Fernando José Ferreira Neto - 2018001665
Giovany da Silva Santos - 2018007758

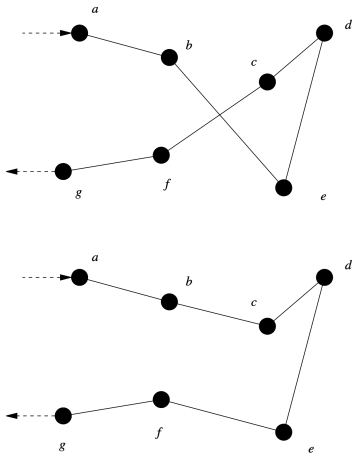
a) Dentre as formas de representar o indivíduo para este problema podemos citar vetor de caracteres, representação binária e representação real. O vetor de caracteres poderia ser reproduzido com letras e números relacionados aos locais, exemplo S1, S2, N1, N2, etc. A representação binária poderia ser feita mostrando cada cidade por um padrão de bits. Por exemplo, vamos imaginar que precisamos visitar 5 cidades, portanto a nossa representação precisaria de 3 bits começando em 001 (cidade 1) e indo até 101 (cidade 5), com isso a representação poderia ser feita dessa forma: 001 010 011 100 101. Os espaços em branco são colocados para auxiliar na leitura. A representação real é feita por meio de um vetor composto por números inteiros, onde cada número inteiro representa um local.

b) Dentre as representações citadas na letra A, a melhor seria a representação real. A representação real é melhor que as demais pois para um grande número de locais, a manipulação dos dados de números inteiros na forma decimal é mais simples do que utilizar caracteres, já que será necessário fazer a combinação de letras e números, e também é melhor que números na forma binária, visto que estes necessitam de uma grande quantidade de bits para serem representados.

c) Guloso - Nessa heurística a função custo é feita pelo melhor decisão do menor caminho local, dessa forma as arestas são escolhidas por menor custo. Obs: usamos a referência em [1]

2-opt: Essa heurística se baseia em uma busca local onde se tem uma solução viável e a partir dela várias melhorias são feitas da seguinte forma:

Nessa figura podemos ver que cruzamentos entre arestas são avaliados, sempre que for possível diminuir o custo entre visitar dois vértices esses cruzamentos serão concluídos. No final da heurística teremos uma solução inicial boa para poder utilizar no algoritmo genético.



Obs: usamos a referência em [2]

d)

Crossover:

-PMX(Partially Mapped Crossover)

Exemplo:

p1= (1 2 3 | 4 5 6 7 | 8 9)

p2 = (4 5 2 | 1 8 7 6 | 9 3)

Mapeamento: 4↔1, 5↔8, 6↔7, 7↔6

Nesse tipo de operador são indicados dois marcadores nos pais, sendo que os elementos entre os marcadores de p1 serão copiados para a mesma posição de d2, que inicialmente também possui os marcadores. Em seguida, serão avaliadas as posições após os segundos marcadores dos descendentes, por exemplo, é feita a comparação dos elementos presentes após o segundo marcador de p1, que são 8 e 9. Como o 8 já está em d1 é colocado um X na posição que ele ocuparia e como o 9 não está em d1 ele é copiado. Logo após isso, é feita a avaliação da mesma maneira com os elementos antes do primeiro marcador. Posteriormente, para as posições que estão com X é feita a análise do mapeamento. Tendo como exemplo os X de d1, olhamos a mesma posição em p1 que correspondem aos elementos 1 e 8, e de acordo com o mapeamento, 1↔4 e 8↔5, então, trocamos os X por 4 e 5. O mesmo procedimento se repete para gerar o d2.

PMX do exemplo:

d1= (XXX|1876|XX) ⇒ (XXX|1876|X9) ⇒ (X23|1876|X9) ⇒(423|1876|59)

d2 = (XXX|4567|XX)⇒ (XXX|4567|93) ⇒ (XX2|4567|93) ⇒ (182|4567|93)

OX (Order Crossover)

Nessa operação é feito dois cortes nos pais e nos descendentes, onde os elementos entre os marcadores de p1 são copiados para as mesmas posições de d1. Em seguida, são obtidos os elementos a partir do segundo corte de p2, no exemplo abaixo origina a seguinte sequência, 5 9 4 2 3 1 8 7 6. Os elementos que estão entre os marcadores de d1 são excluídos dessa sequência e o restante é copiado para o d1 a partir do segundo marcador. O mesmo procedimento se repete para obter o d2.

p1=(1 8 2 | 4 5 6 7 | 9 3)

p2=(4 2 3 | 1 8 7 6 | 5 9)

d1=(3 1 8 | 4 5 6 7 | 9 2)

d2=(2 4 5 | 1 8 7 6 | 9 3)

Mutação:

Swap: Esse operador seleciona de modo aleatório duas cidades, depois faz a troca dessas cidades. Exemplo:

1 2 3 4 5 6 7

1 2 6 4 5 3 7

PI (Position inversion)

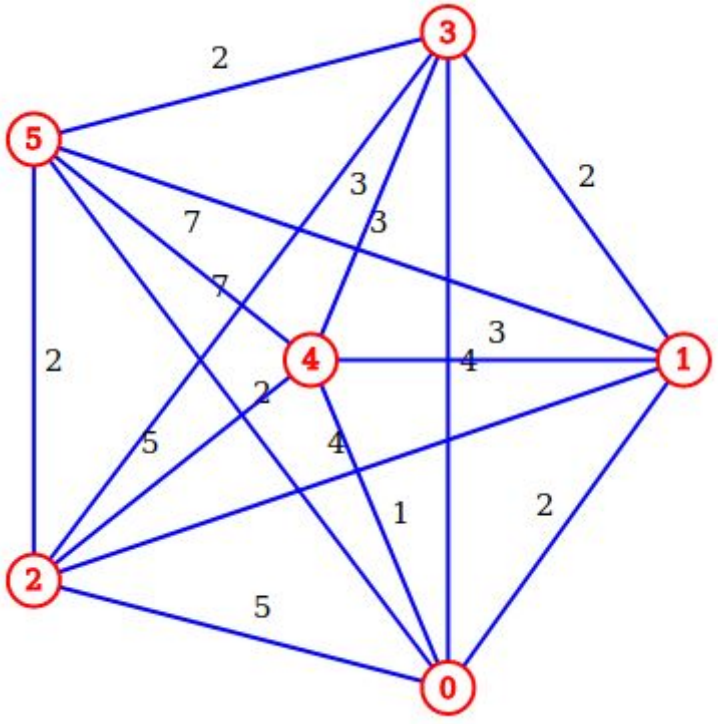
Nesse tipo de mutação é realizado dois pontos de cortes no cromossomo e os elementos que estão entre os dois cortes são invertidos de posição. Exemplo:

(1 2 3 4 5 6 7 8)→(1 2 | 3 4 5 6 | 7 8)→(1 2 | 6 5 4 3 | 7 8)→(1 2 6 5 4 3 7 8)

Obs: usamos a referência em [3]

e)

Exemplo que estamos usando como entrada:



Usamos operador OX (Order crossover) para o cruzamento entre indivíduos, e o Swap para mutação. O código foi desenvolvido em C++. Para o teste das entradas setamos o número de gerações como 100 e o número de indivíduos com 10, e um número máximo de mutação de 3 indivíduos por geração. Além disso utilizamos o grafo acima para fazer nossos testes, um grafo completo composto de 6 cidades.

A cada geração os indivíduos são pareados e é feita o crossover entre eles, lembrando que o número de indivíduos sempre deverá ser par, pois na nossa implementação utilizamos essa abordagem. Além disso, é feita uma mutação com no máximo 3 indivíduos. Enfim sempre é verificado se a solução

atual é melhor que o fitness (melhor valor), caso seja, o vetor solução será copiado para o vetor de melhor solução e o fitness é atualizado.A saída foi formatada da seguinte forma

Solução: x1,x2,x3..xn ,que representa a sequência de nós visitados.

Valor: V ,que representa o custo desse percurso.

Fontes:

[1] https://sites.icmc.usp.br/andretta/ensino/aulas/sme0241-1-19/aula3-TSP.pdf

[2] https://homepages.dcc.ufmg.br/~nivio/cursos/pa03/tp2/tp22/tp22.html

[3] https://www.maxwell.vrac.puc-rio.br/3725/3725_4.PDF

[4] Livro: Inteligência Artificial - George Luger 6ª edição