UNIVERSIDADE FEDERAL DO PARANÁ

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Luan Carlos Maia Cruz (GRR20203891)

Leonardo Marin Mendes Martin (GRR20205675)

MODELAGEM E IMPLEMENTAÇÃO PARA O PROBLEMA DO TRANSPORTE DE CARGA COM PACOTES DE RECURSOS

1. RESUMO

Neste relatório, apresentamos a modelagem e implementação, utilizando a linguagem C, para resolver o problema do transporte de carga com pacotes de recursos. Para a resolução, empregamos técnicas de programação linear.

2. INTRODUÇÃO

O principal objetivo do projeto foi implementar a modelagem do problema de transporte de carga com pacotes de recursos, originando um executável denominado "transporte". Ao ser executado, este programa gera uma entrada para o lp_solve, um software especializado em resolver problemas lineares. O projeto teve como referência os ensinamentos passados em sala de aula, o livro “Understanding and Using Linear Programming” de Jiří Matoušek e Bernd Gärtner - 2007, e foi desenvolvido em linguagem C, incluindo um makefile e os diretórios para os arquivos de código-fonte e exemplos de uso.

3. CONTEXTO DO PROBLEMA

Uma empresa deseja enviar a maior quantidade possível de seu produto de uma fábrica (cidade origem) para um depósito (cidade destino) em outra cidade, visando obter lucro através da venda do produto. O lucro é calculado como o ganho fixo da venda do produto por tonelada transportada, menos o custo de produção (não considerado na implementação) e o custo dos pacotes de recursos necessários para o transporte. A empresa possui uma rede de transporte com capacidades diárias limitadas em cada rota e exigências mínimas de recursos por tonelada transportada em cada rota. Os recursos necessários são adquiridos em pacotes, cada um com uma quantidade específica de recursos e um custo associado. O objetivo da empresa é maximizar seu lucro utilizando a rede de transporte disponível, adquirindo somente os pacotes de recursos necessários.

4. MODELAGEM DO PROBLEMA

VARIÁVEIS DE DECISÃO

$x_{i,j}$:= quantidade de toneladas enviadas da cidade i (origem) para a cidade j (destino)

y_u := quantidade de pacotes do tipo u

VARIÁVEIS DO PROBLEMA

p := preço por tonelada transportada

n := quantidade de cidades

m := quantidade de rotas

e := par (i,j) que representa cada rota válida na rede de transportes

k := quantidade de recursos

q := tipos de pacotes para compra de recursos

$c_{i,j}$:= capacidade máxima diária para a rota e = (i,j)

v_u := custo do pacote do tipo u

FUNÇÃO OBJETIVO

$$p\left(\sum_{j=2}^{m} x_{1,j}\right) - \sum_{u=1}^{q} v_u y_u$$

Onde:

p := preço por tonelada transportada

m := quantidade de rotas

q := quantidade de tipos de pacote

$x_{1,j}$:= quantidade de carga para as rotas e = (1,j) que saem da origem (fábrica), sendo "e" uma rota válida na rede de transportes.

y_u := quantidade de pacotes do tipo u comprados

v_u := custo do pacote do tipo u comprado

A função objetivo foi apresentada junto com o problema, mas mesmo sem essa informação explícita, ela seria relativamente intuitiva de deduzir.

Consideramos apenas as rotas que têm origem em 1, pois são sempre as rotas iniciais, não podem assumir um sentido contrário ($x_{1,j} < 0$) e indicam a quantidade de carga enviada pela fábrica. As demais rotas são utilizadas para determinar o melhor caminho a ser tomado para obter o maior lucro.

RESTRIÇÕES

As restrições também foram fornecidas na apresentação do problema, fizemos a seguinte divisão:

- Restrição de Capacidade
- Restrição de Fluxo
- Restrição de Recursos
- Restrição de Não-negatividade

A seguir, detalharemos cada uma delas:

1. **Restrição de capacidade**: A quantidade de toneladas enviadas por uma rota não pode exceder a capacidade máxima diária da rota

$$-c_{i,j} \leq x_{i,j} \leq c_{i,j}$$

Sendo i,j um par e = (i,j), válido na rede de transportes

Onde:

m := quantidade de rotas

$c_{i,j}$:= capacidade máxima diária de carga da rota e = (i,j), sendo "e" uma rota válida na rede de transportes.

$x_{i,j}$:= quantidade de carga de cada rota e = (i,j)

A variável $t_{i,j}$ está limitada superiormente e inferiormente por $c_{i,j}$, pois pode assumir o sentido contrário em uma rota, ou seja $x_{j,i} = - x_{i,j}$

2. **Restrição de Fluxo**: A soma da quantidade de carga que sai de uma rota com destino j, deve ser igual a soma da quantidade de carga recebida em uma rota com origem j

$$\sum_{i=1}^{n} x_{i,j} = \sum_{i=1}^{n} x_{j,i}$$

Sendo i,j um par e = (i,j), válido na rede de transportes

Onde:

n := quantidade de cidades

$x_{i,j}$:= quantidade de carga de cada rota e = (i,j)

Essa restrição considera o sentido da rota, como explicado na restrição anterior, garantindo que a quantidade de carga enviada da fábrica chegue ao depósito com a mesma quantidade.

3.**Restrição de recursos**: A empresa deve possuir uma quantidade mínima de cada recurso, para cada tonelada transportada através de uma rota.

$$\sum_{i=1}^{m} r_{s:i,j} \left| x_{i,j} \right| \leq \sum_{u=1}^{q} r_{s:u}\, y_u$$

Sendo i,j um par e = (i,j), válido na rede de transportes

s ∈ {1, … , k} e sendo k, a quantidade total de recursos

Onde:

m := quantidade de rotas

q := quantidade de tipos de pacotes

$r_{s:i,j}$:= quantidade de recurso s, exigida para a rota

$r_{s:u}$:= quantidade de recurso s, exigida para pacote

$x_{i,j}$:= quantidade de carga de cada rota e = (i,j)

y_u := quantidade de pacotes do tipo u comprados

Para lidar com o fato de que a variável $x_{i,j}$ pode assumir valores negativos, não podemos simplesmente não contabilizar o recurso r_s de uma rota $x_{i,j}$ no somatório, devemos utilizar o módulo para considerar apenas valores positivos. No entanto, ao introduzir o módulo, o problema deixa de ser linear. Para contornar isso, podemos usar um truque: inserir uma variável extra **não-negativa t** (t >= 0) extra que sempre assume valores positivos. Para garantir esse comportamento, inserimos a seguinte restrição no problema linear:

$$t_{i,j} \geq x_{i,j} \;\; e \;\; t_{i,j} \geq -\, x_{i,j}$$

Assim, a restrição atualizada fica:

$$\sum_{i=1}^{m} r_{s:i,j}\, t_{i,j} \leq \sum_{u=1}^{q} r_{s:u}\, y_u$$

Sendo i,j um par e = (i,j), válido na rede de transportes

s ∈ (1, … , k) e sendo k, a quantidade total de recursos

$t_{i,j}$:= quantidade de carga de cada rota e = (i,j), sendo $t_{i,j}$ não-negativa

4.**Restrição de não-negatividade**: As variáveis que representam o número de pacotes comprados (y_u) e quantidade de carga em módulo ($t_{i,j}$) devem ser não-negativas (>=0)

$$y_u \geq 0$$

$$t_{i,j} \geq 0$$

Sendo i,j um par e = (i,j), válido na rede de transportes

Para cada rota e = (i,j), sendo e uma rota válida na rede de transportes, deve ser criada a variável $t_{i,j}$, não-negativa
Para cada pacote u deve ser criada a variável y_u , não-negativa

5. IMPLEMENTAÇÃO

Como mencionado anteriormente, a implementação foi feita em linguagem C, composta por apenas três arquivos fontes que formam o projeto:

- transporte.c: Implementação do programa principal.
- libAux.h: Cabeçalho para as funções principais.
- libAux.c: Implementação das funções declaradas em libAux.h.

O projeto conta com duas funções principais:

1. leDados: Responsável pela leitura dos dados necessários para a modelagem.
2. produzModelagemLP: Encarregada de gerar a modelagem com base nos dados fornecidos.

Ao executar "*./transporte*", a função "*leDados*" é inicializada e espera como entrada um conjunto de dados que representam a quantidade de cidades, rotas, recursos, pacotes e o valor de ganho por tonelada transportada. Além disso, para cada rota, devem ser fornecidos detalhes como origem, destino, capacidade e recursos exigidos para o transporte pela rota. Da mesma forma, para cada pacote, é necessário informar os recursos exigidos e o custo do pacote.

Após a leitura de todos os dados necessários, a função "*produzModelagemLP*" gera a modelagem que será utilizada como entrada para o "*lp_solve*". A saída padrão (STDOUT) exibe essa modelagem e ela também pode ser salva em um arquivo para análises futuras.

6. COMO EXECUTAR ?

Os seguintes passos devem ser seguidos para executar o projeto

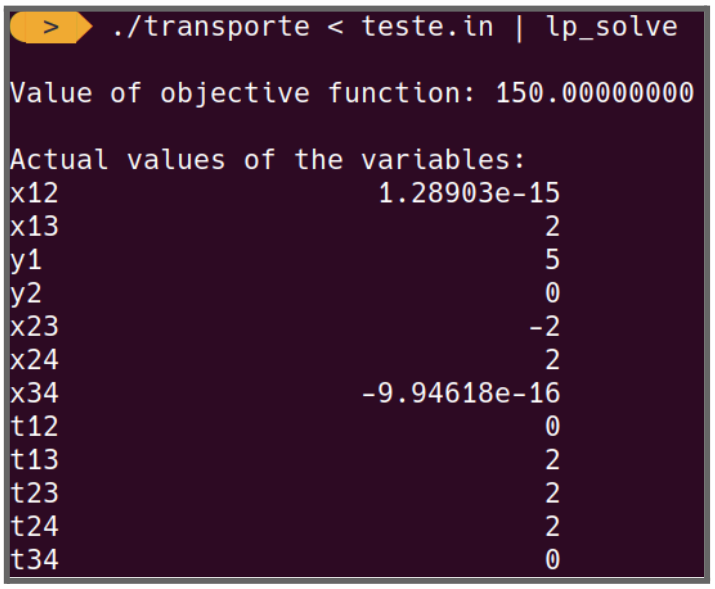
1. Executar o comando "*make*" dentro do diretório do projeto
2. Existem diferentes opções para a entrada do executável:
    a. **Saída no terminal com arquivo de entrada:**
    Executar o comando: ./transporte < dadosEntrada.txt | lp_solve
    b. **Saída no terminal com entrada pelo teclado:**
    Executar o comando: ./transporte | lp_solve
    c. **Saída no terminal e salvando em um arquivo :**
    Utilizar a flag "-f" e o nome de arquivo desejado, após o executável ./transporte para salvar a modelagem em um arquivo
    Executar os comandos:
    ./transporte -f nomeArquivo.lp < dadosEntrada.txt | lp_solve

    ./transporte -f nomeArquivo.lp | lp_solve

Como a modelagem sempre é enviada para STDOUT, pode usar o operador "|" para poder executar o lp_solve após a modelagem ser gerada

Reprodução dos tipos de execução :

a) ./transporte < dadosEntrada.txt | lp_solve



b) ./transporte | lp_solve

```
> ./transporte | lp_solve
4 5 3 2 100
1 2 5 1 1 5
1 3 2 2 2 0
2 3 5 1 2 0
2 4 2 2 1 0
3 4 5 3 4 6
10 4 2 0
20 5 2 1

Value of objective function: 150.00000000

Actual values of the variables:
x12                    1.28903e-15
x13                              2
y1                               5
y2                               0
x23                             -2
x24                              2
x34                   -9.94618e-16
t12                              0
t13                              2
t23                              2
t24                              2
t34                              0
```

c) ./transporte -f nomeArquivo.lp < dadosEntrada.txt | lp_solve

```
> ./transporte -f saida.out < teste.in | lp_solve

Value of objective function: 150.00000000

Actual values of the variables:
x12                    1.28903e-15
x13                              2
y1                               5
y2                               0
x23                             -2
x24                              2
x34                   -9.94618e-16
t12                              0
t13                              2
t23                              2
t24                              2
t34                              0
```

```
> cat saida.out
max: 100x12 + 100x13 - 10y1 - 20y2;

- 5 <= x12 <= 5;
- 2 <= x13 <= 2;
- 5 <= x23 <= 5;
- 2 <= x24 <= 2;
- 5 <= x34 <= 5;

x12 = x23 + x24 + 0;
x13 + x23 = x34 + 0;

t12 >= x12;
t12 >= -x12;
t13 >= x13;
t13 >= -x13;
t23 >= x23;
t23 >= -x23;
t24 >= x24;
t24 >= -x24;
t34 >= x34;
t34 >= -x34;

1t12 + 2t13 + 1t23 + 2t24 + 3t34 <= 4y1 + 5y2;
1t12 + 2t13 + 2t23 + 1t24 + 4t34 <= 2y1 + 2y2;
5t12 + 0t13 + 0t23 + 0t24 + 6t34 <= 0y1 + 1y2;

t12 >= 0;
t13 >= 0;
t23 >= 0;
t24 >= 0;
t34 >= 0;
y1 >= 0;
y2 >= 0;
```

7. EXEMPLOS DE TESTE

Para os exemplos de teste, consideramos o seguinte cenário:

4 cidades

4 rotas

2 recursos

1 tipo de pacote

O valor de ganho por carga transportada será diferente em cada situação, com os seguintes valores:

15 unidades monetárias

5 unidades monetárias

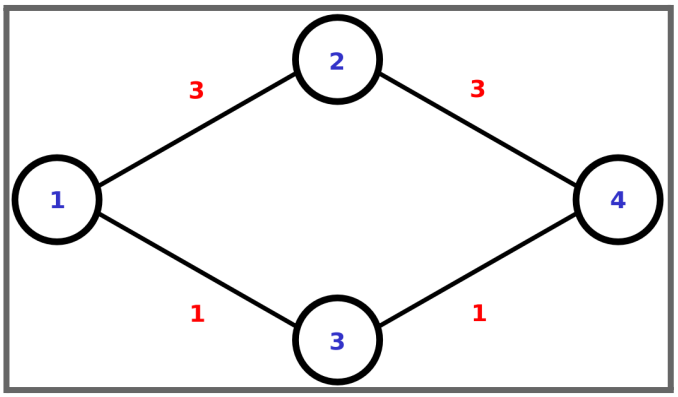
10 unidades monetárias

Os recursos exigidos pelas rotas e pelo pacote são os mesmos:

Recurso 1: 1 unidade

Recurso 2: 1 unidade

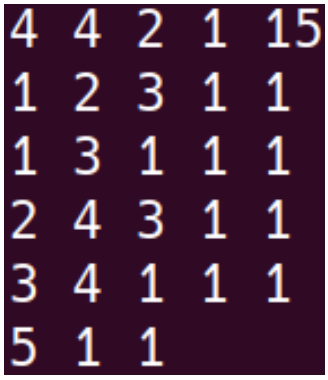
A seguir, uma representação da rede de transportes e da modelagem para os dados fornecidos do cenário.

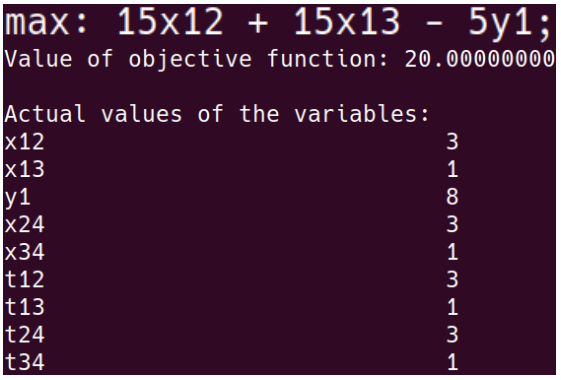



```
- 3 <= x12 <= 3;
- 1 <= x13 <= 1;
- 3 <= x24 <= 3;
- 1 <= x34 <= 1;

x12 = x24 + 0;
x13 = x34 + 0;

t12 >= x12;
t12 >= -x12;
t13 >= x13;
t13 >= -x13;
t24 >= x24;
t24 >= -x24;
t34 >= x34;
t34 >= -x34;

1t12 + 1t13 + 1t24 + 1t34 <= 1y1;
1t12 + 1t13 + 1t24 + 1t34 <= 1y1;

t12 >= 0;
t13 >= 0;
t24 >= 0;
t34 >= 0;
y1 >= 0;
```

Embora seja um exemplo básico, ele ilustra várias situações que podem surgir com a função objetivo. A seguir, apresentamos três cenários utilizando o cenário acima.

1. **Caso em que a fábrica possui lucro - Ganho por quantidade transportada é maior que o custo com pacotes**

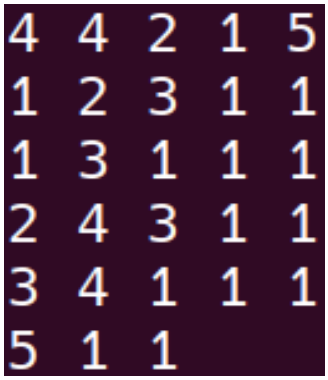


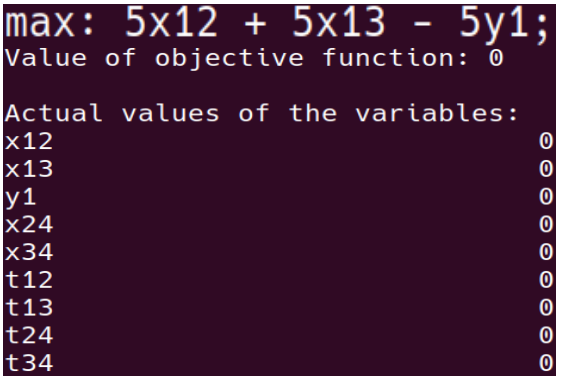


A carga transportada é de 4 toneladas, o ganho por tonelada transportada é de 15.
A quantidade de pacotes de recursos comprados é de 8, cada pacote tem custo de 5.
4 x 15 - 8 x 5 = 20,00 - Lucro de 20 unidades monetárias.

Como sabemos o número máximo de pacotes que podemos comprar, podemos alterar o valor de ganho por tonelada transportada para produzir as próximas duas situações.

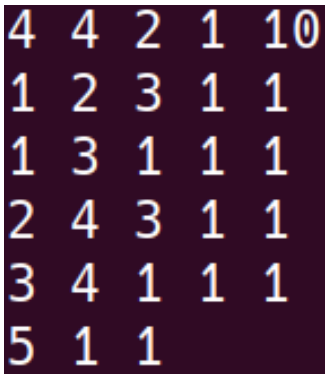
2. **Caso em que a fábrica possui prejuízo - Ganho por quantidade transportada é menor que o custo com pacotes**

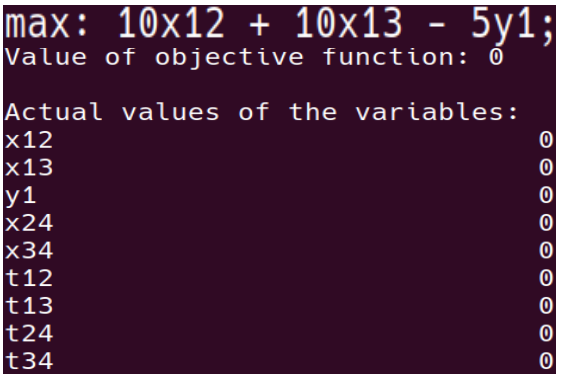




Sabemos através do exemplo 1 que a carga máxima que pode ser enviada é de 4 toneladas, então:
A carga transportada é de 4 toneladas, o ganho por tonelada transportada é de 5.
A quantidade de pacotes de recursos comprados é de 8, cada pacote tem custo de 5.
4 x 5 - 8 x 5 = -20,00 - Prejuízo de 20 unidades monetárias.

3. **Caso em que a fábrica não possui lucro nem prejuízo - Ganho por quantidade transportada é igual que o custo com pacotes**





Sabemos através do exemplo 1 que a carga máxima que pode ser enviada é de 4 toneladas, então:
A carga transportada é de 4 toneladas, o ganho por tonelada transportada é de 10.
A quantidade de pacotes de recursos comprados é de 8, cada pacote tem custo de 5.
4 x 10 - 8 x 5 = 0,00 - Sem ganho ou perda.

8. REFERÊNCIAS

- Livro “Understanding and Using Linear Programming”, Jiří Matoušek, Bernd Gärtner - 2007.