Training Optimization Nutrifood x FINANMOS ITB

Sebuah Catatan Short Course

Ikanx Fadhli @nutrifood

19 April 2021

Contents

Kata Pengantar

Alhamdulillah, *training optimization* telah selesai dilakukan. Peserta merupakan *Nutrifooders* dari tim *market research* dan beberapa tim *member* dari departemen lain yang terkait.

Diharapkan *training* ini memberikan wawasan baru terkait penggunaan metode matematika untuk menyelesaikan masalah *optimization* yang terjadi di banyak area kerja Nutrifood.

Tentunya kalian mungkin menemukan *typo* pada materi ini. Beberapa materi awal merupakan *copy-paste* langsung dari materi **FinanMos** sedangkan pada kasus-kasus merupakan penulisan ulang dari materi **FinanMos**.

Pada beberapa kasus, kalian mungkin juga menemukan perbedaan cara saya menuliskan indeks pada *decision variable*. Itu tidak menjadi masalah karena notasi bisa berbeda tergantung cara penulisan.

Semoga bisa berguna.

Bekasi, 19 April 2021.

Pengepul materi *training*,

Ikang Fadhli

1 Pendahuluan

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

1.1 Tentang *Training* Ini

Training **Pengantar Optimisasi** atau **Riset Operasi** ini adalah training dasar di bidang **Optimisasi** atau **Riset Operasi** yang dapat memberikan wawasan apakah **Optimisasi** atau **Riset Operasi** itu dan perannya dalam penyelesaian masalah-masalah nyata di industri dan pemerintahan.

Penekanan *training* ini adalah pada:

1. Proses pemodelan matematika, yaitu proses abstraksi dari masalah nyata menjadi masalah dalam ekspresi matematika,proses penyelesaian model-model matematika tersebut dengan bantuan optimization *toolbox* yang tersedia di *Phyton* atau **R**.
2. Teknik-teknik penyelesaian matematika tidak ditekankan di sini, karena itu bagi peserta kuliah yang ingin memperoleh pengetahuan bagaimana teknik-teknik tersebut dibangun, dapat mengikuti kuliah-kuliah lain pada kuliah serial **Optimisasi** atau **Riset Operasi** ini.

Optimisasi atau **Riset Operasi** sangat terkait dalam pengambilan keputusan-keputusan strategis di industri dan pemerintahan. Dengan adanya teknologi internet, tingkat kompetisi di industri menjadi sangat tinggi. Perusahaan-perusahaan atau institusi pemerintah harus membuat keputusan-keputusan strategis dengan cepat, agar tetap dapat bersaing dan mendapatkan keuntungan seperti yang diinginkan. Untuk dapat membuat keputusan strategis dalam waktu yang cepat, dibutuhkan tim pembuat keputusan yang semua personilnya mempunyai ilmu dan pengalaman yang cukup tentang masalah yang terkait dengan keputusan yang ingin dibuat.

Di beberapa perusahaan multi nasional besar, tim tersebut biasanya melibatkan tenaga yang mampu menerapkan **Optimisasi** atau **Riset Operasi**, dan penerapan riset operasi ini terbukti dapat memberikan efisiensi biaya yang sangat signifikan. Wawasan ilmu **Optimisasi** atau **Riset Operasi** dari kuliah ini diharapkan dapat memberikan dasar ilmu bagi peserta kuliah untuk mempersiapkan diri menjadi peneliti atau praktisi.

1.2 Definisi Optimisasi atau Riset Operasi

Riset Operasi, yang merupakan terjemahan dari ***Operations Research***, adalah metode antar disiplin ilmu yang digunakan untuk menganalisa masalah nyata dan membuat keputusan untuk kegiatan operasional suatu organisasi atau perusahaan. Disiplin-displin ilmu yang sering digunakan dalam metode ini adalah pemodelan matematika, optimisasi, statistika, algoritma/metoda numerik. Organisasi atau perusahaan yang sering menerapkan riset operasi biasanya memiliki sistem operasi yang kompleks dengan sumber daya yang terbatas, sehingga riset operasi diharapkan dapat memberikan informasi yang cukup dan rasional dalam membuat keputusan-keputusan strategis di organisasi atau perusahaan.

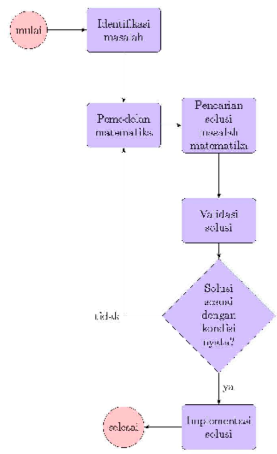
Riset Operasi juga dikenal dengan nama lain. Di Britania Raya, ilmu ini dikenal dengan nama ***Operational Research***. Ilmu ini sering juga dinamakan ***Management Science***, ***System Analysis***, ***Cost-Benefit Analysis***, dan ***Cost-Effectiveness Analysis***. Tetapi ***Operations Research*** dan ***Management Science*** adalah dua nama yang paling banyak ditemui dalam buku atau artikel ilmiah lainnya.

Dalam menerapkan Riset Operasi, masalah nyata dianalisa dan diselesaikan dalam tahapan berikut ini.

1. Proses identifikasi atau penajaman masalah yang akan diselesaikan.
2. Pemodelan masalah menjadi suatu masalah matematika, dalam arti masalah nyata dituliskan dalam sejumlah variabel dan paramater.

3. Pencarian solusi atau penyelesaian dari masalah matematika.
4. Pengujian dari solusi matematika apakah sudah memenuhi seluruh kondisi yang ada di dunia nyata. Jika masih ada kekurangan, maka kembali ke tahap 2 untuk penyempurnaan model matematika.
5. Penerapan solusi matematika untuk mendapatkan solusi masalah nyatanya.

Tahapan tersebut dapat dilihat di diagram alir berikut ini.



1.3 Sejarah Riset Operasi

Metode riset operasi mulai berkembang pesat beberapa saat sebelum **Perang Dunia Kedua** dimulai, yaitu pada tahun 1937 saat Britania Raya mengantisipasi perang udara (tetapi sebagian buku/artikel mengatakan bahwa sebenarnya riset operasi sudah dimulai di akhir Perang Dunia Pertama). Untuk proses antisipasi tersebut, mereka harus mengeksplorasi data-data yang ditunjukkan oleh alat baru, yang sekarang kita kenal dengan istilah radar, yang mereka punyai untuk diubah menjadi informasi penugasan kru dan penggunaan pesawat tempur. Britania Raya menugaskan satu tim multi disiplin untuk melaksanakan riset dalam pengambilan keputusan untuk penugasan kru dan pesawat tempur ini, berdasarkan kondisi operasional radar dan pesawat tempur ini, dan dari sinilah nama ***Operations Research*** ini muncul.

Tim multi disiplin pada awalnya mempunyai tujuan untuk memahami sistem operasi radar dan pesawat tempur yang terdiri dari peralatan, kru, dan kondisi lingkungan tempat operasi seperti cuaca dan waktu operasi (siang atau malam). Pemahaman atas sistem operasi ini kemudian memberikan sejumlah ide untuk meningkatkan kinerja dari sistem. Kerja keras yang dilakukan oleh tim ini telah memberikan hasil yang sangat signifikan aitu kemenangan Britania Raya pada perang udara di Perang Dunia Kedua dan meluasnya penggunaan riset operasi di divisi militer lain tidak hanya di angkatan udara. Bahkan beberapa orang dari tim ini dinobatkan sebagai pemenang hadiah Nobel atas ide-ide orsinil mereka sehingga tercipta kesuksesan Britania Raya dalam perang udara di **Perang Dunia Kedua**.

Tim riset multi disiplin ini juga dibentuk di angkatan bersenjata Amerika Serikat. Tugas tim ini adalah melindungi konvoy tentara, mencari konvoy tentara musuh, meningkatkan perang anti kapal selam, dan meningkatkan efektifitas dari pengeboman. Ada kesamaan dari usaha tim riset angkatan bersenjata Britania Raya dan Amerika Serikat, yaitu:

1. Pengumpulan data,
2. Observasi langsung dari operasi sistem,
3. Pembuatan model matematika,
4. Penentuan rekomendasi untuk peningkatan kinerja sistem,

5. Penentuan *feedback* atas dampak dari setiap perubahan yang dialami sistem.

Kelima tahapan tersebut dilaksanakan dengan tujuan agar agar tim dapat melihat segala sesuatunya berjalan di kondisi nyata sehingga tim dapat memberikan peningkatan kinerja dari sistem. Penerapan riset operasi di angkatan bersenjata Britania Raya dan Amerika Serikat telah memberikan peranan penting dalam penentuan strategi jangka panjang, pembuatan senjata, dan pengelolaan logistik. Tidaklah heran jika kita dapat menemukan ***Center for Operation Research*** di ***USA National Security Agency***.

Mulai tahun 1950an, penerapan riset operasi telah merebak tidak hanya di angkatan bersenjata suatu negara, tetapi juga di perusahaan-perusahaan swata dan instansi pemerintah. Perusahaan swasta pertama yang menerapka riset operasi adalah perusahaan petrokimia, yang menginginkan peningkatan kinerja pabriknya, mengembangkan sumber-sumber alam dan merencanakan strategi perusahaan. Saat ini, riset operasi telah memerankanperan yang sangat penting di berbagai industri, seperti:

1. Industri penerbangan, digunakan untuk penjadwalan pesawat dan kru (*cockpit crew*, *cabin crew*, *ground handling crew*), sistem reservasi, perencanaan pelatihan kru, perencanaan maintenance pesawat dan logistik, dan perencanaan di fasilitas pelayanan lainnya
2. Perusahaan logistik dan transportasi, digunakan terutama untuk *routing* dan *planning*,
3. Industri minyak dan gas, digunakan terutama untuk perencanaan produksi, peningkatan efisiensi produksi, distribusi dan transportasi, serta *supply chain management*,
4. Industri manufaktur, industri ***Fast Moving Consumer Good***, industri lainnya, digunakan terutama untuk penjadwalan mesin, penjadwalan pekerja dan *supply chain management*,
5. Rumah Sakit dan institusi pelayanan kesehatan lainnya, terutama untuk penjadwalan tenaga medis, perencanaan penyediaan alat medis dan obat-obatan,
6. Industri pariwisata (hotel, *restaurant*, *travel agent*), terutama untuk perencanaan dan penjadwalan,
7. Pemerintahan, terutama untuk penentuan rencana-rencana strategis pemerintah,

Seiring dengan berjalannya waktu, terdapat perubahan yang perlu kita catat bahwa yang tadinya tim riset operasi ini merupakan tim multi disiplin dengan banyak sekali disiplin ilmu yang terlibat menjadi suatu tim yang lebih memfokuskan diri pada pemodelan matematika untuk meningkatkan atau bahkan mengoptimalkan kinerja dari sistem di dunia nyata.

1.4 Peran Model Matematika

Model matematika analitik, yang selanjutnya akan kita singkat menjadi **model matematika**, dalam penerapan riset operasi tidak lain adalah abstraksi atau simplifikasi dari objek nyata, proses nyata, atau sistem nyata yang sedang diteliti. Kata matematika ditambahkan setelah kata model karena model dipresentasikan dalam struktur matematika seperti persamaan, ketaksamaan, fungsi, atau matriks. Dengan model matematika inilah peneliti atau praktisi riset operasi bekerja sama dengan para pengambil keputusan untuk memberikan keputusan terbaik untuk menyelesaikan masalah. Secara rinci, model matematika dibangun untuk:

1. Membuat tujuan penyelesaian masalah lebih jelas,
2. Mengkuantifikasi suatu isu atau ukuran tertentu di sistem,
3. Mengidentifikasi variabel keputusan yang dapat dibuat (yang dimaksud dengan variabel keputusan adalah ekspresi dari tindakan yang dapat diambil),
4. Mengekspresikan semua kendala atau aturan yang terkandung di dalam sistem,
5. Membuat komunikasi lebih mudah,

Di mana intuisi dari pembuat model dalam menyelesaikan masalah akan terekspresikan dalam model matematika ini.

Penggunaan model matematika telah teruji memberikan solusi masalah yang membutuhkan biaya relatif murah dan tidak perlu melibatkan sumber daya di sistem selama pencarian solusi dari masalah yang dihadapi.

Artinya, selama pembuatan model matematika dan pencarian solusinya, sistem dapat bekerja seperti sedia kala pada tingkat produktifitas yang ada. Berbeda halnya jika pendekatannya adalah melakukan simulasi dengan melibatkan sumber daya sistem. Ada kemungkinan ketika simulasi berjalan, produkstifitas dari sistem sempat mengalami penurunan, yang berarti ada *financial loss* akibat simulasi tersebut.

1.5 Catatan Penting *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.***

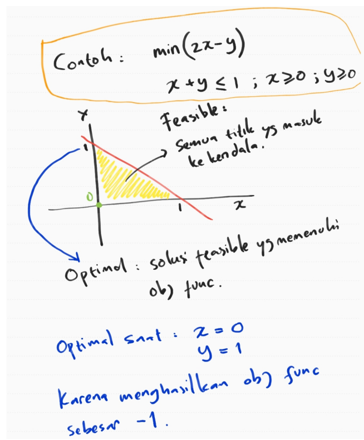
1.6 Kelas-Kelas Masalah Optimisasi

Model-model matematika di bidang riset operasi dapat dibagi menjadi beberapa kelas, berdasarkan kriteria tertentu. Berikut ini kita akan berkenalan dengan klasifikasi dari berdsarkan beberapa kriteria yang paling banyak dibahas.

- Kriteria pertama adalah kriteria berdasarkan karakteristik nilai dari parameter dan variabel yang terkandung di dalam model. Berdasarkan kriteria ini, model matematika dibagi menjadi:
  - Model **deterministik**; adalah model matematika di mana nilai dari semua parameter dan variabel yang terkandung di dalam model merupakan satu nilai pasti. Sebagai contoh, jika jadwal keberangkatan dan kepergian pesawat di suatu bandara sudah ditetapkan oleh Dinas Perhubungan dan banyaknya *cockpit crew* sudah diketahui dengan pasti, maka kita dapat membuat model matematika deterministik untuk penugasan cockpit crew berdasarkan jadwal yang sudah pasti ini.
  - Model **stokastik**; adalah model matematika di mana terdapat ketidak-pastian dari nilai satu atau lebih parameter atau variabelnya. Contoh dari model stokastik adalah model sistem antrian, di mana kedatangan pelanggan ke sistem antrian dan atau waktu pelayanan tidak kita ketahui dengan pasti. Sistem antrian sering kita temui di tempat pelayanan publik seperti di bandara, rumah sakit, atau sistem antrian untuk permintaan printing dokumen di suatu kantor di mana komputer-komputer dari para pegawai di kantor tersebut terhubung ke satu atau beberapa printer dan *Local Area Network*.

- Kriteria kedua adalah kriteria berdasarkan bagaimana variabel yang muncul atau tampil di dalam model. Berdasarkan kriteria ini, model matematika dibagi menjadi:
  - Model Linear. Suatu model matematika dinamakan model linear atau *linear programming* jika fungsi objektifnya berupa fungsi linear dan semua persamaan atau pertaksamaan di kendala model matematika tersebut adalah persamaan atau pertaksamaan linear.
  - Model Non-Linear. Suatu model matematika dinamakan model non-linear atau *non-linear programming* jika minimal salah satu dari fungsi objektif, persamaan atau pertaksamaan di kendala model matematika tersebut adalah persamaan atau pertaksamaan non-linear.

1.7 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 I

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 Koperasi Susu Berkah II

Berikut adalah modifikasi dari permasalah di Koperasi Susu Berkah.

3.1 Masalah

Dari Koperasi Susu Berkah tersebut, sebenarnya untuk penjualan susu cair ada resiko tidak terjualnya keseluruhan susu cair pada hari yang sama sebesar `10%`. Setiap susu yang tidak terjual ini akan memberikan kerugian sebesar `Rp500` per liternya.

> ***Berapa profit maksimal yang masih kita peroleh saat resiko tidak terjualnya susu cair terburuk?***

3.2 Model Matematika

Dari kasus di atas, kita cukup memodifikasi model matematika yang *existing*.

3.2.1 Variabel Penentuan

Saya akan definisikan variabel baru x_4, yakni berapa banyak susu cair yang tidak terjual.

3.2.2 *Objective Function*

Sekarang, *objective function*-nya berubah menjadi:

$$min(1000x_1 + 1200x_2 + 900x_3 - 500x_4)$$

Kenapa dibuat *min*? Karena kita ingin menghitung profit terbaik saat resiko terburuk.

3.2.3 *Constraint*

Sekarang saya tambahkan satu *constraint* terkait x_4.

$$0 \leq x_4 \leq 0.1x_1$$

dan

$$x_1 + x_2 + x_3 + x_4 = 1000$$

3.3 *Solver* R

3.3.1 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)

#  set vector profit
profit = c(1000,1200,900,-500)

# membuat model
model =
  MIPModel() %>%
  # add variables
  # non negative integers
  add_variable(x[i],i = 1:4,
               type = "continuous",
               lb = 0) %>%
  # set obj function
  set_objective(sum_expr(x[i]*profit[i],i = 1:4),
                "min") %>%
  # add constraints
  add_constraint(sum_expr(x[i],i = 1:4) == 1000) %>%
  add_constraint(x[1] <= 500) %>%
  add_constraint(x[2] <= 300) %>%
  add_constraint(x[3] <= 400) %>%
  add_constraint(x[4] - .1*x[1] <= 0)
model
```

```
## Mixed integer linear optimization problem
## Variables:
##   Continuous: 4
##   Integer: 0
##   Binary: 0
## Model sense: minimize
## Constraints: 5
```

3.3.2 *Solving*

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

```
## <SOLVER MSG>  ----
## GLPK Simplex Optimizer, v4.65
## 5 rows, 4 columns, 9 non-zeros
##       0: obj =   0.000000000e+00 inf =   1.000e+03 (1)
##       3: obj =   1.040000000e+06 inf =   0.000e+00 (0)
## *     5: obj =   8.950000000e+05 inf =   0.000e+00 (0)
## OPTIMAL LP SOLUTION FOUND
## <!SOLVER MSG> ----
```

3.3.3 *Final Result*

Berikut adalah *final result*-nya:

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

```
##   variable i value
## 1        x 1   500
## 2        x 2    50
## 3        x 3   400
## 4        x 4    50
```

```
result$objective_value
```

```
## [1] 895000
```

4 Tiga Mesin Filling

4.1 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?***

4.2 Model Matematika

Dari kasus di atas, kita akan membuat model matematikanya.

4.2.1 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:

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

4.2.2 *Constraints*

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

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

4.2.3 *Objective Function*

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

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

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)

#  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
```

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
## 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> ----
```

4.3.3 *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
```

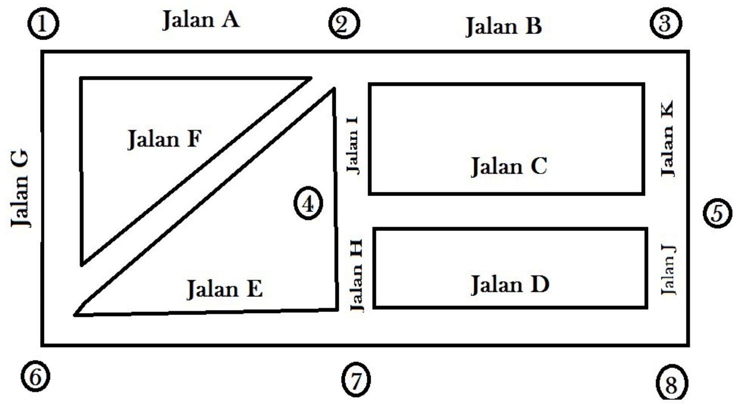
Dengan *cost* minimum sebesar Rp10.8 juta.

5 Lampu Penerangan Jalan

5.1 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?

5.2 Model Matematika

Dari kasus di atas, kita akan membuat model matematikanya.

5.2.1 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:

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

5.2.2 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**.

5.2.3 *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**.

5.2.4 *Objective Function*

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

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

5.3 *Solver R*

5.3.1 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)
```

```r
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
```

5.3.2 *Solving*

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

```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> ----
```

5.3.3 *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.

6 Perusahaan Cat

6.1 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?***

6.2 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.

6.2.1 *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.

6.2.2 *Final Result*

Saya dapatkan urutan terbaik seperti ini:

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

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

6.3 Metode Eksak

Sekarang kita akan menyelesaikan persoalan urutan cat ini dengan metode eksak dengan `ompr`.

6.3.1 Model Matematika

Untuk melakukannya saya akan mendefinisikan beberapa hal sebagai berikut:

6.3.1.1 Parameter Misal saya tuliskan:

- $W = \{1, 2, 3, 4\}$ sebagai himpunan warna cat yang diproduksi. Angka 1 menunjukkan putih, 2 menunjukkan kuning, 3 menunjukkan hitam, dan 4 menunjukkan merah.
- $c_{i,j}$ menunjukkan waktu *cleaning* antara produksi cat warna ke i dan j, $i, j \in W$.

6.3.1.2 Variabel Keputusan Misal saya tuliskan:

$$x_{i,j} = \begin{cases} 1, & \text{jika pabrik memproduksi cat ke } i \text{ dilanjutkan cat ke } j \\ 0, & \text{lainnya.} \end{cases}$$

6.3.1.3 ***Constraints*** Mari kita bangun beberapa *constraints* dari kasus ini.

Constraint pertama adalah satu warna hanya bisa diikuti oleh satu warna yang lain. Maksudnya dari warna ke i, hanya bisa diikuti *unique* oleh warna ke j. Saya tuliskan ekspresinya menjadi:

$$\sum_{j \in W, j \neq i} x_{i,j} = 1$$

Constraint kedua adalah satu warna hanya bisa berasal dari satu warna yang lain. Maksudnya dari warna i, hanya bisa berasal *unique* dari warna ke j. Saya tuliskan ekspresinya menjadi:

$$\sum_{i \in W, i \neq j} x_{i,j} = 1$$

Kedua *constraints* yang di atas ternyata belum cukup untuk menjelaskan kondisi *real*-nya. Kenapa? Kita harus pastikan:

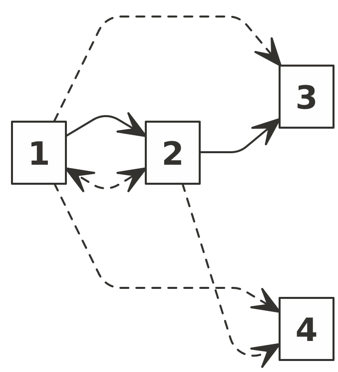
> Semua $i \in W$ terlewati. Solusi yang ada **harus melibatkan semua warna**.

Lantas bagaimana caranya?

Saya akan gunakan induksi sebagai berikut:

Misalkan:

Saya mulai produksi dari titik 1, maka ada kemungkinan saya akan berakhir ke titik $\{2, 3, 4\}$. Jika saya memilih untuk masuk ke titik 2, maka ada kemungkinan saya akan berakhir ke titik $\{3, 4\}$. Seandainya itu adalah langkah yang saya lakukan, maka saya tidak boleh melakukan langkah kembali dari 2 ke 1.



Maka saya tuliskan:

$$x_{1,2} + x_{2,1} \leq x_{1,3} + x_{1,4} + x_{2,3} + x_{2,4}$$

Proses ini akan saya ulangi untuk semua alternatif lainnya:

- Dari 1 ke 3 (termasuk kebalikannya),
- Dari 1 ke 4 (termasuk kebalikannya),
- Dari 2 ke 3 (termasuk kebalikannya),
- Dari 2 ke 4 (termasuk kebalikannya),
- Dari 3 ke 4 (termasuk kebalikannya).

Sehingga *constraint* terakhirnya:

$$x_{1,2} + x_{2,1} \leq x_{1,3} + x_{1,4} + x_{2,3} + x_{2,4}$$

$$x_{1,3} + x_{3,1} \leq x_{1,2} + x_{1,4} + x_{3,2} + x_{3,4}$$

$$x_{1,4} + x_{4,1} \leq x_{1,2} + x_{1,3} + x_{4,2} + x_{4,3}$$

$$x_{2,3} + x_{3,2} \leq x_{2,1} + x_{2,4} + x_{3,1} + x_{3,4}$$

$$x_{3,4} + x_{4,3} \leq x_{3,1} + x_{3,2} + x_{4,1} + x_{4,2}$$

6.3.1.4 *Objective Function* Permasalahan ini adalah meminimalisir waktu *cleaning* dari urutan yang ada, yakni:

$$min(\sum_{i \in W} \sum_{j \in W, j \neq i} x_{i,j} * c_{i,j})$$

6.3.2 *Solver* R

Cara mengerjakan dengan *solver* **R** diberikan kepada pembaca sebagai bahan latihan.

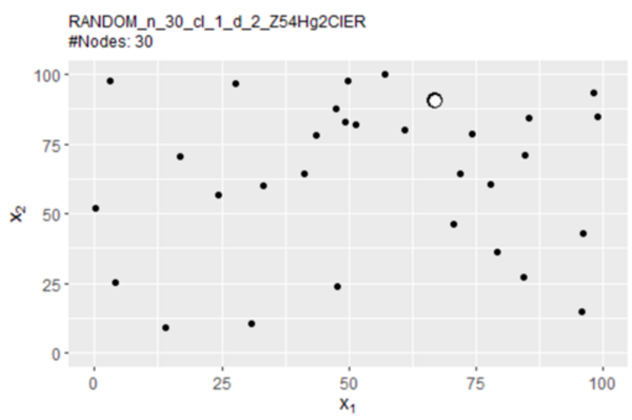
7 *Travelling Salesperson Problem*

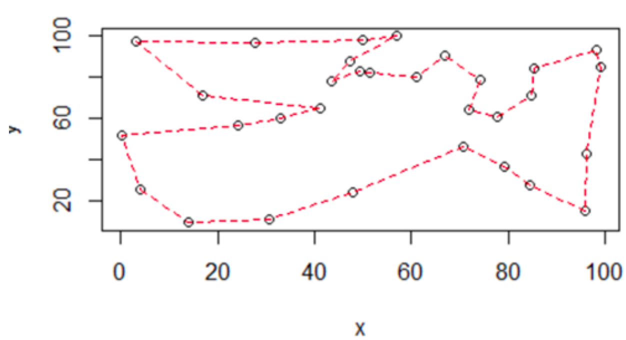
Pada bagian sebelumnya, saya telah menyebutkan bagaimana metode heuristik **TSP** bisa menyelesaikan masalah *schedulling*. Sekarang saya akan menunjukkan bagaimana cara membuat model eksak dari masalah **TSP**.

7.1 Masalah

Secara ringkas, masalah **TSP** dapat dituliskan sebagai berikut:

> ***Diberikan koordinat sejumlah titik, kita ingin menentukan tur dengan jarak terpendek yang mengunjungi semua titik tepat satu kali.***





7.2 Model Matematika dari TSP

7.2.1 Parameter

Diberikan n titik yang diberikan label $0, 1, 2, .., n-1$ dengan label 0 sebagai titik awal. Didefinisikan c_{ij} sebagai jarak antara titik i dan j.

7.2.2 *Decision Variable*

Misalkan $T = (0, 1, .., n-1)$, saya definisikan untuk $i, j \in T$ dengan $i \neq j$

$$y_{ij} = \begin{cases} 1, & \text{jika kita menuju titik } j \text{ setelah mengunjungi titik } i \\ 0, & \text{lainnya} \end{cases}.$$

7.2.3 *Constraints*

7.2.3.1 Kendala I Dari suatu titik, kita menuju **tepat** satu titik lainnya.

$$\forall i \in T, \sum_{j \neq i} y_{ij} = 1$$

7.2.3.2 Kendala II Dari suatu titik, kita menuju **tepat** satu titik lainnya.

$$\forall j \in T, \sum_{i \neq j} y_{ij} = 1$$

7.2.3.3 Kendala III Kendala ketiga dibuat untuk mencegah terjadinya *sub tour* yang melibatkan 2 atau 3 sampai $n-1$ titik.

Untuk setiap $S \subset T$ dengan $2 \leq |S| \leq n-2$:

$$\sum_{i} \sum_{j} y_{ij} \leq |S| - 1$$

7.2.4 *Objective Function*

Masalah dari **TSP** ini adalah:

$$\sum_{i} \sum_{j} c_{ij} y_{ij}$$

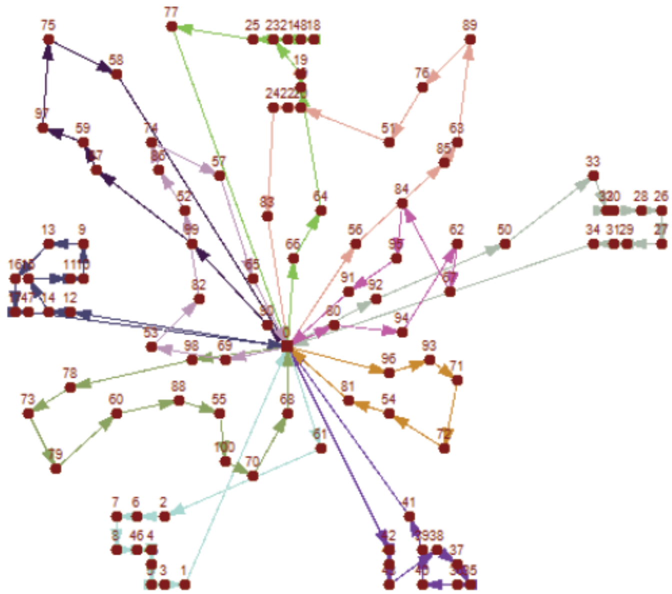
8 *Vehicle Routing Problem*

Di bagian ini kita akan membahas masalah yang dapat dianggap sebagai perluasan dari **TSP**, yang dikenal dengan nama ***Vehicle Routing Problem*** (**VRP**).

8.1 Masalah VRP Klasik

setiap rute bermula dan berakhir di titik tertentu, yang kita namakan titik *origin*, setiap titik dikunjungi tepat satu kali oleh tepat satu *vehicle*. Pengertian biaya optimal di sini dapat bervariasi, bisa berupa rute-rute dengan total jarak terkecil, atau rute-rute yang membutuhkan *vehicle* sesedikit mungkin.

Berikut ini adalah ilustrasi dari penyelesaian **VRP**.



VRP sering sekali kita temui dalam masalah transportasi barang dari satu titik (*plant* atau gudang) ke sejumlah titik distribusi atau titik *demand*. Pada era ekonomi global sekarang ini, kebutuhan transportasi barang ini berkembang sangat pesat membuat kompleksitas **VRP** yang dipunyai makin bertambah. Karena itu, banyak penelitian dilakukan untuk memodelkan variasi dari **VRP** dan teknik-teknik penyelesaian variasi dari **VRP** agar kegiatan transportasi barang makin efisien.

Di kegiatan industri sehari-hari, seringkali masalah yang dihadapi lebih rumit dibandingkan masalah **VRP klasik** yang dijelaskan di atas.

1. Contoh yang pertama adalah masalah **VRP** di mana tujuan kunjungan yang harus dilakukan adalah untuk mengirim sejumlah barang ke titik-titik *demand* dengan *volume demand* yang kadang kala cukup besar, melebihi kapasitas angkut dari *vehicle* yang ada.
2. Contoh yang kedua adalah masalah **VRP** yang mewakili kebutuhan kunjungan oleh teknisi atau *sales*, di mana tidak ada kendala kapasitas *vehicle* tetapi ada kendala total waktu tur yang dibatasi oleh jam kerja teknisi atau *sales*.

Beragamnya kendala yang ada mengakibatkan banyaknya variasi dari **VRP**, yang sebagiannya akan dibahas setelah ini.

8.2 Variasi VRP

- Jika terdapat kendala di mana *vehicle* yang ada mempunyai kapasitas tertentu: ***Capacitated Vehicle Routing Problem*** (**CPRV**).
- Jika terdapat kendala di mana tiap titik harus dikunjungi pada *interval* waktu tertentu: ***Vehicle Routing Problem with Time Windows*** (**PRVTW**).
- Jika terdapat kondisi di mana suatu titik dapat dikunjungi atau dilayani oleh lebih dari satu *vehicle*: ***Vehicle Routing Problem with Split Deliveries*** (**PRVSD**).
- Jika terdapat kondisi di mana sutu titik dapat dikunjungi atau dilayani oleh lebih dari satu *vehicle* dan tiap titik harus dikunjungi pada *interval* waktu tertentu: ***Vehicle Routing Problem with Time Windows and Split Deliveries*** (**PRVTWSD**).
- Di dunia nyata, kadang kita menemukan **VRP** di mana tujuannya bukan menentukan rute-rute dengan total jarak dari seluruh rute terkecil tetapi menentukan rute-rute dengan rute terpanjangnya mempunyai total jarak terkecil: ***Min Max Vehicle Routing Problem*** (**MMVRP**).

Salah satu alasan munculnya **MMVRP** ini adalah adanya kebutuhan untuk membuat rute-rute yang cenderung seragam dalam hal total jarak, atau karena adanya batasan untuk total waktu tempuh dari rute-rute yang akan didapatkan dari penyelesaian **VRP** ini.

8.3 Model Matematika VRP Klasik

Sekarang kita akan membuat model matematika dari permasalahan **VRP** klasik.

8.3.1 Parameter

Misalkan:

- $T = \{1, 2, .., n\}$ adalah himpunan titik *demand*.
- Titik 0 adalah titik depot atau gudang.
- d_{ij} adalah jarak dari i ke j.
- q_i adalah *demand* barang di titik i.
- $V = \{1, 2, .., m\}$ adalah himpunan *vehicles*.

8.3.2 *Decision Variables*

Misal saya definisikan:

$$x_{ijk} = \begin{cases} 1, & \text{jika titik } j \text{ dikunjungi setelah kunjungan ke titik } i \text{ oleh vehicle ke } k \\ 0, & \text{lainnya.} \end{cases}$$

dan

$$y_{ik} = \begin{cases} 1, & \text{jika titik } i \text{ dikunjungi oleh vehicle ke } k, \\ 0, & \text{lainnya.} \end{cases}$$

8.3.3 *Constraints*

Berikut adalah kendala-kendala yang ada:

$$\forall i \in T \bigcup \{0\},\ \forall k \in V, \sum_{j \neq i} x_{ijk} = y_{ik}$$

dan

$$\forall j \in T \bigcup \{0\},\ \forall k \in V, \sum_{i \neq j} x_{ijk} = y_{ik}$$

Kendala ini menjamin di setiap titik *demand* dikunjungi oleh suatu *vehicle* dan *vehicle* tersebut pasti akan meninggalkan titik tersebut.

Sedangkan:

$$\forall i \in T,\ \sum_{k \in V} y_{ik} = 1$$

Kendala ketiga menjamin setiap titik dikunjungi tepat satu *vehicle*.

Lalu:

$\forall S \subset T,$ dengan $2 \leq |S| \leq n-2,$ dan $\forall k \in V$:

$$\sum_{i} \sum_{j} x_{ijk} \leq |S| - 1$$

8.3.4 *Objective Function*

Tujuan utama masalah ini adalah meminimumkan jarak yang ditempuh.

$$\sum_{i \in T} \sum_{j \neq j} \sum_{k \in V} d_{ij} x_{ijk}$$

9 *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.

9.1 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?***

9.2 Model Matematika

9.2.1 Membangun Model Matematika

9.2.1.1 *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.

9.2.1.2 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.***

9.2.2 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*.

9.2.3 Variabel Penentuan

Saya definisikan:

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

9.2.4 *Constraints*

Sekarang kita akan menuliskan *constraints* dalam bahasa matematika.

9.2.4.1 *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.

9.2.4.2 ***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.

9.2.4.3 ***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.

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

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

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

9.2.4.4 ***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:

9.2.4.4.1 **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:

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

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

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

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

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

9.2.5 *Objective Function*

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

9.3 *Solver R*

9.3.1 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
```

9.3.2 *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> ----
```

9.4 Solusi Optimal

9.4.1 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

9.4.2 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

id_perawat	Pagi	Siang	Malam
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?***

9.5 Masalah Baru

Sekarang kita akan paksakan setiap *nurse* mendapatkan *workload* yang sama.

Pertama-tama, mari kita hitung. Berapa banyak shift yang ideal per perawat.

$$\text{banyak shift ideal} = \frac{hari * shift * min(\text{perawat per shift})}{\text{total perawat}}$$

Yakni:

$$\text{banyak shift ideal} = \frac{14 * 3 * 4}{24} = \frac{168}{7} = 7$$

Jadi diharapkan setiap perawat mendapatkan banyak *shift* yang sama, yakni sebanyak 7 *shifts*.

Maka model matematika dari *constraint* ini adalah:

$$\sum_{h \in H} \sum_{s \in S} x_{h,n,s} = 7, \text{untuk } n \in N$$

9.6 *Solver R*

9.6.1 Penulisan Model Matematika di R

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

```
# membuat model
model =
  model %>%
  # contraint baru
  add_constraint(sum_expr(x[n,h,s],
                          h = 1:14,
                          s = 1:3) == 7,
                 n = 1:24)
model
```

```
## Mixed integer linear optimization problem
## Variables:
##   Continuous: 0
##   Integer: 0
##   Binary: 1008
## Model sense: minimize
## Constraints: 1026
```

9.6.2 *Solving*

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

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

```
## <SOLVER MSG>  ----
## GLPK Simplex Optimizer, v4.65
## 1026 rows, 1008 columns, 7248 non-zeros
##       0: obj =   0.000000000e+00 inf =   3.360e+02 (66)
##     372: obj =   1.680000000e+02 inf =   0.000e+00 (0) 1
## OPTIMAL LP SOLUTION FOUND
## GLPK Integer Optimizer, v4.65
## 1026 rows, 1008 columns, 7248 non-zeros
## 1008 integer variables, all of which are binary
## Integer optimization begins...
## Long-step dual simplex will be used
## +   372: mip =     not found yet >=              -inf        (1; 0)
## +  1517: >>>>>   1.680000000e+02 >=   1.680000000e+02   0.0% (64; 1)
## +  1517: mip =   1.680000000e+02 >=     tree is empty   0.0% (0; 129)
## INTEGER OPTIMAL SOLUTION FOUND
## <!SOLVER MSG> ----
```

9.6.3 Solusi Optimal

9.6.3.1 Jadwal Optimal Berikut adalah jadwal optimal dalam kasus ini:

Table 5: Jadwal Perawat: Angka Pada Tanggal Menunjukkan id Perawat

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

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

9.6.3.2 Rekap Jadwal Optimal

Berikut adalah rekapan jadwal optimal per perawat:

Table 6: Rekapan Berapa Kali Perawat Bertugas

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

10 Masalah Inventori Gudang

Kali ini kita akan melihat contoh pemodelan matematika di masalah pengelolaan inventori (*inventory control*) untuk memperluas wawasan jenis-jenis model optimisasi.

10.1 Masalah

Suatu toko *online* rumahan biasa menjual *frozen food* berupa somay dan dimsum. Karena masih beroperasi skala kecil, toko tersebut hanya memiliki 2 buah *freezer* untuk dijadikan tempat penyimpanan stok barang yang hendak dijual. Toko tersebut selalu melakukan restok barang di hari Minggu setiap pekan.

> Setiap pekan barang yang terjual di toko tersebut tidak selalu sama tapi kita bisa hitung rata-rata produk terjual setiap pekannya.

Di masa pandemi ini, usahanya sudah berkembang pesat.

- Kadangkala stok barang sudah habis ketika ada konsumen yang hendak membeli *frozen food*.
- Kadang pula setelah satu pekan masih tersedia stok barang yang belum terjual dan ini membuat biaya penyimpanan bertambah.

Oleh karena itu, toko tersebut perlu mengetahui berapa stok barang yang harus dipesan ke *supplier* setiap kali pemesanan. Lalu apakah toko tersebut harus memesan lebih sering atau lebih jarang.

Berapa banyak barang harus dipesan dan berapa sering barang harus dipesan dikenal dengan nama ***Economic Order Quantity*** (**EOQ**), suatu besaran yang dihitung dalam rangka meminimumkan rata-rata biaya inventori, yaitu biaya-biaya untuk pemesanan dan penyimpanan.

10.2 Parameter

Parameter di masalah kita adalah beberapa besaran yang diketahui, seperti:

1. C_O adalah biaya pemesanan dalam satu kali pesan. Pada kasus ini biaya tidak bergantung pada banyaknya barang yang dipesan ke *supplier*.
2. D adalah rata-rata banyaknya barang yang terjual setiap pekan. Satuan dari D adalah $\frac{pcs}{7hari}$.
3. S adalah biaya penyimpanan setiap barang per hari.

Kita asumsikan semua nilainya tetap (tidak ada perubahan apapun).

10.3 Model Matematika

10.3.1 *Decision Variable*

Kita definisikan *decision variable* sebagai berikut:

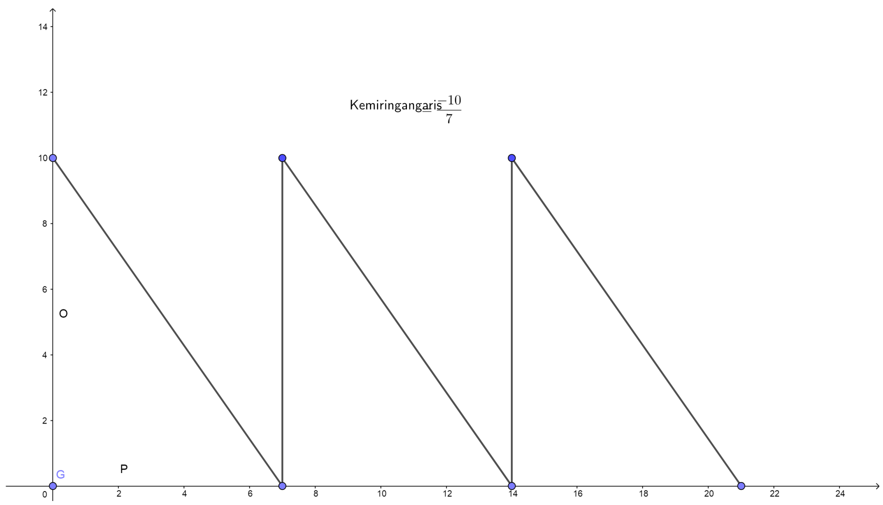
- O adalah banyaknya barang yang dipesan dalam satu kali pesan.
- P adalah panjangnya siklus antara **dua** kali pemesanan (dalam hari).

10.3.2 Masalah Optimisasi

Kita dapat memodelkan masalah optimisasinya dengan terlebih dahulu membuat simplifikasi dinamika inventori barang sebagai berikut:

- Misalkan barang dipesan sebanyak 10 pcs tiap pekan.
- Rata-rata banyaknya barang yang terjual ke konsumen adalah $\frac{10}{7}$ pcs per pekan.
- Asumsikan *lead time*, yaitu periode waktu yang dibutuhkan mulai dari memesan sampai barang datang adalah `0`. Maksudnya adalah saat toko memesan barang ke *supplier*, tingkat inventori adalah 0 dan setelah dipesan tingkat inventori langsung berubah menjadi 10.

Perhatikan grafik berikut:



Tujuan kita adalah menentukan berapa nilai O dan berapa nilai P agar rata-rata biaya inventori minimum.

Agar lebih mudah dalam penurunan fungsi objektif, kita akan gunakan rata-rata biaya inventori per hari. Kita sebenarnya dapat juga menurunkan fungsi objektif berupa rata-rata biaya inventori per pekan atau per bulan karena pada masalah ini biaya pemesanan adalah tetap (tidak tergantung pada berapa banyak barang yang kita pesan).

Jika toko memesan barang tiap P hari, maka:

- Rata-rata biaya pesan per hari adalah $\frac{C_O}{P}$,
- Rata-rata biaya penyimpanan per hari adalah $\frac{OP}{2}$.

Sehingga rata-rata total biaya iventori perhari adalah:

$$\frac{C_O}{P} + \frac{OP}{2}$$

Fungsi di atas adalah fungsi **dua peubah**. Kita bisa mengubahnya menjadi fungsi satu peubah dengan mensubtitusi variabel P dalam variabel O jika kita gunakan asumsi tambahan bahwa kita memesan di saat tingkat inventori sama dengan nol.

Oleh karena itu jika kita gunakan ilustrasi grafik di atas, maka pada **kondisi umum** kemiringan grafiknya adalah $-\frac{D}{7}$. Sehingga didapatkan:

$$P = \frac{7O}{D}$$

Jika saya substitusikan kembali ke persamaan biaya, maka:

$$\frac{C_O}{P} + \frac{OP}{2} = \frac{C_O D}{7O} + \frac{7O^2}{2D}$$

Nilai O yang optimal dapat diperoleh dengan menyamakan turunan pertama fungsi dia atas sama dengan nol (syarat perlu untuk keoptimalan), dan kita dapatkan:

$$EOQ = \sqrt{\frac{2C_O D^2}{7}}$$

dan nilai P yang optimal dapat diperoleh dengan mensubstitusikan nilai EOQ di atas ke persamaan yang mengalikan P dan O, yaitu $P = \frac{7O}{D}$.

11 *Non Linear Modelling*

11.1 Masalah

Ada kalanya kita bertemu dengan masalah *optimization* yang tidak linear. Justru biasanya masalah *real world* tidak berbentuk linear. Sayangnya *solver* di **R** yang ada sekarang masih terbatas di *linear problem* saja.

> ***Bagaimana menyelesaikannya?***

Salah satu solusinya adalah dengan mengkonversi masalah **tidak linear** menjadi **linear**.

11.2 Contoh Masalah *Optimization*

Minimumkan:

$$x_1^2 + x_1x_2 + x_2^2 + x_1 + x_2$$

Terhadap:

$$4x_1 + 6x_2 \geq 10, \text{ dimana: } x_1, x_2 \in \{0, 1\}$$

11.2.1 *Solver* di R

Untuk mengecek apakah `ompr` bisa menyelesaikan masalah *non linear* di atas, maka saya akan tuliskan model matematikanya dalam **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_variable(x[i],
#                i = 1:2,
#                type = "binary",
#                lb = 0) %>%
#   set_objective(x[1]^2 + x[1] * x[2] + x[2]^2 + x[1] + x[2],
#                  "min") %>%
#   add_constraint(4*x[1] + 6*x[2] >= 10)

# model
```

`ompr` tidak mampu menyelesaikan masalah *non linear* seperti di atas.

11.2.2 Konversi ke Masalah Linear

11.2.2.1 Definisi Variabel y Untuk mengubahnya ke dalam masalah linear, saya akan membuat pemisalan sebagai berikut:

$$y = x_1 x_2$$

Karena x_1 dan x_2 adalah *binary*, maka:

$$x_1^2 = x_1$$

$$x_2^2 = x_2$$

$$y \leq x_1$$

$$y \leq x_2$$

$$y \geq x_1 + x_2 - 1$$

11.2.2.2 Perubahan Model *Optimization* Dari persamaan-persamaan di atas, kita telah mendapatkan perubahan linear dari masalah awalnya.

Yakni:

Minimumkan:

$$y + 2x_1 + 2x_2$$

Terhadap:

$$4x_1 + 6x_2 \geq 10$$

$$-x_1 + y \leq 0$$

$$-x_2 + y \leq 0$$

$$x_1 + x_2 - y \leq 1$$

$$x_1, x_2, y \in \{0, 1\}$$

11.2.2.3 Penulisan Model di R Sekarang kita akan menuliskan model di atas ke dalam **R** sebagai berikut:

```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_variable(x[i],
               i = 1:2,
               type = "binary",
               lb = 0) %>%
  add_variable(y,
               type = "binary",
               lb = 0) %>%
  set_objective(y + 2*x[1] + 2*x[2],
                "min") %>%
  add_constraint(4*x[1] + 6*x[2] >= 10) %>%
  add_constraint(-x[1] + y <= 0) %>%
  add_constraint(-x[2] + y <= 0) %>%
  add_constraint(x[1] + x[1] - y <= 1)

model
```

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

11.2.2.4 ***Solving*** Kemudian saya *solve* dengan **R**:

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

```
## <SOLVER MSG>  ----
## GLPK Simplex Optimizer, v4.65
## 4 rows, 3 columns, 8 non-zeros
##       0: obj =   0.000000000e+00 inf =   1.000e+01 (1)
##       3: obj =   5.000000000e+00 inf =   0.000e+00 (0)
## *     5: obj =   5.000000000e+00 inf =   0.000e+00 (0)
## OPTIMAL LP SOLUTION FOUND
## GLPK Integer Optimizer, v4.65
## 4 rows, 3 columns, 8 non-zeros
## 3 integer variables, all of which are binary
## Integer optimization begins...
## Long-step dual simplex will be used
```

```
## +     5: mip =     not found yet >=              -inf        (1; 0)
## +     5: >>>>>   5.000000000e+00 >=   5.000000000e+00   0.0% (1; 0)
## +     5: mip =   5.000000000e+00 >=     tree is empty   0.0% (0; 1)
## INTEGER OPTIMAL SOLUTION FOUND
## <!SOLVER MSG> ----
```

11.2.2.5 Solusi Optimal Berikut adalah solusi optimal yang didapatkan:

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

```
##   variable i value
## 1        x 1     1
## 2        x 2     1
```

12 Regresi Linear

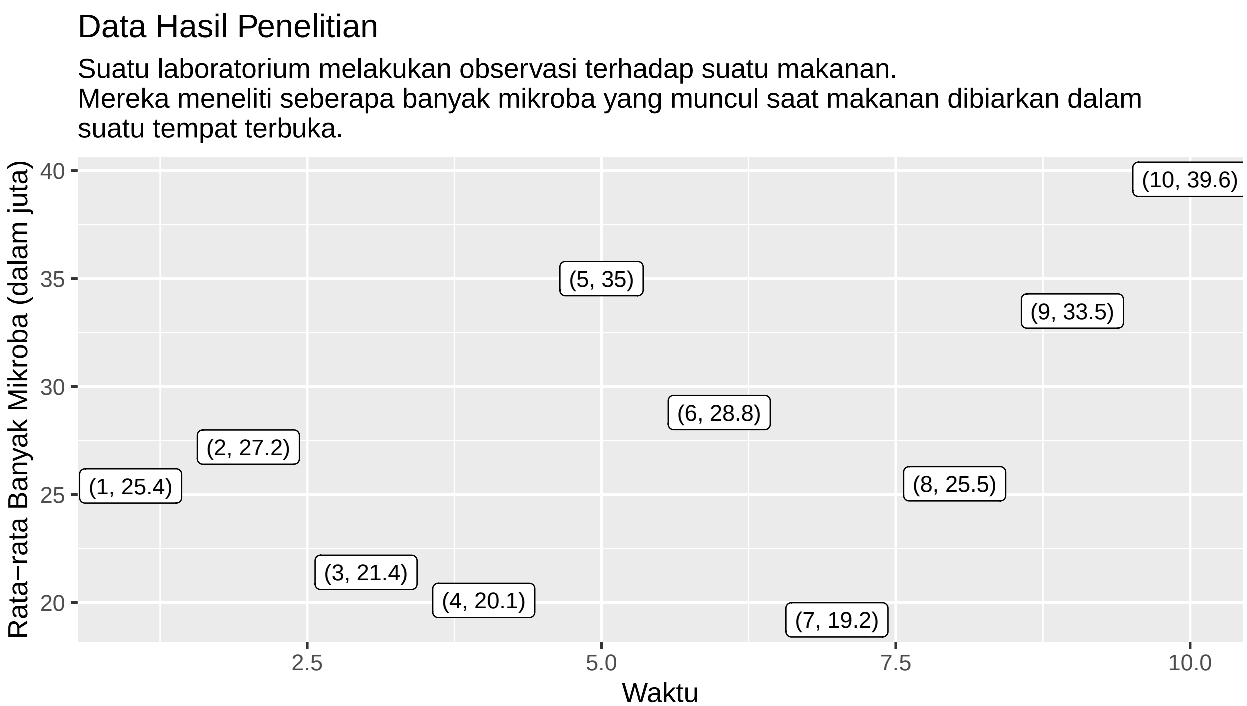
Suatu permasalahan regresi linear bisa dipandang sebagai masalah *optimization*. *Kok bisa?*

Mari kita lihat contoh kasus berikut ini:

12.1 Masalah

Suatu laboratorium hendak melakukan penelitian tentang mikroba dalam beberapa makanan basi. Mereka mengambil sampel mikroba dalam selang waktu tertentu dan mencatat berapa banyak mikroba yang ada di beberapa makanan tersebut.

Berikut adalah datanya:



Bisakah kita membuat model prediksi berapa banyak mikroba dari waktu tertentu?

12.2 Permodelan Matematika

Misalkan persamaan regresi linear yang akan saya cari adalah sebagai berikut:

$$y = at + b$$

Dengan y adalah banyaknya mikroba dan t adalah waktu.

Jika kita perhatikan kembali, suatu persamaan regresi linear disebut **terbaik** saat *error* yang dihasilkan sangat kecil. Misal saya tulis $\hat{y_i}$ sebagai hasil prediksi model regresi pada waktu i.

Maka *error* bisa saya tuliskan sebagai $e_i = \hat{y_i} - y_i$.

Kalau kita ingat, ada satu parameter *goodness of fit* bernama **RMSE** (*Root Mean Square Error*). Kelak **RMSE** ini akan saya jadikan *objective function* dari masalah *optimization*.

12.2.1 *Objective Function*

$$\min(e_1^2) = \min(\hat{y}_i - y_i)^2 = \min(at_i + b - y_i)^2$$

12.2.2 *Decision Variables*

Variabel keputusan yang dicari adalah:

$$a, b \in \mathrm{R}$$

12.2.3 *Constraint*

Constraints pada masalah ini hanyalah pada *boundaries* t dan y yang ada pada data berikut ini:

Table 7: Data

t	y
1	25.4
2	27.2
3	21.4
4	20.1
5	35.0
6	28.8
7	19.2
8	25.5
9	33.5
10	39.6

12.3 *Solver* R

Cara mengerjakan dengan *solver* **R** diberikan kepada pembaca sebagai bahan latihan.

12.4 Metode Lain: Menggunakan *Linear Modelling*

Untuk menyelesaikan masalah regresi linear di **R**, kita bisa menggunakan *function* `lm()`. Salah satu tutorial lengkap terkait regresi linear di **R** bisa dilihat di sini.

12.4.1 Menyelesaikan `lm()` di R

```
regresi_model = lm(y ~ t, data)
summary(regresi_model)
```

```
##
## Call:
## lm(formula = y ~ t, data = data)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -10.038  -4.485   1.356   3.350   7.986
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  21.4533     4.2361   5.064 0.000972 ***
## t             1.1121     0.6827   1.629 0.141967
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.201 on 8 degrees of freedom
## Multiple R-squared:  0.2491, Adjusted R-squared:  0.1552
## F-statistic: 2.654 on 1 and 8 DF,  p-value: 0.142
```

Dari hasil di atas, kita dapatkan:

$y = 21.4533333 + 1.1121212\ t$

Jika kita hitung **RMSE**, didapatkan nilai:

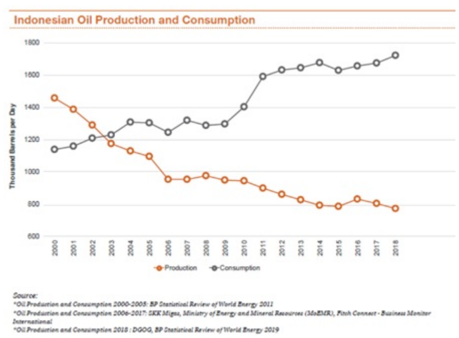
```
caret::RMSE(regresi_model$fitted.values,data$y)
```

```
## [1] 5.546385
```

13 Penyediaan BBM

Di bagian ini kita akan mempelajari simplifikasi dari masalah penyediaan bahan bakar minyak (BBM) di Indonesia. Kebutuhan BBM di Indonesia sangat tinggi, mengingat banyaknya penduduk di Indonesia yang memerlukanbahan bakar gas. Walaupun Indonesia adalah negara penghasil minyak mentah, yang merupakan bahan pembuat BBM, tetapi konsumsi BBM jauh lebih tinggi dibandingkan total produksi BBM yang dapat dihasilkan dari seluruh minyak mentah yang dikilang di kilang dalam negeri. Gambar berikut memperlihatkan kondisi kekurangan tersebut, sehingga sejak tahun 2004 Indonesia adalah negara pengimpor minyak mentah dan bahan bakar minyak.

```
## [1] "Grafik produksi dan konsumsi BBM Indonesia. Sumber: PWC Indonesia."
```



Mengapa disebut negara pengimpor minyak mentah dan juga bahan bakar minyak?

13.1 Masalah

Dalam setahun, Indonesia:

- Membutuhkan BBM sejumlah tertentu. Di grafik di atas, pada tahun 2018 kebutuhan BBM Indonesia adalah sekitar 1.750 juta kilo liter per hari.
- Dari sejumlah kilang yang ada, Indonesia hanya mampu memproses minyak mentah menjadi BBM sejumlah tertentu. Di tahun 2018, kemampuan produksinya hanyalah sejumlah 638 ribu kilo liter.
- Dari sejumlah blok sumur minyak yang ada, Indonesia dapat menghasilkan sejumlah minyak mentah jenis tertentu. Di grafik di atas, pada tahun 2018 volume produksi minyak mentah Indonesia adalah sekitar 800 ribu kilo liter. Tetapi:
  - (a) yang mutlak dimiliki pemerintah Indonesia hanya sekitar 10% nya saja, sisanya dimiliki oleh pihak KKKS yang bebas menjualnya ke dalam negeri atau untuk eksport. Selain itu,
  - (b) tidak semua minyak yang dihasilkan optimal dikilang di kilang-kilang yang ada di Indonesia.

Karena alasan (a) dan (b) di atas, sebagian dari minyak mentah yang dihasilkan diekspor ke manca negara, dan pengilangan minyak mentah di kilang-kilang dalam negeri bahan mentahnya adalah minyak mentah hasil *lifting* dalam negeri dan minyak mentah impor dari luar negeri (contohnya dari Arab Saudi, Nigeria, Australia dan Aljazair).

13.2 Asumsi

Di model optimisasi kita melihat simplifikasi masalah dengan mengasumsikan:

1. Hanya ada dua jenis BBM hasil pengilangan, yaitu BBM 1 dan BBM 2.
2. Ada empat negara dari mana kita mengimport minyak mentah, yaitu negara Arab Saudi, Nigeria, Australia dan Aljazair,
3. *Yield factor* dari minyak mentah yang diimpor dari tiap negara berbeda.
   - *Yield factor* adalah proporsi untuk tiap jenis BBM yang dihasilkan dari pengilangan minyak mentah.
   - Sebagai contoh, jika *yield factor* dari suatu minyak mentah dalam menghasilkan BBM 1 adalah 0.7 maka dari satu juta kilo liter minyak mentah tersebut yang dikilang dihasilkan 700 ribu kilo liter BBM 1.
4. Ada satu negara dari mana kita mengimpor BBM 1 dan BBM 2.
5. Biaya pengilangan minyak mentah di semua kilang sama, sehingga di model ini kita boleh menganggap hanya ada satu kilang saja.

Berdasakan deskripsi di atas, kita dapat menyimpulkan masalah optimisasi dalam penyediaan BBM di Indonesia adalah berapa banyak ekspor minyak mentah, berapa banyak impor minyak mentah (dan dari negara mana saja), berapa banyak kita mengimpor tiap jenis BBM, agar kebutuhan BBM terpenuhi dengan total biaya penyediaan BBM minimum.

13.3 Parameter

Berikut ini kita akan melihat parameter-parameter yang terlibat dalam pemodelan masalah penyediaan BBM ini, setelah ada beberapa proses simplifikasi dari masalah nyatanya. Demikian pula dengan nilai-nilai paramter yang ada dibuat dummy, mengingat beberapa nilai parameter nilainya berubah seiring perubahan harga minyak dunia dan kondisi infrastruktur pengilangan minyak mentah di Indonesia.

Untuk memudahkan pemodelan, kita mendefinisikan himpunan indeks terlebih dahulu. Memodelkan masalah penyediaan BBM ini setingkat lebih rumit dibandingkan memodelkan perencaan produksi di koperasi susu sebelumnya karena bahan baku produksi BBM berasal dari lebih dari satu pemasok (supplier), di mana keputusan menggunakan bahan baku dari pemasok mana saja adalah hal yang akan menjadi bagian dari solusi masalah. Sementara di masalah koperasi susu, susu sudah tersedia dan tidak ada kebutuhan mengetahui susu dari mana saja itu berasal. Variabel keputusan dapat didefinisikan dengan mudah setelah kita mempunyai himpunan indeks ini secara lengkap, demikian pula ketika kita membuat kendala-kendala di masalah optimisasi himpunan indeks ini akan sangat membantu kita membuat ekspresi matematikanya.

Misalkan:

- $I = \{1, 2\}$ adalah himpunan jenis BBM.
- $J = \{1, 2, .., 5\}$ adalah himpunan pemasok BBM. Sebagai catatan $J = 5$ berarti pemasok dalam negeri.
- D_i adalah *demand* BBM jenis i dalam satuan kilo liter.
- CO_i adalah harga impor BBM jenis i dalam satuan USD per satu juta kilo liter.
- S_i adalah harga jual BBM jenis i dalam satuan USD per satu juta kilo liter.
- CR_j adalah harga impor minyak mentah dari negara j dalam satuan USD per satu juta kilo liter.
- $Y_j i$ adalah *yield factor* dari minyak mentah j dalam menghasilkan BBM jenis i.
- OP adalah biaya kilang BBM dalam satuan USD per satu juta kilo liter.
- P adalah kapasitas produksi kilang minyak dalam negeri dengan satuan USD per satu juta kilo liter.

13.4 *Decision Variable*

- Misalkan x_j adalah banyaknya minyak mentah yang dibeli dari negara j. Ingat kembali untuk $j = 5$ variabel ini terkait dengan penggunaan BBM dari dalam negeri.
- Misalkan z_i adalah banyaknya impor BBM jenis i.

13.5 *Constraints*

Berikut adalah kendala-kendala yang ada:

- *Demand* dari tiap jenis BBM harus terpenuhi, baik dari hasil pengilangan minyak mentah ataupun dari impor BBM.

$$\sum_{j \in J} Y_{ji} x_j + z_i \geq D_i$$

- Kapasitas produksi kilang di dalam negeri terbatas, yaitu paling banyak sebanyak P.

$$\sum_{j \in J} x_j \leq P$$

- Banyaknya minyak mentah hasil lifting minyak mentah dalam negeri yang diproses di kilang adalah terbatas, yaitu paling banyak L.

$$x_5 \leq L$$

13.6 *Objective Function*

Fungsi objektif dari masalah ini adalah biaya penyediaan BBM. Biaya terdiri dari:

1. Biaya untuk mengimpor minyak mentah,
2. Biaya untuk mengimpor BBM,
3. Biaya untuk mengilang minyak mentah.

Semua biaya tersebut kita keluarkan, dan di masalah ini masih ada peluang kita mempunyai pendapatan dari mengekspor minyak mentah dan penjualan BBM. Karena itu, lebih baik kita menggunakan besaran keuntungan, yaitu pendapatan dikurangi biaya, sebagai fungsi objektif. Sekarang kita rinci berapa biaya yang kita keluarkan dan berapa pendapatan yang mungkin.

- Biaya impor/pembelian minyak mentah sebanyak $\sum_{j=1}^{5} C_j x_j$.
- Biaya impor BBM sebanyak $\sum_{i=1}^{2} CO_i z_i$.
- Biaya pengilangan minyak mentah sebanyak $OP \sum_{j=1}^{5} x_j$.
- Pendapatan dari penjualan BBM sebanyak $\sum_{i=1}^{2} S_i (\sum_{j=1}^{5} Y_{ji} x_j)$,
- Pendapatan dari eksport minyak mentah sebesar $C_5(L - x_5)$.

Sehingga kita mempunyai fungsi objektif berupa fungsi keuntungan dengan ekspresi.

$$\text{Obj Function} = \sum_{i=1}^{2} S_i(\sum_{j=1}^{5} Y_{ji}x_j) + C_5(L - x_5) - \sum_{j=1}^{5} C_j x_j - \sum_{i=1}^{2} CO_i z_i - OP\sum_{j=1}^{5} x_j$$

Maka tujuan kita adalah max(Obj Function).

14 Penentuan *Supplier*

Kali ini, kita akan membahas terkait masalah *real* yang dihadapi Nutrifood di *plant*.

14.1 Latar Belakang

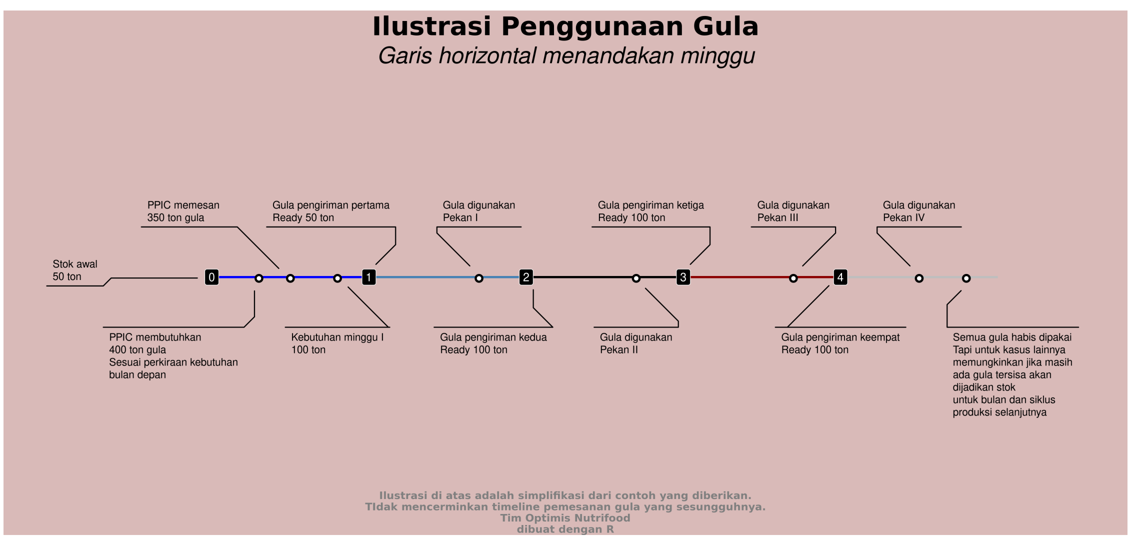
Dalam rangkaian produksi, **Nutrifood** menggunakan banyak sekali bahan baku. Salah satu bahan baku utama yang paling sering digunakan adalah **gula**.

Selama ini **Nutrifood** memesan gula secara **langsung tiap bulannya** dengan besarnya pemesanan disesuaikan dengan:

1. Angka *forecast* masing-masing produk yang menggunakan gula-gula tersebut.
2. *Existing stock* gula yang ada di gudang bahan baku.
3. *Minimum order* per jenis gula yang ditetapkan *vendor*.

Informasi terkait pengiriman gula:

- Pengiriman bahan baku gula oleh para *vendor* dilakukan sebanyak **4** kali dalam sebulan dengan jumlah sesuai dengan aturan berikut:
  - Banyaknya gula pada **pengiriman pertama** disesuaikan dengan **stok *existing*** dan ***demand*** produk terkait gula tersebut pada minggu I.
  - Sedangkan banyaknya gula pada **pengiriman kedua hingga keempat** dibuat proporsional.
  - Contoh pada suatu bulan tertentu:
    * Kebutuhan gula diperkirakan sebesar **400 ton**.
    * Stok *existing* gula di gudang bahan baku ada **50 ton**.
    * Maka Nutrifood perlu memesan gula sebesar **350 ton**.
    * Pengiriman dilakukan **4 kali**.
    * Pada minggu I, diperkirakan kebutuhan gula ada sebesar **100 ton**. Oleh karena itu, **pengiriman gula pertama** adalah sebesar **50 ton** saja.
    * Pada minggu II, III, dan IV pengiriman gula adalah proporsional sebesar **100 ton**.
  - Oleh karena itu, kelak pada model matematika perlu ada *constraints* terkait hal ini.



- **Waktu pengiriman dan inspeksi** selama **17** hari setelah pemesanan gula sampai akhirnya gula tersebut dapat digunakan untuk produksi.

Oleh karena itu, perencanaan pembelian gula dilakukan setidaknya sebulan sebelum gula tersebut akan digunakan.

Saat ini, ada **6** jenis gula yang bisa dipesan ke **6** *vendor* yang berbeda. Masing-masing gula digunakan untuk membuat produk tertentu.

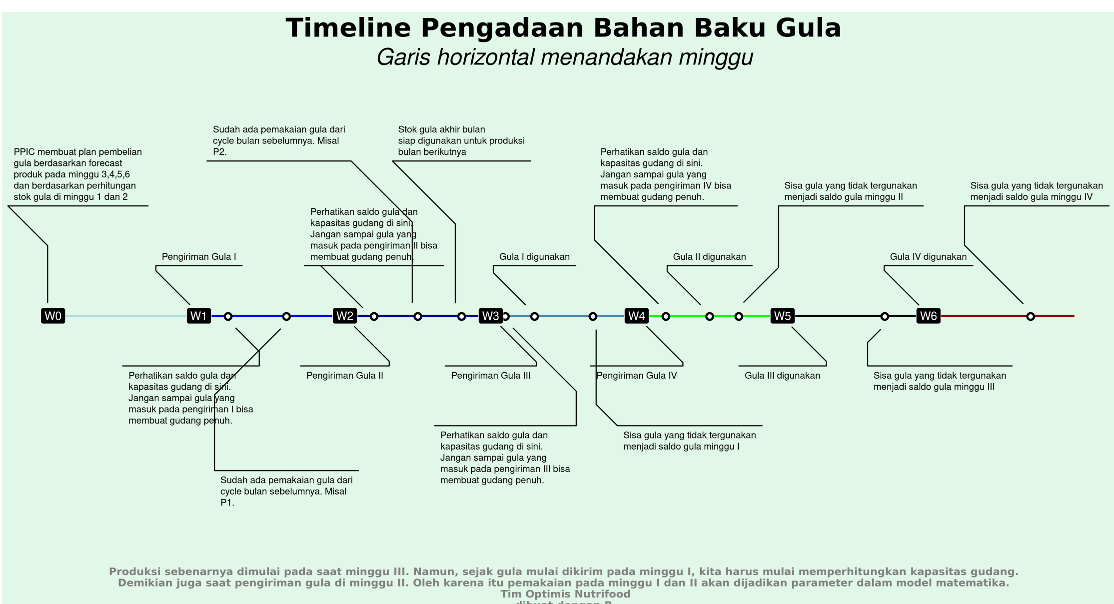
Informasi lain yang perlu diketahui adalah:

- **Tidak ada kewajiban** bagi Nutrifood untuk membeli semua jenis gula.
- Terkait penggunaan bahan baku gula:
  - Sebagian kecil dari produk hanya bisa diproduksi dengan **satu jenis gula saja**.
  - Sebagian besar lainnya memungkinkan untuk diproduksi dengan **dua atau lebih jenis gula**.
    - Unit per jenis gula yang digunakan untuk membuat produk adalah sama walau berbeda jenis gula.
- Setidaknya minimal ada **2 jenis gula** yang dibeli Nutrifood sebagai *back up* substitusi bahan baku.
- Pembelian gula harus memenuhi *minimum order* yang ditetapkan oleh *vendor* **tapi** jika pembelian di atas *minimum order* harus dilakukan **pembulatan**. Misalkan:
  - *Minimum order* gula adalah **10 ton**, maka Nutrifood:
    - Boleh membeli **11 ton**.
    - Tidak boleh membeli **10.5 ton**.
- Harga masing-masing jenis gula berbeda. Namun untuk lama pengiriman gula, pada kasus ini semua gula memiliki lama pengiriman yang sama.

Berapa banyak gula yang harus dipesan ke masing-masing vendor `dalam sebulan` agar menghasilkan cost yang terendah?

14.1.1 Alur Pengadaan Bahan Baku Gula

Berikut adalah *summary* alur pengadaan bahan baku gula yang dilakukan Nutrifood:



Model matematika yang dibuat akan berdasarkan *flow* di atas.

14.2 Contoh Ilustrasi

14.2.1 Case I: Minimal 2 jenis gula sebagai *back up*

Dalam suatu bulan tertentu, untuk memproduksi produk **A**, **B**, dan **C** dibutuhkan gula x_1, x_2, **atau** x_3.

Untuk memastikan tidak ada masalah di kemudian hari (sebagai *back up*), maka minimal harus ada **2** jenis gula yang harus dibeli. Alternatifnya:

1. Membeli x_1 dan x_2,
2. Membeli x_1 dan x_3,
3. Membeli x_2 dan x_3, atau
4. Membeli x_1, x_2, dan x_3.

14.2.2 Case II: Unit gula yang digunakan sama

Dalam suatu bulan tertentu, untuk membuat produk **A**, kita bisa menggunakan:

1. 100 unit x_1 atau,
2. 100 unit x_2.

Sedangkan untuk membuat produk **B**, kita bisa menggunakan:

1. 100 unit x_2 atau,
2. 100 unit x_3.

Dari kasus di atas, kita bisa menuliskan bahwa:

1. Kebutuhan gula 1 ada sebesar $x_1 \leq 100$.
2. Kebutuhan gula 2 ada sebesar $x_2 \leq 200$.
3. Kebutuhan gula 3 ada sebesar $x_3 \leq 100$.

Karena minimal harus ada **2 gula yang dipilih**, maka alternatif solusi yang ada adalah:

1. 100 unit x_1 dan 100 unit x_2.
2. 100 unit x_1 dan 100 unit x_3.
3. 100 unit x_2 dan 100 unit x_3.
4. 100 unit x_1, 100 unit x_2 dan 100 unit x_3.

14.3 Model Matematika

14.3.1 *Time Frame*

Time frame dari masalah ini adalah penentuan berapa banyak jenis gula yang harus dibeli setiap **bulan** dalam rentang **6 minggu**.

14.3.2 *Known Parameter*

Dari informasi di atas, kita bisa dapatkan informasi:

- $M = \{1, 2, .., 6\}$ adalah himpunan minggu dalam *timeframe* masalah ini.
- $\hat{M} = M \setminus \{1, 6\}$
- $P = P_1 \bigcup P_2 \bigcup \cdots \bigcup P_6$ adalah himpunan produk yang diproduksi selama 6 minggu tersebut.
- $\hat{P}$ adalah bagian dari P, berupa himpunan produk yang menggunakan paling sedikit dua jenis gula.
- $\dot{P}$ adalah bagian dari P, berupa himpunan produk yang menggunakan satu jenis gula saja.
- $G = \{1, 2, .., 6\}$ adalah himpunan jenis gula.
- D adalah kebutuhan gula di bulan perencanaan, yaitu: $week_3, week_4, week_5,$ dan $week_6$.
- $maxcap$ adalah kapasitas gudang atau tempat penyimpanan gula.
- $\forall i \in P_j$, g_{ijk} adalah kebutuhan gula (dalam ton) dari produk i di minggu ke k.
- $\forall k \in G$, c_k adalah harga gula (dalam ton) dari pemasok atau jenis ke k.
- $\forall k \in G$, ϵ_k adalah minimum order gula jenis k.
- $\forall k \in G$, $\hat{d}_{2k}$ adalah total kebutuhan gula jenis ke k di minggu ke 2.
- $\forall k \in G$, Z_{ik} adalah ***level of inventory*** dari gula jenis k di akhir minggu 1.

14.3.3 *Decision Variables*

Misalkan saya definisikan $\forall k \in G, x_k$ sebagai banyaknya gula jenis k (dalam ton) yang dibeli dari *supplier* k. Kita ketahui bahwa tidak ada kewajiban bagi Nutrifood untuk membeli semua jenis gula tersebut. Perlu diingat juga bahwa $x_k \in \mathbb{Z}$.

Artinya:

- $x_k = 0$ jika kita tidak membeli gula k
- $\epsilon_k \leq x_k \leq maxcap$ jika kita membeli gula.

Untuk menangani nilai diskontinu dari x_k, misalkan $\forall k \in G$ saya definisikan:

$$y_k = \begin{cases} 1, & x_k = 0 \\ 0, & \epsilon_k \leq x_k \leq maxcap \end{cases}$$

Lalu $\forall j \in M \setminus \{1, 6\}$, $\forall i \in P_j$, $\forall k \in G$, saya definisikan $\hat{x_{jk}}$ sebagai banyaknya pengiriman gula jenis k di awal minggu ke j.

Saya definisikan juga:

$$a_{ijk} = \begin{cases} 1, & \text{produk ke } i \text{ di minggu ke } j \text{ menggunakan gula jenis } k \\ 0, & \text{lainnya,} \end{cases}$$

$$b_{ijk} = \text{proporsi penggunaan gula jenis } k \text{ dari seluruh kebutuhan gula produk ke } i \text{ di minggu ke } j.$$

b_{ijk} berada di *range* $0 \leq b_{ijk} \leq 1$\$

Lalu $\forall j \in M \setminus \{1\}$, $\forall k \in G$, saya definisikan z_{jk} sebagai *level of inventory* gula jenis k di akhir minggu j, dengan $0 \leq z_{jk} \leq maxcap$.

14.3.4 *Constraints*

14.3.4.1 ***Constraint* I** memberikan hubungan yang benar antara variabel keputusan biner y_k dan variabel keputusan integer atau variabel kontinyu x_k yang berkaitan dengannya.

$\forall k \in G$,

$$x_k \leq D y_k$$

$$x_k \geq \epsilon_k y_k$$

$\forall j \in M \setminus \{1, 2\}$, $\forall i \in P_j$, $\forall k \in G$,

$$b_{ijk} \leq a_{ijk}$$

$$b_{ijk} \geq \mu a_{ijk}, \text{ untuk nilai } \mu \text{ yang sangat kecil.}$$

14.3.4.2 ***Constraint* II** dibuat agar total pemesanan gula tidak kurang dari *total demand* di bulan perencanaan.

$$\sum_{k \in G} x_k \geq D$$

14.3.4.3 ***Constraint* III** mengatur hubungan antara total pembelian gula dan pengiriman tiap minggu, $\forall k \in G$ maka:

$$x_k = \sum_{j \in \hat{M}} \hat{x}_{jk}$$

14.3.4.4 ***Constraint* IV** menjaga volume setiap jenis gula yang dikirim di pengiriman kedua, ketiga dan keempat selalu sama, $\forall k \in G$ maka:

$$\hat{x}_{3k} = \hat{x}_{4k} \text{ dan } \hat{x}_{4k} = \hat{x}_{5k}$$

14.3.4.5 ***Constraint* V** dibuat untuk menjaga komposisi gula yang diinginkan.

Pertama-tama untuk produk yang menggunakan lebih dari satu jenis gula, $\forall j \in M \setminus \{1, 2\}, \forall i \in \hat{P}_j$, maka:

$$\sum_{k \in G} a_{ijk} \geq 2$$

$$\sum_{k \in G} b_{ijk} = 1$$

Kedua untuk produk yang hanya bisa menggunakan satu jenis gula, $\forall j \in M \setminus \{1, 2\}, \forall i \in \dot{P}_j$, maka:

$$\sum_{k \in G} a_{ijk} = 1$$

$$\sum_{k \in G} b_{ijk} = 1$$

14.3.4.6 ***Constraint* VI** dibuat untuk menjaga *level of inventory* sesaat setelah pengiriman gula tidak melebihi kapasitas gudang atau tempat penyimpanan.

$$\sum_{k \in G} (Z_{1k} + \hat{x}_{1k} - \hat{d}_{2k} + z_{jk}) = maxcap$$

$\forall j \in M \setminus \{1, 2\}$,

$$\sum_{k \in G} (z_{(j-1)k} + \hat{x}_{(j-1)k}) - \sum_{i \in P_j} b_{ijk} g_{ijk} + z_{jk} = maxcap$$

14.3.5 *Objective Function*

Berdasarkan pembahasan di atas, maka masalah pemiihan pemasok gula dan volume pemesanannya dapat dituliskan menjadi masalah:

$$\min \sum_{k \in G} c_k x_k$$

Epilog