Relatório Técnico sobre o Trabalho Prático, problemas intratáveis

Henrique Almeida, Gabriel Dolabela, João Pedro, Lucas Lima

Pontifícia Universidade Católica de Minas Gerais

1. Introdução

Este relatório descreve as implementações e análises de diferentes algoritmos para a distribuição de rotas igualitariamente entre caminhões.

2. Sobre o problema

O problema consiste em distribuir rotas entre caminhões de forma que a quilometragem percorrida por cada caminhão seja a mais próxima possível.

O problema é NP-difícil, portanto, não há algoritmo polinomial que resolva o problema. Assim, foram implementadas estratégias de backtracking, guloso, divisão e conquista e programação dinâmica.

3. Sobre a implementação

3.1. Backtracking

A estratégia de backtracking foi implementada de forma recursiva, onde a cada chamada recursiva, um novo nó é adicionado a solução, e a partir dele, novas chamadas recursivas são feitas, até que a solução seja encontrada ou que não haja mais nós a serem adicionados.

Sobre a poda, o código verifica se a distribuição atual é melhor do que a melhor conhecida até o momento, se não for o caso, o algoritmo não continua a busca nessa direção.

3.2. Guloso

3.2.1. Guloso com Ordem das Rotas

A implementação da estratégia gulosa baseada na ordem das rotas segue uma abordagem simples. As rotas são ordenadas em ordem crescente de quilometragem e, em seguida, distribuídas sequencialmente para os caminhões. A ordem de distribuição é invertida a cada ciclo, garantindo uma distribuição mais equitativa do que originalmente.

3.2.2. Guloso com Quilometragem Acumulada

A segunda estratégia gulosa utiliza a quilometragem acumulada como critério para a distribuição de rotas. As rotas são distribuídas para os caminhões com base na menor quilometragem acumulada.

É utilizada uma fila de prioridade para a implementação dessa estratégia. A cada iteração, o caminhão com menor quilometragem acumulada é retirado da fila e a rota é distribuída para ele. Em seguida, o caminhão é inserido novamente na fila, com a quilometragem acumulada atualizada.

A classe interna Caminhao auxilia o uso da fila de prioridade implementando a interface Comparable, que permite a comparação entre os caminhões com base na quilometragem acumulada.

3.3. Divisão e Conquista

Na Divisão e Conquista, o algoritmo verifica se é possível incluir ou não a rota para manter a distribuição equitativa.

Esta verificação é feita através de um valor máximo e mínimo passados como parâmetro. Caso a inclusão da rota não ultrapasse o valor máximo, o item é adicionado a solução e a próxima chamada recursiva é feita com o valore da rota recém adicionada subtraido de máximo e mínimo.

Os valores são calculados com base na média de quilometragem das rotas pelo número de caminhões, o máximo é 10% maior que a média e o mínimo é 10% menor que a média.

O caso base se torna a validação de se a soma das rotas está entre o máximo e o mínimo.

Caso no final haja uma rota sem alocação, o algoritmo usa o método guloso com quilometragem acumulada para alocar a rota restante.

3.4. Programação Dinâmica

Na implementação da programação dinâmica, foi utilizado um outro problema semelhante para auxiliar na resolução das rotas. Através da soma de subconjuntos, é possível encontrar a aproximação de um conjunto de números que seja igual a um valor desejado. Nesse caso, o valor desejado é 10% da média de quilometragem das rotas pelo número de caminhões.

Na matriz de programação dinâmica, cada linha representa uma rota, cada coluna representa um valor de 0 ao valor desejado, e cada célula representa a soma de subconjuntos para a rota e o valor da coluna.

Assim, mesmo que a célula [i][j] termine com o valor 0, é possível decrementar o valor de j até que o valor da célula seja 1, e assim encontrar uma aproximação para o valor desejado.

Então, o algoritmo calcula a as rotas incluídas na solução, as alocando para um caminhão, e as retira da lista de rotas. Em seguida, o algoritmo recalcula a matriz de programação dinâmica, e repete o processo até que todas as rotas sejam alocadas.

Como no caso da divisão e conquista, o algoritmo usa o método guloso com quilometragem acumulada para alocar a rota restante caso não seja possível alocar todas as rotas.

4. Análise das execuções

4.1. Backtracking

O algoritmo de backtracking foi capaz de resolver instâncias de até 17 rotas em menos de 30 segundos.

Nota-se um crescimento exponencial $O(3^n)$ no tempo de execução do algoritmo, o que é esperado para um algoritmo de backtracking.

Tamanho	Tempo (ms)
6	1
7	1
8	1
9	3
10	9
11	14
12	38
13	101
14	299
15	904
16	3123
17	9192

4.2. Guloso

Em ambas as implementações, o algoritmo guloso foi capaz de resolver instâncias de até 10T em menos de 1 ms. Entretanto, a implementação baseada na ordem das rotas se mostrou inferior à implementação baseada na quilometragem acumulada no quesito de soluções.

Tamanho	Tempo (ms)
18	0
36	0
54	0
72	0
90	0
108	0
126	0
144	0
162	0
180	0
198	0

4.3. Divisão e Conquista

Tamanho	Tempo (ms)
18	1

4.4. Programação Dinâmica

O algoritmo de programação dinâmica foi capaz de resolver as instâncias de 10T em menos de 2 ms.

Tamanho	Tempo (ms)
18	0
36	1
54	1
72	1
90	1
108	1
126	1
144	2
162	1
180	0
198	0

5. Conclusão

O algoritmo de Backtracking demonstrou eficácia para instâncias de até 17 rotas em menos de 30 segundos. No entanto, a complexidade exponencial $O(3^n)$ no tempo de execução, conforme observado, mostra sua limitação para instâncias maiores.

O Guloso, embora rápido, apresenta soluções não ótimas, principalmente a implementação de ordem das rotas. Resolvendo instâncias de até 10T em menos de 1 ms, sua eficiência é notável.

A Divisão e Conquista demonstrou bom desempenho. Essa abordagem pode ser uma escolha equilibrada, oferecendo eficiência sem comprometer a otimalidade.

A Programação Dinâmica se destacou ao resolver instâncias de 10T em tempo mínimo e com as melhores soluções. Como que o tempo passa de 0 -> 1 -> 2 -> 1 -> 0 é um grande mistério.