PUC Minas
A B A K Ó S
Instituto de Ciências Exatas e Informática

Trabalho Prático 2 - Projeto e Análise de Algoritmos*

Caio Massote Andrade[1]
Henrique Moura de Sousa Belo[2]
Rodrigo Paiva Fentanes[3]

Resumo

Os algoritmos são a base fundamental da resolução de problemas em ciências da computação.Problemas de otimização são bastante comuns em vários setores produtivos, sendo assim escrever programas eficientes é importante por diversas razões. Em primeiro lugar, programas mais eficientes utilizam recursos do computador de forma mais adequada, o que pode levar a uma melhora significativa no desempenho e na velocidade de execução. Além disso, programas mais eficientes tendem a consumir menos recursos, como memória e energia, o que é especialmente importante em dispositivos com recursos limitados, como smartphones e dispositivos de internet das coisas. O objetivo desse trabalho é um estudo sobre como otimizar um programa fornecido pelo professor utilizando técinas de projeto estudadas em sala.

Palavras-chave: Branch and Bound. Força Bruta. Algoritmos. Tempo. Python

*Submetido em 13/03/2023
[1]PUC-Minas, Brasil– thecaiobr@gmail.com
[2]PUC-Minas, Brasil– henriquebelo10@gmail.com
[3]PUC-Minas, Brasil– fentanesrodrigo@gmail.com

1 INTRODUÇÃO

Problemas de otimização são comuns em diversos setores produtivos, como o planejamento do transporte de cargas entre diferentes pontos geográficos. Neste trabalho, abordaremos o desafio de coletar e entregar produtos, visando minimizar o consumo de combustível. O objetivo é encontrar uma solução eficiente que permita ao caminhão visitar todas as lojas apenas uma vez, levando em consideração restrições de capacidade e rendimento do veículo. Para isso, será desenvolvido dois algoritmos que consideram a distância percorrida, o rendimento do caminhão e as características de cada loja, como a localização geográfica e a quantidade de produtos a serem transportados. As informações sobre as lojas serão lidas a partir de um arquivo de texto chamado "lojas.txt". Ao otimizar o processo de coleta e entrega de produtos, espera-se obter uma solução funcional e satisfatória para o problema descrito anteriormente.

2 ALGORITMO DE FORÇA BRUTA

2.1 Solução Proposta

Solução por força bruta para gerar as soluções possíveis e escolher a que gasta a menor quantidade de combustível, respeitando as restrições quanto à carga máxima do caminhão. O programa tambem inclui uma interface gráfica, que mostra em animação a sequência das viagens do caminhão e a variação de sua carga e combustivel ao longo do trajeto.

2.2 Implementação

2.2.1 Detalhes da solução

A partir da leitura do arquivo lojas.txt, o algoritmo extrai os dados sobre as lojas e os pacotes a serem entregados dentro da função "ler lojas do arquivo" e organiza esses dados, respectivamente nos vetores lojas e destinos.

Logo após, partimos para a parte de maior interesse do algoritmo para esse trabalho, localizada dentro da função "calcular rota otima" onde, de forma recursiva, permutamos as possíveis combinações de trajetos para as lojas presentes na entrada, para então, iterar em cada uma dessas permutações, sobre cada loja, com a finalidade de calcular o combustível gasto por meio da função "calcular combustivel", de acordo com a fórmula apresentada no enunciado do trabalho, e validar se cada uma das rotas respeita o limite de carga do caminhão, por meio de uma estrutura de dados da linguagem Python chamada set, similar a um array regular. Para cada pacote coletado, adicionamos o seu destino ao set das cargas, e caso o tamanho deste set

exceda o valor máximo suportado pelo caminhão, a rota gerada será considerada imediatamente inválida.

Ao final de cada iteração, conferimos o tamanho do set de cargas, pois caso este não encontre-se vazio ao fim da rota do caminhão, significa que algum pacote não foi entregue, tornando a rota inválida para o nosso objetivo. Caso a rota passe por todas as condições de validade impostas e registre um valor de combustível gasto menor que o mais eficiente registrado até o momento, aquela rota passa a ocupar o valor de rota ótima, portanto é registrada em uma variável, assim como o valor de combustível gasto, até que outra rota mais eficiente seja encontrada.

Após checarmos todas as permutações de rota, teremos assim a garantia de que encontramos a solução ótima para aquele ambiente de lojas, pacotes e cargas dentro das variáveis "rota otima" e "menor combustivel total" respeitando as condições impostas de capacidade do caminhão e entrega de pacotes.

2.2.2 Parte gráfica

A parte gráfica do nosso algoritmo, foi desenvolvida utilizando a biblioteca matplotlib do python. A biblioteca matplotlib é uma biblioteca popular de visualização de dados em Python. Ela fornece uma ampla gama de ferramentas para criar gráficos, diagramas, histogramas, dispersões e muitos outros tipos de visualizações. É uma das bibliotecas mais utilizadas para visualização de dados em Python devido à sua flexibilidade e capacidade de produzir gráficos de alta qualidade.

A biblioteca matplotlib possui vários módulos, sendo o módulo mais comumente usado chamado pyplot. O pyplot fornece uma interface semelhante ao MATLAB, permitindo criarmos o gráfico das rotas, também utilizamos o módulo Animation, para executarmos a animaçao das rotas para facilitar a visualizaçao dos dados.

2.3 Relatório de Testes

A partir da implementação realizamos os testes do algoritmo de força bruta chegando aos seguintes resultados e conclusões:

Obtivemos a seguinte rota ótima: 0,1,2,3,5,4,9,7,8,6,0. Essa é uma rota correta pois além de gastar a menor quantidade de combustível possível que é de 102.7L, sem violar as restrições quanto à carga máxima do caminhão, podendo passar apenas uma única vez por cada uma das cidades e realizando todas as entregas propostas no arquivo de entrada "lojas.txt".

Além disso, obtivemos o seguinte gasto de combustível por trecho:

- De loja 0 para loja 1: 15.78947 litros
- De loja 1 para loja 2: 11.76471 litros

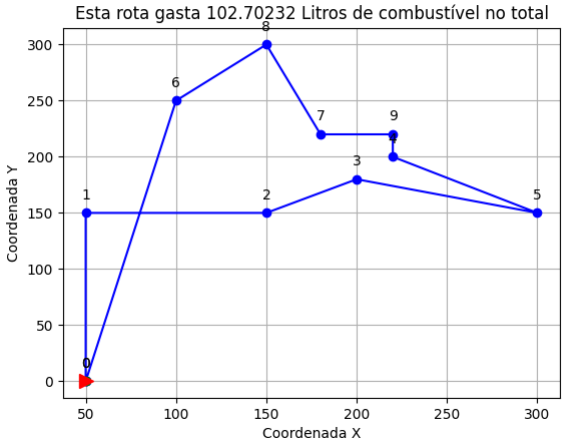
- De loja 2 para loja 3: 6.47884 litros
- De loja 3 para loja 5: 11.60034 litros
- De loja 5 para loja 4: 11.09880 litros
- De loja 4 para loja 9: 2.10526 litros
- De loja 9 para loja 7: 4.44444 litros
- De loja 7 para loja 8: 10.05177 litros
- De loja 8 para loja 6: 8.31890 litros
- De loja 6 para loja 0: 29.99423 litros

Por fim, obtivemos um tempo de execução de 3.75s.

```
3
Rota ótima: [0, 1, 2, 3, 5, 4, 9, 7, 8, 6, 0]
Combustível total gasto: 102.70232 litros
Combustível gasto por trecho:
De loja 0 para loja 1: 15.78947 litros
De loja 1 para loja 2: 11.76471 litros
De loja 2 para loja 3: 6.47884 litros
De loja 3 para loja 5: 11.60034 litros
De loja 5 para loja 4: 11.09880 litros
De loja 4 para loja 9: 2.10526 litros
De loja 9 para loja 7: 4.44444 litros
De loja 7 para loja 8: 10.05177 litros
De loja 8 para loja 6: 8.31890 litros
De loja 6 para loja 0: 29.99423 litros
Tempo de execução: 5.040630102157593
```

Saida de dados do programa

Logo abaixo, criamos também um grafo dinâmico que ilustra o caminho percorrido pelo caminhão(seta vermelha) para a realização da melhor rota entre cada uma das lojas, mostrando também o consumo de combustível gasto em cada trecho da viagem.



Grafo Força Bruta

3 ALGORITMO DE BRANCH AND BOUND

3.1 Solução Proposta

Solução por branch-and-bound, criada a partir do item anterior, de forma a eliminar os ramos da árvore de soluções que tenham um custo maior que o já obtido anteriormente.

3.2 Implementação

3.2.1 Detalhes da solução

O código implementa um algoritmo para encontrar a rota ótima para um caminhão que precisa visitar várias lojas e entregar produtos nelas. A rota ótima é aquela que minimiza o consumo de combustível do caminhão.

A partir da leitura do arquivo lojas.txt, o algoritmo extrai os dados sobre as lojas e os pacotes a serem entregados dentro da função "ler lojas do arquivo" e organiza esses dados, respectivamente nos vetores lojas e destinos.

Em seguida, são definidas várias funções auxiliares. A função calculardistancia calcula a distância euclidiana entre duas lojas. A função calcularcombustivel calcula a quantidade de combustível gasto para percorrer uma determinada distância, levando em consideração a carga atual do caminhão e as restrições estipuladas pelo problema.

A função calcularlowerbound calcula uma estimativa inferior (lower bound) para o consumo de combustível total restante, com base nas lojas restantes e suas distâncias. Essa estimativa é usada para otimizar o algoritmo de busca, por meio de critérios de poda, ou seja, se um caminho não for promissor e não levar ao resultado final desejado o caminho é interrompido na hora e já vai em busca de um outro que seja mais interessante, evitando assim cálculos e atribuições desnecessárias, melhorando assim o desempenho do codigo quando comparado ao força bruta em que ele avalia todos os caminhos possiveis primeiro para depois encontrar o melhor.

A função principal calcularrotaotima implementa em si o branch and bound, onde nela contem a utilização do lowerbound no calculo das rotas para identificar o melhor caminho a ser tomado, além de realizar corretamente os métodos para a sua execução como os critérios de poda, a remoção e update do caminho. A função backtrackrota é uma função auxiliar que percorre todas as possíveis rotas, atualizando a rota ótima e o consumo de combustível total conforme encontra rotas melhores.

Por fim, a função main lê as lojas de um arquivo, recebe a capacidade do caminhão como entrada do usuário e chama as funções adequadas para calcular a rota ótima e exibir os resultados. Além disso, a função exibiranimacao mostra uma animação da rota percorrida pelo caminhão.

3.2.2 Parte gráfica

Implementação da parte gráfica segue os mesmos princípios do força bruta.

3.3 Relatório de Testes

A partir da implementação realizamos os testes do algoritmo de branch and bound chegando aos seguintes resultados e conclusões:

Obtivemos a seguinte rota ótima: 0,1,2,3,5,4,9,7,8,6,0. Logicamente ela é a mesma rota encontrada pelo algoritmo de força bruta, no entanto, o branch and bound possui uma performance muito melhor quando comparado com o algoritmo anterior, isso se deve pelo metodo de implementação do branch and bound explicado na sessão anterior. Essa é uma rota correta pois além de gastar a menor quantidade de combustível possível que é de 102.7L, sem violar as restrições quanto à carga máxima do caminhão, podendo passar apenas uma única vez por cada uma das cidades e realizando todas as entregas propostas no arquivo de entrada "lojas.txt".

Além disso, obtivemos o seguinte gasto de combustível por trecho:

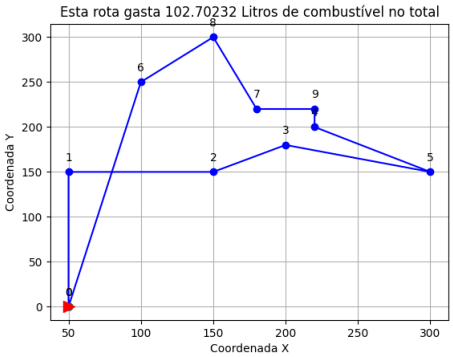
- De loja 0 para loja 1: 15.78947 litros
- De loja 1 para loja 2: 11.76471 litros
- De loja 2 para loja 3: 6.47884 litros
- De loja 3 para loja 5: 11.60034 litros
- De loja 5 para loja 4: 11.09880 litros
- De loja 4 para loja 9: 2.10526 litros
- De loja 9 para loja 7: 4.44444 litros
- De loja 7 para loja 8: 10.05177 litros
- De loja 8 para loja 6: 8.31890 litros
- De loja 6 para loja 0: 29.99423 litros

Por fim, obtivemos um tempo de execução de 1.005s, um tempo muito menor se comparado ao algoritmo de força bruta.

Logo abaixo, criamos também um grafo dinâmico que ilustra o caminho percorrido pelo caminhão(seta vermelha) para a realização da melhor rota entre cada uma das lojas, mostrando também o consumo de combustível gasto em cada trecho da viagem.

```
Rota ótima: [0, 1, 2, 3, 5, 4, 9, 7, 8, 6, 0]
Combustível total gasto: 102.70 litros
Combustível gasto por trecho:
De loja 0 para loja 1: 15.79 litros
De loja 1 para loja 2: 11.76 litros
De loja 2 para loja 3: 6.48 litros
De loja 3 para loja 5: 11.60 litros
De loja 5 para loja 4: 11.10 litros
De loja 4 para loja 9: 2.11 litros
De loja 9 para loja 7: 4.44 litros
De loja 7 para loja 8: 10.05 litros
De loja 8 para loja 6: 8.32 litros
De loja 6 para loja 0: 29.99 litros
Tempo de execução: 1.005061388015747
```

Saida de dados do programa Branch and Bound



Grafo Força Bruta

4 CONCLUSÃO

4.1 Ambiente de Testes

Os experimentos foram realizados em um Desktop Dell, com i7, Windows 10 e 8gb de RAM.

5 CONSIDERAÇÕES FINAIS

Para concluir, nesse projeto foi desenvolvido com sucesso uma versão otimizada do código fornecido pelo professor, exercitando os conhecimentos obtidos em sala de aula, e ao mesmo tempo o uso de métodos de pesquisa e testes para chegar em um resultado final satisfatório.

A partir disso pudemos notar que o algoritmo de branch and bound é mais eficiente quando comparado com o de força bruta, não só em tempo de execução, que são, respectivamente, 1.005s e 3,75s. Além disso, a ordem de complexidade também é diferente sendo o de força bruta é O(N! * N) sendo N o numero de lojas, enquanto o branch and bound tem a complexidade de O(N!).