

Penelitian Mandiri Sains Komputasi I

Update Progress

Mohammad Rizka Fadhli
Ikang
20921004@mahasiswa.itb.ac.id

07 October 2021

Contents

1	CHAPTER I	7
	SEJARAH	7
1.1	Optimisasi	7
1.2	Riset Operasi	8
2	CHAPTER II	9
	JENIS-JENIS OPTIMISASI	9
2.1	Berdasarkan Tipe Variabel	9
2.2	Berdasarkan Karakteristik Masalah	10
3	CHAPTER III	11
	PENJELASAN LEBIH LANJUT TERKAIT JENIS OPTIMISASI	11
3.1	<i>Optimization Under Uncertainty</i>	11
3.2	<i>Continuous Optimization</i>	11
3.2.1	<i>Linear Programming</i>	12
3.2.2	<i>Non Linear Programming</i>	12
3.2.3	<i>Quadratic Programming</i>	13
3.3	<i>Discrete Optimization</i>	13
3.3.1	<i>Integer Programming</i>	14
3.3.2	<i>Mixed Integer Linear Programming (MILP)</i>	14
3.3.3	<i>Mixed Integer Non Linear Programming (MINLP)</i>	14
4	CHAPTER IV	16
	ALGORITMA OPTIMISASI	16
4.1	<i>Exact Method</i>	16
4.2	<i>Approximate Method</i>	17
5	CHAPTER V	19

<i>SIMPLEX METHOD</i>	19
5.1 Sejarah	19
5.2 Cara Kerja	19
5.2.1 Contoh Masalah Optimisasi	19
5.2.2 <i>Flowchart</i> Metode Simplex dari Contoh Masalah	22
5.3 Metode Simplex dengan Operasi Matriks	23
5.3.1 Contoh Penyelesaian Masalah Optimisasi dengan <i>Simplex</i>	23
5.4 <i>Post-Optimality Analysis</i>	26
5.5 <i>Sensitivity Analysis</i>	26
6 CHAPTER VI	28
BEBERAPA R <i>PACKAGES</i> YANG DIGUNAKAN UNTUK OPTIMISASI	28
6.1 ROI <i>Packages</i> di R	28
6.1.1 ROI <i>Modelling</i>	28
6.1.2 <i>Conclusion</i>	32
6.2 ompr <i>Packages</i> di R	33
6.2.1 ompr <i>Modelling</i>	33
6.2.2 <i>Conclusion</i>	36
7 CHAPTER VII	38
R <i>CODES FOR SIMPLEX METHOD</i>	38
7.1 library(<i>boot</i>)	38
7.1.1 Penggunaan	38
7.1.2 Catatan Khusus	39
7.1.3 Contoh	39
CHAPTER VIII	42
8 INTEGER DAN BINARY PROGRAMMING	42
8.1 <i>Integer Programming</i>	42
8.1.1 Contoh <i>Integer Programming</i>	42
8.2 <i>Binary Programming</i>	45

CHAPTER IX	47
9 MIXED INTEGER LINEAR PROGRAMMING	47
10 MILP IN SUPPLIER SELECTION	48
11 R CODES INTEGER PROGRAMMING	48
12 R CODES BINARY PROGRAMMING	48
13 SOLVING SUPPLIER SELECTION PROBLEM	48

List of Figures

1	Optimisasi Berdasarkan Jenis Variabel	9
2	Optimisasi Berdasarkan Kategori Masalah	10
3	Price Elasticity	13
4	Algoritma Penyelesaian Optimisasi	16
5	Grafik Permasalahan Optimisasi	20
6	Algoritma Metode Simplex	23

List of Tables

1	Titik yang termasuk ke dalam CPF	21
2	Adjacent CPF	22
3	Initial Condition Bentuk Matriks Simplex	24
4	Pemilihan Baris Pivot	24
5	OBE Iterasi 1	24
6	OBE Iterasi 2	25
7	Pemilihan Baris Pivot Kembali	25
8	OBE Iterasi 3	25
9	OBE Iterasi 4	26
10	Tabel Kebutuhan Nakes Harian	42
11	Konfigurasi Penjadwalan Nakes	43

UPDATE MINGGU I

1 CHAPTER I

SEJARAH

1.1 Optimisasi

Optimisasi adalah **proses mencari nilai yang optimal** dari suatu masalah tertentu. Dalam matematika, optimisasi merujuk pada pencarian nilai minimal atau maksimal dari suatu *fungsi real*¹. Notasi matematikanya dapat ditulis sebagai berikut:

Misalkan suatu fungsi f yang memetakan dari himpunan A ke bilangan *real*.

$$f : A \rightarrow \mathbb{R}$$

Cari suatu nilai $x_0 \in A$ sedemikian sehingga:

- $f(x_0) \leq f(x), \forall x \in A$ untuk proses **minimalisasi**.
- $f(x_0) \geq f(x), \forall x \in A$ untuk proses **maksimalisasi**.

Di dalam kalkulus, kita mengetahui salah satu pendekatan optimisasi di fungsi satu variabel bisa didapatkan dari turunan pertama yang bernilai **nol** (bisa berupa nilai maksimum atau minimum dari fungsi tersebut).

Nilai $x_0 \in [a, b]$ disebut minimum atau maksimum di f unimodal saat memenuhi:

$$\frac{d}{dx}f(x_0) = 0$$

Pierre De Fermat dan **Joseph-Louis Lagrange** adalah orang-orang yang pertama kali menemukan formula kalkulus untuk mencari nilai optimal. Sementara **Isaac Newton** dan **Johann C. F. Gauss** mengusulkan metode iteratif untuk mencari nilai optimal².

Salah satu bentuk optimisasi yakni *linear programming* dimulai oleh **Leonid Kantorovich** pada 1939. **Metode Simplex** merupakan salah satu metode penyelesaian optimisasi yang terkenal, pertama kali diperkenalkan pada 1947 oleh **George Dantzig** sementara di tahun yang sama *Theory of Duality* diperkenalkan oleh **John von Neumann**.

¹<https://id.wikipedia.org/wiki/Optimisasi>

²<https://empowerops.com/en/blogs/2018/12/6/brief-history-of-optimization>

Masalah optimisasi adalah masalah matematika yang mewakili masalah nyata (*real*). Dari ekspresi matematika tersebut, ada beberapa hal yang perlu diketahui³, yakni:

1. **Variabel** adalah suatu simbol yang memiliki banyak nilai dan nilainya ingin kita ketahui. Setiap nilai yang mungkin dari suatu variabel muncul akibat suatu kondisi tertentu di sistem.
2. **Parameter** di suatu model matematika adalah suatu konstanta yang menggambarkan suatu karakteristik dari sistem yang sedang diteliti. Parameter bersifat *fixed* atau *given*.
3. **Constraints** (atau kendala) adalah kondisi atau batasan yang harus dipenuhi. Kendala-kendala ini dapat dituliskan menjadi suatu persamaan atau pertaksamaan. Suatu masalah optimisasi dapat memiliki hanya satu kendala atau banyak kendala.
4. **Objective function** adalah satu fungsi (pemetaan dari variabel-variabel keputusan ke suatu nilai di daerah *feasible*) yang nilainya akan kita minimumkan atau kita maksimumkan.

Ekspresi matematika dari model optimisasi adalah sebagai berikut:

Cari x yang meminimumkan atau memaksimumkan $f_i(x)$, $i = 1, \dots, m$ dengan kendala $g_j(x)$, $j = 1, \dots, n$ dan $x \geq 0$.

Dari ekspresi tersebut, kita bisa membagi-bagi masalah optimisasi tergantung dari:

1. Tipe variabel yang terlibat.
2. Jenis fungsi yang ada (baik *objective function* ataupun *constraint*).

1.2 Riset Operasi

Riset operasi adalah metode antar disiplin ilmu yang digunakan untuk menganalisa masalah nyata dan membuat keputusan untuk kegiatan operasional organisasi atau perusahaan⁴. Riset operasi dimulai pada era Perang Dunia II. Oleh karena peperangan, diperlukan suatu cara yang efektif untuk mengalokasikan *resources* yang ada sehingga pihak militer Inggris dan Amerika Serikat mengumpulkan ilmuwan-ilmuwan untuk mencari pendekatan yang saintifik dalam memecahkan masalah⁵.

³Pengantar Riset Operasi dan Optimisasi, KampusX: PO101

⁴Pengantar Riset Operasi dan Optimisasi, KampusX: PO101

⁵Introduction to Operations Research, 7th Edition. Hillier / Lieberman hal. 1

2 CHAPTER II

JENIS-JENIS OPTIMISASI

2.1 Berdasarkan Tipe Variabel

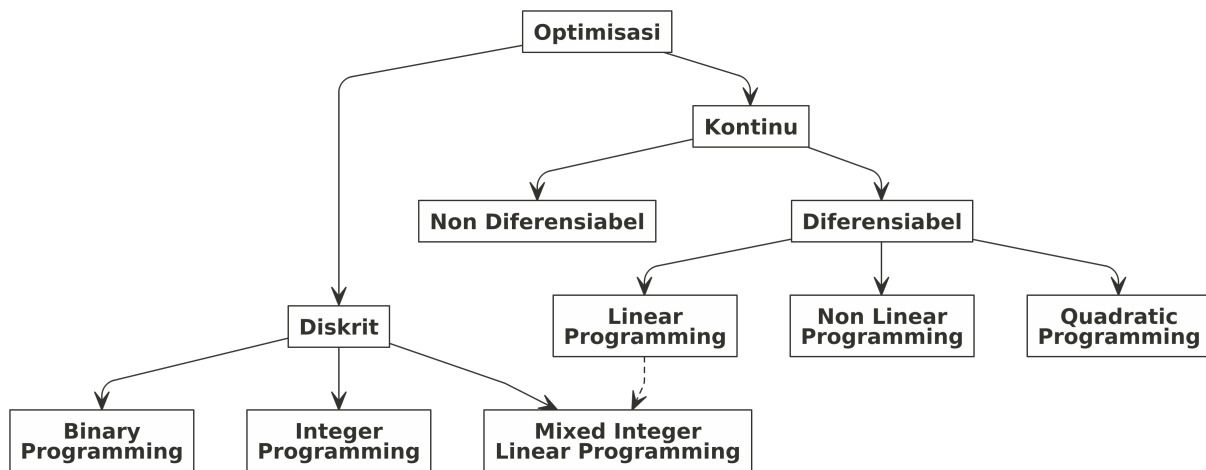


Figure 1: Optimisasi Berdasarkan Jenis Variabel

Masalah optimisasi bisa dibagi dua menjadi dua kategori berdasarkan tipe *variables* yang terlibat⁶, yakni:

1. *Discrete Optimization*: merupakan masalah optimisasi di mana variabel yang terkait merupakan variabel diskrit, seperti *binary* atau *integer*.
2. *Continuous Optimization*: merupakan masalah optimisasi saat fungsi *objective* kontinu (termasuk fungsi yang *multimodal*).
 - Sebagaimana yang telah kita pelajari di kalkulus, suatu fungsi kontinu **bisa terdiferensiabel** atau **tidak terdiferensiabel** di selang tertentu.
 - Hal ini mengakibatkan pendekatan yang berbeda juga dalam penyelesaian optimisasinya. Pendekatan penyelesaian fungsi yang terdiferensiabel menggunakan *gradient* kurva fungsi di titik tertentu.

⁶Optimization problem. https://en.wikipedia.org/wiki/Optimization_problem

2.2 Berdasarkan Karakteristik Masalah

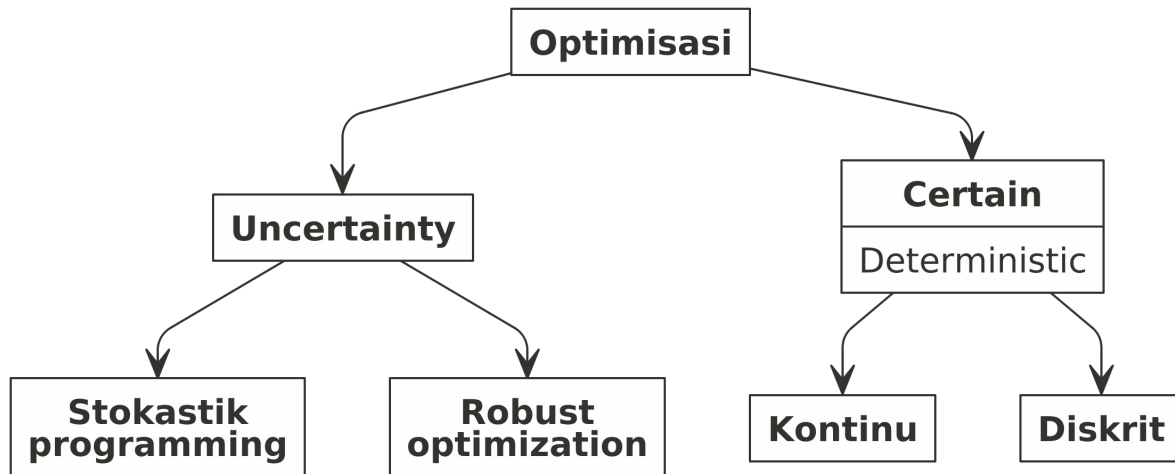


Figure 2: Optimisasi Berdasarkan Kategori Masalah

Selain itu, kita juga bisa membagi masalah optimisasi berdasarkan **kategori masalah** yang dihadapi sebagai berikut:

1. *Optimization under uncertainty*⁷; Pada beberapa kasus di dunia *real*, data dari masalah tidak dapat diketahui secara akurat karena berbagai alasan. Hal ini mungkin terjadi akibat:
 - Kesalahan dalam pengukuran, atau
 - Data melibatkan sesuatu di masa depan yang belum terjadi atau tidak pasti. Contoh: *demand* produk, harga barang, dan sebagainya.
2. *Deterministic optimization*;
 - Model deterministik adalah model matematika di mana nilai dari semua parameter dan variabel yang terkandung di dalam model merupakan satu nilai pasti⁸.
 - Pendekatan deterministik memanfaatkan sifat analitik masalah untuk menghasilkan barisan titik yang konvergen ke solusi optimal⁹.
 - Semua algoritma perhitungan mengikuti pendekatan matematis yang ketat¹⁰.

⁷<https://neos-guide.org/content/optimization-under-uncertainty>

⁸Pengantar Riset Operasi dan Optimisasi, KampusX: PO101

⁹<https://www.hindawi.com/journals/mpe/2012/756023/>

¹⁰https://link.springer.com/chapter/10.1007/978-3-642-31187-1_4

3 CHAPTER III

PENJELASAN LEBIH LANJUT TERKAIT JENIS OPTIMISASI

3.1 *Optimization Under Uncertainty*

Untuk menyelesaikan masalah *optimization under uncertainty*, ada dua metode yang *frameworks* digunakan. Yakni:

1. *Stochastic Programming*.

- Pendekatan yang digunakan adalah menggunakan variabel acak yang memiliki distribusi probabilitas tertentu untuk mengkarakterisasi *uncertainty* dan mengoptimalkan nilai yang diharapkan dari fungsi tujuan.

2. *Robust Optimization*.

- Berbeda dengan *stochastic programming* yang memilih pendekatan probabilistik, *robust optimization* menggunakan pendekatan basis himpunan *uncertainty*¹¹.

3.2 *Continuous Optimization*

Di dalam *continuous optimization*, variabel di dalam model diperbolehkan berisi semua nilai dalam selang yang diberikan. Berbeda dengan *discrete optimization* yang biasanya memiliki variabel berisi *binary* atau *integer*¹².

Oleh karena *continuous optimization* bekerja pada fungsi yang kontinu, maka optimisasi ini erat dengan teknik-teknik kalkulus. Seperti yang telah dijelaskan pada bagian sebelumnya, fungsi kontinu bisa jadi terdiferensiabel atau tidak terdiferensiabel dalam suatu selang. Hal ini memiliki konsekuensi terkait bagaimana kita menyelesaikan masalah optimisasinya.

Salah satu pembeda masalah optimisasi kontinu dengan masalah kontinu lainnya adalah pada keberadaan *constraints* di variabelnya, sehingga masalah optimisasi kontinu bisa dikelompokkan menjadi:

1. *Unconstrained optimization* adalah optimisasi yang **tidak mengandung** *constraints* pada *decision variable*-nya.
2. *Constrained optimization* adalah optimisasi yang **mengandung** *constraints* pada *decision variable*-nya.

Namun demikian, kita bisa mengubah model matematika dari *constrained optimization* menjadi *unconstrained optimization* dengan menambahkan syarat pada fungsi objektifnya.

¹¹https://stanford.edu/class/ee364b/lectures/robust_notes.pdf

¹²<https://neos-guide.org/content/continuous-optimization>

3.2.1 Linear Programming

Linear programming adalah bentuk metode optimisasi sederhana yang memanfaatkan relasi linear (semua fungsi dan *constraints* merupakan fungsi linear).

Contoh Masalah Linear Programming Saya memiliki area parkir seluas $1.960 m^2$. Luas rata-rata untuk mobil berukuran kecil adalah $4 m^2$ dan mobil besar adalah $20 m^2$. Daya tampung maksimum hanya 250 kendaraan, biaya parkir mobil kecil adalah Rp 7.000 per jam dan mobil besar adalah Rp 12.000 per jam. Jika dalam 1 jam area parkir saya terisi penuh dan tidak ada kendaraan yang pergi dan datang, maka berapa pendapatan maksimum yang bisa saya dapatkan dari tempat parkir itu?

Dari kasus di atas kita bisa tuliskan model matematikanya sebagai berikut:

Misal x_1 adalah mobil kecil dan x_2 adalah mobil besar.

$$\max(7000x_1 + 12000x_2)$$

Dengan *constraints*:

$$4x_1 + 20x_2 \leq 1960$$

dan

$$x_1 + x_2 \leq 250$$

serta $x_1 \geq 0, x_2 \geq 0$.

3.2.2 Non Linear Programming

Beberapa permasalahan tidak bisa ditulis sebagai fungsi linear, oleh karena itu muncullah permasalahan *non linear programming*. Bentuk dasar pemodelannya sama dengan *linear programming* tapi ada fungsi yang tidak linear di dalamnya. Sebagai contoh fungsi yang terlibat ada logaritmik atau fungsi trigonometri.

Contoh Masalah Non Linear Programming Suatu perusahaan menjual suatu produk yang memiliki model *price elasticity* seperti di bawah ini:

Di mana $p(x)$ adalah harga saat menjual produk sebanyak x .

Omset perusahaan tersebut didefinisikan sebagai **terjual x harga**, yakni $xp(x)$. Sedangkan profit adalah omset dikurangi dengan biaya produksi per produk yang terjual.

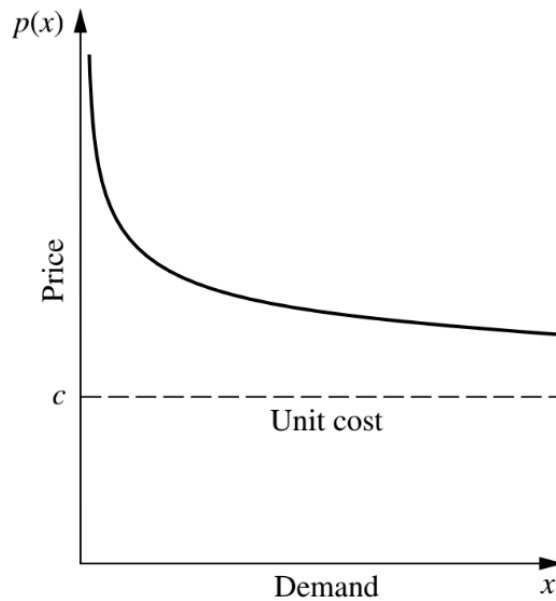


Figure 3: Price Elasticity

Sehingga didapatkan:

$$P(x) = xp(x) - cx, \text{ dengan } c \text{ adalah fixed cost}$$

Karena $p(x)$ non linear, maka didapatkan $P(x)$ juga non linear¹³.

3.2.3 Quadratic Programming

Lantas apa perbedaan *linear programming*, *non linear programming* dengan *quadratic programming*?

Perbedaan mencoloknya ada pada perkalian antar variabel pada *objective function*¹⁴. Misalkan x_j^2 atau $x_i x_j$, untuk $(i \neq j)$.

3.3 Discrete Optimization

Di dalam *discrete programming*, variabel-variabel yang terlibat di dalamnya harus bertipe diskrit. Bisa dalam bentuk *binary*, *integer*, atau kombinasi antara keduanya¹⁵.

¹³Introduction to Operations Research, 7th Edition. Hillier / Lieberman hal. 654

¹⁴Introduction to Operations Research, 7th Edition. Hillier / Lieberman hal. 683

¹⁵Introduction to Operations Research, 7th Edition. Hillier / Lieberman hal. 576

3.3.1 *Integer Programming*

Pada *integer programming*, semua variabel yang terkait **wajib** berupa *integer* atau *binary*¹⁶.

Contoh Masalah *Integer Programming* adalah *nurse schedulling problem*, yakni bagaimana memasang perawat tepat ke *shift* kerja yang tersedia¹⁷.

3.3.2 *Mixed Integer Linear Programming* (MILP)

Pada **MILP**, sebagian variabel memiliki tipe *binary* atau *integer* dan sebagian lainnya diperbolehkan bertipe kontinu (*real values*)¹⁸.

Contoh Masalah *Mixed Integer Linear Programming* adalah masalah pemilihan produksi yang optimal¹⁹ di mana ada dua jenis variabel keputusan, yakni numerik dan *binary*.

3.3.3 *Mixed Integer Non Linear Programming* (MINLP)

Pada **MINLP**, variabel yang terkait bisa berbentuk *integer* atau *real values* serta fungsi yang ada berupa non linear²⁰.

¹⁶<https://neos-guide.org/content/integer-linear-programming>

¹⁷<https://ikanx101.com/blog/nurse-schedulling/>

¹⁸<https://neos-guide.org/content/integer-linear-programming>

¹⁹<https://ikanx101.com/blog/produk-baru/>

²⁰<https://neos-guide.org/content/Mixed-Integer-Nonlinear-Programming>

UPDATE MINGGU II

4 CHAPTER IV

ALGORITMA OPTIMISASI

Pada bagian ini kita akan membahas macam-macam algoritma yang digunakan untuk menyelesaikan masalah optimisasi.

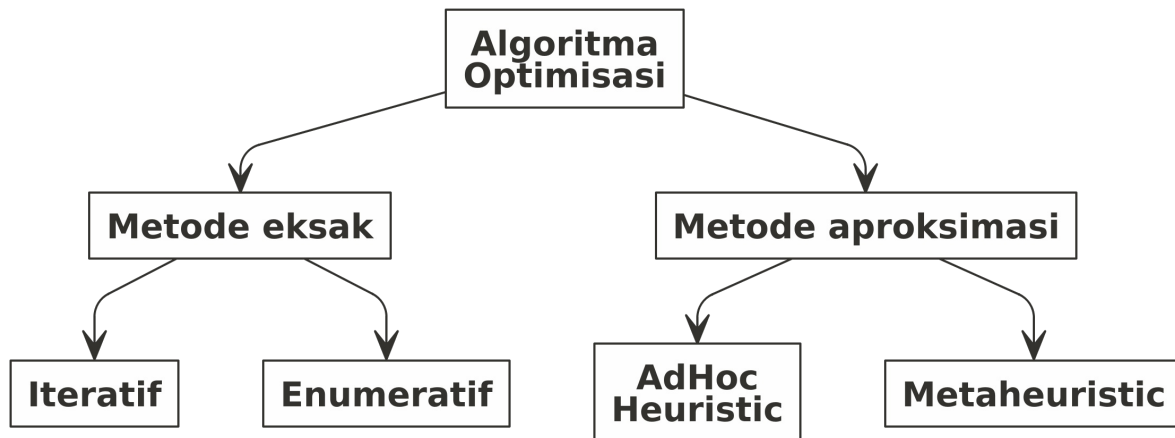


Figure 4: Algoritma Penyelesaian Optimisasi

Secara garis besar ada dua kelompok besar algoritma optimisasi, yakni:

1. *Exact method*,
2. *Approximate method*.

Perbedaan keduanya adalah pada **konsep atau pendekatan apa yang digunakan** untuk menyelesaikan masalah optimisasi. Kita akan bahas satu-persatu pada bagian selanjutnya.

Dalam beberapa kasus, kita bisa mendapatkan *exact method* bisa untuk menyelesaikan masalah optimisasi dengan efisien. Namun di kasus lain yang lebih kompleks tidak demikian. Kelemahan utama metode *exact* adalah pada waktu komputasinya yang relatif lebih lama²¹.

4.1 *Exact Method*

Ciri khas dari *exact method* adalah metode ini menjamin penyelesaian yang optimal karena menggunakan penyelesaian analitis metode matematika²².

²¹<http://hpurnomo.blog.uksw.edu/2012/06/metode-optimisasi.html>

²²<https://hindriyanto.wordpress.com/2012/09/26/antara-optimisasi-heuristik-dan-metaheuristik/>

4.2 *Approximate Method*

Ciri khas dari *approximate method* adalah metode ini tidak menjamin penyelesaian yang optimal karena bersifat *aproksimasi* atau pendekatan atau hampiran²³. Oleh karena itu kita perlu melakukan definisi di awal **seberapa dekat** nilai **hampiran** tersebut bisa kita terima.

Metode ini bisa dibagi menjadi dua berdasarkan keterkaitannya dengan suatu masalah, yakni:

1. *Heuristic*, metode ini bersifat *problem dependent*. Artinya metode tersebut hanya bisa dipakai untuk jenis permasalahan tertentu.
 - Contoh: metode *nearest neighborhood* hanya bisa dipakai untuk menyelesaikan masalah dalam lingkup *travelling salesperson problem* (**TSP**).
2. *Meta heuristic*, metode ini bersifat *problem independent*. Artinya metode tersebut tidak tergantung dari jenis permasalahan tertentu.
 - Contoh: *genetic algorithm* bisa dipakai untuk berbagai jenis permasalahan.

Namun demikian kedua metode ini bisa saling melengkapi dalam prakteknya.

²³<https://www.math.unipd.it/~luigi/courses/metmodoc1819/m02.meta.en.partial01.pdf>

UPDATE MINGGU III

5 CHAPTER V

SIMPLEX METHOD

5.1 Sejarah

Metode *simplex* adalah salah satu metode yang paling umum digunakan dalam menyelesaikan permasalahan *linear programming*. Metode ini dikembangkan oleh seorang profesor matematika bernama George Dantzig²⁴ pada 1947 pasca perang dunia II. Sedangkan nama *simplex* diusulkan oleh Theodore Motzkin²⁵.

5.2 Cara Kerja

Metode *simplex* menggunakan prosedur aljabar²⁶. Namun *underlying concept* dari metode ini adalah *geometric*.

Jika kita bisa memahami konsep geometrinya, kita bisa mengetahui bagaimana cara kerjanya dan kenapa metode ini sangat efisien.

Saya akan ambil satu contoh masalah optimisasi sederhana untuk memberikan ilustrasi bagaimana cara kerja metode ini.

5.2.1 Contoh Masalah Optimisasi

Cari x_1, x_2 yang $\max (Z = 3x_1 + 5x_2)$ dengan *constraints*:

$$\begin{aligned}x_1 &\leq 4 \\2x_2 &\leq 12 \\3x_1 + 2x_2 &\leq 18 \\ \text{serta } x_1 &\geq 0, x_2 \geq 0\end{aligned}$$

Masalah di atas jika dibuat grafiknya:

²⁴https://en.wikipedia.org/wiki/George_Dantzig

²⁵https://en.wikipedia.org/wiki/Theodore_Motzkin

²⁶Introduction to Operations Research, 7th Edition. Hillier / Lieberman hal. 109

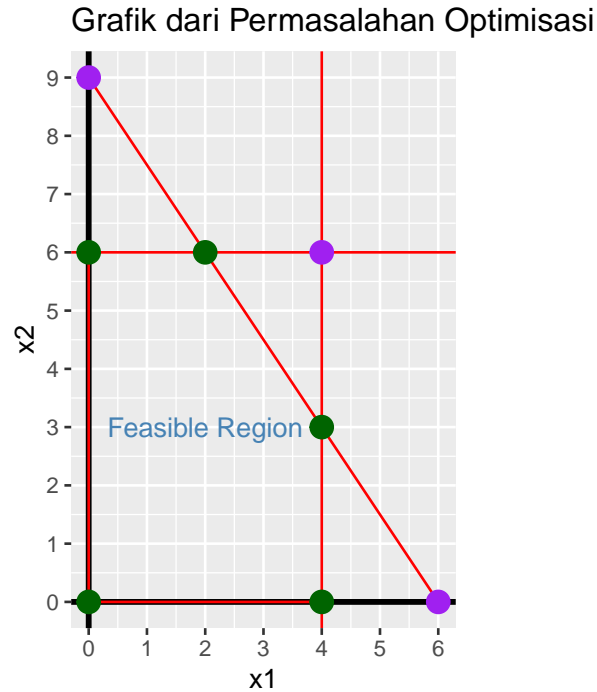


Figure 5: Grafik Permasalahan Optimisasi

Titik-titik hijau merupakan **beberapa titik** solusi yang *feasible* karena berada pada area penerimaan seluruh *constraints* yang ada. Titik hijau ini menjadi spesial karena berada pada perpotongan 2 garis *constraints*. Selanjutnya titik hijau ini akan didefinisikan sebagai **CPF** (*corner point feasible*).

*For a linear programming problem with n decision variables, each of its corner-point solutions lies at the intersection of n constraint boundaries.*²⁷

Sedangkan titik ungu merupakan titik solusi non *feasible* karena solusi yang ada tidak berlaku untuk semua *constraints*.

²⁷Introduction to Operations Research, 7th Edition. Hillier / Lieberman hal. 110

Table 1: Titik yang termasuk ke dalam CPF

Titik.ke	CPF
1	(0, 0)
2	(0, 6)
3	(2, 6)
4	(4, 3)
5	(4, 0)

Properties of CPF Solutions Untuk setiap permasalahan *linear programming* yang memiliki *feasible solutions* dan *feasible region* yang terbatas:

Property 1: (a) If there is exactly one optimal solution, then it must be a CPF solution. (b) If there are multiple optimal solutions (and a bounded feasible region), then at least two must be adjacent CPF solutions.

Property 2: There are only a finite number of CPF solutions.

Property 3: If a CPF solution has no adjacent CPF solutions that are better (as measured by Z), then there are no better CPF solutions anywhere. Therefore, such a CPF solution is guaranteed to be an optimal solution (by Property 1), assuming only that the problem possesses at least one optimal solution (guaranteed if the problem possesses feasible solutions and a bounded feasible region).

Untuk mulai melakukan metode simplex kita perhatikan kembali grafik di atas. Kita bisa temukan beberapa pasang **CPF** berbagi *constraint* yang sama satu sama lain.

Sebagai contoh:

1. CPF_1 dan CPF_2 berbagi *constraint* yang sama, yakni saat $x_1 \geq 0$.
2. CPF_2 dan CPF_3 berbagi *constraint* yang sama, yakni saat $x_2 \leq 6$.

Definisi umum:

*For any linear programming problem with n decision variables, two CPF solutions are **adjacent** to each other if they share $n - 1$ constraint boundaries. The two adjacent CPF solutions are connected by a line segment that lies on these same shared constraint boundaries. Such a line segment is referred to as an **edge** of the feasible region.*

Feasible region di atas memiliki 5 *edges* di mana setiap 2 *edges* memotong / memunculkan **CPF**. Setiap **CPF** memiliki 2 **CPF** lainnya yang *adjacent*.

Table 2: Adjacent CPF

Titik.ke	CPF	Adjacent.CPF
1	(0, 0)	(0, 6) dan (4, 0)
2	(0, 6)	(2, 6) dan (0, 0)
3	(2, 6)	(4, 3) dan (0, 6)
4	(4, 3)	(4, 0) dan (2, 6)
5	(4, 0)	(0, 0) dan (4, 3)

CPF pada kolom pertama *adjacent* terhadap dua **CPF** di kolom setelahnya tapi kedua **CPF** tersebut tidak saling *adjacent* satu sama lain.

Optimality test: Consider any linear programming problem that possesses at least one optimal solution. If a CPF solution has no adjacent **CPF** solutions that are better (as measured by Z), then it must be an optimal solution.

Berdasarkan *optimality test* tersebut, kita bisa mencari solusi optimal dari **CPF** dengan cara mengambil **initial CPF** untuk dites secara rekursif.

- **STEP 1** Pilih *initial CPF*, misal (0, 0). Kita akan hitung nilai $Z(0, 0) = 0$. Bandingkan dengan *adjacent CPF*-nya, yakni $Z(0, 6) = 30$ dan $Z(4, 0) = 12$.
- **STEP 2** Oleh karena $Z(0, 6)$ memiliki nilai tertinggi, maka kita akan pilih titik ini di iterasi pertama. Kita akan bandingkan terhadap *adjacent CPF*-nya, yakni: $Z(2, 6) = 36$. Perhatikan bahwa *adjacent CPF* (0, 0) sudah kita evaluasi pada langkah sebelumnya.
- **STEP 3** Oleh karena $Z(2, 6)$ memiliki nilai tertinggi, maka kita akan pilih titik ini di iterasi kedua. Kita akan bandingkan terhadap *adjacent CPF*-nya, yakni: $Z(4, 3) = 27$. Kita dapatkan bahwa titik (2, 6) menghasilkan Z tertinggi.

Kesimpulan: (2, 6) merupakan titik yang bisa memaksimumkan Z .

5.2.2 Flowchart Metode Simplex dari Contoh Masalah

Secara garis besar, *flowchart* dari metode simplex untuk masalah di atas adalah:

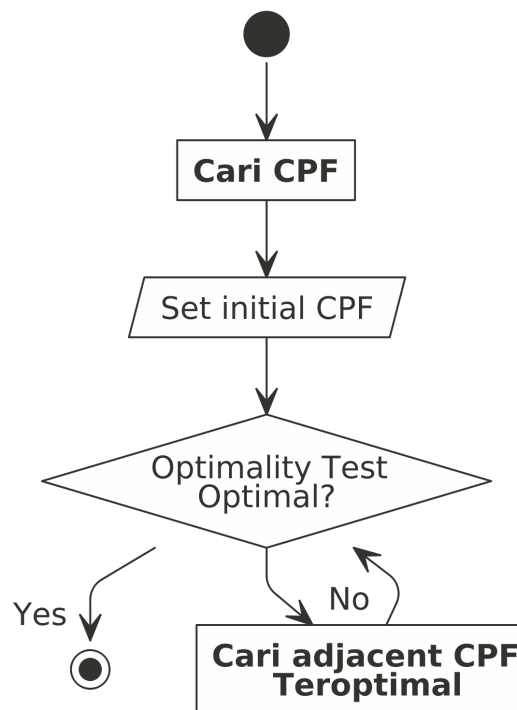


Figure 6: Algoritma Metode Simplex

Algoritma di atas akan sangat mudah dilakukan saat kita berhadapan dengan masalah optimisasi dengan 2 *decision variables* (atau 3 *decision variables*). Pada contoh di atas ada x_1, x_2 .

Bagaimana jika masalah yang dihadapi memiliki banyak *decision variables*?

Tentunya kita tidak bisa melakukan analisa secara visual seperti di atas. Namun kita bisa menggunakan bantuan aljabar dan operasi baris elementer untuk menemukan solusi yang optimal.

5.3 Metode Simplex dengan Operasi Matriks

Suatu masalah optimisasi bisa kita tulis dalam bentuk matriks sehingga bisa diselesaikan dengan melakukan *operasi baris elementer*.

5.3.1 Contoh Penyelesaian Masalah Optimisasi dengan *Simplex*

Cari x, y sehingga $\max (P = 5x + 4y)$ dengan *constraints*:

$$\begin{aligned} 3x + 5y &\leq 78 \\ 4x + y &\leq 36 \\ \text{serta } x &\geq 0, y \geq 0 \end{aligned}$$

Untuk menyelesaikannya, kita perlu menambahkan u, w sebagai variabel pembantu. Fungsi objektif P juga harus diubah (dipindah sisi namun P tetap positif).

$$\begin{aligned} 3x + 5y + u &\leq 78 \\ 4x + y + w &\leq 36 \\ -5x - 4y + P &= 0 \end{aligned}$$

Setelah itu kita buat matriks (dalam hal ini saya akan buat tabelnya) sebagai berikut:

Table 3: Initial Condition Bentuk Matriks Simplex

x	y	u	w	P	b
3	5	1	0	0	78
4	1	0	1	0	36
-5	-4	0	0	1	0

STEP 1 Kita akan pilih kolom yang memiliki nilai **negatif terbesar** pada baris terakhir, yakni kolom x . Selanjutnya kita akan pilih baris mana yang akan menjadi pivot dengan cara menghitung rasio $\frac{b}{x}$ untuk semua baris dan memilih baris dengan **rasio terendah**.

Table 4: Pemilihan Baris Pivot

x	y	u	w	P	b	rasio
3	5	1	0	0	78	26
4	1	0	1	0	36	9
-5	-4	0	0	1	0	0

STEP 2 Kita akan buat baris 2 kolom x menjadi bernilai 1, caranya dengan melakukan OBE seperti: $Row_2 = \frac{Row_2}{4}$.

Table 5: OBE Iterasi 1

x	y	u	w	P	b
3	5.00	1	0.00	0	78
1	0.25	0	0.25	0	9
-5	-4.00	0	0.00	1	0

STEP 3 Sekarang tujuan kita selanjutnya adalah membuat kolom x baris 1 dan 3 menjadi bernilai **nol**. Caranya adalah:

$$Row_1 = Row_1 - 3Row_2$$

$$Row_3 = Row_3 + 5Row_2$$

Table 6: OBE Iterasi 2

x	y	u	w	P	b
0	4.25	1	-0.75	0	51
1	0.25	0	0.25	0	9
0	-2.75	0	1.25	1	45

STEP 4 Kita akan lakukan hal yang sama pada *step 1*, yakni memilih kolom dengan negatif terbesar. Yakni kolom y . Lalu kita akan hitung rasio setiap baris dan akan memilih rasio paling rendah.

Table 7: Pemilihan Baris Pivot Kembali

x	y	u	w	P	b	rasio
0	4.25	1	-0.75	0	51	12.00000
1	0.25	0	0.25	0	9	36.00000
0	-2.75	0	1.25	1	45	-16.36364

STEP 5 Maka kita akan pilih baris 1 menjadi pivot. Kolom y pada baris 1 harus bernilai 1 sehingga kita harus membuat $Row_1 = \frac{4Row_1}{17}$.

Table 8: OBE Iterasi 3

x	y	u	w	P	b
0	1.00	0.2352941	-0.1764706	0	12
1	0.25	0.0000000	0.2500000	0	9
0	-2.75	0.0000000	1.2500000	1	45

STEP 6 Kita akan buat kolom y di baris 2 dan 3 menjadi **nol** dengan cara:

$$Row_2 = Row_2 - \frac{Row_1}{4}$$

$$Row_3 = Row_3 + \frac{11Row_1}{4}$$

Table 9: OBE Iterasi 4

x	y	u	w	P	b
0	1	0.2352941	-0.1764706	0	12
1	0	-0.0588235	0.2941176	0	6
0	0	0.6470588	0.7647059	1	78

Dari tabel terakhir di atas, kita bisa menuliskan $x = 6, y = 12$ dan nilai $\max(P) = 78$. Bagaimana dengan nilai u dan w ? Karena tidak ada nilai 1 ditemukan pada kolom variabel tersebut, kita bisa simpulkan bahwa $u = 0, w = 0$.

5.4 *Post-Optimality Analysis*

Post-optimality analysis adalah analisa yang dilakukan pasca kita telah menemukan *optimal solution* dari hasil perhitungan. Contohnya kita bisa melakukan **reoptimisasi**.

5.5 *Sensitivity Analysis*

Salah satu proses dalam membuat model optimisasi adalah *parameter estimation*. Ada kalanya perubahan data mengakibatkan berubahnya suatu parameter. *Sensitivity analysis* bertujuan untuk mengidentifikasi parameter yang sensitif (parameter yang harus dihitung dengan baik untuk menghindari kesalahan saat mencari solusi optimal).

UPDATE MINGGU IV

6 CHAPTER VI

BEBERAPA R *PACKAGES* YANG DIGUNAKAN UNTUK OPTIMISASI

Untuk menyelesaikan masalah optimisasi menggunakan **R**, ada beberapa *packages* yang bisa digunakan. Saya akan bahas beberapa *packages* yang biasa digunakan untuk menyelesaikan masalah optimisasi di **R**, yakni:

1. ROI *packages*.
2. ompr *packages*.

6.1 ROI *Packages* di R

ROI merupakan singkatan dari **R Optimization Infrastructure** merupakan salah satu *packages* yang memberikan infrastruktur untuk menyelesaikan *linear programming*, *quadratic programming*, *conic*, dan *general non linear programming*.

ROI dikembangkan oleh **WU Vienna University of Economics and Business**²⁸, yakni:

- Kurt Hornik,
- David Meyer,
- Florian Schwendinger,
- Stefan Theussl,
- Diethelm Wuertz.

ROI bekerja dengan memanfaatkan berbagai *solver* (disebut dengan *plugins*) yang dikembangkan oleh pihak-pihak lain. Dari masalah yang ada, kita **bisa melihat dan menentukan** *solver* apa yang bisa kita pakai untuk menyelesaikan permasalahan tersebut.

6.1.1 ROI *Modelling*

Framework untuk menuliskan model optimisasi menggunakan ROI adalah sebagai berikut:

Objective Function dimasukkan ke dalam *script* dengan format tergantung dari masalah yang dihadapi:

1. Jika berupa *linear programming*, *objective function* akan berupa **vector** numerik.
2. Jika berupa *quadratic programming*, *objective function* akan berupa **matriks**.

²⁸<https://epub.wu.ac.at/5858/>

Constraints dalam ROI dimasukkan dalam bentuk pisahan berikut:

$$(parameter) + (direction) + (rhs)$$

Bounds atau batas *decision variables* termasuk tipenya (*integer* dan *kontinu*).

6.1.1.1 Contoh Penyelesaian *Linear Programming* I

$$\text{minimize: } 7x_1 + 8x_2$$

$$\begin{aligned} \text{subject to: } & 3x_1 + 4x_2 = 9 \\ & 2x_1 + x_2 \geq 3 \\ & -100 \leq x_1, x_2 \leq 100 \end{aligned}$$

Berikut adalah penyelesaian di **R**:

```
rm(list=ls())

# libraries dan solver yang digunakan
library(ROI)
library(ROI.plugin.glpk)
library(ROI.plugin.qpoases)
library(ROI.plugin.ecos)
library(ROI.plugin.scs)
library(ROI.plugin.alabama)
library(ROI.plugin.lpsolve)

# pendefinisian objective function
obj_func = L_objective(c(7, 8), names=c("x", "y"))
obj_func
```

A linear objective of length 2.

```
# pendefinisian constraints
const = L_constraint(L = rbind(c(3, 4),
                                c(2, 1)),
                     dir = c("==", ">="),
                     rhs = c(9, 3)
                     )
const
```

An object containing 2 linear constraints.

```
# pendefinisian bounds
bou = V_bound(li = 1:2, # x dan y
              ui = 1:2, # x dan y
              lb = c(-100, -100),
              ub = c(100, 100))
bou
```

```
## ROI Variable Bounds:
##
## 2 lower and 2 upper non-standard variable bounds.
```

```
# menggabungkan semua komponen
linear_problem = OP(objective = obj_func,
                    constraints = const,
                    bounds = bou)
linear_problem
```

```
## ROI Optimization Problem:
##
## Minimize a linear objective function of length 2 with
## - 2 continuous objective variables,
##
## subject to
## - 2 constraints of type linear.
## - 2 lower and 2 upper non-standard variable bounds.
```

Untuk melihat *solver* ROI, kita akan menggunakan perintah sebagai berikut:

```
ROI_applicable_solvers(linear_problem)
```

```
## [1] "alabama" "ecos"      "glpk"      "lpsolve" "qpoases" "scs"
```

Terlihat ada 6 *solvers* yang bisa dipilih. Proses mencari solusi dilakukan dengan perintah sebagai berikut:

```
ROI_solve(model ROI, solver = "nama solver")
```

Sebagai contoh, saya akan menggunakan *solver* **glpk**, maka:

```
solusi = ROI_solve(linear_problem,
                   solver = "glpk")
```

Berikut adalah hasilnya:

```
solution(solusi)
```

```
##      x      y
## 0.6 1.8
```

6.1.1.2 Contoh Penyelesaian *Linear Programming* II

maximize: $7x_1 + 3x_2 + x_3$

subject to:

$$\begin{aligned} 6x_1 + 4x_2 + 5x_3 &\leq 60 \\ 8x_1 + x_2 + 2x_3 &\leq 80 \\ 9x_1 + x_2 + 7x_3 &\leq 70 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Berikut adalah penyelesaiannya di **R**:

```
linear_problem_2 =
  OP(# objective function
    objective = L_objective(c(7, 1, 3),
                           names = c("x", "y", "z")),
    # constraints
    constraints = L_constraint(L = rbind(c(6, 4, 5),
                                          c(8, 0, 2),
                                          c(9, 1, 7)
                                         ),
                              dir = c("<=", "<=", "<="),
                              rhs = c(60, 80, 70)),
    # tidak ada bounds yang perlu diset
    # pengaturan agar memaksimalkan obj function
    maximum = TRUE)

# proses pencarian solusi
# tanpa memanggil solver (default)
solusi_2 = ROI_solve(linear_problem_2)
# output
solution(solusi_2)
```

```
##      x      y      z
## 7.333333 4.000000 0.000000
```

Perlu diperhatikan pada baris-baris akhir, `linear_problem_2` diselesaikan tanpa kita harus memanggil *solver* yang ada. ROI akan memilihkan *default solver*.

6.1.1.3 Contoh Penyelesaian *Mixed Integer Linear Programming*

$$\begin{aligned} &\text{maximize: } 7x_1 + 3x_2 + x_3 \\ &6x_1 + 4x_2 + 5x_3 \leq 60 \\ &8x_1 + x_2 + 2x_3 \leq 80 \\ &\text{subject to: } 9x_1 + x_2 + 7x_3 \leq 70 \\ &x_3 \geq 0 \\ &x_1, x_2 \in \mathbb{Z}_{\geq 0} \end{aligned}$$

Masalah kali ini adalah perpaduan antara variabel kontinu dan *integer*.

Kita cukup memodifikasi perintah di **R** untuk mendefinisikan tipe variabel yang terlibat sebagai berikut:

```
mixed_ilm =
  OP(# objective function
    objective = L_objective(c(7, 1, 3),
                           names = c("x", "y", "z")),
    # constraints
    constraints = L_constraint(L = rbind(c(6, 4, 5),
                                         c(8, 0, 2),
                                         c(9, 1, 7)
                                         ),
                              dir = c("<=", "<=", "<="),
                              rhs = c(60, 80, 70)),
    # tidak ada bounds yang perlu diset

    # pendefinisian tipe variabel
    types = c("I","I","C"),
    # pengaturan agar memaksimalkan obj function
    maximum = TRUE)

# proses pencarian solusi
# tanpa memanggil solver (default)
solusi_3 = ROI_solve(mixed_ilm)
# output
solution(solusi_3)
```

```
##   x   y   z
## 7.0 4.0 0.4
```

6.1.2 Conclusion

Salah satu ciri khas dalam ROI adalah *input object* berupa **matriks** dan **vektor**.

6.2 *ompr* Packages di **R**

Ada satu *packages* lain di **R** yang bisa digunakan untuk menyelesaikan masalah optimisasi, yakni bernama *ompr*. *Packages ompr* dibuat oleh **Dirk Schumacher** pada 2018²⁹.

Salah satu keuntungan dari *library* ini adalah penggunaan operator *pipe* `%>%` pada perumusan algoritmanya. Sehingga bagi *user* yang biasa menggunakan prinsip *tidyverse* akan merasa sangat terbantu.

6.2.1 *ompr* Modelling

Framework untuk menuliskan model optimisasi menggunakan *ompr* adalah sebagai berikut:

```
# mulai membangun model
MIPModel() %>%

# menambah variabel
add_variable() %>%

# set objective
set_objective() %>%

# menambah constraints
add_constraint()
```

Decision Variable harus didefinisikan sejak awal. Ada berapa dan tipenya seperti apa. Kita bisa menggunakan *indexed variables* untuk menghemat notasi. Berikut adalah contohnya:

```
MIPModel() %>%

# menambah variabel integer
add_variable(x, type = "integer") %>%

# menambah variabel kontinu
add_variable(y, type = "continuous") %>%

# menambah variabel binary integer
add_variable(z, type = "binary") %>%

# menambah variabel dengan lower bound
add_variable(x, lb = 10) %>%
```

²⁹<https://www.r-orms.org/>

```
# menambah variabel dengan upper dan lower bounds
add_variable(y, lb = 5, ub = 10) %>%

# menambah 10 variabel berindeks
add_variable(p[i], i = 1:10)
```

Objective Function dan Constraints dalam *ompr* bisa dituliskan sebagai fungsi matematika biasa. Bahkan kita bisa menuliskan *summation* ke dalam algoritmanya. Berikut adalah contohnya:

Misal ada 3 variabel x_1, x_2, x_3 , dengan *objective function* $\sum_i x_i$ dengan *constraint* $\sum_i x_i \leq 7$.

```
MIPModel() %>%
  add_variable(x[i], i = 1:3) %>%
  set_objective(sum_expr(x[i], i = 1:3)) %>%
  add_constraint(sum_expr(x[i], i = 1:3) <= 7)
```

6.2.1.1 Contoh Penyelesaian *Mixed Integer Linear Programming*

$$\begin{aligned} &\text{maximize: } 7x_1 + 3x_2 + x_3 \\ &6x_1 + 4x_2 + 5x_3 \leq 60 \\ &8x_1 + x_2 + 2x_3 \leq 80 \\ \text{subject to: } &9x_1 + x_2 + 7x_3 \leq 70 \\ &x_3 \geq 0 \\ &x_1, x_2 \in \mathbb{Z}_{\geq 0} \end{aligned}$$

Mari kita tuliskan dalam *ompr framework* berikut:

```
rm(list=ls())

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

# membuat model
milp_new =
  MIPModel() %>%

  # membuat 2 variabel integer
  add_variable(x1, type = "integer", lb = 0) %>%
```

```

add_variable(x2,type = "integer",lb = 0) %>%

# membuat 1 variabel kontinu
add_variable(x3,type = "continuous",lb = 0) %>%

# set obj function
set_objective(7*x1 + 3*x2 + x3,
              "max") %>%

# menuliskan semua constraints
add_constraint(6*x1 + 4*x2 + 5*x3 <= 60) %>%
add_constraint(8*x1 + x2 + 2*x3 <= 80) %>%
add_constraint(9*x1 + x2 + 7*x3 <= 70)

milp_new

```

```

## Mixed integer linear optimization problem
## Variables:
##   Continuous: 1
##   Integer: 2
##   Binary: 0
## Model sense: maximize
## Constraints: 3

```

Mari kita *solve* modelnya:

```

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

```

```

## <SOLVER MSG> ----
## GLPK Simplex Optimizer, v4.65
## 3 rows, 3 columns, 9 non-zeros
## *      0: obj = -0.000000000e+00 inf =  0.000e+00 (3)
## *      2: obj =  6.333333333e+01 inf =  0.000e+00 (0)
## OPTIMAL LP SOLUTION FOUND
## GLPK Integer Optimizer, v4.65
## 3 rows, 3 columns, 9 non-zeros
## 2 integer variables, none of which are binary
## Integer optimization begins...
## Long-step dual simplex will be used
## +      2: mip =      not found yet <=          +inf      (1; 0)
## +      4: >>>>  6.140000000e+01 <=  6.166666667e+01  0.4% (2; 0)
## +      4: mip =  6.140000000e+01 <=      tree is empty  0.0% (0; 3)
## INTEGER OPTIMAL SOLUTION FOUND
## <!SOLVER MSG> ----

```

```
result
```

```
## Status: optimal  
## Objective value: 61.4
```

Berikut adalah hasilnya:

```
result %>% get_solution(x1)
```

```
## x1  
## 7
```

```
result %>% get_solution(x2)
```

```
## x2  
## 4
```

```
result %>% get_solution(x3)
```

```
## x3  
## 0.4
```

6.2.2 *Conclusion*

Salah satu ciri khas *ompr* adalah penulisannya yang mirip dengan notasi matematika sehingga saat kita memiliki suatu model dengan banyak variabel, kita tidak perlu menginputnya ke dalam bentuk matriks.

UPDATE MINGGU V

7 CHAPTER VII

R CODES FOR SIMPLEX METHOD

Pada bagian sebelumnya, kita telah membahas bagaimana cara melakukan metode *simplex* dengan operasi baris elementer.

Apakah proses operasi tersebut bisa diformalkan dalam bentuk algoritma atau **R codes**?

Salah satu dari sekian banyak *packages* yang memiliki *function* metode *simplex* yang siap pakai adalah `library(boot)`. Pada bagian ini, kita akan membahas *library* tersebut.

7.1 library(boot)

Fungsi `simplex()` dari `library(boot)` digunakan untuk menyelesaikan masalah optimisasi linear ax dengan *constraints* $A_1x \leq b_1$, $A_2x \leq b_2$, $A_3x \leq b_3$, dan $x \geq 0$. Secara *default*, *function* ini akan mengukur *minimize*. Namun kita bisa mengubahnya menjadi *maximize*.

7.1.1 Penggunaan

Penggunaannya adalah sebagai berikut:

```
simplex(a, A1 = NULL, b1 = NULL, A2 = NULL, b2 = NULL, A3 = NULL,
       b3 = NULL, maxi = FALSE, n.iter = n + 2 * m, eps = 1e-10)
```

Dimana:

- **a** = *vector* yang merupakan koefisien dari *objective function*.
- **A1** = merupakan matriks berisi koefisien dari *constraints* bertipe \leq .
- **b1** = *vector* pasangan dari matriks A1. Harus berisi *non-negative*.
- **A2** = merupakan matriks berisi koefisien dari *constraints* bertipe \geq .
- **b2** = *vector* pasangan dari matriks A2. Harus berisi *non-negative*. Perhatikan bahwa *constraints* ≥ 0 secara *default* sudah masuk.
- **A3** = merupakan matriks berisi koefisien dari *constraints* bertipe $=$.
- **b3** = *vector* pasangan dari matriks A3. Harus berisi *non-negative*. Perhatikan bahwa *constraints* ≥ 0 secara *default* sudah masuk.
- **maxi** = *logical*, secara *default* akan mencari *minimize*.

7.1.2 Catatan Khusus

Penulis *packages* ini memberikan suatu catatan khusus, yakni:

The method employed here is suitable only for relatively small systems.

7.1.3 Contoh

Saya akan ambil contoh masalah *linear programming* yang ada di bagian sebelumnya:

$$\begin{aligned} & \text{maximize: } 7x_1 + 3x_2 + x_3 \\ & \text{subject to: } \begin{aligned} 6x_1 + 4x_2 + 5x_3 &\leq 60 \\ 8x_1 + x_2 + 2x_3 &\leq 80 \\ 9x_1 + x_2 + 7x_3 &\leq 70 \\ x_1, x_2, x_3 &\geq 0 \end{aligned} \end{aligned}$$

```
rm(list=ls())
# panggil library
library(boot)

# set objective function
obj = c(7, 3, 1)
# membuat matriks A1
c1 = c(6, 4, 5)
c2 = c(8, 1, 2)
c3 = c(9, 1, 7)
A1 = rbind(c1,c2,c3)
# membuat rhs dari matriks A1
b1 = c(60, 80, 70)
# solving masalah optimisasi
simplex(a = obj, A1 = A1, b1 = b1,maxi = TRUE)

##
## Linear Programming Results
##
## Call : simplex(a = obj, A1 = A1, b1 = b1, maxi = TRUE)
##
## Maximization Problem with Objective Function Coefficients
## x1 x2 x3
## 7 3 1
##
##
```

```
## Optimal solution has the following values
##      x1      x2      x3
## 7.333333 4.000000 0.000000
## The optimal value of the objective function is 63.3333333333333.
```


UPDATE MINGGU VI

CHAPTER VIII

8 INTEGER DAN BINARY PROGRAMMING

Pada bagian ini, kita akan membahas lebih detail terkait *integer* dan *binary programming*.

8.1 *Integer Programming*

Integer programming adalah bentuk metode optimisasi di mana variabel yang terlibat merupakan bilangan bulat (*integer*). Jika fungsi-fungsi yang terkait merupakan *linear*, maka disebut dengan *integer linear programming*. Sebagai contoh, variabel yang merupakan bilangan bulat adalah banyak orang.

8.1.1 Contoh *Integer Programming*

Jadwal Kebutuhan Tenaga Kesehatan Suatu rumah sakit membutuhkan tenaga kesehatan setiap harinya dengan spesifikasi berikut:

Table 10: Tabel Kebutuhan Nakes Harian

hari	Min Nakes Required	Max Nakes Required
Senin	24	29
Selasa	22	27
Rabu	23	28
Kamis	11	16
Jumat	16	21
Sabtu	20	25
Minggu	12	17

Di rumah sakit tersebut berlaku kondisi sebagai berikut:

1. Setiap nakes hanya diperbolehkan bekerja selama 5 hari berturut-turut dan harus libur selama 2 hari berturut-turut.
2. Tidak ada pemberlakuan *shift* bagi nakes.

Berapa banyak nakes yang harus dipekerjakan oleh rumah sakit tersebut? Bagaimana konfigurasi penjadwalannya?

Untuk memudahkan dalam mencari solusi permasalahan di atas, kita bisa membuat tabel ilustrasi berikut:

Table 11: Konfigurasi Penjadwalan Nakes

hari	Min Nakes Required	Max Nakes Required	x1	x2	x3	x4	x5	x6	x7
Senin	24	29	x			x	x	x	x
Selasa	22	27	x	x			x	x	x
Rabu	23	28	x	x	x			x	x
Kamis	11	16	x	x	x	x			x
Jumat	16	21	x	x	x	x	x		
Sabtu	20	25		x	x	x	x	x	
Minggu	12	17			x	x	x	x	x

Kolom $x_i, i = 1, 2, 3, 4, 5, 6, 7$ menandakan kelompok nakes yang perlu dipekerjaan pada hari-hari tertentu. Setiap nilai x_i tersebut merupakan **bilangan bulat positif** $x \geq 0, x \in \mathbb{Z}$.

Dari ilustrasi di atas, kita bisa membuat model optimisasinya sebagai berikut:

Objective Function

$$\min \sum_{i=1}^7 x_i$$

Constraints

- Hari Senin: $24 \leq \sum x_i \leq 29, i \in \{1, 4, 5, 6, 7\}$.
- Hari Selasa: $22 \leq \sum x_i \leq 27, i \in \{1, 2, 5, 6, 7\}$.
- Hari Rabu: $23 \leq \sum x_i \leq 28, i \in \{1, 2, 3, 6, 7\}$.
- Hari Kamis: $11 \leq \sum x_i \leq 16, i \in \{1, 2, 3, 4, 7\}$.
- Hari Jumat: $16 \leq \sum x_i \leq 21, i \in \{1, 2, 3, 4, 5\}$.
- Hari Sabtu: $20 \leq \sum x_i \leq 25, i \in \{2, 3, 4, 5, 6\}$.
- Hari Minggu: $12 \leq \sum x_i \leq 17, i \in \{3, 4, 5, 6, 7\}$.

Kita juga perlu perhatikan bahwa $x_i \geq 0, i \in \{1, 2, 3, 4, 5, 6, 7\}$.

Kita akan selesaikan dengan `library(ompr)` di **R**. Berikut adalah skripnya:

```
rm(list=ls())

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

# membuat model
integer_prog =
```

```

MIPModel() %>%
# membuat variabel
add_variable(x[i],
             type = "integer",
             lb = 0,
             i = 1:7) %>%

# set fungsi objective
set_objective(sum_expr(x[i], i = 1:7), "min") %>%
# memasukkan constraints
# senin
add_constraint(sum_expr(x[i], i = c(1,4,5,6,7)) >= 24) %>%
add_constraint(sum_expr(x[i], i = c(1,4,5,6,7)) <= 29) %>%
# selasa
add_constraint(sum_expr(x[i], i = c(1,2,5,6,7)) >= 22) %>%
add_constraint(sum_expr(x[i], i = c(1,2,5,6,7)) <= 27) %>%
# rabu
add_constraint(sum_expr(x[i], i = c(1,2,3,6,7)) >= 23) %>%
add_constraint(sum_expr(x[i], i = c(1,2,3,6,7)) <= 28) %>%
# Kamis
add_constraint(sum_expr(x[i], i = c(1,2,3,4,7)) >= 11) %>%
add_constraint(sum_expr(x[i], i = c(1,2,3,4,7)) <= 16) %>%
# jumat
add_constraint(sum_expr(x[i], i = 1:5) >= 16) %>%
add_constraint(sum_expr(x[i], i = 1:5) <= 21) %>%
# sabtu
add_constraint(sum_expr(x[i], i = 2:6) >= 20) %>%
add_constraint(sum_expr(x[i], i = 2:6) <= 25) %>%
# minggu
add_constraint(sum_expr(x[i], i = 3:7) >= 12) %>%
add_constraint(sum_expr(x[i], i = 3:7) <= 17)

integer_prog

```

```

## Mixed integer linear optimization problem
## Variables:
##   Continuous: 0
##   Integer: 7
##   Binary: 0
## Model sense: minimize
## Constraints: 14

```

```

hasil = integer_prog %>% solve_model(with_ROI(solver = "glpk", verbose = T))

```

```

## <SOLVER MSG> ----

```

```

## GLPK Simplex Optimizer, v4.65
## 14 rows, 7 columns, 70 non-zeros
##      0: obj =  0.000000000e+00 inf =  1.280e+02 (7)
##      9: obj =  2.766666667e+01 inf =  0.000e+00 (0)
## *    10: obj =  2.766666667e+01 inf =  0.000e+00 (0)
## OPTIMAL LP SOLUTION FOUND
## GLPK Integer Optimizer, v4.65
## 14 rows, 7 columns, 70 non-zeros
## 7 integer variables, none of which are binary
## Integer optimization begins...
## Long-step dual simplex will be used
## +    10: mip =      not found yet >=                -inf          (1; 0)
## +    12: >>>>  2.800000000e+01 >=  2.800000000e+01  0.0% (2; 0)
## +    12: mip =  2.800000000e+01 >=      tree is empty  0.0% (0; 3)
## INTEGER OPTIMAL SOLUTION FOUND
## <!SOLVER MSG> ----

```

```

# solusi yang dihasilkan
hasil$solution

```

```

## x[1] x[2] x[3] x[4] x[5] x[6] x[7]
##    8    3    1    0    4   12    0

```

8.2 Binary Programming

Binary programming adalah bentuk metode optimisasi di mana variabel yang terlibat merupakan bilangan biner (0,1). Biasanya metode ini dipakai dalam masalah penjadwalan yang memerlukan prinsip *matching* antar kondisi yang ada.

UPDATE MINGGU VII

CHAPTER IX

9 MIXED INTEGER LINEAR PROGRAMMING

HOMEWORKS

UPDATE MINGGU VIII dan IX

10 MILP IN SUPPLIER SELECTION

UPDATE MINGGU X

11 R CODES INTEGER PROGRAMMING

UPDATE MINGGU XI

12 R CODES BINARY PROGRAMMING

UPDATE MINGGU XII - XIII

13 SOLVING SUPPLIER SELECTION PROBLEM