



Universidade Federal do Ceará
Campus Sobral
Curso de Engenharia da Computação
Disciplina: Inteligência Computacional
Professor: Jarbas Joaci de Mesquita Sá Júnior

Algoritmo Genético baseado em ordem

Relatório de Projeto

Aluno: José Lopes de Souza Filho
Matrícula: 389097

SOBRAL
DEZEMBRO DE 2019

ÍNDICE

1. INTRODUÇÃO.....	3
2. DOMÍNIO DO PROBLEMA.....	4
3. REQUISITOS OPERACIONAIS.....	5
4. FUNCIONAMENTO DO ALGORITMO.....	6
4.1 Principais etapas de implementação do código.....	6
4.1.1 Descritivo das principais variáveis utilizadas no sistema:.....	6
4.1.2 Código comentado - Principais etapas.....	6
5. RESULTADOS OBTIDOS E DISCUSSÃO.....	8
6. CONCLUSÃO.....	10
7. REFERÊNCIAS.....	11

1. INTRODUÇÃO

Este projeto se propõe a desenvolver uma solução para o problema do caixeiro viajante utilizando algoritmos genéticos adaptados para problemas de ordem.

Algoritmos genéticos são uma classe de algoritmos baseados na biologia da evolução que apresenta conceitos como população, genes, cromossomos, hereditariedade, mutação, etc.

Esta classe de algoritmos visa buscar soluções aproximadas para problemas de busca e otimização.

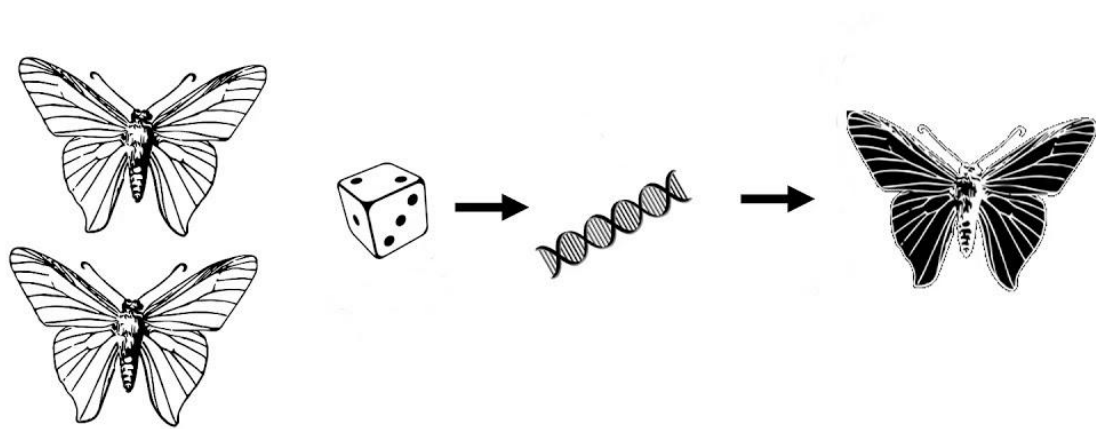


Figura 1 - Inteligência Artificial - Algoritmos Genéticos

2. DOMÍNIO DO PROBLEMA

Crie um algoritmo genético para o problema do caixeiro viajante representado por um grafo completo não direcionado de 14 vértices (cidades) cuja matriz de adjacência, que representa as distâncias entre as cidades, é:

Cidades	01	02	03	04	05	06	07	08	09	10	11	12	13	14
01	0	1	2	4	6	2	2	3	3	5	6	1	4	5
02		0	3	2	1	3	6	3	4	4	2	4	4	4
03			0	1	3	3	2	3	4	1	3	5	5	6
04				0	5	1	4	2	3	4	4	8	2	2
05					0	2	1	6	5	2	3	4	2	2
06						0	3	1	2	3	5	7	3	4
07							0	2	1	2	5	2	4	3
08								0	5	5	1	5	3	6
09									0	1	4	4	5	3
10										0	5	4	4	2
11											0	4	2	1
12												0	1	3
13													0	1
14														0

O algoritmo deve exibir o melhor caminho encontrado e o seu custo de percurso.

3. REQUISITOS OPERACIONAIS

Para a correta utilização deste sistema, deverá ser utilizado o Software Scilab na versão mais recente (na data de elaboração deste relatório a versão 6.0.2).

4. FUNCIONAMENTO DO ALGORITMO

4.1 PRINCIPAIS ETAPAS DE IMPLEMENTAÇÃO DO CÓDIGO

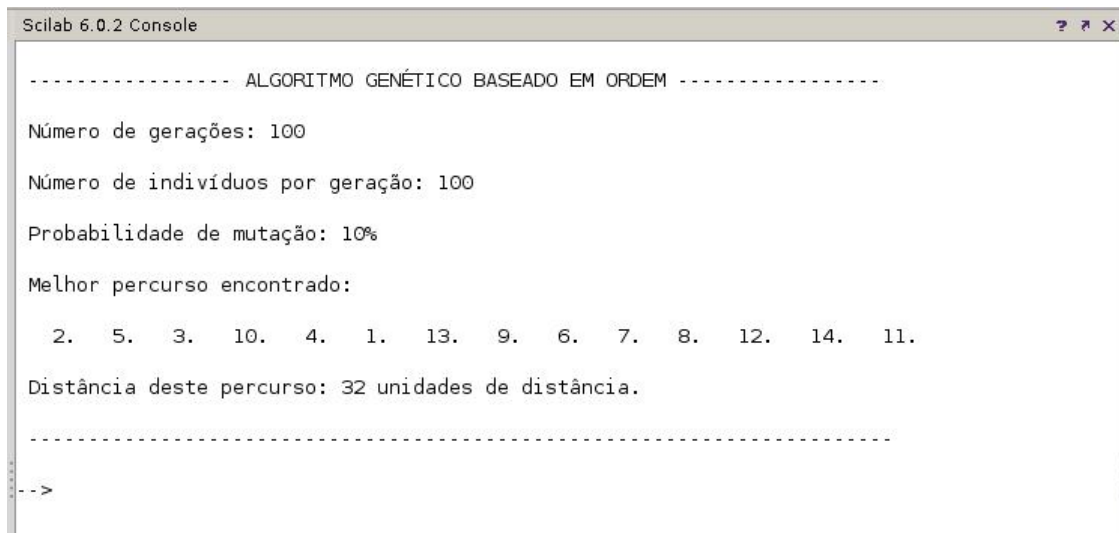
4.1.1 Descritivo das principais variáveis utilizadas no sistema:

1. AV: Vetor de avaliação. Neste vetor são armazenados os valores das distâncias percorridas nos cromossomos de mesmo índice. A cada geração este vetor é atualizado.
2. MA: Matriz de adjacência. Aqui ficam armazenadas as distâncias usadas na avaliação.
3. POP: População de cromossomos atual. A cada geração este vetor é atualizado.
4. POP_INICIAL: População inicial. Armazenada apenas para fins de comparação e avaliação de resultados.
5. SS: String de seleção gerada aleatoriamente para estabelecer critério de crossover.
6. aleatorio: Número entre 0 e 1 gerado aleatoriamente para determinar se haverá mutação no filho gerado.
7. filho: Cromossomo do último filho gerado no último processo de crossover.
8. gene1 e 2: Genes de valores 1 a 14 gerados aleatoriamente que determinarão quais posições serão trocadas em caso de mutação no filho.
9. geracoes: Número de gerações utilizadas na execução do algoritmo.
10. i-l: Variáveis auxiliares para os laços.
11. melhor_avaliacao: Menor valor de avaliação encontrado na última geração de filhos.
12. Melhor_percurso: Cromossomo de melhor avaliação encontrado na última geração de filhos.
13. n_pop: Número de cromossomos gerados aleatoriamente em cada geração.
14. pais: par de cromossomos selecionados pela roleta viciada para darem origem a um filho.
15. prob: Grau de probabilidade de ocorrer mutação no filho gerado

4.1.2 Código comentado - Principais etapas

- I. Inicialmente criou-se a geração inicial de cromossomos de tamanho 14 (número de cidades a serem percorridas pelo caixeiro). Esses valores foram criados de forma aleatória e o número de indivíduos foi estabelecido em 100 (podendo ser alterado na variável n_pop).
- II. Em seguida, foi criado o vetor de avaliação (AV) que recebe o cálculo das distâncias percorridas em cada cromossomo. Para este problema, quanto menor esse valor melhor o fitness daquele cromossomo.
- III. Na etapa seguinte, fizemos uso da roleta viciada para selecionar os pais. Neste caso em específico, buscamos fazer com que a roleta desse maior probabilidade aos menores valores. Esta solução foi encontrada subtraindo o valor máximo do vetor AV, somado a 1 (para que o maior valor ainda tenha possibilidade - ainda que mínima - de ser escolhido) menos os valores de cada avaliação. Desta forma temos uma “inversão” dos valores de avaliação. Os menores valores passaram a ter maior fitness e os maiores, menor fitness. Em seguida bastou passar estes valores no algoritmo padrão da roleta viciada.
- IV. Neste ponto ocorreu algo importante. Por algum motivo, a roleta (raramente, porém ocorria) retornava valores acima do tamanho máximo da população (100). Para resolver esse problema, foi criado um algoritmo de “proteção” que nas raras vezes que isso ocorria, garantia que a roleta seria girada novamente respeitando as regras de escolha e fitness, até que um valor abaixo ou igual a 100 fosse selecionado.

- V. Para a geração dos filhos foi utilizado o método de crossover baseado em ordem, gerando 100 novos filhos que serão a nova população na matriz POP.
- VI. Antes de serem lançados na matriz POP, cada filho gerado é submetido a um algoritmo de mutação baseado em ordem que dada uma probabilidade (estabelecida em 10% aqui), pode ou não alterar a posição de dois genes de cada filho gerado escolhidos aleatoriamente.
- VII. Todo o processo anterior é iterado 100 vezes (n_pop) e o cromossomo de avaliação mais baixa é exibido no console (melhor percurso) junto de sua distância (melhor avaliação).



```
Scilab 6.0.2 Console
----- ALGORITMO GENÉTICO BASEADO EM ORDEM -----
Número de gerações: 100
Número de indivíduos por geração: 100
Probabilidade de mutação: 10%
Melhor percurso encontrado:
  2.  5.  3. 10.  4.  1. 13.  9.  6.  7.  8. 12. 14. 11.
Distância deste percurso: 32 unidades de distância.
-----
-->
```

Figura 2 - Retorno do console ao fim da execução

5. RESULTADOS OBTIDOS E DISCUSSÃO

Para análise dos resultados e comportamento do algoritmo desenvolvido, estabeleceu-se como padrão a melhor avaliação conforme alterou-se o número de gerações em diferentes números de população. O algoritmo foi então executado 5 vezes para cada número de gerações, os dados foram coletados e quatro tabelas foram preenchidas: para 100, 50, 20 e 15 indivíduos. As tabelas e os valores coletados estão demonstrados abaixo.

Gerações/ Execuções	1	2	5	10	100	500	1000
Exec. 1	30	29	27	24	30	30	25
Exec. 2	30	30	30	30	31	32	30
Exec. 3	29	27	31	29	32	29	31
Exec. 4	28	29	26	26	30	25	31
Exec. 5	30	30	28	33	31	27	30
Médias	29,4	29	28,4	28,4	30,8	28,6	29,4

Tabela 1: Tabela de avaliações para uma população de 100 indivíduos e probabilidade de mutação de 10%

Gerações/ Execuções	1	2	5	10	100	500	1000
Exec. 1	24	32	35	33	36	32	33
Exec. 2	30	32	31	32	27	30	32
Exec. 3	29	28	31	32	32	30	31
Exec. 4	29	34	32	32	33	31	31
Exec. 5	28	27	34	32	30	31	32
Médias	28	30,6	32,6	32,2	31,6	30,8	31,8

Tabela 2: Tabela de avaliações para uma população de 50 indivíduos e probabilidade de mutação de 10%

Gerações/ Execuções	1	2	5	10	100	500	1000
Exec. 1	30	36	35	32	36	33	36
Exec. 2	30	29	33	40	30	32	33
Exec. 3	32	30	32	34	37	37	39
Exec. 4	33	33	30	32	35	36	36
Exec. 5	36	32	34	31	36	30	40
Médias	32,2	32	32,8	33,8	34,8	33,6	36,8

Tabela 3: Tabela de avaliações para uma população de 20 indivíduos e probabilidade de mutação de 10%

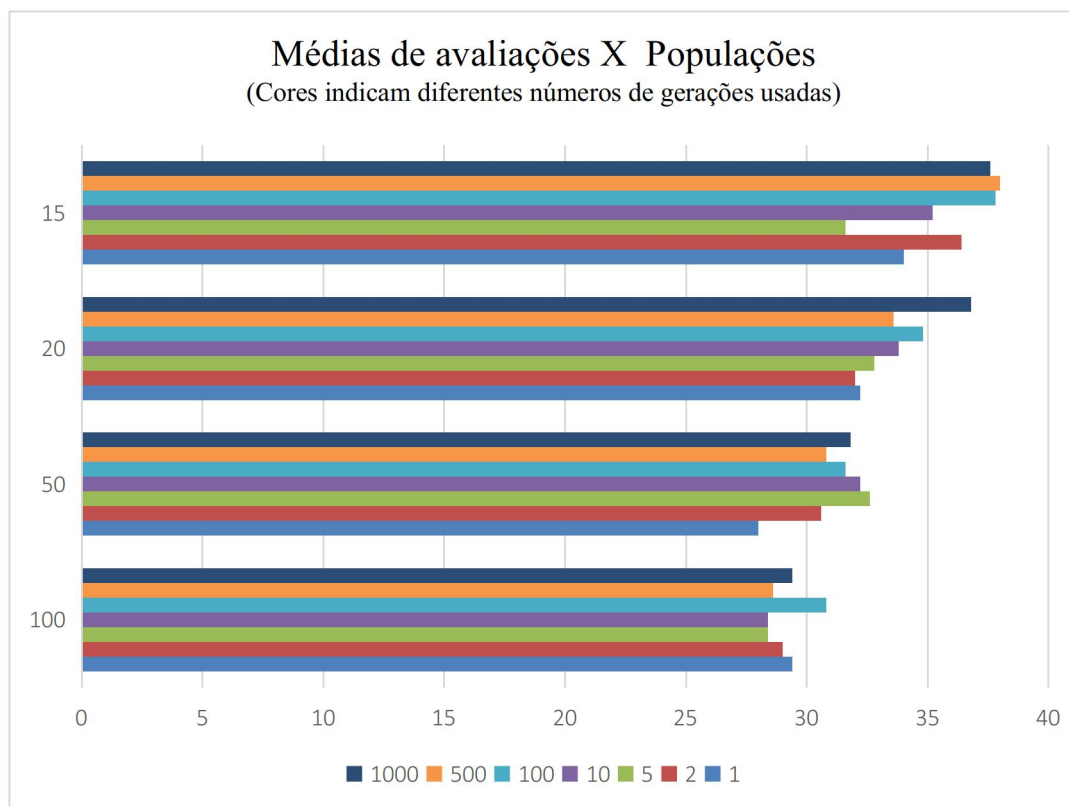
Algoritmo Genético baseado em ordem							
Gerações/ Execuções	1	2	5	10	100	500	1000
Exec. 1	38	38	32	35	39	39	43
Exec. 2	33	36	33	31	40	39	41
Exec. 3	30	36	30	36	37	33	33
Exec. 4	36	37	33	38	41	39	34
Exec. 5	33	35	30	36	32	40	37
Médias	34	36,4	31,6	35,2	37,8	38	37,6

Tabela 2: Tabela de avaliações para uma população de 15 indivíduos e probabilidade de mutação de 10%

Com os dados coletados podemos resumi-los em um único quadro:

Gerações/ População	1	2	5	10	100	500	1000
100	29,4	29	28,4	28,4	30,8	28,6	29,4
50	28	30,6	32,6	32,2	31,6	30,8	31,8
20	32,2	32	32,8	33,8	34,8	33,6	36,8
15	34	36,4	31,6	35,2	37,8	38	37,6

Tabela 2: Tabela de médias de avaliações para diversas populações diferentes gerações e probabilidade de mutação de 10%



Pelo gráfico acima podemos observar que conforme o número de gerações diminui a média das melhores avaliações encontradas tende a aumentar, porém observamos que o número de gerações não parece exercer um poder de influência tão grande no problema proposto.

6. CONCLUSÃO

Este trabalho teve como proposta desenvolver um algoritmo genético que se propunha a resolver o problema do caixeiro viajante com o menor custo possível. Ao executar o algoritmo criado sob diferentes cenários, pode-se observar que os dados encontram um ponto ótimo de forma muito rápida (bastam poucas gerações), o que pode ser comprovado ao analisar a estabilidade das soluções conforme se alteram as gerações, o que leva a crer que para este problema em específico, o número de gerações não tem tanto poder de influência. Porém ao alterar o número da população de indivíduos podemos ver uma alteração bem clara. Conforme o número de indivíduos por população diminui, a qualidade da solução tende a piorar o que mais uma vez pode estar ligado a essa rápida convergência das avaliações.

Portanto, podemos concluir que o algoritmo se comporta da forma esperada, comprovando a eficácia do método e da forma como ele foi implementado.

7. REFERÊNCIAS

1. LEITE, Mário. Scilab. Uma abordagem prática e didática: 2. ed.: Ciência moderna, 2015.
2. COPPIN, Ben. Inteligência Artificial: 1. ed.: LTC, 2017.
3. LINDEN, Ricardo. Algoritmos Genéticos: 3. ed.: Ciência moderna, 2012.