Relatório - Etapa 2

Laura Silva Lopes e Leon Augusto Okida Gonçalves

Abril 2022

1 Introdução

O problema a ser resolvido consiste em uma série de cargas que devem ser transportadas por uma empresa em seu único caminhão. Cada carga tem seu respectivo peso, assim como o caminhão. A soma dos pesos das cargas em cada viagem não pode exceder o peso do caminhão e, por isso, provavelmente terão que ser feitas múltiplas viagens. Além disso, podem existir restrições de ordem entre as entregas de cada carga. Temos como objetivo calcular como fazer o menor número K de viagens possível.
Na primeira etapa, fizemos uma modelagem do problema para uma Programação Linear Relaxada, que nos dá uma solução aproximada do problema, sendo um número menor ou igual à solução inteira real.
Para a segunda etapa do trabalho, utilizamos essa modelagem para obter soluções parciais de subproblemas e usá-las para resolver o problema com a técnica de *branch and bound*.

2 Modelagem da solução parcial

A modelagem elaborada foi:

$$\min \sum_{i=1}^{n} K_i \tag{1}$$

$$\text{s.a.} \sum_{j=1}^{n} w_j x_{ij} \leq CK_i, \forall i = 1, ..., n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \forall j = 1, ..., n \tag{3}$$

$$\sum_{i=1}^{n} i x_{ia} < \sum_{i=1}^{n} i x_{ib}, \forall (a, b) \tag{4}$$

$$K_i \geq K_{i+1}, \forall i = 1, ..., n-1 \tag{5}$$

$$0 \leq K_i \leq 1, \forall i = 1, ..., n \tag{6}$$

$$0 \leq x_{ij} \leq 1, \forall i, j = 1, ..., n \tag{7}$$

$$v(y) x_{v(y)y} = v(y), \forall y \in P \tag{8}$$

Onde:

n	:	número de cargas
i	:	índice de viagem
j	:	índice de carga
K_i	:	se viagem i ocorre
w_j	:	peso da carga j
x_{ij}	:	se carga j é transportada na viagem i
C	:	limite de peso máximo
P	:	conjunto de soluções parciais
y	:	índice de carga no conjunto das soluções parciais P
$v(y)$	:	índice de viagem da carga y, $y \in P$

3 Explicação da modelagem da solução parcial

A função objetiva em (1) busca minimizar o número K de viagens feitas, que no pior caso é n (uma viagem para transportar cada carga).
A restrição em (2) faz com que a soma dos pesos de cada carga j que é transportada na viagem i (se ela ocorre) não passe do limite C.
A restrição em (3) faz com que cada carga j seja transportada uma única vez.
A restrição em (4) faz com que a viagem i da carga a tenha índice menor que a da carga b, o que faz com que ela seja transportada antes, em cada restrição de ordem (a, b).
A restrição em (5) faz com que uma viagem i ocorra apenas quando uma viagem de índice $i - 1$ já tenha ocorrido antes.
A restrição em (6) serve para determinar que a variável K_i seja binária, representando se a viagem i ocorre ou não. Na modelagem elaborada, essa restrição foi relaxada.
A restrição em (7) serve para determinar que a variável x_{ij} seja binária, representando se a carga j é transportada na viagem i ou não. Na modelagem elaborada, essa restrição foi relaxada.
A restrição em (8) serve para fixar uma solução parcial já encontrada, fazendo com que uma carga de índice $y \in P$ seja transportada na viagem $v(y)$.

4 Implementação

Para solucionar o problema, foi desenvolvido um programa em C, com o uso da biblioteca LP Solve.
O programa principal `envio` lê os dados de entrada e ordena as cargas a serem transportadas de acordo com as restrições de ordem, isto é, as cargas que devem vir antes de outras são processadas primeiro.
Se as restrições não formarem um ciclo, é chamada a função `resolucao` que retornará o número inteiro mínimo de viagens necessárias para transportar todas as cargas.
Para cada carga j e viagem i, é verificado se, transportando a carga na viagem, o peso do caminhão não ultrapassa a capacidade máxima e se não infringe alguma restrição de ordem de entrega. Sendo possível o transporte, é calculada a função limitante, descrita na Seção 4.1, fixando a carga j à viagem i.
Se o valor retornado pela função limitante for melhor que a solução ótima até então encontrada, então a função `resolucao` é chamada recursivamente para as demais cargas. No momento em que todas as cargas são designadas a uma viagem, e, caso o número de viagens realizadas for menor que a solução ótima, então a variável `opt`, que representa a solução ótima e que é usada de limitante para as demais ramificações do algoritmo, é atualizada, recebendo o número de viagens realizadas.
Por fim, o programa principal imprime o valor da solução ótima e a viagem em que cada carga foi transportada.

4.1 Função limitante

A função `parcial` recebe como parâmetro todos os dados de entrada e o conjunto das soluções parciais P. Essa função chama `escreveModelo`, que imprime a modelagem descrita no Capítulo 2 no arquivo `entrada.lp`. Após isso, a biblioteca LP Solve é usada para resolver o modelo e obter a solução aproximada, que é retornada no fim da função. Essa função foi desenvolvida na primeira etapa do trabalho.
Notamos que a solução aproximada obtida é sempre $(\sum_{j=1}^{n} w_j)/C$, ou seja, a soma dos pesos w de todas as cargas j dividido pela capacidade máxima C do caminhão. Por isso, utilizamos este cálculo como função limitante no algoritmo principal, ao invés de usarmos a solução parcial relaxada.
Como para a segunda abordagem não há leitura e escrita em arquivos, naturalmente a solução, para problemas pequenos o suficiente, é encontrada mais rapidamente do que utilizando a função `parcial`. Entretanto, para os testes realizados, os dois métodos retornam o mesmo resultado.
Optamos pela função $(\sum_{j=1}^{n} w_j)/C$ como limitante, conforme combinado em sala de aula.

5 Execução

Para o programa funcionar adequadamente, é necessário configurar a variável de ambiente `LD_LIBRARY_PATH`, atribuindo a ela o endereço do diretório local. Assim, o programa consegue carregar a biblioteca LP Solve.

```
$ LD\_LIBRARY\_PATH=. ./envio
```

Considere a entrada abaixo:

```
5 2 10
5 6 4 8 5
2 3
5 1
```

Na primeira linha, são informados o número de cargas, de restrições de ordem e o peso máximo suportado.
Na segunda linha, são informados os pesos das cargas.
Nas terceira e quarta linhas, são informados os pares de restrições, seguindo a ordem (a, b).
As entradas suportadas pelo programa seguem esse padrão.

A saída esperada para esse exemplo é:

```
4
3
1
2
4
2
```