Perbandingan Algoritma Greedy dan Algoritma Branch and Bound
Kasus: Penjadwalan Proyek

Nazwa Aulia Rakhma[1]
Program Studi S1 Teknik Informatika
Fakultas Informatika
Institut Teknologi Telkom Purwokerto, Jl. D.I. Panjaitan No.128 Purwokerto 53147, Indonesia
[1]*21102015@ittelkom-pwt.ac.id*

Abstrak—Penjadwalan Proyek merupakan tantangan penting dalam manajemen pekerjaan yang melibatkan alokasi sumber daya dan penentuan urutan aktivitas untuk mencapai tujuan proyek yang efisien. Dalam literatur penjadwalan proyek, terdapat beberapa pendekatan algoritma yang telah dikembangkan untuk menyelesaikan masalah ini, diantaranya adalah dengan menggunakan Algoritma Greedy dan Algoritma Branch and Bound. Penulis melakukan eksperimen dengan menggunakan contoh data set aktivitas proyek yang semata mata dibuat untuk dibandingkan dalam pengujian kedua program. Penulis mengimplementasikan kedua algoritma dalam bahasa pemrograman Python dan menganalisis kinerja keduanya berdasarkan faktor-faktor seperti waktu eksekusi dan kualitas solusi yang dihasilkan. Hasil eksperimen menunjukkan bahwa Algoritma Greedy cenderung memiliki waktu eksekusi yang lebih cepat dibandingkan dengan Algoritma Branch and Bound dalam kasus-kasus tertentu. Namun Algoritma Greedy tidak menjamin solusi optimal. Di sisi lain, Algoritma Branch and Bound mampu menemukan solusi optimal tetapi membutuhkan waktu eksekusi yang lebih lama karena kompleksitasnya yang lebih tinggi. Oleh karena itu, keputusan dalam memilih pendekatan yang sesuai harus didasarkan pada karakteristik spesifik masalah dan tujuan penjadwalan proyek yang diinginkan.

Abstract— Project scheduling is an important challenge in work management that involves allocating resources and determining the sequence of activities to achieve efficient project goals. In the project scheduling literature, there are several algorithmic approaches that have been developed to solve this problem, including Greedy and Branch and Bound Algorithms. The author conducted experiments using a sample dataset of project activities that was solely created for comparison in testing the two programs. The author implemented both algorithms in Python and analyzed their performance based on factors such as execution time and the quality of the resulting solutions. The experimental results show that the Greedy Algorithm tends to have a faster execution time compared to Branch and Bound Algorithms in certain cases. However, the Greedy Algorithm does not guarantee an optimal solution. On the other hand, the Branch and Bound Algorithm is able to find the optimal solution but requires longer execution time due to its

higher complexity. Therefore, the decision in choosing a suitable approach should be based on the specific characteristics of the problem and the desired project scheduling objectives.

I. PENDAHULUAN

Dalam sebuah pengembangan sistem, penjadwalan proyek merupakan salah satu aspek penting yang harus diperhatikan. Penjadwalan proyek yang baik dapat membantu mengoptimalkan alokasi sumber daya, meminimalkan waktu penyelesaian dan meningkatkan efisiensi keseluruhan proyek. Untuk mencapai penjadwalan proyek yang efektif, terdapat berbagai algoritma yang dapat digunakan masing-masing dengan pendekatan dan karakteristik unik.

Dalam penelitian ini, memfokuskan pada perbandingan dua algoritma yang umum digunakan yaitu Algoritma Greedy dan Algoritma Branch and Bound. Tujuan utama dalam pembuatan jurnal ini adalah untuk menganalisis dan membandingkan kinerja kedua algoritma ini dalam menyelesaikan masalah penjadwalan proyek.

Algoritma Greedy adalah pendekatan yang sederhana dan mudah diimplementasikan di mana keputusan pengambilan langkah selanjutnya didasarkan pada informasi lokal terbaik pada saat itu. Algoritma ini cenderung memberikan solusi yang cepat namun tidak menjamin solusi yang optimal. Dalam konteks penjadwalan proyek, algoritma Greedy sering digunakan untuk memilih aktivitas berdasarkan estimasi waktu terpendek atau kriteria lainnya.

Di sisi lain, Algoritma Branch and Bound adalah pendekatan yang lebih kompleks. Algoritma ini menggunakan teknik pemotongan (*pruning*) untuk mencapai solusi optimal dengan cara membatasi ruang pencarian. Dengan melakukan pemotongan cerdas, Algoritma Branch and Bound dapat mencapai solusi yang optimal pada masalah penjadwalan proyek yang kompleks. Namun, implementasi algoritma ini lebih rumit dan memerlukan perhitungan tambahan untuk menentukan pemotongan yang efektif.

Diharapkan dengan adanya penelitian ini, akan memberikan pemahaman yang lebih baik tentang kelebihan dan kelemahan masing-masing algoritma serta memberikan panduan praktis bagi pengembang dan peneliti dalam memilih algoritma yang paling sesuai untuk penyelesaian masalah penjadwalan proyek secara efektif dan efisien.

II. DASAR TEORI

1. Penjadwalan Proyek

   Penjadwalan proyek adalah proses menentukan urutan aktivitas dalam proyek, alokasi sumber daya dan estimasi waktu yang diperlukan untuk menyelesaikan proyek secara efisien (Husein, 2011). Penjadwalan proyek bertujuan untuk mengoptimalkan penggunaan sumber daya dan mengatur aktivitas proyek sedemikian rupa sehingga tujuan proyek dapat dicapai dengan efektif.

2. Algoritma Greedy

   Algoritma Greedy adalah pendekatan penyelesaian masalah yang memilih langkah terbaik secara lokal pada setiap iterasi tanpa mempertimbangkan keseluruhan gambaran (Rinaldi Munir, “Algoritma Greedy”, ITB). Pada penjadwalan proyek, Algoritma Greedy memilih aktivitas berdasarkan kriteria tertentu, seperti durasi terpendek atau ketergantungan pada setiap langkah. Namun, Algoritma Greedy tidak menjamin solusi optimal dan dapat menghasilkan solusi yang suboptimal.

3. Algoritma Branch and Bound

   Algoritma Branch and Bound adalah pendekatan penyelesaian masalah yang menggunakan *backtracking* dan *pruning* untuk mencari solusi optimal atau mendekati solusi optimal (Rinaldi Munir, Nur Ulfa Maulidevi, Masayu Leylia Khodra, ITB). Pada penjadwalan

proyek, Algoritma Branch and Bound mencoba semua kemungkinan penjadwalan aktivitas dan menggunakan pemotongan untuk mempercepat pencarian.

4. Kriteria Perbandingan

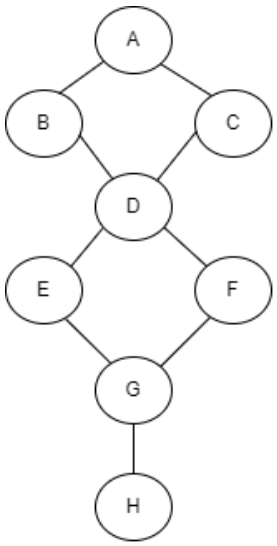
Dalam jurnal ini, kriteria perbandingan antara Algoritma Greedy dan Algoritma Branch and Bound meliputi waktu eksekusi dan kualitas solusi yang dihasilkan. Waktu eksekusi mengacu pada waktu yang diperlukan oleh masing-masing algoritma untuk menyelesaikan penjadwalan proyek. Kualitas solusi mencerminkan sejauh mana solusi yang dihasilkan oleh masing-masing algoritma mendekati solusi optimal dengan mempertimbangkan ketergantungan antaraktivitas dan durasi aktivitas.

III. IMPLEMENTASI

Untuk dapat mengisi hasil dari penelitian ini, diperlukan sebuah implementasi dari contoh penjadwalan proyek. Misalnya CEO PT. Badak memiliki proyek konstruksi bangunan. Setiap aktivitas memiliki estimasi waktu penyelesaian dan ketergantungan terhadap aktivitas lain dalam proyek.

Aktivitas	Estimasi Waktu (hari)	Aktivitas Sebelumnya
A	4	-
B	5	A
C	2	A
D	3	B,C
E	6	D
F	4	D
G	2	F,
H	3	E,G

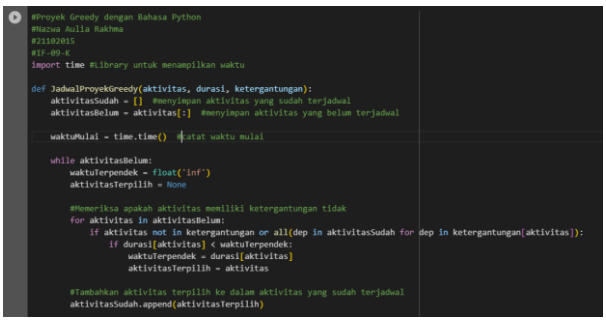
Tabel 1.1 Penjadwalan Proyek



Gambar 1.2 Graf Ketergantungan Setiap Proyek

IV. PENGUJIAN DAN ANALISIS

Hasil penelitian menunjukkan bahwa implementasi Algoritma Greedy dan Algoritma Branch and Bound dalam penjadwalan proyek menghasilkan temuan yang berbeda. Dalam pengujiannya menggunakan *Google Colab* dengan kapasitas laptop RAM 4GB dan Prosesor *Intel(R) Celeron(R)* CPU N2840 @ 2.16GHz 2.16 GHz. Untuk Algoritma Greedy diberikan hasil seperti pada gambar 1.3 dan 1.4. Sedangkan untuk implementasi Algoritma Branch and Bound ditampilkan pada gambar 1.5, 1.6 dan 1.7.



```
#Proyek Greedy dengan Bahasa Python
#Nazwa Aulia Rakhma
#21102015
#IF-09-K
import time #Library untuk menampilkan waktu

def JadwalProyekGreedy(aktivitas, durasi, ketergantungan):
    aktivitasSudah = []  #menyimpan aktivitas yang sudah terjadwal
    aktivitasBelum = aktivitas[:]  #menyimpan aktivitas yang belum terjadwal

    waktuMulai = time.time()  #catat waktu mulai

    while aktivitasBelum:
        waktuTerpendek = float('inf')
        aktivitasTerpilih = None

        #Memeriksa apakah aktivitas memiliki ketergantungan tidak
        for aktivitas in aktivitasBelum:
            if aktivitas not in ketergantungan or all(dep in aktivitasSudah for dep in ketergantungan[aktivitas]):
                if durasi[aktivitas] < waktuTerpendek:
                    waktuTerpendek = durasi[aktivitas]
                    aktivitasTerpilih = aktivitas

        #Tambahkan aktivitas terpilih ke dalam aktivitas yang sudah terjadwal
        aktivitasSudah.append(aktivitasTerpilih)
```

Gambar 1.3 *Coding Python* untuk Penjadwalan dengan Algoritma Greedy

```
        #Hapus aktivitas terpilih dari aktivitas yang belum terjadwal
        aktivitasBelum.remove(aktivitasTerpilih)

    waktuSelesai = time.time()  #catat waktu selesai
    lamaEksekusi = waktuSelesai - waktuMulai  #lama waktu eksekusi

    return aktivitasSudah, lamaEksekusi

#Inisialisasi data aktivitas, durasi dan ketergantungan
aktivitas = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
durasi = {'A': 4, 'B': 5, 'C': 2, 'D': 3, 'E': 6, 'F': 4, 'G': 2, 'H': 3}
ketergantungan = {'B': ['A'], 'C': ['A'], 'D': ['B', 'C'], 'E': ['D'], 'F': ['D'], 'G': ['F'], 'H': ['E', 'G']}

#Panggil fungsi JadwalProyekGreedy untuk mendapatkan jadwal penyelesaian proyek dan lama waktu eksekusi
aktivitasSudah, lamaEksekusi = JadwalProyekGreedy(aktivitas, durasi, ketergantungan)

#Tampilkan hasil jadwal penyelesaian proyek dan lama waktu eksekusi
print("Jadwal Penyelesaian Proyek Greedy:")
print(aktivitasSudah)
print("Lama waktu eksekusi:", lamaEksekusi, "detik")

Jadwal Penyelesaian Proyek Greedy:
['A', 'C', 'B', 'D', 'F', 'G', 'E', 'H']
Lama waktu eksekusi: 4.3392181396484375e-05 detik
```

Gambar 1.4 Hasil Penjadwalan dengan Algoritma Greedy

```
# Proyek Branch and Bound
# Nazwa Aulia Rakhma
# 21102015
# IF-05-K
import time  # Library untuk menampilkan waktu

def JadwalProyekBB(aktivitas, durasi, ketergantungan):
    def backtracking(jadwalSekarang, aktivitasLain):  # Lainnya dalam artian sisa dari aktivitas
        nonlocal jadwalTerbaik

        # Jika semua aktivitas telah ditempatkan pada jadwal
        if not aktivitasLain:
            if not jadwalTerbaik or totalDurasi(jadwalSekarang) < totalDurasi(jadwalTerbaik):
                jadwalTerbaik = jadwalSekarang
            return

        for aktivitas in aktivitasLain:
            if aktivitas not in ketergantungan or all(dep in jadwalSekarang for dep in ketergantungan[aktivitas]):
                jadwalBaru = jadwalSekarang + [aktivitas]
                sisaAktBaru = aktivitasLain[:]
                sisaAktBaru.remove(aktivitas)
                backtracking(jadwalBaru, sisaAktBaru)
```

Gambar 1.5 *Coding Python* 1 untuk Penjadwalan dengan Algoritma Branch and Bound

```
    def totalDurasi(jadwal):
        total = 0
        for aktivitas in jadwal:
            total += durasi[aktivitas]
        return total

    jadwalSudah = []  # Menyimpan aktivitas yang telah terjadwal
    aktivitasBelum = aktivitas[:]  # Menyimpan aktivitas yang belum terjadwal
    jadwalTerbaik = None

    waktuMulai = time.time()  # Catat waktu mulai

    # Panggil fungsi backtracking untuk mencari solusi terbaik
    backtracking(jadwalSudah, aktivitasBelum)

    waktuSelesai = time.time()  # Catat waktu selesai
    lamaEksekusi = waktuSelesai - waktuMulai  # Lama waktu eksekusi

    return jadwalTerbaik, lamaEksekusi

# Inisialisasi data aktivitas, durasi, dan ketergantungan
aktivitas = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
durasi = {'A': 4, 'B': 5, 'C': 2, 'D': 3, 'E': 6, 'F': 4, 'G': 2, 'H': 3}
ketergantungan = {'B': ['A'], 'C': ['A'], 'D': ['B', 'C'], 'E': ['D'], 'F': ['D'], 'G': ['F'], 'H': ['E', 'G']}

# Panggil fungsi JadwalProyekBB untuk mendapatkan jadwal penyelesaian proyek dan lama waktu eksekusi
jadwalTerbaik, lamaEksekusi = JadwalProyekBB(aktivitas, durasi, ketergantungan)
```

Gambar 1.6 *Coding Python* 2 untuk Penjadwalan dengan Algoritma Branch and Bound

```
# Tampilkan hasil jadwal penyelesaian proyek dan lama waktu eksekusi
print("Jadwal Penyelesaian Proyek B&B:")
print(jadwalTerbaik)
print("Lama waktu eksekusi:", lamaEksekusi, "detik")

Jadwal Penyelesaian Proyek B&B:
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
Lama waktu eksekusi: 9.036064147949219e-05 detik
```

Gambar 1.7 Hasil Penjadwalan dengan Algoritma Branch and Bound

Berdasarkan pengujian yang dilakukan, Algoritma Greedy menunjukkan waktu eksekusi yang lebih cepat dengan kompleksitas waktu linier terhadap jumlah aktivitas. Namun solusi yang diberikan oleh Algoritma Greedy cenderung suboptimal karena hanya mempertimbangkan pilihan terbaik pada setiap langkah tanpa mempertimbangkan konsekuensi jangka panjang.

Di sisi lain, Algoritma Branch and Bound menunjukkan waktu eksekusi yang lebih lama, terutama pada kasus dengan jumlah aktivitas yang besar. Namun algoritma ini mampu mencari solusi optimal dengan memperhitungkan batasan dan menggunakan teknik pemotongan untuk mengurangi ruang pencarian.

Berdasarkan hasil pengujian tersebut, pemilihan antara Algoritma Greedy dan Algoritma Branch and Bound dalam penjadwalan proyek harus didasarkan pada pertimbangan antara waktu eksekusi dan kualitas solusi yang diinginkan. Jika waktu eksekusi menjadi faktor kritis dan solusi yang cukup baik sudah memadai maka Algoritma Greedy dapat menjadi pilihan yang tepat. Namun jika diperlukan solusi optimal dengan mengorbankan waktu eksekusi yang lebih lama, Algoritma Branch and Bound lebih disarankan.

VI. PENUTUP

6.1 KESIMPULAN

1. Dalam penyelesaian menggunakan Algoritma Greedy, cenderung menghasilkan solusi yang suboptimal hal ini karena Algoritma Greedy hanya mempertimbangkan pilihan terbaik pada setiap langkah tanpa mempertimbangkan konsekuensi jangka panjang.
2. Dalam penyelesaian menggunakan Algoritma Branch and Bound, meskipun membutuhkan waktu eksekusi yang lebih lama terutama pada kasus dengan jumlah aktivitas yang besar tapi akan menghasilkan solusi optimal.
3. Jika waktu eksekusi yang cepat lebih diutamakan dan solusi suboptimal dapat diterima, Algoritma Greedy dapat menjadi pilihan yang baik. Namun jika diperlukan solusi optimal yang mempertimbangkan

batasan dan ketergantungan antar aktivitas, Algoritma Branch and Bound lebih sesuai.

6.2 SARAN

1. Untuk penelitian selanjutnya, disarankan untuk melakukan eksperimen yang lebih luas dengan berbagai jenis proyek dan skenario yang berbeda.
2. Melakukan penelitian selanjutnya untuk membandingkan keefektifan dengan menggunakan algoritma lain.
3. Mengembangkan dan menguji varian atau modifikasi dari Algoritma Greedy dan Algoritma Branch and Bound untuk meningkatkan kinerja atau memenuhi kebutuhan khusus penjadwalan proyek lainnya yang lebih kompleks.

REFERENCES

[1] Rochman, Nur dan Budi Santosa. (2017). Penerapan Algoritma Branch and Bound dan Greedy pada Permasalahan Penjadwalan Proyek. *Jurnal Teknik Pomits*. Diakses 24 Juni 2023 pukul 09.21 WIB

[2] Wulandari, R., & Suryani, E. (2019). Penerapan Algoritma Branch and Bound dan Greedy pada Penyelesaian Persoalan Jarak Terpendek. *Jurnal Ilmiah JATI UNIKOM*. Diakses 24 Juni 2023 Pukul 09.48 WIB

PERNYATAAN

Dengan ini saya menyatakan bahwa *paper* yang penulis tulis ini adalah tulisan sendiri, bukan saduran atau terjemahan dari makalah orang lain dan bukan plagiasi.

Purwokerto, 5 Juli 2023

Nazwa Aulia Rakhma

21102015