Perbandingan Algoritma Greedy dan Branch and Bound dalam Penjadwalan Job Freelance yang Memaksimalkan Keuntungan

Melvin Kent Jonathan -13521052
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521052@std.stei.itb.ac.id

***Abstrak*— Pekerjaan sebagai seorang *freelancer* mulai diminati oleh masyarakat kini. Fleksibilitas yang ditawarkan memampukan para *freelancer* untuk menerima berbagai pekerjaan sekaligus dalam satu periode waktu. Hal ini bertujuan untuk memaksimalkan upah yang diterima oleh sang *freelancer*. Namun, untuk memaksimalkan upah yang dapat diterima dari sekumpulan pekerjaan dengan durasi pengerjaan, tenggat waktu, dan bayaran yang berbeda dibutuhkan sebuah mekanisme penentu. Persoalan ini menjadi persoalan maksimasi keuntungan dengan mengatur urutan pengerjaan pekerjaan. Makalah ini membahas perbandingan algoritma *greedy* dan *branch and bound* dalam memaksimalkan keuntungan yang diterima seorang *freelancer* melalui penjadwalan *job freelance*. Hasil pengujian implementasi menunjukan bahwa algoritma *branch and bound* menghasilkan solusi penjadwalan dengan keuntungan atau bayaran yang lebih besar ketimbang solusi yang dihasilkan algoritma *greedy*.**

Kata kunci—greedy; branch and bound; freelance; penjadwalan pekerjaan

I. PENDAHULUAN

Seiring perkembangan zaman, pekerjaan sebagai seorang pekerja lepas (*freelancer*) semakin banyak diminati oleh masyarakat karena berbagai keuntungan yang ditawarkan, mulai dari fleksibilitas, hingga besarnya total bayaran yang dapat diperoleh dalam satu waktu. Pekerjaan yang diterima dari *freelance job* umumnya membebaskan sang pekerja untuk mengatur waktunya. Pemberi kerja atau *client* hanya menentukan rincian pekerjaan, tenggat waktu, serta bayaran yang diberikan kepada *freelancer*. Fleksibilitas ini sering kali dijadikan kesempatan oleh para *freelancer* untuk mengambil berbagai *job*/pekerjaan yang berbeda dalam satu waktu untuk memaksimalkan potensi bayaran yang dapat ia peroleh. Namun, untuk dapat memaksimalkan bayaran, seorang *freelancer* harus dapat mengatur urutan pekerjaan yang harus dikerjakannya terlebih dahulu. Hal ini karena umumnya para *freelancer* akan memberikan jaminan diskon harga sekian persen dari total kesepakatan pembayaran di awal per hari keterlambatan yang terjadi. Oleh karena itu, dibutuhkan suatu mekanisme yang dapat membantu para freelancer untuk menentukan penjadwalan pekerjaan yang diterimanya demi memaksimalkan upah atau bayaran yang dapat diterimanya.

II. LANDASAN TEORI

A. *Algoritma Greedy*

Pengertian

Algoritma *greedy* merupakan sebuah metode pemecahan persoalan optimasi yang populer digunakan karena tingkat abstraksi yang tergolong sederhana. Terdapat dua jenis persoalan optimasi, yakni maksimasi (*maximization*) dan minimasi (*minimization*). Algoritma *greedy* memecahkan pesoalan optimasi dengan cara memilih solusi optimum lokal langkah per langkah tanpa memperhatikan konsekuensi ke depan dengan harapan sekuens atau urutan solusi optimum lokal tersebut akan mengarah kepada solusi akhir yang paling optimum (optimum global). Cara kerja algoritma *greedy* mudah dipahami dengan menyandingkannya dengan prinsip *greedy*, yakni "*take what you can get now!"*

Karakteristik Persoalan *Greedy*

Persoalan yang cocok untuk diselesaikan dengan algoritma *greedy* dapat diidentifikasi dengan meninjau dua sifat berikut:

- Solusi optimal dari persoalan dapat ditentukan dari solusi optimal sub persoalan tersebut
- Pada setiap sub persoalan (langkah untuk menyelesaikan persoalan besar yang awal), terdapat suatu langkah yang dapat dilakukan yang mana langkah tersebut menghasilkan solusi optimal pada sub persoalan tersebut. Dalam terminologi algoritma *greedy*, langkah ini disebut sebagai *greedy choice*.

Elemen-Elemen Algoritma Greedy:

- Himpunan Kandidat (C)

  Himpunan kandidat merupakan himpunan yang berisi opsi-opsi yang akan dipilih pada setiap langkah.

- Himpunan Solusi (S)

  Himpunan solusi merupakan himpunan yang berisi kandidat-kandidat yang telah dipilih.

- Fungsi Solusi (*Solution Function*)

Fungsi solusi merupakan fungsi yang menentukan apakah pilihan himpunan kandidat telah menghasilkan solusi.

- Fungsi Seleksi (*Selection Function*)

  Fungsi seleksi merupakan fungsi yang berfungsi memilih kandidat berdasarkan strategi *greedy* tertentu yang bersifat heuristik.

- Fungsi Kelayakan (*Feasibility Function*)

  Fungsi kelayakan merupakan fungsi yang mengevaluasi apakah kandidat yang dipilih layak untuk dimasukkan ke dalam himpunan solusi

- Fungsi Objektif (objective function)

  Fungsi objektif merupakan fungsi yang memaksimumkan atau meminimumkan suatu variabel dalam persoalan.

Pembuktian Optimalitas

Pembuktikan bahwa solusi yang dihasilkan dari algoritma *greedy* merupakan solusi yang optimal dapat dilakukan dilakukan secara matematis. Pembuktian matematis ini umumnya menimbulkan tantangan tersendiri dalam proses formulasi persoalan menjadi frasa matematika yang dapat menggeneralisasi persoalan dan solusi. Di sisi lain, pembuktian bahwa solusi yang dihasilkan algoritma *greedy* tidaklah optimal dapat dilakukan dengan menyajikan *counter example*, yakni sebuah kasus yang menunjukkan bahwa solusi optimal tidak diperoleh.

B. Algoritma Branch and Bound

Pengertian

Algoritma *branch and bound* merupakan suatu metode penyelesaian persoalan optimasi, yakni melakukan maksimasi atau minimasi suatu fungsi objektif yang tidak melanggar batasan persoalan (*constraints*). Algoritma *branch and bound* dapat dipandang sebagai perpaduan antara *Breadth First Search* (BFS) dengan *least cost search*.

Pada algoritma *branch and bound*, setiap simpul diberi sebuah nilai *cost* ĉ(i), yakni nilai taksiran atau estimasi ongkos termurah (*lower* bound) lintasan dari simpul i ke simpul status tujuan. Perhitungan nilai ĉ(i) dilakukan secara heuristik. Tak seperti pada *Breadth First Search*, pembangkitan simpul yang akan di-*expand* berikutnya dilakukan berdasarkan *cost* atau ongkos terkecil (pada kasus minimasi).

Pemangkasan oleh Fungsi Pembatas

Selain itu, algoritma *branch and bound* akan melakukan pemangkasan atau *bound* pada jalur yang teridentifikasi tidak mengarah pada solusi atau tujuan. Terdapat beberapa kriteria untuk mengidentifikasi simpul yang perlu dipangkas, yakni:

- Nilai simpul tidak lebih baik dari nilai terbaik sejauh ini (the best solution so far).
- Simpul tidak merepresentasikan solusi yang ‘feasible’ karena ada batasan yang dilanggar.
- Solusi pada simpul tersebut hanya terdiri atas satu titik → tidak ada pilihan lain; bandingkan nilai fungsi obyektif dengan solusi terbaik saat ini, yang terbaik yang diambil.

Rangkaian Algoritma

1. Masukkan simpul akar (*root node*) ke dalam antrian Q. Jika simpul akar merupakan simpul solusi (simpul tujuan), maka solusi telah ditemukan. Jika hanya satu solusi yang diinginkan, maka hentikan proses.
2. Jika antrian Q kosong, hentikan proses.
3. Jika antrian Q tidak kosong, pilih simpul I dari anrian Q dengan nilai ‘cost’ ĉ(j) yang paling kecil. Apabila didapati beberapa simpul I yang memenuhi syarat tersebut, pilih salah satunya secara acak.
4. Jika simpul i merupakan simpul solusi, berarti solusi telah ditemukan. Jika hanya satu solusi yang diinginkan, maka hentikan proses. Dalam kasus optimisasi dengan pendekatan pencarian biaya terendah, periksa biaya semua simpul yang hidup. Jika biaya simpul tersebut lebih besar dari biaya simpul solusi, maka nonaktifkan simpul tersebut.
5. Jika simpul i bukan simpul solusi, bangkitkan seluruh anak simpul tersebut. Jika i tidak memiliki anak, kembali ke langkah 2.
6. Untuk setiap anak j dari simpul i, hitung ĉ(j) dan masukkan semua anak tersebut ke dalam Q.
7. Kembali ke langkah 2.

III. ANALISIS PERSOALAN

Persoalan penjadwalan job *freelance* yang diangkat pada tulisan ini merupakan persoalan optimasi dengan objektif untuk memaksimalkan keuntungan atau bayaran yang diperoleh *freelancer*. Algoritma *greedy* menjadi pilihan yang natural dalam menyelesaikan persoalan optimasi ini. Namun, penyelesaian persoalan dengan algoritima *greedy* tidak menjamin keuntungan total yang diperoleh adalah maksimal global. Oleh karena itu, algoritma *branch and bound* dipakai sebagai alternatif sekaligus pembanding performa penjadwalan job yang ada. Terlepas dari hasil akhir yang diperoleh masing-masing algoritma, tidak berarti algortima yang satu menjadi pilihan algoritma terbaik mutlak ketimbang algoritma lainnya. Hal ini dikarenakan terdapat faktor-faktor di luar tinjauan tulisan ini yang dapat memengaruhi pengambilan keputusan *freelancer* dalam menjadwalkan pekerjaan-pekerjaannya.

A. Analisis Elemen Persoalan

Terdapat beberapa elemen persoalan yang menjadi konsiderasi dalam pengurutan jadwal pekerjaan, yakni:

1. Bayaran

   Dalam menerima suatu pekerjaan, bayaran menjadi salah satu faktor yang disepakati oleh *freelancer* dan *client* sebelum memulai perjanjian kerja. Setiap pekerjaan yang diterima *freelancer* memiliki besar

bayaran yang berbeda. Setiap *freelancer* memiliki metode standardisasi bayaran yang diterimanya. Ada yang menetapkan tarif per jam pengerjaan. Ada pula yang mempertimbangkan tingkat kesulitan serta tenggat yang diberikan oleh *client*. Pada tulisan ini, bayaran yang diterima oleh *freelancer* untuk satu pekerjaan diasumsikan tidak memiliki korelasi khusus dengan durasi pengerjaan maupun tenggat waktunya. Penentuan bayaran akan dilakukan secara eksak di awal untuk mempermudah perhitungan dan penerapan algoritma.

2. Durasi Pengerjaan

   Durasi pengerjaan menjadi faktor penting dalam penentuan urutan pekerjaan sebab pergantian pekerjaan dalam kasus yang dibahas pada tulisan ini dibatasi agar hanya terjadi ketika pekerjaan sebelumnya telah selesai dilakukan. Artinya, tidak ada pengerjaan pekerjaan yang boleh dihentikan untuk pengerjaan pekerjaan lain sebelum pekerjaan tersebut selesai. Adapun satuan yang dipakai dalam menghitung durasi pengerjaan adalah satuan hari. *Freelancer* juga diasumsikan bekerja setiap hari tanpa adanya hari istirahat. Oleh karena durasi pengerjaan setiap pekerjaan telah ditentukan di awal serta adanya batasan bahwa setiap pekerjaan harus diselesaikan, total durasi pengerjaan pun dapat dihitung dari awal.

3. Tenggat Waktu

   Tenggat waktu merupakan ekspektasi selesainya pengerjaan suatu pekerjaan. Setiap pekerjaan diasumsikan hanya memiliki satu tenggat waktu dan telah ditentukan di awal kesepakatan. Tenggat waktu untuk setiap pekerjaan akan memakai satuan hari yang merepresentasikan sisa hari tenggat pengerjaan dari hari ini.

4. Keterlambatan

   Keterlambatan pengerjaan menjadi salah satu hal yang tidak diinginkan oleh *client*. Oleh karena itu, pemberian diskon bayaran menjadi salah satu jaminan yang dapat diajukan oleh seorang *freelancer* untuk memberikan rasa aman bagi *client*-nya. Diskon yang diterapkan umumnya akan membesar seiring bertambahnya waktu keterlambatan. Untuk mempermudah perhitungan, diskon yang diberikan diasumsikan sebesar 5% dari tarif yang disepakati di awal untuk setiap hari keterlambatan.

B. Identifikasi Elemen Algoritma Greedy

- Himpunan Kandidat (C)

  Himpunan kandidat pada persoalan ini merupakan pekerjaan-pekerjaan yang dapat dipilih untuk setiap urutan.

- Himpunan Solusi (S)

  Himpunan solusi pada persoalan ini merupakan permutasi urutan seluruh pekerjaan.

- Fungsi Solusi (solution function)

  Fungsi solusi pada persoalan ini didefinisikan sebagai hari diselesaikannya seluruh pekerjaan yang diterima *freelancer*. Hari tersebut dapat dihitung dengan menghitung total durasi pengerjaan dari pekerjaan-pekerjaan yang ada.

- Fungsi Seleksi (selection function)

  Fungsi seleksi pada persoalan ini akan didasari pada bayaran yang didapat untuk setiap pekerjaan saat telah diselesaikannya pekerjaan tersebut. Fungsi seleksi akan memilih pekerjaan selanjutnya dengan bayaran terbesar. Bayaran yang dimaksud telah memperhitungkan diskon yang diterapkan apabila akan terjadi keterlambatan.

- Fungsi Kelayakan (feasibility function)

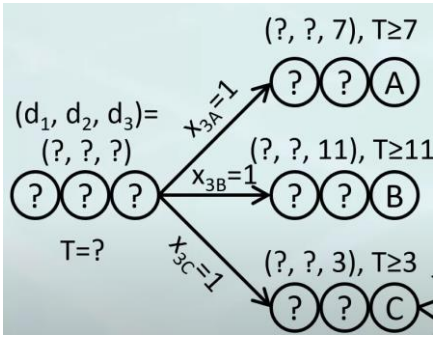
  Oleh karena tidak adanya batasan bahwa pekerjaan yang telah lewat tenggat waktu tidak lagi dapat dikerjakan oleh freelancer, fungsi kelayakan pada persoalan ini akan mengganggap seluruh pekerjaan atau sekuens pekerjaan tergolong layak.

- Fungsi Objektif (objective function)

  Fungsi objektif pada persoalan ini adalah memaksimalkan bayaran yang diterima oleh *freelancer* dengan mempertimbangkan diskon yang diterapkan apabila terjadi keterlambatan.

C. Identifikasi Elemen Algoritma Branch and Bound

Pada persoalan penjadwalan job freelance, dapat digambarkan sebuah pohon yang berekspansi. Setiap simpul pada suatu level yang sama dari pohon menggambarkan opsi pekerjaan yang dipilih untuk urutan pekerjaan pada level tersebut.



Gambar 1. Pohon Ekspansi *Job Scheduling* (Sumber: https://www.youtube.com/watch?v=UGvc-qujB-o, diakses pada 22 Mei 2023)

Simpul hidup yang menjadi simpul expand ialah simpul yang memiliki *cost* terkecil. *Cost* dihitung berdasarkan diskon yang diberikan oleh *freelancer* apabila terjadi keterlambatan pengerjaan pada pekerjaan yang dipilih. Adapun perumusan perhitungan *cost* pada simpul *i* dapat dilihat pada persamaan berikut.

Identify applicable sponsor/s here. If no sponsors, delete this text box (*sponsors).*

$c(i) = (\text{hari penyelesaian}_i - \text{hari tenggat}_i) * 0.05 * \text{bayaran}_i$ (1)

Optimasi Ekpansi Simpul

Untuk melakukan optimasi ekspansi simpul, tahap pemilihan pekerjaan akan dilakukan dari pekerjaan paling akhir ke pekerjaan paling awal. Hal ini memudahkan penyeleksian simpul yang akan di-*expand* sebab keterlambatan cenderung muncul di akhir waktu pengerjaan. Apabila pemilihan pekerjaan dilakukan dari pekerjaan paling awal, *cost* untuk setiap simpul pasti akan bernilai 0 sehingga setiap simpul akan mendapat prioritas ekspansi dan membuat proses ekpansi menjadi tidak optimal.

IV. Implementasi

Input

```
def input_job():
    n = int(input("Masukkan banyaknya pekerjaan: "))

    list_job = []
    for i in range(n):
        nama = input(f"Nama pekerjaan ke-{i+1}: ")
        durasi = int(input(f"Durasi pekerjaan ke-{i+1}: "))
        tenggat = int(input(f"Tenggat pekerjaan ke-{i+1}: "))
        bayaran = int(input(f"Bayaran pekerjaan ke-{i+1}: "))
        list_job.append([nama, durasi, tenggat, bayaran])

    return list_job
```

Greedy

```
def hitung_bayaran(job, waktu_mulai):
    keterlambatan = waktu_mulai + job[1] - job[2]

    if (keterlambatan < 0):
        keterlambatan = 0

    diskon = 0.05 * keterlambatan * job[3]
    bayaran = job[3] - diskon

    return bayaran

def pilih_index_job(list_job, waktu_mulai):
    index_max = 0
    bayaran_max = hitung_bayaran(list_job[index_max], waktu_mulai)
    for i in range(len(list_job)):
        if (i!=0) :
            bayaran = hitung_bayaran(list_job[i], waktu_mulai)
            if (bayaran > bayaran_max):
                index_max = i
                bayaran_max = bayaran

    return index_max, bayaran_max

def greedy(list_job):
    new_list_job = list_job[:]
    banyak_job = len(new_list_job)
    waktu_mulai = 0
    total_bayaran = 0

    job_terurut = []

    for i in range(banyak_job):
        index, bayaran = pilih_index_job(new_list_job, waktu_mulai)
        total_bayaran += bayaran
        removed_job = new_list_job.pop(index)
        job_terurut.append(removed_job)
        waktu_mulai = waktu_mulai + removed_job[1]

    print("\n===== Greedy =====")
    print("Urutan pekerjaan: ")
    print(job_terurut)
    print("Bayaran yang diterima: ", total_bayaran)
```

Branch and Bound

```
def hitung_diskon(job, waktu_selesai):
    keterlambatan = waktu_selesai - job[2]

    if (keterlambatan < 0):
        keterlambatan = 0

    diskon = 0.05 * keterlambatan * job[3]

    return diskon

def enqueue(queue, new_element):
    # new element : [cost, list_job_terpilih, list_job_tersisa]

    for i in range(len(queue)):
```

```
        if (queue[i][0] <= new_element[0]):
            queue.insert(i, new_element)
            return
    queue.append(new_element)

def dequeue(queue):
    return queue.pop(0)

def bnb(list_job):
    queue = []
    waktu_selesai = 0
    total_bayaran = 0
    for i in range(len(list_job)):
        waktu_selesai += list_job[i][1]
        total_bayaran += list_job[i][3]

    for i in range(len(list_job)):
        cost = hitung_diskon(list_job[i], waktu_selesai)
        waktu_selesai_next = waktu_selesai -
list_job[i][1]
        list_job_terpilih = [list_job[i]]
        list_job_tersisa = list_job[:]
        list_job_tersisa.pop(i)
        enqueue(queue, [cost, waktu_selesai_next,
list_job_terpilih, list_job_tersisa])

    found = False
    list_job_solusi = []
    solution_cost = 0
    # expand
    while (len(queue) > 0):
        expand = dequeue(queue)
        cost = expand[0]
        waktu_selesai = expand[1]
        list_job_terpilih = expand[2]
        list_job_tersisa = expand[3]
        if ((not found) and list_job_tersisa == []):
            found = True
            solution_cost = cost
            list_job_solusi = list_job_terpilih[:]
        elif (found and list_job_tersisa == []):
            if (cost < solution_cost):
                solution_cost = cost
                list_job_solusi = list_job_terpilih[:]
        elif (not found or (found and cost <=
solution_cost)) :
            for i in range(len(list_job_tersisa)):
                new_cost = cost +
hitung_diskon(list_job_tersisa[i], waktu_selesai)
                new_waktu_selesai = waktu_selesai -
list_job_tersisa[i][1]
                new_list_job_terpilih =
list_job_terpilih[:]
                new_list_job_tersisa =
list_job_tersisa[:]

new_list_job_terpilih.append(new_list_job_tersisa.pop(0))
                # print(new_list_job_terpilih)
                # print(new_list_job_tersisa)
                enqueue(queue, [new_cost,
new_waktu_selesai, new_list_job_terpilih,
new_list_job_tersisa])

    list_job_solusi.reverse()
    total_bayaran -= solution_cost

    print("\n===== Branch and Bound =====")
    print("Urutan pekerjaan: ")
    print(list_job_solusi)
    print("Bayaran yang diterima: ", total_bayaran)
```

Main

```
from bnb import *
from greedy import *

list_job = input_job()


print("===== HASIL =====")

greedy(list_job)
bnb(list_job)
```

V. Pengujian Dan Analisis

A. Pengujian

TABLE I. KASUS UJI 1

Job	Durasi (Hari)	Tenggat Waktu (Hari)	Bayaran ($)
A	2	5	120

B	3	4	150
C	4	6	100

Hasil kasus uji 1:

```
===== HASIL =====

===== Greedy =====
Urutan pekerjaan:
[['C', 4, 6, 300], ['B', 3, 4, 150], ['A', 2, 5, 120]]
Bayaran yang diterima:  523.5

===== Branch and Bound =====
Urutan pekerjaan:
[['C', 4, 6, 300], ['A', 2, 5, 120], ['B', 3, 4, 150]]
Bayaran yang diterima:  532.5
```

TABLE II. KASUS UJI 2

Job	Durasi (Hari)	Tenggat Waktu (Hari)	Bayaran ($)
A	5	6	150
B	2	3	100
C	3	5	180
D	4	4	120

Hasil kasus uji 2:

```
===== HASIL =====

===== Greedy =====
Urutan pekerjaan:
[['C', 3, 5, 180], ['A', 5, 6, 150], ['D', 4, 4, 120], ['B', 2, 3, 100]]
Bayaran yang diterima:  432.0

===== Branch and Bound =====
Urutan pekerjaan:
[['C', 3, 5, 180], ['B', 2, 3, 100], ['A', 5, 6, 150], ['D', 4, 4, 120]]
Bayaran yang diterima:  460.0
```

TABLE III. KASUS UJI 3

Job	Durasi (Hari)	Tenggat Waktu (Hari)	Bayaran ($)
A	3	4	160
B	4	5	120
C	2	3	180
D	5	6	140
E	6	7	100

Hasil kasus uji 3:

```
===== HASIL =====

===== Greedy =====
Urutan pekerjaan:
[['C', 2, 3, 180], ['A', 3, 4, 160], ['D', 5, 6, 140], ['B', 4, 5, 120],
 ['E', 6, 7, 100]]
Bayaran yang diterima:  545.0

===== Branch and Bound =====
Urutan pekerjaan:
[['D', 5, 6, 140], ['C', 2, 3, 180], ['B', 4, 5, 120], ['A', 3, 4, 160],
 ['E', 6, 7, 100]]
Bayaran yang diterima:  558.0
```

TABLE IV. KASUS UJI 4

Job	Durasi (Hari)	Tenggat Waktu (Hari)	Bayaran ($)
A	4	5	130
B	3	4	150

Hasil kasus uji 4:

```
===== HASIL =====

===== Greedy =====
Urutan pekerjaan:
[['B', 3, 4, 150], ['A', 4, 5, 130]]
Bayaran yang diterima:  267.0

===== Branch and Bound =====
Urutan pekerjaan:
[['B', 3, 4, 150], ['A', 4, 5, 130]]
Bayaran yang diterima:  267.0
```

TABLE V. KASUS UJI 5

Job	Durasi (Hari)	Tenggat Waktu (Hari)	Bayaran ($)
A	2	3	180
B	5	7	110
C	3	4	140
D	4	6	120

Hasil kasus uji 5:

```
===== HASIL =====

===== Greedy =====
Urutan pekerjaan:
[['A', 2, 3, 180], ['C', 3, 4, 140], ['D', 4, 6, 120], ['B', 5, 7, 110]]
Bayaran yang diterima:  486.5

===== Branch and Bound =====
Urutan pekerjaan:
[['D', 4, 6, 120], ['C', 3, 4, 140], ['A', 2, 3, 180], ['B', 5, 7, 110]]
Bayaran yang diterima:  493.5
```

B. *Analisis*

Pada setiap kasus pengujian, algoritma *greedy* menunjukkan hasil bayaran yang diterima yang tidak lebih baik daripada solusi yang dihasilkan algoritma *branch and bound*. Hal ini membutkikan bahwa serangkaian langkah dengan kondisi optimum lokal di tiap langkahnya tidak menjamin solusi yang dihasilkan menjadi optimum global. Di sisi lain, algoritma *branch and bound* selalu menunjukkan hasil penjadwalan pekerjaan yang menghasilkan keuntungan atau bayaran maksimal. Bayaran maksimal tersebut diperoleh dengan meminimasi kerugian yang dihasilkan dengan pemberian diskon oleh karena keterlambatan pada suatu pekerjaan. Dengan kata lain, algoritma *branch and bound* pada makalah ini menganggap bahwa memaksimalkan keuntungan ekivalen dengan meminimalkan kerugian pada persoalan penjadwalan *job freelance*.

VI. KESIMPULAN

- Algortima *Branch and Bound* menghasilkan solusi yang lebih optimal ketimbang algoritma *greedy* dalam penyelesaian persoalan penjadwalan job freelance yang memaksimalkan keuntungan.
- Usaha maksimasi keuntungan atau bayaran yang diterima oleh *freelancer* ekivalen dengan usaha minimasi pemberian diskon oleh karena keterlambatan pengerjaan suatu pekerjaan.

VII. UCAPAN TERIMA KASIH

Penulis berterima kasih kepada:

1. Tuhan Yang Maha Esa atas rahmat anugerah-Nya sehingga penulis dapat menyelesaikan makalah ini dengan baik.
2. Bapak Dr. Ir. Rinaldi Munir, M. T., Ibu Dr. Nur Ulfa Maulidevi, S.T., M.Sc, dan Bapak Ir.Rila Mandala, M.Eng.,Ph.D., atas bimbingan dan pengajarannya dalam mata kuliah IF2211 Strategi Algoritma, khususnya kepada Ibu Ulfa sebagai dosen pengajar di K-02 Teknik Informatika ITB.
3. Teman-teman penulis yang telah mendukung penulisan makalah ini.

VIDEO LINK AT YOUTUBE

https://youtu.be/XExeqyoXyTI

REFERENSI

[1] Leite, Bruno Scalia C.F. "The Job-Shop Scheduling Problem: Mixed-Integer Programming Models" https://towardsdatascience.com/the-job-shop-scheduling-problem-mixed-integer-programming-models-4bbee83d16ab. Diakses pada 22 Mei 2023.

[2] Munir, Rinaldi. 2021. "Algoritma *Greedy* (Bagian 1)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf. Diakses pada 22 Mei 2023.

[3] Munir, Rinaldi. 2021. "Algoritma *Greedy* (Bagian 2)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf. Diakses pada 22 Mei 2023.

[4] Munir, Rinaldi. 2022. "Algoritma *Greedy* (Bagian 3)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf. Diakses pada 22 Mei 2023.

[5] Munir, Rinaldi. 2021. "Algoritma *Divide and Conquer* (Bagian 1)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf. Diakses pada 22 Mei 2023.

[6] Munir, Rinaldi. 2021. "Algoritma *Divide and Conquer* (Bagian 2)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian2.pdf. Diakses pada 22 Mei 2023.

[7] Munir, Rinaldi. 2021. "Algoritma *Divide and Conquer* (Bagian 3)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian3.pdf. Diakses pada 22 Mei 2023.

[8] Munir, Rinaldi. 2022. "Algoritma *Divide and Conquer* (Bagian 4)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2021)-Bagian4.pdf. Diakses pada 22 Mei 2023.

PERNYATAAN

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

Bandung, 22 Mei 2023

Melvin Kent Jonathan
13521052