Trabalho de Algorítmos Genéticos
Parte 1

Fábio Beranizo Fontes Lopes
Escola de Artes, Ciências e Humanidades (EACH)
Universidade de São Paulo (USP)
Email: f.lopes@usp.br

I. Introdução

Este relatório apresenta a discussão da parte 1 do trabalho de algorítmos genéticos (GA) para a disciplina de Inteligência Computacional. Foi avaliada a utilização de GA para otimização de um conjunto funções. O foco se manteve em analisar o comportamento do algorítmo mediante a utilização de diferentes técnicas, operadores e parametrizações.
O trabalho foi desenvolvido utilizando a linguagem de programação Python, especificamente sua distribuição Anaconda.

II. Otimização de Funções

O problema da otimização de funções consiste em maximizar ou minimizar uma função matemática através da escolha de valores para as variáveis, respeitando-se o domínio da função. Para o trabalho desenvolvido foram selecionadas 4 funções para análise:

$$max.f_0(x) = 20 + \sum_{i=1}^{2} x_i^2 - 10\cos(2\pi x_i)$$

intervalo para otimização: [-5, 5]

$$min.f_1(x) = \sum_{i=1}^{30} x_i^2$$

intervalo para otimização: [-100, 100]

$$min.f_2(x) = \sum_{i=1}^{30} |x_i + 0.5|^2$$

intervalo para otimização: [-100, 100]

$$min.f_3(x) = \sum_{i=1}^{30} -x_i \sin(\sqrt{(}\,|x_i|))$$

intervalo para otimização: [-500, 500]

Para facilitar a implementação as funções para minimização foram transformadas no oposto e maximizadas.

III. Cromossomo

Para a representação de uma solução do problema de otimização de funções foi necessário utilizar uma estrutura que armazenasse o valor das variáveis da solução. A escolha natural foi adotar vetores como estrutura de dados. Essa decisão facilitou a implementação dos operadores de crossover e de mutação, além de tornar simples o cálculo das funções fitness. Cada variável da solução podia assumir um valor real dentro de um domínio. Para dispor as variáveis dentro do vetor foram realizadas duas abordagens, cromossomo de valores reais e cromossomo de valores binários.

A. Cromossomo Real

No cromossomo real cada elemento do vetor assume um valor *float* que representa o valor da variável. O genótipo do indivíduo é do tipo:

$$[float_{var_1}, float_{var_2}, ..., float_{var_n}]$$

B. Cromossomo Binário

No cromossomo binário cada elemento do vetor representa um bit de uma das variáveis da solução. Cada variável ocupa 32 bits contínuos, ou seja, 32 posições do vetor de um indivíduo. O genótipo do indivíduo é do tipo:

$$[bit1_{var_1}, bit2_{var_1}, ..., bit32_{var_n}]$$

Na seção de discussão dos resultados serão avaliadas as vantagens e desvantagens de cada abordagem de cromossomo.

IV. Seleção de Indivíduos

Somente um operador de seleção de indivíduos foi analisado neste trabalho, o método da roleta. Ele operador foi escolhido por ser de fácil implementação e por apresentar bons resultados segundo a literatura. No entanto, durante o desenvolvimento foi possível perceber um problema: este operador aumenta consideravelmente o tempo de execução do algorítmo para grandes populações, uma vez que todos os indivíduos participam do sorteio.

A. Método da Roleta

O método da roleta leva em consideração o fitness de um indivíduo em relação ao fitness total da população. Nesse processo cada indivíduo possui uma probabilidade de ser selecionado (uma fatia da roleta) proporcional ao seu fitness. Embora não garanta que as melhores soluções sobrevivam, este método dá maiores probabilidades de sobrevida às soluções com bom fitness.

V. Operadores Genéticos

Os indivíduos selecionados passam por um dos 3 tipos de operadores genéticos: reprodução, crossover e mutação. Neste trabalho foram analisados dois tipos de crossover e dois tipos de mutação, que são descritos abaixo:

A. Crossover

O operador crossover permite a criação de novos indivíduos a partir de organismos pais. Os descendentes possuem genótipo de ambos pais, embora geralmente sejam diferentes. O crossover é uma forma de ampliar o espaço de busca por soluções, porém mantendo-se material genético de boa qualidade de gerações anteriores.

1) Crossover de um Ponto: O crossover de um ponto seleciona aleatoriamente um ponto de crossover nos cromossomos pais e os recombina gerando dois filhos. Este operador favorece a herança de trechos contínuos de cromossomo, o que pode ser bom quando estes trechos guardam algum significado para o fitness.

2) Crossover Uniforme: O crossover uniforme define cada gene de um novo indivíduo copiando-se o valor de um de seus pais com uma dada probabilidade. Essa abordagem, ao contrário do crossover de um ponto, quebra trechos contínuos de cromossomo e podendo levar a uma maior diversidade do espaço de busca.

B. Mutação

O operador de mutação permite inserir diversidade genética entre gerações de indivíduos trocando o valor de genes no cromossomo. Com maior variedade genética o espaço de busca é ampliado, possibilitando sair de situações de mínimo local. Entretanto, os experimentos realizados indicaram que este operador deve ser usado com parcimônia, com baixa probabilidade de ocorrer. Nas parametrizações em que sua probabilidade foi alta a busca não convergiu para bons valores, mantendo quase um comportamento de busca aleatória.

1) Mutação Simples: A operação de mutação básica consiste em sortear uma posição do cromossomo e substituí-la por um valor aleatório do domínio da solução. Essa operação deve manter a solução válida, evitando-se processamento para verificação do cromossomo.

2) Mutação por Permutação: Outro operador avaliado foi a mutação por permutação. Neste tipo de mutação, dois genes do cromossomo são sorteados e trocam de posição entre si. Este tipo de mutação não causou impacto no cromossomo real, uma vez que nas funções avaliadas as variáveis tinham o mesmo peso. No entanto, para o cromossomo binário essa técnica é interessante pois promove mudança significativa dos genes e é executada mais rapidamente que a mutação simples.

VI. Elitismo

Em algorítmos genéticos o elitismo é a operação de copiar uma dada proporção dos melhores indivíduos para a próxima geração. Esta técnica garante que boas soluções permaneçam na população, evitando-se que o algorítmo deixe de explorar a vizinhança de espaços de busca promissores. Na solução desenvolvida elitismo foi implementado conservando 10% dos indivíduos com melhor fitness da geração anterior.

VII. Critério de Parada

A evolução do algorítmo genético prossegue até que um critério de parada seja atingido. Neste trabalho foram comparadas 2 alternativas para término da evolução: número de gerações e tempo de execução. Ambas técnicas não são sofisticadas, nem são ótimas do ponto de vista de resultados, no entanto permitem realizar uma análise mais adequada dos operadores. Comparando-se diferentes parametrizações mediante um mesmo tempo de execução, ou um determinado número de gerações é possível avaliar de forma justa as vantagens e desvantagens de cada escolha.

A. Número de Gerações

Este critério de parada de faz o com o algorítmo prossiga até que um determinado número de gerações seja atingido. Dessa forma é possível comparar quais parametrizações convergem mais rapidamente, além de avaliar o quanto são propensas a ficar presas em mínimos locais.

B. Tempo de Execução

No segundo critério avaliado é dado um tempo limite para a execução do algorítmo. Este critério é interessante para avaliar quais técnicas são mais custosas do ponto de vista de processamento, e quais se sairiam melhor em situação de restrição do tempo disponível. Nos testes realizados foi fixado 1 segundo para evolução em cada parametrização.

VIII. Grid Search

O resultado da execução do algorítmo genético está ligado diretamente aos parâmetros utilizados. Tais parâmetros podem ser selecionados eficientemente através de uma busca de parametrizações geradas a partir de um *grid*. Este processo é denominado *Grid Search*.
Para o algorítmo genético desenvolvido, os parâmetros otimizados foram: tamanho da população de indivíduos

Fig. 1: Resultados do Grid Search para f0(x) e 100 gerações

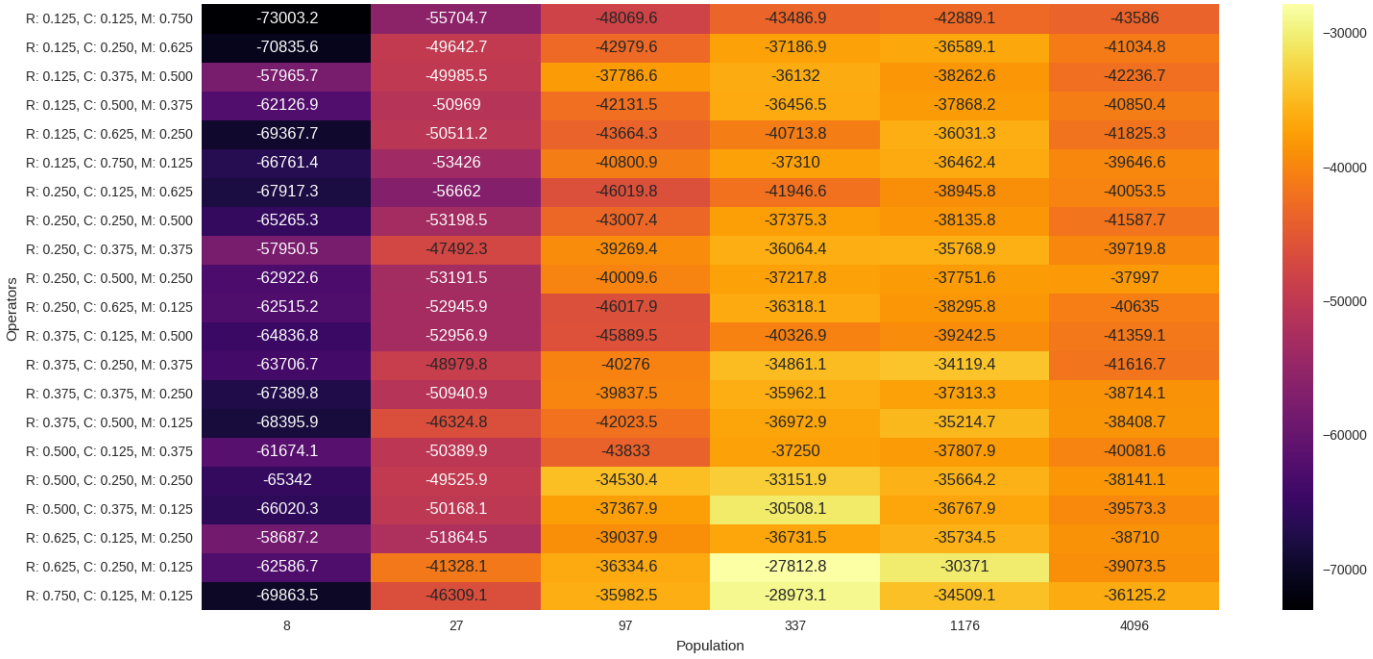


Fig. 2: Resultados do Grid Search para f2(x) e 1 segundo de execução

e probabilidades dos operadores de reprodução, crossover e mutação.

A. Parâmetros Variados

Visando avaliar o comportamento do algorítmo com populações de tamanhos bem distintos, foram testadas 6

possíveis variações de população: de 8 a 4096 indivíduos dispostos em escala logarítimica.
As probabilidades dos operadores foram variadas de 0.125 a 0.875 em espaços de 0.125.
Arbitrariamente foi definido que a escolha da melhor parametrização seria feita utilizando uma configuração base de operadores. Após definida a melhor escolha de parâmetros, os mesmos foram aplicados para outros tipos de operadores, técnicas e cromossomo, e uma análise comparativa foi realizada.

Configuração base de operadores:

- Cromossomo real
- Crossover de um ponto
- Mutação simples
- Não há elitismo

Para que os resultados do do *Grid Search* fossem mais confiáveis, cada parametrização foi executada 10 vezes. A média dos melhores fitness obtidos nas 10 execuções foram então dispostas nos gráficos abaixo para avaliação:

Observando-se as figuras 1 e 2 é possível elaborar uma hipótese que há relação direta entre o tamanho da população e a convergência do algorítmo para melhores valores. De fato, em todos os testes executados, as parametrizações com maior população alcançaram melhores resultados. A explicação para este fenômeno é que em populações maiores a diversidade genética é maior e consequentemente o espaço de busca também. Mesmo no caso da figura 2 em que o tempo de execução para todas parametrizações é o mesmo, a população se mostrou mais determinante para obter bons resultados que o tempo de execução do algorítmo.
Há no entanto uma ressalva: embora populações maiores contribuam para bons resultados, o tempo de execução do algorítimo é fortemente impactado por este grande número de indivíduos. Dependendo da aplicação isso deve ser levado em consideração.

Para as comparações foram fixados os seguintes parâmetros:

- Taxa de reprodução = 0.375
- Taxa de crossover = 0.500
- Taxa de mutação = 0.125
- Tamanho da população = 27

O tamanho da população foi propositalmente mantido baixo para que o resultado pudesse convergir mais lentamente. As probabilidade de cada operador foi definida após análise das figuras 1 e 2, tomando valores que levaram a melhores resultados na maioria das populações avaliadas.

IX. Operadores e Estratégias

Dada a parametrização definida anteriormente, foram avaliados a seguir diferentes operadores e estratégias genéticas.

A. Elitismo

A Figura 3 mostra uma comparação das evoluções com e sem elitismo. Analisando os resultados é possível perceber que a solução com elitismo evita a perda da melhor solução, que ocorre na versão sem elitismo. Além disso, a aplicação desta técnica permitiu que a solução convergisse para melhores valores mais rapidamente.

A tabela 1 mostra valores de fitness obtidos em execuções do algorítmo com e sem elitismo:

	Fit max s/ Elitismo	Fit max c/ Elitismo
f0(x)	-11.6314	-0.8034
f1(x)	-52811.7721	-24321.2968
f2(x)	-64560.2019	-21427.4450
f3(x)	2958.9352	5007.2206

Tabela 1 - Valores de fitness para evolução c/ e s/ elitismo

B. Operadores de Crossover

A comparação dos operadores de crossover mostrou que para a maioria das funções o crossover de um ponto obteve melhores resultados. No entanto para f2 a diferença foi pouco significativa. A tabela 2 mostra que para as funções f1, f2 e f3 o crossover de um ponto obteve fitness melhores e para f0 o crossover uniforme foi superior.
De fato, para o cromossomo real nenhum dos operadores escolhidos possui alguma vantagem uma vez que não há esquemas, nem sequências especiamente importantes.

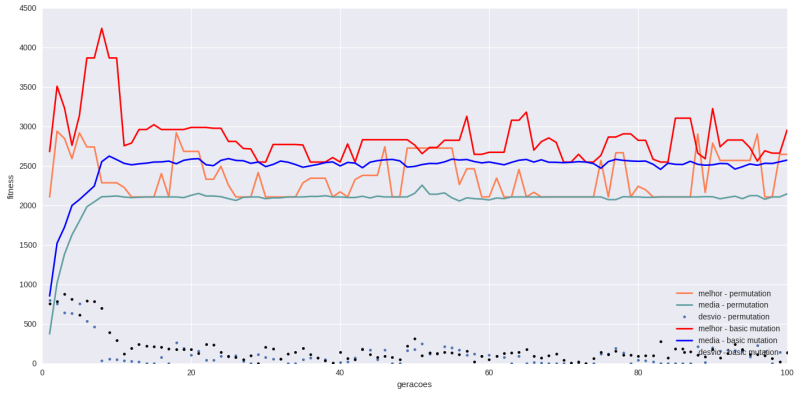
	Fit max C. um ponto	Fit max C. uniforme
f0(x)	-11.6314	-2.2871
f1(x)	-52811.7721	-66625.7284
f2(x)	-64560.2019	-65527.9226
f3(x)	2958.9352	1204.0767

Tabela 2 - Valores de fitness para evolução c/ crossover de um ponto e uniforme

C. Operadores de Mutação

A Tabela 3 mostra a comparação das evoluções dos operadores de mutação básica e por permutação. Observando-se os resultados, a mutação simples levou a melhores fitness nas execuções. Conforme discutido anteriormente, este resultado se deve ao fato da mutação por permutação não alterar o resultado da função fitness para cromossomos reais.
A figura 4 mostra o comportamento das evoluções de f3(x) para permutação e mutação simples. É possível perceber que a mutação simples diversificou mais o espaço de busca e provocou mais saltos no fitness. No entanto, por não estar aliada a uma técnica de elitismo, os bons resultados alcançados acabaram perdidos com a evolução.

(a) Fig. 3: Evolução do algorítimo genético para f0(x) c/ s s/elitismo



(b) Fig. 4: Evolução do algorítimo genético para f3(x) c/ mutação simples e permutação

	Fit max M. simples	Fit max Permutação
f0(x)	-11.6314	-9.1838
f1(x)	-52811.7721	-56952.5563
f2(x)	-64560.2019	-53526.0925
f3(x)	2958.9352	2648.5310

Tabela 3 - Valores de fitness para evolução c/ mutação simples e permutação

D. Cromossomo Binário

O cromossomo binário foi implementado, porém os resultados não foram totalmente analisados devido a problemas nos testes executados. No entanto, é possível supor que sua característica poderia levar a convergência exata para o ótimo nas funções f0, f1 e f2, cujo máximo global é zero. Além disso, o uso do operador de permutação se tornaria mais relevante, inserindo variedade genética para a função avaliada. O operador de crossover uniforme por sua vez se comportaria como quase que uma mutação, uma vez que modificaria significantemente a composição dos valores das variáveis.

References

[1] S. Peres, Buscas - Inteligência Artificial, Apresentação realizada durante a disciplina SIN5002. 10 mar 2016.

[2] S. Russell and P. Norvig, Artificial intelligence. Englewood Cliffs, N.J.: Prentice Hall, 1995.