TUGAS MANDIRI

PERANCANGAN & ANALISIS ALGORITMA

“Strassen’s Matrix Multiplication”

202323430048

DOSEN PENGAMPU:

Randi Proska Sandra, S.Pd, M.Sc

OLEH:

M Rizal Saputra

22343055

Informatika (NK)

PRODI SARJANA INFORMATIKA

DEPARTEMEN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2024

A. PENJELASAN SINGKAT

Metode Simpleks merupakan pendekatan penyelesaian model program linier dengan tangan menggunakan variabel slack, tableaus, dan variabel pivot sebagai sarana untuk mencari solusi optimal suatu masalah optimasi. Program linier adalah metode untuk mencapai hasil terbaik dengan persamaan maksimum atau minimum dengan batasan linier. Kebanyakan program linier dapat diselesaikan menggunakan solver online seperti MatLab, namun metode Simplex adalah teknik untuk menyelesaikan program linier dengan tangan. Untuk menyelesaikan model program linier dengan menggunakan metode Simplex diperlukan langkah-langkah sebagai berikut:

- Bentuk standar
- Memperkenalkan variabel slack
- Membuat tablo
- Variabel pivot
- Membuat tablo baru
- Memeriksa optimalitasnya
- Identifikasi nilai optimal

Metode Simplex menjadi langkah-langkah di atas dan mengikuti contoh model pemrograman linier yang ditunjukkan di bawah ini di seluruh dokumen untuk menemukan solusi optimal.

B. PSEUDOCODE

Input:

- c: Array koefisien fungsi tujuan yang akan diminimalkan
- A: Matriks kendala
- b: Vektor sisi kanan

Step 1: Inisialisasi

- Bangun tabel awal:

- Biarkan m menjadi jumlah kendala (jumlah baris A)
- Biarkan n menjadi jumlah variabel (jumlah kolom A)
- Buat matriks tabel dengan ukuran (m+1) x (n+m+1) diinisialisasi dengan nol
- Atur baris pertama menjadi -c, dengan menambahkan nol untuk setiap variabel slack
- Atur m baris berikutnya menjadi matriks teraugmentasi [A | b]
- Identifikasi variabel dasar dan atur kolom yang sesuai menjadi matriks identitas

Step 2: Iterasi sampai optimal

- Selama ada nilai negatif di baris tujuan:

- Temukan variabel masuk: pilih koefisien paling negatif di baris tujuan

- Untuk setiap baris dengan nilai positif di kolom variabel masuk:

- Hitung rasio nilai sisi kanan terhadap koefisien variabel masuk
- Pilih baris dengan rasio terkecil sebagai variabel keluar
- Lakukan operasi pivot untuk membuat koefisien variabel masuk di baris keluar sama dengan 1, dan semua koefisien lain di kolom masuk sama dengan 0

Step 3: Ekstrak solusi optimal

- Jika baris tujuan berisi semua nilai non-negatif:

- Ekstrak solusi optimal dari tabel
- Kembalikan nilai variabel dasar sebagai solusi optimal

- Lainnya:

- Masalah tidak dibatasi

Output:

- Vektor solusi optimal x

C. SOURCECODE

```python
import numpy as np

def simplex(c, A, b):
    m, n = A.shape

    # Step 1: Inisialisasi
    tableau = np.zeros((m + 1, n + m + 1))
    tableau[0, 1:n + 1] = -c
    tableau[1:, 0:n] = A
    tableau[np.arange(1, m + 1), np.arange(n + 1, n + m + 1)] = 1
    tableau[1:, -1] = b

    # Step 2: Iterasi hingga optimal
    while np.any(tableau[0, 1:] < 0):
        entering_var = np.argmin(tableau[0, 1:])

        ratios = tableau[1:, -1] / tableau[1:, entering_var + 1]
        leaving_var = np.argmin(ratios) + 1

        pivot = tableau[leaving_var, entering_var + 1]
        tableau[leaving_var, 1:] /= pivot
        for i in range(m + 1):
            if i != leaving_var:
                tableau[i, 1:] -= tableau[i, entering_var + 1] * tableau[leaving_var, 1:]

    # Step 3: Ekstrak solusi optimal
    optimal_solution = np.zeros(n)
    for i in range(m):
        row = np.where(tableau[i + 1, n + 1:] == 1)[0]
        if len(row) == 1:
            optimal_solution[row] = tableau[i + 1, 0]

    return optimal_solution
```

```
# Contoh penggunaan
c = np.array([2, 3, 0, 0])
A = np.array([[1, 4, 1, 0],
              [2, 3, 0, 1]])
b = np.array([12, 18])

solusi_optimal = simplex(c, A, b)
print("Solusi optimal:", solusi_optimal)
```

D. HASIL PROGRAM :

```
  ratios = tableau[1:, -1] / tableau[1:, entering_var + 1]
Solusi optimal: [1. 0. 0. 0.]
```

E. ANALISIS KEBUTUHAN WAKTU

Untuk menganalisis kebutuhan waktu algoritma Simplex, kita dapat menggunakan tiga pendekatan yang berbeda:

1) Analisis Detail Operasi/Instruksi:

Kita dapat melihat setiap operasi/instruksi yang dieksekusi dalam algoritma dan memperkirakan jumlah operasi/instruksi yang dihasilkan. Ini akan memberikan perkiraan yang cukup akurat tentang kompleksitas waktu algoritma.

Misalnya, kita dapat menghitung jumlah operasi/instruksi untuk setiap iterasi loop, termasuk operasi penugasan, pembandingan, dan operasi aritmatika. Kemudian, kita dapat mengalikan jumlah operasi/instruksi dengan jumlah iterasi yang diharapkan.

2) Analisis Berdasarkan Jumlah Operasi Abstrak:

Dalam pendekatan ini, kita tidak memperhitungkan operasi/instruksi individu, tetapi fokus pada operasi abstrak yang dijalankan dalam algoritma. Misalnya, dalam metode Simplex, kita dapat memperhitungkan jumlah operasi perbandingan antara elemen-elemen matriks dan operasi aritmatika matriks.

3) Analisis Kasus Terbaik, Terburuk, dan Rata-rata:

Dalam pendekatan ini, kita menganalisis kinerja algoritma berdasarkan kasus terbaik (misalnya, ketika masalah sudah dalam bentuk optimal), kasus terburuk (misalnya, ketika algoritma membutuhkan banyak iterasi untuk mencapai solusi optimal), dan kasus rata-rata (mungkin menggunakan analisis probabilitas jika memungkinkan).

Contoh Analisis:

- Analisis Detail Operasi/Instruksi: Kita dapat menghitung jumlah operasi penugasan, pembandingan, dan aritmatika dalam setiap iterasi loop, kemudian mengalikannya dengan jumlah iterasi yang diperlukan.

- Analisis Berdasarkan Jumlah Operasi Abstrak: Dalam metode Simplex, kita dapat memperkirakan kompleksitas waktu berdasarkan jumlah operasi perbandingan dan aritmatika yang diperlukan dalam setiap iterasi loop.
- Analisis Kasus Terbaik, Terburuk, dan Rata-rata: Kita dapat menganalisis kinerja algoritma dalam kasus terbaik (misalnya, ketika masalah sudah optimal dari awal), kasus terburuk (misalnya, ketika algoritma memerlukan banyak iterasi untuk menemukan solusi optimal), dan kasus rata-rata (mungkin menggunakan distribusi probabilitas untuk kasus-kasus yang berbeda).

F. REFERENSI

DANTZIG, George B. Origins of the simplex method. In: A history of scientific computing. 1990. p. 141-151.

G. LINK GITHUB

https://github.com/rizals27/-Strassen-s-Matrix-Multiplication-/upload/main