Training Optimization Nutrifood x FINANMOS ITB

Sebuah Catatan Short Course

Ikanx Fadhli @nutrifood

26 Maret 2021

Contents

1 Pendahuluan

Catatan ini berisi *R Markdown* penyelesaian dari beberapa kasus yang diberikan pada saat ***optimization training*** oleh **FINANMOS ITB** 2021.

1.1 Catatan Penting Tentang *Optimization*

Optimization berarti proses pencarian suatu nilai yang **optimal**. Kondisi optimal bisa terjadi saat sesuatu bernilai **maksimum** atau **minimum**.

Hal tersebut yang harus kita pahami.

Permodelan matematika terkait *optimization* tidak lepas dari `4` hal berikut ini:

1. *Decision Variable*, yakni nilai yang ingin dicari. Diharapkan dari nilai ini akan tercipta kondisi yang optimal.
2. *Parameter*, yakni nilai yang besarannya *given*. Jika kita melihat pada kasus *real*, parameter adalah nilai yang tidak kita kontrol.
3. *Constraints*, yakni *boundaries* (limitasi) yang ada pada *problem* yang dihadapi. Bisa jadi dalam suatu kasus, kita membuat permodelan matematika yang tidak memiliki *constraints*.
4. *Objective function*, yakni kondisi optimal yang harus dipenuhi.

Parameter dan *decision variable* akan muncul pada *constraints* dan *objective function*.

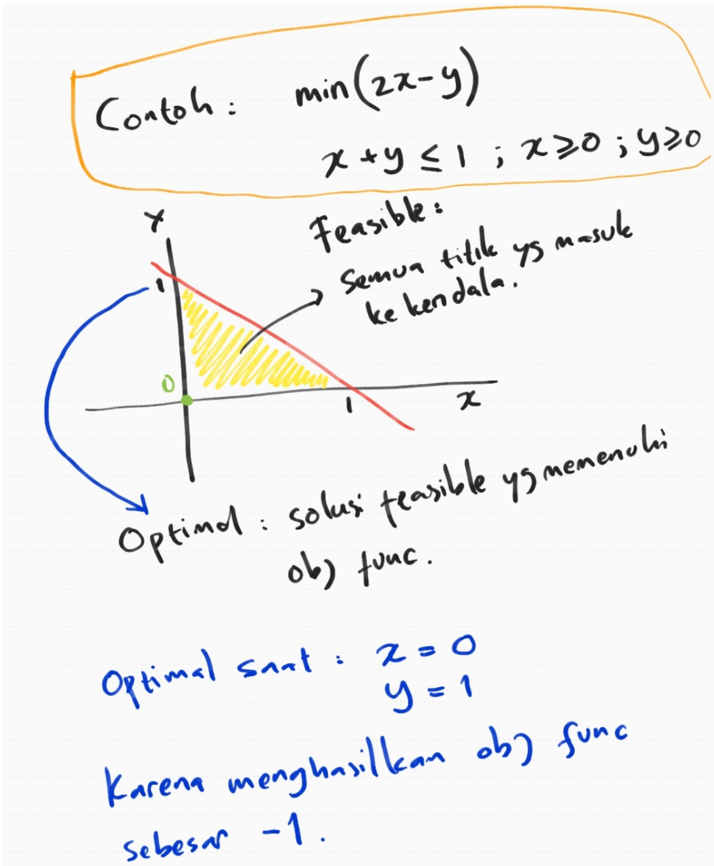
Suatu *decision variable* disebut *feasible* jika:

> ***Decision variable yang didapatkan tidak melanggar constraints.***

Suatu *decision variable* disebut optimal jika:

> ***Decision variable yang didapatkan memenuhi objective function.***

1.2 Contoh Kasus yang Dibahas



Saya menggunakan **R** untuk membuat model *optimization* dan di-*solve* dengan *engine* yang tersedia di `library(ompr)`. Tipe variabel penentuan yang didukung oleh `ompr` antara lain:

1. `binary`
2. `continuos`
3. `integer`

Agak berbeda dengan *libraries* yang akan digunakan saat *training* dengan **FINANMOS ITB** kelak namun tetap menghasilkan hasil yang sama. Karena persoalan yang pentingnya adalah bagaimana memodelkan masalah *real* ke permodelan matematikanya. Jika sudah termodelkan, akan mudah dimasukkan ke berbagai *libraries* yang ada di **R**.

2 Koperasi Susu Berkah

2.1 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.

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 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:

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

2.2.2 *Constraints*

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

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

2.2.3 *Objective Function*

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

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

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 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
```

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 =  -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> ----
```

2.3.3 *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.

3 Tiga Mesin Filling

3.1 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?***

3.2 Model Matematika

Dari kasus di atas, kita akan membuat model matematikanya.

3.2.1 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:

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

3.2.2 *Constraints*

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

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

3.2.3 ***Objective Function***

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

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

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)

#  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
```

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
## 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> ----
```

3.3.3 *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
```

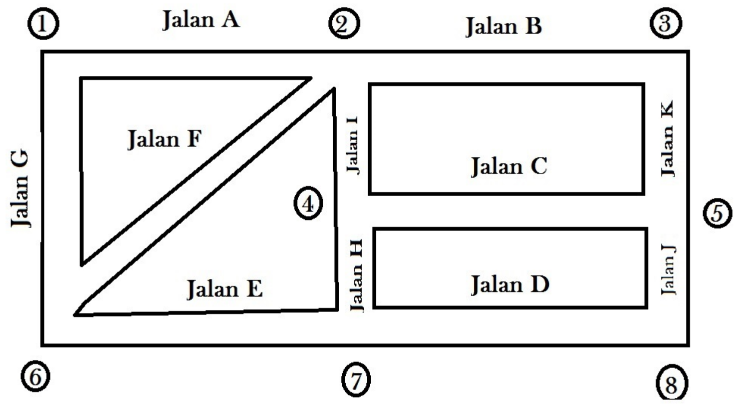
Dengan *cost* minimum sebesar Rp10.8 juta.

4 Lampu Penerangan Jalan

4.1 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?

4.2 Model Matematika

Dari kasus di atas, kita akan membuat model matematikanya.

4.2.1 Parameter

Misalkan saya notasikan Jl sebagai himpunan jalan, yakni:

$$Jl = \{A, B, C, D, E, F, G, H, I, J, K\}$$

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

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

4.2.2 Variabel Penentuan

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

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

Misalkan:

- $x_1 = 0$, artinya lampu di titik 1 **tidak dipasang lampu**.
- $x_2 = 1$, artinya lampu di titik 2 **dipasang lampu**.

4.2.3 *Constraints*

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

1. $x_1 + x_2 \geq 1$ untuk **Jalan A**.
2. $x_2 + x_3 \geq 1$ untuk **Jalan B**.
3. $x_1 + x_6 \geq 1$ untuk **Jalan G**.
4. $x_2 + x_6 \geq 1$ untuk **Jalan F**.
5. $x_2 + x_4 \geq 1$ untuk **Jalan I**.
6. $x_4 + x_7 \geq 1$ untuk **Jalan H**.
7. $x_4 + x_5 \geq 1$ untuk **Jalan C**.
8. $x_7 + x_8 \geq 1$ untuk **Jalan D**.
9. $x_3 + x_5 \geq 1$ untuk **Jalan K**.
10. $x_6 + x_7 \geq 1$ untuk **Jalan E**.
11. $x_5 + x_8 \geq 1$ untuk **Jalan J**.

4.2.4 *Objective Function*

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

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

4.3 *Solver R*

4.3.1 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
```

4.3.2 *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> ----
```

4.3.3 *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.

5 Perusahaan Cat

5.1 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?***

5.2 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.

5.2.1 *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.

5.2.2 *Final Result*

Saya dapatkan urutan terbaik seperti ini:

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

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

6 *Nurse Schedulling*

Di bagian ini kita akan mempelajari masalah penjadwalan perawat (*nurse scheduling*) yang mempunyai aturan kerja yang tidak terlalu banyak. Aturan kerja yang dibahas di sini mungkin saja merupakan aturan di suatu rumah sakit saja, yang sedikit berbeda dengan aturan kerja perawat di rumah sakit lainnya. Tetapi tujuan dibuat aturan kerja ini di rumah sakit manapun adalah sama, yaitu aturan yang dibuat agar jam kerja perawat diatur sedemikian hingga sehingga para perawat berada pada kondisi yang baik ketika bekerja.

6.1 Masalah

Lingkungan kerja para perawat yang kita bahas adalah sebagai berikut:

- Para perawat bekerja pada suatu shift kerja
- Terdapat 3 shift kerja yaitu:
  - *day shift* (8.00 - 16.00),
  - *evening shift* (16.00 - 24.00), dan
  - *night shift* (24.00 - 8.00)
- Pada setiap shift dibutuhkan 4 orang perawat.

Selain itu, terdapat beberapa aturan kerja perawat yang harus dipenuhi, yakni:

1. Setiap perawat dalam satu hari dapat ditugaskan paling banyak dalam satu *shift*.
2. Jika seorang perawat ditugaskan pada *shift* malam, maka dia tidak dapat ditugaskan di *shift* pagi di hari berikutnya.
3. Jika seorang perawat ditugaskan dalam 3 hari berturut-turut, maka hari keempatnya harus diberi libur.
4. Jika seorang perawat ditugaskan pada suatu *shift* di *weekend*, maka dia tidak dapat ditugaskan di *weekend* berikutnya.

Bagaimana jadwal yang optimal? Berapa banyak perawat yang dibutuhkan?

6.2 Model Matematika

6.2.1 Membangun Model Matematika

6.2.1.1 *Time Frame* Untuk memudahkan membuat model matematika *nurse schedulling*, saya akan mendefinisikan terlebih dahulu *time frame* yang hendak digunakan. Apakah akan dibuat jadwal selama seminggu, sebulan, atau periode tertentu.

Untuk itu, saya akan melihat **aturan kerja ke-4**, yakni:

> Jika seorang perawat ditugaskan pada suatu *shift* di *weekend*, maka dia tidak dapat ditugaskan di *weekend* berikutnya.

Dari kasus di atas, setidaknya *time frame* penjadwalan **tersingkat** yang bisa buat adalah dalam waktu 2 minggu.

Dari *time frame* tersebut, kita juga bisa mengandaikan berapa banyak perawat yang dibutuhkan.

6.2.1.2 Berapa banyak perawat yang dibutuhkan? Dari penjelasan kasus di atas, sebenarnya tidak ada batasan maksimal berapa perawat yang bisa ditugaskan di rumah sakit tersebut. Namun, dari **aturan kerja ke-4** kita bisa hitung secara kasar berapa minimal perawat yang harus ditugaskan.

Bagaimana caranya?

Mari kita ingat beberapa hal berikut ini:

1. *Weekend* terjadi pada hari Sabtu dan Minggu.
2. Setiap hari ada 3 *shifts*.
3. Setiap *shift* minimal ada 4 perawat.
4. Perawat yang sudah ditugaskan di *weekend* I, tidak boleh ditugaskan di *weekend* II.

Oleh karena itu, pada *weekend* I paling sedikit kita bisa menugaskan $3 * 4 = 12$ orang perawat.

Pada *weekend* I, kita bisa mempekerjakan 12 perawat pada Sabtu dan Minggu.

Ke-12 orang perawat ini tidak boleh ditugaskan di *weekend* II. Sehingga dibutuhkan $3*4 = 12$ orang perawat lainnya di *weekend* II.

Sehingga dibutuhkan minimal 24 orang perawat untuk 2 minggu ini.

6.2.2 Parameter

Dari penjelasan-penjelasan di atas, saya akan mendefinisikan beberapa hal, yakni:

- $H = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$ adalah himpunan hari dalam *time frame* 2 minggu. Saya tuliskan 1 sebagai Senin. *Weekend* terjadi pada $H_w = \{6, 7, 13, 14\}$.
- $S = \{1, 2, 3\}$ adalah himpunan *shift* kerja harian perawat.
- $N = \{1, 2, 3, 4, ..., 24\}$ adalah himpunan banyaknya perawat yang dibutuhkan. Pada mulanya, saya akan *set* dulu sebanyak 24 orang perawat. Jika ternyata tidak *feasible*, maka akan saya tambah satu demi satu sehingga ditemukan solusi *feasible*.

6.2.3 Variabel Penentuan

Saya definisikan:

$$x_{n,h,s} = \begin{cases} 1, & \text{jika di perawat ke } n \text{ bekerja di hari } h \text{ pada shift ke } s \\ 0, & \text{lainnya.} \end{cases}$$

6.2.4 *Constraints*

Sekarang kita akan menuliskan *constraints* dalam bahasa matematika.

6.2.4.1 *Constraint* I Setiap perawat dalam satu hari dapat ditugaskan paling banyak dalam satu *shift*.

$$x_{n,h,1} + x_{n,h,2} + x_{n,h,3} \leq 1, \text{ untuk } n \in N \text{ dan } h \in H$$

Ekspresi matematika di atas memastikan bahwa seorang perawat hanya bisa ditugaskan dalam **satu shift** per hari **atau** tidak ditugaskan sama sekali.

6.2.4.2 *Constraint* II

Jika seorang perawat ditugaskan pada *shift* malam, maka dia tidak dapat ditugaskan di *shift* pagi.

$$x_{n,h,3} + x_{n,h+1,1} \leq 1, \text{ untuk } n \in N \text{ dan } h \in \{1, 2, .., 13\}$$

Ekspresi matematika di atas memastikan bahwa seorang perawat yang bertugas *night shift* pada hari h tidak boleh bertugas pada *shift* pagi esok harinya $(h + 1)$ **atau** perawat tersebut tidak ditugaskan sama sekali.

6.2.4.3 *Constraint* III

Jika seorang perawat ditugaskan dalam 3 hari berturut-turut, maka hari keempatnya harus diberi libur.

Jadi seorang perawat bisa saja bertugas di berbagai *shift* selama 3 hari berturut-turut tapi **tidak diperbolehkan** untuk bertugas di hari keempat.

$$x_{n,h,1} + x_{n,h+1,1} + x_{n,h+2,1} + x_{n,h+3,1}+$$

$$x_{n,h,2} + x_{n,h+1,2} + x_{n,h+2,2} + x_{n,h+3,2}+$$

$$x_{n,h,3} + x_{n,h+1,3} + x_{n,h+2,3} + x_{n,h+3,3} \leq 3, \text{ untuk } n \in N \text{ dan } h \in \{1, 2, .., 11\}$$

6.2.4.4 *Constraint* IV

Jika seorang perawat ditugaskan pada suatu *shift* di *weekend*, maka dia tidak dapat ditugaskan di *weekend* berikutnya.

Saya telah menuliskan *weekend* terjadi saat $H \in \{6, 7, 13, 14\}$.

Bagi saya, *constraint* IV merupakan *constraint* yang tersulit untuk dituliskan model matematikanya. Oleh karena itu, saya akan gunakan induksi sebagai berikut:

6.2.4.4.1 Tanggal 6

Jika seorang perawat bertugas di hari 6 (*shift* apapun), dia tidak boleh bertugas di hari 13 dan 14. Tapi jika dia tidak bertugas di hari 6, maka dia diperbolehkan bertugas di hari 13 **dan atau** 14. Akibatnya:

- Jika $x_{n,6,s} = 1$ maka $x_{n,13,s} + x_{n,14,s} = 0$
- Jika $x_{n,6,s} = 0$ maka $x_{n,13,s} + x_{n,14,s} \leq 2$ karena perawat tersebut bisa bertugas di tanggal 13 **dan atau** 14.

Maka model matematika pada *constraint* tanggal 6 adalah sebagai berikut:

$$2 \sum_{s \in S} x_{n,6,s} + \sum_{s \in S} x_{n,13,s} + \sum_{s \in S} x_{n,14,s} \leq 2, \text{ untuk } n \in N$$

6.2.4.4.2 Tanggal 7

Dengann prinsip yang sama dengan tanggal 6, saya bisa dapatkan model matematika pada *constraint* tanggal 7 adalah sebagai berikut:

$$2 \sum_{s \in S} x_{n,7,s} + \sum_{s \in S} x_{n,13,s} + \sum_{s \in S} x_{n,14,s} \leq 2, \text{ untuk } n \in N$$

6.2.4.5 *Constraint* V Ada satu *constraint* terakhir yang kita tidak boleh lupakan. Apa itu? Setiap *shift* harus dilayani minimal 4 orang perawat.

$$\sum_{n \in N} x_{n,h,s} \geq 4, \text{ untuk } h \in H, \text{ dan } s \in S$$

6.2.5 *Objective Function*

$$min \sum_{n \in N} \sum_{h \in H} \sum_{s \in S} x_{n,h,s}$$

6.3 *Solver R*

6.3.1 Penulisan Model Matematika di R

Berikut adalah penulisan model matematika di **R**:

```
# 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
  # non negative integers
  add_variable(x[n,h,s],
               n = 1:24,
               h = 1:14,
               s = 1:3,
               type = "binary",
               lb = 0) %>%
  # set obj function
  set_objective(sum_expr(x[n,h,s],
                         n = 1:24,
                         h = 1:14,
                         s = 1:3),
                "min") %>%
  # add constraints
  # constraint I
  add_constraint(x[n,h,1] + x[n,h,2] + x[n,h,3] <= 1,
                 n = 1:24,
                 h = 1:14) %>%
  # constraint II
  add_constraint(x[n,h,3] + x[n,h+1,1] <= 1,
                 n = 1:24,
                 h = 1:13) %>%
  # constraint III
  add_constraint(x[n,h,1] + x[n,h+1,1] + x[n,h+2,1] + x[n,h+3,1] +
                 x[n,h,2] + x[n,h+1,2] + x[n,h+2,2] + x[n,h+3,2] +
                 x[n,h,3] + x[n,h+1,3] + x[n,h+2,3] + x[n,h+3,3] <= 3,
                 n = 1:24,
                 h = 1:11) %>%
  # constraint IV tanggal 6
  add_constraint(2*(x[n,6,1] + x[n,6,2] + x[n,6,3]) +
                 sum_expr(x[n,13,s],
                          s = 1:3) +
                 sum_expr(x[n,14,s],
                          s = 1:3) <= 2,
                 n = 1:24) %>%
```

```
  # constraint IV tanggal 7
  add_constraint(2*(x[n,7,1] + x[n,7,2] + x[n,7,3]) +
                 sum_expr(x[n,13,s],
                          s = 1:3) +
                 sum_expr(x[n,14,s],
                          s = 1:3) <= 2,
                 n = 1:24) %>%
  # contraint V
  add_constraint(sum_expr(x[n,h,s],
                          n = 1:24) >= 4,
                 h = 1:14,
                 s = 1:3)
model
```

```
## Mixed integer linear optimization problem
## Variables:
##   Continuous: 0
##   Integer: 0
##   Binary: 1008
## Model sense: minimize
## Constraints: 1002
```

6.3.2 *Solving*

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

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

```
## <SOLVER MSG>  ----
## GLPK Simplex Optimizer, v4.65
## 1002 rows, 1008 columns, 6240 non-zeros
##       0: obj =   0.000000000e+00 inf =   1.680e+02 (42)
##     264: obj =   1.680000000e+02 inf =   0.000e+00 (0)
## Perturbing LP to avoid stalling [367]...
## Removing LP perturbation [410]...
## *   410: obj =   1.680000000e+02 inf =   0.000e+00 (0)
## OPTIMAL LP SOLUTION FOUND
## GLPK Integer Optimizer, v4.65
## 1002 rows, 1008 columns, 6240 non-zeros
## 1008 integer variables, all of which are binary
## Integer optimization begins...
## Long-step dual simplex will be used
## +   410: mip =     not found yet >=              -inf        (1; 0)
## +   410: >>>>>   1.680000000e+02 >=   1.680000000e+02   0.0% (1; 0)
## +   410: mip =   1.680000000e+02 >=     tree is empty   0.0% (0; 1)
## INTEGER OPTIMAL SOLUTION FOUND
## <!SOLVER MSG> ----
```

6.4 Solusi Optimal

6.4.1 Jadwal Optimal

Berikut adalah jadwal optimal dalam kasus ini:

Table 3: Jadwal Perawat: Angka Pada Tanggal Menunjukkan id Perawat

tanggal	Pagi	Siang	Malam
1	1,2,3,4	5,6,7,8	9,10,11,12
2	1,2,3,4	5,6,7,8	13,14,15,16
3	1,2,3,4	9,10,11,12	13,14,15,16
4	5,6,7,8	9,10,11,12	13,14,15,16
5	1,2,3,4	5,6,7,8	17,18,19,20
6	1,2,3,4	9,10,11,12	17,18,19,20
7	1,2,3,4	9,10,11,12	17,18,19,20
8	5,6,7,8	9,10,11,12	13,14,15,16
9	1,2,3,4	5,6,7,8	13,14,15,16
10	1,2,3,4	5,6,7,8	9,10,11,12
11	1,2,3,4	9,10,11,12	13,14,15,16
12	5,6,7,8	9,10,11,12	17,18,19,20
13	5,6,7,8	13,14,15,16	21,22,23,24
14	5,6,7,8	13,14,15,16	21,22,23,24

6.4.2 Rekap Jadwal Optimal

Berikut adalah rekapan jadwal optimal per perawat:

Table 4: Rekapan Berapa Kali Perawat Bertugas

id_perawat	Pagi	Siang	Malam
1	9	~	~
2	9	~	~
3	9	~	~
4	9	~	~
5	5	5	~
6	5	5	~
7	5	5	~
8	5	5	~
9	~	7	2
10	~	7	2
11	~	7	2
12	~	7	2
13	~	2	6
14	~	2	6
15	~	2	6
16	~	2	6
17	~	~	4
18	~	~	4
19	~	~	4
20	~	~	4
21	~	~	2
22	~	~	2
23	~	~	2
24	~	~	2

Setelah kita lihat bersama, ternyata ada beberapa perawat yang **hanya** mendapatkan porsi kecil dalam jadwal tersebut. Lantas muncul pertanyaan:

> ***Apakah kita bisa meratakan workload antar perawat?***