Trabalho de Algoritmos Genéticos - Somatória

Grupo: Bruno Borges e Rafael Morais de Assis (UFU 2018-1)

Resumo

Esse documento faz um estudo de uma possível solução para uma instância do *subset sum problem* baseada em algoritmos genéticos.

Palavras-chaves: Algoritmos genéticos, subset sum problem

1. Introdução

O problema da soma dos n números consiste em encontrar uma soma de n termos adequada a um resultado r previamente fornecido. O problema é uma instância de um problema bastante conhecido em Ciência da Computação chamado *subset sum problem*, pertencente a classe de problemas NP-Completo.

2. Apresentação do problema.

O problema consiste em procurar uma sequência de n números, onde $\sum_{i=1}^{n} xi$ = r, onde r e n são fornecidos pelo usuário e $xi \neq xj$ para todo i, j tal que $i \neq j$.

3. Algoritmos genéticos

Algoritmos genéticos são algoritmos de aprendizado e otimização que possuem como princípio a ideia de seleção natural e hereditariedade. Eles diferem de algoritmos comuns por apresentar, principalmente, os seguintes aspectos:

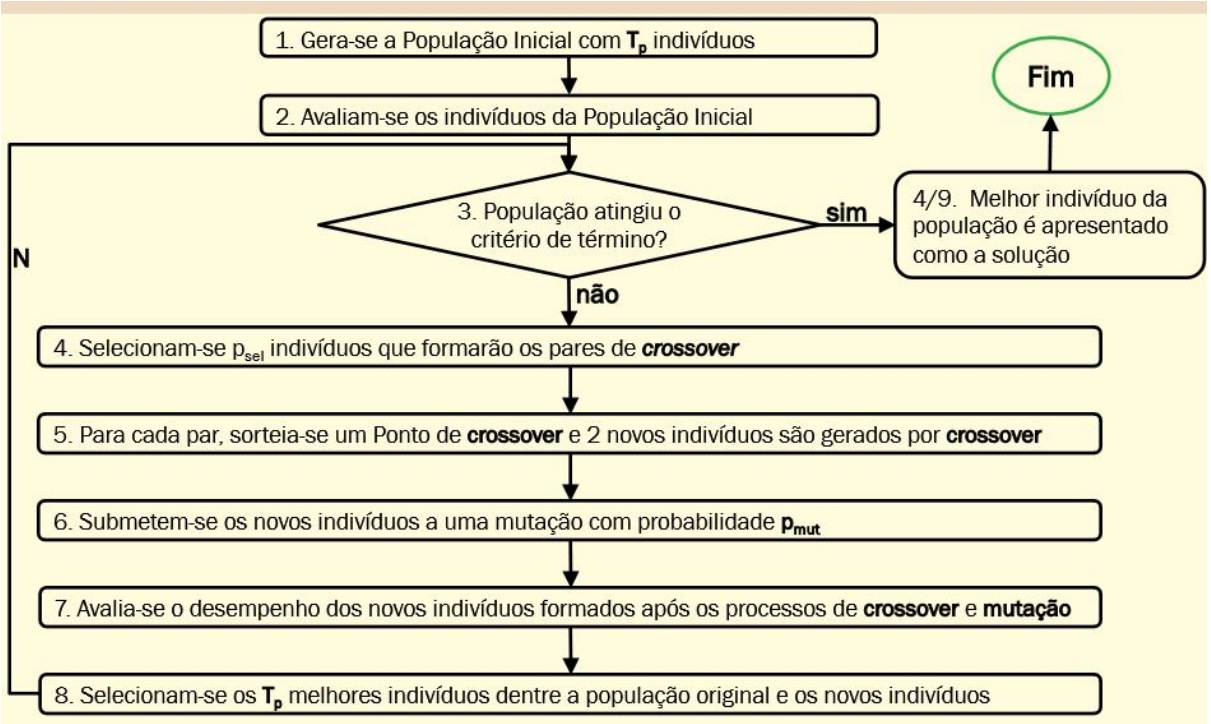
- Não há necessidade de conhecimento prévio do problema, apenas uma forma de avaliação.
- Há vários possíveis resultados para um mesmo problema.
- A evolução não é baseada em regras, mas em probabilidade.

A estratégia de busca dos algoritmos genéticos é baseada em “aptidão” ou seja, os indivíduos que se destacam em uma população possuem mais chance de sobreviverem, um exemplo clássico é o tamanho do pescoço Girafas, girafas com pescoço grandes conseguiam se alimentar mais, então sobreviviam para gerar mais filhos, logo as girafas de pescoço pequeno não conseguiam se alimentar e não sobrevivam para gerar filhos. Nesse exemplo a aptidão é o tamanho do pescoço da girafa.

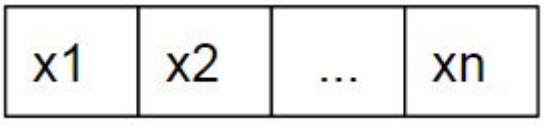
O ponto de partida dos algoritmos genéticos é como estruturar um cromossomo para trabalhar adequadamente com o problema, onde comumente, o cromossomo é estruturado por um vetor finito sobre um conjunto de termos(ou alfabeto) finito.

Em contraste com a seleção natural, os cromossomos se reproduzem, em tal reprodução dois pais são selecionados e então se aplica operações como o cruzamento(crossover) e mutação para gerar novos indivíduos semelhantes aos pais.

O fluxograma que mostra o funcionamento do algoritmo é detalhado a seguir.



4. Configuração do cromossomo.



O cromossomo é representado por um array de n posições, onde cada gene xi é um termo do somatório $\sum_{i=1}^{n} xi$.

5. População Inicial.

A população inicial é composta de 100 indivíduos que são possíveis candidatos a resolução do problema, gerados aleatoriamente no intervalo de $[0,\ r-1]$, onde r é o valor desejado da somatória, respeitando a restrição de não igualdade entre os n genes.

6. Avaliação da População

O fitness é a distância entre o valor desejado a somatória dos xi termos.

$$FITNESS = \left| r - \sum_{i=1}^{n} xi \right|, \; onde\, r = valor\, desejado$$

O melhor indivíduo é aquela que apresentar menor distância e o melhor o que tem *fitness* 0. Foi escolhido dessa forma pois representa melhor o quão distante está o valor da soma do esperado, e considera quando a somatória ultrapassa e quando é menor.

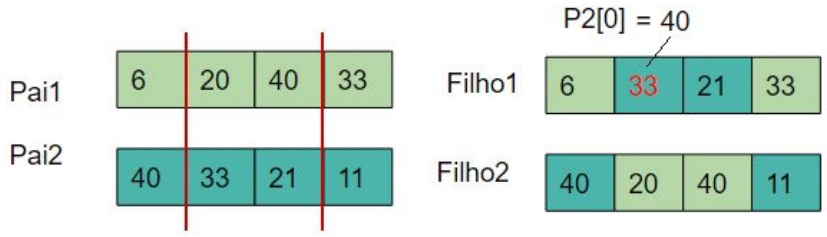
7. Seleção

A operação de seleção visa escolher o indivíduo a fazer o *crossover*. É construída de forma a representar a seleção natural, é feita de forma aleatória mas com a escolher indivíduos mais aptos (melhor *fitness*).

Foi escolhido o método de torneio, onde escolhe aleatoriamente 20 indivíduos dos 100 da população para fazer parte de um grupo, o melhor deles é então escolhido.

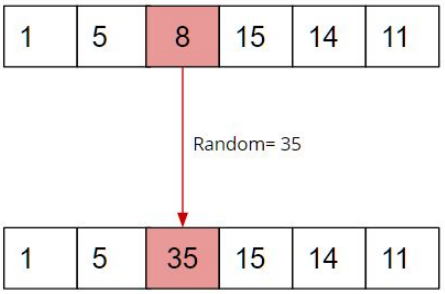
8. Crossover

Para geração de filhos, Utilizado o método crossover múltiplo com dois pais escolhidos pela seleção. É selecionado uma faixa de genes onde é feito a troca de genes, se existir elementos repetidos é feito troca do elemento por outro do outro pai, até que não tenham elementos repetidos.



9. Mutação

Na mutação é gerado um número randômico entre 0 e o valor desejado - 1, então verifica-se a restrição de desigualdade entre os genes, caso se repetir, gera-se outro valor corretivo até a restrição ser atendida, então o valor aleatório gerado substituirá o gene marcado para mutação.



10. Reinserção

É feito a reinserção pura: todos os filhos gerados substituem a população anterior. São gerados 100 filhos mantendo assim o tamanho da população original.

11. Condição de término.

Foi utilizado como condição de término o número de 500 gerações no caso do fitness não convergir para o valor 0. Quando chegar à um indivíduo de *fitness = 0* o programa pará e fornece a resposta, se não é dado o melhor indivíduo de todas as gerações executadas.

12. Parâmetros.

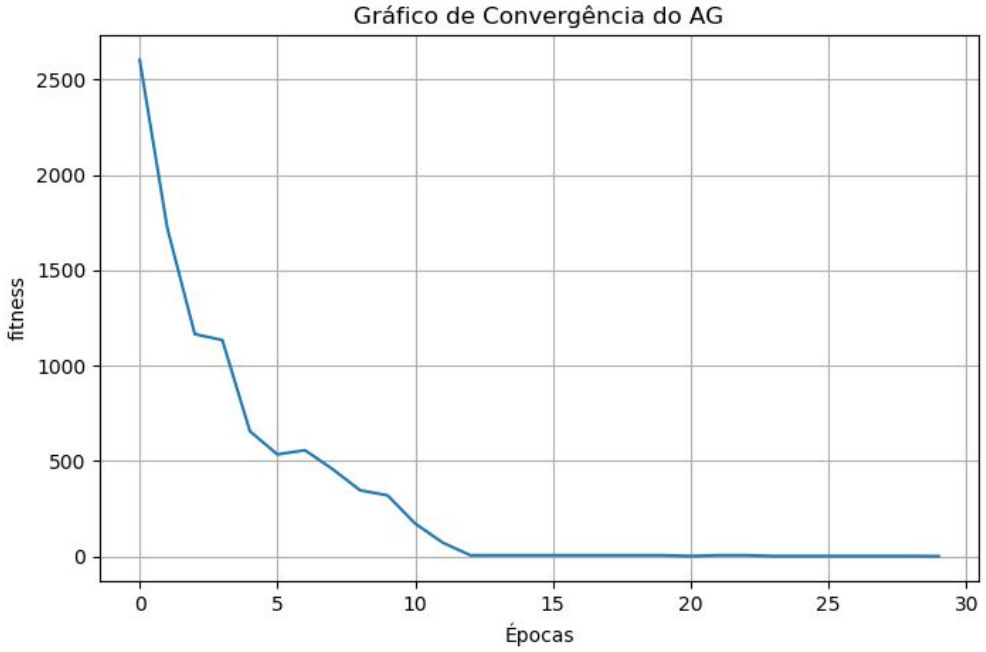
Foi utilizado 100 indivíduos na população inicial, a taxa de crossover é de 100%, ou seja, 100 indivíduos filhos por geração e a taxa de mutação é igual a 10% para cada gene do cromossomo.

13. Resultados Obtidos

13.1. Gráfico de Convergência da AG

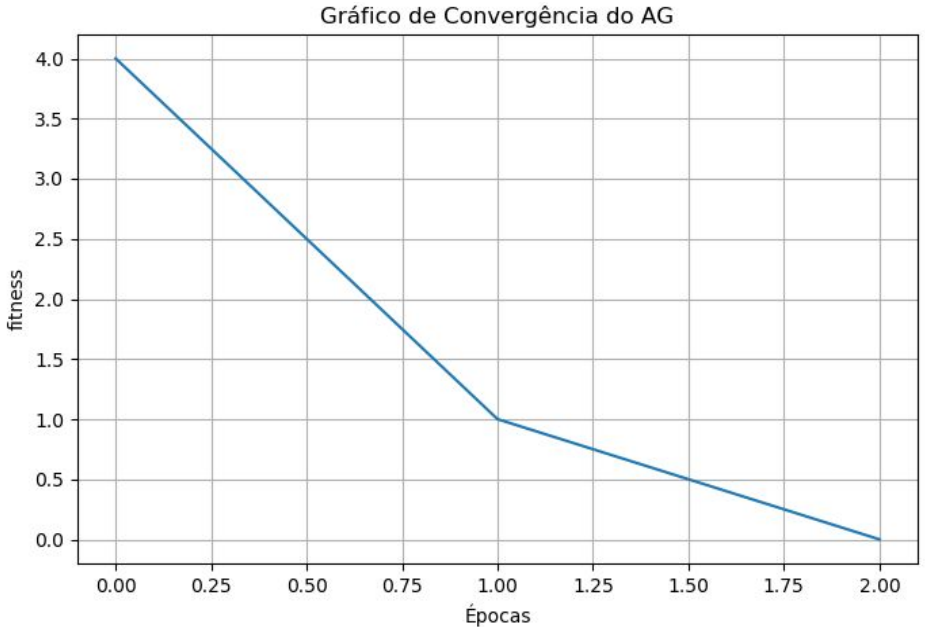
A seguir é apresentado gráficos de convergência, onde o eixo *x* representa a época e *y* o fitness. Mesmo que não encontre a resposta, o menor valor de *y* é oferecido no final.

- Para x1+x2+...+x15 = 800, temos que:



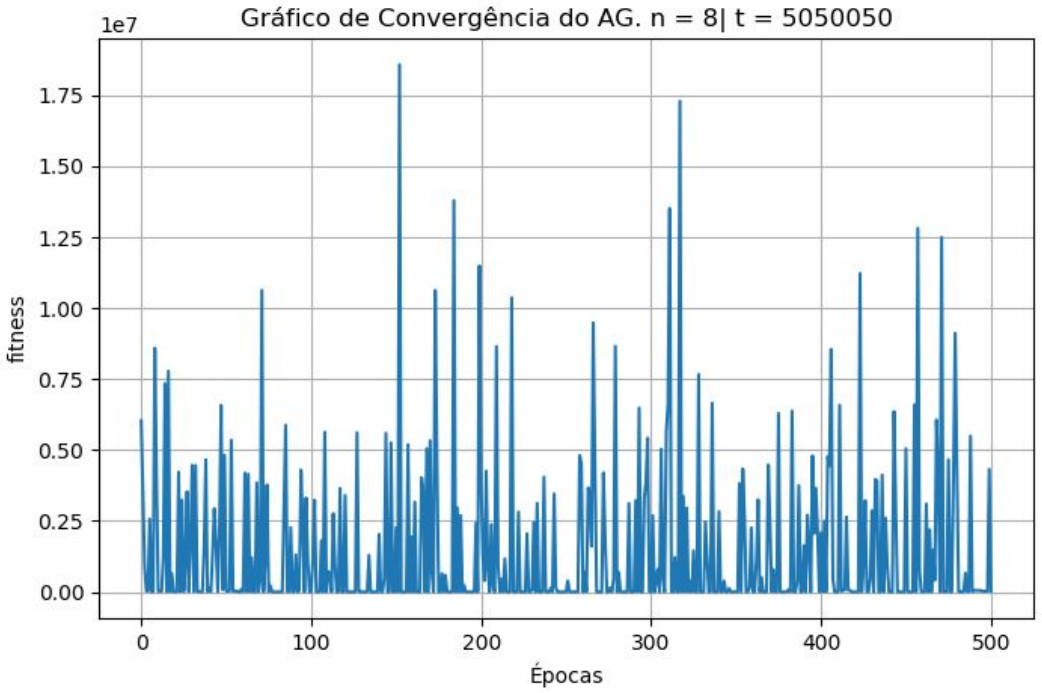
Tempo de Treinamento: 0.359 segundos.

- Para x1+x2+x3+x4 = 50, temos que:



Tempo de Treinamento: 0.016 segundos.

- Para x1 + x2 + x3 + x4 + x5 = 5.050.050



Tempo de Treinamento: 5.19 segundos

13.2 Resultados da execução do código

Abaixo temos mais alguns exemplos de resultados obtidos na execução

```
Ag - Somatoria - Buscando 13 numeros para 983467

        Chegou ao Limite de epocas: 500

        A Melhor Resposta das Geracoes:
Geracao: 438| Fitness: 2
 Cromossomo: [77033, 112650, 74651, 55607, 37472, 125550,
       27419, 60754, 120272, 31221, 41159, 65574, 154107]

Sum: 983469
Tempo de Treinamento : 5.481 segundos
```

```
Ag - Somatoria - Buscando 5 numeros para 12345

        Encontrado uma Resposta Ideal:
Geracao: 213| Fitness: 0
 Cromossomo: [817, 618, 6240, 838, 3832]

Sum: 12345
Tempo de Treinamento : 1.825 segundos
```

```
Ag - Somatoria - Buscando 13 numeros para 983467

        Chegou ao Limite de epocas: 500

        A Melhor Resposta das Geracoes:
Geracao: 481| Fitness: 13
 Cromossomo: [37581, 45302, 124416, 84896, 7350,
  99304, 22481, 132448, 81692, 30, 1672, 37986, 6854, 1498]

Sum: 983480
Tempo de Treinamento : 5.553 segundos
```

```
Ag - Somatoria - Buscando 10 numeros para 40000

        Chegou ao Limite de epocas: 500

        A Melhor Resposta das Geracoes:
Geracao: 302| Fitness: 1
 Cromossomo: [54, 13621, 3390, 10006,
     6747, 1380, 3916, 336, 232, 317]
Sum: 39999
Tempo de Treinamento : 4.904 segundos
```

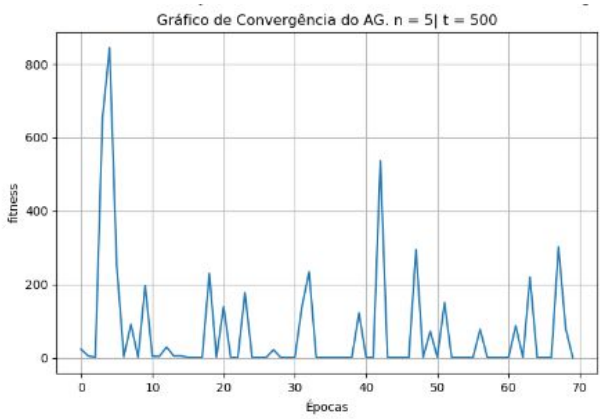
```
Ag - Somatoria - Buscando 20 numeros para 30

        Chegou ao Limite de epocas: 500

        A Melhor Resposta das Geracoes:
Geracao: 17| Fitness: 160
 Cromossomo: [14, 17, 16, 6, 3, 4, 12, 19,
  13, 2, 10, 11, 1, 9, 5, 0, 15, 8, 7, 18]

Sum: 190
Tempo de Treinamento : 9.189 segundos
```

```
Ag - Somatoria - Buscando 11 numeros para 1000000

        Chegou ao Limite de epocas: 500

        A Melhor Resposta das Geracoes:
Geracao: 247| Fitness: 28
 Cromossomo: [103455, 126228, 974, 165848, 29895,
     92056, 130171, 108512, 96478, 29107, 117304]

Sum: 1000028
Tempo de Treinamento : 5.148 segundos
```

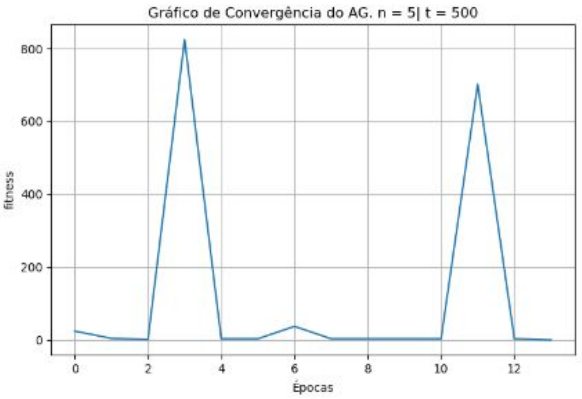
14. Análise dos Resultados

14.1. Aleatoriedade da AG

Observa-se que a execução do algoritmo genéticos dá resultados distintos para cada execução. No exemplo abaixo é mostrado duas execuções de n = 5 e t = 500, ou seja, buscando 5 números entre 0 e 499 cuja soma resulte em 500. Como é possível ver, o gráfico a direita encontrou um resultado mais rápido que o da esquerda, isso ocorre devido ao caráter probabilístico do método por usar aleatoriedade em quase todas as fases : na criação inicial, na seleção, no crossover e na mutação.



Geração 68: [104, 107, 13, 3, 273]
0.607 segundos



Geração 13 : [83, 37, 95, 51, 234]
0.119 segundos

14.2 Quando a soma dos n termos for maior que o número buscado

Sendo *n* a quantidade de valores e *t* o valor buscado, se a soma de 0 à n for maior que o número buscado, então não tem como alcançar um cromossomo ideal pois é impossível a somatória chegar a t com valores distintos.

Isso ocorre no exemplo acima com *n = 20* e *t = 30.* A soma de 0 à 19 dá 190 que é maior que 30. Por causa disso o melhor caso que se pode obter é ter esses n termos (0 à 19 nesse caso) como cromossomo, como ocorre no exemplo. Assim, mesmo não atingindo *fitness = 0,* atingiu o melhor resultado possível.

14.3 Imperfeição do método

Como apresentado nos exemplos acima, há casos em que não consegue alcançar um resultado ideal mesmo chegando ao final das épocas. Isso é devido além da aleatoriedade ao fato da combinação do número de números e o valor buscado, que podem proporcionar.

15. Conclusão

A execução do algoritmo neste trabalho se mostrou um sucesso devido a melhor resposta obtida ser encontrada em cerca de no máximo 10 segundos.

A solução Não é perfeita devido a aleatoriedade e ao espaço de busca que pode ficar imenso de forma fácil mas apesar disso mostrou-se eficiente para boa parte dos casos, e, mesmo quando não consegue o resultado final tende a ser próximo do esperado, além disso por o fitness ser a distância até o valor, mesmo que não atinja , pode-se escolher um dos genes e incrementar ou decrementar (depende se a somatória passou ou não do valor buscado) por esse valor, obtendo um novo gene e consequentemente a resposta ideal.

16. Referências Bibliográficas

SANTOS, D. D. ; PASOTTO, D. ; OLIVEIRA, G. M. B. . **Estudo comparativo de algoritmos genéticos aplicados ao escalonamento de tarefas**, São Paulo, v. 4, n.1, p. 99-112, 2004.

Material de curso cedido pela Profa. Dra. Rita Maria da Silva Julia, disponível em <https://www.moodle.ufu.br/mod/folder/view.php?id=155943>