Modelagem do problema Cadeia de Suprimentos:

Problema de especificação 7

Lucas Guedes, Matheus Batista Honório, Victoria Monteiro Pontes

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2020

Lucas Guedes, Matheus Batista Honório, Victoria Monteiro Pontes

Modelagem do problema Cadeia de Suprimentos

Relatório apresentado à disciplina Pesquisa Operacional do curso Engenharia de Computação do Centro de Informática, da Universidade Federal da Paraíba.

Professor: Teobaldo Bulhões

Agosto de 2020

AGRADECIMENTOS

O agradecimento é opcional

RESUMO

Um resumo de trabalho de conclusão de curso é do tipo informativo e deve conter somente um parágrafo. A estrutura do resumo deve conter essencialmente os seguintes tópicos: apresentar inicialmente os objetivos do trabalho (o que foi feito?), a justificativa (porquê foi feito) e, finalmente, os resultados alcançados. O resumo deve informar ao leitor todas as informações importantes para o que o leitor possa entender o trabalho desenvolvido, quais foram as finalidades, a metodologia que o autor utilizou e os resultados obtidos. Deve conter frases curtas, porém completas (evitar estilo telegráfico); usar o tempo verbal no passado para os principais resultados e presente para comentários ou para salientar implicações significativas. O resumo em português e inglês são obrigatórios e não devem passar de 200 palavras.

Palavras-chave: <Primeira palavra>, <segunda palavra>, <até 5 palavras>. < Obs.: as palavras-chave devem ser escolhidas com bastante rigor, pois devem representar adequadamente os principais temas abordados pela pesquisa.>

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

SIGLA – NOME COMPLETO

LUMO – Laboratório de computação Móvel e Ubíqua

UbiComp – Computação Ubíqua

Sumário

1 INTRODUÇÃO

O trabalho envolve a área de Pesquisa Operacional, utilizada para auxiliar na tomada de decisões diante de um determinado problema. No caso deste trabalho, utilizaremos da modelagem matemática para alcançar o mesmo propósito citado.

1.1 Problema da Cadeia de Suprimento

O problema da cadeia de suprimento é um problema proposto pelo professor da cadeira. Este é um problema de transporte e distribuição, que inclui fábricas, centros de distribuição e cidades que possuem demandas associadas aos produtos produzidos nas fábricas.

1.2 Descrição e dados do problema

Uma empresa de cimento possui *n* fábricas e deve atender a *m* cidades (regiões metropolitanas). A capacidade anual e o custo de produção de cada fábrica *i* são conhecidos e dados por *CAPi* e *Ci*. Cada cidade *j* possui um valor *Dj* de demanda anual estimada. Até *Dj* toneladas podem ser vendidas a cidade *j* ao preço de *P* reais/ton. O transporte das fábricas até as cidades pode ser feito de duas formas. Da primeira forma, caminhões transportam diretamente da fábrica *i* para a cidade *j* ao custo de *CC* reais/ton/km. Da segunda forma, pode-se usar centros de distribuição intermediários, havendo *K* desses centros. O transporte da fábrica *i* até o centro *k* é feito por ferrovia e custa *CF* reais/ton/km, o transporte do centro *k* até a cidade *j* é feito por caminhão e custa *CC* reais/ton/km. Entretanto, para usar o centro de distribuição *k*, deve-se pagar uma taxa fixa anual de *Fk* reais.

1.2.1 Objetivo geral

Deve-se determinar o quanto cada fábrica deve produzir e quanto deve ser transportado para cada cidade de forma a maximizar o lucro da empresa no ano.

1.2.2 Objetivos específicos

- Criar variáveis de decisão e determinar seus domínios
- Criar uma função objetivo que se encaixe com o objetivo de resolução do problema
- Adicionar restrições que simulam as limitações do problema citado.

2 METODOLOGIA

2.1 Dados de entrada do problema e legendas

Antes de modelar qualquer problema é necessário organizar os dados de entrada assim como fornecer legendas para saber o que está associado a eles. Portanto, no problema da cadeia de suprimentos, esses são os seguintes dados de entrada:

Observação: Alguns dados de entrada foram descritos com legendas diferentes do que foi descrito no problema devido à facilidade de confundí-las. Exemplo: *C* é um conjunto de cidades/clientes, porém *Ci* é um custo de produção em reais/ton de uma fábrica *i*, portanto *Ci* foi substituída por *CTi* para evitar equívocos.

- *ij* representa os arcos de uma instância, *i* é o lugar de onde se sai e *j* é o lugar de destino
- *F* o conjunto de fábricas
- *K* o conjunto de centros de distribuição
- *C* o conjunto de cidades atendidas
- *CPAi* a capacidade produtiva máxima de uma fábrica *i* em toneladas
- *CTi* custo de produção em reais/ton em uma fábrica *i*
- *Dj* a demanda anual em toneladas de uma cidade atendida *j*
- *CF* Custo de transporte por ferrovias em reais/ton/km. Este custo é utilizado apenas em arcos que ligam as fábricas aos centros de distribuição
- *CC* Custo de transporte por caminhão em reais/ton/km. Este custo é utilizado apenas em arcos que ligam fábricas diretamente às cidades ou que ligam centros de distribuição às cidades
- *Fk* Taxa anual pela utilização de um centro de distribuição *k*
- d_{ij} Distância de um arco *ij*
- *P* Preço de venda em reais/ton

2.2 Determinação de variáveis de decisão

No problema devem ser identificadas quais decisões tomadas são necessárias para que uma solução do problema seja elaborada. Essas decisões são separadas em dois grupos:

- Decidir quantas toneladas do produto passam por cada arco da cadeia de suprimentos
- Decidir quais centros de distribuição serão utilizados ou não

Desse modo, dois tipos de variáveis serão criadas:

$X_{ij} \geq 0; \forall i \in F \cup K, \forall j \in K \cup C, i \neq j$

$Y_k \in \{0, 1\}, \forall k \in K$

2.3 Determinando a função objetivo

Depois da definição das variáveis de decisão, podemos utilizar os dados fornecidos pela instância e determinar como será a função objetivo. No caso, o propósito no problema é maximizar o lucro. Seguindo esse raciocínio, a função objetivo consiste na subtração de todos os custos da receita total de todas as fábricas. E o objetivo é maximizar essa função. Então a função objetivo é formulada desta maneira:

$$max Z = \sum_{j \in C} \sum_{i \in K \cup F} P \cdot X_{ij} - \sum_{i \in F} \sum_{j \in K \cup C} CT_i \cdot X_{ij} - \sum_{k \in K} F_k \cdot Y_k - \sum_{j \in C} \sum_{i \in K \cup F} CC \cdot X_{ij} \cdot d_{ij} - \sum_{i \in F} \sum_{j \in K} CF \cdot X_{ij} \cdot d_{ij}$$

2.4 Adicionando restrições ao modelo

1. Restrição de conservação de fluxo nos centros de distribuição

$$\sum_{j \in C} X_{kj} = \sum_{j \in F} X_{jk}, \forall k \in K$$

   Essa restrição garante que o fluxo de produto que entra no centro de distribuição seja o mesmo que sai.

2. Restrição de capacidade produtiva nas fábricas

$$\sum_{j \in K \cup C} X_{ij} \leq CAP_i, \forall i \in F$$

   Essa restrição garante que nenhuma das fábricas exceda a sua respectiva capacidade máxima de produção

3. Restrição de demanda de cada cidade

$$\sum_{i \in K \cup F} X_{ij} \leq D_j, \forall j \in C$$

   Essa restrição garante que as cidades não recebam um fluxo de produto maior que suas respectivas demandas

4. Restrição dos centros de distribuição utilizados

$$\sum_{j \in C} X_{kj} \leq \sum_{j \in C} D_j \cdot Y_k, \forall k \in K$$

3 IMPLEMENTAÇÃO DO MODELO E ANÁLISE DOS RESULTADOS

3.1 Instância

3.1.1 Legenda

- P - Preço reais*toneladas.
- CC - Custo por caminhão.
- CF - Custo por Ferrovia.
- n - Quantidade de Fábricas.
- K - Quantidade de Centros de Distribuição.
- m - Número de Cidades (regiões metropolitanas)
- Fábrica 1 / Custo de produção da fábrica 1 / Capacidade da fábrica 1

  ...
- Fábrica i / Custo de produção da fábrica i / Capacidade da fábrica i

- Centro 1 / Capacidade do Centro 1

  ...
- Centro K / Capacidade do Centro K

- Cidade 1 / Demanda da Cidade 1

  ...
- Cidade j / Demanda da Cidade j

- Cidade de origem n / Cidade de demanda m / Distância em km pela estrada

Número e cidade que o representa na instância:

- 1 - CASCAVEL
- 2 - CAMPINAS
- 3 - FEIRA DE SANTANA
- 4 - IMPERATRIZ
- 5 - CUIABÁ
- 6 - BRASÍLIA
- 7 - PETROLINA
- 8 - ARACAJÚ
- 9 - BELÉM
- 10 - BELO HORIZONTE
- 11 - BOA VISTA
- 12 - CAMPO GRANDE

- 13 - CURITIBA
- 14 - FLORIANÓPOLIS
- 15 - FORTALEZA
- 16 - GOIÂNIA
- 17 - JOÃO PESSOA
- 18 - MACAPÁ
- 19 - MACEIÓ
- 20 - MANAUS
- 21 - NATAL
- 22 - PORTO ALEGRE
- 23 - PORTO VELHO
- 24 - RECIFE
- 25 - RIBEIRÃO PRETO
- 26 - RIO BRANCO
- 27 - RIO DE JANEIRO
- 28 - SALVADOR
- 29 - SÃO LUÍS
- 30 - SÃO PAULO
- 31 - TERESINA
- 32 - VITÓRIA
- 33 - CUIABÁ

3.1.2 Instância da Modelagem:

```
600
0.1
0.04
4
3
26
1 2 2000
2 3 3000
3 4 4500
4 3 3500
5 50
6 70
7 65
8 600
9 400
10 150
```

```
11 100
12 300
13 480
14 150
15 300
16 180
17 200
18 80
19 90
20 400
21 600
22 700
23 500
24 400
25 100
26 100
27 210
28 450
29 120
30 300
31 450
32 200
33 150
1 5 1341
1 6 1432
1 7 1350
1 8 2950
1 9 3242
1 10 1369
1 11 4467
1 12 637
1 13 498
1 14 805
1 15 3707
1 16 1235
1 17 3745
1 18 2385
1 19 3226
1 20 3682
1 21 3720
1 22 2155
1 23 885
```

```
1 24 2781
1 25 3433
1 26 3315
1 27 1333
1 28 2746
1 29 3279
1 30 908
1 31 3211
1 32 1786
1 33 1340
2 5 1485
2 6 921
2 7 566
2 8 2182
2 9 2842
2 10 601
2 11 4665
2 12 1012
2 13 476
2 14 773
2 15 3133
2 16 835
2 17 2775
2 18 2805
2 19 2458
2 20 3880
2 21 2952
2 22 1727
2 23 1177
2 24 2979
2 25 2665
2 26 3513
2 27 511
2 28 1982
2 29 2879
2 30 99
2 31 2698
2 32 959
2 33 1484
3 5 2449
3 6 1380
3 7 399
```

```
3 8 322
3 9 1984
3 10 1256
3 11 5678
3 12 2537
3 13 2269
3 14 2566
3 15 1273
3 16 1612
3 17 3485
3 18 2844
3 19 598
3 20 4893
3 21 1092
3 22 2389
3 23 2974
3 24 3992
3 25 805
3 26 4526
3 27 1533
3 28 116
3 29 1483
3 30 1846
3 31 1047
3 32 1124
3 33 2450
4 5 2141
4 6 1387
4 7 52
4 8 1909
4 9 609
4 10 2231
4 11 5484
4 12 2343
4 13 2594
4 14 2901
4 15 1401
4 16 1418
4 17 1991
4 18 1134
4 19 2003
4 20 4699
```

```
4 21 1938
4 22 624
4 23 3255
4 24 3789
4 25 1904
4 26 4332
4 27 2657
4 28 1930
4 29 636
4 30 2334
4 31 777
4 32 2747
4 33 2140
5 8 2773
5 9 2941
5 10 1594
5 11 3142
5 12 694
5 13 1679
5 14 1986
5 15 3406
5 16 934
5 17 3366
5 18 3046
5 19 3049
5 20 2357
5 21 3543
5 22 1784
5 23 2206
5 24 1456
5 25 3255
5 26 1990
5 27 2017
5 28 2566
5 29 2978
5 30 1614
5 31 2910
5 32 2119
5 33 1
6 8 1650
6 9 2140
6 10 741
```

```
6 11 4275
6 12 1134
6 13 1366
6 14 1673
6 15 2200
6 16 209
6 17 2245
6 18 2499
6 19 1930
6 20 3490
6 21 2422
6 22 973
6 23 2027
6 24 2589
6 25 2135
6 26 3123
6 27 1148
6 28 1446
6 29 2157
6 30 1015
6 31 1789
6 32 1239
6 33 1070
7 8 490
7 9 1589
7 10 1650
7 11 5898
7 12 2672
7 13 2663
7 14 2966
7 15 868
7 16 1747
7 17 875
7 18 2085
7 19 723
7 20 5113
7 21 923
7 22 1473
7 23 3368
7 24 4127
7 25 770
7 26 4661
```

```
7 27 1927
7 28 513
7 29 1088
7 30 2246
7 31 652
7 32 1519
7 33 2595
```

3.2 Código de implementação

```
#Leitura do arquivo e armazenando os dados
import pylab as pb
#Armazenando o preco, custo CC (caminhao) e CF (ferrovia)
   respectivamente
prices_info = pb.loadtxt('projeto2.txt', dtype = float, max_rows = 3)

#Armazenando o n mero de fabricas, centros e cidades
instance_info = pb.loadtxt('projeto2.txt', dtype = int, skiprows = 3,
   max_rows = 3)
print(instance_info[2])

#Armazenando dados associados a cada fabrica, centro e cidade:
dataFab = pb.loadtxt('projeto2.txt', dtype = int, skiprows = 6,
   max_rows = instance_info[0])
dataCenter = pb.loadtxt('projeto2.txt', dtype = int, skiprows = 6 +
   instance_info[0], max_rows = instance_info[1])
dataCity = pb.loadtxt('projeto2.txt', dtype = int, skiprows = 6 +
   instance_info[0] + instance_info[1],
                      max_rows = instance_info[2])


skip_rows = sum(instance_info) + len(prices_info) + len(instance_info)

#Armazenando os dados dos arcos
dataSet = pb.loadtxt('projeto2.txt',dtype = int, skiprows = skip_rows)

#Criando um dicion rio que organiza melhor os dados relacionados aos
   arcos (distancia):
distanceList = {}
for i in range(len(dataSet)):
    distanceList[(dataSet[i][0], dataSet[i][1])] = dataSet[i][2]

```

```
print(distanceList)

#Criando conjunto de fabricas, centros e cidades
#Criando ranges para facilitar a manipulacao dos dados
fab_r = range(instance_info[0])
cent_r = range(instance_info[1])
city_r = range(instance_info[2])

#Criando conjunto de Fabricas:
factories=[]
for i in fab_r:
    factories.insert(i, dataFab[i][0])

#Criando conjunto de centros:
centers=[]
for i in cent_r:
    centers.insert(i, dataCenter[i][0])

#Criando conjunto de cidades:
cities=[]
for i in city_r:
    cities.insert(i, dataCity[i][0])

#Criando unioes entre conjunto de fabricas, centros e cidades:
#Fabrica uniao centros
factoryUcenter = factories + centers

#Centros uniao cidades
centerUcity = centers + cities

#Criando conjunto com os custos de producao por tonelada Ci de cada
   fabrica (utilizando um dicionario):
costFactory = {}
for i in fab_r:
    costFactory[(dataFab[i][0])] = dataFab[i][1]
print(costFactory)

#Criando conjunto com as taxas anuais a serem pagas por cada centro de
    distribuicao utilizado:
anualTax={}
for i in cent_r:
    anualTax[(dataCenter[i][0])] = dataCenter[i][1]
```

```
print(anualTax)

#Criando conjunto com as capacidades maximas de producao de cada
   fabrica (utilizando um dicionario):
capacities={}
for i in fab_r:
    capacities[(dataFab[i][0])] = dataFab[i][2]
print(capacities)

#Criando conjunto com as demandas de cada cidade (utilizando um
   dicionario):
demands ={}
for i in city_r:
    demands[(dataCity[i][0])] = dataCity[i][1]
print(demands)

#Modelagem
from docplex.mp.model import Model
mdl = Model("Problema de especificacao 7")

#Criando variaveis X_i_j
x = {(i, j): mdl.continuous_var(name="x_{0}_{1}".format(i, j)) for (i,
    j) in distanceList}

mdl.print_information()

#Criando variaveis Y_k, sendo k o centro de distribuicao:
y = {i: mdl.binary_var(name="y_{0}".format(i)) for i in centers}

mdl.print_information()
print(y)

#Adicionando restricao de conservacao de fluxo:
for k in centers:
    mdl.add_constraint(mdl.sum(x[k, j] for j in cities) == mdl.sum(x[i
       , k] for i in factories))

#Adicionando restricao sobre a capacidade produtiva maxima de cada
   fabrica:
for i in factories:
    mdl.add_constraint(mdl.sum(x[i, j] for j in centerUcity) <=
       capacities[i])
```

```
103
104 #Adicionando restricao que limita a quantidade de X_i_j enviada
105 for j in cities:
106     mdl.add_constraint(mdl.sum(x[i, j] for i in factoryUcenter) <=
        demands[j])
107
108 #Adicionando restricao que conserva o fluxo associado a utilizacao ou
    nao dos centros de distribuicao:
109 for k in centers:
110     mdl.add_constraint(mdl.sum(x[k, j] for j in cities) <= mdl.sum(
        demands[j]*y[k]))
111
112 #Adicionando a funcao objetivo:
113 mdl.maximize(mdl.sum(prices_info[0]* x[i, j] for j in cities for i in
    factoryUcenter) -
114              mdl.sum(costFactory[i]*x[i, j] for i in factories for j
                 in centerUcity) -
115              mdl.sum(anualTax[k]*y[k] for k in centers) -
116              mdl.sum(prices_info[1] * x[i, j] * distanceList[i, j] for
                  j in cities for i in factoryUcenter) -
117              mdl.sum(prices_info[2] * x[i, j] * distanceList[i, j] for
                  i in factories for j in centers))
118
119 solution = mdl.solve(log_output=True)
120
121 solution.display()
```

3.3 Resultados

Print da Lista de Distâncias de cada cidade:

```
1 {(1, 5): 1341, (1, 6): 1432, (1, 7): 1350, (1, 8): 2950, (1, 9): 3242,
    (1, 10): 1369, (1, 11): 4467, (1, 12): 637, (1, 13): 498, (1, 14):
    805, (1, 15): 3707, (1, 16): 1235, (1, 17): 3745, (1, 18): 2385,
   (1, 19): 3226, (1, 20): 3682, (1, 21): 3720, (1, 22): 2155, (1, 23)
   : 885, (1, 24): 2781, (1, 25): 3433, (1, 26): 3315, (1, 27): 1333,
   (1, 28): 2746, (1, 29): 3279, (1, 30): 908, (1, 31): 3211, (1, 32):
    1786, (1, 33): 1340, (2, 5): 1485, (2, 6): 921, (2, 7): 566, (2,
   8): 2182, (2, 9): 2842, (2, 10): 601, (2, 11): 4665, (2, 12): 1012,
    (2, 13): 476, (2, 14): 773, (2, 15): 3133, (2, 16): 835, (2, 17):
   2775, (2, 18): 2805, (2, 19): 2458, (2, 20): 3880, (2, 21): 2952,
   (2, 22): 1727, (2, 23): 1177, (2, 24): 2979, (2, 25): 2665, (2, 26)
   : 3513, (2, 27): 511, (2, 28): 1982, (2, 29): 2879, (2, 30): 99,
```

```
  (2, 31): 2698, (2, 32): 959, (2, 33): 1484, (3, 5): 2449, (3, 6):
  1380, (3, 7): 399, (3, 8): 322, (3, 9): 1984, (3, 10): 1256, (3,
  11): 5678, (3, 12): 2537, (3, 13): 2269, (3, 14): 2566, (3, 15):
  1273, (3, 16): 1612, (3, 17): 3485, (3, 18): 2844, (3, 19): 598,
  (3, 20): 4893, (3, 21): 1092, (3, 22): 2389, (3, 23): 2974, (3, 24)
  : 3992, (3, 25): 805, (3, 26): 4526, (3, 27): 1533, (3, 28): 116,
  (3, 29): 1483, (3, 30): 1846, (3, 31): 1047, (3, 32): 1124, (3, 33)
  : 2450, (4, 5): 2141, (4, 6): 1387, (4, 7): 52, (4, 8): 1909, (4,
  9): 609, (4, 10): 2231, (4, 11): 5484, (4, 12): 2343, (4, 13):
  2594, (4, 14): 2901, (4, 15): 1401, (4, 16): 1418, (4, 17): 1991,
  (4, 18): 1134, (4, 19): 2003, (4, 20): 4699, (4, 21): 1938, (4, 22)
  : 624, (4, 23): 3255, (4, 24): 3789, (4, 25): 1904, (4, 26): 4332,
  (4, 27): 2657, (4, 28): 1930, (4, 29): 636, (4, 30): 2334, (4, 31):
   777, (4, 32): 2747, (4, 33): 2140, (5, 8): 2773, (5, 9): 2941, (5,
   10): 1594, (5, 11): 3142, (5, 12): 694, (5, 13): 1679, (5, 14):
  1986, (5, 15): 3406, (5, 16): 934, (5, 17): 3366, (5, 18): 3046,
  (5, 19): 3049, (5, 20): 2357, (5, 21): 3543, (5, 22): 1784, (5, 23)
  : 2206, (5, 24): 1456, (5, 25): 3255, (5, 26): 1990, (5, 27): 2017,
   (5, 28): 2566, (5, 29): 2978, (5, 30): 1614, (5, 31): 2910, (5,
  32): 2119, (5, 33): 1, (6, 8): 1650, (6, 9): 2140, (6, 10): 741,
  (6, 11): 4275, (6, 12): 1134, (6, 13): 1366, (6, 14): 1673, (6, 15)
  : 2200, (6, 16): 209, (6, 17): 2245, (6, 18): 2499, (6, 19): 1930,
  (6, 20): 3490, (6, 21): 2422, (6, 22): 973, (6, 23): 2027, (6, 24):
   2589, (6, 25): 2135, (6, 26): 3123, (6, 27): 1148, (6, 28): 1446,
  (6, 29): 2157, (6, 30): 1015, (6, 31): 1789, (6, 32): 1239, (6, 33)
  : 1070, (7, 8): 490, (7, 9): 1589, (7, 10): 1650, (7, 11): 5898,
  (7, 12): 2672, (7, 13): 2663, (7, 14): 2966, (7, 15): 868, (7, 16):
   1747, (7, 17): 875, (7, 18): 2085, (7, 19): 723, (7, 20): 5113,
  (7, 21): 923, (7, 22): 1473, (7, 23): 3368, (7, 24): 4127, (7, 25):
   770, (7, 26): 4661, (7, 27): 1927, (7, 28): 513, (7, 29): 1088,
  (7, 30): 2246, (7, 31): 652, (7, 32): 1519, (7, 33): 2595}
```

Printando as Informações do Modelo:

```
Model: Problema de especificacao 7
 - number of variables: 194
   - binary=0, integer=0, continuous=194
 - number of constraints: 0
   - linear=0
 - parameters: defaults
 - objective: none
 - problem type is: LP
```

Printando as informações do solver e do solution

```
Version identifier: 12.10.0.0 | 2019-11-27 | 843d4de
CPXPARAM_Read_DataCheck                          1
CPXPARAM_RandomSeed                              201903125
Found incumbent of value 0.000000 after 0.05 sec. (0.01 ticks)
Tried aggregator 1 time.
Reduced MIP has 36 rows, 197 columns, and 469 nonzeros.
Reduced MIP has 3 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.02 sec. (0.15 ticks)
Probing time = 0.00 sec. (0.01 ticks)
Tried aggregator 1 time.
Detecting symmetries...
Reduced MIP has 36 rows, 197 columns, and 469 nonzeros.
Reduced MIP has 3 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.00 sec. (0.18 ticks)
Probing time = 0.00 sec. (0.01 ticks)
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 8 threads.
Root relaxation solution time = 0.06 sec. (0.20 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt
           Gap

*     0+    0                          0.0000   1.60262e+07
                 ---
*     0     0      integral     0  3822446.0000  3822446.0000       49
       0.00%
Elapsed time = 0.18 sec. (0.75 ticks, tree = 0.00 MB, solutions = 2)

Root node processing (before b&c):
  Real time             =    0.19 sec. (0.76 ticks)
Parallel b&c, 8 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    0.19 sec. (0.76 ticks)
```

Printando o resultado do Problema, função objetivo e resultado dos arcos.

```
solution for: Problema de especificacao 7
```

```
objective: 3822446.000
x_1_5 = 150.000
x_1_11 = 100.000
x_1_12 = 300.000
x_1_20 = 400.000
x_1_23 = 500.000
x_1_24 = 400.000
x_1_26 = 100.000
x_2_6 = 150.000
x_2_10 = 150.000
x_2_13 = 480.000
x_2_14 = 150.000
x_2_16 = 30.000
x_2_27 = 210.000
x_2_30 = 300.000
x_2_32 = 200.000
x_3_8 = 600.000
x_3_15 = 300.000
x_3_19 = 90.000
x_3_21 = 600.000
x_3_25 = 100.000
x_3_28 = 450.000
x_4_7 = 150.000
x_4_9 = 400.000
x_4_17 = 50.000
x_4_18 = 80.000
x_4_22 = 700.000
x_4_29 = 120.000
x_4_31 = 450.000
x_5_33 = 150.000
x_6_16 = 150.000
x_7_17 = 150.000
y_5 = 1
y_6 = 1
y_7 = 1
```

4 CONCLUSÕES

O objetivo do trabalho foi concluído, uma vez que a modelagem matemática e sua implementação em Python foi completa e a instância sugerida foi resolvida com um lucro máximo de 3822446.000 com o auxílio do CPLEX. Assim como todas as informações necessárias do trabalho foram exibidas junto com a resolução da instância.