Trabalho Prático Computação Natural - Regressão Simbólica

Diego Matos [*1]

1. Introdução

A Programação Genética (GP) é uma técnica de otimização baseada em algoritmos genéticos que busca soluções para problemas complexos através da evolução de programas. Ela utiliza conceitos inspirados na biologia e na teoria da evolução, como reprodução, mutação, seleção natural e herança genética. A GP é aplicada em diversos domínios, incluindo aprendizado de máquina, otimização, programação automática, entre outros.

Neste trabalho, é apresentado uma implementação de um algoritmo de programação genética e a análise de seu desempenho em relação a diferentes parâmetros e técnicas, como geração de população, função de fitness, operadores de seleção e probabilidade de cruzamento e mutação. A implementação é baseada em árvores de expressão, que representam soluções candidatas para o problema a ser resolvido. Além disso, foram realizados experimentos para analisar o impacto das variações destes parâmetros na qualidade das soluções obtidas e na velocidade de convergência do algoritmo.

O objetivo deste trabalho é compreender os aspectos fundamentais da programação genética e explorar como as diferentes técnicas de seleção e alteração de parâmetros afetam seu desempenho.

2. Implementação

O código gerado define a estrutura básica de uma árvore de expressão e as operações necessárias para manipular e avaliar essas árvores. A classe Node é a unidade fundamental da árvore de expressão e pode representar variáveis, constantes ou operadores.

As variáveis e constantes são chamadas de ”terminais” e os outros operadores são chamados de ”não-terminais”. A implementação atual suporta terminais como ”X1”, ”X2”, ”X3”...”Xn” e ”const”, e operadores como ”+”, ”-”, ”*”, ”sin”, ”cos” e ”ln” (os operadores ”/” e ”exp” foram evitados durante a experimentação para evitar o comportamento destrutivo das árvores durante as operações de mutação e cruzamento).

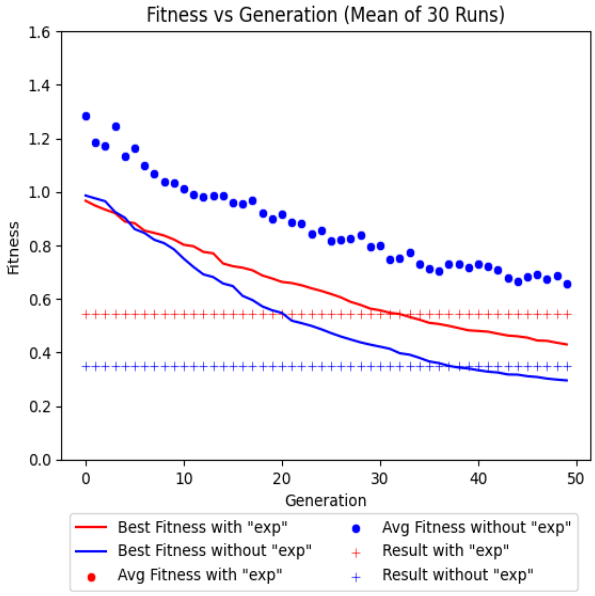

Figure 1. Variação da fitness média geral e do melhor indivíduo por gerações e os resultados ao utilizar ou não o operador ”exp” nos indivíduos aplicados no conjunto de dados 1.

É possível notar que o uso do operador ”exp” atrapalha a conversão do modelo quando aplicado ao dataset ”Synth 1”. Isso pode ser verificado observando a qualidade do resultado final aplicada no conjunto de teste e, principalmente, a média de fitness. A média de fitness sem o ”exp” é bem comportada, enquanto a fitness com ”exp” quase sempre gera pelo menos um indivíduo com um erro considerado ”infinito”, desta forma a fitness da geração tende ao infinito e atrapalha a coleta estatística. Isso é um sintoma do efeito destrutor da mutação e cruzamento, onde fazer uma pequena alteração de um operador qualquer para um operador ”exp” aumenta extremamente a fitness do indivíduo.

A classe Node fornece métodos para criar e manipular árvores de expressão, como ”evaluate”, ”copy”, ”subtree”, ”replace subtree”, ”count nodes”, ”get depth” e outros. Esses métodos permitem a criação e manipulação de árvores de expressão (indivíduos) durante a evolução, além de devolver informações relevantes a respeito dos indivíduos.

Foram implementados técnicas de geração de população amplamente conhecidas em programação genética, são elas,

a ”ramped half”, ”grow” e ”full”, que são utilizadas para gerar indivíduos iniciais com diferentes propriedades, variando a profundidade e quantidade de nós. Esses métodos são importantes na fase de inicialização de uma população de árvores de expressão e ajudam a garantir a diversidade na população.

A função de fitness utilizada foi a ”root mean squared error” (RMSE). A função recebe como entrada o indivíduo, o conjunto de terminais (X) e o conjunto dos resultados esperados (y). A função começa calculando uma normalização de y, subtraindo a média de y, elevando o resultado ao quadrado e somando todos os valores. Essa normalização é usada como um fator de escala para avaliar o erro de previsão do modelo em relação à variabilidade dos valores de saída. Em seguida, a função faz a previsão dos valores de saída para cada conjunto de entrada X, usando o método ”evaluate” do indivíduo e armazena a previsão em y pred. Então é calculado o erro quadrático real (real diff) entre os valores previstos e os valores reais para cada ponto de dados, armazenando-o em uma lista. Se algum dos erros é infinito, a função retorna infinito como fitness. Caso contrário, a função calcula a raiz quadrada do erro quadrático médio (RMSE) dividindo a soma dos erros quadráticos pela normalização e tirando a raiz quadrada. Essa medida é usada para avaliar o quão bem o indivíduo é capaz de prever os valores de saída em relação à variabilidade dos dados.

Também foram implementados operadores de seleção por torneio, por roleta e épsilon-lexicase. A seleção por roleta é uma abordagem que usa um mecanismo de seleção proporcional, em que a probabilidade de um indivíduo ser selecionado é proporcional à sua adequação relativa em relação aos outros indivíduos da população. Em outras palavras, indivíduos mais aptos têm uma probabilidade maior de serem selecionados para reprodução do que indivíduos menos aptos. A seleção por torneio é uma abordagem de seleção de indivíduos que envolve a realização de torneios em que vários indivíduos são selecionados aleatoriamente da população e comparados com base em sua adequação (fitness). O indivíduo mais adequado é selecionado para reprodução e seu genoma é copiado para a próxima geração. O operador de seleção épsilon-lexicase é uma abordagem mais complexa que envolve a seleção de um subconjunto de indivíduos da população com base em um conjunto de critérios de seleção hierárquicos. Esses critérios são definidos por valores limite (épsilon) que determinam quais indivíduos são selecionados em cada critério. O objetivo é selecionar indivíduos que sejam capazes de se adaptar a uma variedade de condições e ambientes, em vez de selecionar apenas os mais aptos em uma única medida de aptidão.

A função ”mutate” tem como objetivo aplicar uma mutação aleatória em uma árvore de expressão que representa um indivíduo. A mutação aleatória é uma das principais operações usadas em GP para criar novas soluções, e consiste em fazer pequenas alterações na árvore para criar uma nova solução. Essas alterações podem envolver a remoção ou adição de nós, a troca de operadores ou variáveis, entre outras possibilidades. A implementação define um tamanho máximo de sub-árvore para ser aleatoriamente inserida no indivíduo, e essa sub-árvore é construída a partir do método ramped-half and half, que gera árvores com tamanhos variados entre 1 e 7 níveis, com uma distribuição uniforme de tamanhos. Como é requerido que cada indivíduo tenha no máximo 7 níveis de profundidade, é verificado se a profundidade do indivíduo gerado é menor ou igual a 7. Se for, o indivíduo gerado é adicionado na nova população.

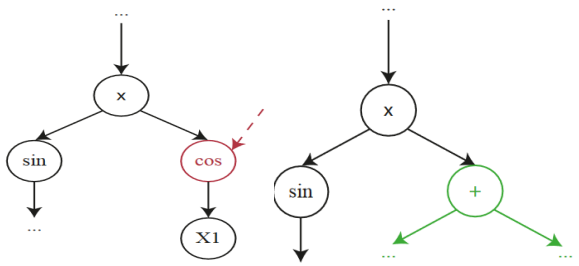

Figure 2. Representação da mutação: seleção aleatória de um nó do indivíduo, seguida da substituição do nó por uma sub-arvore aleatória.

Os cruzamentos seguiram a seguinte estratégia: Dado dois indivíduos pais, iremos selecionar aleatoriamente um nós de mesma profundidade entre os dois indivíduos. Ou seja, caso o só selecionado seja de profundidade 7, então significa que os dois pais devem ter tamanho 7 e apenas um nó folha será trocado; caso seja selecionado um nó de profundidade 6, então uma sub-arvore de no máximo tamanho 2 será selecionada. Essa estratégia garante que o tamanho dos filhos gerados não ultrapasse o tamanho máximo definido.

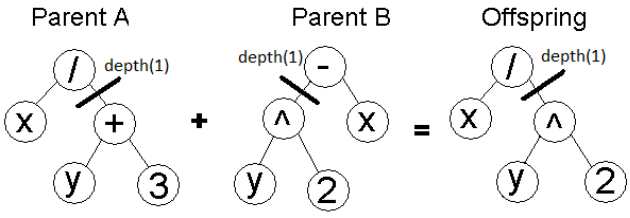

Figure 3. Representação do cruzamento: sub-arvore do pai 1 é escolhido com profundidade 1, portanto o a segunda sub-arvore também deve ter a mesma profundidade.

Após a execução do algoritmo são armazenadas as estatísticas por geração sobre: a quantidade de indivíduos iguais, melhor fitness, melhor indivíduo, pior indivíduo,

média da fitness, total de cruzamento e total de mutações.

3. Experimentação

Nesta seção serão mostrados experimentos e suas repectivas análises. Cada experimento foi executado 30 vezes com 30 diferentes sementes aleatórias, onde cada gráfico mostra os valores médios das 30 execuções. Os parâmetros padrões cada experimento foram:

Table 1. Parâmetros

Parâmetro	Valor
Taxa de mutação	0.6
Taxa de crossover	0.3
Tamanho da população	100
Gerações	50
Elitismo	True
Tamanho do torneio	2

Como os gráficos foram calculados com a média de 30 execuções do algoritmo, eles aparentam ser mais bem comportados do que a realidade de uma única execução.

Ao variar a probabilidade de cruzamento é possível controlar a frequência com que os indivíduos são cruzados durante a geração da nova população. Isso pode afetar a diversidade da população e também a rapidez com que a população converge para soluções boas.

Quando a probabilidade de cruzamento é alta, há uma maior probabilidade de que os indivíduos sejam cruzados, o que pode ajudar a aumentar a diversidade genética na população, promovendo a recombinação dos genes presentes nos indivíduos. Por outro lado, quando a probabilidade de cruzamento é baixa, há uma menor probabilidade de que os indivíduos sejam cruzados, o que pode reduzir a diversidade da população, e também pode levar a uma convergência mais lenta para máximos locais.

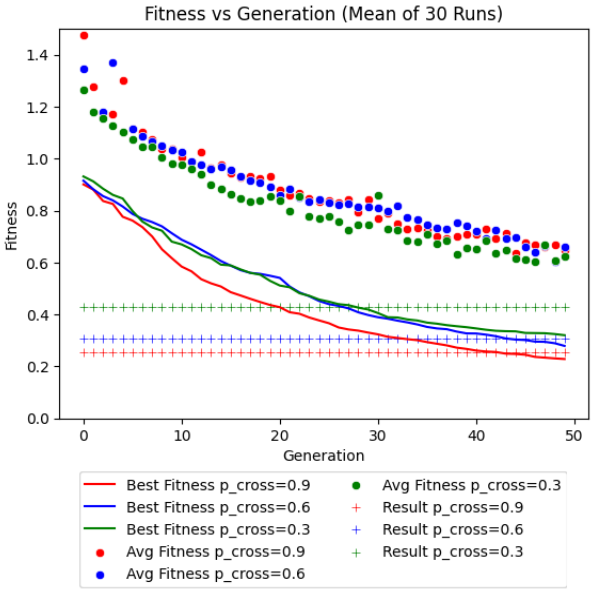

Figure 4. Variação da média da fitness e do melhor indivíduo e o resultado do indivíduo no conjunto de teste ao alterar a probabilidade de cruzamento no conjunto de dados 1.

Observando a fitness média das execuções com baixa probabilidade de cruzamento, pode-se notar que a população tende a convergir para um mínimo local rapidamente, reduzindo a diversidade genética e a capacidade de explorar outras soluções potenciais. Embora a fitness média possa parecer promissora, quando é testado no conjunto de testes e avaliado a média dos melhores indivíduos, percebe-se que o melhor indivíduo não é tão bom, o que indica que a população não foi capaz de encontrar soluções realmente boas ou diversificadas. Isso significa que a maior parte desses indivíduos deve ser semelhante, indicando uma convergência prematura para um mínimo local.

Por outro lado, quando a probabilidade de cruzamento é alta, a diversidade dos indivíduos no algoritmo genético é maior. Isso sugere que há uma boa diversidade de indivíduos na população, espalhados pelo espaço de busca de forma a permitir que alguns indivíduos ruins elevem a média da fitness e alguns indivíduos bons a abaixem. No entanto, é importante saber que a alta diversidade também pode tornar a convergência para um ótimo mais difícil, já que pode ser mais difícil explorar soluções específicas com tanta variabilidade na população (embora neste caso uma taxa de cruzamento alta conseguiu gerar bons indivíduos).

Assim como na probabilidade de cruzamento, a escolha da probabilidade de mutação ideal em um algoritmo genético pode afetar significativamente seu desempenho. Quando a probabilidade de mutação é muito baixa, pode ocorrer uma convergência prematura para mínimos locais e a ocorrência de overfitting, pois a população não será capaz de explorar

completamente o espaço de solução em busca de soluções de alta qualidade e se concentrará em máximos locais. No entanto, valores muito altos podem prejudicar a convergência e dificultar a identificação de soluções de alta qualidade.

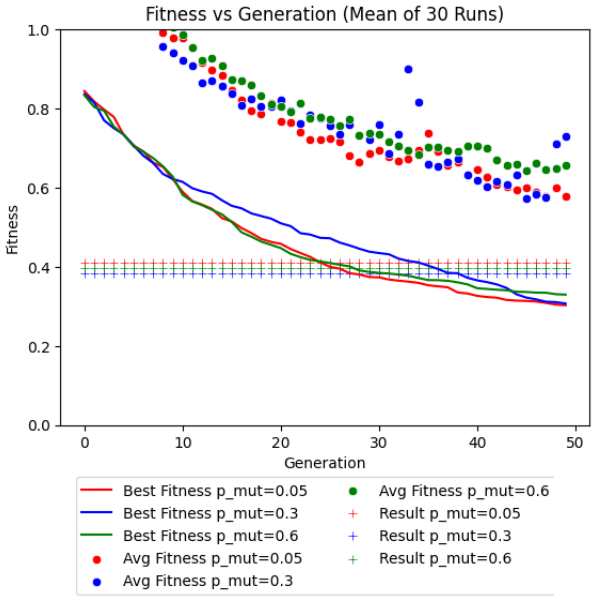

Figure 5. Variação da fitness média e do melhor indivíduo por gerações e resultados ao alterar a probabilidade de mutação dos indivíduos aplicado no conjunto de dados 1.

Nos resultados obtidos é possível notar que assim como no cruzamento a média de fitness é baixa no caso onde há pouca mutação (p mut=0.05). Isso deve ocorrer pois todos os indivíduos são semelhantes entre a população, desta forma a busca se concentra em mínimos locais. Outra evidência disso é o quanto o melhor indivíduo neste caso foi bem no treinamento mas não generalizou para os casos de teste. Isso indica um sintoma de overfitting, onde provavelmente a solução ficou complexa demais num mínimo local e não conseguiu bons resultados para novos dados.

Em oposição, nós temos o caso onde a probabilidade de mutação é muito alta (p mut=0.6). Analisando a fitness média é possível perceber que temos piores resultados, pois existe maior diversidade na população. Desta forma existem tanto indivíduos ruins que jogam a media para cima, quantos bons que jogam a média para baixo. O problema neste caso é que o algoritmo fica muito imprevisível na busca e dificilmente foca seus indivíduos vales que minimizam a fitness. Porém neste caso ele ainda obteve melhores resultados que o caso anterior. Embora a fitness média tenha ficado mais alta, os melhores indivíduos provavelmente apresentaram funções mais simples e mais fáceis de generalizar para o conjunto de teste.

Elitismo é uma técnica em algoritmos genéticos que consiste em manter os melhores indivíduos da população atual inalterados na próxima geração. Isso significa que, independentemente do processo de seleção, alguns indivíduos são protegidos de serem substituídos por outros que podem ter uma menor aptidão.

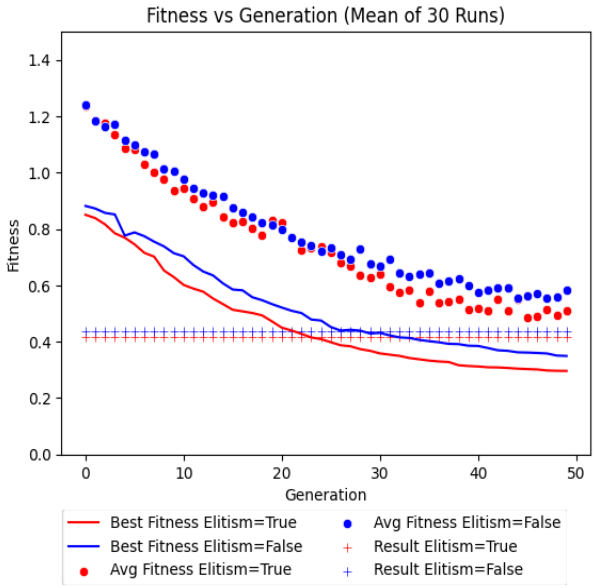

Figure 6. Variação da fitness média por gerações com e sem elitismo aplicados no conjunto de dados 1.

É importante lembrar que o gráfico representa a média de 30 execuções. Portanto ele é muito bem comportado em comparação ao gráfico obtido por uma única execução. Observando a média dos melhores indivíduos é possível notar que com o elitismo = False, não existe garantia de que a próxima população será tão boa quanto a população atual. Ou seja, não há garantia que o melhor indivíduo da ultima geração será o melhor indivíduo já existente entre todas as gerações. Isso fica evidente ao observar no início do gráfico (por volta da geração 4) que há uma melhora seguida de uma piora considerável na média dos melhores indivíduos das 30 runs.

Logo, é esperado que a execução com elitismo = True tenha resultados melhores e garantidos em relação ao elitismo = Falso.

Apesar de uma população maior necessitar de maior poder computacional ela pode ajudar a encontrar melhores soluções mais rapidamente, pois há uma maior probabilidade de encontrar bons indivíduos aleatoriamente. Além disso, um conjunto maior de soluções candidatas pode aumentar a eficácia das operações de crossover e mutação, aprimorando a qualidade dos indivíduos gerados nas gerações subsequentes. Por outro lado, ao utilizar uma população pequena, podemos saturar o modelo rapidamente, já que

os indivíduos tendem a convergir para um mínimo local e acabam se tornando muito semelhantes uns aos outros

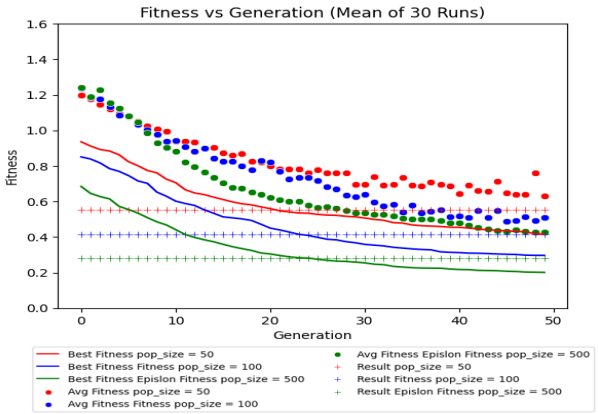

Figure 7. Variação da fitness média por gerações a partir do tamanho da população de indivíduos aplicados no conjunto de dados 1.

Com o uso de uma população muito pequena, o algoritmo converge rapidamente para uma solução local, enquanto com o uso de uma população maior, uma solução melhor é encontrada aleatoriamente, o que deve contribuir para uma convergência mais eficiente em gerações futuras.

Como esperado, o operador seletivo que define a probabilidade de seleção com base na aptidão (fitness) obtida e utiliza essas estatísticas globais teve o pior desempenho. É comum que a seleção por roleta acabe selecionando previamente uma maioria de indivíduos bons, gerando muita pressão seletiva logo no início da execução. Isso leva o algoritmo a convergir precocemente para um mínimo local.

Por outro lado, a seleção por torneio tem um controle maior da pressão seletiva por meio do parâmetro K, que define a quantidade de indivíduos que serão selecionados aleatoriamente para entrar no torneio. Com isso, é possível afirmar que a diversidade utilizando o método de seleção por torneio tende a ser maior quanto menor o valor de K.

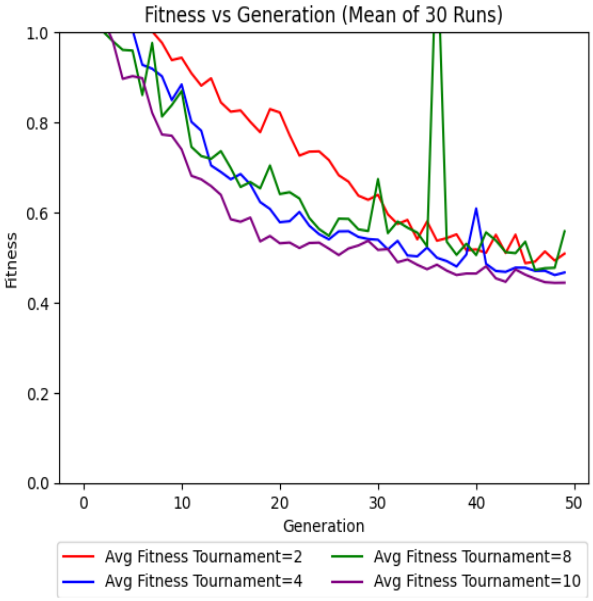

Figure 9. Variação da fitness média por geração ao alterar o valor de K.

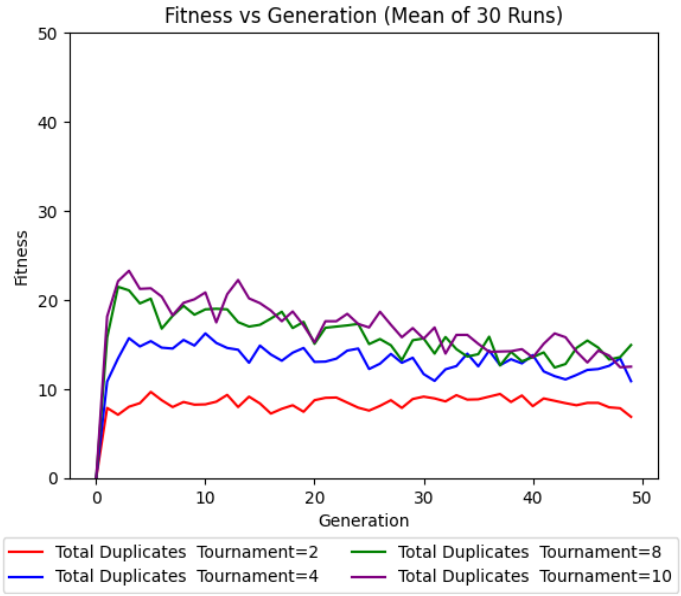

Figure 8. Variação da quantidade média de indivíduos iguais por geração ao alterar o valor de K.

É possível notar que o valor de K define o equilíbrio entre pressão seletiva e diversidade. Com um valor de K muito elevado, teremos uma pressão seletiva muito grande e, consequentemente, mais indivíduos iguais serão selecionados (como no caso em que k=10). Isso se reflete, também, na fitness média da população: Quando a maioria dos indivíduos

são iguais ou muito parecidos eles possuirão uma média parecida. Ou seja, onde há mais diversidade a fitness média oscila mais, onde há menos a fitness média fica bem comportada dentro do minimo local. Isso justifica o caso k=10 ter uma fitness mais bem comportada que os outros casos.

É importante ressaltar que no modelo implementado, se os dois indivíduos selecionados não passarem por cruzamento e/ou por mutações eles serão diretamente inseridos na nova população sem alterações. Portanto a probabilidade de mutação e cruzamento, nesta implementação, também será relacionada a quantidade de indivíduos iguais.

A seleção lexicase é uma técnica de seleção de pais em algoritmos genéticos que visa preservar a diversidade na população e evitar a convergência prematura. Essa abordagem considera a performance dos indivíduos em múltiplos casos de teste, em vez de apenas um valor de aptidão agregado. Dessa forma, a seleção lexicase enfatiza a importância de resolver bem todos os casos de teste e promove a evolução de soluções especializadas para diferentes partes do problema.

Com isso é possível selecionar indivíduos que se saem muito bem em alguns casos de teste mesmo que em outros eles sejam piores. Isso ajuda a preservar diversidade e ainda mantém indivíduos ”uteis” distribuídos no espaço de busca. Com o uso do épsilon para definir o intervalo de erro é possível manter intervalos que evoluem junto com a população de forma a ficar cada vez mais ”apertada” e gerar indivíduos dentro de faixas de erro cada vez menores.

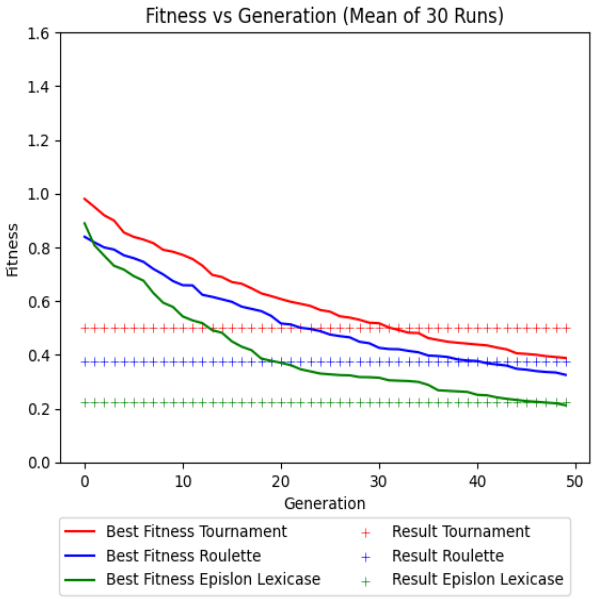

Figure 10. Melhor indivíduo por geração em cada tipo de seleção usada aplicado no conjunto de dados 1 (elitismo = True).

Dentre os métodos de seleção utilizados o que performou melhor foi a épsilon-lexicase, isso se deve a todas as suas características citadas e principalmente ao fato deste mecanismo manter a diversidade sem manter indivíduos muito ruins na população.

4. Resultados

Os modelos propostos foram utilizados em outras bases de dados e seus resultados foram:

Table 2. Resultados

Dataset	Teste Fitness
Synth 1	0.0414
Synth 2	0.5381
Concrete	1.4949

Como esperado, o algoritmo apresentou ótimos resultados no dataset 1, que decreve uma função simples. No dataset ”Synth 2” e ”Concrete” os resultados não foram tão bons. Isso pode se dever a alguns fatores, os dados descritos por esses dois conjunto de dados são consideravelmente mais complexos, principalmente o conjunto ”Concrete”. Desta forma, já é esperado uma convergência mais lenta do algoritmo.

Outro motivo que colabora para que os resultados não tenham sido bons Foi a decisão de não utilizar os operadores ”Exp” e ”/”. Funções complexas necessitam de operadores complexos, desta forma, ao remover os operadores ”exp” e ”/” para evitar os efeitos destrutores das árvores podemos estar limitando diversas operações necessárias para descrever as funções ”Concrete” e ”synth 2”. Portanto os indivíduos gerados durante a evolução podem tender a serem mais complexos para tentar imitar o efeito destes operadores sem nunca conseguir descrever a função esperada.

5. Conclusão

O desenvolvimento de algoritmos genéticos para resolver problemas complexos envolve uma série de considerações e ajustes de parâmetros, como a probabilidade de cruzamento, probabilidade de mutação, tamanho da população e o método de seleção utilizado. Os resultados experimentais apresentados nesta análise demonstraram que encontrar um equilíbrio adequado entre exploração e explotação é crucial para garantir a convergência e a obtenção de soluções de alta qualidade.

A seleção épsilon-lexicase mostrou-se a melhor opção dentre os métodos de seleção analisados, devido à sua capacidade de preservar a diversidade na população e evitar a convergência prematura. O método leva em consideração a performance dos indivíduos em múltiplos casos de teste e promove a evolução de soluções especializadas para difer-

entes partes do problema.

A maior dificuldade encontrada foi implementar os operadores de mutação e cruzamento. Durante a etapa de implementação foram testados inúmeras formas de implementar o cruzamento e a mutação. Foi utilizando cruzamento com elitismo, cruzamento clássico, cruzamento penalizando infinitamente filhos gerados com profundidade maior que 7, cruzamento que gera apenas indivíduos de tamanho 7 ou menos e várias variações da mutação com a mesma ideia. Percebi também que o esses operadores acabavam gerando indivíduos muito diferentes ao trocar um único nó da ávore (muito comum se trocar o operador para um exp, a fitness tende a ir para números extremamente altos). Muito provável que o uso de árvore acaba gerando uma dificuldade de localidade dos indivíduos, o que dificulta a convergência do algoritmo. Para tentar solucionar este problema os operadores "exp" e "/" foram retirados, embora eles sejam importantes para descrever certos tipos de dados.

References

Foram utilizadas as aulas e slides das aulas como referência.