TUGAS MANDIRI

Perancangan dan Analisis Algoritma

The Simplex Method

202423430076

DOSEN PENGAMPU :

Randi Proska Sandra, M.Sc

OLEH :

Fajri Alhidra Fahalevi

23343062

Informatika

PROGRAM STUDI INFORMATIKA

DEPARTEMEN ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2025

A. Ringkasan Singkat

Kita telah menjumpai pemrograman linier, masalah umum mengoptimalkan fungsi linier beberapa variabel yang tunduk pada serangkaian kendala linier: Kami menyebutkan di sana bahwa banyak masalah praktis penting dapat dimodelkan sebagai contoh pemrograman linier. Dua peneliti, L.V. Kantorovich dari bekas Uni Soviet dan T. C. Koopmans, warga negara Belanda-Amerika, bahkan dianugerahi Penghargaan Nobel pada tahun 1975 atas kontribusi mereka pada teori pemrograman linier dan penerapannya pada ekonomi. Rupanya karena tidak ada Penghargaan Nobel dalam matematika, Akademi Ilmu Pengetahuan Kerajaan Swedia gagal menghormati matematikawan AS G. B. Dantzig, yang secara universal diakui sebagai bapak pemrograman linier dalam bentuk modernnya dan penemu metode simpleks, algoritma klasik untuk memecahkan masalah tersebut.(Levitin, 2012)

Untungnya, ternyata ada algoritma yang biasanya hanya memeriksa sebagian kecil titik ekstrem dari wilayah yang layak sebelum mencapai titik optimal. Algoritma terkenal ini disebut metode simpleks. Ide algoritma ini dapat dijelaskan dalam istilah geometris sebagai berikut. Mulailah dengan mengidentifikasi titik ekstrem dari wilayah yang layak. Kemudian periksa apakah titik tersebut dapat memperoleh nilai fungsi objektif yang ditingkatkan dengan menuju titik ekstrem yang berdekatan. Jika tidak demikian, titik saat ini optimal—berhenti; jika demikian, lanjutkan ke titik ekstrem yang berdekatan dengan nilai fungsi objektif yang ditingkatkan. Setelah sejumlah langkah yang terbatas, algoritma akan mencapai titik ekstrem di mana solusi optimal terjadi atau menentukan bahwa tidak ada solusi optimal.(Levitin, 2012)

Banyak metode yang dapat dipakai untuk optimalisasi produksi. Salah satu metode yang dapat digunakan untuk menyelesaikan solusi optimal dalam permasalahan ini adalah metode Simpleks. Metode Simpleks adalah salah satu prosedur yang paling luas penggunaannya untuk pemecahan persoalan pemrograman linier, bahkan digunakan untuk penyelesaian dari program program komputer.(Chandra, 2015)

Pada dasarnya, dikembangkan untuk metode-metode yang memecahkan model program linier ditujukan untuk mencari solusi dari beberapa alternatif solusi yang dibentuk untuk persamaan-persamaan pembatas cara sehingga diperoleh nilai fungsi tujuan yang optimal. Ada dua yang bisa digunakan untuk menyelesaikan persoalan program linier ini yaitu dengan cara grafis dan metode simpleks. Metode simpleks merupakan teknik yang paling berhasil dikembangkan untuk memecahkan persoalan program linier yang mempunyai jumlah variabel keputusan dan pembatas yang besar.(Chandra, 2015)

Metode grafik cocok untuk menyelesaikan program linier yang jumlah variabelnya sama dengan dua walaupun variabelnya sama dengan tiga bisa juga diselesaikan tetapi grafiknya akan sangat sulit untuk dibaca dan dipahami, apalagi lebih dari tiga variable. Untuk itu, maka dipilih metode simpleks dalam penyelesaian masalah program linier yang jumlah peubahnya lebih dari dua.(Chandra, 2015)

Beberapa kelebihan dari metode simpleks adalah jumlah variabel bisa lebih dari satu, hasil akhir lebih akurat dan terlihat sisa dari keterbatasan yang ada.(Chandra, 2015)

1. Bentuk Baku dan Bentuk Tabel Metode Simpleks

   Sebelum melakukan perhitungan iteratif untuk menentukan solusi optimal, pertama sekali bentuk umum pemrograman linear diubah ke dalam bentuk baku terlebih dahulu. Bentuk baku dalam metode simpleks tidak hanya mengubah persamaan kendala ke dalam bentuk sama dengan, tetapi juga setiap fungsi kendala harus diwakili oleh satu variabel basis awal. Variabel basis awal menunjukkan status sumber daya pada kondisi sebelum ada aktivitas yang dilakukan. Dengan kata lain, variabel keputusan semuanya masih bernilai nol. Dengan demikian, meskipun fungsi kendala pada bentuk umum pemrograman linier sudah dalam bentuk persamaan, fungsi kendala tersebut masih harus tetap berubah.(Sriwidadi & Agustina, n.d.)

   Ada beberapa hal yang harus diperhatikan dalam membuat bentuk baku, yaitu: (1) Fungsi kendala dengan pertidaksamaan ≤ dalam bentuk umum, diubah menjadi persamaan

(=) dengan menambahkan satu variabel slack; (2) Fungsi kendala dengan pertidaksamaan ≥ dalam bentuk umum, diubah menjadi persamaan (=) dengan mengurangkan satu variabel surplus; (3) Fungsi kendala dengan persamaan dalam bentuk umum, ditambahkan satu variabel artifisial (variabel buatan).(Sriwidadi & Agustina, n.d.)

Dalam perhitungan iterative, digunakan tabel. Bentuk baku yang sudah diperoleh, harus dibuat ke dalam bentuk tabel. Semua variabel yang bukan variabel basis mempunyai solusi (nilai kanan) sama dengan nol dan koefisien variabel basis pada baris tujuan harus sama dengan 0. Oleh karena itu, pembentukan tabel awal harus dibedakan berdasarkan variabel basis awal.(Sriwidadi & Agustina, n.d.)

2. Tahap-tahap Metode Simpleks

Berikut adalah tahap dalam menyelesaikan program linear dengan metode simpleks. Pertama, memeriksa tabel layak atau tidak. Kelayakan tabel simpleks dilihat dari solusi (nilai kanan). Jika solusi ada yang bernilai negatif, tabel tidak layak. Tabel yang tidak layak tidak dapat diteruskan untuk dioptimalkan. Kedua, menentukan kolom pivot. Penentuan kolom pivot dilihat dari koefisien fungsi tujuan (nilai di sebelah kanan baris z) dan tergantung dari bentuk tujuan. Jika tujuan maksimisasi, kolom pivot adalah kolom dengan koefisien paling negatif. Jika tujuan minimisasi, kolom pivot adalah kolom dengan koefisien positif terbesar. Jika kolom pivot ditandai dan ditarik ke atas, variabel keluar akan diperoleh. Jika nilai paling negatif (untuk tujuan maksimisasi) atau positif terbesar (untuk tujuan minimisasi) lebih dari satu, pilih salah satu secara sembarang. Ketiga, menentukan baris pivot. Baris pivot ditentukan setelah membagi nilai solusi dengan nilai kolom pivot yang bersesuaian (nilai yang terletak dalam satu baris). Dalam hal ini, nilai negatif dan 0 pada kolom pivot tidak diperhatikan, artinya tidak ikut menjadi pembagi. Baris pivot adalah baris dengan rasio pembagian terkecil. Jika baris pivot ditandai dan ditarik ke kiri, variabel keluar akan diperoleh. Jika rasio pembagian terkecil lebih dari satu, pilih salah sau secara sembarang. Keempat, menentukan elemen pivot. Elemen pivot merupakan nilai yang terletak pada perpotongan kolom dan baris pivot. Kelima, membentuk tabel simpleks baru. Tabel simpleks baru dibentuk dengan pertama sekali menghitung nilai baris pivot baru. Baris pivot baru adalah baris pivot lama dibagi dengan elemen pivot. Baris baru lainnya merupakan pengurangan nilai kolom pivot baris yang bersangkutan dikali baris pivot baru dalam satu kolom terhadap baris lamanya yang terletak pada kolom tersebut. Keenam, memeriksa jika tabel sudah optimal. Keoptimalan tabel dilihat dari koefisien fungsi tujuan (nilai pada baris z) dan tergantung dari bentuk tujuan. Untuk tujuan maksimisasi, tabel sudah optimal jika semua nilai pada baris z sudah positif atau 0. Pada tujuan minimisasi, tabel sudah optimal jika semua nilai pada baris z sudah negatif atau 0. Jika belum, kembali ke langkah kedua; jika sudah optimal, baca solusi optimal.(Sriwidadi & Agustina, n.d.)

3. Metode Revised Simplex (Metode Simpleks yang Diperbaiki)

Metode ini didesain untuk mencapai hal yang tepat seperti pada metode simpleks yang asli. Metode ini menghitung dan menyimpan hanya informasi yang diperlukan sekarang dan data yang penting disimpan dalam bentuk lebih padat. Metode revised simplex secara eksplisit memakai manipulasi matriks, maka masalah harus dinyatakan dalam notasi matriks.(Sriwidadi & Agustina, n.d.)

B. PSEUDOCODE

```
def inisialisasi_matriks_simpleks(matriks_awal):
    return matriks_awal.copy()

def cari_kolom_pivot(matriks):
    return matriks[0].index(min(matriks[0][:-1]))

def cari_baris_pivot(matriks, kolom_pivot):
    rasio = []
    for i in range(1, len(matriks)):
        if matriks[i][kolom_pivot] > 0:
            rasio.append(matriks[i][-1] / matriks[i][kolom_pivot])
        else:
            rasio.append(float('inf'))
    return rasio.index(min(rasio)) + 1

def lakukan_operasi_baris(matriks, baris_pivot, kolom_pivot):
    pivot = matriks[baris_pivot][kolom_pivot]
    matriks[baris_pivot] = [x / pivot for x in matriks[baris_pivot]]

    for i in range(len(matriks)):
        if i != baris_pivot:
            faktor = matriks[i][kolom_pivot]
            matriks[i]    =    [matriks[i][j]    -    faktor    *
matriks[baris_pivot][j] for j in range(len(matriks[i]))]

def metode_simpleks(matriks_awal):
    matriks = inisialisasi_matriks_simpleks(matriks_awal)

    while any(x < 0 for x in matriks[0][:-1]):
        kolom_pivot = cari_kolom_pivot(matriks)
        baris_pivot = cari_baris_pivot(matriks, kolom_pivot)
        lakukan_operasi_baris(matriks, baris_pivot, kolom_pivot)

    return matriks

# Contoh pemanggilan metode simpleks
data_matriks = [
    [1, -3, -5, 0, 0, 0, 0],
    [0, 1,  1, 1, 0, 0, 4],
    [0, 2,  0.5, 0, 1, 0, 6],
    [0, 0,  1, 0, 0, 1, 3]
]

hasil = metode_simpleks(data_matriks)
for baris in hasil:
    print(baris)
```

C. Source Code

```
def inisialisasi_matriks_simpleks(matriks_awal):
    return [baris[:] for baris in matriks_awal]

def cari_kolom_pivot(matriks):
    return matriks[0].index(min(matriks[0][:-1]))

def cari_baris_pivot(matriks, kolom_pivot):
    rasio = []
    for i in range(1, len(matriks)):
        if matriks[i][kolom_pivot] > 0:
            rasio.append(matriks[i][-1] / matriks[i][kolom_pivot])
        else:
            rasio.append(float('inf'))
    return rasio.index(min(rasio)) + 1

def lakukan_operasi_baris(matriks, baris_pivot, kolom_pivot):
    pivot = matriks[baris_pivot][kolom_pivot]
    matriks[baris_pivot] = [x / pivot for x in matriks[baris_pivot]]

    for i in range(len(matriks)):
        if i != baris_pivot:
            faktor = matriks[i][kolom_pivot]
            matriks[i] = [matriks[i][j] - faktor *
matriks[baris_pivot][j] for j in range(len(matriks[i]))]

def metode_simpleks(matriks_awal):
    matriks = inisialisasi_matriks_simpleks(matriks_awal)

    while any(x < 0 for x in matriks[0][:-1]):
        kolom_pivot = cari_kolom_pivot(matriks)
        baris_pivot = cari_baris_pivot(matriks, kolom_pivot)
        lakukan_operasi_baris(matriks, baris_pivot, kolom_pivot)

    return matriks

# Contoh pemanggilan metode simpleks
data_matriks = [
    [1, -3, -5, 0, 0, 0, 0],
    [0, 1,  1, 1, 0, 0, 4],
    [0, 2,  0.5, 0, 1, 0, 6],
    [0, 0,  1, 0, 0, 1, 3]
]

hasil = metode_simpleks(data_matriks)
print("Matriks Akhir setelah Simpleks:")
for baris in hasil:
    print([round(x, 2) for x in baris])
```

```python
def inisialisasi_matriks_simpleks(matriks_awal):
    return [baris[:] for baris in matriks_awal]

def cari_kolom_pivot(matriks):
    return matriks[0].index(min(matriks[0][:-1]))

def cari_baris_pivot(matriks, kolom_pivot):
    rasio = []
    for i in range(1, len(matriks)):
        if matriks[i][kolom_pivot] > 0:
            rasio.append(matriks[i][-1] / matriks[i][kolom_pivot])
        else:
            rasio.append(float('inf'))
    return rasio.index(min(rasio)) + 1

def lakukan_operasi_baris(matriks, baris_pivot, kolom_pivot):
    pivot = matriks[baris_pivot][kolom_pivot]
    matriks[baris_pivot] = [x / pivot for x in matriks[baris_pivot]]

    for i in range(len(matriks)):
        if i != baris_pivot:
            faktor = matriks[i][kolom_pivot]
            matriks[i] = [matriks[i][j] - faktor * matriks[baris_pivot][j] for j in range(len(matriks[i]))]

def metode_simpleks(matriks_awal):
    matriks = inisialisasi_matriks_simpleks(matriks_awal)

    while any(x < 0 for x in matriks[0][:-1]):
        kolom_pivot = cari_kolom_pivot(matriks)
        baris_pivot = cari_baris_pivot(matriks, kolom_pivot)
        lakukan_operasi_baris(matriks, baris_pivot, kolom_pivot)
```

(parameter) matriks: Any

```python
    return matriks

# Contoh pemanggilan metode simpleks
data_matriks = [
    [1, -3, -5, 0, 0, 0, 0],
    [0, 1,  1, 1, 0, 0, 4],
    [0, 2,  0.5, 0, 1, 0, 6],
    [0, 0,  1, 0, 0, 1, 3]
]

hasil = metode_simpleks(data_matriks)
print("Matriks Akhir setelah Simpleks:")
for baris in hasil:
    print([round(x, 2) for x in baris])
```

```
PS D:\SEMESTER 4\Pemograman Saintifik\Pertemuan 4> & (
matriks_simpleks(matrik.py"
Matriks Akhir setelah Simpleks:
[1.0, 0.0, 0.0, 3.0, 0.0, 2.0, 18.0]
[0.0, 1.0, 0.0, 1.0, 0.0, -1.0, 1.0]
[0.0, 0.0, 0.0, -2.0, 1.0, 1.5, 2.5]
[0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 3.0]
PS D:\SEMESTER 4\Pemograman Saintifik\Pertemuan 4>
```

D. Analisis Kebutuhan Waktu

1. Analisis Menyeluruh (Berdasarkan Operator Penugasan dan Aritmatika)
   Kita akan menghitung jumlah operasi berdasarkan setiap fungsi dalam program:
   a. inisialisasi_matriks_simpleks(matriks_awal)
      - Operasi penugasan (=): 1 kali untuk setiap baris → O(m)O(m)O(m), dengan mmm sebagai jumlah baris.
      - Total kompleksitas: O(m)O(m)O(m)

   b. cari_kolom_pivot(matriks)
      - Operasi minimum (min): O(n)O(n)O(n) dengan nnn sebagai jumlah kolom.
      - Operasi pencarian indeks (index): O(n)O(n)O(n)
      - Total kompleksitas: O(n)O(n)O(n)

   c. cari_baris_pivot(matriks, kolom_pivot)
      - Iterasi melalui m−1m-1m−1 baris: O(m)O(m)O(m)
      - Operasi pembagian (/): O(m)O(m)O(m)
      - Operasi minimum (min): O(m)O(m)O(m)
      - Total kompleksitas: O(m)O(m)O(m)

   d. lakukan_operasi_baris(matriks, baris_pivot, kolom_pivot)
      - Normalisasi pivot: O(n)O(n)O(n)
      - Iterasi melalui semua baris untuk eliminasi: O(m·n)O(m \cdot n)O(m·n)
      - Total kompleksitas: O(m·n)O(m \cdot n)O(m·n)

   e. metode_simpleks(matriks_awal)
      - Iterasi selama ada elemen negatif di baris pertama: Worst case O(n)O(n)O(n) iterasi.
      - Setiap iterasi membutuhkan:
        - O(n) untuk mencari kolom pivot.
        - O(m) untuk mencari baris pivot.
        - O(m·n) untuk eliminasi.
      - Total kompleksitas: O(n·(n+m+m·n))=O(n2+mn+mn2)O(n \cdot (n + m + m \cdot n)) = O(n^2 + mn + m n^2)O(n·(n+m+m·n))=O(n2+mn+mn2)

2. Analisis Berdasarkan Operasi Abstrak
   Operasi utama dalam metode simpleks adalah:
   - Pencarian pivot (kolom dan baris): O(m+n)O(m + n)O(m+n) per iterasi.
   - Operasi eliminasi Gaussian: O(m·n)O(m \cdot n)O(m·n) per iterasi.
   - Jumlah iterasi maksimal: O(n)O(n)O(n), berdasarkan jumlah variabel dalam model LP.

   Sehingga total operasi abstrak dalam kasus terburuk adalah O(n·(m+n+m·n))O(n \cdot (m + n + m \cdot n))O(n·(m+n+m·n)) atau O(mn2)O(m n^2)O(mn2).

3. Analisis Best-Case, Worst-Case, dan Average-Case
    a. Best-Case (Kasus Terbaik)
        - Jika solusi optimal ditemukan dalam satu iterasi:
        - Kompleksitasnya hanya O(mn)O(mn)O(mn).

    b. Worst-Case (Kasus Terburuk)
        - Jika algoritma harus melakukan O(n)O(n)O(n) iterasi dengan setiap iterasi memiliki kompleksitas O(mn)O(mn)O(mn).
        - Total O(mn2)O(mn^2)O(mn2).

    c. Average-Case (Kasus Rata-rata)
        - Biasanya sekitar O(mn2)O(m n^2)O(mn2) berdasarkan eksperimen dan penelitian tentang metode simpleks.

E. REFERENSI

Chandra, T. (2015). Penerapan Algoritma Simpleks dalam Aplikasi Penyelesaian Masalah Program Linier. In *Jurnal TIMES: Vol. IV*.

Levitin, Anany. (2012). *Introduction to the design & analysis of algorithms*. Pearson.

Sriwidadi, T., & Agustina, E. (n.d.). *Jam Kerja Mesin = 2,5 A + 3 B + 2 C + 2,5 D ≤ 86.400 detik, Jam Tenaga Kerja = 5 A + 5 B + 5 C + 5 D ≤ 115.200 detik, Permintaan GRX 25 = A ≤ 2400 pcs, Permintaan GTW 25 = B ≤ 7200 pcs*.

F. Lampiran Link GitHub

https://github.com/f4hlevi/Perancangan-dan-Analisis-Algoritma