Experimento - Problema do Caixeiro Viajante

## Eliton Zimmermann, Gabriel Filippi, Lucas Costa

Univille - Universidade da Regia˜o de Joinville

Joinville – SC – Brazil

***Resumo.*** *Este artigo irá explicar a respeito do experimento “Problema do caixeiro viajante” atividade da matéria de introdução a teoria da computação, ministrado pelo professor Leanderson André*

# Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV) é um problema clássico de otimização trabalhado por diversos pesquisadores pela sua simplicidade na formulação, complexidade na resolução e sua larga aplicabilidade no mundo real, e sua resolução podem trazer inúmeros benefícios para a sociedade.

O caso que será tratado na atividade será um conjunto de cidades, todas elas sendo representadas por pontos nos grafos. Cada trajeto ou ligações do grafo terão um custo de distância e valores numéricos diferentes. Para ser concluído o trajeto deve-se passar por todas as cidades sem repetir nenhuma, no entanto, tentando ter o menor custo possível para fazer o processo.

O ponto de saída é aleatório e o ponto de chegada é variável, dependendo quando passar por todas as cidades, caso um ponto vá para ele mesmo, o custo será 0, como no caso de viagem da cidade para própria cidade.

A ideia para este experimento e´ executar treˆs algoritmos ao menos 30 vezes e

identificar a me´dia, o desvio de padra˜o e assim qual a melhor soluc¸a˜o dentre esses treˆs.

# Descric¸a˜o dos Algoritmos

## Algoritmo Aleato´rio

Com o algoritmo aleatório, como o próprio nome diz, não existe muita lógica nos caminhos escolhidos, são apenas usados de forma aleatória. Logo o algoritmo sairá de um ponto aleatório para um caminho aleatório como um viajante sem rumo.

## Algoritmo Guloso

Já o algoritmo guloso, a partida também é de uma cidade aleatória, será feito a escolha dos caminhos com base na próxima menor distância, desta forma o algoritmo irá ficar preso muitas vezes, não tendo a melhor solução.

## Algoritmo Semi-Guloso ou Híbrido

O algoritmo h´ıbrido e´ uma combinac¸a˜o entre o algoritmo aleato´rio e o algoritmo guloso. Neste caso é sorteado um número inteiro, que irá selecionar qual vai ser a forma de escolha do próximo caminho do algoritmo, aleatório ou guloso. Um exemplo para demonstrar como funciona é colocar um range de 0 a 200, caso o número sorteado por exemplo for 20 e o valor do caminho for maior escolhemos a forma guloso caso o contrário será escolhido o aleatório

# Resultados

Ao executar 30 vezes cada um dos algoritmos, os seguintes resultados foram obtidos utilizando da me´dia aritme´tica, desvio padra˜o e diagrama de caixa:

Aleatório – Média(Distância): 3138,97KM – Desvio padrão (Distância): 1415.68KM

Guloso - Média(Distância): 409.97KM – Desvio padrão (Distância): 1042.13KM

Híbrido - Média(Distância): 1041.82KM – Desvio padrão (Distância): 1042.17KM

# Concluso˜es

Analisando os testes realizados, e´ poss´ıvel concluir um item que quase se equivalente aos algoritmos: seu tempo de execuc¸a˜o. Podemos relacionar isso com a quantidade de dados

processados, pois na˜o colocamos os algoritmos em testes de estresse, para validar qual algoritmo demoraria mais para encontrar sua soluc¸a˜o com grandes quantidades de dados. Por via de regra, o algoritmo aleato´rio tende a ser o que pode achar uma soluc¸a˜o mais ra´pido devido a sua lo´gica, se pegar qualquer caminho indiferente da distaˆncia, porem isso na˜o analisa o ponto mais relevante deste experimento que se trata de encontrar a melhor soluc¸a˜o. Analisando cada algoritmo com os dados levantados, podemos constar que:

Em todas as situac¸o˜es o algoritmo aleato´rio se mostrou o pior para determinar a rota mais curta. Sua me´dia e desvio padra˜o ultrapassa a soma dos respectivos valores dos outros dois algoritmos. Em relac¸a˜o ao diagrama de caixa, e´ o que possui os maiores outliers e tambe´m o que possui maior dispersa˜o dos seus pontos, sendo assim ”a maior caixa”. Podemos visualizar que o algoritmo aleato´rio gera soluc¸o˜es capazes de se igualar ao algoritmo guloso e h´ıbrido porem tambe´m acaba por gerar soluc¸o˜es que extrapolam os valores, tornando-se a pior soluc¸a˜o.

O algoritmo guloso, surpreendentemente e´ o que gerou as menores rotas, mesmo levando em considerac¸a˜o o risco dele ter problema nos ”o´timos locais”. Dos treˆs algorit- mos foi o que gerou as melhores soluc¸o˜es, tanto analisando os seus valores de me´dia e desvio, quando ao diagrama de caixa, no qual podemos visualizar que grande parte de suas soluc¸o˜es se encontram em valores pro´ximos. Tendo a soluc¸a˜o mais longa equivalendo a me´dia das soluc¸o˜es do algoritmo h´ıbrido.

O algoritmo h´ıbrido aparentou ser o mais equilibrado dentre os treˆs quando anali- samos o diagrama de caixa. Se analisarmos a sua me´dia e desvio padra˜o, ele aparenta se aproximar do algoritmo aleato´rio, o que nos indicaria que na˜o compensa utiliza-lo, porem olhando para o diagrama de caixa, podemos visualizar que o que causou esses valores elevados nas me´dias foram seus outliers. Se considerarmos que mais de 80 porcento dos casos na˜o entram nesta condic¸a˜o, ele acaba se tornando um algoritmo com soluc¸o˜es bem equilibradas e capaz de contornar o problema de o´timos locais do algoritmo guloso.

Como conclusa˜o, neste experimento, levando em considerac¸a˜o os dados usados, o melhor algoritmo seria o guloso, porem, e´ necessa´rio levantar a ressalva de que, e´ poss´ıvel que acontecem problemas de o´timos locais, o que podem gerar outliers para no´s. Podemos especular que, nesta situac¸a˜o o algoritmo guloso se apresentou a melhor soluc¸a˜o pela quantidade de dados que o algoritmo precisa consumir, e o numero de tentativas, pois em teoria o algoritmo hibrido que combina os outros dois tendem a ser uma boa soluc¸a˜o quando temos uma grande quantidade de dados.