiana de média 0 e desvio padrão σ que deve ser atualizado de forma *online*.

A mutação gaussiana respeita a teoria biológica de que filhos herdam características dos pais, assim como pequenas variações são mais frequentes do que grandes variações (BäCK *et al.*, 1991).

O indivíduo que sofreu mutação é inserido na nova geração apenas se ele possuir uma aptidão melhor que a do pai. Apesar de existir apenas um indivíduo, essa estratégia também é conhecida como de dois membros pois o filho compete com o pai (MICHALEWICZ, 1996).

Para utilizar o conceito de população nas EE, Eigen (1973) introduziu o aumento do número de pais ($\mu > 1$) mas mantendo um único filho. É uma abordagem que se assemelha ao método de gradiente probabilístico, do qual tem perturbação Gaussiana e adaptativa.

Com a possibilidade de mais de um pai, o cruzamento passa a ser possível via um operador de recombinação denominado *operador de recombinação discreto* e é equivalente ao *crossover uniforme* em algoritmos genéticos (BäCK *et al.*, 2000). O operador de seleção então remove o indivíduo menos apto dentre os pais e filhos.

As EE passaram a trabalhar com populações de indivíduos (BORN, 1980; RAGGETT, 1982). Na $(\mu + \lambda)$-EE μ pais produzem λ filhos e a população gerada pelos dois é reduzida para μ indivíduos. Dessa forma, pais que possuem aptidão melhores que as dos filhos são mantidos até que um filho superior a eles seja produzido.

Schwefel (1993) sugeriu restringir o período de vida de cada indivíduo para uma geração apenas. Então, na (μ, λ)-EE os μ melhores filhos são escolhidos para sobreviver até a geração seguinte como representado no Algoritmo 6.

3.5 REGRESSÃO SIMBÓLICA

Regressão simbólica é um tipo de análise regressiva que busca no espaço das expressões matemáticas encontrar um modelo que melhor se ajusta a um dado conjunto de dados (KEIJZER, 2008). Diversos problemas podem ser modelados matematicamente, desde uma análise de custo de um produto até problemas mais complexos como testar a eficácia de um tratamento médico. Esse tipo de problema é comumente utilizado para testes de desempenho em Programação Genética.

3.5.1 AVALIAÇÃO DE APTIDÃO

Durante o processo de evolução é necessário realizar uma avaliação da expressão, para assim saber o quão adequada essa expressão é em relação ao conjunto de dados.

Dois fatores podem ser levados em conta para avaliar a aptidão de uma expressão: sim-

Algoritmo 6: Pseudo-código da $(\mu, \lambda) - EE$ (BROWNLEE, 2012)

```
(μ, λ) − EE (μ, λ, TP)
    Entradas: Quantidade de pais μ; Quantidade de filhos λ; Tamanho da
              população TP
    Saída   : Melhor indivíduo gerado denotado como S_melhor
    início
        Pop ← InicializarPopulacao(μ,TP);
        AvaliarPopulacao(Pop);
        S_melhor ← ObterMelhor(Pop,1);
        while ¬ CriterioDeParada() do
            Filhos ← ∅;
            for i ← 0 to λ do
                Pai_i ← ObterPai(Pop,i);
                S_i ← ∅;
                Si_problema ← Mutar(Pi_problema,Pi_estrategia);
                Si_estrategia ← Mutar(Pi_estrategia);
                Filho ← S_i;
            end for
            AvaliarPopulacao(Filhos);
            S_melhor ← ObterMelhor(Filhos +S_melhor,1);
            Pop ← SelecionarMelhor(Pop,Filhos,μ);
        end while
        return S_melhor;
    fim
```

plicidade e desempenho. Para simplicidade, diferentes indivíduos podem gerar a mesma expressão, apenas simplificando sua interpretação. A equação

$$f(x) = \frac{x * x + 2}{x^2 + 3.5 * 0.3 * 2 - 0.1}$$

também pode ser escrita em uma forma mais simplificada como

$$f(x) = \frac{x^2 + 2}{x^2 + 2}$$

ou

$$f(x) = 1$$

Na regressão simbólica no entanto o que é mais comumente levado em conta para avaliar uma boa resposta para o problema é o erro dos valores indicados pelo modelo candidato quando comparado com os valores esperados. Ou seja, o quão boa essa expressão é em relação aos dados. Para avaliar uma expressão, pode-se verificar o quão próxima é a curva da função avaliada aos pontos.

A Tabela 3.1 é um exemplo de conjunto de dados para uma regressão a qual o y_i corresponde a saída de cada x_i. Com isso é possível calcular sua aptidão usando a Equação 3.4.

Tabela 3.1: Exemplo de dados de um problema de regressão

x_i	y_i
-1.0	-1.0
-0.5	0.0
0.0	1.0
0.5	2.0
1.0	3.0

$$aptidão = \sum_{i=1}^{m} |y_i - f(x_i)| \tag{3.4}$$

Dessa forma para calcular a aptidão de um indivíduo que represente a função $f(x) = 2x - 1$ tem-se

$$aptidão = |-1-(-3)| + |0-(-2)| + |1-(-1)| + |2-0| + |3-1| = 10$$

Para um problema artificial e sem ruído, a aptidão de um indivíduo em uma regressão simbólica é zero quando ele alcança a equação que gerou os dados. Para a Tabela 3.1 o indivíduo $f(x) = 2x + 1$ é a função que gerou tais dados, logo sua aptidão será como

$$aptidão = |-1-(-1)| + |0-0| + |1-1| + |2-2| + |3-3| = 0$$

A aptidão é comumente utilizada como forma de representar a qualidade da solução candidata, neste trabalho o problema a ser otimizado é de minimização, ou seja, quanto mais próximo de 0 melhor é a solução candidata.

Na regressão simbólica, um problema a ser tratado é o tamanho do espaço de busca das expressões matemáticas, dada a infinidade de possibilidades de expressões.

Ótimos locais são outro problema constantemente abordado em regressão simbólica. Um algoritmo pode ficar preso em uma resposta por não haver melhores opções em seu entorno.

3.5.2 VANTAGENS DE UTILIZAR PROGRAMAÇÃO GENÉTICA PARA REGRESSÃO SIMBÓLICA

Diversos métodos podem realizar regressão simbólica (DEKLEL *et al.*, 2017; WATANA-CHATURAPORN, 2016; DIVEEV *et al.*, 2017; MELO *et al.*, 2015). As redes neuronais, por exemplo conseguem mapear dados de entrada em dados de saída. A vantagem do uso da Programação Genética é a possibilidade de extrair conhecimento através da função modelada (PAPPA *et al.*, 2008; DEVANEY; HAGEDORN, 2002) e redes neuronais, por sua vez, geram funções as quais a interpretação é uma tarefa muito difícil.

Apesar da PG produzir modelos interpretáveis (MEIER *et al.*, 2013; KOZA, 1999), ela demanda alto custo computacional, dado o número de avaliações de função objetivo necessárias para alcançar uma boa solução. Para modelar fenômenos dinâmicos, por exemplo, pode requerer a solução numérica de equações diferenciais ordinárias durante a

fase de avaliação.

3.6 CLASSIFICAÇÃO

Classificação é o processo que utiliza dados já classificados para gerar um modelo classificador (AMARAL, 2016). Uma vez definido esse modelo os dados não são mais necessários. A cada nova instância com novos dados, estes são aplicados ao modelo que vai prever, com uma margem de erro, a qual classe o dado pertence.

Um exemplo simples é identificar se uma determinada compra realizada por cartão de crédito é do perfil do usuário ou não. Assim, a operadora pode realizar uma verificação se a compra realmente foi feita pelo seu cliente. Para isso, a operadora utiliza as características de compras anteriores realizadas por seu cliente, como: local, tipo de produto, valor do produto, entre outros. Dessa forma é possível determinar qual o perfil do usuário. Possibilitando que qualquer compra realizada possa ser verificada como condizente ou não a esse perfil.

Quanto maior a base de dados para classificação, maior a quantidade de elementos para treinar um classificador, dessa forma o modelo criado pode apresentar uma menor margem de erro (SUG, 2010).

As formas de classificação podem ser vistas na Tabela 3.2 onde VP ocorre quando instancias positivas são classificadas corretamente, FN quando instâncias verdadeiras são classificadas de forma errada, FP ocorre quando negativos são classificados como positivos e VN quando valores negativos são classificados corretamente.

Tabela 3.2: Matriz de confusão

	Objeto previsto	Objeto não previsto
Objeto pertencente	Verdadeiro positivo (VP)	Falso negativo (FN)
Objeto não pertencente	Falso positivo (FP)	Verdadeiro negativo (VN)

Na Tabela 3.3 tem-se representada uma base de dados com conjuntos de dados balanceados, onde a coluna y_i indica a classe correspondente aos dados x_1 e x_2.

Sendo um problema de maximização (utilizando calculo da porcentagem de acerto) é possível então realizar o cálculo de aptidão com a seguinte fórmula:

$$aptidão = \frac{VP + VN}{VP + VN + FP + FN} \tag{3.5}$$

Tabela 3.3: Exemplo de dados balanceados de classificação

x_1	x_2	y_i
0	0	1
0.2	-0.12	1
1	0	0
0.4	-3	1
1.7	1.1	0
2.1	0.8	0

Tendo como objetivo a maximização da aptidão do indivíduo representado no Algoritmo 7, com base na Tabela 3.3 têm-se:

$$aptidão = \frac{2+3}{2+3+0+1} = 0.8333... \tag{3.6}$$

Algoritmo 7: Exemplo de indivíduo aleatório de classificação

```
if x_1 < 0 then
    return 1;
end if
else
    if x_2 ≥ 0 then
        return 0;
    end if
    else
        return 1;
    end if
end if
```

Dessa forma este indivíduo tem uma acurácia de 83.33%.

4 MÉTODO PROPOSTO

No método proposto neste trabalho, é sugerido a execução da ED/EE para realizar otimizações nas constantes geradas pela PGG tradicional. Existem duas variantes no fluxo de execução do método proposto, são elas: Execução da ED/EE durante a evolução da PGG e execução da ED/EE após a evolução da PGG.

Para a execução do otimizador durante a evolução, a PGG realiza seu fluxo de execução normalmente, porém logo após a execução da mutação é realizada uma verificação, caso o ED/EE não deva ser executado nessa geração o algoritmo segue seu fluxo padrão. Caso o otimizador deva ser executado, são executadas Θ instâncias do otimizador, onde Θ representa os melhores indivíduos da PGG, no método proposto definimos $\Theta = 1$ (apenas no melhor indivíduo) e $\Theta = elite$. Para cada instância a ser executada, um indivíduo é passado para a ED/EE como um indivíduo da população inicial do método, os demais indivíduos da ED/EE são gerados aleatoriamente. Após isso a ED/EE realiza a otimização das constantes desses indivíduos e retorna os indivíduos otimizados para a PGG como pode ser visto na Figura 4.1 e 4.2, nesse passo a gramática da PGG é atualizada com as novas constantes geradas dentro da ED/EE. A partir desse passo a PGG segue para a próxima geração. A Figura 4.3 representa o fluxograma desse processo. O Algoritmo 8 descreve um pseudo-código do método criado.

Para o caso de execução do otimizador após a evolução, a PGG realiza toda sua evolução normalmente e apenas após o término da execução da PGG a ED/EE é executada. Uma instância da ED/EE para cada um dos Θ indivíduos é criada, e da mesma forma a ED/EE executa sua evolução.

4.1 ED/EE DURANTE A EVOLUÇÃO DA PGG

Nessa abordagem, a ED/EE é executada 10 vezes em intervalos intercalados. À medida que a ED/EE encontra novas constantes elas são inseridas na gramática, para que assim a PGG possa utilizá-las durante sua evolução padrão.

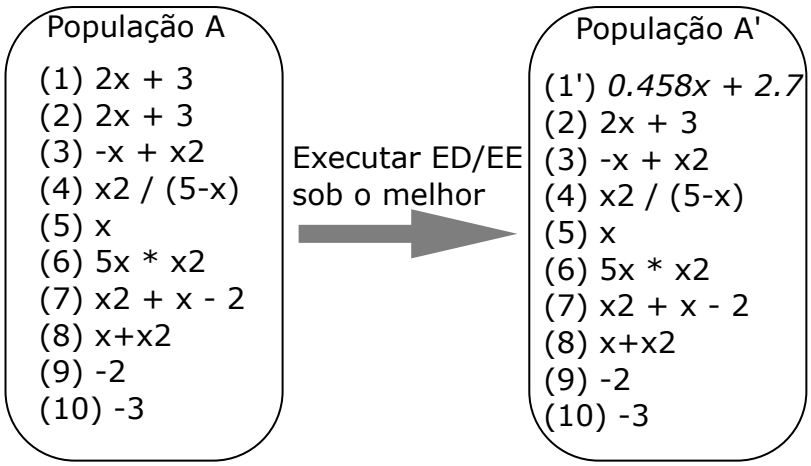


Figura 4.1: Exemplo de uma execução da ED/EE apenas no melhor indivíduo da população. Caso o indivíduo não apresente melhora no cálculo de aptidão então sua versão original é mantida.

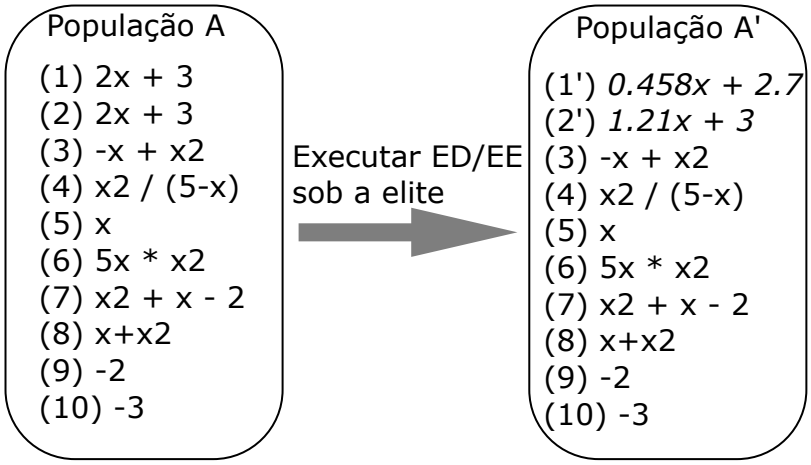


Figura 4.2: Exemplo de uma execução da ED/EE na elite da população, com uma elite de 20%.

Algoritmo 8: Pseudo-código para PGGED/PGGEE

PGGED/PGGEE (P, N, FC, FM, Θ)
Entradas: Número limite de avaliações de função objetivo P, Tamanho da população TP, Fator de cruzamento CR, Fator de mutação MUT, Número de indivíduos a ser otimizado Θ
início
Criar população inicial Pop de tamanho TP;
Inicializar Pop_{tmp} vazio;
Avaliar Pop com otimizador ; `// Otimizador pode ser tanto ED como EE`
while *P não é atingido* **do**
while $|Pop_{tmp}| < TP$ **do**
Selecionar p_1 e p_2 por torneio;
Copiar p_1 para p'_1 e p_2 para p'_2;
if $aleatorio < CR$ **then**
Aplicar cruzamento em p'_1 e p'_2;
end if
if $aleatorio < MUT$ **then**
Aplicar mutação em p'_1 e p'_2;
end if
Avaliar p'_1 e p'_2;
Inserir p'_1 e p'_2 em Pop_{tmp};
end while
if *Executar otimizador nessa geração* **then**
for *cada indivduo em ObterMelhoresIndividuos(*Θ*)* **do**
Otimizador(*individuo*) ; `// Algoritmo 5 e Algoritmo 6`
end for
end if
$pop \leftarrow$ elite de $Pop \cup$ elite de Pop_{tmp};
Esvaziar Pop_{tmp};
end while
fim

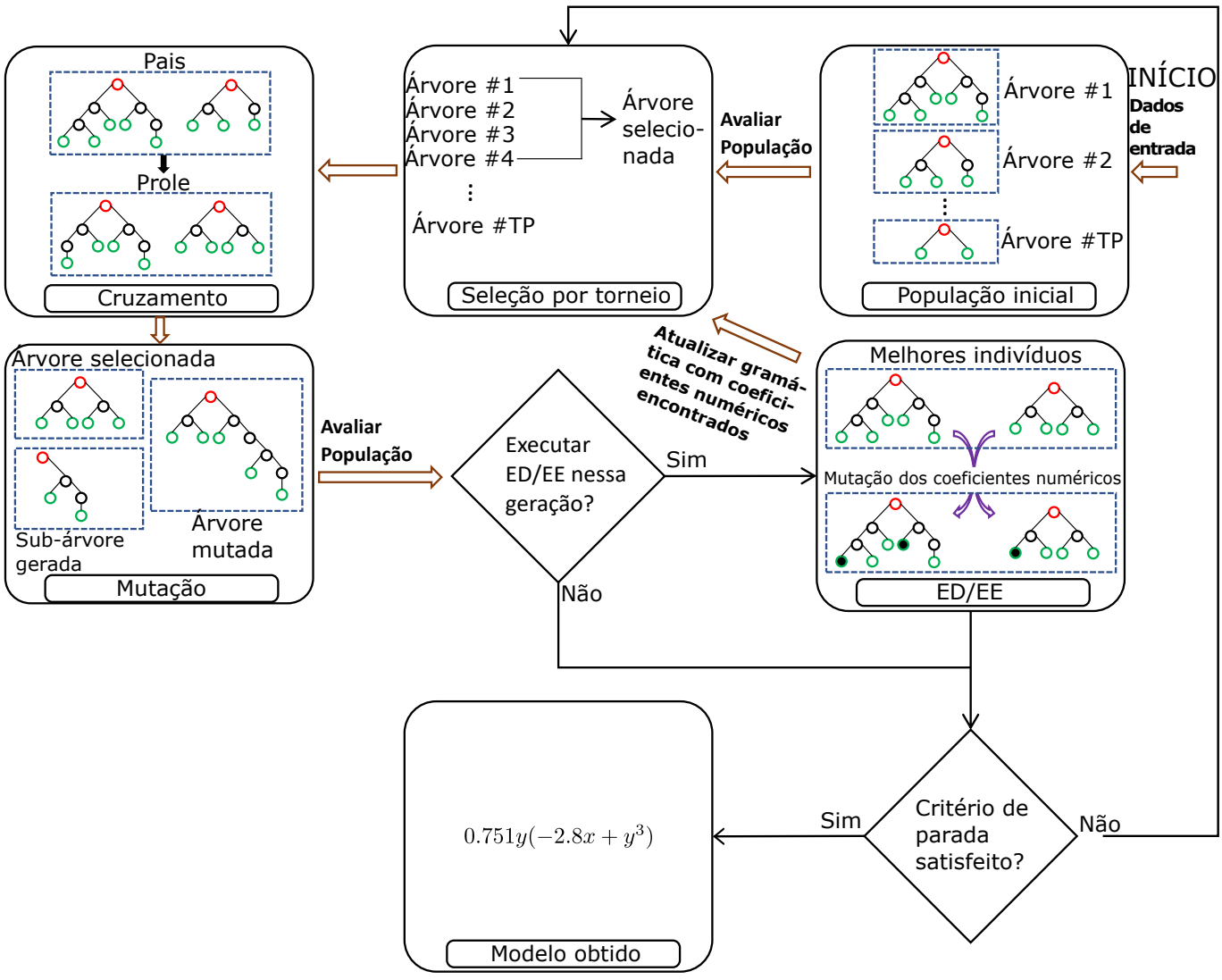


Figura 4.3: Fluxograma do método proposto

4.2 APÓS O TÉRMINO DA EVOLUÇÃO DA PGG

Um orçamento Ω é definido pelo usuário, e esse valor representa a porcentagem do custo computacional inicial que será utilizado para evolução da ED/EE. Por exemplo, se definido $\Omega = 10$, então a PGG evolui utilizando 90% do custo computacional e a ED/EE utiliza os 10% restantes para o ajuste dos coeficientes.

5 RESULTADOS

Foram escolhidas 7 bases de classificação para realizar os experimentos do método proposto. Para regressão simbólica, 23 funções foram escolhidas para os testes. Foram então utilizados os Perfis de Desempenho (PD) para comparar as técnicas propostas. Foram realizadas 30 execuções independentes para cada variante, os resultados obtidos são apresentados através de tabelas contendo valores mínimos encontrados, mediana, média, desvio padrão e máximo das execuções. Para uma comparação estatística foi utilizado o teste não paramétrico de Wilcoxon, sendo assim foi possível comparar as técnicas em cada um dos problemas testados.

A técnica que utiliza a ED para otimização das constantes foi denominada como Programação Genética Gramatical com Evolução Diferencial (PGGED). Analogamente, a utilização da EE como otimizador foi denominada Programação Genética Gramatical com Estratégia Evolutiva (PGGEE).

5.1 IMPLEMENTAÇÃO

O algoritmo foi implementado utilizando a linguagem C++. A linguagem foi escolhida por ser muito difundida entre aplicações científicas. Também com a possibilidade de programação orientada a objetos, fazendo dessa linguagem a de escolha para implementação. O código para a técnica proposta pode ser acessado em: https://github.com/Gloomtail/PGG.git

5.1.1 DEFINIÇÃO DOS PARÂMETROS

Para realizar os testes, os dados foram divididos em três conjuntos: conjunto de treinamento (usado para evoluir a população), conjunto de validação (usado para escolher qual modelo deve ter menor erro esperado de generalização), e conjunto de teste, que é utilizado para avaliar o modelo criado. A divisão realizada foi feita de forma que o treinamento conteve 60% dos dados, a validação utilizou 20% e o teste os outros 20%.

Experimentos preliminares foram realizados para determinar os parâmetros μ e λ da PGGEE. O parâmetro λ da PGGEE foi escolhido do conjunto $\{1, 2, 4\}$, o valor de μ foi fixado em 1. Também foram feitos experimentos para determinar o número de chamadas

(Ω) do otimizador. Foi escolhido $\Omega = 10$; ou seja, o algoritmo executará a ED/EE 10 vezes em intervalos igualmente intercalados durante todo o processo.

O indivíduo que é transferido da população da PGG é utilizado como um indivíduo da população inicial na ED. Os outros 9 que compõe a população inicial são gerados aleatoriamente dentro de um determinado intervalo. Esse intervalo é determinado com base nos problemas a serem resolvidos, o que foi feito, é a utilização dos valores máximo e mínimo que as variáveis assumem para gerar os dados.

Para inicialização da EE o indivíduo a ser otimizado é a população inicial, e assim é feita a evolução padrão da EE.

Por existir diversas variantes, foi criada uma nomenclatura para as técnicas testadas. Para todos os casos em que a EE é utilizada, o nome inicia-se com PGGEE, quando a ED é utilizada, o nome então inicia-se com PGGED. Para os casos com EE a próxima parte refere-se ao valor λ utilizado.

Para todos os casos em que a ED/EE ocorre após a execução da PGG, o valor de Ω é adicionado ao nome. A próxima parte dos nomes refere-se otimizador ser aplicado apenas ao melhor individuo da PGG (sem símbolo adicionado ao nome) ou aplicado a elite da população, denominado como "E".

Portanto tem-se:

1. PGG (PGG na sua forma padrão)
2. PGGED (Executa-se a ED no melhor indivíduo durante a evolução)
3. PGGEDE (Executa-se a ED na elite durante a evolução)
4. PGGED5 (Executa-se a ED no melhor indivíduo após a evolução com 5% do orçamento total)
5. PGGED5E (Executa-se a ED na elite após a evolução com 5% do orçamento total)
6. PGGED10 (Executa-se a ED no melhor indivíduo após a evolução com 10% do orçamento total)
7. PGGED10E (Executa-se a ED na elite após a evolução com 10% do orçamento total)
8. PGGEE1 (Executa-se a EE(1+1) no melhor indivíduo durante a evolução)

9. PGGEE2 (Executa-se a EE(1+2) no melhor indivíduo durante a evolução)
10. PGGEE4 (Executa-se a EE(1+4) no melhor indivíduo durante a evolução)
11. PGGEE1E (Executa-se a EE(1+1) na elite durante a evolução)
12. PGGEE2E (Executa-se a EE(1+2) na elite durante a evolução)
13. PGGEE4E (Executa-se a EE(1+4) na elite durante a evolução)
14. PGGEE15 (Executa-se a EE(1+1) no melhor indivíduo após a evolução com 5% do orçamento total)
15. PGGEE25 (Executa-se a EE(1+2) no melhor indivíduo após a evolução com 5% do orçamento total)
16. PGGEE45 (Executa-se a EE(1+4) no melhor indivíduo após a evolução com 5% do orçamento total)
17. PGGEE15E (Executa-se a EE(1+1) na elite após a evolução com 5% do orçamento total)
18. PGGEE25E (Executa-se a EE(1+2) na elite após a evolução com 5% do orçamento total)
19. PGGEE45E (Executa-se a EE(1+4) na elite após a evolução com 5% do orçamento total)
20. PGGEE110 (Executa-se a EE(1+1) no melhor indivíduo após a evolução com 10% do orçamento total)
21. PGGEE210 (Executa-se a EE(1+2) no melhor indivíduo após a evolução com 10% do orçamento total)
22. PGGEE410 (Executa-se a EE(1+4) no melhor indivíduo após a evolução com 10% do orçamento total)
23. PGGEE110E (Executa-se a EE(1+1) na elite após a evolução com 10% do orçamento total)

24. PGGEE210E (Executa-se a EE(1+2) na elite após a evolução com 10% do orçamento total)

25. PGGEE410E (Executa-se a EE(1+4) na elite após a evolução com 10% do orçamento total)

Para cada técnica, 30 execuções independentes foram executadas em cada problema solucionado.

Os parâmetros da PGG em todos os casos podem ser vistos na Tabela 5.1.

Tabela 5.1: Parâmetros utilizados na evolução da PGG

Parâmetro	PGG
Avaliações de função objetivo	100000
População	1000
Elite	10%
Altura máxima da árvore	10
Fator de cruzamento	0.9
Fator de mutação	0.9
Seleção por torneio	2 indivíduos aleatórios

O número de avaliações de função objetivo referentes a ED/EE variam de acordo com a abordagem utilizada. Para a execução durante a evolução da PGG, a técnica utiliza 10% do orçamento da PGG, dividido em 10 execuções intercaladas.

A ED foi definida com uma população de tamanho 10, uma escala de mutação e taxa de cruzamento em 0.9. A EE foi definida com um μ igual a 1, o λ varia entre 1,2,4.

5.2 PERFIS DE DESEMPENHO

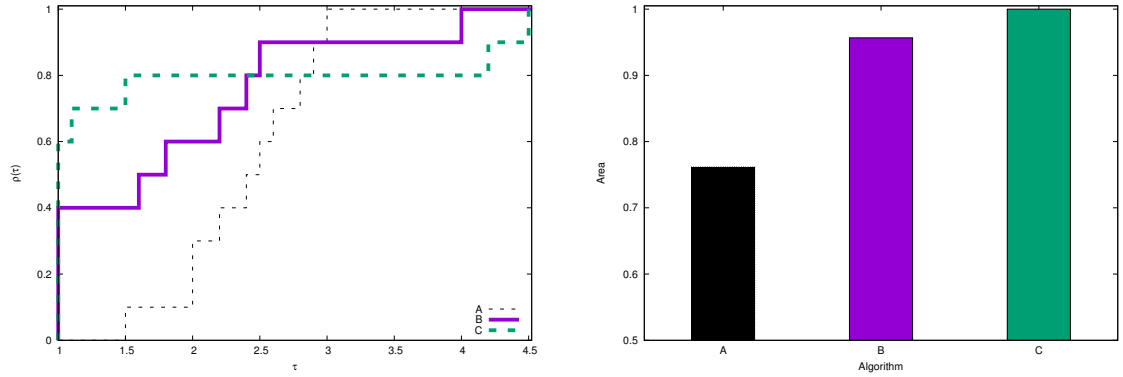
Para facilitar a visualização do desempenho de técnicas de otimização, Dolan e Moré (2002) criaram os perfis de desempenho. Barbosa *et al.* (2010) sugeriram o uso de Perfis de Desempenho (PD) para comparar o desempenho de técnicas de busca estocástica para solucionar problemas de otimização. É uma ferramenta analítica para visualização e interpretação dos resultados obtidos.

Barbosa *et al.* (2010) apresentaram um exemplo para ilustrar o uso dos PD. A Tabela 5.2 mostra o desempenho hipotético para as técnicas A, B, C. A Figura 5.1a corresponde os PD dessas técnicas.

Na Figura 5.1a, pode-se enxergar o ponto ρ_x sendo x a técnica. Quando τ vale um, tem-se a representação de quantas vezes cada técnica foi a melhor no conjunto de problemas

Tabela 5.2: Tabela com exemplo de desempenho das técnicas A, B, C executadas para os problemas P0 a P9

	A	B	C
P0	1.5	1.0	4.5
P1	2.0	1.0	4.2
P2	2.0	1.0	1.5
P3	2.2	1.0	1.1
P4	2.4	1.6	1.0
P5	2.5	1.8	1.0
P6	2.6	2.2	1.0
P7	2.8	2.4	1.0
P8	2.9	2.5	1.0
P9	3.0	4.0	1.0



(a) Exemplo do desempenho gráfico das técnicas, A, B, C.

(b) Gráfico de barra para as AsC normalizadas

Figura 5.1: PD das técnicas A, B e C

analisados. Nesse exemplo tem-se $\rho_C(1) > \rho_B(1) > \rho_A(1)$ portanto a técnica C é a que mais comumente resolve melhor os problemas do conjunto. Pode-se perceber que $\rho_A(1)$ é igual a 0, o que significa que esta técnica nunca apresenta melhor resultado para os problemas propostos. A Figura 5.1b representa o gráfico de barra para as áreas normalizadas sob a curva (AsC). A área composta pelos algoritmos segue a ordem $AsC_C > AsC_B > AsC_A$. A técnica A pode ser considerada a de melhor confiabilidade, pois é a primeira a alcançar o topo do gráfico (com $\tau = 3$).

É utilizado PD nos valores de medianas obtidos para comparar o desempenho das técnicas propostas. Como os PD podem não ser conveniente para analisar simultaneamente um grande número de técnicas, a análise dos métodos será realizada em dois estágios. Primeiro, uma análise de parâmetros da proposta híbrida entre PGG e PGGED (MOTTA *et al.*, 2017) foi realizado. As melhores variantes da PGGEE e PGGED foram então

selecionados de acordo com a AsC, e finalmente comparadas ao método PGG padrão.

5.3 REGRESSÃO SIMBÓLICA

A gramática formal para os problemas de regressão simbólica é composta pelo seguinte conjunto de regras de derivação:

$$
\begin{aligned}
< expr > ::= & < expr >< expr >< op > \mid < var > \mid < expr >< uop > \\
& \mid < const > \\
< op > ::= & + \mid - \mid * \mid / \mid pow \\
< uop > ::= & \, exp \\
< var > ::= & \, x_1 \mid x_2 \\
< const > ::= & -5 \mid -4 \mid -3 \mid -2 \mid -1 \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5
\end{aligned}
$$

Nicolau *et al.* (2015) forneceram diretrizes para definir problemas de *benchmark*, assim como apresentou algumas funções a serem utilizadas para testes. Motta *et al.* (2017) adaptaram diversas funções apresentadas por Nicolau *et al.* para que contenham coeficientes com valores reais.

Funções de ambos os trabalhos foram utilizadas aqui para executar os testes e estão representadas na Tabela 5.3.

Os PD das técnicas que executam a ED em conjunto com a PGG estão representadas na Figura 5.2a. Entre as técnicas comparadas, é possível perceber que utilizar a ED para evoluir apenas o melhor indivíduo da população da PGG durante a execução da mesma apresenta melhores resultados.

Os resultados de todas as técnicas que usam EE durante a evolução da PGG podem ser vistos na Figura 5.3a. De forma geral a PGGEE1E obteve melhores resultados, e a AsC é a maior, como representado na Figura 5.3b.

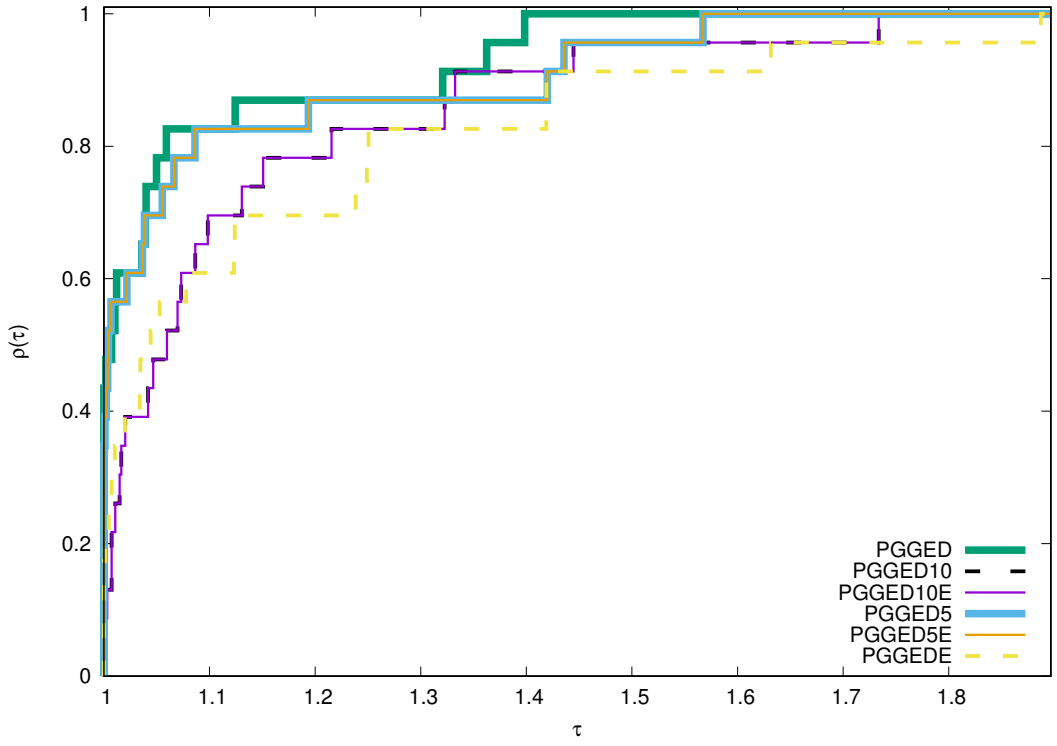
Para as técnicas que utilizam 5% dos cálculos de função objetivo nos Θ indivíduos utilizando a EE após a evolução da PGG, o que apresenta melhor resultado é a estratégia $(1 + 4)$, independente de utilizar a elite ou o melhor indivíduo para a melhoria.

Por fim, serão apresentadas as técnicas que utilizam 10% dos cálculos de função objetivo nos Θ indivíduos utilizando a EE após a evolução da PGG. Pode-se perceber que

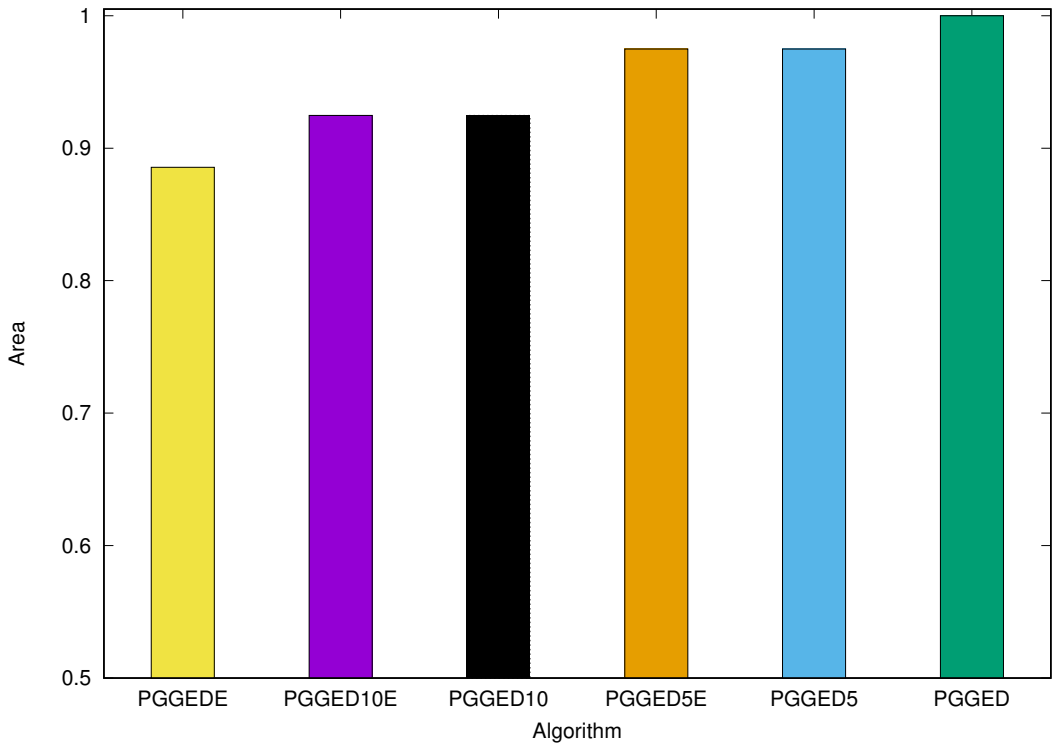
Tabela 5.3: Definição das funções utilizadas nos experimentos computacionais.

Funções de (NICOLAU *et al.*, 2015)	
F1	$f(x_1, x_2) = 6sin(x_1)cos(x_2)$
F2	$f(x_1, x_2) = (x_1 - 3)(x_2 - 3) + 2sin((x_1 - 4)(x_2 - 4))$
F3	$f(x_1, x_2) = \frac{(x_1-3)^4+(x_2-3)^3-(x_2-3)}{(x_2-2)^4+10}$
F4	$f(x_1, x_2) = \frac{1}{1+{x_1}^{-4}} + \frac{1}{1+{x_2}^{-4}}$
F5	$f(x_1, x_2) = {x_1}^4 - {x_1}^3 + {x_2}^2/2 - x_2$
F6	$f(x_1, x_2) = \frac{8}{2+{x_1}^2+{x_2}^2}$
F7	$f(x_1, x_2) = {x_1}^3/5 + {x_2}^3/2 - x_2 - x_1$
F8	$f(x_1, x_2) = \frac{e^{-(x_1-1)^2}}{1.2+(x_2-2.5)^2}$

Funções de (MOTTA *et al.*, 2017)	
F9	$f(x) = x^4 + 2x^3 - 13x^2 - 14x + 24$
F10	$f(x) = 2x^4 - 9x^3 - 4x^2 + 1$
F11	$f(x) = x^4 + 9x^3 - 3x^2 - x + 1$
F12	$f(x) = x^4 - x^2 + 25x + 2$
F13	$f(x) = 0.5x^4 + 1.5x^3 - 13.5x^2 - 13.5x + 24.5$
F14	$f(x) = 2.5x^4 - 9.5x^3 - 3.5x^2 + 1.5$
F15	$f(x) = 1.5x^4 + 9.5x^3 - 2.5x^2 - 0.5x + 1.5$
F16	$f(x) = 1.5x^4 + 0.5x^3 - 0.5x^2 + 25.5x + 1.5$
F17	$f(x_1, x_2) = 6.7sin(x_1)cos(x_2)$
F18	$f(x_1, x_2) = (x_1 - 3.91)(x_2 - 3.33) +$ $2.26sin((x_1 - 4.74)(x_2 - 4.53))$
F19	$f(x_1, x_2) = \frac{(x_1-3.96)^4+(x_2-3.41)^3-(x_2-3.11)}{(x_2-2.94)^4+10.57}$
F20	$f(x_1, x_2) = \frac{1.91}{1.93+{x_1}^{-4}} + \frac{1.5}{1.64+{x_2}^{-4}}$
F21	$f(x_1, x_2) = {x_1}^4 - {x_1}^3 + {x_2}^2/2.71 - x_2$
F22	$f(x_1, x_2) = \frac{8.47}{2.59+{x_1}^2+{x_2}^2}$
F23	$f(x_1, x_2) = {x_1}^3/5.92 + {x_2}^3/2.98 - x_2 - x_1$

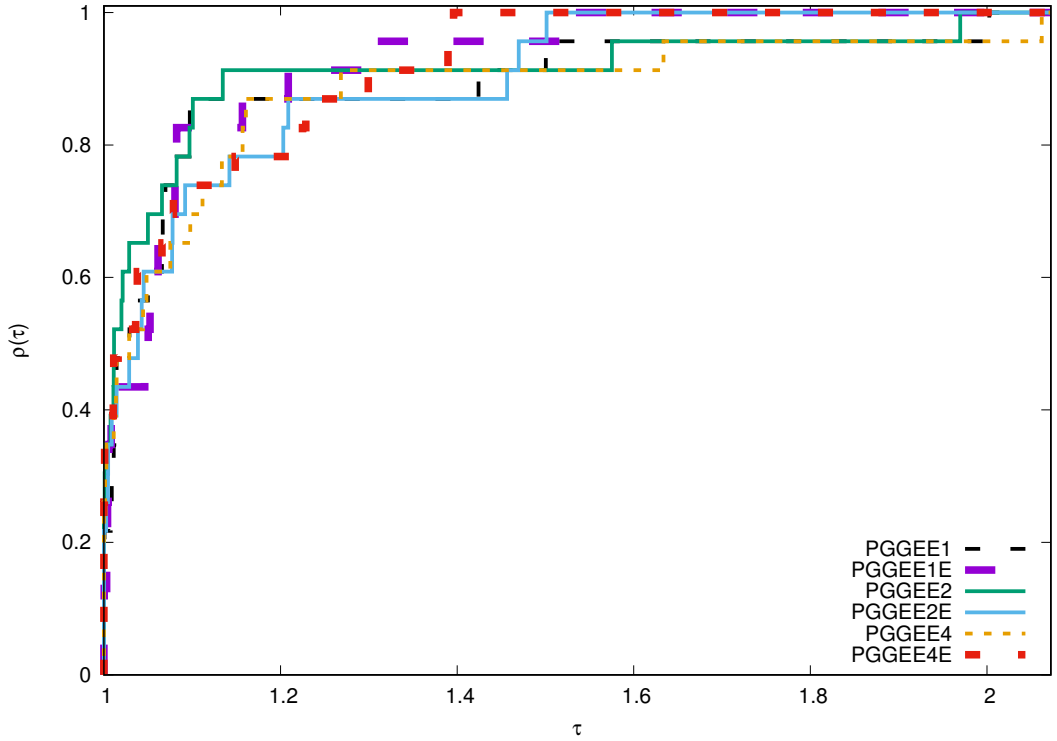


(a) As AsC são, 1.0, 0.925, 0.925, 0.975, 0.975 e 0.886, respectivamente para PGGED, PGGED10, PGGED10E, PGGED5, PGGED5E, PGGEDE.

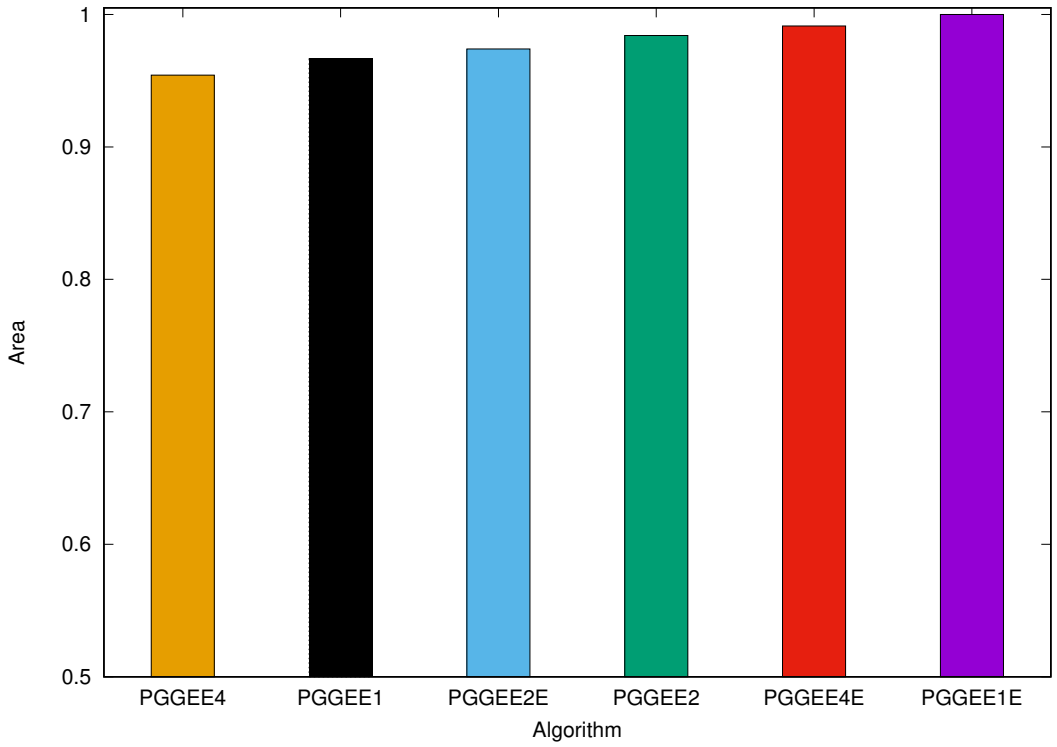


(b) Gráfico de barra para as AsC normalizadas

Figura 5.2: PD das técnicas que utilizam ED para regressão simbólica

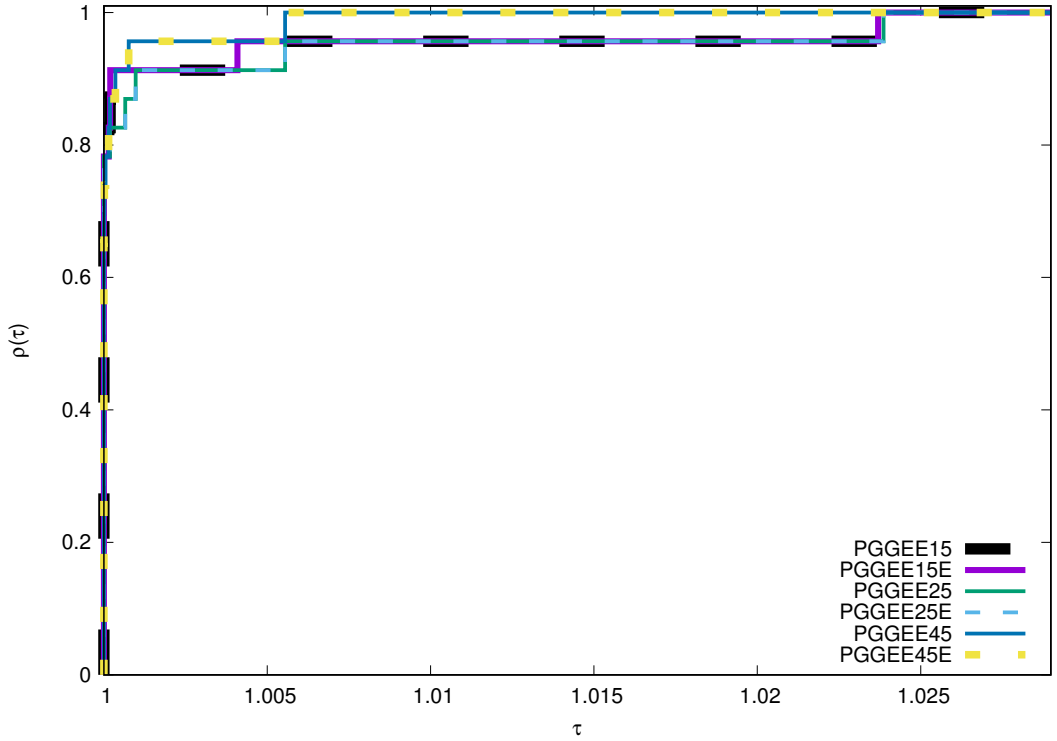


(a) As AsC são, 0.967, 1.0, 0.984, 0.974, 0.954 e 0.991, respectivamente para PGGEE1, PGGEE1E, PGGEE2, PGGEE2E, PGGEE4, PGGEE4E.

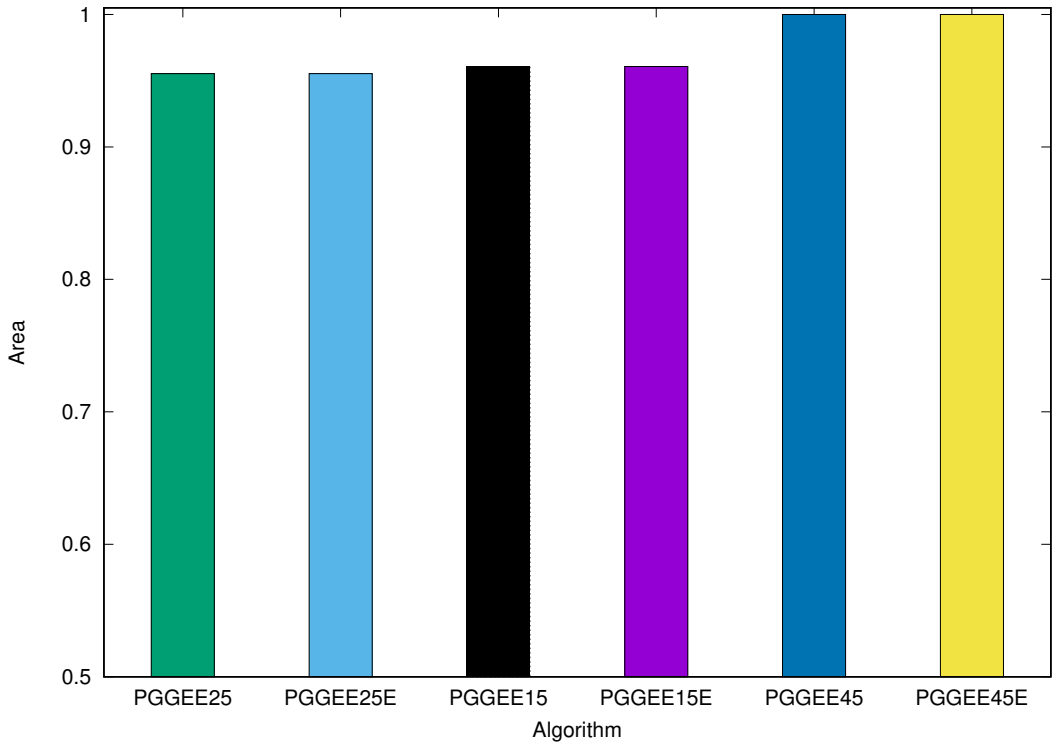


(b) Gráfico de barra para as AsC normalizadas

Figura 5.3: PD das técnicas que utilizam EE durante a evolução da PGG para regressão simbólica



(a) As AsC normalizadas são, 0.961, 0.961, 0.955 e 0.955, 1.0 e 1.0, respectivamente para PGGEE15, PGGEE15E, PGGEE25, PGGEE25E, PGGEE45, PGGEE45E.



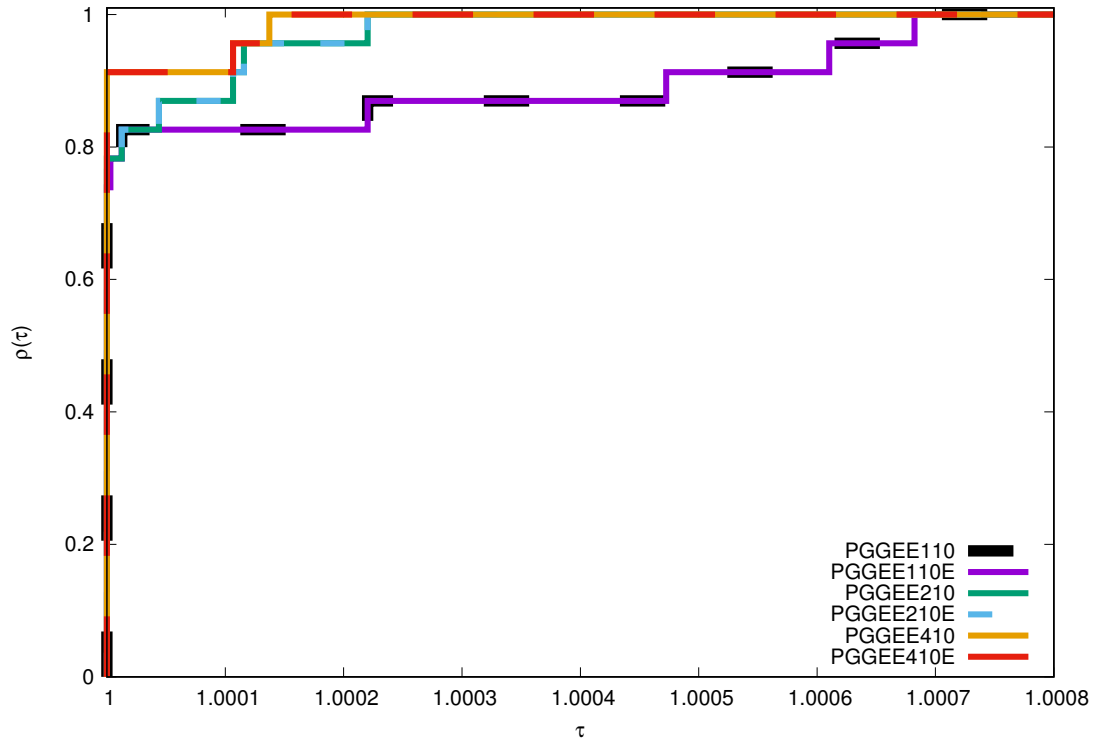
(b) Gráfico de barra para as AsC normalizadas

Figura 5.4: PD das técnicas que utilizam EE após a evolução da PGG com 5% de orçamento para regressão simbólica

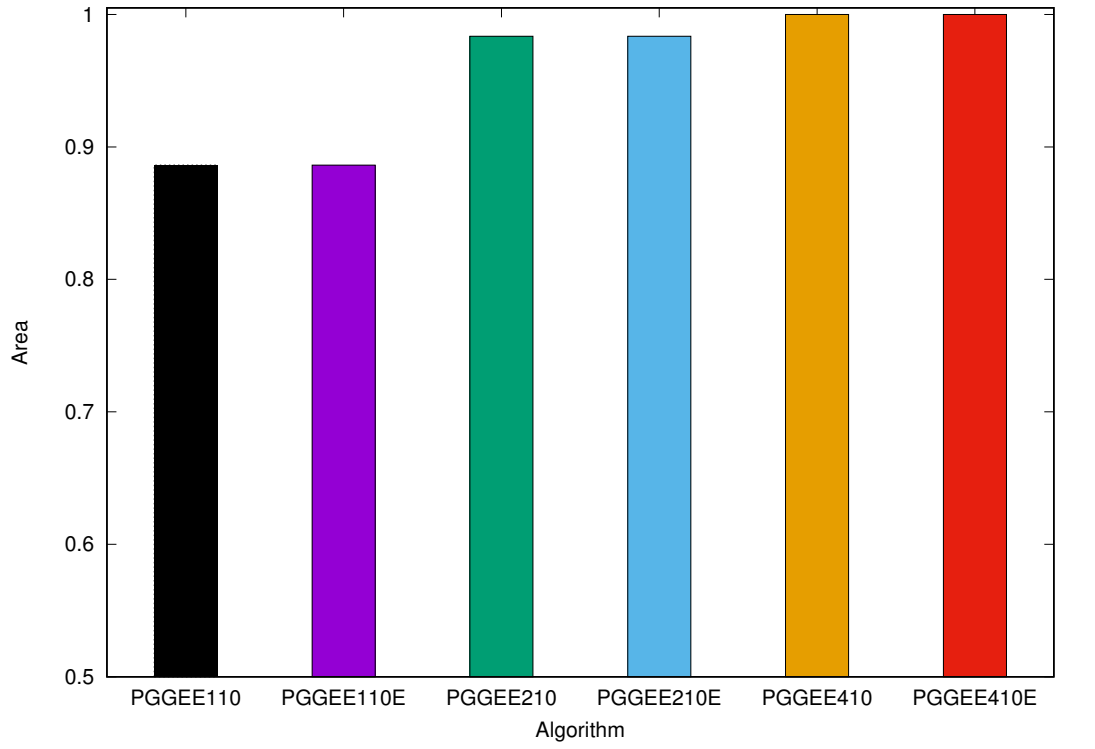
a utilização de elite ou melhor indivíduo não altera o resultado final. A PGGEE410E obteve melhores resultados quando analisado a AsC como representado na Figura 5.5b.

A Figura 5.6a apresenta os PD das quatro melhores técnicas selecionadas anteriormente com a inclusão da PGG padrão. Na Figura 5.6b, pode-se ver o gráfico de barra para as AsC normalizadas.

Quando comparadas as melhores técnicas com a PGG, a PGGED obteve menor AsC e piores resultados do que a PGG padrão. Apesar disso, ela apresenta melhor confiabilidade, devido aos melhores resultados obtidos no pior caso.

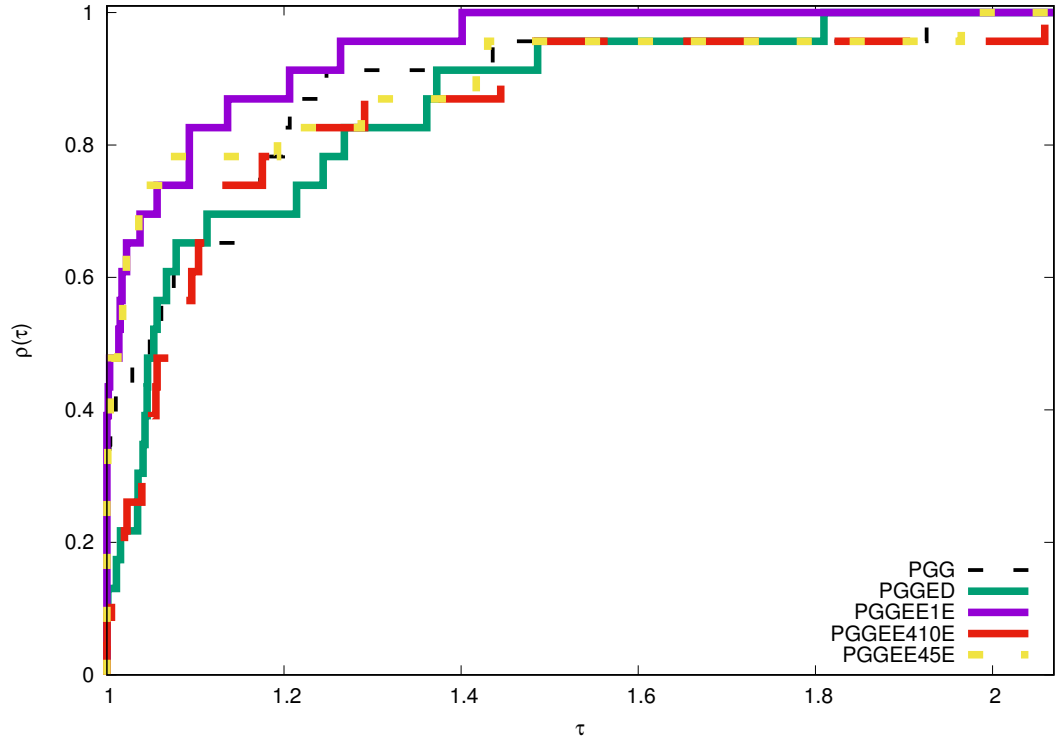


(a) As AsC normalizadas são, 0.886, 0.886, 0.983, 0.983, 1.0 e 1.0, respectivamente para PGGEE110, PGGEE110E, PGGEE210, PGGEE210E, PGGEE410, PGGEE410E.

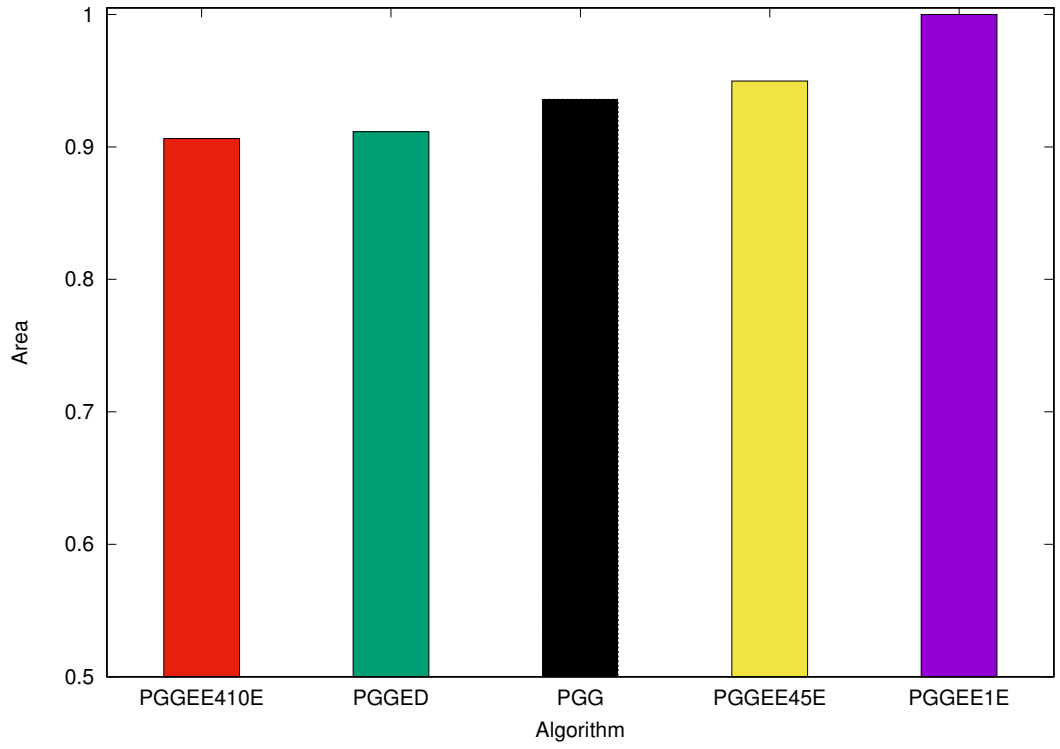


(b) Gráfico de barra para as AsC normalizadas

Figura 5.5: PD das técnicas que utilizam EE após a evolução da PGG com 10% de orçamento para regressão simbólica



(a) As AsC normalizadas são, 0.936, 0.911, 1.0, 0.906 e 0.950, respectivamente para PGG, PGGED, PGGEE1E, PGGEE410E, PGGEE45E,



(b) Gráfico de barra para as AsC normalizadas

Figura 5.6: PD das melhores técnicas selecionadas anteriormente e a PGG padrão para regressão simbólica

As Tabelas 5.4, 5.5 e 5.6 apresentam os seguintes valores para os erros dos modelos: mínimo, mediana, média, desvio padrão (DP) e valor máximo para cada técnica testada em todas as funções utilizadas. Os melhores resultados encontrados estão destacados em negrito.

Para melhor visualização dos resultados obtidos pelas técnicas a Tabela 5.7 representa um somatório do número de vezes que determinada técnica obteve o melhor resultado para cada categoria analisada.

A PGGEE45E obteve a maioria dos melhores resultados (9 dos 23 problemas de regressão considerados aqui, como visto na Tabela 5.7). A PGGEE1E obteve o melhor valor para mediana no pior caso, significando que é mais confiável que os outros. PGGEE1E obteve também a maior AsC normalizada, 9% maior que a técnica com ED, 6% maior que a PGG padrão e 5% maior que a segunda melhor EE.

Considerando o valor máximo (o pior valor encontrado nas 30 execuções independentes), a PGGED superou os outros algoritmos em 8 dos problemas.

Com relação aos melhores resultados encontrados, a PGGEE45E obteve melhor desempenho, tendo obtido o melhor resultado em 9 dos 23 problemas. Tanto a PGG padrão como a PGGED obtiveram pior desempenho nesse quesito, tendo cada uma encontrado o melhor resultado em 5 das funções analisadas.

É importante perceber que analisando o desvio padrão, a PGGED conseguiu alcançar o melhor resultado em 11 (de 23) funções. Isso sugere que a PGGED é mais robusta que as outras técnicas.

Comparando os valores da mediana entre a PGGEE1E com a PGG padrão, a proposta híbrida obteve melhores resultados em 3 das 8 funções de Nicolau *et al.* Observando as funções de Motta *et al.* , foram alcançados os melhores resultados em 10 das 15 funções. No geral a PGGEE1E obteve melhor resultado de mediana em 11 dos 23 problemas. Sendo 7 em 8 em relação as funções de uma variável.

Também é interessante perceber que na média obtida, a PGGEE1E obteve melhores resultados em 9 das 23 funções testadas, sendo a de melhor desempenho entre todas técnicas.

Para análise estatística foi utilizado o teste de Wilcoxon. Este teste elicita quando dois conjuntos podem ser considerados estatisticamente diferentes. Sempre que o teste indica diferença entre os resultados de duas técnicas, a de pior é marcada com um asterisco.

Tabela 5.4: Resultados para as funções em Nicolau *et al.* (2015).

F	Técnica	Min	Mediana	Média	DP	Max
	PGG	8.55986e+00	9.03084e+00	2.93727e+139	1.58177e+140	8.81181e+140
	PGGED	**8.31154e+00**	9.00507e+00	9.79067e+35	5.27244e+36	2.93720e+37
F1	PGGEE1E	8.49919e+00	**8.99557e+00**	**1.27945e+01**	**2.05469e+01**	**1.23435e+02**
	PGGEE45E	8.49894e+00	9.02636e+00	1.22680e+66	6.60652e+66	3.68040e+67
	PGGEE410E	8.49894e+00	9.02952e+00	1.07991e+61	5.81549e+61	3.23973e+62
	PGG	1.91048e+00	5.52579e+01	5.78408e+01	3.17618e+01	1.27485e+02
	PGGED	**1.88257e+00**	5.59574e+01	5.77267e+01	**2.75899e+01**	**1.11872e+02**
F2	PGGEE1E	1.91048e+00	**4.60891e+01**	**5.29214e+01**	3.12387e+01	1.20429e+02
	PGGEE45E	2.78832e+00	5.49738e+01	5.68279e+01	3.31688e+01	1.24353e+02
	PGGEE410E	2.78832e+00	5.49711e+01	5.68489e+01	3.31718e+01	1.24353e+02
	PGG	4.46397e+02	1.25409e+03	inf	nan	inf
	PGGED	7.07300e+02	1.37913e+03	1.44596e+03	4.71864e+02	2.31376e+03
F3	PGGEE1E	**2.96217e+02**	1.27009e+03	1.26373e+03	6.22658e+02	2.48989e+03
	PGGEE45E	5.04233e+02	1.02869e+03	**1.14408e+03**	**4.53971e+02**	**2.21432e+03**
	PGGEE410E	5.04233e+02	**1.00488e+03**	1.15386e+03	4.56161e+02	**2.21432e+03**
	PGG	9.71693e-02	1.76233e-01	1.76717e-01	3.37827e-02	**2.28535e-01**
	PGGED	1.13561e-01	1.80404e-01	1.83878e-01	3.26310e-02	2.47603e-01
F4	PGGEE1E	1.33358e-01	1.87282e-01	1.87338e-01	**3.04711e-02**	2.45933e-01
	PGGEE45E	**9.12514e-02**	**1.71325e-01**	**1.76182e-01**	3.97268e-02	2.53018e-01
	PGGEE410E	**9.12514e-02**	**1.71325e-01**	1.76273e-01	3.96146e-02	2.53023e-01
	PGG	6.99766e+02	**2.12322e+03**	3.53820e+03	**2.58538e+03**	**8.89310e+03**
	PGGED	5.51637e+02	2.89011e+03	4.35083e+03	3.24561e+03	9.59374e+03
F5	PGGEE1E	**2.95440e+02**	2.12972e+03	3.81496e+03	2.82688e+03	8.89908e+03
	PGGEE45E	3.86655e+02	**2.12322e+03**	3.41892e+03	2.68302e+03	9.08872e+03
	PGGEE410E	3.86655e+02	**2.12322e+03**	**3.41812e+03**	2.68301e+03	9.08872e+03
	PGG	2.40122e-01	3.37980e-01	**3.37526e-01**	3.37993e-02	**3.96340e-01**
	PGGED*	2.86783e-01	3.59618e-01	3.60859e-01	**2.61792e-02**	4.22752e-01
F6	PGGEE1E	**1.76623e-01**	3.42697e-01	3.44083e-01	3.87147e-02	4.04482e-01
	PGGEE45E	2.84118e-01	**3.36873e-01**	3.38041e-01	2.71599e-02	**3.96340e-01**
	PGGEE410E	2.83739e-01	3.36935e-01	3.38028e-01	2.71883e-02	**3.96340e-01**
	PGG	4.83534e+01	**9.57940e+01**	9.29019e+01	1.65596e+01	1.23517e+02
	PGGED	4.48136e+01	1.03285e+02	9.80973e+01	**1.54329e+01**	1.18742e+02
F7	PGGEE1E	**3.53868e+01**	9.79214e+01	**9.26309e+01**	2.07353e+01	1.30663e+02
	PGGEE45E	4.51185e+01	9.74953e+01	9.28218e+01	1.71080e+01	**1.08696e+02**
	PGGEE410E	4.51185e+01	9.74957e+01	9.28303e+01	1.71118e+01	**1.08696e+02**
	PGG	4.75414e-03	1.07161e-02	1.04455e-02	**2.81370e-03**	1.55907e-02
	PGGED*	**4.63069e-03**	1.16188e-02	1.18554e-02	3.84851e-03	2.30298e-02
F8	PGGEE1E	5.02520e-03	1.06111e-02	**9.86379e-03**	3.07961e-03	**1.53554e-02**
	PGGEE45E	4.71422e-03	**9.37627e-03**	1.00397e-02	2.96666e-03	1.65019e-02
	PGGEE410E	4.71422e-03	**9.33808e-03**	1.00306e-02	2.96212e-03	1.65019e-02

Tabela 5.5: Resultados para as funções em Motta *et al.* (2017).

F	Técnica	Min	Mediana	Média	DP	Max
	PGG	1.29678e+02	7.98313e+02	7.58367e+02	1.52837e+02	9.67035e+02
	PGGED	5.92367e+02	7.63638e+02	7.86367e+02	**1.17038e+02**	9.99348e+02
F9	PGGEE1E	5.09145e+02	**7.52078e+02**	7.61722e+02	1.47871e+02	1.31488e+03
	PGGEE45E	**1.24546e+02**	7.68676e+02	7.31483e+02	1.43593e+02	9.16186e+02
	PGGEE410E	**1.24546e+02**	7.68676e+02	**7.30306e+02**	1.42249e+02	**8.82025e+02**
	PGG	9.35517e+01	3.73518e+03	4.03321e+03	**2.85533e+03**	1.29154e+04
	PGGED	1.06695e+02	2.45982e+03	3.10349e+03	3.04951e+03	1.43499e+04
F10	PGGEE1E	**1.10938e+01**	**1.93974e+03**	**2.96434e+03**	2.92488e+03	**1.28108e+04**
	PGGEE45E	3.06463e+01	3.81089e+03	3.78743e+03	3.06909e+03	1.40452e+04
	PGGEE410E	3.07473e+01	3.81089e+03	3.78743e+03	3.06909e+03	1.40452e+04
	PGG	1.08140e+02	4.11884e+03	6.09788e+03	4.74892e+03	1.94962e+04
	PGGED	8.15904e+02	3.53245e+03	**4.70760e+03**	**3.65790e+03**	**1.90697e+04**
F11	PGGEE1E	9.99613e+02	**3.41395e+03**	5.13493e+03	5.47124e+03	2.72265e+04
	PGGEE45E	**8.88269e+01**	3.61293e+03	4.84355e+03	4.11148e+03	1.93562e+04
	PGGEE410E	**8.88269e+01**	3.61293e+03	4.84429e+03	4.11220e+03	1.93562e+04
	PGG	4.63924e+00	3.22735e+02	**4.76098e+02**	5.96483e+02	3.16440e+03
	PGGED	1.89650e+01	4.97906e+02	5.25856e+02	**3.91861e+02**	**1.56364e+03**
F12	PGGEE1E	3.06325e+01	**2.75155e+02**	4.90343e+02	5.88165e+02	2.65138e+03
	PGGEE45E	**3.90095e+00**	3.54209e+02	5.57976e+02	7.50069e+02	3.17666e+03
	PGGEE410E	**3.90095e+00**	3.54209e+02	5.57918e+02	7.50099e+02	3.17666e+03
	PGG	3.53437e+02	6.48406e+02	**6.43104e+02**	1.22059e+02	9.89653e+02
	PGGED	5.69943e+02	6.69520e+02	6.98879e+02	**1.04421e+02**	**9.77665e+02**
F13	PGGEE1E	2.75153e+02	6.42876e+02	6.46422e+02	1.60315e+02	9.90008e+02
	PGGEE45E	**1.97256e+02**	**6.41918e+02**	6.48612e+02	1.46429e+02	9.89653e+02
	PGGEE410E	1.97591e+02	6.41928e+02	6.48340e+02	1.46440e+02	9.89653e+02
	PGG	**9.98863e+01**	2.69640e+03	4.25789e+03	5.07320e+03	2.22566e+04
	PGGED	2.11112e+02	2.39309e+03	**2.66089e+03**	**1.62410e+03**	**5.98448e+03**
F14	PGGEE1E	1.09798e+02	2.31166e+03	4.18319e+03	7.04165e+03	3.92575e+04
	PGGEE45E	**9.98863e+01**	**3.28749e+03**	4.47970e+03	5.04059e+03	2.27698e+04
	PGGEE410E	**9.98863e+01**	3.30572e+03	4.48091e+03	5.04028e+03	2.27698e+04
	PGG*	1.50551e+03	9.10249e+03	1.12164e+04	7.64453e+03	3.05093e+04
	PGGED	**2.27566e+02**	6.34047e+03	7.35257e+03	**4.41348e+03**	**2.24148e+04**
F15	PGGEE1E	9.99613e+02	**3.41395e+03**	**5.13493e+03**	5.47124e+03	2.72265e+04
	PGGEE45E*	1.09950e+03	8.98445e+03	1.07079e+04	6.82002e+03	2.38952e+04
	PGGEE410E*	1.09950e+03	8.98445e+03	1.07078e+04	6.81966e+03	2.38952e+04
	PGG	**6.13112e+02**	**1.49147e+03**	1.86961e+03	1.13160e+03	6.37766e+03
	PGGED	7.91915e+02	1.66019e+03	**1.78506e+03**	**7.84720e+02**	**3.69971e+03**
F16	PGGEE1E	9.36107e+02	2.08976e+03	2.36118e+03	1.35777e+03	6.39645e+03
	PGGEE45E	**6.13112e+02**	1.49212e+03	1.95292e+03	1.25680e+03	6.34435e+03
	PGGEE410E	**6.13112e+02**	1.49325e+03	1.95295e+03	1.25689e+03	6.34480e+03
	PGG	1.01437e+01	**1.11092e+01**	5.55350e+09	2.99065e+10	1.66605e+11
	PGGED	**1.01427e+01**	1.12288e+01	1.14610e+01	1.60068e+00	1.98315e+01
F17	PGGEE1E	1.02531e+01	1.12599e+01	**1.12503e+01**	**4.33217e-01**	**1.20471e+01**
	PGGEE45E	1.01437e+01	1.11215e+01	5.55350e+09	2.99065e+10	1.66605e+11
	PGGEE410E	1.01437e+01	1.11215e+01	5.55350e+09	2.99065e+10	1.66605e+11
	PGG*	3.46555e+01	7.60644e+01	8.38878e+01	**3.76000e+01**	1.90323e+02
	PGGED*	2.63412e+01	1.02578e+02	8.21005e+02	3.00087e+03	1.59688e+04
F18	PGGEE1E	**1.59719e+01**	7.54346e+01	9.08657e+01	4.28061e+01	**1.78512e+02**
	PGGEE45E	3.41315e+01	6.90215e+01	**7.87513e+01**	4.00894e+01	1.94658e+02
	PGGEE410E	3.44179e+01	**6.90147e+01**	7.87820e+01	4.00666e+01	1.94658e+02
	PGG	1.43219e+03	3.11123e+03	3.13171e+03	**8.56063e+02**	**5.14068e+03**
	PGGED	1.65645e+03	**2.89304e+03**	3.25887e+03	1.33690e+03	6.94200e+03
F19	PGGEE1E	**1.13734e+03**	2.93676e+03	2.99717e+03	1.17913e+03	6.61260e+03
	PGGEE45E	1.43219e+03	2.92932e+03	2.94337e+03	9.63226e+02	5.29452e+03
	PGGEE410E	1.43219e+03	2.92932e+03	**2.94280e+03**	9.62423e+02	5.29466e+03

Tabela 5.6: Resultados para as funções em (MOTTA *et al.*, 2017)

F20	PGG	**9.46260e-02**	1.56097e-01	1.58346e-01	2.87419e-02	2.37879e-01
	PGGED	1.05100e-01	1.55742e-01	1.57239e-01	2.54110e-02	2.03227e-01
	PGGEE1E	1.05004e-01	**1.48891e-01**	**1.52120e-01**	**2.44020e-02**	**2.01838e-01**
	PGGEE45E	9.87611e-02	1.54265e-01	1.54571e-01	2.83066e-02	2.03392e-01
	PGGEE410E	9.87624e-02	1.54265e-01	1.54638e-01	2.83821e-02	2.04685e-01
F21	PGG	**5.65314e+01**	**1.98899e+03**	2.24950e+04	1.08169e+05	6.04884e+05
	PGGED	5.51637e+02	2.89011e+03	4.35083e+03	3.24561e+03	**9.59374e+03**
	PGGEE1E	3.58893e+02	2.10183e+03	**3.16602e+03**	**2.44331e+03**	1.00446e+04
	PGGEE45E	2.58323e+02	2.08673e+03	3.76134e+03	7.46448e+03	4.21457e+04
	PGGEE410E	2.58299e+02	2.08673e+03	3.76156e+03	7.46439e+03	4.21457e+04
F22	PGG	1.67640e-01	2.48479e-01	**2.47770e-01**	2.67083e-02	**3.00313e-01**
	PGGED*	2.23265e-01	2.58630e-01	2.57323e-01	**1.98019e-02**	3.21704e-01
	PGGEE1E	2.19848e-01	**2.48478e-01**	2.52500e-01	2.00361e-02	3.24735e-01
	PGGEE45E	**1.46593e-01**	2.49717e-01	2.48003e-01	3.01043e-02	**3.00313e-01**
	PGGEE410E	1.46718e-01	2.49668e-01	2.47964e-01	3.00476e-02	**3.00313e-01**
F23	PGG	**3.10290e+01**	**4.73336e+01**	**4.60150e+01**	4.10836e+00	5.07818e+01
	PGGED	3.60356e+01	4.94871e+01	4.77123e+01	5.86100e+00	6.64206e+01
	PGGEE1E	3.77568e+01	4.91047e+01	4.73692e+01	**3.47028e+00**	**5.06090e+01**
	PGGEE45E	**3.10290e+01**	4.81859e+01	4.70008e+01	4.12407e+00	5.27829e+01
	PGGEE410E	**3.10290e+01**	4.81859e+01	4.70017e+01	4.12448e+00	5.27829e+01

Tabela 5.7: Somatório do número de vezes em que determinada técnica obteve melhor resultado nos problemas de regressão para cada uma das categorias analisadas. O número de vezes que determinada técnica foi considerada inferior pelo teste não paramétrico de Wilcoxon está representado pelo asterisco

	Somatório						
	Min	Mediana	Média	DP	Max	Total	*
PGG	5	6	5	5	5	26	2
PGGED	5	1	3	11	8	28	4
PGGEE1E	7	9	9	6	7	38	0
PGGEE45E	9	6	3	1	4	23	1
PGGEE410E	7	5	3	0	5	20	1

Em todas funções utilizadas, a PGGEE1E teve no mínimo a mesma relevância estatística quando comparada a PGG, PGGED, PGGEE45E e PGGEE410E, como mostrado na Tabela 5.4, Tabela 5.5 e Tabela 5.6. Isso significa que mesmo quando essa técnica tem pior resultado nos valores das medianas, não pode ser considerado estatisticamente pior do que o de melhor resultado. A PGG padrão foi marcada em dois casos, ou seja, nessas funções ela pode ser considerada estatisticamente diferente da melhor. Estatisticamente a técnica de pior resultado é a PGGED, sendo marcada em 4 funções.

Finalmente, é possível encontrar valores "Inf" como resultados de F3. Isso pode ser explicado por o pior modelo obtido pela PGG conter indefinição (como por exemplo divisão por zero) quando avaliado no conjunto de teste. Essa indefinição gera um resultado "*Not a number (nan)*"(não é um número) no desvio padrão.

5.4 CLASSIFICAÇÃO

A gramática formal para os problemas de classificação é composta pelo seguinte conjunto de regras de derivação:

$$
\begin{aligned}
< expr > &::= \ < s > else < result > \\
< s > &::= \ < cond > if < result > \ | \ < s >< s > \\
< cond > &::= \ < var >< val >< comparison > \ | \ < cond >< cond >< logical > \\
< logical > &::= and \mid or \\
< comparison > &::= \ < \mid \ <= \ \mid \ > \ \mid \ >= \\
< var > &::= x_1 \mid x_2 \mid ... \mid x_{dimensao} \\
< val > &::= \ < const > \ \mid \ < const >< op >< const > \\
< result > &::= zero \mid ... \mid numeroDeClasses \\
< op > &::= \ + \ \mid \ - \ \mid \ * \ \mid / \mid pow \\
< const > &::= \ -5 \mid \ -4 \mid \ -3 \mid \ -2 \mid \ -1 \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5
\end{aligned}
$$

As bases de classificação utilizadas neste trabalho são:

5.4.1 *IONOSPHERE*

Criada pelo grupo de física espacial da universidade Johns Hopkins. A base de dados descreve registros de radares feitos na ionosfera. Foi criada para observar elétrons livres da ionosfera, os registros mostram quando algum tipo de estrutura foi detectada (SIGILLITO *et al.*, 1989).

É composta por 361 registros, cada um composto por 33 atributos, sendo 30 contínuos e 3 inteiros. As classes são descritas como *Good* (g) que ocorre quando o radar retorna evidencias de algum tipo de estrutura na ionosfera. E *Bad* para quando não há evidencias; seu sinal passou diretamente pela ionosfera. A frequência de registros das classes está descrita na Tabela 5.8.

Tabela 5.8: Frequência de elementos para cada classe da base *Ionosphere*

Classe	g	b
Quantidade	225	126

5.4.2 IRIS

Provavelmente a base de dados mais conhecida utilizada para classificação, a base foi catalogada em 1935 por Anderson (1935) e disponibilizada por Fisher (1936). Foi criada para descrever diversas medições realizadas sobre a planta iris, é composta por 150 registros, contendo quatro atributos cada, sendo eles: comprimento e largura da sépala, assim como comprimento e largura das pétalas (todos valores reais). Os dados são distribuídos uniformemente em três classes diferentes: setosa, versicolor e virginica. Numeradas de 1 a 3 respectivamente, a distribuição está descrita na Tabela 5.9.

Tabela 5.9: Frequência de elementos para cada classe da base Iris

Classe	1	2	3
Quantidade	50	50	50

5.4.3 *SEED*

Base criada por Charytanowicz *et al.* (2010), ela descreve através de 7 atributos reais, três variedades de trigo, Kama, Rosa e Canadense. Cada um com 70 elementos na base, totalizando 210 instâncias, assim como pode ser visto na Tabela 5.10.

Tabela 5.10: Frequência de elementos para cada classe da base *Seed*

Classe	1	2	3
Quantidade	70	70	70

5.4.4 *STEEL PLATES*

Essa base descreve placas de aço através de 33 atributos, como, luminosidade e orientação. É composta por 1941 instâncias cada uma classificada como defeituosa ou não, a distribuição das classes pode ser vista na Tabela 5.11. A base foi doada por *Semeion, Research Center of Sciences of Communication, Via Sersale 117, 00128, Rome, Italy.* E utilizada pela primeira vez em (BUSCEMA *et al.*, 2010).

Tabela 5.11: Frequência de elementos para cada classe da base *Steel Plates*

Classe	1	2
Quantidade	1269	673

5.4.5 *VERTEBRAL*

Base de dados construída em 2011 pelo *Group of applied Research in Orthopaedics (GARO)* em Lyon, França. A classificação é utilizada para determinar se o paciente está com a coluna normal ou tem hérnia de disco e Espondilolistese. Caracterizadas por 6 atributos reais, a base contem 310 instâncias, das quais a distribuição pode ser vista na Tabela 5.12.

Tabela 5.12: Frequência de elementos para cada classe da base *Vertebral*

Classe	1	2
Quantidade	210	100

5.4.6 *BREAST CANCER*

Uma das bases mais utilizadas para avaliar técnicas de classificação, criada na universidade de Wisconsin Hospital (MANGASARIAN *et al.*, 1990). Cada registro contém nove valores relacionados a tumores localizados nas mamas das pacientes. As amostras tem atributos que descrevem a forma do tumor, como por exemplo, uniformidade no formato da célula e espessura do tumor.

A base contém 32 dimensões, todas de valores reais. São 569 amostras que classificam o câncer como maligno ou benigno, e a quantidade de dados de cada classe pode ser vista

na Tabela 5.13. Sendo 1 para benigno e 2 para maligno.

Tabela 5.13: Frequência de elementos para cada classe da base *Breast Cancer*

Classe	1	2
Quantidade	357	212

5.4.7 *WINE*

Essa base contém resultados de análises químicas de vinhos cultivados em uma mesma região da Itália porém vindos de três diferentes cultivadores. São 178 registros contendo a quantidade de 13 constituintes encontrados em cada um dos três tipos de vinho. A base contém apenas valores numéricos, sendo estes representados por valores reais. A quantidade presente de tipos de vinho está representada na Tabela 5.14

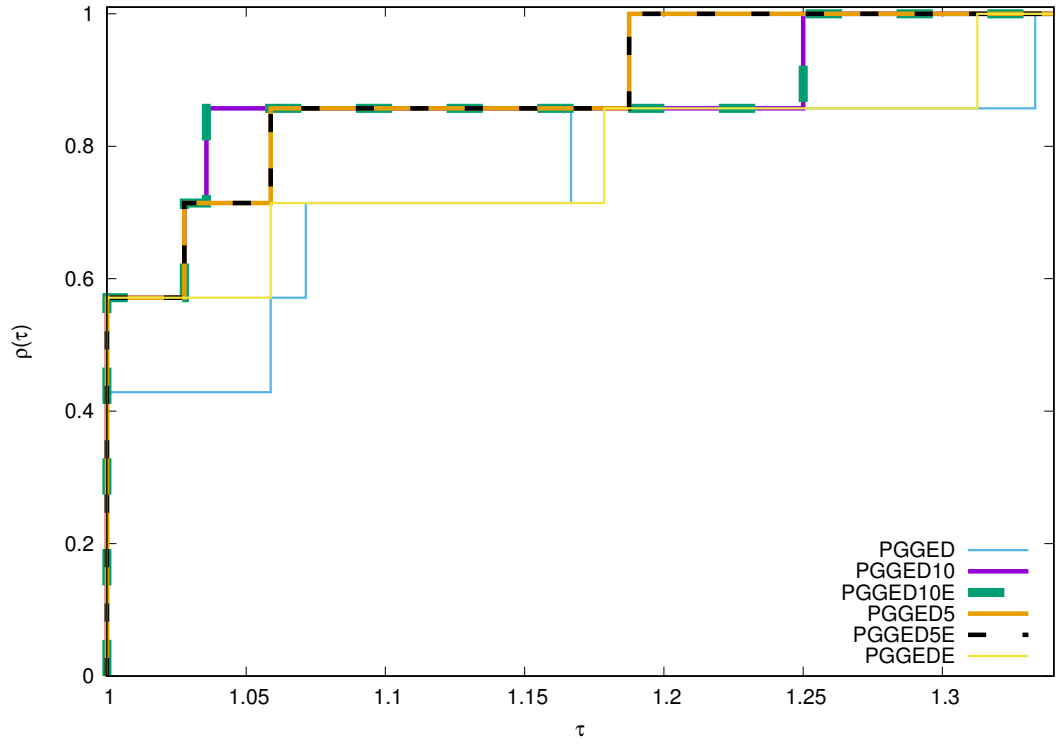
Tabela 5.14: Frequência de elementos para cada classe da base *Wine*

Classe	1	2	3
Quantidade	59	71	48

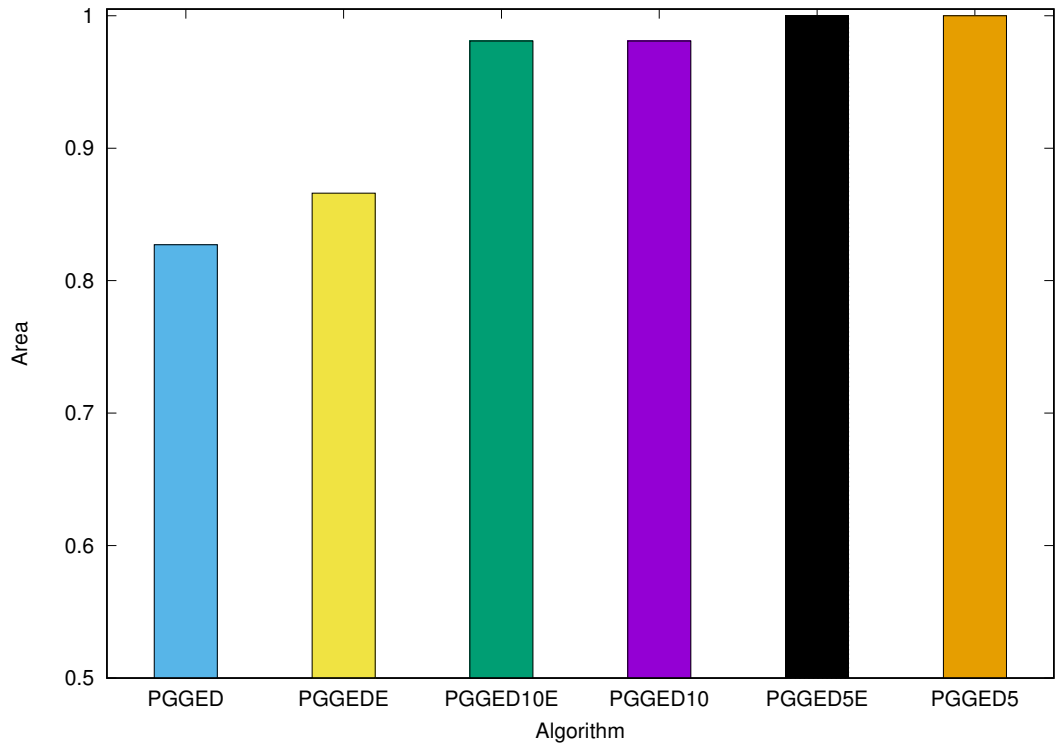
5.4.8 RESULTADOS

A Figura 5.7a representa os PD das técnicas que executam a ED em conjunto com a PGG. Entre as técnicas comparadas, é possível perceber que ao contrário dos problemas de regressão simbólica, utilizar a ED para evoluir apenas o melhor indivíduo da população da PGG durante a execução da mesma apresenta os piores resultados como pode ser visto na Figura 5.7b. Nesse caso a utilização da ED após a evolução da PGG apresentou os melhores resultados. Além disso é possível perceber que em todos os casos que a técnica executa após a PGG não há diferença entre melhorar apenas o melhor indivíduo ou a elite da população. Segue-se então com a PGGED5 para comparações futuras.

Os resultados de todas as técnicas que usam EE durante a evolução da PGG podem ser vistos na Figura 5.8a. De forma geral a PGGEE4 obteve melhores resultados, e a AsC é a maior como representado na Figura 5.8b. Também é interessante perceber que a PGGEE1E que foi a de melhor desempenho para os problemas de regressão simbólica, aqui apresenta o pior desempenho. As técnicas que otimizaram a elite durante a evolução



(a) As AsC normalizadas são, 0.845, 0.995, 0.995, 1.0, 1.0 e 0.841, respectivamente para PGGED, PGGED10, PGGED10E, PGGED5, PGGED5E, PGGEDE.



(b) Gráfico de barra para as AsC normalizadas

Figura 5.7: PD das técnicas que utilizam ED para classificação

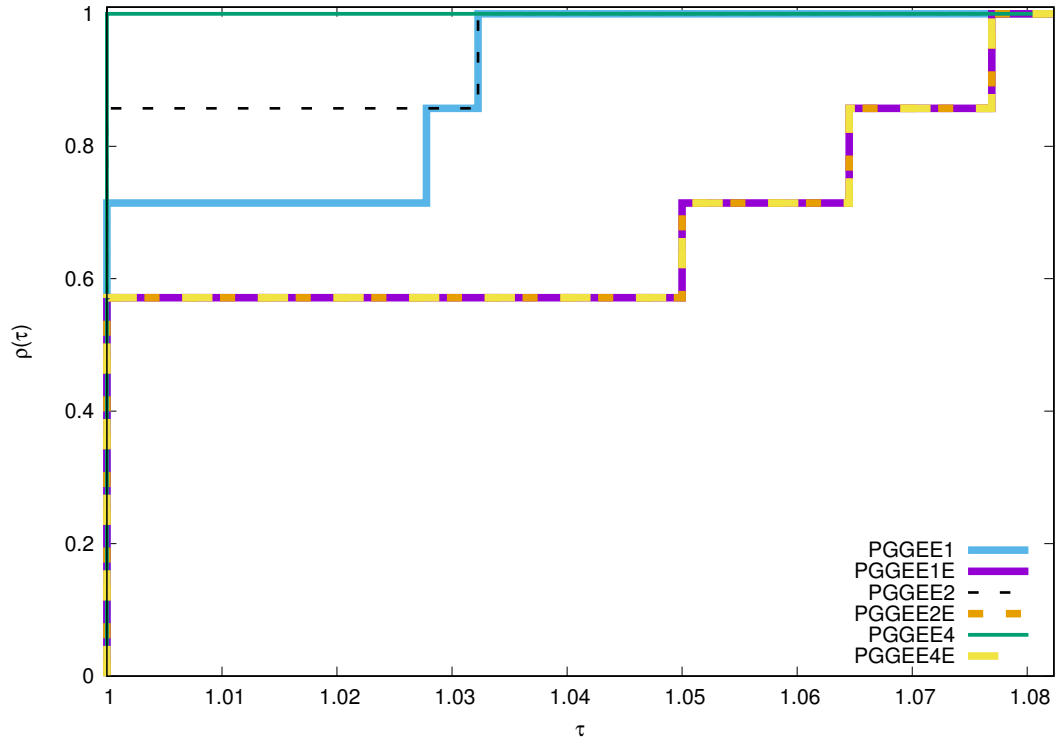
também foram piores que a versão de otimização para um indivíduo para este conjunto de problemas.

Para as técnicas que utilizam 5% dos cálculos de função objetivo nos Θ indivíduos utilizando a EE após a evolução da PGG, todas que aplicam a otimização apenas no melhor indivíduo apresentam melhores resultados, como pode ser visto nas Figuras 5.9a e 5.9b. Nesse caso a estratégia PGGEE25E será utilizada para análise futura.

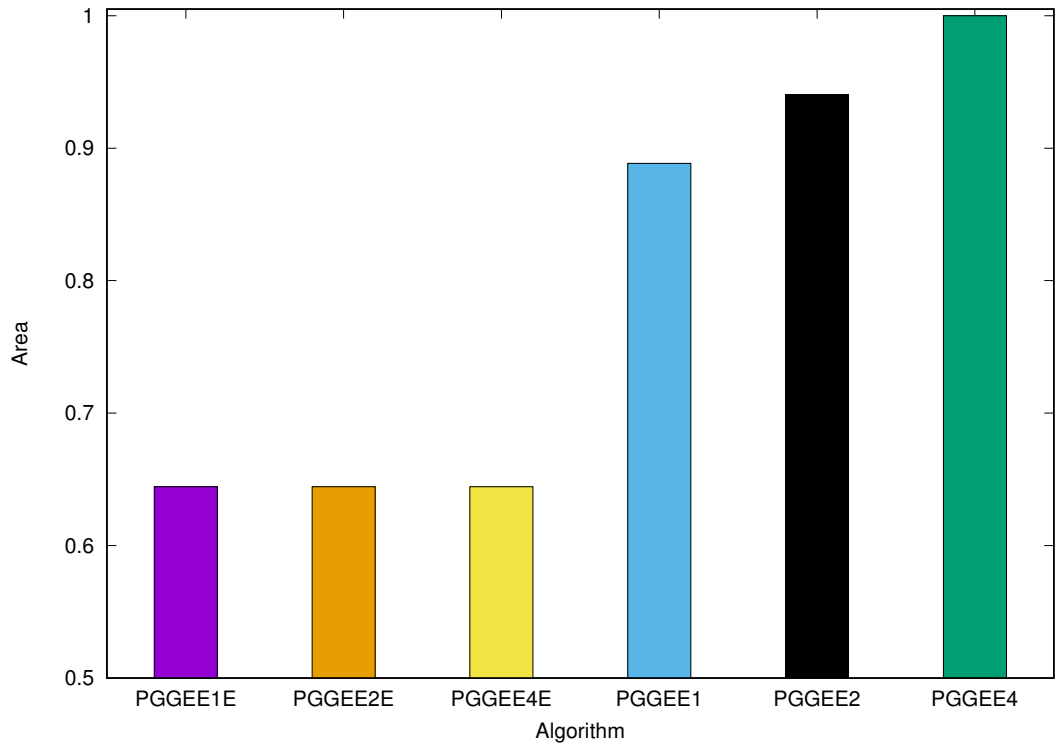
Por fim, as técnicas que utilizam 10% dos cálculos de função objetivo nos Θ indivíduos utilizando a EE após a evolução da PGG. Através da análise da Figura 5.10a pode-se perceber que a otimização apenas no melhor indivíduo apresenta melhora no resultado final. PGGEE210 obteve melhores resultados quando analisado a AsC como representado na Figura 5.5b.

A Figura 5.11a apresenta os PD das quatro melhores técnicas selecionadas anteriormente com a inclusão da PGG padrão. Na Figura 5.11b, tem-se o gráfico de barra para as AsC normalizadas.

A PGGED5 obteve a maior AsC normalizada, 6.5% maior que a melhor técnica com EE e 7% maior que a PGG padrão. Junto com a PGG padrão, a PGGEE4 obteve o melhor valor no pior caso, significando que são mais confiáveis que os outros.

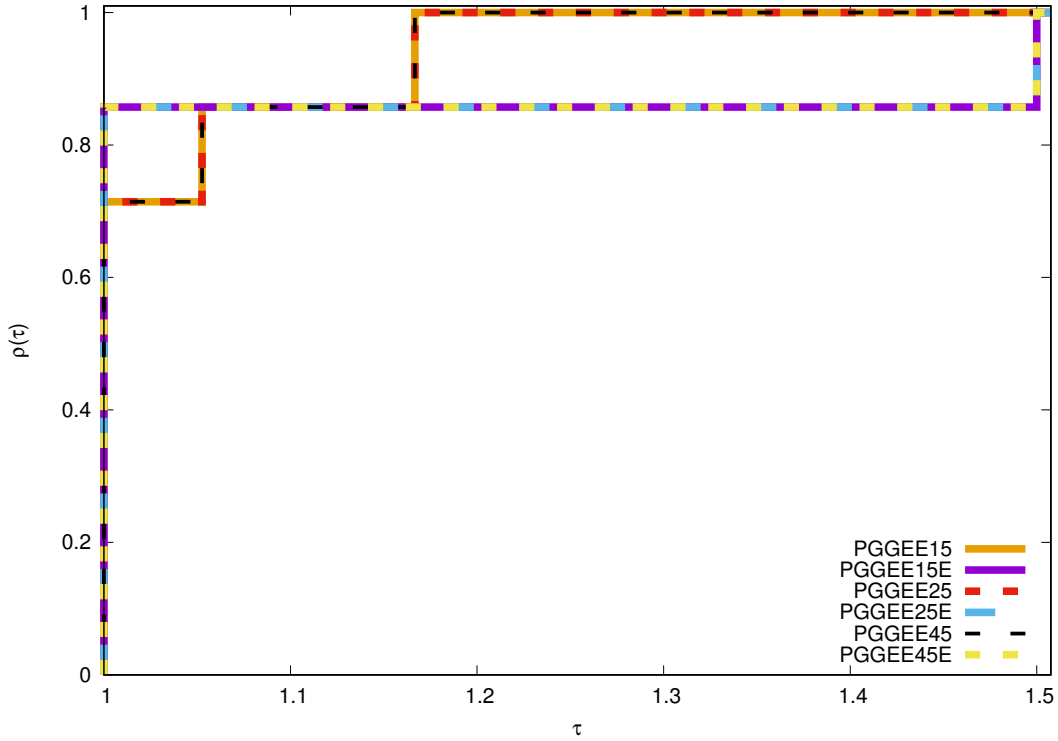


(a) As AsC normalizadas são, 0.881, 0.645, 0.947, 0.645, 1.0 e 0.645, respectivamente para PGGEE1, PGGEE1E, PGGEE2, PGGEE2E, PGGEE4, PGGEE4E.

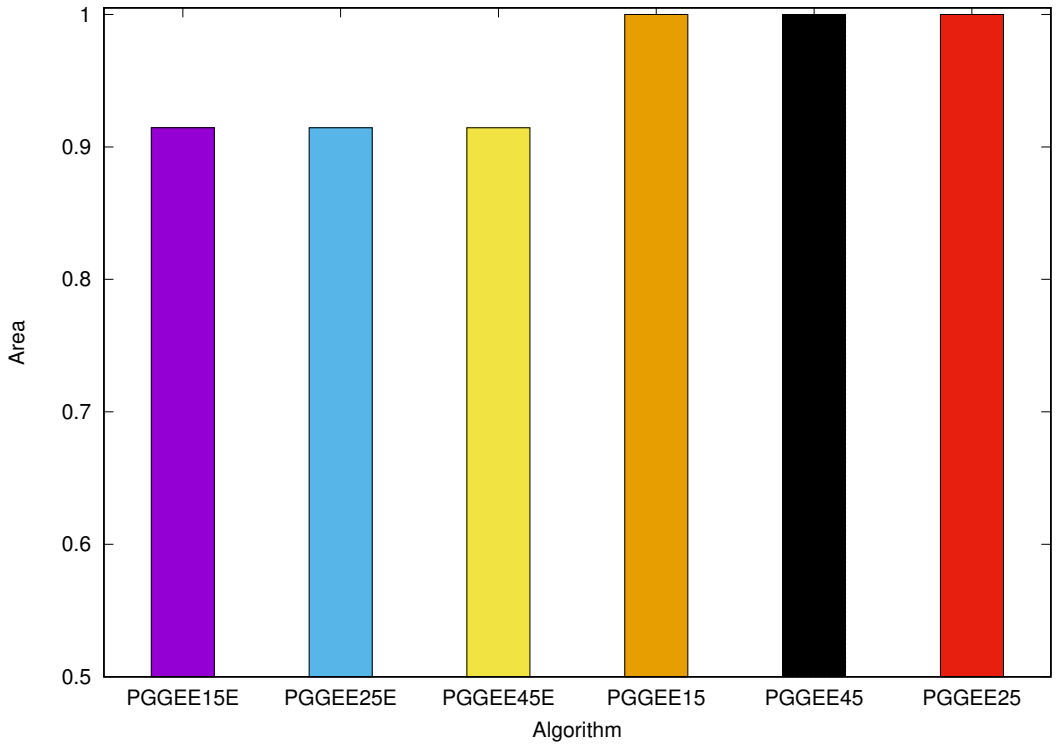


(b) Gráfico de barra para as AsC normalizadas

Figura 5.8: PD das técnicas que utilizam EE durante a evolução da PGG para classificação

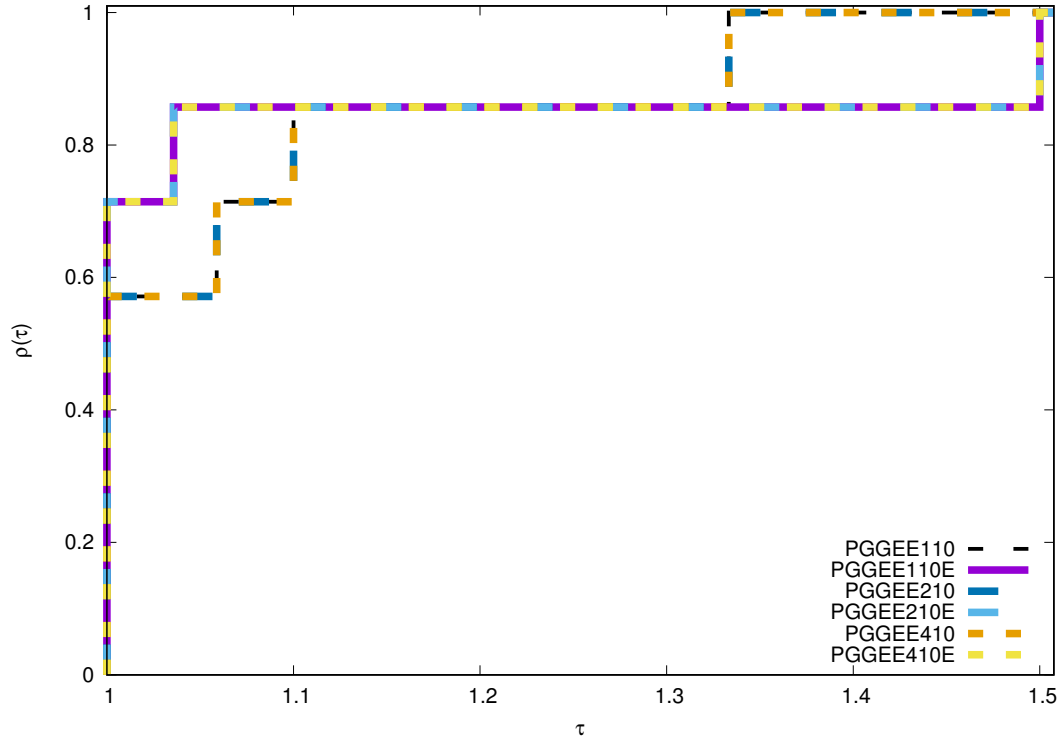


(a) As AsC normalizadas são, 1.0, 0.921, 1.0, 0.921, 1.0 e 0.921, respectivamente para PGGEE15, PGGEE15E, PGGEE25, PGGEE25E, PGGEE45, PGGEE45E.

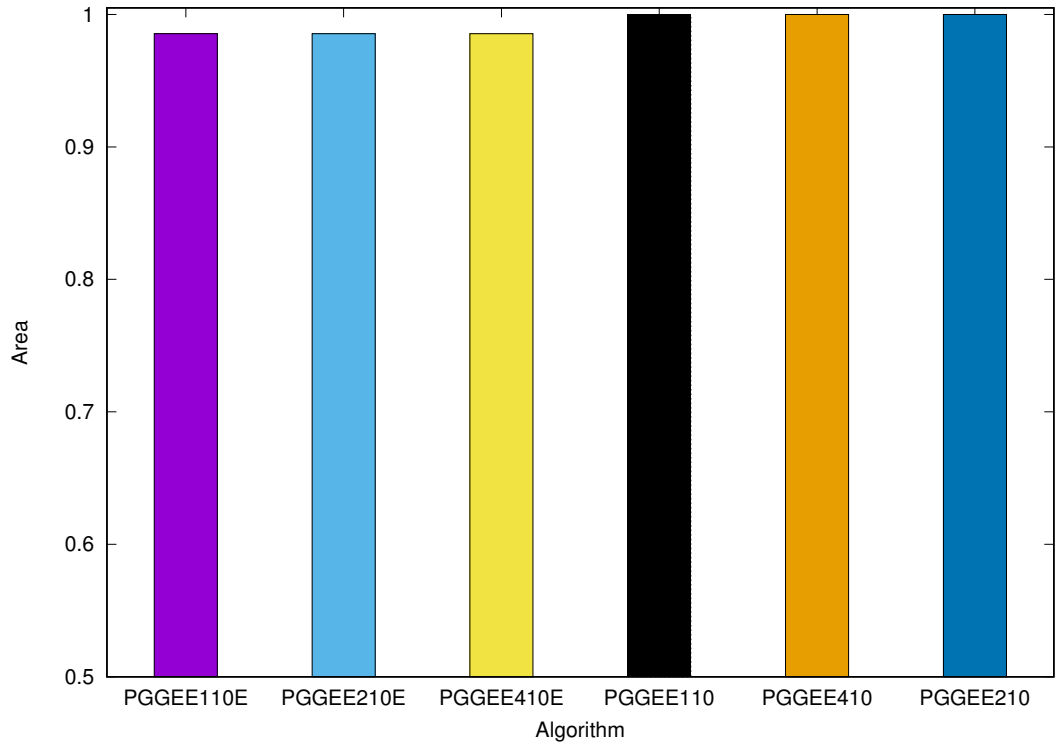


(b) Gráfico de barra para as AsC normalizadas

Figura 5.9: PD das técnicas que utilizam EE após a evolução da PGG com 5% de orçamento para classificação

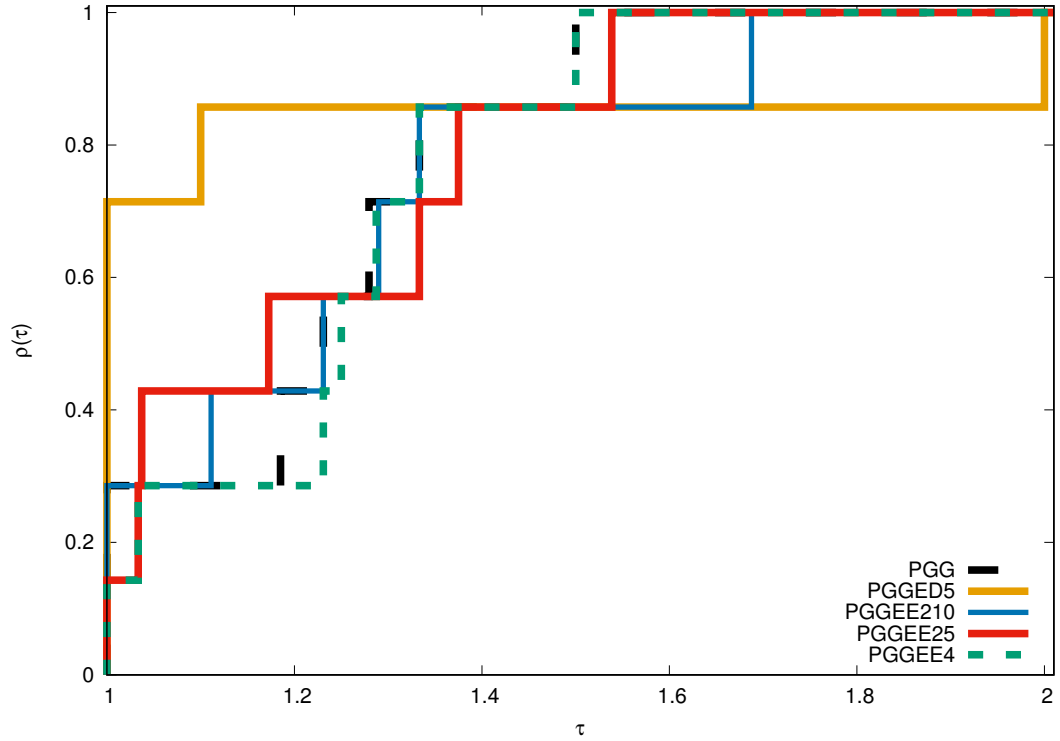


(a) As AsC normalizadas são, 1.0, 0.987, 1.0, 0.987, 1.0 e 0.987, respectivamente para PGGEE110, PGGEE110E, PGGEE210, PGGEE210E, PGGEE410, PGGEE410E.

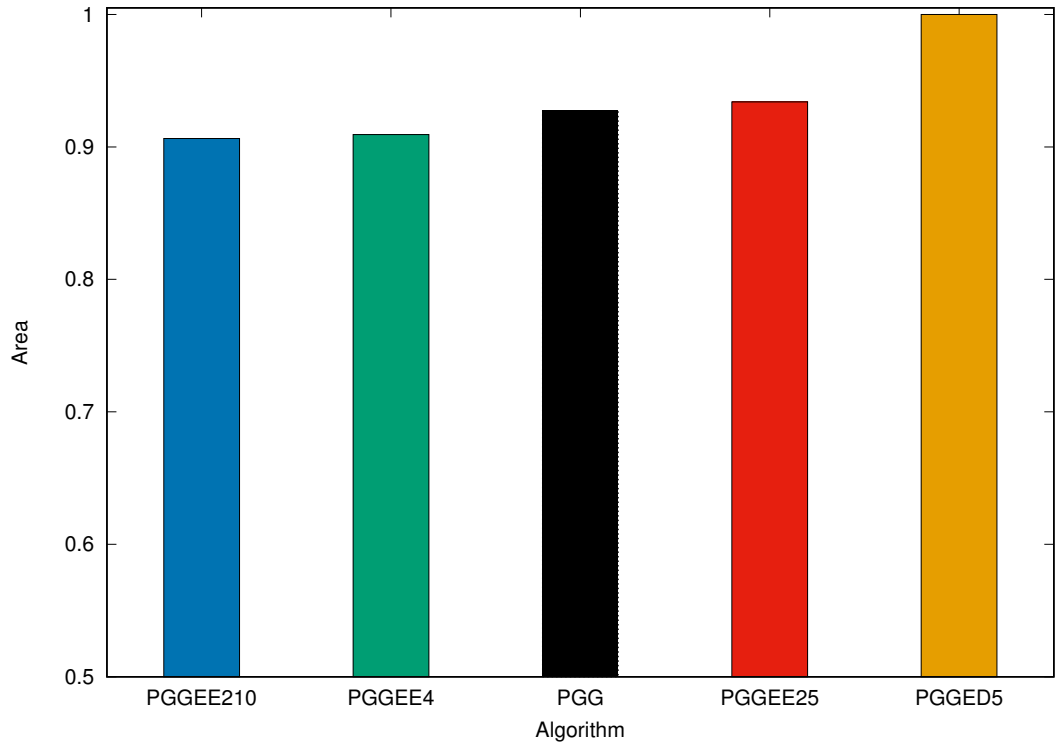


(b) Gráfico de barra para as AsC normalizadas

Figura 5.10: PD das técnicas que utilizam EE após a evolução da PGG com 10% de orçamento para classificação



(a) As AsC normalizadas são, 0.927, 1.0, 0.906, 0.934, 0.909 respectivamente para PGG, PGGED5, PGGEE210, PGGEE25, PGGEE4.



(b) Gráfico de barra para as AsC normalizadas

Figura 5.11: PD das melhores técnicas selecionadas anteriormente e a PGG padrão para classificação

Tabela 5.15: Resultados para classificação

P	Técnica	Min	Mediana	Média	DP	Max
	PGG	0.00000e+00	**0.00000e+00**	**2.33333e-02**	**3.45875e-02**	**1.00000e-01**
	PGGED5	0.00000e+00	**0.00000e+00**	**2.33333e-02**	**3.45875e-02**	**1.00000e-01**
Iris	PGGEE4*	0.00000e+00	3.33333e-02	5.66667e-02	5.97217e-02	1.66667e-01
	PGGEE25	0.00000e+00	**0.00000e+00**	**2.33333e-02**	**3.45875e-02**	**1.00000e-01**
	PGGEE110	0.00000e+00	**0.00000e+00**	**2.33333e-02**	**3.45875e-02**	**1.00000e-01**
	PGG	1.80412e-01	**2.79640e-01**	2.78350e-01	3.32997e-02	3.47938e-01
	PGGED5	1.82990e-01	2.87371e-01	2.81615e-01	3.19209e-02	3.50515e-01
Steel-plates	PGGEE4	**1.18557e-01**	2.87371e-01	**2.77234e-01**	4.50044e-02	**3.42784e-01**
	PGGEE25*	1.82990e-01	2.89949e-01	2.83333e-01	**3.08773e-02**	3.45361e-01
	PGGEE110	1.80412e-01	2.80928e-01	2.80498e-01	3.20120e-02	3.50515e-01
	PGG*	6.19469e-02	1.06195e-01	1.13274e-01	2.76467e-02	1.59292e-01
	PGGED5*	**4.42478e-02**	1.32743e-01	1.17404e-01	3.03073e-02	1.59292e-01
breast-cancer	PGGEE4	5.30973e-02	**8.84956e-02**	**9.35103e-02**	**2.40390e-02**	**1.32743e-01**
	PGGEE25*	6.19469e-02	1.19469e-01	1.15044e-01	2.84472e-02	1.50442e-01
	PGGEE110*	**4.42478e-02**	1.19469e-01	1.10914e-01	2.53137e-02	1.59292e-01
	PGG	7.14286e-02	1.42857e-01	**1.28095e-01**	**2.93891e-02**	2.14286e-01
	PGGED5	7.14286e-02	1.42857e-01	1.31429e-01	4.01358e-02	2.42857e-01
ionosphere	PGGEE4	7.14286e-02	1.42857e-01	1.33810e-01	3.71582e-02	2.14286e-01
	PGGEE25	7.14286e-02	1.42857e-01	1.30476e-01	3.12404e-02	**2.00000e-01**
	PGGEE110	7.14286e-02	1.42857e-01	1.32381e-01	3.90243e-02	2.42857e-01
	PGG	**1.42857e-01**	**1.90476e-01**	2.07143e-01	4.43948e-02	3.57143e-01
	PGGED5	**1.42857e-01**	**1.90476e-01**	2.09524e-01	4.45011e-02	3.57143e-01
seed	PGGEE4	**1.42857e-01**	**1.90476e-01**	**1.95853e-01**	4.97634e-02	**3.33333e-01**
	PGGEE25	**1.42857e-01**	**1.90476e-01**	2.09524e-01	**4.32084e-02**	3.57143e-01
	PGGEE110*	1.66667e-01	2.61905e-01	2.56349e-01	4.54602e-02	3.57143e-01
	PGG	1.61290e-01	2.58065e-01	2.39247e-01	3.97453e-02	3.06452e-01
	PGGED5	1.61290e-01	2.41935e-01	2.38710e-01	3.84853e-02	3.22581e-01
vertebral	PGGEE4	1.61290e-01	**2.17742e-01**	**2.17742e-01**	4.42205e-02	**2.90323e-01**
	PGGEE25	**1.45161e-01**	2.41935e-01	2.30107e-01	4.03623e-02	**2.90323e-01**
	PGGEE110	1.61290e-01	2.25806e-01	2.22043e-01	**2.75247e-02**	3.06452e-01
	PGG	2.85714e-02	**2.85714e-02**	**5.04762e-02**	4.81965e-02	2.57143e-01
	PGGED5	2.85714e-02	**2.85714e-02**	5.52381e-02	4.83375e-02	2.57143e-01
wine	PGGEE4	**0.00000e+00**	4.28572e-02	5.71429e-02	**4.42628e-02**	**1.71429e-01**
	PGGEE25*	2.85714e-02	**2.85714e-02**	5.42857e-02	4.91285e-02	2.57143e-01
	PGGEE110*	2.85714e-02	**2.85714e-02**	5.90476e-02	5.56633e-02	2.28571e-01

A Tabela 5.15 apresenta os seguintes valores para os erros dos modelos: mínimo, mediana, média, desvio padrão (DP) e valor máximo para cada técnica testada em todas funções utilizadas. Os melhores resultados encontrados estão destacados em negrito.

Assim como nos problemas de regressão, para melhor visualização dos resultados obtidos pelas técnicas a Tabela 5.16 representa um somatório do número de vezes que determinada técnica obteve o melhor resultado para cada categoria analisada.

A PGGEE4 obteve a maioria dos melhores resultados (5 dos 7 problemas de classificação considerados aqui, como visto na Tabela 5.16).

No geral, a PGGEE4 obteve uma das menores AsC e piores resultados do que a PGG padrão, apesar disso ela apresenta melhor confiabilidade, devido aos melhores resultados obtidos no pior caso. Considerando o valor máximo, a PGGEE4 superou os outros

Tabela 5.16: Somatório do número de vezes em que determinada técnica obteve melhor resultado nos problemas de classificação para cada uma das categorias analisadas. O número de vezes que determinada técnica foi considerada inferior pelo teste não paramétrico de Wilcoxon está representado pelo asterisco

	Somatório						
	Min	Mediana	Média	DP	Max	Total	*
PGG	3	4	3	2	1	13	1
PGGED5	4	3	1	1	1	10	1
PGGEE4	5	4	4	2	5	20	1
PGGEE25	4	3	1	3	3	14	3
PGGEE110	3	3	1	2	1	10	3

algoritmos em 5 dos 7 problemas.

Com relação aos melhores resultados encontrados, a PGGEE4 obteve melhor desempenho, tendo obtido o melhor resultado em 5 dos 7 problemas.

A PGGEE25 obteve melhores resultados quando analisado o desvio padrão, ela conseguiu alcançar o melhor resultado em 3 (de 7) funções. Isso sugere que a técnica seja mais robusta que as outras.

Apesar de apresentar o melhor resultado de mediana em 4 dos 7 problemas testados. A PGG obteve melhor resultado sem empate em apenas 1 das 7 funções de classificação. Olhando dessa forma, a PGGEE4 foi a de melhor desempenho sem empate, com 2 problemas.

Também pode-se perceber que em relação a média, a PGGEE4 obteve melhores resultados em 4 das 7 funções testadas, sendo a de melhor desempenho entre todas as técnicas.

Para análise estatística, mais uma vez foi utilizado o teste de Wilcoxon. Assim como nos casos de regressão, sempre que o teste indica diferença entre duas técnicas, a pior é marcada com um asterisco.

Para esse conjunto de problemas de classificação, todas as técnicas obtiveram uma relevância estatística inferior à de melhor resultado em pelo menos um problema. Ambas as técnicas que utilizam a EE após a evolução apresentaram relevância estatística inferior que as demais em 3 problemas testados. Na base *breast-cancer* a PGGEE4 superou estatisticamente as demais técnicas. No geral, tanto a PGG, como a PGGED5 e a PGGEE4 foram marcadas em apenas um caso, ou seja, nessas funções elas podem ser consideradas estatisticamente diferentes da melhor.

Para as 3 bases com 30 ou mais dimensões, a PGGEE4 obteve um desempenho melhor

quanto ao pior valor obtido em 2 dos três casos. Nessas bases, tanto a PGGEE4, PGGED5 e PGGEE110 obtiveram melhor resultados em 2 casos. Quanto ao teste de Wilcoxon, nesses 3 casos, apenas a PGGEE4 obteve desempenho igual ou superior aos demais.

5.5 CONSIDERAÇÕES FINAIS

Para os problemas de regressão, a proposta que utiliza a EE durante a execução da PGG apresentou melhores resultados, enquanto nos problemas de classificação a proposta com ED após a evolução da PGG foi a de maior AsC nos PD. As propostas com melhores resultados para os problemas de regressão, não foram as mesmas que as para os problemas de classificação. O que indica uma diferença na análise do tipo de problema.

Para os problemas de regressão simbólica, o número de indivíduos a serem otimizados nas propostas que utilizam EE após a evolução da PGG não é fator determinante para o resultado, uma vez que esse valor não alterou o desempenho final. Das 24 propostas testadas em classificação, todas que utilizaram elite para evolução das constantes obtiveram pior resultado que sua versão para apenas um indivíduo.

6 CONCLUSÕES

A Hibridização da Programação Genética Gramatical (PGG) com Evolução Diferencial (ED) e Estratégia Evolutiva (EE) foi proposta neste trabalho, no qual utiliza-se a ED/EE para melhorar os coeficientes numéricos dos modelos encontrados pela PGG. Também foi sugerido uma atualização da gramática durante a evolução do algoritmo, com a inserção dos coeficientes numéricos encontrados pelo otimizador. As técnicas desenvolvidas foram comparadas entre si e com a PGG padrão. A utilização de um otimizador exclusivo para as constantes dos indivíduos da PGG apresentou melhora no modelo criado.

Para os problemas de regressão simbólica, a técnica que utiliza a estratégia (1 + 1)-EE aplicada na elite da população durante a evolução da PGG foi a de melhor resultado. Particularmente, os melhores valores de mediana foram encontrados pela PGGEE1E em 12 das 23 funções testadas.

Nos problemas de classificação, a técnica que utiliza ED após a evolução da PGG padrão com 5% de orçamento foi a que apresentou maior AsC. No total, o melhor valor de mediana dos 7 problemas testados.

No geral é possível perceber que os algoritmos propostos apresentam bons resultados, tanto em problemas de classificação como em problemas de regressão simbólica. A diferença de comportamento das técnicas propostas entre os dois tipos de problemas é visível. Enquanto nos problemas de regressão simbólica a aplicação do otimizador na elite ou no melhor indivíduo não surtiu muita diferença, a variação dos parâmetros do otimizador (principalmente o número de filhos da EE) fez o resultado das áreas normalizadas sob a curva variar. Já para os problemas de classificação, o contrário pode ser percebido. A variação dos parâmetros do otimizador não surtiram muita diferença, mas a definição do valor Θ (número de indivíduos a terem seus coeficientes numéricos melhorados) trouxe melhora aos resultados.

Diversas modificações podem ser estudadas em trabalhos futuros, como por exemplo: uma análise de sensibilidade dos parâmetros (Momentos de execução do otimizador, tamanho do Θ e orçamento do otimizador). Também pode ser realizado um estudo sobre as técnicas de melhor desempenho aplicadas a problemas mais complexos, como exemplo, a inferência de modelos de redes de regulação gênica.

REFERÊNCIAS

ALONSO, C. L.; MONTAñA, J. L.; BORGES, C. E. Evolution strategies for constants optimization in genetic programming. In: **IEEE International Conference on Tools with Artificial Intelligence**, 2009. p. 703–707.

AMARAL, F. **Introdução à ciência de dados, Alta Books**, 2016.

ANDERSON, E. The Irises of the Gaspe Peninsula. **Bulletin of the American Iris Society**, v. 59, p. 2–5, 1935.

BANZHAF, W. Genetic programming for pedestrians. In: **Proceedings of the 5th International Conference on Genetic Algorithms**, 1993. p. 628–.

BARBOSA, H. J. C.; BERNARDINO, H. S.; BARRETO, A. M. S. Using performance profiles to analyze the results of the 2006 cec constrained optimization competition. In: **IEEE Congress on Evolutionary Computation**, 2010. p. 1–8.

BERSINI M. DORIGO, S. L. G. S. L. G. H. Results of the first international contest on evolutionary optimisation (1st iceo). In: **Proceedings of IEEE International Conference on Evolutionary Computation**, 1996.

BOCCATO, L.; ATTUX, R.; ZUBEN, F. **Evolução Diferencial - Introdução e conceitos básicos**. 2009. University Lecture.

BORN, J. Numerische optimierung von computer-modellen mittels der evolutionsstrategie. **ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik**, WILEY-VCH Verlag, v. 60, n. 5, p. 272–272, 1980.

BROWNLEE, J. **Clever Algorithms: Nature-Inspired Programming Recipes, Lulu Enterprises**, 2012. 438 p.

BUSCEMA, M.; TERZI, S.; TASTLE, W. A new meta-classifier. In: **Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS**, 2010. p. 1–7. ISBN 978-1-4244-7859-0.

BäCK, T.; FOGEL, D.; MICHALEWICZ, Z. **Evolutionary Computation 1—Basic Algorithms and Operators, CRC Press**, 2000.

BäCK, T.; HOFFMEISTER, F.; SCHWEFEL, H.-P. A survey of evolution strategies. In: **Proceedings of the International Conference on Genetic Algorithms**, 1991. p. 2–9.

CHARYTANOWICZ, M.; NIEWCZAS, J.; KULCZYCKI, P.; KOWALSKI, P. A.; LUKASIK, S.; ZAK, S. Complete gradient clustering algorithm for features analysis of x-ray images. In: PIETKA, E.; KAWA, J. (Ed.). **Information Technologies in Biomedicine**, 2010. p. 15–24.

CHENG, S.-L.; HWANG, C. Optimal approximation of linear systems by a differential evolution algorithm. **IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans**, v. 31, n. 6, p. 698–707, Nov 2001.

CHOMSKY, N. Three models for the description of language. **IRE Transactions on Information Theory**, v. 2, n. 3, p. 113–124, September 1956.

CHOMSKY, N. **Syntactic Structures, Mouton & Co**, 1957.

DARWIN, C. **On the origin of species by means of natural selection, or, The preservation of favoured races in the struggle for life, London:John Murray**, 1859. 564 p.

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

DEKLEL, A. K.; SALEH, M. A.; HAMDY, A. M.; SAAD, E. M. Transfer learning with long term artificial neural network memory (ltann-mem) and neural symbolization algorithm (nsa) for solving high dimensional multi-objective symbolic regression problems. In: **National Radio Science Conference (NRSC)**, 2017. p. 343–352.

DEVANEY, J. E.; HAGEDORN, J. G. The role of genetic programming in describing the microscopic structure of hydrating plaster. In: **Late Breaking papers at the Genetic and Evolutionary Computation Conference (GECCO)**, 2002. p. 9–13.

DIVEEV, A. I.; SHMALKO, E. Y.; SOFRONOVA, E. A.; ZHADNOV, V. V. Automatic search of reliability function by symbolic regression. In: **International Conference on Control, Decision and Information Technologies (CoDIT)**, 2017. p. 0061–0066.

DOLAN, E.; MORé, J. Benchmarking optimization software with performance profiles. **Mathematical Programming**, v. 91, p. 201–213, 2002.

DRAPER, N. R.; SMITH, H. **Applied Regression Analysis, Third Edition, Wiley-Interscience**, 2014. 1-13 p.

EIGEN, M. Evolutionsstrategie - optimierung technischer systeme nach prinzipien der biologischen evolution. **mit einem Nachwort von Manfred Eigen**, v. 45, p. 46–47, 1973.

EL-MIHOUB, T. A.; HOPGOOD, A. A.; NOLLE, L.; BATTERSBY, A. Hybrid genetic algorithms: A review. **Engineering Letters**, v. 13, p. 124–137, 2006.

EVETT, M.; FERN, T. Numeric mutation: Improved search in genetic programming. In: **Proc. of the 11th Intl. Florida Artificial Intelligence Research Society Conference**, 1998.

FISHER, R. A. The use of multiple measurements in taxonomic problems. **Annals Eugen.**, v. 7, p. 179–188, 1936.

FOGEL, D. B. **Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, Wiley-IEEE Press**, 2005.

FUNIE, A. I.; SALMON, M.; LUK, W. A hybrid genetic-programming swarm-optimisation approach for examining the nature and stability of high frequency trading strategies. In: **2014 13th International Conference on Machine Learning and Applications**, 2014. p. 29–34.

GEYER-SHULZ, A. **Fuzzy Rule-Based Expert Systems and Genetic Machine Learning, Physica-Verlag Heidelberg**, 1997.

GREFENSTETTE, J. J.; BAKER, J. E. How genetic algorithms work: A critical look at implicit parallelism. In: **Proceedings of the International Conference on Genetic Algorithms**, 1989. p. 20–27.

HANSEN, N.; ARNOLD, D.; AUGER, A. Evolution strategies. In: J., K.; W., P. (Ed.). **Springer Handbook of Computational Intelligence**, 2015. cap. 44, p. 871–898.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, University of Michigan Press**, 1975.

HOPCROFT J.E.; MOTWANI, R. U. J. **Automata Theory, Languages, and Computation, 3rd ed., Pearson,**, 2014.

HOWARD, L. M.; D'ANGELO, D. J. The GA-P: a genetic algorithm and genetic programming hybrid. **IEEE Expert**, v. 10, p. 11–15, Jun 1995.

KEIJZER, M. Symbolic regression. In: **Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation**, 2008. (GECCO '08), p. 2895–2906.

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

KOZA, J. R. Darwinian invention and problem solving by means of genetic programming. In: **IEEE Intl. Conf. on Systems, Man, and Cybernetics,**, 1999. v. 3, p. 604–609 vol.3.

LANGDON, W. B.; POLI, R.; MCPHEE, N. F.; KOZA, J. R. Genetic programming: An introduction and tutorial, with a survey of techniques and applications. In: **Computational Intelligence: A Compendium**, 2008. p. 927–1028.

LEWIS, P. R.; PLATZNER, M.; RINNER, B.; TRRESEN, J.; YAO, X. **Self-aware Computing Systems: An Engineering Approach, Springer Publishing Company, Incorporated**. 1st. ed., 2016.

MANGASARIAN, O.; SETIONO, R.; WOLBERG, W. H. **Pattern Recognition Via Linear Programming: Theory And Application To Medical Diagnosis**. 1990.

MCKAY, R. I.; HOAI, N. X.; WHIGHAM, P. A.; SHAN, Y.; O'NEILL, M. Grammar-based genetic programming: a survey. **Genetic Programming and Evolvable Machines**, Kluwer Academic Publishers, v. 11, n. 3-4, p. 365–396, 2010.

MEIER, A.; GONTER, M.; KRUSE, R. Accelerating convergence in cartesian genetic programming by using a new genetic operator. In: **Proc. of the Annual Conf. on Genetic and Evolutionary Computation**, 2013. p. 981–988.

MELO, V. V. d.; FOWLER, B.; BANZHAF, W. Evaluating methods for constant optimization of symbolic regression benchmark problems. In: **Brazilian Conference on Intelligent Systems (BRACIS)**, 2015. p. 25–30.

MICHALEWICZ, Z. **Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.), Springer-Verlag**, 1996.

MILLER, J. F. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In: **Proceedings of the Annual Conference on Genetic and Evolutionary Computation - Volume 2**, 1999. p. 1135–1142.

MILLER, J. F.; THOMSON, P. Cartesian genetic programming. In: **Proceedings of the European Conference on Genetic Programming**, 2000. p. 121–132.

MILLER, J. F.; THOMSON, P.; FOGARTY, T.; NTRODUCTION, I. **Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study**. 1997.

MOTTA, F. A.; BERNARDINO, H. S.; BARBOSA, H. J. C.; FREITAS, J. M.; OLIVEIRA, I. L.; SOUZA, F. R. A hybrid approach of grammar-based genetic programming and differential evolution for symbolic regression. In: **Proc. of the Brazilian Congress on Computational Intelligence**, 2017.

MOTTA, F. A.; BERNARDINO, H. S.; BARBOSA, H. J. C.; FREITAS, J. M.; OLIVEIRA, I. L.; SOUZA, F. R. A hybrid grammar-based genetic programming for symbolic regression problems. In: **IEEE Congress on Evolutionary Computation (CEC), *In Press***, 2018.

MüLLER, S. **Grammatical theory: From transformational grammar to constraint-based approaches, Language Science Press**, 2016.

NICOLAU, M.; AGAPITOS, A.; O'NEILL, M.; BRABAZON, A. Guidelines for defining benchmark problems in genetic programming. In: **IEEE Congress on Evolutionary Computation (CEC)**, 2015. p. 1152–1159.

NORDIN, P. A compiling genetic programming system that directly manipulates the machine code. In: Kinnear, Jr., K. E. (Ed.). **Advances in Genetic Programming**, 1994. cap. 14, p. 311–331.

PAPPA, G. L.; TALBOT, H.; MENOTTI, D.; MEIGNAN, M. Towards automated lymphoma prognosis based on pet images. In: **IEEE Workshop on Machine Learning for Signal Processing**, 2008. p. 279–284.

PLAGIANAKOS, V.; TASOULIS, D.; VRAHATIS, M. A review of major application areas of differential evolution. In: **Advances in Differential Evolution**, 2008. p. 197–238.

POLI, R.; LANGDON, W. B.; MCPHEE NICHOLAS, L. E. **A Field Guide to Genetic Programming**, 2008.

RAGGETT, G. F. Numerical optimization of computer models, hans-paul schwefel, wiley, chichester, 1981. **Optimal Control Applications and Methods**, Wiley Subscription Services, Inc., A Wiley Company, v. 3, n. 1, p. 97–97, 1982.

ROY, P.; ISLAM, M. J.; ISLAM, M. M. Self-adaptive genetically programmed differential evolution. In: **Intl. Conf. on Electrical and Computer Engineering**, 2012. p. 639–642.

SCHWEFEL, H.-P. P. **Evolution and Optimum Seeking: The Sixth Generation, John Wiley & Sons, Inc.**, 1993.

SIGILLITO, V. G.; WING, S. P.; HUTTON, L. V.; BAKER, K. B. Classification of radar returns from the ionosphere using neural networks. v. 10, 07 1989.

STORN, R.; PRICE, K. **Differential Evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. International Computer Science Institute, Berkeley**, 1995.

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

SUG, H. The effect of training set size for the performance of neural networks of classification. **W. Trans. on Comp.**, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, v. 9, n. 11, p. 1297–1306, nov. 2010.

TOPCHY, A.; PUNCH, W. F. Faster genetic programming based on local gradient search of numeric leaf values. In: SPECTOR, L.; GOODMAN, E. D.; WU, A.; LANGDON, W. B.; VOIGT, H.-M.; GEN, M.; SEN, S.; DORIGO, M.; PEZESHK, S.; GARZON, M. H.; BURKE, E. (Ed.). **Proceedings of the Genetic and Evolutionary Computation Conference**, 2001. p. 155–162.

UMADEVI, S.; MARSELINE, K. S. J. A survey on data mining classification algorithms. In: **International Conference on Signal Processing and Communication (ICSPC)**, 2017. p. 264–268.

WATANACHATURAPORN, P. Identification of rice using symbolic regression. In: **8th International Conference on Information Technology and Electrical Engineering (ICITEE)**, 2016. p. 1–4.

WHIGHAM, P. A. Grammatically-based genetic programming. In: ROSCA, J. P. (Ed.). **Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications**, 1995. p. 33–41.

WONG, M. L.; LEUNG, K. S. Applying logic grammars to induce sub-functions in genetic programming. In: **Proceedings of IEEE International Conference on Evolutionary Computation**, 1995. v. 2, p. 737–740 vol.2.

WONG, M. L.; LEUNG, K. S. An induction system that learns programs in different programming languages using genetic programming and logic grammars. In: **Proceedings of IEEE International Conference on Tools with Artificial Intelligence**, 1995. p. 380–387.

WONG, M. L.; LEUNG, K. S. Evolutionary program induction directed by logic grammars. **Evolutionary Computation**, v. 5, n. 2, p. 143–180, 1997.