Pontifícia Universidade Católica de Minas Gerais

Bacharelado em Engenharia de Software – Unidade Praça da Liberdade

Fundamentos de Projeto e Análise de Algoritmos

Prof. João Caram - Semestre 2/2023

Trabalho prático em grupo - Valor: 20 pontos

Relatório Técnico Problemas Intratáveis - Distribuição de Rotas de Caminhões

Alunos: Ana Luiza Santos Gomes, Andressa Assunção Nunes, Bernardo Biondini

Introdução:

Este relatório apresenta uma análise técnica das implementações de diferentes estratégias para resolução do problema de distribuição de rotas entre caminhões, usando estratégias e algoritmos aprendidos em sala de aula, e também abordando decisões tomadas em cada um deles, com seu funcionamento e uma comparação de resultados em termos de tempo de execução e qualidade dos resultados obtidos.

Estratégias de Distribuição de Rotas:

a) Projetar e implementar uma solução para o problema apresentado utilizando ***backtracking***. A solução deve incluir uma estratégia de poda para soluções não promissoras

Decisões Tomadas:

No backtracking, decidimos por calcular a distância total ideal que cada caminhão deveria percorrer, se o total de km for 300 por exemplo, cada caminhão deveria percorrer 100km. Ainda, foi definido um valor de dispersão, dessa forma, um caminhão pode fazer entre 100-x e 100+x km, ou seja, se o valor de dispersão for 10, cada caminhão pode fazer entre 90 e 110km. Nesse sentido, o backtracking vai fazendo combinações entre as rotas de forma a atingir uma distância total no intervalo determinado.

Assim, cada **chamada recursiva** do algoritmo representa uma decisão sobre a inclusão, ou não, de uma rota em um conjunto. O algoritmo avalia a soma das rotas até o momento. Se a **soma das rotas é menor que o intervalo**, a rota é inserida no conjunto e o algoritmo executa para o próximo item, **se a soma está entre o intervalo**, adiciona a rota no conjunto e retorna, **se a soma está acima do limite**, o conjunto é desconsiderado. Esse passo é feito para cada item e seus seguintes. No final, teremos uma lista de possibilidades de rotas que estão entre o intervalo determinado. Então um segundo algoritmo seleciona uma combinação em que caminhões diferentes não possam fazer uma mesma rota. Desse modo, conseguiremos encontrar uma solução viável, dentro dos limites colocados, mas que pode não ser a melhor.

Funcionamento:

O Funcionamento do algoritmo implementado se dá pela seguinte forma:

- É calculado o valor médio que os caminhões devem percorrer
- O método backtrack é chamado pela primeira vez com o vetor de rotas, i e j que representam o item e o próximo, a soma que inicia com 0, a dispersão e duas listas. A primeira lista representa todas as possibilidades possíveis de rotas por caminhão e a segunda é o conjunto de rotas naquele momento.

- Verifica se a soma está abaixo da faixa (média - dispersão), se está acima de (média + dispersão) ou se está entre esses valores
- Se estiver abaixo a rota é adicionada no subconjunto e o algoritmo é executado para a próxima rota
- Caso esteja acima da faixa, aquele conjunto pode ser desconsiderado
- Caso esteja entre, é considerado um bom conjunto, então adicionamos na lista de opções e o algoritmo segue para as próximas execuções
- Por fim, um algoritmo refina o resultado para encontrar uma solução em que as rotas não se repitam

Resultados:

- Conjuntos de tamanho crescente, até atingir um tamanho T que não consiga ser resolvido em até 30 segundos pelo algoritmo.

Tempo de execução médio backtracking para 982 rotas: 18152.5

Tempo de execução máximo atingido no backtracking para 983 rotas

- Para o conjunto: 35, 34, 33, 23, 21, 32, 35, 19, 26, 42

Resultado backtracking:

[[(35,0), (34,1), (33,2)],

[(23,3), (32,5), (35,6)], [

(21,4), (19,7), (26,8), (42,9)]]

Tempo: 2ms

- Para o conjunto: 40,36,38,29,32,28,31,35,31,30,32,30,29,39,35,38,39,35,32,38,32,33,29,33,32,39,28

Resultado backtracking:

[[(40,0), (36,1), (38,2), (29,3), (32,4), (28,5), (31,6), (35,7), (31,8)],

[(30,9), (32,10), (30,11), (29,12), (39,13), (35,14), (38,15), (39,16), (35,17)],

[(32,18), (38,19), (32,20), (33,21), (29,22), (33,23), (32,24), (39,25), (28,26)]]

Tempo: 11ms

Considerações:

Observou-se que o tempo aumenta consideravelmente a medida que a quantidade de rotas também aumenta. Tornando-se mais "devagar" cada vez mais rápido, à medida que a quantidade de rotas aumenta.

b) Projetar e implementar soluções para o problema apresentado utilizando ***algoritmo guloso***. Neste caso, o grupo deve utilizar pelo menos duas estratégias gulosas diferentes na implementação, comparando seus resultados

Estratégia Gulosa 01:

Decisões Tomadas:

Nessa primeira estratégia, para a **distribuição das rotas** optamos pela menor distância acumulada, no array routes, entre os caminhões até o momento. Dessa forma, poderíamos pelo menos garantir que a menor distância seria alcançada.

Já para a **escolha dos caminhões**, optamos por fazer um loop que percorre os caminhões existentes, identificando o caminhão com a menor distância acumulada (minDistance). O índice desse caminhão (minIndex) é utilizado para adicionar a rota à lista correspondente.

Dessa forma, nossa estratégia gulosa 1 busca minimizar a diferença total entre as distâncias percorridas pelos caminhões, a escolha é feita atribuindo cada rota ao caminhão com a menor distância acumulada até o momento.

Funcionamento

O funcionamento dessa estratégia se dá pela seguinte forma;

- Inicializa o array distances para manter o somatório das distâncias percorridas por cada caminhão.
- Para cada rota no array routes, escolhe em qual caminhão atribuir a rota.
- Utiliza um loop para percorrer os caminhões existentes e identificar o caminhão com a menor distância acumulada até o momento.
- Adiciona a rota à lista do caminhão escolhido em result.
- Atualiza a distância do caminhão escolhido, adicionando a distância da rota recém-atribuída à distância acumulada.
- O resultado final é uma lista de listas (result) representando a distribuição otimizada das rotas nos caminhões.

Resultados

- Mesmos conjuntos de tamanho T utilizados no backtracking. EM seguida, aumente os tamanhos dos conjuntos de T em T até atingir o tamanho 10T

Resultados guloso1 para 982 rotas (Estratégia gulosa 1): 1.9

Resultados guloso1 para 1964 rotas (Estratégia gulosa 1): 1.0

Resultados guloso1 para 3928 rotas (Estratégia gulosa 1): 0.9

Resultados guloso1 para 7856 rotas (Estratégia gulosa 1): 0.7

- Para o conjunto: 35, 34, 33, 23, 21, 32, 35, 19, 26, 42

Resultado guloso1:

[[35, 32, 26], [34, 21, 35], [33, 23, 19, 42]]

Tempo: 1ms

- Para o conjunto: 40,36,38,29,32,28,31,35,31,30,32,30,29,39,35,38,39,35,32,38,32,33,29,33,32,39,28

Resultado guloso1:

[[40, 28, 35, 30, 39, 35, 38, 33, 28], [36, 29, 31, 30, 29, 38, 32, 33, 32], [38, 32, 31, 32, 35, 39, 32, 29, 39]]

Tempo: 1ms

Estratégia Gulosa 02:

Decisões tomadas:

Nessa segunda estratégia, optamos por usar tempos aleatórios para a decisão das escolhas das rotas e ordenar esses, para garantir que seria escolhido realmente a menor rota global. Assim, utilizamos a classe Comparator para ordenar a lista routeList com base no tempo associado a cada rota. Para cada uma na lista ordenada, escolhemos o caminhão com a menor distância acumulada até o momento, conforme definido pelo tempo.

Dessa forma, nossa segunda estratégia gulosa busca minimizar a distância total percorrida pelos caminhões, considerando os tempos associados a cada rota, ordenando-as com base no tempo e distribuindo cada rota ao caminhão com a menor distância acumulada até o momento.

Funcionamento

O funcionamento dessa estratégia se dá pela seguinte forma;

- Inicializa routeList, uma lista de objetos Route, cada um representando uma rota com um tempo associado.
- Utiliza a classe Comparator para ordenar routeList com base no tempo de cada rota.
- As rotas são ordenadas pelo tempo, priorizando as rotas mais rápidas.
- Para cada rota ordenada, determina em qual caminhão atribuir a rota.
- Itera sobre os caminhões existentes, escolhendo o caminhão com a menor distância acumulada até o momento, levando em consideração os tempos das rotas.
- Adiciona a rota ao caminhão escolhido em result.

Resultados

- Mesmos conjuntos de tamanho T utilizados no backtracking. EM seguida, aumente os tamanhos dos conjuntos de T em T até atingir o tamanho 10T

Resultados guloso2 para 982 rotas (Estratégia gulosa 2): 10.9

Resultados guloso2 para 1964 rotas (Estratégia gulosa 2): 7.5

Resultados guloso2 para 3928 rotas (Estratégia gulosa 2): 22.3

Resultados guloso2 para 7856 rotas (Estratégia gulosa 2): 50.2

- Para o conjunto: 35, 34, 33, 23, 21, 32, 35, 19, 26, 42

Resultado guloso2:

[[35, 32, 26], [34, 21, 35], [33, 23, 19, 42]]

Tempo: 0ms

- Para o conjunto: 40,36,38,29,32,28,31,35,31,30,32,30,29,39,35,38,39,35,32,38,32,33,29,33,32,39,28=

Resultado guloso2:

[[40, 28, 35, 30, 39, 35, 38, 33, 28], [36, 29, 31, 30, 29, 38, 32, 33, 32], [38, 32, 31, 32, 35, 39, 32, 29, 39]]

Tempo: 0ms

c) Projetar e implementar uma solução para o problema apresentado utilizando ***divisão e conquista***. O grupo deve decidir se vai utilizar o método demonstrado em aula ou outro à escolha.

Decisões Tomadas:

Para a divisão e conquista, optamos pelo método de ordenação **MergeSort** pois ele é eficiente na ordenação de arrays, e garante que as rotas estejam organizadas em ordem crescente ou decrescente com base em seus comprimentos. Dessa forma, a ordenação é crucial para facilitar a distribuição eficiente das rotas entre os caminhões, especialmente quando o objetivo é diminuir a diferença nas quilometragens.

Outra razão por temos escolhidos o merge é porque é um algoritmo estável, o que significa que ele preserva a ordem de elementos com chaves iguais. Isso é vantajoso quando há rotas com a mesma quilometragem, garantindo consistência nas decisões de distribuição. Sendo assim, o MergeSort foi eficaz para organizar as rotas, permitindo uma distribuição mais justa entre os caminhões e, assim, otimizando a quilometragem total percorrida pela frota.

Funcionamento

- O array de rotas é ordenado usando o Merge Sort, resultando em uma sequência ordenada
- As rotas ordenadas são então divididas em grupos, dependendo do número de caminhões disponíveis
- A distribuição das rotas entre os caminhões é realizada com base na ordenação, visando uma distribuição igual ou minimização da diferença nas quilometragens.
- O resultado é uma distribuição otimizada das rotas entre os caminhões, atendendo aos critérios de equidade ou minimização da diferença nas quilometragens.

Resultados

- Utilize os mesmos conjuntos de tamanho T utilizados no backtracking.

- Para o conjunto: 35, 34, 33, 23, 21, 32, 35, 19, 26, 42

Resultado Divisão e Conquista: 2ms

Caminhão 1: [23, 33, 34, 35]

Caminhão 2: [21, 32, 35]

Caminhão 3: [19, 26, 42]

- Para o conjunto: 40, 36, 38, 29, 32, 28, 31, 35, 31, 30, 32, 30, 29, 39, 35, 38, 39, 35, 32, 38, 32, 33, 29, 33, 32, 39, 28

Resultado Divisão e Conquista: 1ms

Caminhão 1: [28, 29, 31, 31, 32, 35, 36, 38, 40]

Caminhão 2: [29, 30, 30, 32, 35, 35, 38, 39, 39]

Caminhão 3: [28, 29, 32, 32, 32, 33, 33, 38, 39]

d) Projetar e implementar uma solução para o problema apresentado utilizando programação
dinâmica. O grupo deve decidir se vai utilizar o método demonstrado em aula ou outro à escolha.
d1) Aqui, utilize os mesmos conjuntos de teste do algoritmo guloso.

Decisões Tomadas:

Para a Programação Dinâmica optamos por criar duas matrizes: dp (para armazenar a soma das distâncias) e choices (para as escolhas feitas). As matrizes são preenchidas dinamicamente com base na busca pela melhor divisão para otimizar a distribuição de rotas. O preenchimento dinâmico foi feito da seguinte maneira; O loop com a var k percorre os índices das rotas de 1 até i – 1, cada valor de k representa um ponto possível de divisão entre as rotas, para cada valor de k é calculado um custo que representa a melhor divisão entre as rotas e caminhões.

Exemplo essas rotas: {10, 20, 30, 40} para serem distribuídas entre 2 caminhões

A var k passará por cada uma das rotas calculando qual a melhor forma de distribuir elas por caminhão

Iteração 01:

Divisão 1: Atribuir 20 ao Caminhão 1 e 10 ao Caminhão 2.

Diferença total: |20 - 10| = 10 (distância Caminhão 1 - distância Caminhão 2).

Divisão 2: Atribuir 10 ao Caminhão 1 e 20 ao Caminhão 2.

Diferença total: |10 - 20| = 10 (distância percorrida pelo Caminhão 1 - distância percorrida pelo Caminhão 2).

E assim sucessivamente até achar a combinação da rota que têm a menor diferença de distância entre os caminhões. Dessa forma, A variável bestIndex é atualizada para armazenar o índice da melhor combinação encontrada no loop. Assim, esse resultado é usado para reconstruir as rotas atribuidas a cada caminhão.

Funcionamento

- Duas matrizes são inicializadas, dp e choices, para armazenar informações sobre as soluções parciais e as escolhas feitas durante o processo.
- A matriz dp é preenchida com os valores iniciais. A ideia é calcular a soma acumulativa das distâncias para cada possível subconjunto de rotas.
- O algoritmo preenche dinamicamente as matrizes dp e choices para encontrar a melhor distribuição de rotas. Ele usa um loop duplo para percorrer as matrizes.
- Reconstrução das Rotas Atribuídas aos Caminhões: O algoritmo reconstrói as rotas atribuídas aos caminhões usando as informações armazenadas na matriz choices
- Retorno do Resultado: O método retorna a lista de rotas atribuídas aos caminhões.

Resultados

Resultado Programação Dinâmica:

- Conjuntos de tamanho crescente, até atingir um tamanho T que não consiga ser resolvido em até 30 segundos pelo algoritmo.

Tempo de execução máximo atingido na programação dinâmica para o tamanho 658
Média de Tempo: 31.6 ms

Resultado Programação Dinâmica:

- Para o conjunto: 40,36,38,29,32,28,31,35,31,30,32,30,29,39,35,38,39,35,32,38,32,33,29,33,29,39,28

Caminhão 1: rotas 32 38 32 33 29 33 29 39 28 - total 293km

Caminhão 2: rotas 30 32 30 29 39 35 38 39 35 - total 307km

Caminhão 3: rotas 40 36 38 29 32 28 31 35 31 - total 300km

- Para o conjunto: 32,51,32,43,42,30,42,51,43,51,29,25,27,32,29,55,43,29,32,44,55,29,53,30,24,27

Caminhão 1: rotas 29 32 44 55 29 53 30 24 27 - total 323km

Caminhão 2: rotas 43 51 29 25 27 32 29 55 43 - total 334km

Caminhão 3: rotas 32 51 32 43 42 30 42 51 - total 323km