TUGAS BESAR STRATEGI ALGORITMA

Anggota:

Ralphi Adam Subekti (1301204190)

UNIVERSITAS TELKOM

TAHUN AJARAN 2023/2024

I. Abstrak

Laporan ini membahas penyelesaian Assignment Problem menggunakan dua strategi algoritma berbeda yaitu algoritma greedy dan algoritma brute force. Assignment Problem adalah masalah optimasi yang mengharuskan kita menetapkan tugas kepada agen dengan biaya total minimum. Kami mengimplementasikan kedua algoritma tersebut dan membandingkan kinerjanya dalam hal efisiensi dan optimalitas solusi. Algoritma greedy memberikan solusi dengan cepat namun tidak selalu optimal, sedangkan algoritma brute force memberikan solusi optimal tetapi dengan kompleksitas waktu yang jauh lebih tinggi.

II. Pendahuluan

Assignment Problem adalah masalah klasik dalam bidang optimasi dan alokasi sumber daya. Dalam masalah ini, kita memiliki sejumlah tugas yang harus dialokasikan ke sejumlah agen dengan tujuan meminimalkan biaya total. Masalah ini sering muncul dalam berbagai aplikasi nyata, seperti penugasan karyawan ke proyek, alokasi pekerjaan ke mesin, dan sebagainya. Tujuan dari laporan ini adalah untuk mempelajari dan membandingkan dua pendekatan algoritmik untuk menyelesaikan Assignment Problem: algoritma greedy dan algoritma brute force.

III. Dasar Teori

- Penjelasan Umum Kasus

Assignment Problem dapat diformulasikan sebagai berikut: kita memiliki n tugas dan n agen, dan sebuah matriks biaya n x n yang menunjukkan biaya untuk menetapkan setiap tugas ke setiap agen. Tujuan kita adalah menemukan penugasan tugas ke agen sedemikian rupa sehingga biaya total diminimalkan.

- Strategi Algoritma Terpilih

Metode Greedy:

Dalam Bahasa Inggris, "greedy" berarti rakus/tamak. Seperti Namanya, metode Greedy cenderung mengambil keputusan yang 'egois' di setiap langkahnya. Yang dimaksud egois di sini adalah sikapnya yang mengambil keputusan yang terlihat baik, tanpa mempertimbangkan kondisi secara umum ataupun efeknya pada hasil akhir. Prinsip utama dari Greedy adalah "ambil apa yang bisa kamu ambil sekarang".

Dalam metode Greedy, kita mengambil keputusan pada setiap langkahnya, dan tidak pernah kembali lagi untuk memperbaiki keputusan yang telah dibuat. Keputusan yang diambil, dibuat berdasarkan apa yang tampaknya baik pada titik tersebut. Ini berarti, metode Greedy memilih optimum lokal di setiap langkahnya. Dengan selalu memilih optimum lokal, kita berharap untuk dapat mencapai optimum global.

Metode Brute Force:

Secara umum, algoritma Brute Force dapat dikatakan tidak "pintar" dan tidak efisien. Yang dimaksud dengan "pintar" di sini adalah kepintarannya dalam hal perancangan algoritma. Kata "force" pada Brute Force sendiri merepresentasikan kekuatan yang kita gunakan untuk menyelesaikan suatu permasalan, bukan kekuatan dari pikiran kita dalam mencari cara yang paling baik dalam menyelesaikan permasalahan tersebut. Karena kelempengannya, algoritma Brute Force juga dikenal sebagai "naïve algorithm", atau algoritma naif.

Algoritma Brute Force biasanya digunakan untuk menyelesaikan permasalahan berukuran kecil, karena algoritma Brute Force biasanya sederhana dan mudah diimplementasikan. Algoritma Brute Force juga sering digunakan sebagai pembanding untuk melihat keefisienan algoritma yang menggunakan strategi lainnya. Ini dikarenakan oleh kemampuan Brute Force dalam menyelesaikan masalah, yaitu karena hampir semua permasalahan dapat diselesaikan dengan Brute Force.

IV. Implementasi

Berikut implementasi strategi algoritma greedy dan brute force terhadap kasus assignment problem:

Diberikan soal Assignment Problem

	Job 1	Job 2	Job 3
Person 1	4	2	8
Person 2	2	5	7
Person 3	3	1	6

Algoritma greedy dengan bahasa python menggunakan bantuan library numpy

```python
#Greedy
import numpy as np
def greedy_assignment(cost_matrix):
    num_tasks = cost_matrix.shape[0]
    assigned_agents = [-1] * num_tasks
    used_agents = [False] * num_tasks

    for task in range(num_tasks):
        min_cost = float('inf')
        best_agent = -1
        for agent in range(num_tasks):
            if not used_agents[agent] and cost_matrix[task, agent] < min_cost:
                min_cost = cost_matrix[task, agent]
                best_agent = agent
        assigned_agents[task] = best_agent
        used_agents[best_agent] = True

    total_cost = sum(cost_matrix[task, assigned_agents[task]] for task in range(num_tasks))
    return assigned_agents, total_cost

# Contoh matriks biaya
cost_matrix = np.array([
    [4, 2, 8],
    [2, 5, 7],
    [3, 1, 6]
])

# Menggunakan algoritma greedy
greedy_assignment_result, greedy_total_cost = greedy_assignment(cost_matrix)

print("Hasil menggunakan algoritma greedy:")
for task, agent in enumerate(greedy_assignment_result):
    print(f"Job {task+1} diberikan kepada Person {agent+1} dengan biaya {cost_matrix[task, agent]}")
print(f"Total biaya greedy: {greedy_total_cost}")

```

```
Hasil menggunakan algoritma greedy:
Job 1 diberikan kepada Person 2 dengan biaya 2
Job 2 diberikan kepada Person 1 dengan biaya 2
Job 3 diberikan kepada Person 3 dengan biaya 6
Total biaya greedy: 10
```

Algoritma greedy dengan bahasa python menggunakan bantuan library numpy dan itertools

```python
#Brute Force
from itertools import permutations
import numpy as np
# Fungsi untuk menghitung biaya total dari penugasan tertentu
def calculate_total_cost(cost_matrix, assignment):
    total_cost = 0
    for task, agent in enumerate(assignment):
        total_cost += cost_matrix[task, agent]
    return total_cost

# Fungsi untuk memecahkan Assignment Problem menggunakan algoritma brute force
def solve_assignment_problem_brute_force(cost_matrix):
    num_tasks = cost_matrix.shape[0]
    all_assignments = permutations(range(num_tasks))

    min_cost = float('inf')
    best_assignment = None

    for assignment in all_assignments:
        current_cost = calculate_total_cost(cost_matrix, assignment)
        if current_cost < min_cost:
            min_cost = current_cost
            best_assignment = assignment

    return best_assignment, min_cost

# Contoh matriks biaya
cost_matrix = np.array([
    [4, 2, 8],
    [2, 5, 7],
    [3, 1, 6]
])

# Menggunakan algoritma brute force
brute_force_assignment_result, brute_force_total_cost = solve_assignment_problem_brute_force(cost_matrix)

print("\nHasil menggunakan algoritma brute force:")
for task, agent in enumerate(brute_force_assignment_result):
    print(f"Job {task} diberikan kepada Person {agent+1} dengan biaya {cost_matrix[task, agent]}")
print(f"Total biaya brute force: {brute_force_total_cost}")
```

```
Hasil menggunakan algoritma brute force:
Job 1 diberikan kepada Person 2 dengan biaya 2
Job 2 diberikan kepada Person 1 dengan biaya 2
Job 3 diberikan kepada Person 3 dengan biaya 6
Total biaya brute force: 10
```

V. Analisis:

Kompleksitas Waktu

- Algoritma Greedy: Kompleksitas waktu adalah O(n^2) karena untuk setiap dari n tugas, algoritma mencari melalui n agen yang belum digunakan.
- Algoritma Brute Force: Kompleksitas waktu adalah O(n!) karena algoritma mencoba semua permutasi dari penugasan tugas ke agen, yang berjumlah n! kemungkinan.

Perbandingan:

- Efisiensi: Algoritma greedy jauh lebih efisien dalam hal waktu komputasi dibandingkan dengan algoritma brute force. Greedy memiliki kompleksitas kuadrat, sedangkan brute force memiliki kompleksitas faktorial.
- Optimalitas: Algoritma brute force memberikan solusi optimal karena mencoba semua kemungkinan. Sebaliknya, algoritma greedy tidak selalu menghasilkan solusi optimal karena memilih penugasan dengan biaya minimum secara lokal tanpa mempertimbangkan dampak global.

Hasil Eksperimen:
Menggunakan matriks biaya yang sama, berikut adalah hasil dari kedua algoritma:
Greedy: Total biaya = 10
Brute Force: Total biaya = 10
Dalam contoh ini, kebetulan algoritma greedy menghasilkan solusi yang sama dengan brute force. Namun, ini tidak selalu terjadi untuk semua kasus.

VI. Kesimpulan

Algoritma greedy menawarkan solusi cepat dengan kompleksitas waktu yang lebih rendah tetapi tidak selalu optimal. Di sisi lain, algoritma brute force memastikan solusi optimal dengan biaya komputasi yang sangat tinggi. Pemilihan algoritma tergantung pada kebutuhan spesifik dan ukuran masalah yang dihadapi. Untuk kasus kecil, brute force dapat digunakan untuk mendapatkan solusi optimal, sementara untuk kasus besar, pendekatan greedy lebih praktis meskipun harus menerima risiko hasil yang tidak optimal.

VII. Daftar Pustaka

Wulandari, Gia S., Ph.D., dan Sa'adah, Siti, S.T., M.T. 2021. Pengantar Strategi Algoritma. KBM INDONESIA.

VIII. Lampiran