Optimalisasi Penjadwalan Rapat dalam Satu Ruangan dengan Algoritma Greedy untuk Penggunaan Maksimal

Christian Justin Hendrawan - 13522135
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13522135@std.stei.itb.ac.id

***Abstract*— Penjadwalan rapat yang efektif merupakan tantangan umum dalam berbagai organisasi, terutama ketika terbatas pada penggunaan satu ruangan sepanjang hari. Tujuan dari makalah ini adalah untuk mengkaji penggunaan algoritma greedy dalam mengoptimalkan penjadwalan rapat guna memaksimalkan penggunaan ruangan tersebut. Algoritma greedy dipilih karena sifatnya yang sederhana dan efisien dalam menangani masalah optimasi dengan kompleksitas tinggi. Dalam penelitian ini, algoritma greedy diterapkan untuk memilih slot waktu rapat yang memungkinkan penggunaan maksimal dari satu ruangan, memastikan tidak ada tumpang tindih waktu antar rapat. Hasil penelitian menunjukkan bahwa algoritma greedy mampu menyusun jadwal yang efisien dan efektif, memaksimalkan jumlah rapat yang dapat diadakan dalam satu ruangan sepanjang hari.**

Keywords— penjadwalan rapat;algoritma greedy;optimasi; organisasi

I. PENDAHULUAN

Penjadwalan rapat merupakan komponen penting dalam operasional sehari-hari banyak organisasi, mulai dari organisasi mahasiswa hingga instansi pemerintahan. Efektivitas penjadwalan rapat tidak hanya mempengaruhi produktivitas tetapi juga kolaborasi dan pengambilan keputusan yang efektif. Namun, penjadwalan rapat yang melibatkan banyak rapat dalam satu ruangan sering kali menjadi tantangan tersendiri. Kendala seperti jadwal yang padat dan keterbatasan waktu seringkali menghambat tercapainya penggunaan ruangan yang efektif dan efisien.

Masalah penjadwalan ini menjadi semakin kompleks dengan bertambahnya jumlah rapat yang harus dikoordinasikan dalam satu ruangan. Seiring dengan meningkatnya skala dan kompleksitas organisasi, kebutuhan akan solusi penjadwalan yang lebih efisien dan efektif semakin mendesak. Di sinilah peran algoritma optimasi menjadi sangat relevan. Algoritma greedy, yang dikenal dengan pendekatannya yang sederhana dan efisien dalam menyelesaikan berbagai masalah optimasi, menawarkan potensi solusi yang menjanjikan untuk masalah penjadwalan ruangan ini.

Algoritma greedy bekerja dengan prinsip memilih opsi yang paling optimal pada setiap langkah, dengan harapan bahwa solusi lokal ini akan mengarah pada solusi global yang optimal. Meskipun tidak selalu menjamin solusi optimal secara keseluruhan, algoritma ini sering kali memberikan hasil yang memuaskan dalam waktu yang relatif singkat, membuatnya sangat cocok untuk aplikasi praktis seperti penjadwalan ruangan rapat.

Makalah ini bertujuan untuk mengkaji penerapan algoritma greedy dalam konteks penjadwalan rapat, dengan fokus pada memaksimalkan penggunaan satu ruangan sepanjang hari. Penelitian ini diharapkan dapat memberikan wawasan baru dalam pengelolaan penjadwalan rapat yang lebih efektif dan efisien, serta mendorong adopsi teknik optimasi dalam praktik penjadwalan di berbagai organisasi.

II. LANDASAN TEORI

A. Algoritma Greedy

Algoritma Greedy merupakan salah satu pendekatan yang populer dalam pemecahan persoalan optimasi. Persoalan optimasi bertujuan untuk mencari solusi terbaik atau optimal di antara sejumlah solusi yang mungkin berdasarkan kriteria tertentu. Algoritma greedy menyelesaikan persoalan secara langkah demi langkah, di mana pada setiap langkah, kita memilih pilihan yang tampaknya paling menguntungkan pada saat itu, tanpa mempertimbangkan konsekuensi di masa depan. Harapannya adalah dengan memilih solusi lokal yang optimal pada setiap langkah, kita akan mencapai solusi global yang optimal.

Salah satu keuntungan utama algoritma greedy adalah efisiensi komputasinya. Banyak algoritma greedy memiliki kompleksitas waktu yang linear O(n) atau hampir linear O(n log n), membuatnya sangat cepat bahkan untuk input yang besar. Contohnya, algoritma Dijkstra untuk mencari jalur terpendek dalam graf berbobot positif dan algoritma Huffman untuk kompresi data. Keuntungan lainnya adalah

kesederhanaan implementasi, yang mengurangi kemungkinan kesalahan pemrograman.

Penting untuk dicatat bahwa pilihan solusi optimal lokal tidak selalu menghasilkan solusi global yang optimal. Ini dapat terjadi karena algoritma greedy tidak mempertimbangkan semua kemungkinan solusi secara menyeluruh dan atribut yang dipertimbangkan tidak selalu tepat dalam menghasilkan solusi optimal. Untuk beberapa masalah, seperti *Traveling Salesman Problem* (TSP), pendekatan greedy bisa menghasilkan solusi yang jauh dari optimal. Dalam kasus seperti ini, teknik lain seperti pemrograman dinamis atau pencarian lengkap mungkin lebih sesuai, meskipun dengan biaya komputasi yang lebih tinggi.

Oleh karena itu, pemilihan algoritma harus mempertimbangkan *trade-off* antara kecepatan komputasi dan keoptimalan solusi. Dalam praktiknya, algoritma greedy sering digunakan dalam aplikasi dunia nyata di mana keputusan cepat lebih penting daripada keoptimalan absolut. Contohnya dalam penjadwalan pekerjaan atau rapat, routing paket dalam jaringan komputer, atau pengambilan keputusan dalam sistem rekomendasi. Dalam kasus-kasus ini, solusi "cukup baik" yang diperoleh dengan cepat oleh algoritma greedy sering lebih berharga daripada solusi optimal yang memerlukan waktu lama untuk dihitung.

Dalam penerapannya, terdapat beberapa elemen-elemen yang digunakan dalam algoritma greedy[1]:

1. Himpunan kandidat, C : berisi kandidat yang akan dipilih pada setiap langkah. (misal: simpul/sisi di dalam graf, job, task, koin, benda, karakter, dsb)
2. Himpunan solusi, S : berisi kandidat yang sudah dipilih.
3. Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.
4. Fungsi seleksi (selection function): memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
5. Fungsi kelayakan (feasible): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).
6. Fungsi obyektif : memaksimumkan atau meminimumkan.

Dengan menggunakan elemen-elemen di atas, maka dapat dikatakan bahwa Algoritma greedy melibatkan pencarian sebuah himpunan bagian, S, dari himpunan kandidat, C; yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu S menyatakan suatu solusi dan S dioptimisasi oleh fungsi obyektif. [1]

Untuk dapat lebih memahami algoritma greedy, maka Berikut ini adalah skema umum algoritma greedy:

```
function greedy(C : himpunan_kandidat) → himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }
Deklarasi
  x : kandidat
  S : himpunan_solusi

Algoritma:
  S ← {}      { inisialisasi S dengan kosong }
  while (not SOLUSI(S)) and (C ≠ {} ) do
      x ← SELEKSI(C)     { pilih sebuah kandidat dari C}
      C ← C – {x}          { buang x dari C karena sudah dipilih }
      if LAYAK(S ∪ {x}) then      { x memenuhi kelayakan untuk dimasukkan ke dalam himpunan solusi }
          S ← S ∪ {x}      { masukkan x ke dalam himpunan solusi }
      endif
  endwhile
  {SOLUSI(S) or C = {} }

  if SOLUSI(S) then    { solusi sudah lengkap }
    return S
  else
    write('tidak ada solusi')
  endif
```

Gambar 2.1 Skema Umum Algoritma Greedy

(Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf)

B. *Persoalan Pemilihan Aktivitas*

Persoalan pemilihan aktivitas, atau *Activity Selection Problem* (ASP), adalah masalah optimasi yang berkaitan dengan pemilihan sejumlah aktivitas dari himpunan aktivitas yang tersedia sedemikian rupa sehingga tidak ada dua aktivitas yang saling tumpang tindih dan jumlah aktivitas yang dipilih menjadi maksimal. Setiap aktivitas dalam himpunan tersebut memiliki waktu mulai (si) dan waktu selesai (fi) yang telah ditentukan.

Dalam konteks ASP, waktu mulai dan waktu selesai adalah dua parameter kunci yang digunakan untuk menentukan apakah dua aktivitas dapat dijalankan secara bersamaan atau tidak. Tujuan utama dari masalah ini adalah untuk menemukan kombinasi aktivitas yang dapat dilakukan tanpa adanya konflik waktu, dengan cara memaksimalkan jumlah aktivitas yang dipilih.

Masalah pemilihan aktivitas sering muncul dalam berbagai konteks praktis seperti penjadwalan proyek, di mana tugas-tugas harus dijadwalkan untuk memaksimalkan penggunaan sumber daya dan waktu; alokasi sumber daya, di mana sumber daya yang terbatas harus dialokasikan secara efisien di antara berbagai kegiatan; dan penjadwalan ruangan rapat, di mana tujuan utamanya adalah memaksimalkan penggunaan ruangan sepanjang hari tanpa adanya tumpang tindih antara jadwal rapat.

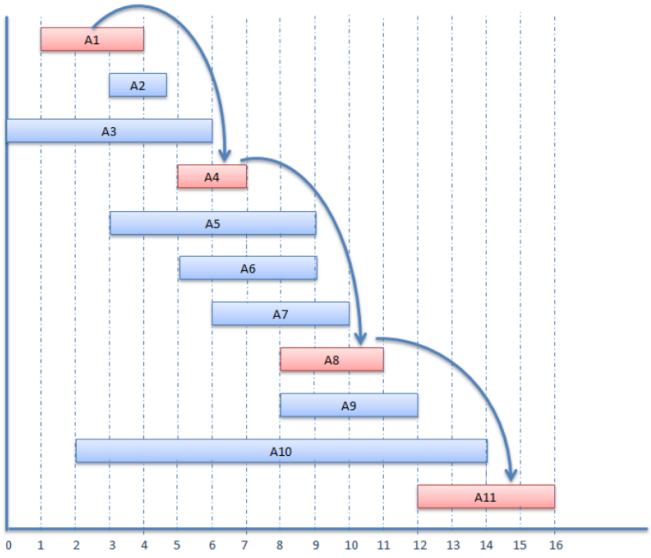
Sebagai contoh, dalam penjadwalan ruangan rapat, setiap rapat memiliki waktu yang tersedia tertentu, dan tugas dari penjadwal adalah menemukan slot waktu yang memungkinkan penggunaan maksimal ruangan tanpa adanya tumpang tindih antara jadwal rapat yang berbeda. Dalam hal ini, ASP menjadi sangat relevan karena dapat membantu mengidentifikasi waktu optimal untuk rapat yang memanfaatkan satu ruangan secara efisien.

Untuk menyelesaikan ASP, biasanya dilakukan pengurutan aktivitas berdasarkan waktu selesai mereka. Hal ini penting karena dengan mengurutkan aktivitas berdasarkan waktu selesai, kita dapat lebih mudah memilih aktivitas yang tidak tumpang tindih dengan aktivitas yang telah dipilih sebelumnya.

Dengan demikian, proses seleksi menjadi lebih terstruktur dan sistematis, memungkinkan untuk mencapai solusi yang efisien dalam memaksimalkan penggunaan ruangan.

Berikut merupakan contoh dari persoalan pemilihan aktivitas:

Start time (si)	finish time (fi)	Activity name
1	4	A1
3	5	A2
0	6	A3
5	7	A4
3	9	A5
5	9	A6
6	10	A7
8	11	A8
8	12	A9
2	14	A10
12	16	A11



Gambar 2.2 Contoh Persoalan Pemilihan Aktivitas

(Sumber: https://scanftree.com/Data_Structure/activity-selection-problem)

Algoritma greedy bekerja dengan prinsip " take what you can get now!" dan membuat keputusan lokal optimal pada setiap langkah dengan harapan solusi tersebut juga optimal secara global. Dalam konteks ASP, algoritma greedy memilih aktivitas berdasarkan urutan waktu selesai yang paling awal. Aktivitas yang selesai lebih awal memungkinkan lebih banyak waktu tersisa untuk memilih aktivitas lainnya, sehingga memaksimalkan jumlah aktivitas yang dapat dipilih.

Untuk memahami bagaimana algoritma greedy bekerja dalam menyelesaikan ASP, berikut disajikan *pseudocode* dari algoritma Greedy-Activity-Selector:

```
function Greedy-Activity-Selector(s1, s2, ..., sn : integer, f1, f2, ..., fn : integer) → set of integer
{ Asumsi: aktivitas sudah diurut terlebih dahulu berdasarkan waktu selesai:  f1 <= f2 <= ... <= fn }
Deklarasi
   i, j, n : integer
   A : set of integer
Algoritma:
   n ← length(s)
   A ← {1}     { aktivitas nomor 1 selalu terpilih }
   j ← 1
   for i ← 2 to n do
      if si ≥ fj then
         A ← A ∪ {i}
         j ← i
      endif
   endif
```

Gambar 2.3 *Pseudocode* Algoritma Greedy untuk ASP

(Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf)

Algoritma ini dimulai dengan memilih aktivitas pertama dalam daftar yang telah diurutkan berdasarkan waktu selesai. Kemudian, secara iteratif, algoritma memilih aktivitas berikutnya yang waktu mulainya tidak tumpang tindih dengan waktu selesai dari aktivitas terakhir yang dipilih. Dengan cara ini, algoritma greedy memastikan bahwa solusi yang dihasilkan adalah serangkaian aktivitas yang tidak tumpang tindih dengan jumlah maksimal.

Dengan menggunakan pendekatan greedy, persoalan pemilihan aktivitas dapat diselesaikan dengan efisien, memberikan solusi optimal dalam waktu komputasi yang relatif singkat. Ini menunjukkan keunggulan algoritma greedy dalam menangani masalah optimasi dengan banyak variabel yang saling bersaing.

C. Penjadwalan Rapat

Penjadwalan rapat adalah kegiatan mengatur alokasi waktu dan ruang untuk serangkaian rapat, dengan tujuan utama untuk mengoptimalkan penggunaan sumber daya yang terbatas seperti ruangan dan waktu yang tersedia. Optimalisasi penjadwalan ini sangat penting untuk memastikan bahwa semua rapat dapat berlangsung dengan lancar, memaksimalkan pemanfaatan ruangan, dan menghindari bentrokan jadwal yang dapat mengganggu efektivitas pertemuan. Penjadwalan yang efektif tidak hanya mempertimbangkan aspek teknis dari ketersediaan ruangan dan waktu, tetapi juga kenyamanan dan kebutuhan partisipan, serta fleksibilitas dalam mengakomodasi perubahan mendadak.

Algoritma greedy adalah salah satu metode yang sering digunakan dalam penjadwalan rapat karena kesederhanaan dan efisiensinya. Dalam proses penjadwalan rapat, algoritma greedy dapat diterapkan dengan mengurutkan rapat berdasarkan kriteria tertentu seperti waktu selesai, banyaknya partisipan, atau prioritas lainnya. Setelah pengurutan, rapat yang memiliki prioritas paling tinggi dan tidak bertabrakan dengan rapat lain yang sudah dijadwalkan akan dipilih. Langkah-langkah ini diulang hingga semua rapat yang memungkinkan sudah dijadwalkan.

Kriteria optimalisasi dalam penjadwalan rapat meliputi memastikan tidak ada bentrokan jadwal serta memaksimalkan penggunaan waktu yang tersedia. Misalnya, mengurutkan rapat berdasarkan waktu selesai (earliest finish time) memungkinkan lebih banyak rapat dijadwalkan dalam satu hari karena rapat yang lebih pendek atau lebih awal selesai akan diprioritaskan terlebih dahulu.

Optimalisasi penjadwalan ini sangat penting dalam manajemen waktu dan sumber daya organisasi. Dengan menerapkan algoritma greedy, proses penjadwalan dapat dilakukan dengan lebih cepat dan sederhana, menghasilkan jadwal yang hampir optimal dalam waktu yang singkat. Hal ini mendukung keberhasilan pelaksanaan berbagai kegiatan dan pertemuan, serta membantu dalam mencapai tujuan organisasi secara keseluruhan.

III. Implementasi dan Pembahasan

A. Transformasi Permasalahan ke Algoritma Greedy

Dalam mengoptimalkan penjadwalan rapat pada satu ruangan menggunakan algoritma greedy, permasalahan ini dimodifikasi agar sesuai dengan karakteristik dan kriteria seleksi yang ditentukan. Perbedaan utama terletak pada pengambilan waktu pelaksanaan rapat yang tidak statis, tidak seperti pada persoalan pemilihan aktivitas pada umumnya. Waktu pelaksanaan rapat biasanya memiliki jam yang pasti seperti 09.00-11.00. Namun pada implementasi penjadwalan yang akan dilakukan, waktu pelaksanaan rapat tidak ditentukan secara pasti. Pada penjadwalan ini, sebuah rapat memiliki waktu ketersediaan dan waktu tidak ketersediaan. Waktu ketersediaan adalah waktu dimana suatu rapat bersedia untuk dijadwalkan dan waktu tidak ketersediaan adalah waktu dimana rapat sudah tidak bersedia untuk dijadwalkan.

Contohnya, rapat dapat memiliki waktu ketersediaan jam 08.00 dan waktu tidak ketersediaan jam 15.00, ini berarti kita dapat menjadwalkan rapat tersebut di antara rentang 08.00-15.00. Untuk suatu rapat sendiri memiliki durasi yang pasti seperti 2 jam. Jadi misalnya suatu rapat memiliki rentang ketersediaan dari jam 08.00-15.00 dan memiliki durasi 2 jam. Program dapat menjadwalkan rapat tersebut jam 10.00-12.00 karena jam tersebut masih terdapat dalam rentang 08.00-15.00. Dengan modifikasi ini, program dapat mengatur waktu mulai dan perkiraan selesai setiap rapat secara dinamis dengan melakukan *update* yang mempertimbangkan durasi rapat yang tetap dan ketersediaan waktu ruangan.

Pentingnya membuat waktu pelaksanaan rapat menjadi dinamis terletak pada fleksibilitas dan efisiensi yang ditawarkan dalam penjadwalan. Dalam dunia nyata, ketersediaan ruang rapat sering kali bervariasi dan tidak dapat diprediksi secara kaku. Dengan memberikan rentang waktu ketersediaan dan tidak ketersediaan, sistem penjadwalan dapat lebih mudah menyesuaikan dengan perubahan dan kebutuhan yang berbeda dari setiap rapat. Hal ini memungkinkan pemanfaatan waktu yang lebih optimal dan menghindari kekosongan atau konflik jadwal yang tidak diinginkan. Dengan demikian, pendekatan dinamis ini meningkatkan efisiensi dan produktivitas organisasi dengan memastikan penggunaan ruang rapat yang maksimal untuk setiap rapat yang diselenggarakan.

Untuk memberikan gambaran yang lebih konkret tentang bagaimana konsep penjadwalan dinamis ini diterapkan, berikut adalah contoh kasus dengan menggunakan tabel:

Nama	Tersedia	Tak Tersedia	Durasi	Perkiraan Selesai
R1	9.00	13.00	3	12.00
R2	11.00	14.00	2	13.00
R3	11.00	14.00	3	14.00
R4	15.00	18.00	2	17.00

a. Solusi Saat ini : []

Perhatikan bahwa R1 memiliki perkiraan selesai terkecil saat ini, maka R1 dimasukkan ke dalam solusi.

Nama	Tersedia	Tak Tersedia	Durasi	Perkiraan Selesai
R2	11.00	14.00	2	13.00
R3	11.00	14.00	3	14.00
R4	15.00	18.00	2	17.00

b. Solusi Saat ini : [R1] dengan waktu perkiraan selesai solusi : 12.00

Perhatikan bahwa waktu tersedia R2 (11.00) lebih awal dari waktu perkiraan selesai solusi sebelumnya (12.00). Dalam pemilihan aktivitas biasa (ASP), R2 akan langsung disingkirkan atau tidak dimasukkan dalam jadwal hari ini. Namun, dengan modifikasi yang kita buat, R2 memiliki rentang waktu tersedia dari 11.00 hingga 14.00. Oleh karena itu, waktu mulai R2 akan diupdate secara dinamis menjadi 12.00, dan waktu perkiraan selesainya juga akan diubah menjadi 12.00 + durasi R2 (2 jam) = 14.00. Hal yang sama berlaku untuk R3, di mana waktu mulai dan waktu perkiraan selesai R3 akan diupdate sesuai dengan penyesuaian ini.

Nama	Tersedia	Tak Tersedia	Durasi	Perkiraan Selesai
R2	12.00	14.00	2	14.00
R3	12.00	14.00	3	15.00
R4	15.00	18.00	2	17.00

c. Solusi Saat ini : [R1] dengan waktu perkiraan selesai solusi : 12.00

Karena waktu perkiraan selesai R2 (14.00) lebih awal dari waktu perkiraan selesai R3 (15.00), maka R2 akan diambil sebagai solusi karena memiliki waktu perkiraan selesai terkecil

saat ini. Selain itu, perhatikan bahwa setelah diupdate, waktu perkiraan selesai pada R3 (15.00) melebihi waktu tak tersedia R3 (14.00). Karena waktu tak tersedia lebih awal dari waktu perkiraan selesai, rapat R3 dapat dicoret dari jadwal.

Nama	Tersedia	Tak Tersedia	Durasi	Perkiraan Selesai
~~R3~~	~~12.00~~	~~14.00~~	~~3~~	~~15.00~~
R4	15.00	18.00	2	17.00

d. Solusi Saat ini : [R1,R2] dengan waktu perkiraan selesai solusi : 14.00.

Perhatikan bahwa R4 adalah rapat terakhir yang tersisa pada tabel, maka R4 akan dimasukkan ke dalam solusi dan R4 dihapus dari tabel.

Nama	Tersedia	Tak Tersedia	Durasi	Perkiraan Selesai

e. Solusi Saat ini : [R1,R2,R4] dengan waktu perkiraan selesai solusi : 17.00.

Jadi solusi yang didapat untuk contoh tersebut adalah ruangan dapat dipakai untuk mengadakan 3 rapat yakni R1,R2, dan R4 dengan waktu perkiraan selesai 17.00.

B. Pemetaan Persoalan

Untuk dapat menyelesaikan persoalan penjadwalan rapat dalam satu ruangan dengan algoritma greedy, persoalan ini harus sebelumnya dipetakan ke elemen-elemen utama dari algoritma greedy. Berikut adalah pemetaannya :

1. Himpunan kandidat (C): Berisi semua rapat yang akan dijadwalkan.
2. Himpunan solusi (S): Akan diisi dengan rapat-rapat yang sudah terjadwal setelah melalui proses seleksi.
3. Fungsi solusi: Mengecek apakah jumlah rapat yang dijadwalkan tersebut sudah maksimal
4. Fungsi seleksi (selection function): Memilih rapat dengan waktu perkiraan selesai nya paling kecil.
5. Fungsi kelayakan: Periksa apakah dengan menambah rapat tersebut membuat ada waktu yang overlapping.
6. Fungsi obyektif : memaksimalkan jumlah rapat.

C. Implementasi Kode

Berikut ini adalah implementasi penjadwalan rapat dalam satu ruangan secara optimal menggunakan bahasa Python. Implementasi ini mengadopsi prinsip algoritma greedy untuk memastikan penggunaan ruangan yang efisien tanpa ada tumpang tindih waktu antar rapat.

Kelas Rapat

```
class Rapat:
    def __init__(self, name, availability, unavailability, duration):
        self.name = name
        self.availability = availability
        self.unavailability = unavailability
        self.duration = duration
        self.estimate_of_complete = self.availability + self.duration
```

Kelas Rapat mendefinisikan suatu rapat atau pertemuan. Setiap rapat memiliki beberapa atribut, yaitu name yang merupakan nama dari rapat tersebut, availability menunjukkan waktu ketersediaan untuk memulai rapat. Misalnya, jika availability adalah jam 10, maka rapat tidak bisa dimulai sebelum jam 10. Ini seperti jendela waktu pembukaan untuk rapat. unavailability menunjukkan batas waktu kapan rapat harus sudah selesai. Misalnya, jika unavailability adalah jam 15, maka rapat harus sudah selesai sebelum atau tepat pada jam 15. Ini seperti jendela waktu penutupan untuk rapat. Dan duration yang merupakan durasi atau lama rapat tersebut berlangsung.

Selain itu, ada juga atribut estimate_of_complete yang merupakan perkiraan waktu kapan rapat tersebut akan selesai. Perkiraan ini dihitung dengan menambahkan waktu ketersediaan (availability) dan durasi (duration) dari rapat tersebut. Dengan demikian, setiap rapat yang dibuat dengan kelas ini akan memiliki informasi lengkap tentang kapan rapat tersebut bisa dimulai, berapa lama rapat tersebut berlangsung, dan kapan rapat tersebut diperkirakan akan selesai. Informasi ini sangat penting untuk merencanakan dan mengatur jadwal rapat dengan efisien.

Implementasi Algoritma Greedy untuk Penjadwalan

```
def schedule_rapats(rapats):
    # Inisialisasi daftar untuk menyimpan rapat yang telah dijadwalkan
    scheduled_rapats = []
    # Inisialisasi variabel untuk menyimpan waktu selesai terakhir dari rapat yang telah dijadwalkan
    max_complete = 0

    while rapats:
        # Temukan rapat dengan waktu selesai perkiraan terawal yang dapat diselesaikan sebelum menjadi tidak tersedia
        rapat = min((e for e in rapats if max(e.availability, max_complete) + e.duration <= e.unavailability),
```

```
                    key=lambda e: max(e.availability,
max_complete) + e.duration, default=None)

        # Jika tidak ada rapat seperti itu ditemukan,
hentikan loop
        if rapat is None:
            break

        # Perbarui waktu ketersediaan dan perkiraan
selesai dari rapat
        rapat.availability = max(rapat.availability,
max_complete)
        rapat.estimate_of_complete = rapat.availability +
rapat.duration

        # Perbarui max_complete
        max_complete = rapat.estimate_of_complete

        # Jadwalkan rapat dan hapus dari daftar rapat
        scheduled_rapats.append(rapat)
        rapats.remove(rapat)

    return scheduled_rapats
```

Fungsi schedule_rapats dirancang untuk mengoptimalkan penjadwalan rapat berdasarkan ketersediaan, durasi, dan batas waktu ketidaktersediaan. Fungsi ini menggunakan pendekatan algoritma greedy, yang berusaha untuk memaksimalkan jumlah rapat yang dapat dijadwalkan dengan memprioritaskan rapat yang dapat diselesaikan paling awal. Berikut adalah penjelasan rinci dari setiap langkah dalam fungsi ini:

1. Fungsi schedule_rapats dimulai dengan menginisialisasi daftar kosong scheduled_rapats untuk menyimpan rapat yang telah dijadwalkan, dan variabel max_complete untuk melacak waktu selesai terakhir dari rapat yang telah dijadwalkan.
2. Fungsi kemudian memasuki loop while, yang berlanjut selama masih ada rapat dalam daftar rapats yang belum dijadwalkan.
3. Dalam setiap iterasi loop, fungsi mencari rapat dengan waktu selesai terawal yang mungkin yang dapat diselesaikan sebelum waktunya tidak tersedia. Ini dicapai menggunakan fungsi min dengan ekspresi generator. Ekspresi generator ini mengiterasi setiap rapat e dalam rapats dan memeriksa apakah maksimum dari e.availability dan max_complete ditambah e.duration kurang dari atau sama dengan e.unavailability. Fungsi key untuk fungsi min menghitung waktu selesai dari setiap rapat berdasarkan maksimum dari e.availability dan max_complete ditambah e.duration.
4. Jika tidak ada rapat seperti itu yang ditemukan (yaitu, rapat adalah None), loop dihentikan dan fungsi melanjutkan ke pernyataan return.
5. Jika rapat yang sesuai ditemukan, atribut availability dari rapat diperbarui menjadi maksimum dari rapat.availability dan max_complete. Atribut estimate_of_complete dari rapat kemudian dihitung ulang berdasarkan rapat.availability yang telah diperbarui ditambah rapat.duration.
6. Variabel max_complete kemudian diperbarui menjadi estimate_of_complete dari rapat yang dijadwalkan rapat.
7. Rapat yang dijadwalkan rapat kemudian ditambahkan ke daftar scheduled_rapats dan dihapus dari daftar rapats.
8. Proses ini diulangi sampai semua rapat dalam rapats telah dijadwalkan atau tidak ada lagi rapat yang bisa dijadwalkan.
9. Akhirnya, fungsi mengembalikan daftar scheduled_rapats, yang berisi semua rapat yang telah berhasil dijadwalkan.

D. Pengujian

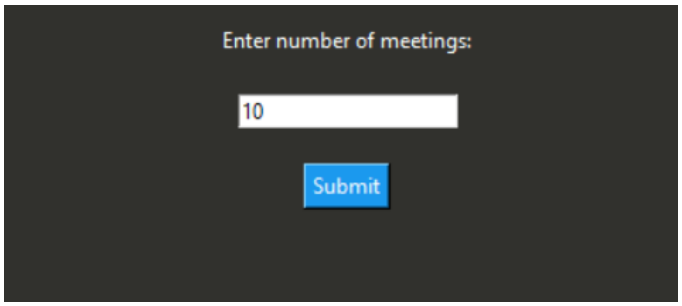
Pada bagian ini, akan dilakukan pengujian terhadap implementasi algoritma greedy untuk penjadwalan rapat menggunakan contoh persoalan dengan 10 rapat. Rentang waktu ketersediaan untuk memulai rapat berkisar antara pukul 09.00 hingga 14.00, sedangkan rentang waktu tidak ketersediaan (batas waktu kapan rapat sudah tidak dapat dijalankan) berkisar antara pukul 19.00 hingga 22.00. Durasi setiap rapat berkisar antara 2 hingga 6 jam. Berikut adalah contoh data persoalan yang digunakan:

Tabel 3.1 Tabel Data Test Case 1

Nama	Tersedia	Tak Tersedia	Durasi	Perkiraan Selesai
A	10	20	3	13
B	9	21	4	13
C	11	19	2	13
D	12	22	5	17
E	14	20	3	17
F	13	21	4	17
G	10	19	2	12
H	11	22	3	14
I	9	20	4	13
J	12	21	2	14

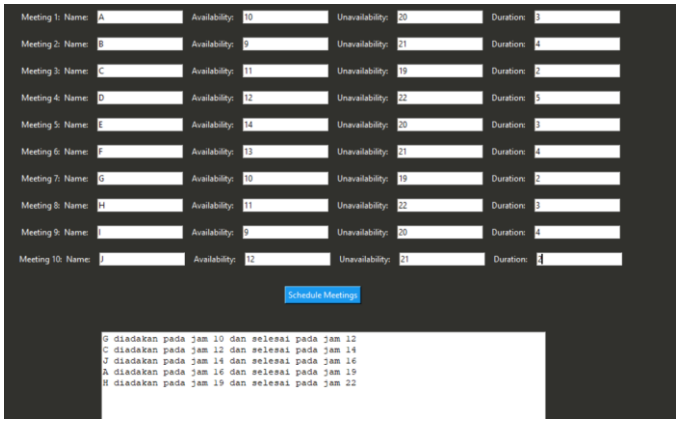
Langkah berikutnya adalah menjalankan algoritma greedy yang telah diimplementasikan dengan data di atas untuk mendapatkan jadwal rapat yang optimal. Hasil dari pengujian

ini akan menunjukkan urutan dan waktu pelaksanaan rapat yang paling efisien dan tidak ada dua jadwal rapat yang tumpang tindih.



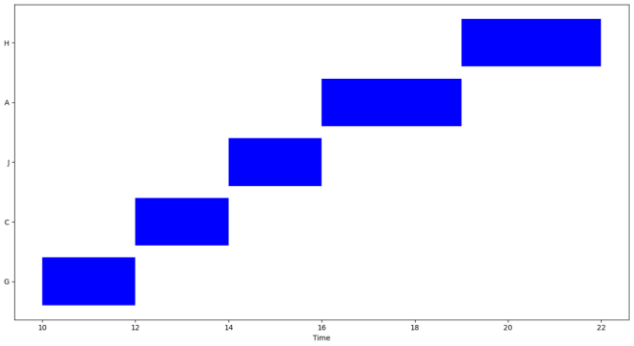
Gambar 3.1 Tampilan GUI Program Input Jumlah Rapat

(Sumber: Penulis)



Gambar 3.2 Tampilan GUI Program Menampilkan Solusi Testcase 1

(Sumber: Penulis)



Gambar 3.3 Tampilan Visualisasi Solusi Testcase 1

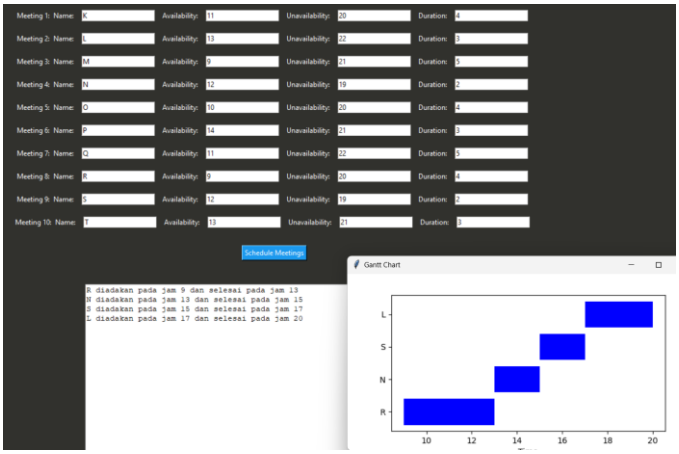
(Sumber: Penulis)

Jadi pada test case 1, jumlah rapat yang diadakan maksimal adalah 5 rapat, dengan alurnya rapat G diadakan pada jam 10 dan selesai pada jam 12, lalu rapat C diadakan pada jam 12 dan selesai pada jam 14, lalu rapat J diadakan pada jam 14 dan selesai pada jam 16, lalu rapat A diadakan pada jam 16 dan selesai pada jam 19, lalu rapat H diadakan pada jam 19 dan selesai pada jam 22.

Tabel 3.2 Tabel Data Test Case 2

Nama	Tersedia	Tak Tersedia	Durasi	Perkiraan Selesai
K	11	20	4	15
L	13	22	3	16
M	9	21	5	14
N	12	19	2	14
O	10	20	4	14
P	14	21	3	17
Q	11	22	5	16
R	9	20	4	13
S	12	19	2	14
T	13	21	3	16

Hasil dari program adalah sebagai berikut.



Gambar 3.3 Tampilan GUI Program Menampilkan Solusi Testcase 1

(Sumber: Penulis)

Pada test case 2, jumlah rapat yang diambil berjumlah 4, dengan alurnya rapat R diadakan pada jam 9 dan selesai pada jam 13, lalu rapat N diadakan pada jam 13 dan selesai pada jam 15, lalu rapat S diadakan pada jam 15 dan selesai pada jam 17, lalu rapat L diadakan pada jam 17 dan selesai pada jam 20.

Berdasarkan hasil pengujian yang dilakukan pada dua set data, dapat disimpulkan bahwa solusi penjadwalan rapat menggunakan algoritma greedy yang diusulkan telah mencapai tingkat optimalisasi yang sangat baik.

Pada set data pertama, solusi ini berhasil memaksimalkan pemanfaatan waktu yang tersedia dengan meminimalkan waktu terbuang, hanya menyisakan selang waktu 1 jam yang tidak dapat digunakan untuk rapat. Selain itu, solusi ini juga berhasil memaksimalkan jumlah rapat yang dapat dijadwalkan, yakni sebanyak 5 rapat dari total 10 rapat yang tersedia, tanpa adanya

tumpang tindih jadwal maupun pelanggaran terhadap kendala durasi rapat.

Pada set data kedua, meskipun solusi ini berhasil menjadwalkan 4 rapat secara efisien tanpa adanya tumpang tindih jadwal, terdapat selang waktu 2 jam yang tidak dapat digunakan atau terbuang. Namun, jumlah waktu terbuang ini masih cukup minimal jika dibandingkan dengan keseluruhan waktu yang tersedia.

Dengan hasil pengujian ini, dapat disimpulkan bahwa algoritma greedy yang diimplementasikan mampu menghasilkan solusi penjadwalan yang optimal dalam memanfaatkan waktu dengan sangat efisien. Meskipun terdapat sedikit waktu terbuang pada beberapa kasus, namun hal ini tidak signifikan jika dibandingkan dengan keberhasilan dalam memaksimalkan jumlah rapat yang dapat dijadwalkan dan memenuhi seluruh kendala yang ada. Oleh karena itu, algoritma greedy terbukti menjadi metode yang andal dan efektif dalam menyelesaikan permasalahan penjadwalan rapat pada satu ruangan dengan berbagai batasan waktu ketersediaan dan durasi rapat.

IV. Kesimpulan

Algoritma greedy terbukti menjadi metode yang efektif dan efisien dalam mengoptimalkan penjadwalan rapat pada satu ruangan. Dengan pendekatan sederhana yang memilih solusi lokal terbaik pada setiap langkah, algoritma ini mampu menghasilkan solusi penjadwalan yang optimal dengan memanfaatkan waktu secara maksimal dan menghindari tumpang tindih jadwal rapat.

Penerapan algoritma greedy dalam masalah penjadwalan rapat melibatkan transformasi permasalahan ke dalam elemen-elemen yang sesuai, seperti himpunan kandidat (daftar rapat), himpunan solusi (jadwal rapat), fungsi seleksi (memilih rapat dengan waktu selesai terkecil), dan fungsi kelayakan (memastikan tidak ada tumpang tindih waktu). Modifikasi penting yang dilakukan adalah membuat waktu pelaksanaan rapat menjadi dinamis dengan menggunakan rentang waktu ketersediaan dan ketidaktersediaan, memberikan fleksibilitas yang lebih besar dalam penjadwalan.

Implementasi dalam bahasa Python dan pengujian dengan contoh data menunjukkan bahwa solusi yang dihasilkan algoritma greedy berhasil memaksimalkan jumlah rapat yang dapat dijadwalkan dalam satu ruangan sepanjang hari, dengan waktu terbuang yang minimal. Meskipun terdapat sedikit waktu terbuang pada beberapa kasus, namun hal ini tidak signifikan jika dibandingkan dengan keberhasilan dalam memenuhi seluruh kendala dan memaksimalkan penggunaan ruangan.

Secara keseluruhan, algoritma greedy terbukti menjadi pilihan yang tepat dan efisien untuk menyelesaikan permasalahan penjadwalan rapat dalam satu ruangan, dengan kemampuannya untuk menghasilkan solusi optimal dalam waktu komputasi yang relatif singkat. Pendekatan ini dapat memberikan manfaat nyata bagi berbagai organisasi dalam mengoptimalkan pengelolaan waktu dan sumber daya mereka.

Repository dan Video Link Youtube

Link Youtube : https://youtu.be/X4AbmyL2c14

Link Github : https://github.com/ChrisCS50X/Aplikasi_Greedy_13522135

Ucapan Terima Kasih

Penulis panjatkan puji dan syukur kepada Tuhan yang Maha Esa atas berkat dan rahmat-Nya, makalah Strategi Algoritma berjudul ”Optimalisasi Penjadwalan Rapat dalam Satu Ruangan dengan Algoritma Greedy untuk Penggunaan Maksimal” ini dapat diselesaikan dengan tepat waktu. Terima kasih sebesar-besarnya juga kepada Pak Ir. Rila Mandala, M.Eng., Ph.D. dan Monterico Adrian, S.T., M.T. selaku dosen pengampuh mata kuliah Strategi Algoritma atas jasa beliau yang sudah mengajarkan penulis agar dapat melancarkan pengerjaan makalah ini. Terima kasih juga sebesar-besarnya kepada seluruh keluarga dan teman-teman penulis yang sudah membantu dan menyemangati dalam pengerjaan makalah ini. Ucapan terima kasih juga ingin penulis sampaikan kepada diri sendiri karena sudah mampu mengeluarkan seluruh tenaga, waktu, dan hal lainnya dalam pengerjaan makalah ini.

Daftar Pustaka

[1] Munir, R. (2021) Bahan Kuliah IF2211 strategi algoritma algoritma greedy bagian (1). Available at: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf (Diakses pada 9 Juni 2024)

[2] Munir, R. (2021) Bahan Kuliah IF2211 strategi algoritma algoritma greedy bagian (2). Available at: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf (Diakses pada 9 Juni 2024)

[3] Munir, R. (2021) Bahan Kuliah IF2211 strategi algoritma algoritma greedy bagian (3). Available at: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf (Diakses pada 9 Juni 2024)

[4] https://www.geeksforgeeks.org/activity-selection-problem-greedy-algo-1/ (Diakses pada 10 Juni 2024)

[5] https://web.stanford.edu/class/archive/cs/cs161/cs161.1166/lectures/lecture14.pdf (Diakses pada 11 Juni 2024)

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, 12 Juni 2024

Christian Justin Hendrawan - 13522135