

UNIVERSIDADE DO SUL DE SANTA CATARINA

JONATHAN NEVES

Algoritmo Genético:

Desafio do Caixeiro Viajante

TUBARÃO

2020

# Qual forma de representação dos indivíduos foi utilizada para representar cada solução.

Foi utilizado para representar os indivíduos a permutação, onde cada cidade receberá uma Letra em ordem alfabética de cadastro.

**Ex.: A->D->B->C.**

Foi escolhido permutação, pois ela é a melhor forma de representar uma ordem específica de tarefas como é o caso do caixeiro viajante, onde devem passar por todas as cidades sem repeti-las.

# Como foi feito o cálculo do *fitness* de cada possível solução

Para realizar o cálculo do fitness foi feito a soma dos tempos entre a sequência das cidades. Lembrando que também foi necessário realizar a soma do tempo entre a cidade final com a cidade inicial. Pois o caixeiro passa por todas as cidades e depois retorna para cidade inicial.

Ex.: A para B = 30min; A para C = 15 min; B para C = 5min;

A->C->B->A = [*tempoAC+tempoCB+tempoBA]* = 50min;

# Qual percentual de mutação genética (caso tenha sido utilizada)

Não foi utilizado nenhuma taxa de mutação pois foi realizado a mutação selecionando 2 cidades aleatórias de um indivíduo aleatório e trocado de lugar entre si.

Ex.: [A->**C**->B->**D**] => [A->**D**->B->**C**]

Este é o melhor método para utilizar em permutação para evitar casos inválidos, onde o indivíduo possui um caminho com cidades repetidas.

# Como se deu o método de seleção dos indivíduos para reprodução no momento de estabelecer as próximas gerações

Os 4 indivíduos são avaliados com base no seu fitness e são escolhidos os 2 indivíduos com menor tempo (*o mais apto*) entre os 4 da geração atual para gerar descendentes para a próxima população. Logo o objetivo do problema é encontrar a sequência de cidades com o menor tempo.

O algoritmo inicia a geração com 2 indivíduos, e gera mais dois indivíduos através do crossover, e por fim realiza a mutação em um deles, finalizando com 4 indivíduos. No final é escolhido apenas os 2 indivíduos mais apto que serão mantidos para a próxima geração e repete o processo até cumprir o critério de parada.

Para o crossover foi utilizado *Non-Wrapping Ordered Crossover (NWOX).* Consiste em pegar uma posição inicial e uma posição final, e adicionar todos os valores presente no pai entre estas posições ao filho. Após isso, é preciso completar o indivíduo com as cidades restantes, então é realizado uma verificação se o filho 1 não contém a cidade do pai 2, ou vice-versa, então é adicionado ao indivíduo.

***Posição inicial***= 2; ***Posição Final*** = 4;

Ex.: Pai 1 [A->**B->C->D**->E] >> Filho 1[**B->C->D**->\_->\_] >> Filho 1[B->C->D->**E->A**]

Pai 2 [E->**D->C->B**->A] >> Filho 2[**D->C->B**->\_->\_] >> Filho 2[D->C->B->**A->E**]

# Qual o critério de parada.

O critério de parada é a escolha do usuário, são dois gerações ou automática:

* Geração: O usuário pode escolher o número de gerações desejadas, ou seja, o algoritmo irá executar *n* vezes. (Dependendo do número de gerações pode ou não chegar no melhor resultado);
* Automático: O algoritmo em tempos em tempos verifica se a melhor solução está sendo alterada, se em 50 gerações o algoritmo não achar nenhuma solução melhor, o algoritmo para.

# Fluxograma do algoritmo.

