Training Optimization FINANMOS

Penyelesaian dengan R

Ikanx Fadhli @nutrifood

26 Maret 2021

Contents

1 Kasus *Optimization* I

1.1 Masalah

Manajemen **Koperasi Susu Berkah** (**KSB**) setiap harinya menerima `1000` liter susu dari para anggotanya untuk diproduksi menjadi *yogurt* atau keju *mozarella*.

- Keuntungan dari setiap liter susu yang terjual adalah `Rp1.000`.
- Keuntungan dari *yogurt* yang terjual dari bahan satu liter susu adalah `Rp1.200`
- Sedangkan keuntungan keju *mozarella* dari bahan satu liter susu adalah `Rp900`.

Setelah menganalisa data penjualan, manajemen koperasi mendapatkan informasi sebagai berikut:

- Paling banyak `500` liter susu.
- *Yogurt* paling banyak bisa dibuat dari bahan `300` liter susu.
- Keju *mozarella* paling banyak bisa dibuat dari bahan `400` liter susu.

Dari informasi di atas, manajemen **KSB** ingin menentukan berapa banyak susu yang harus dibuat *yogurt*, susu yang harus dibuat keju *mozarella*, dan susu yang dijual langsung, agar keuntungan yang didapat maksimal.

1.2 Model Matematika

Dari kasus di atas, kita akan membuat model matematikanya.

1.2.1 Variabel Penentuan

Misalkan saya notasikan `3` variabel berikut ini:

- x_1 sebagai seberapa banyak (dalam liter) susu yang bisa dijual langsung,
- x_2 sebagai seberapa banyak (dalam liter) susu yang bisa dibuat *yogurt*, dan
- x_3 sebagai seberapa banyak (dalam liter) susu yang bisa dibuat keju *mozarella*.

Di mana:

$$x_1, x_2, x_3 \in \mathbb{Z}$$

1.2.2 *Constraints*

Berikut adalah *constraints* yang ada pada kasus di atas:

- $0 \leq x_1 \leq 500$
- $0 \leq x_2 \leq 300$
- $0 \leq x_3 \leq 400$
- $x_1 + x_2 + x_3 = 1000$

1.2.3 *Objective Function*

Tujuan utama permodelan matematika ini adalah **memaksimalkan** *profit* yang ingin dicapai **KSB**, yakni:

$$max(1000x_1 + 1200x_2 + 900x_3)$$

1.3 *Solver R*

1.3.1 Penulisan Model Matematika di R

Untuk menyelesaikan masalah ini, saya akan menggunakan *solver* di **R**. Berikut adalah model yang saya buat:

```
# dimulai dengan hati yang bersih
rm(list=ls())

# memanggil libraries
library(dplyr)
library(ompr)
library(ompr.roi)
library(ROI.plugin.glpk)

#  set vector profit
profit = c(1000,1200,900)

# membuat model
model =
  MIPModel() %>%
  # add variables
  # non negative integers
  add_variable(x[i],i = 1:3,
               type = "integer",
               lb = 0) %>%
  # set obj function
  set_objective(sum_expr(x[i]*profit[i],i = 1:3),
                "max") %>%
  # add constraints
  add_constraint(x[1] <= 500) %>%
  add_constraint(x[2] <= 300) %>%
  add_constraint(x[3] <= 400) %>%
  add_constraint(sum_expr(x[i],i = 1:3) == 1000)
model
```

```
## Mixed integer linear optimization problem
## Variables:
##   Continuous: 0
##   Integer: 3
##   Binary: 0
## Model sense: maximize
## Constraints: 4
```

1.3.2 *Solving*

Kemudian saya *solve* dengan **R**:

```
result = solve_model(model, with_ROI(solver = "glpk", verbose = TRUE))
```

```
## <SOLVER MSG>  ----
## GLPK Simplex Optimizer, v4.65
## 4 rows, 3 columns, 6 non-zeros
##       0: obj =  -0.000000000e+00 inf =   1.000e+03 (1)
##       3: obj =   1.040000000e+06 inf =   0.000e+00 (0)
## OPTIMAL LP SOLUTION FOUND
## GLPK Integer Optimizer, v4.65
## 4 rows, 3 columns, 6 non-zeros
## 3 integer variables, none of which are binary
## Integer optimization begins...
## Long-step dual simplex will be used
## +     3: mip =     not found yet <=              +inf        (1; 0)
## +     3: >>>>>   1.040000000e+06 <=   1.040000000e+06   0.0% (1; 0)
## +     3: mip =   1.040000000e+06 <=     tree is empty   0.0% (0; 1)
## INTEGER OPTIMAL SOLUTION FOUND
## <!SOLVER MSG> ----
```

1.3.3 *Final Result*

Saya dapatkan konfigurasi terbaik seperti ini:

```
result %>% get_solution(x[i])
```

```
##   variable i value
## 1        x 1   500
## 2        x 2   300
## 3        x 3   200
```

Dengan *profit* maksimum sebesar Rp1.04 juta.

2 Kasus *Optimization* II

2.1 Masalah

Di sebuah perusahaan, departemen *filling* dan *packing* memiliki tiga jenis mesin yang selalu beroperasi setiap harinya. Setiap mesin memiliki kapasitas, biaya proses per unit produk, dan biaya *setup* masing-masing.

Berikut adalah datanya:

Table 1: Data Mesin Filling dan Packing

mesin	biaya_setup	biaya_proses_unit	kapasitas
1	300	2	600
2	100	10	800
3	200	5	1200

Mengingat di setiap mesin harus ada pekerja yang ditugaskan untuk menjalankannya, manajemen mengambil keputusan bahwa jika suatu mesin digunakan, maka mesin tersebut paling sedikit harus memproses 400 unit produk.

Di suatu hari, terdapat beban kerja sebanyak 2000 unit produk yang harus diproses *filling* dan *packing*-nya.

> ***Berapa konfigurasi produk per mesin yang paling optimal?***

2.2 Model Matematika

Dari kasus di atas, kita akan membuat model matematikanya.

2.2.1 Variabel Penentuan

Misalkan saya notasikan 3 variabel berikut ini:

- x_1 sebagai seberapa banyak (dalam unit) produk yang dijalankan di mesin I,
- x_2 sebagai seberapa banyak (dalam unit) produk yang dijalankan di mesin II, dan
- x_3 sebagai seberapa banyak (dalam unit) produk yang dijalankan di mesin III.

Di mana:

$$x_1, x_2, x_3 \in \mathbb{Z}$$

2.2.2 *Constraints*

Berikut adalah *constraints* yang ada pada kasus di atas:

- $400 \leq x_1 \leq 600$
- $400 \leq x_2 \leq 800$
- $400 \leq x_3 \leq 1200$
- $x_1 + x_2 + x_3 = 2000$

2.2.3 *Objective Function*

Tujuan utama permodelan matematika ini adalah **meminimalkan** *cost* yang terjadi di semua mesin, yakni:

$$min((300 + 2x_1) + (100 + 10x_2) + (200 + 5x_3))$$

2.3 *Solver R*

2.3.1 Penulisan Model Matematika di R

Untuk menyelesaikan masalah ini, saya akan menggunakan *solver* di **R**. Berikut adalah model yang saya buat:

```
# dimulai dengan hati yang bersih
rm(list=ls())

# memanggil libraries
library(dplyr)
library(ompr)
library(ompr.roi)
library(ROI.plugin.glpk)

#  set vector fixed cost
fixed_cost = c(300000,100000,200000)

# set vector cost per unit
cost_per_unit = c(2000,10000,5000)

# membuat model
model =
  MIPModel() %>%
  # add variables
  # non negative integers
  add_variable(x[i],i = 1:3,
               type = "integer",
               lb = 400) %>%
  # set obj function
  set_objective((fixed_cost[1]+cost_per_unit[1]*x[1]) + (fixed_cost[2]+cost_per_unit[2]*x[2]) + (fixed_
                "min") %>%
  # add constraints
  add_constraint(x[1] <= 600) %>%
  add_constraint(x[2] <= 800) %>%
  add_constraint(x[3] <= 1200) %>%
  add_constraint(sum_expr(x[i],i = 1:3) == 2000)
model
```

```
## Mixed integer linear optimization problem
## Variables:
##   Continuous: 0
##   Integer: 3
##   Binary: 0
## Model sense: minimize
## Constraints: 4
```

2.3.2 *Solving*

Kemudian saya *solve* dengan **R**:

```
result = solve_model(model, with_ROI(solver = "glpk", verbose = TRUE))
```

```
## <SOLVER MSG>  ----
## GLPK Simplex Optimizer, v4.65
## 4 rows, 3 columns, 6 non-zeros
##       0: obj =   6.800000000e+06 inf =   8.000e+02 (1)
##       3: obj =   1.220000000e+07 inf =   0.000e+00 (0)
## *     4: obj =   1.020000000e+07 inf =   0.000e+00 (0)
## OPTIMAL LP SOLUTION FOUND
## GLPK Integer Optimizer, v4.65
## 4 rows, 3 columns, 6 non-zeros
## 3 integer variables, none of which are binary
## Integer optimization begins...
## Long-step dual simplex will be used
## +     4: mip =     not found yet >=              -inf        (1; 0)
## +     4: >>>>>   1.020000000e+07 >=   1.020000000e+07   0.0% (1; 0)
## +     4: mip =   1.020000000e+07 >=     tree is empty   0.0% (0; 1)
## INTEGER OPTIMAL SOLUTION FOUND
## <!SOLVER MSG> ----
```

2.3.3 *Final Result*

Saya dapatkan konfigurasi terbaik seperti ini:

```
result %>% get_solution(x[i])
```

```
##   variable i value
## 1        x 1   600
## 2        x 2   400
## 3        x 3  1000
```

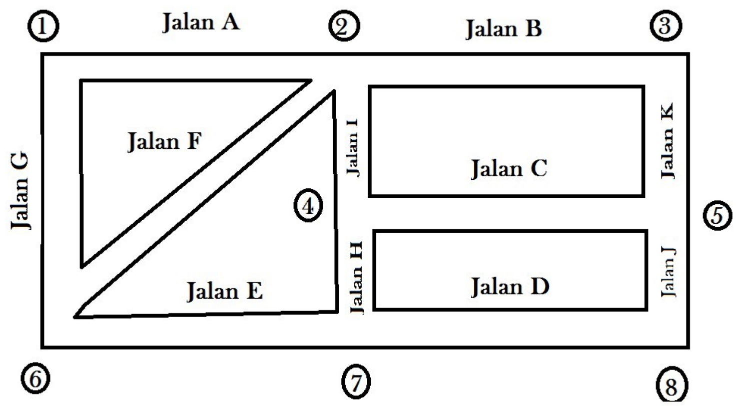
Dengan *cost* minimum sebesar Rp10.8 juta.

3 Kasus *Optimization* III

3.1 Masalah

Perhatikan gambar di bawah ini:

```
## [1] "Courtesy of: FINANMOS ITB 2021"
```



Suatu komplek perumahan dengan denah seperti di atas memiliki `11` jalan. Setiap pertemuan jalan, diberikan tanda nomor `1` hingga `8`. *Town management* hendak memasang lampu penerangan di **setiap pertemuan jalan tersebut**.

Tujuan utama mereka adalah memasang lampu sehingga semua jalan diterangi paling sedikit satu lampu.

> ***Di titik mana saja town management harus memasang lampu-lampu tersebut?***

3.2 Model Matematika

Dari kasus di atas, kita akan membuat model matematikanya.

3.2.1 Variabel Penentuan

Misalkan saya notasikan J sebagai himpunan titik-titik pertemuan jalan, yakni:

$$J = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

Kemudian saya akan tuliskan x_j sebagai *binary number* yang menyatakan apakah lampu dipasang atau tidak di titik $j \in J$.

$$x_j = \begin{cases} 1, & \text{jika di titik } j \text{ dipasang lampu} \\ 0, & \text{lainnya.} \end{cases}$$

Constraints

Dengan variabel keputusan seperti di atas, maka sesuai keinginan kita menerangi setiap jalan paling sedikit dengan satu lampu, kita mempunyai kendala:

- $x_1 + x_2 \geq 1$
- $x_2 + x_3 \geq 1$
- $x_1 + x_6 \geq 1$
- $x_2 + x_6 \geq 1$
- $x_2 + x_4 \geq 1$
- $x_4 + x_7 \geq 1$
- $x_4 + x_5 \geq 1$
- $x_7 + x_8 \geq 1$
- $x_3 + x_5 \geq 1$
- $x_6 + x_7 \geq 1$
- $x_5 + x_8 \geq 1$

3.2.2 *Objective Function*

Tujuan utama permodelan matematika ini adalah **meminimalkan** banyaknya titik yang dipasang lampu penerangan, yakni:

$$min(\sum x_j, j \in J)$$

3.3 *Solver R*

3.3.1 Penulisan Model Matematika di R

Untuk menyelesaikan masalah ini, saya akan menggunakan *solver* di **R**. Berikut adalah model yang saya buat:

```
# dimulai dengan hati yang bersih
rm(list=ls())

# memanggil libraries
library(dplyr)
library(ompr)
library(ompr.roi)
library(ROI.plugin.glpk)

# membuat model
model =
  MIPModel() %>%
  # add variables
  # binary
  add_variable(x[i],
               i = 1:8,
               type = "binary",
               lb = 0) %>%
  # set obj function
  set_objective(sum_expr(x[i],i = 1:8),
                "min") %>%
  # add constraints
  add_constraint(x[1] + x[2] >= 1) %>%
  add_constraint(x[2] + x[3] >= 1) %>%
  add_constraint(x[1] + x[6] >= 1) %>%
  add_constraint(x[2] + x[6] >= 1) %>%
```

```
  add_constraint(x[2] + x[4] >= 1) %>%
  add_constraint(x[4] + x[7] >= 1) %>%
  add_constraint(x[4] + x[5] >= 1) %>%
  add_constraint(x[7] + x[8] >= 1) %>%
  add_constraint(x[3] + x[5] >= 1) %>%
  add_constraint(x[6] + x[7] >= 1) %>%
  add_constraint(x[5] + x[8] >= 1)
model
```

```
## Mixed integer linear optimization problem
## Variables:
##   Continuous: 0
##   Integer: 0
##   Binary: 8
## Model sense: minimize
## Constraints: 11
```

3.3.2 *Solving*

Kemudian saya *solve* dengan **R**:

```
result = solve_model(model, with_ROI(solver = "glpk", verbose = TRUE))
```

```
## <SOLVER MSG>  ----
## GLPK Simplex Optimizer, v4.65
## 11 rows, 8 columns, 22 non-zeros
##       0: obj =   0.000000000e+00 inf =   1.100e+01 (11)
##       4: obj =   4.000000000e+00 inf =   0.000e+00 (0)
## *     9: obj =   4.000000000e+00 inf =   0.000e+00 (0)
## OPTIMAL LP SOLUTION FOUND
## GLPK Integer Optimizer, v4.65
## 11 rows, 8 columns, 22 non-zeros
## 8 integer variables, all of which are binary
## Integer optimization begins...
## Long-step dual simplex will be used
## +     9: mip =     not found yet >=              -inf        (1; 0)
## +     9: >>>>>   4.000000000e+00 >=   4.000000000e+00   0.0% (1; 0)
## +     9: mip =   4.000000000e+00 >=     tree is empty   0.0% (0; 1)
## INTEGER OPTIMAL SOLUTION FOUND
## <!SOLVER MSG> ----
```

3.3.3 *Final Result*

Saya dapatkan konfigurasi terbaik seperti ini:

```
result %>% get_solution(x[i]) %>% filter(value == 1)
```

```
##   variable i value
## 1        x 1     1
## 2        x 2     1
## 3        x 5     1
## 4        x 7     1
```

Dengan banyak lampu minimum terpasang sebanyak 4 buah.

4 Kasus *Optimization* IV

4.1 Masalah

Suatu perusahaan memproduksi 4 warna cat yaitu:

- Putih,
- Kuning,
- Hitam, dan
- Merah.

Keempat cat tersebut diproduksi di mesin-mesin yang sama, sehingga ada keperluan untuk mencuci mesin-mesin tersebut di antara produksi 2 cat yang berbeda warna.

Kita mempunyai masalah untuk menentukan urutan produksi cat harian yang *optimal*, yakni urutan produksi cat yang menghasilkan total waktu pencucian paling kecil.

> ***Urutan harian ini akan dipakai tiap hari, karena perusahaan setiap hari harus memproduksi keempat cat tersebut.***

Tabel berikut menampilkan waktu pencucian antara produksi cat di suatu baris jika akan dilanjutkan dengan cat di suatu kolom.

Table 2: Matriks Cleaning Mesin Cat (dalam menit)

	putih	kuning	hitam	merah
putih	~	10	17	15
kuning	20	~	19	18
hitam	50	44	~	25
merah	45	40	20	~

> ***Urutan produksi cat seperti apa yang meminimalkan waktu cleaning?***

4.2 Metode Heuristik

Sebenarnya masalah di atas mirip sekali dengan **Travelling Salesperson Problem**, yakni suatu masalah *optimization* yang mencari rute terpendek dari beberapa tempat.

Jadi alih-alih menggunakan metode *Mixed Integer Linear Programming* (**MILP**) yang biasa saya pakai untuk menyelesaikan *optimization*, saya akan menggunakan cara **TSP** saja.

4.2.1 *How to*

Langkah pertama adalah menyiapkan matriks cleaning terlebih dahulu, yakni dengan mengubah `data` yang berupa *dataframe* ke bentuk *matrix* di **R**.

```
data[is.na(data)] = 0
level = rownames(data)
matriks = as.matrix(data)
matriks
```

```
##        putih kuning hitam merah
## putih      0     10    17    15
## kuning    20      0    19    18
## hitam     50     44     0    25
## merah     45     40    20     0
```

Jika kita perhatikan dengan baik. Matriks *cleaning* di atas berbentuk asimetris. Artinya waktu *cleaning* dari cat 1 ke 2 **tidak sama** dengan waktu *cleaning* dari cat 2 ke 1.

Oleh karena itu, saya akan membuat *problem* **Assymetric TSP** (**ATSP**) untuk kemudian di-*solve*.

```
library(TSP)
problem = as.ATSP(matriks)
hasil = solve_TSP(problem)
hasil
```

```
## object of class 'TOUR'
## result of method 'arbitrary_insertion+two_opt' for 4 cities
## tour length: 98
```

Didapatkan waktu *cleaning* terkecil adalah sebesar 98 menit.

4.2.2 *Final Result*

Saya dapatkan urutan terbaik seperti ini:

```
paste(level[as.integer(hasil)],collapse = " - ")
```

```
## [1] "merah - hitam - putih - kuning"
```