TP0 - Programação Genética para o problema de Regressão Simbólica

Ramon Gonçalves Gonze

03 de outubro de 2017

Resumo

O objetivo deste trabalho é apresentar um algoritmo evolucionário que utiliza Programação Genética (GP) para resolver o problema de regressão simbólica. Para isto, conceitos aprendidos em sala de aula como operações sobre indivíduos, técnicas para aumento de diversidade e avaliação de parâmetros foram utilizados no desenvolvimento do algoritmo. São demonstrados os resultados obtidos com diferentes combinações de parâmetros e o motivo da utilização de cada técnica.

1 Introdução

O problema de regressão simbólica consiste em, dada uma função f desconhecida e um conjunto de dados que representa essa função - pontos da curva dessa função - o intuito é encontrar uma expressão simbólica para esta função que melhor se adequa à base de dados fornecida, como na Figura 1.

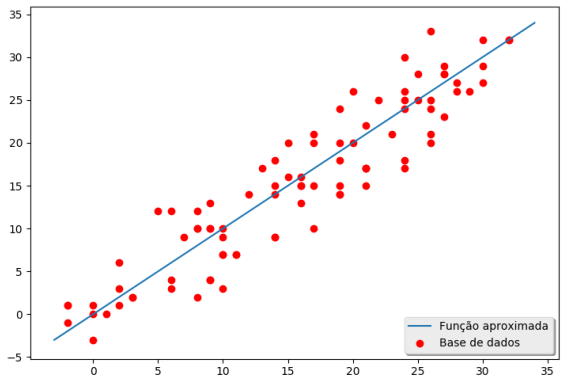

Figura 1: Para os pontos vermelhos (a base de dados), é construída uma função (linha azul) que busca aproximar o comportamento desses dados.

O primeiro passo para utilizar GP para a resolução do problema, é fazer a representação. Os indivíduos, que serão expressões simbólicas, foram representados como árvores binárias, sendo cada nó um terminal ou uma função. Seguindo o conceito de Algoritmos Evolucionários, cada indivíduo possui uma *fitness*, que represenda a qualidade de uma solução do problema. Para o problema de regressão simbólica, a fitness foi calculada utilizando a raiz quadrada do erro quadrático médio (RMSE):

$$f(i) = \sqrt{\frac{1}{n}\sum_{x=1}^{n}(evaluate(i,x) - y)^2}$$

onde i é o indivíduo avaliado, n é a quantidade de exemplos na base de dados fornecida, $evaluate(i,x)$ é a função que retorna o resultado da equação do indivíduo i na entrada x, e y é a saída correta para a entrada x.

Foram utilizadas as operações de *cruzamento*, *mutação* e *reprodução*. Para aumentar a diversidade da população, foi útilizado o *fitness sharing*, um método de niching, discutido na **Seção 2.6**. Para a seleção de indivíduos, foi utilizado o método de seleção por Torneio. Na **Seção 4** são apresentados os resultados dos experimentos e a análise destes, que demonstram o resultado da variação dos parâmetros do algoritmo.

2 Modelagem

Aqui serão descritas as estruturas de dados, técnicas e os métodos utilizados para modelar o problema, assim como suas funcionalidades para o funcionamento do algoritmo.

2.1 Indivíduo

Como já dito, os indivíduos são representados como ávores binárias. Através de um caminhamento pós-ordem na árvore, é construída uma lista de elementos

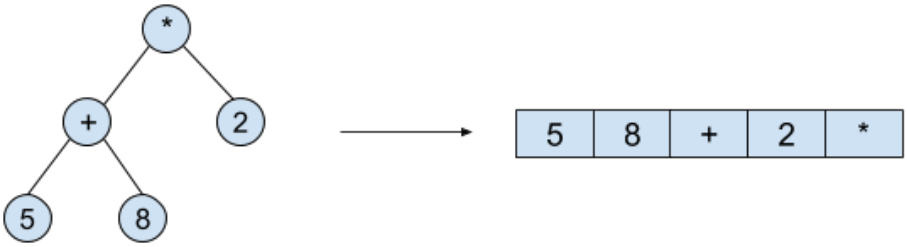

Figura 2: Árvore, que representa um indivíduo.

na qual pode ser feito o cálculo da equação, através de uma pilha. A equação apresentada está escrita em Notação Polonesa Inversa[2].

Cada indivíduo representado por uma árvore representa uma possível solução para o problema. Foi estabelecida uma altura máxima de 7 níveis para uma árvore (sendo a raiz o nível 1). Os métodos utilizados para a construção de indivíduos são o *grow* e *full*[1], comuns em Algoritmos genéticos e Programação Genética.

2.2 População

A população é um conjunto de indivíduos na qual busca evoluir a cada geração. Para essa evolução ser possível, foram aplicados os seguintes métodos e conceitos:

- **Inicialização**: O método utilizado para gerar a população inicial é o denominado *Ramped half-and-half*. Ele consiste em utilizar em conjunto os métodos *grow* e *full* para gerar indivíduos. Seja n o tamanho da população e max o limite de altura de um indivíduo. Serão gerados $\frac{n}{max-1}$ indivíduos para cada altura máxima, entre $2, 3, \ldots, max$. Por exemplo, para $max = 5$ e $n = 32$, serão gerados $32/4 = 8$ indivíduos de nível 2, 8 indivíduos de nível 3, e assim sucessivamente. E para aumentar ainda mais a diversidade, entre estes 8 indivíduos de cada nível, metade será gerado pelo método *grow* e a outra metade pelo método *full*.

- **Elitismo**: O melhor indivíduo da população é passado diretamente sem nenhuma alteração para a geração seguinte, para assim manter crescente a qualidade do melhor indivíduo das populações entre as gerações. Foi testado utilizar o elitismo passando-se x indivíduos para a nova população ($x \in [1, 10]$), porém, não houve melhoramento da solução com essa alteração.

2.3 Conjuntos de terminais e funções

O conjunto de terminais utilizado consiste no conjunto de variáveis da base de dados utilizada na execução juntamente com o conjunto dos números reais definido no intervalo $[-1, 1]$. Logo

$$terminal\_set = \{x, y, z, \ldots\} \cup num$$

onde *num*, quando escolhido para ser inserido no nó de uma árvore, é um número real no intervalo $[-1, 1]$ (escolhido aleatoriamente).

Por outro lado o conjunto de funções variou entre as bases de dados (vide as bases de dados utilizadas na **Seção 3**). O conjunto base de funções

$$function\_set = \{+, -, *, /\}$$

foi utilizado em todas as bases de dados. Para as bases em que se tem uma noção do comportamento da função desconhecida (como é o caso da base de dados **keijzer-7-train**), foi acrescentado a função de logaritmo natural (ln).

2.4 Operações

Eis as três operações realizadas com os indivíduos: *reprodução*, *cruzamento*, e *mutação*.

2.4.1 Reprodução

A operação de reprodução é feita sob um indivíduo apenas. O que ela faz é simplesmente passar o indivíduo para a próxima geração sem nenhuma alteração (semelhante ao elitismo visto na **Seção 2.2**).

2.4.2 Cruzamento

A operação de cruzamento é feita entre dois indivíduos. O objetivo é haver troca de *genes* entre ambos. Como os indivíduos são árvores binárias, a troca de duas sub-árvores - estas devem necessariamente pertencer à **região em comum** entre os dois indivíduos, expicada logo abaixo - representa um cruzamento. São necessárias duas etapas para a execução da operação:

1. Determinar a **região em comum** dos dois indivíduos.
2. Escolher um nó aleatório que pertenca a **região em comum**.

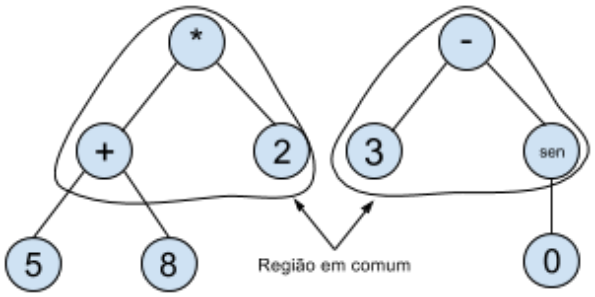

Figura 3: As regiões em destaques são a região em comum entre os dois indivíduos.

A região em comum entre dois indivíduos é uma parte das duas árvores que possui a mesma forma, como na Figura 3. Determinada a região em comum, um nó dessa região é escolhido aleatoriamente, e então as duas sub-árvores correspondentes ao nó escolhido são trocadas.

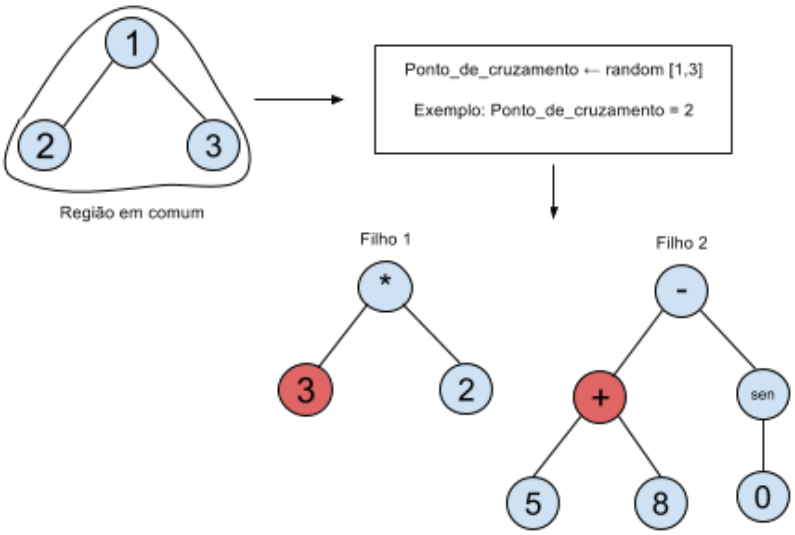

Figura 4: Exemplo de cruzamento entre os indivíduos da Figura 3 no nó 2 da região em comum.

Este tipo de cruzamento é classificado como *homólogo*, por preservar a posição dos genes trocados. Como é escolhido somente um nó para ser feito a troca, o cruzamento recebe o nome de *cruzamento de um ponto*.

2.4.3 Mutação

A operação de mutação é feita somente sob um indivíduo. Foram implementadas três tipos de mutação:

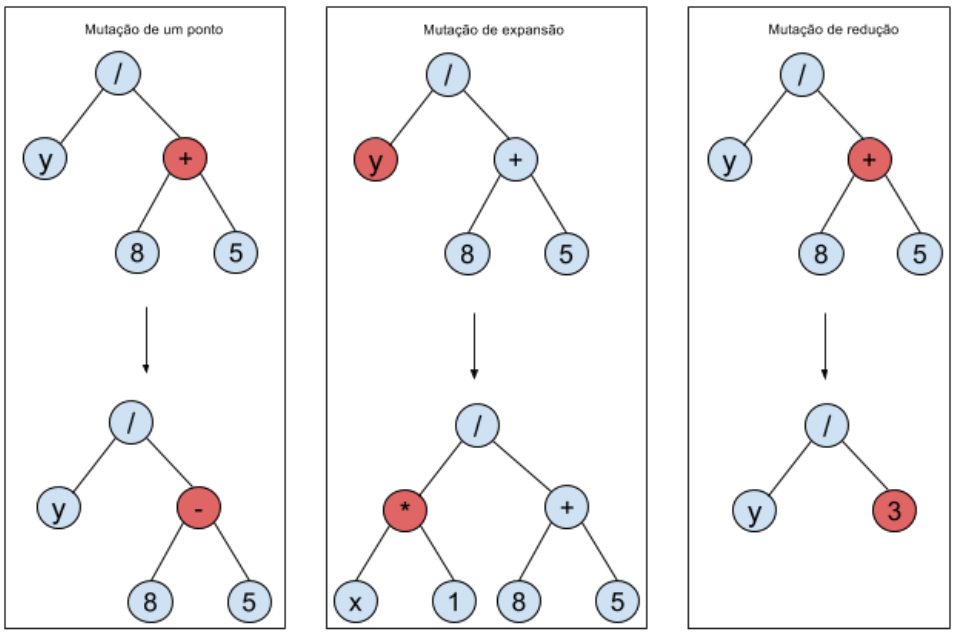

Figura 5: Demonstração dos três tipos de mutação. Os nós com a cor vermelha são os pontos selecionados para a mutação

- **Mutação de um ponto**: Neste tipo de mutação, o nó selecionado para sofrer a mutação é trocado por um do mesmo conjunto, ou seja, terminal trocado por terminal e função trocada por função. No caso das funções, há ainda uma divisão, entre funções sob dois terminais $(+, -, *, /, \ldots)$ e funções sob um terminal $(seno, log, \ldots)$. Cada tipo de função é trocada por outra do mesmo tipo.

- **Mutação de expansão**: O objetivo desta mutação é aumentar o tamanho do indivíduo, fazendo a troca de um nó terminal por uma outra sub-árvore aleatória gerada por *grow* ou *full*.
- **Mutação de redução**: A mutação de redução faz o oposto da mutação de expansão. Um nó que contém uma função, juntamente com toda a sua sub-árvore abaixo, é trocado por um terminal aleatório. É utilizado para reduzir o tamanho de indivíduos muito grandes.

2.5 Seleção

O método de seleção utilizado é o de torneio, como já mencionado. Para uma população de n indivíduos, são selecionados k. Dentre os k indivíduos, o melhor deles - que possui o menor valor - "ganha o torneio", e é escolhido para efetuar operações de cruzamento, mutação ou reprodução com outros indivíduos escolhidos também por outros torneios. Um ponto a se observar é que o indivíduo ganha o torneio pelo valor da sua *fitness_ sharing*, a função f' explicada na próxima seção. Logo, são necessários $n-1$ (1 indivíduo é escolhido pelo elitismo) torneios para que seja formada uma nova população na qual irá evoluir. A variação de k é feita para controlar a *pressão seletiva* do algoritmo. Quanto menor é o valor de k, menor é a velocidade de convergência da população, e quanto maior o valor, maior a velocidade de convergência.

2.6 Fitness Sharing

Este método parte do pressuposto de que indivíduos em um mesmo local tem de compartilhar o alimento disponível. Indivíduos semelhantes a muitos outros da população terão uma probabilidade menor de serem escolhidos na seleção. Isto faz com que o espaço de busca seja melhor explorado, através da seleção de indivíduos que se parecem pouco com o restante da população. Para o problema da regressão simbólica, o objetivo é minimzar o erro, portanto, indivíduos com uma fitness menor são classificados como melhores. Na prática, o fitness sharing é uma função f' que é calculada a partir da fitness original do indivíduo. Ela é definida por

$$f'(i) = \begin{cases} f(i) * m_i & \text{se} \quad m_i > 0 \\ f(i) & c.c. \end{cases}$$

onde m_i calcula aproximadamente o número de indivíduos na qual a função $f(i)$ é compartilhada. O valor de m_i se dá pela soma da função *sharing* de todos os indivíduos da população:

$$m_i = \sum_{j=1}^{n} sharing(d_{ij})$$

A quantidade de indivíduos na população é representada por n e *sharing* é a função que calcula a similaridade entre dois indivíduos. Sua definição se segue abaixo:

$$sharing(d_{ij}) = \begin{cases} 1 - (\frac{d_{ij}}{\sigma}) & \text{se} \quad d_{ij} < \sigma \\ 0 & c.c. \end{cases}$$

Calcular esta função baseada na similaridade do fenótipo de dois indivíduos traz resultados melhores do que se basear na similaridade do genótipo (DEB, K; GOLDBERG E. D, 1987). O fenótipo de um indivíduo i é a sua fitness, ou seja, $f(i)$. A distância d_{ij} entre indivíduos i e j pode ser então determinada pelo valor absoluto de $f(i) - f(j)$. Para se calcular a similaridade entre dois indivíduos, é preciso estabelcer um limiar que determina o quanto dois indivíduos são semelhantes. Para cada base de dados esse valor é diferente. O valor deste limiar é representado por σ.

3 Base de Dados e Execução

Foram utilizadas três bases de dados para a realização dos experimentos, sendo cada uma composta por dois arquivos, um de *treino* e um de *teste*.

Cada um dos arquivos possui uma quantidade fixa de variáveis que formam (ou não, no caso de uma variável ser irrelevante para a solução) a função desconhecida e uma quantidade de pontos dessa função. Chamemos os pontos da função de *exemplos*. Cada exemplo (linha) do arquivo

Arquivo	Variáveis	Exemplos
keijzer-7-train	x	100
keijzer-7-test	x	100
keijzer-10-train	x_1, x_2	100
keijzer-10-test	x_1, x_2	100
house-train	$x_1, x_2, \ldots, x_8$	5000
house-test	$x_1, x_2, \ldots, x_8$	15000

Tabela 1: Quantidade de variáveis e exemplos por base de dados.

contém a(s) entrada(s) da(s) variável(eis) e a saída correta para essa(s) entrada(s). A Tabela 1 contém os dados sobre todas as bases utilizadas.

O trabalho foi implementado em Python na versão 3.6.1. A função principal do arquivo está no arquivo **main.py** (**Seção 3.2**), que recebe como parâmetro o nome do arquivo que contém a base de dados. O arquivo deve estar em formato csv, com a seguinte configuração:

$$x_1, x_2, \ldots, x_n, y$$
$$x_1, x_2, \ldots, x_n, y$$
$$\ldots$$

Sendo $x_1, x_2, \ldots, x_n$ as entradas da função, y a saída correta e cada linha um exemplo. Para executar o programa para a base de dados **keijzer-7-train** por exemplo, deve ser executado:

python3 diretorio/keijzer-7-train.csv

A alteração de parâmetros como probabilidade de cruzamento, tamanho da população, quantidade de gerações, tamanho do torneio, entre outros, deve ser feita no arquivo **constants.py**.

A saída do algoritmo é feita pela saída padrão (**stdin**), e ela contém o valor da fitness do melhor indivíduo encontrado e a sua expressão simbólica, em notação polonesa inversa.

Obs.: As variáveis de entrada da base de dados $x_1, x_2, \ldots, x_n$ estão representadas na equação, respectivamente como '1', '2', ..., 'n'.

3.1 Conjuntos

Para as bases de dados **keijzer-7** e **keijzer-10**, como já é sabido que o comportamento das funções se assemelham à curva de uma função logarítmica, o conjunto de funções utilizado foi $\{+, -, *, /, ln\}$, e o conjunto de terminais para **keijzer-7** $\{x, num\}$ e para **keijzer-10** $\{x_1, x_2, num\}$.

Já para a base **house**, o conjunto de funções utilizado foi $\{+, -, *, /\}$ e o de terminais $\{x_1, x_2, \ldots, x_8, num\}$.

3.2 Arquivos

Os arquivos que compõe este trabalho, seguidos de suas funções, são:

- **fitness.py**: Possui as funções responsáveis por calcular a fitness de um indivíduo ou de uma população inteira, baseado no RMSE. Também se encontram as funções que calculam a função f' dos indivíduos, relativa ao fitness sharing.
- **population.py**: Possui as funções responsáveis por construir os indivíduos: os métodos *grow*, *full* e *Ramped half-and-half*.
- **selection.py**: Funções responsáveis pela seleção por torneio.
- **operations.py**: Neste arquivo se encontram as funções que realizam as operações de *cruzamento* e *mutação* com os indivíduos.
- **constants.py**: Aqui se encontram todos os parâmetros do algoritmo evolucionário como tamanho da população, quantidade de gerações, o valor de sigma (para o fitness sharing), base de dados, elitismo, conjunto de funções, entre outros.

- **main.py**: O arquivo que contém a função principal do algoritmo, que é responsável por fazer chamadas de todos os outros métodos citados acima. O Algoritmo 1 representa o funcionamento da função:

Algorithm 1

```
1: procedure DoEvolution
2:     population ← RampedHalfAndHalf(population_size)
3:     for i := 1 to number_of_generations do
4:         fitness(population)
5:         fitnessSharing(population)
6:         new_population ← bestIndividual(population)
7:         tournamentSelection(new_population)
8:         operations(new_population) //Crossover, mutation and reproduction
9:         population ← new_population
10:    return bestIndividual(population)
```

4 Experimentos

Para os experimentos abaixo, considere p_c a probabilidade de cruzamento, p_m a probabilidade de mutação e p_r a probabilidade de reprodução, sendo $p_c + p_m + p_r = 1$. Para o tamanho do torneio, considere k o número de indivíduos escolhidos para competir no torneio.

Como explicado na **Seção 2.6**, σ é o limiar que determina o quanto dois indivíduos são semelhantes. Os melhores valores encontrados para que o conceito de niching funcionasse, e assim a diversidade da população aumentasse, foram $\sigma = 0.5$ para as bases **keijzer-7** e **keijzer-10**. Já para a base **house**, o melhor valor foi $\sigma = 15,000$.

Todos os experimentos que serão mostrados a seguir foram executados 30 vezes e extraída a média.

4.1 Experimento 1 - População & Gerações

O primeiro experimento feito, utilizando as três bases, teve o objetivo de verificar qual seria o tamanho da população e o número de gerações mais apropriados para conseguir um melhor resultado.

Arquivo	Tamanho da população	Número de gerações	Melhor	Média	Pior
keijzer-7-train	50	50	0.158	0.298	1.426
		100	0.003	9.504	318.089
		500	0	9.446	466.981
	100	50	0	1.961	183.183
		100	0	2.183	195.731
		500	0	0.232	4.103
keijzer-10-train	50	50	0.158	0.298	1.426
		100	0.119	0.231	1.845
		500	0.088	0.199	2.812
	100	50	0.116	0.334	4.212
		100	0.095	0.311	9.301
		500	0.082	0.232	4.103
house-train	50	50	552,622	$103,601 \times 10$	$189,680 \times 10^2$
		100	522,164	$239,834 \times 10$	$894,348 \times 10^2$
		500	256,544	$118,436 \times 10^{10}$	$592,168 \times 10^{27}$
	100	50	530,076	711,095	$341,087 \times 10$
		100	289,110	$270,485 \times 10^6$	$253,869 \times 10^9$
		500	273,612	$440,563 \times 10^2$	$435,034 \times 10^4$

Tabela 2: Quantidade de variáveis e exemplos por base de dados. Melhor, Média e Pior são relativos aos indivíduos da população. O melhor desempenho, em todas as bases de dados, foi uma população com **100** indivíduos e **500** gerações.

É intuitivo pensar que quanto maior o número de gerações, melhor será o resultado obtido. Entretanto, para uma população de 100 indivíduos, através de 500 gerações, o algoritmo gasta cerca de 3 horas aproximadamente (na base de dados **house-train**) para executar. Números acima destes deixariam inviável o tempo de execução.

4.2 Experimento 2 - Probabilidade das operações

Aqui foram fixados o tamanho da população e o número de gerações como **100** e **500** respectivamente. Neste segundo experimento, foram testados diferentes combinações de p_c, p_m. O valor de p_r permaneceu fixo para todos os experimentos, com uma probabilidade de 0.05.

Arquivo	p_c	p_m	Melhor	Média	Pior
keijzer-7-train	0.65	0.35	0	24.401	1,727.015
	0.40	0.55	0	42.419	2,264.902
	0.20	0.75	0	213.699	18,765.452
keijzer-10-train	0.65	0.35	0.065	2.245	187.756
	0.40	0.55	0.058	12.105	1,094.821
	0.20	0.75	0.020	33.126	3,145.181
house-train	0.65	0.35	277,148	854,906 $\times 10^8$	854,905 $\times 10^{10}$
	0.40	0.55	270,763	417,166 $\times 10^6$	417,019 $\times 10^8$
	0.20	0.75	259,321	$206,197 \times 10^7$	$196,484 \times 10^{10}$

Tabela 3: Quantidade de variáveis e exemplos por base de dados. Melhor, Média e Pior são relativos aos indivíduos da população.

A partir da Tabela 3 pode-se observar que a melhor solução em todos os experimentos é superior à combinação de $p_c = 0.9$ e $p_m = 0.05$. Outro ponto a se observar é o valor da média da fitness da população e do pior indivíduo. Estes valores são mais distantes da fitness do melhor indivíduo quando comparados aos valores de quando $p_c = 0.9$ e $p_m = 0.05$. Quanto maior a probabilidade de mutação, mais aleatória fica a população.

4.3 Experimento 3 - Torneio

Para este experimento, foram mantidos fixos os seguintes valores: $p_c = 0.2$, $p_m = 0.75$, $p_r = 0.05$, tamanho da população = 100 e número de gerações = 500. Alterando o tamanho do torneio, foram obtidos os seguintes resultados:

Arquivo	k	Melhor	Média	Pior
keijzer-7-train	2	0	24.401	547.882
	5	0	126.254	3,968.476
	10	0	213.699	18,765.452
keijzer-10-train	2	0.086	33.126	3,145.181
	5	0.057	1.325	7.441
	10	0.021	0.484	1.532
house-train	2	259,321	$206,197 \times 10^7$	$196,484 \times 10^{10}$
	5	250,391	$795,948 \times 10^{10}$	$795,948 \times 10^{27}$
	10	247,802	$144,531 \times 10^8$	$144,530 \times 10^{10}$

Tabela 4: Melhor, Média e Pior são relativos aos indivíduos da população.

Pode-se observar com os dados da Tabela 4 que os melhores resultados foram obtidos para $k = 10$.

4.4 Experimento 4 - Elitismo

Considere os mesmo parâmetros do **Experimento 3** e $k = 10$.

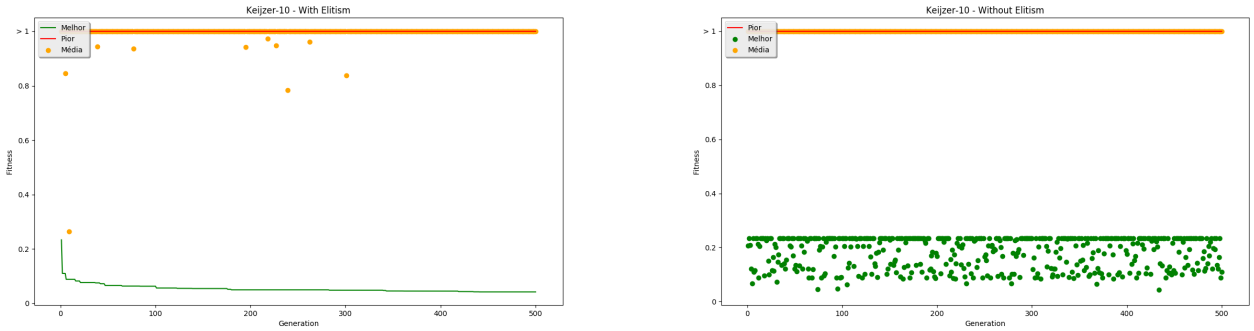

Figura 6: Experimento utlizando a base de dados **keijzer-10-train**. Pode-se observar claramente a fitness do melhor indivíduo com e sem o elitismo.

Na Figura 7 podemos observar a variância que a fitness do melhor indivíduo da população possui. Com o elitismo, a fitness é crescente. Além da variância, no experimento acima, foi obtida uma melhor solução com o elitismo:

- Com elitismo: 0.0371
- Sem elitismo: 0.0439

4.5 Resultados

A partir de várias execuções do algoritmo, os melhores resultados encontrados para cada uma das bases foi (com aproximação de 6 casas decimas):

4.5.1 Keijzer-7

Equação: x,ln
Fitness do arquivo de treino: 0
Fitness do arquivo de teste: 0.1

4.5.2 Keijzer-10

Equação: $x_2, x_1, /, x$,$x_1, +, 0.428, \times, x_1, ln, \times, -0.325, \times, 0.006, x_2, +, ln, x_2, x_2, x_1, \times, \times, ln, /, /, -$
Fitness do arquivo de treino: 0.005440
Fitness do arquivo de teste: 0.005649

4.5.3 House

Equação:
$x_2, x_2, \times, x_6, x_8, x_6, -, -, x_6, x_8, -, x_8, x_6, -, -, +, \times, x_3, +, x_6, x_6, x_5, x_1, /, x_5, \times, +, x_3, x_2, /, x_3, x_3, x_8, +, +, +, \times, \times, +$
Fitness do arquivo de treino: 241,503
Fitness do arquivo de teste: 233,503.761811

5 Conclusão

A partir dos experimentos realizados e as análises sobre as mudanças de parâmetros, pode-se concluir que:

- Para uma população pequena e um número de gerações também pequenos, tem-se o melhor, a média e o pior indivíuo bem próximos. A medida em que se aumenta o tamanho da população e a quantidade de gerações, obtem-se um resultado melhor (a fitness do melhor indivíduo é mais baixa), entretando a média e o pior indivíduo se distanciam muito do melhor, ou seja, a população se torna mais diversa. Isto pode ser obervado na Tabela 2.

- Apesar de não ter sido apresentado os valores neste documento, o tempo de execução do algoritmo é consideravelmente maior quando o valor de probabilidade de cruzamento é maior. O tempo de execução para, por exemplo $p_c = 0.2$ e $p_m = 0.75$, foi menor comparado à execução quando $p_c = 0.9$ e $p_m = 0.05$.
- Um conhecimento prévio de qual o comportamento da função desconhecida (semelhança com funções matemáticas conhecidas), como o caso de **keijzer-7** e **keijzer-10** se assemelhando com ln, faz com que o algoritmo seja mais eficiente.
- Na **Seção 4.5.3** pode-se concluir que os componentes x_4 e x_7 da base **house** não são muito influentes na curva que representa a função desconhecida da base de dados.

Referências

[1] DEB, K; GOLDBERG E. D. **An investigation of niche and species formation in genetic function optimization**, in *Proc. 3rd Int. Conf. Genetic Algorithms*,J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 42-50.

[2] WIKIPEDIA. **Notação Polonesa Inversa**. Disponível em: <https://pt.wikipedia.org/wiki/Nota%C3%A7%C3%A3o_polonesa_inversa>. Acesso em: 03. Out. 2017.

[3] SARENI, B; KRÄHENBÜHL L. **Fitness sharing and niching methods revisited**. IEEE Transactions on Evolutionary Computation, Institute of Electrical and Electronics Engineers, 1998, 2 (3), pp.97 - 106.