UPDATE PROGRESS

Penelitian Mandiri dalam Sains Komputasi

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

5 November 2021

METODE *SIMPLEX*

Metode *simplex* adalah salah satu metode yang paling umum digunakan dalam menyelesaikan permasalahan *linear programming*. Metode ini dikembangkan oleh seorang profesor matematika bernama George Dantzig[1] pada 1947 pasca perang dunia II. Sedangkan nama *simplex* diusulkan oleh Theodore Motzkin[2].

Metode *simplex* menggunakan prosedur aljabar[@lieberman]. Namun *underlying concept* dari metode ini adalah *geometric*.

Metode Simplex dengan Ilustrasi Geometris

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.

Contoh Masalah Optimisasi

Cari $x_1, x2$ yang $\max(Z = 3x_1 + 5x_2)$ dengan *constraints*:

$$x1 \leq 4$$
$$2x_2 \leq 12$$
$$3x_1 + 2x_2 \leq 18$$
$$\text{serta } x_1 \geq 0, x_2 \geq 0$$

Masalah di atas jika dibuat grafiknya:

[1] https://en.wikipedia.org/wiki/George_Dantzig
[2] https://en.wikipedia.org/wiki/Theodore_Motzkin

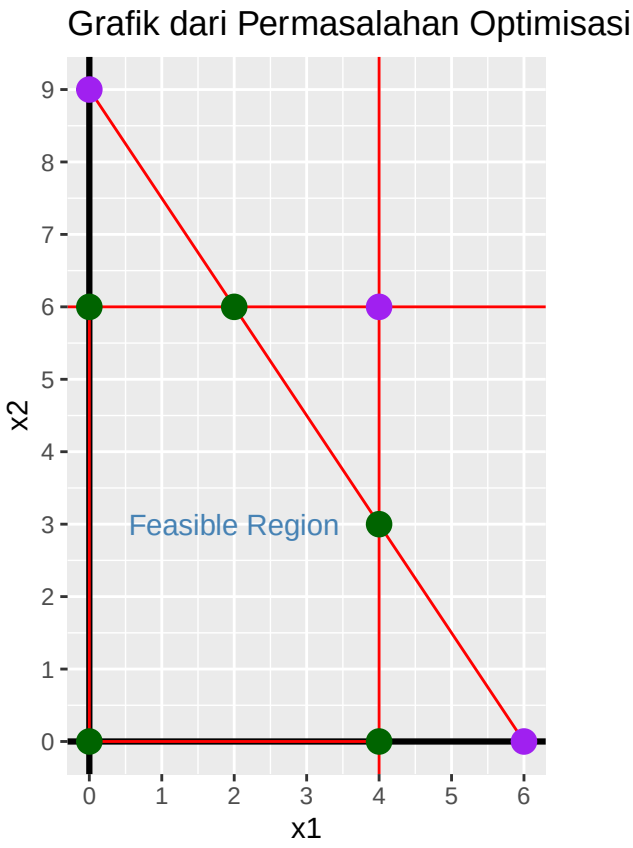

Figure 1: Grafik Permasalahan Optimisasi

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

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, berlaku:

- **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).

Properties di atas menjamin keberadaan solusi optimal pada CPF dari suatu masalah optimisasi *linear programming*.

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.

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.

Transisi Geometris ke Aljabar

Pada penjelasan sebelumnya kita bisa melihat ilustrasi geometris dari suatu masalah optimisasi di mana solusi berada di **CPF**. Namun jika kita berhadapan dengan $n > 2$ variabel, kita tidak bisa menggambarkan visualnya. Oleh karena itu kita akan menggunakan skema aljabar untuk menyelesaikannya.

Ide dasarnya adalah dengan mengubah pertaksamaan yang ada di *constraints* menjadi sebuah persamaan dengan menambahkan beberapa variabel *dummy*. Persamaan-persamaan tersebut akan kita jadikan SPL dan dicari solusinya dengan kondisi **semua kombinasi di mana $n - m$ variabel dibuat sama dengan nol** (m banyaknya persamaan dan n banyaknya variabel).

- $n - m$ variabel yang dibuat **nol** disebut dengan *non basic variables*,
- Sedangkan variabel m sisanya disebut dengan *basic variables*. Solusi dari SPL ini disebut dengan *basic solution* [@taha].

Dengan contoh masalah yang sama dengan sebelumnya, kita akan selesaikan sebagai berikut:

Masalah Optimisasi

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

$$
\begin{gathered}
x1 \leq 4 \\
2x_2 \leq 12 \\
3x_1 + 2x_2 \leq 18 \\
\text{serta } x_1 \geq 0, x_2 \geq 0
\end{gathered}
$$

Pada *constraints* yang mengandung pertaksamaan $\leq$, *right hand side* menunjukkan batas dari *resources* sementara *left hand side* menunjukkan *usage* dari *resources*. Selisih antara *rhs* dan *lhs* menunjukkan **sisa** *resources* yang tidak terpakai. Kita perlu mengubah pertaksamaan yang ada menjadi bentuk persamaan dengan cara menambahkan u, v, w sebagai ***non negative slack variables*** [@taha].

Oleh karena itu kita tuliskan *constraints* menjadi sebagai berikut:

$$
\begin{gathered}
x1 + u = 4 \\
2x_2 + v = 12 \\
3x_1 + 2x_2 + w = 18 \\
\text{dengan } x_1 \geq 0, x_2 \geq 0, u \geq 0, v \geq 0, w \geq 0
\end{gathered}
$$

Perhatikan bahwa $m = 3$ dan $n = 5$ sehingga $n - m = 2$. Maka kita akan buat semua kombinasi *non basic variables* (berisi 2 variabel).

```
##  [1] "(x1,x2)" "(x1,u)"  "(x1,v)"  "(x1,w)"  "(x2,u)"  "(x2,v)"  "(x2,w)"
##  [8] "(u,v)"   "(u,w)"   "(v,w)"
```

Kemudian menyelesaikan *SPL* yang ada pada kondisi *non basic variables* tersebut **nol**. Dari masing-masing solusi yang ada, kita akan lihat apakah *feasible* atau tidak? Serta dievaluasi nilai z-nya.

Berikut adalah algoritma dan tabel hasilnya:

```r
# set SPL dari constraints
A = data.frame(x1 = c(1,0,3),
               x2 = c(0,2,2),
               u = c(1,0,0),
               v = c(0,1,0),
               w = c(0,0,1))
# rhs
c = c(4,12,18)
# obj function
obj_f = function(data){
  # filter var
  x1 = sol_val %>% filter(var %in% c("x1"))
  x2 = sol_val %>% filter(var %in% c("x2"))
  # ambil val
  if(nrow(x1) != 1){x1 = 0}else{x1 = x1$value}
  if(nrow(x2) != 1){x2 = 0}else{x2 = x2$value}
  return(3*x1 + 5*x2)
  }
# set template hasil
hasil = data.frame(non_basic_var = paste0("(",non_basic$v1,",",non_basic$v2,")"),
                   basic_var = NA,
                   solusi = NA,
                   z = NA)
# iterasi
for (i in 1:nrow(non_basic)) {
  # siap print basic var
  basic_var = var_[!grepl(non_basic$v1[i],var_)]
  basic_var = basic_var[!grepl(non_basic$v2[i],basic_var)]
  basic_var = paste(basic_var,collapse = ",")
  hasil$basic_var[i] = paste0("(",basic_var,")")

  # hitung solusi SPL
  B = A %>% select(-contains(non_basic$v1[i])) %>% select(-contains(non_basic$v2[i])) %>% as.matrix()
  if(det(B) == 0){sol_print = NA}
  else{
    sol = solve(B) %*% c
    sol_print = paste(row.names(sol),sol,sep = " = ")
    sol_print = paste(sol_print,collapse = "; ")
  }
  hasil$solusi[i] = sol_print
  # evaluasi obj function
  if(det(B) == 0){z_hit = NA}
  else{
    sol_val = data.frame(var = row.names(sol),value = sol)
    z_hit = obj_f(obj_f)
  }
  hasil$z[i] = z_hit
  }
```

Table 3: Hasil Perhitungan Simplex dengan Metode Aljabar

Non Basic Var	Basic Var	solusi	z	feasible
(x1,x2)	(u,v,w)	u = 4; v = 12; w = 18	0	Yes

Non Basic Var	Basic Var	solusi	z	feasible
(x1,u)	(x2,v,w)	NA	NA	No
(x1,v)	(x2,u,w)	x2 = 6; u = 4; w = 6	30	Yes
(x1,w)	(x2,u,v)	x2 = 9; u = 4; v = -6	45	No
(x2,u)	(x1,v,w)	x1 = 4; v = 12; w = 6	12	Yes
(x2,v)	(x1,u,w)	NA	NA	No
(x2,w)	(x1,u,v)	x1 = 6; u = -2; v = 12	18	No
(u,v)	(x1,x2,w)	x1 = 4; x2 = 6; w = -6	42	No
(u,w)	(x1,x2,v)	x1 = 4; x2 = 3; v = 6	27	Yes
(v,w)	(x1,x2,u)	x1 = 2; x2 = 6; u = 2	36	Yes

Terlihat di atas bahwa $\max z = 36$ terletak pada saat $x_1 = 2, x_2 = 6$. Sama persis dengan perhitungan dengan pendekatan geometris.

Metode Simplex dengan *Tableau*

Pendekatan aljabar di atas bisa kita buat menjadi suatu operasi baris elementer di matriks. Berikut adalah contohnya:

Operasi Baris Elementer Matriks *Simplex*

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

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

Pada *constraints* yang mengandung pertaksamaan $\leq$, *right hand side* menunjukkan batas dari *resources* sementara *left hand side* menunjukkan *usage* dari *resources*. Selisih antara *rhs* dan *lhs* menunjukkan **sisa** *resources* yang tidak terpakai.

Kita perlu mengubah pertaksamaan yang ada menjadi bentuk persamaan dengan cara menambahkan u, w sebagai ***non negative slack variables*** [@taha]. Fungsi objectif P juga harus diubah (dipindah sisi namun P tetap positif).

$$3x + 5y + u = 78, \text{ dengan } u \geq 0$$
$$4x + y + w = 36, \text{ dengan } w \geq 0$$
$$-5x - 4y + P = 0$$

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

Table 4: 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 5: 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 6: 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 7: 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 8: 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 9: 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 klom 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 10: 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 nilau u dan w? Karena tidak ada nlai `1` ditemukan pada kolom variabel tersebut, kita bisa simpulkan bahwa $u = 0, w = 0$.