CI1238 - Otimização

Transporte de carga com pacotes de recursos

Universidade Federal do Paraná - UFPR

Felipe Quaresma Vieira

1. Função objetivo

Sabemos que toda a quantidade de tonelada transportada da cidade de origem até o destino passa pelas rotas iniciais. Sendo assim, para sabermos quantas toneladas estão sendo transportadas, precisamos somar todas as rotas que saem da cidade inicial.

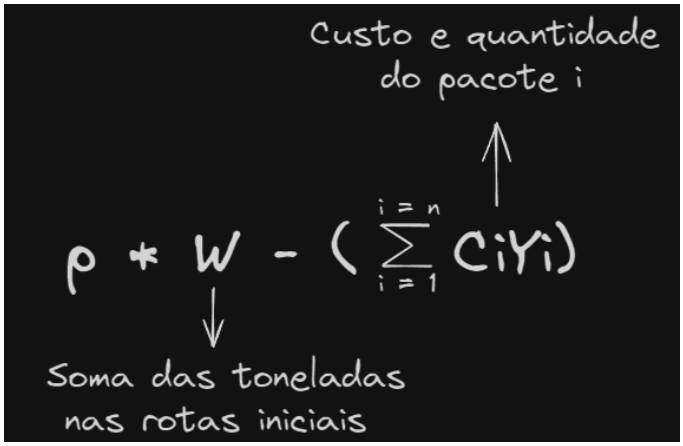
Então, temos que o somatório de todas as rotas que partem da cidade de origem é igual à quantidade de toneladas de produtos transportados.

Nesse caso, já que a variável *p* é utilizada para o valor fixo de cada tonelada transportada, multiplicaremos *W* por *p*. Assim, temos que:

Onde W representa o número de toneladas transportadas em suas rotas iniciais.

Com isso, temos o valor total máximo que se pode conseguir. No entanto, temos que contabilizar o custo dos pacotes necessários, já que precisamos dele para obtermos os recursos utilizados para cada rota.

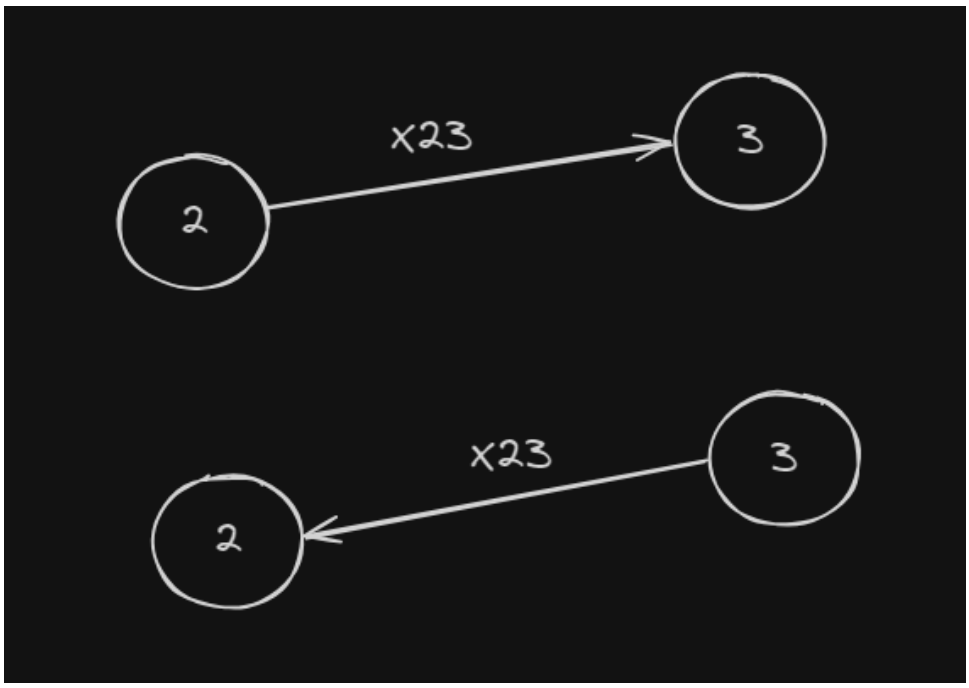
Então, vamos ter que a função objetivo é:



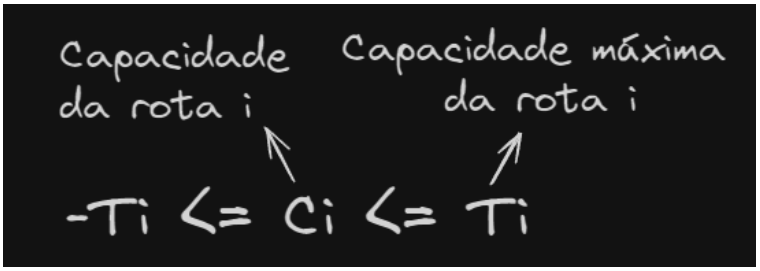
Por fim, pode-se resumir a função objetivo pelo valor fixo de tonelada transportada multiplicado por W, que representa o somatório das toneladas nas rotas que saem da cidade de origem, subtraído pelo somatório de 1 até n, em que n é o número de pacotes existentes, do custo do pacote *i* multiplicado pela quantidade de pacotes *i* comprados na trajetória.

2. Restrições - Capacidade

Para restringirmos a capacidade, temos que entender que a rota pode ter sentidos diferentes. Vamos explicar isso utilizando o seguinte exemplo:



Pode-se ter a mercadoria indo do vértice 2 até o vértice 3 e, nesse caso, o fluxo será positivo. Ou, também, podemos ter a mercadoria “voltando” da cidade 3 para a cidade 2 e, nesse caso, o fluxo será negativo. Então, para uma capacidade máxima T, temos que:



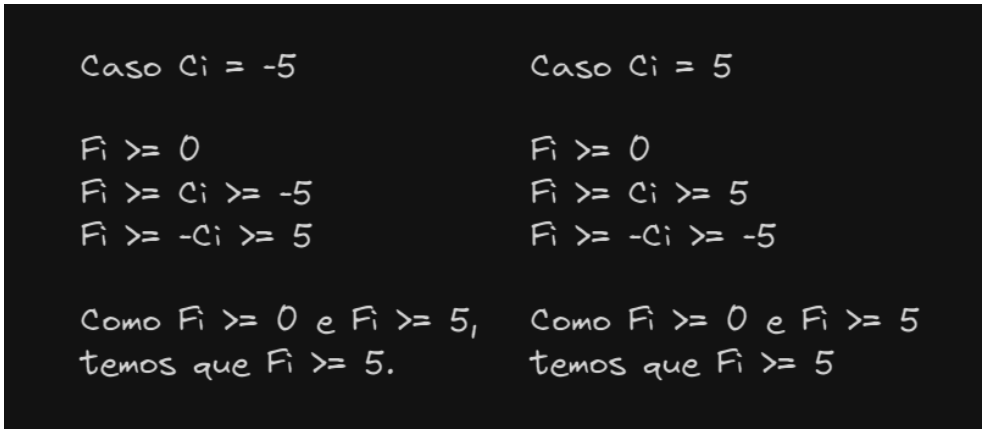
3. Restrições - Recursos e Pacotes

3.1. Módulo

Já temos em mente que a carga de uma rota pode ser tanto positiva quanto negativa, dependendo do seu sentido. Entretanto, não podemos usar valores negativos nas expressões de recursos e pacotes. Sendo assim, se vê a necessidade do uso do módulo da variável.

Para que o módulo seja feito, cria-se uma variável *F*, em que *F >= 0*, *F >= Ci* e *F >= -Ci*.

O exemplo a seguir mostra a funcionalidade da variável e das expressões:



Em que *Ci* = carga da rota *i* e *Fi* = módulo da carga da rota *i*.

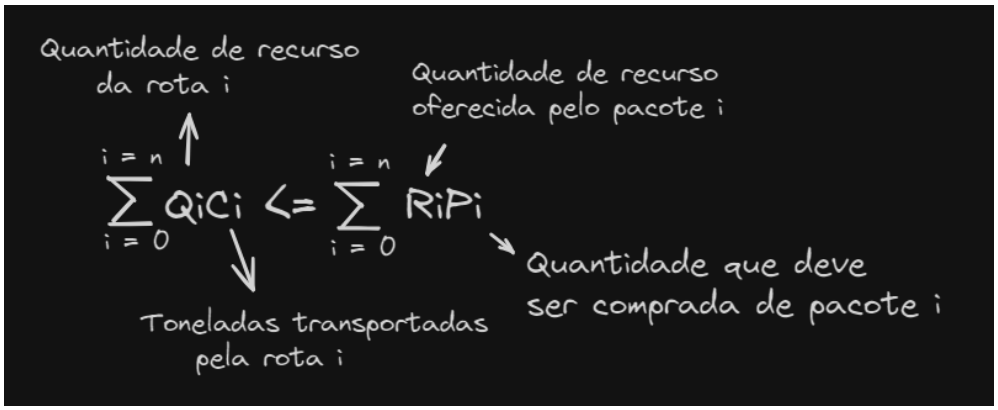
3.2. Restrição

Com o módulo feito, pode-se seguir adiante, já que **não podemos inserir um valor negativo nas restrições**, pois não é viável ter um valor negativo como resultado da multiplicação resultante da quantidade de recurso necessário vezes a carga da rota (isso iria resultar em uma quantidade de recurso negativa, o que não faz sentido).

Inicialmente, o principal objetivo da equação de pacotes e recursos é encontrar o valor mínimo de pacotes comprados. Sendo que estes pacotes devem possuir recursos suficientes para toda a trajetória.

Então, precisamos que a quantidade de recurso necessário para cada rota multiplicado pela carga seja menor ou igual à quantidade de recurso oferecida por cada pacote vezes a quantidade de pacote comprados.

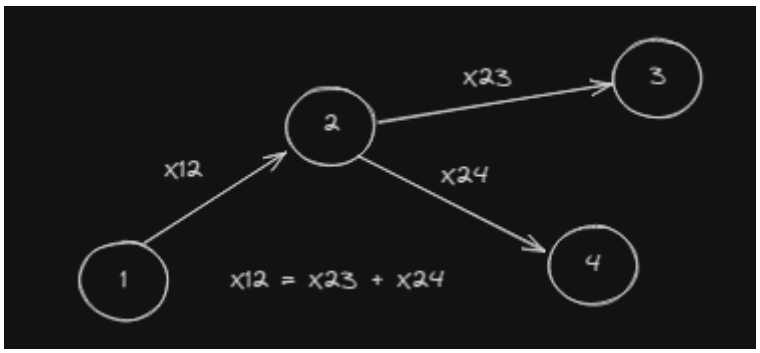
De forma visual e mais clara, temos que:



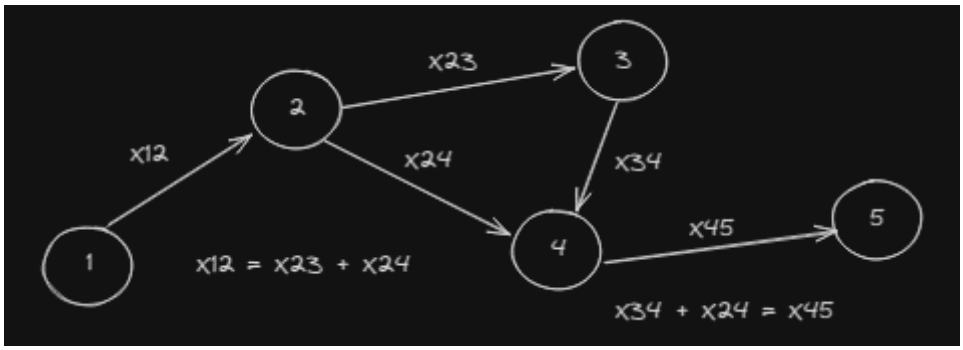
Assim, tem-se o menor valor necessário de pacotes comprados para se obter os recursos.

4. Equações das rotas

Para as equações das rotas, deve-se avaliar os seguintes casos:

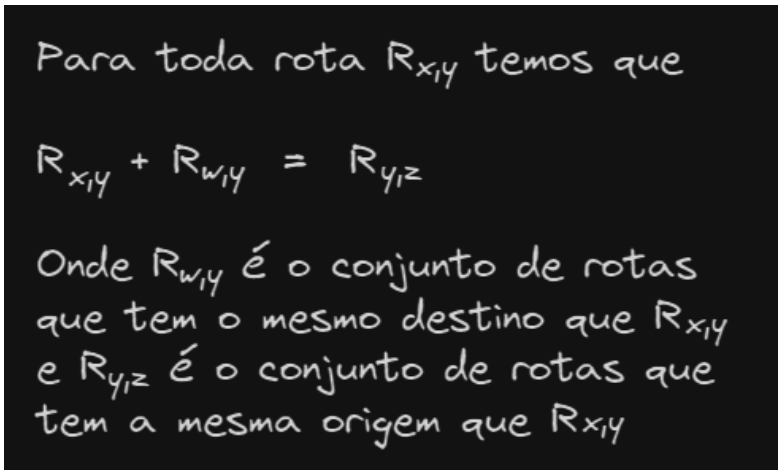


Nesse caso, para sabermos o valor de X12, precisamos somar os valores de X23 e X24. Já que os valores somados dos dois vértices resultam no valor inicial de X12.



Já nessa situação, foi inserido um novo vértice 5 que realiza a rota X45. Assim, para se ter seu valor, precisamos somar as rotas que chegam em 4 que, nesse caso, são X34 e X24.

De forma genérica, tem-se que:



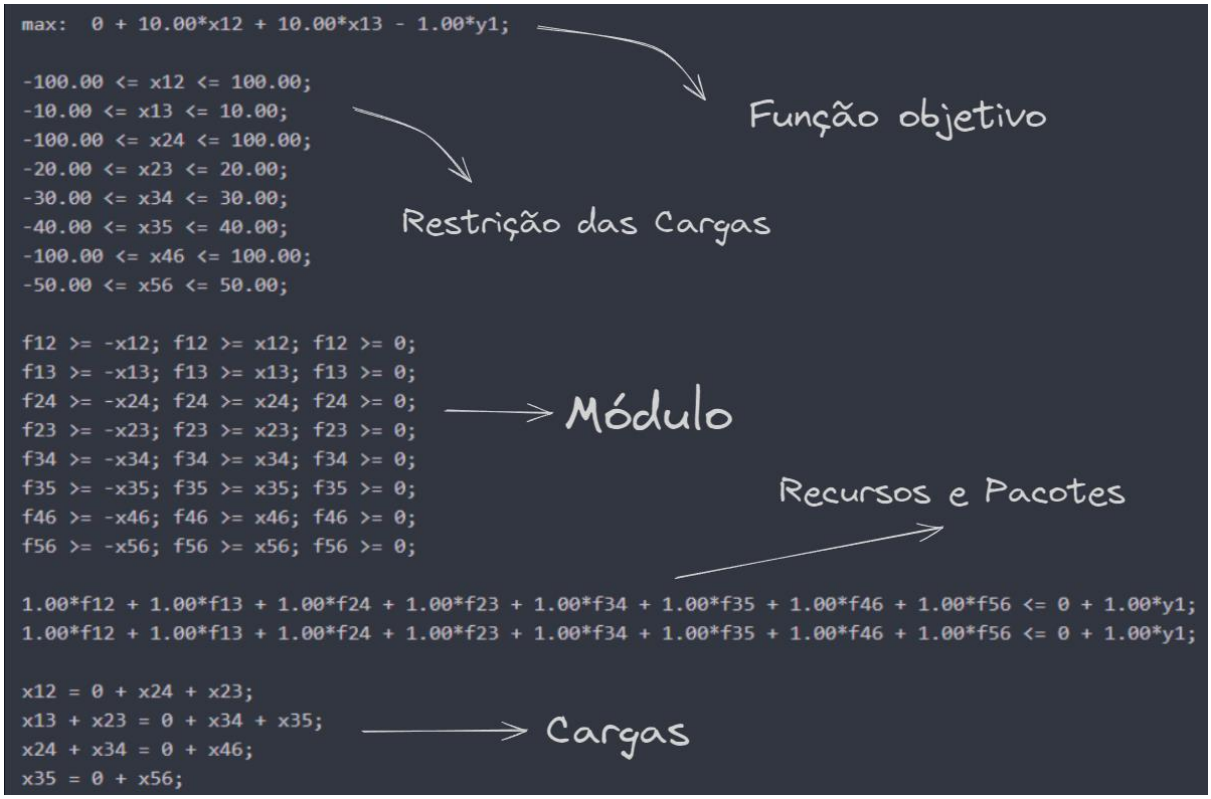
5. Exemplos

5.1. Valor fixo > Custo

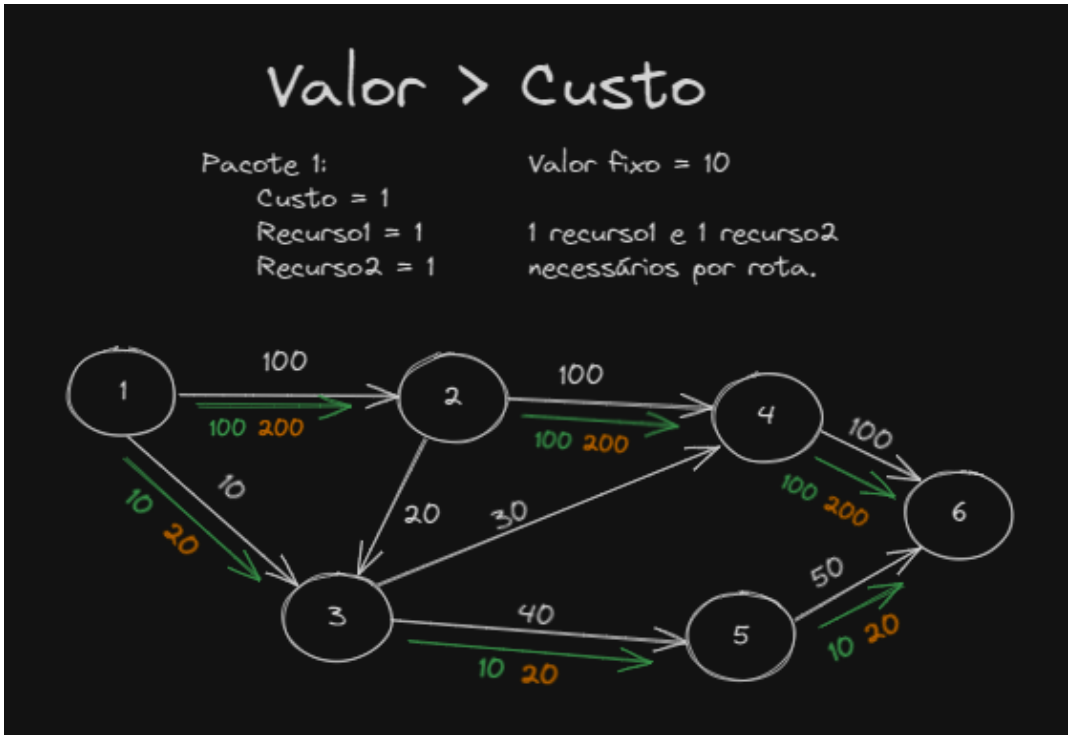
A imagem a seguir mostra uma entrada que representa um exemplo em que o valor fixo total por cada tonelada transportada é maior que o custo dos pacotes comprados.

```
6 8 2 1 10
1 2 100 1 1
1 3 10 1 1
2 4 100 1 1
2 3 20 1 1
3 4 30 1 1
3 5 40 1 1
4 6 100 1 1
5 6 50 1 1
1 1 1
```

A saída do programa resulta em:



Acabei juntando as linhas das equações do módulo para caber inteiramente em uma só imagem. De qualquer forma, representando de forma gráfica, tem-se que:



As setas e valores da cor verde representam a quantidade de toneladas transportadas pela rota. Já as setas e valores de cor laranja representam a quantidade de recursos necessárias para a travessia da rota.

Nesse caso, temos 1100 toneladas transportadas e 660 recursos, sendo 330 recursos do tipo 1 e 330 recursos do tipo 2, necessários ao todo. Assim, como cada pacote custa 1, temos que 1100 – 330 = 770. Sendo esse o resultado da função objetiva.

O resultado gerado pelo lp_solve confirma o que foi dito acima. Então, pode-se afirmar que esse é um exemplo em que a empresa gerou lucro.

```
Value of objective function: 770.00000000

Actual values of the variables:
x12                           100
x13                            10
y1                            330
x24                           100
x23                 -2.16514e-14
x34                 -5.41285e-15
x35                            10
x46                           100
x56                            10
f12                           100
f13                            10
f24                           100
f23                             0
f34                             0
f35                            10
f46                           100
f56                            10
```

5.2. Valor fixo = Custo

Utilizaremos a seguinte entrada para esse exemplo:

```
5 5 1 1 10
1 2 50 0
1 3 100 1
2 4 50 1
3 4 100 0
4 5 150 0
150 10
```

Na saída do programa, obtêm-se:

```
max:  0 + 10.00*x12 + 10.00*x13 - 150.00*y1;

-50.00 <= x12 <= 50.00;
-100.00 <= x13 <= 100.00;
-50.00 <= x24 <= 50.00;
-100.00 <= x34 <= 100.00;
-150.00 <= x45 <= 150.00;

f12 >= -x12;
f12 >= x12;
f12 >= 0;
f13 >= -x13;
f13 >= x13;
f13 >= 0;
f24 >= -x24;
f24 >= x24;
f24 >= 0;
f34 >= -x34;
f34 >= x34;
f34 >= 0;
f45 >= -x45;
f45 >= x45;
f45 >= 0;

0.00*f12 + 1.00*f13 + 1.00*f24 + 0.00*f34 + 0.00*f45 <= 0 + 10.00*y1;


x12 = 0 + x24;
x13 = 0 + x34;
x24 + x34 = 0 + x45;
```

Função objetivo

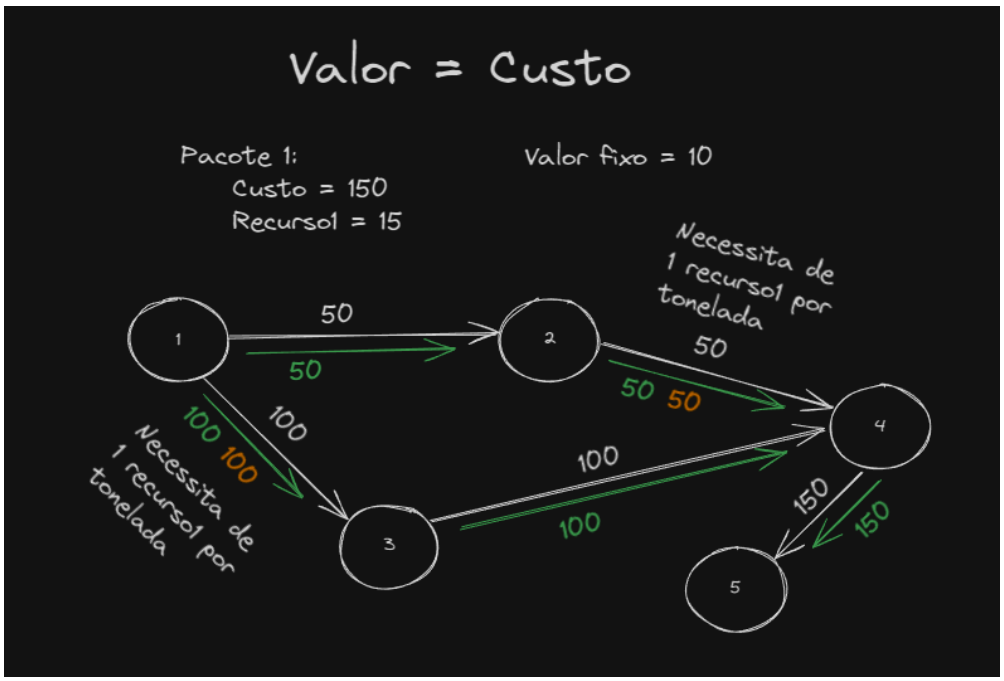
Restrição das Cargas

Módulo

Recursos e Pacotes

Cargas

Graficamente, temos:



Seguindo o mesmo padrão de cores do exemplo anterior, temos um valor fixo de 150*10 = 1500 e, como cada pacote tem 15 recursos e só há um único tipo de recurso, 100+50 recursos necessários. Assim, sabendo que cada pacote custa 150 e precisaremos de 10 pacotes, o custo total de pacotes será 10*150 = 1500.

Dessa forma, temos uma igualdade entre o valor fixo total e o custo total dos pacotes, ou seja, um resultado nulo.

Nesse exemplo, lp_solve zerou as variáveis e a função objetivo (o que é uma solução viável), como é apresentado a seguir:

```
Value of objective function: 0

Actual values of the variables:
x12                    1.33614e-14
x13                              0
y1                               0
x24                     9.978e-15
x34                              0
x45                              0
f12                              0
f13                              0
f24                              0
f34                              0
f45                              0
```

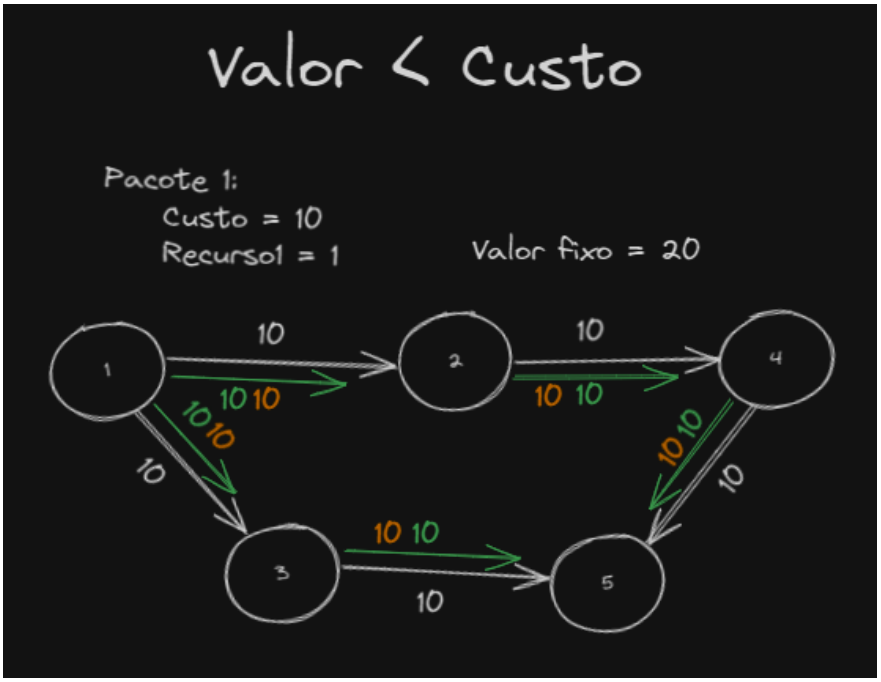
5.3 Valor fixo < Custo

Para esse exemplo, usaremos a seguinte entrada:

```
5 5 1 1 20
1 2 10 10
1 3 10 10
2 4 10 10
3 5 10 10
4 5 10 10
10 1
```

Na saída, temos o seguinte resultado:

Então, graficamente se tem:



Nesse caso, temos 10+10 = 20 toneladas transportadas, gerando 20*20 = 400 de valor total das vendas. Ao mesmo tempo, temos 10+10+10+10+10 = 50 recursos necessários. Sendo assim, terá que ser feito a compra de 50 pacotes, o que gera 10*50 = 500 de custo.

Dessa forma, o custo da compra de pacotes será maior que a venda de toneladas. Portanto, nesse exemplo a empresa sairá no prejuízo.

Mais uma vez, o lp_solve mostra a solução viável em que as variáveis e a função objetivo são nulas, como é visto na imagem a seguir:

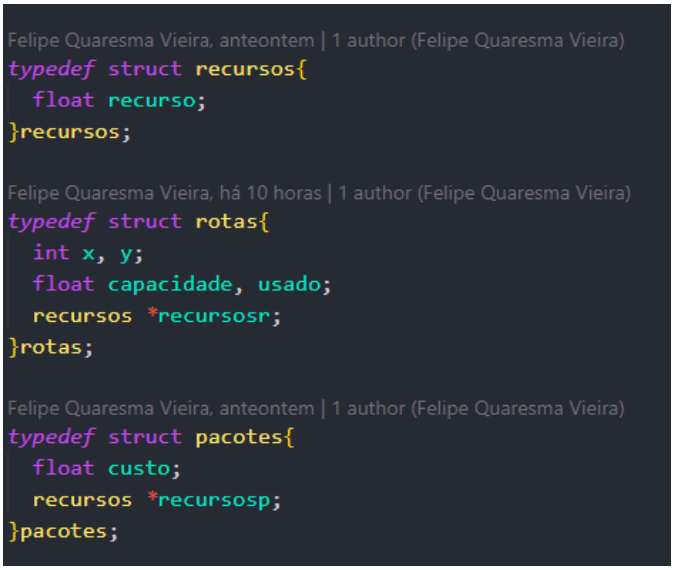
```
Value of objective function: 0

Actual values of the variables:
x12                    1.33614e-14
x13                              0
y1                               0
x24                     9.978e-15
x34                              0
x45                              0
f12                              0
f13                              0
f24                              0
f34                              0
f45                              0
```

6. Código

Nessa seção, falarei um pouco sobre algumas curiosidades sobre a implementação e a linha de raciocínio utilizada para o desenvolvimento do código.

6.1. Estruturas

As estruturas utilizadas para o armazenamento de dados foram as seguintes:



Cada rota tem sua origem e destino, além de sua capacidade e uma variável inteira representando se ela já foi usada (vamos explicar seu uso mais para frente). Além disso, rotas e pacotes possuem um vetor de recursos, em que cada índice representa o tipo de recurso utilizado.

Nos pacotes, há uma variável para seu custo. E os recursos têm um float que indica quantos recursos são necessários (no caso das rotas) ou sua quantidade em um pacote (no caso dos pacotes).

6.2 Equações das rotas

A equação das rotas tem quatro peculiaridades que a tornam mais complexa de ser implementada, essas são:

1 – Nenhuma rota que aparece do lado esquerdo da equação é mostrada de novo em outras equações.

2 – As rotas do lado esquerdo possuem o mesmo destino da primeira variável.

3 – As rotas do lado direito da equação possuem mesma origem (sendo que esta origem, é o destino da primeira variável do lado esquerdo).

4 – As rotas que possuem como destino a cidade final não devem aparecer do lado esquerdo da equação.

Para que as condições 2, 3 e 4 fossem cumpridas, foram colocadas condicionais que impedissem a escrita errada nessas situações. Já para a condição 1, foi criada uma variável no struct de cada rota chamada “usado” que indicasse se aquela rota já foi utilizada em uma equação ou se ela está disponível.

Para facilitar o entendimento, a implementação da equação de rotas ficou dessa maneira:

```
//Equações das rotas

for(int i = 0; i < m; i++){

  //condição 1 e 4
  if (rotas_array[i].usado == 0 && rotas_array[i].y != n){
    fprintf(arq, "x%d%d", rotas_array[i].x, rotas_array[i].y);
    x = rotas_array[i].x;
    y = rotas_array[i].y;

    for(int j = i; j < m; j++){
      //condição 2
      if(rotas_array[j].y == y && rotas_array[j].x != x){
        fprintf(arq, " + x%d%d", rotas_array[j].x, rotas_array[j].y);
        rotas_array[j].usado = 1;
      }
    }

    fprintf(arq, " = 0");

    for(int k = i; k < m; k++){
      //condicao 3 e 1
      if(rotas_array[k].x == y && rotas_array[k].y != x && rotas_array[k].usado == 0)
        fprintf(arq, " + x%d%d", rotas_array[k].x, rotas_array[k].y);

    }

    fprintf(arq, ";\n");

  }
  //condição 1
  rotas_array[i].usado = 1;
}
```

Foram utilizados arrays de structs para armazenar tanto as rotas, como pacotes e recursos.

6.3 Compilação

Para compilar o código, deve-se usar o comando “make”. Após isso, você pode inserir os dados manualmente ou utilizar um arquivo com os dados da entrada como “./transporte < Exemplos/exemplo1.txt”.

Depois disso, você pode obter o resultado do lp_solve com o comando “lp_solve entrada.txt”.