Penerapan Algoritma Greedy pada Perancangan Jadwal Kuliah yang Optimal dengan Memperhatikan Preferensi Dosen

Bernardus Willson - 13521021
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521021@std.stei.itb.ac.id

Penelitian ini membahas perancangan jadwal kuliah yang optimal dengan mempertimbangkan preferensi dosen. Menggunakan pendekatan algoritma greedy, tujuan utamanya adalah menghasilkan jadwal kuliah yang memenuhi preferensi dosen dengan meminimalkan tumpang tindih waktu antara mata kuliah. Dalam penelitian ini, juga diperhatikan efisiensi komputasional untuk mengatasi kompleksitas perancangan jadwal kuliah. Hasilnya diharapkan dapat membantu pengelola sistem pendidikan dalam menyusun jadwal kuliah yang efisien dan memenuhi preferensi dosen, serta meningkatkan pengalaman belajar mahasiswa.

Kata Kunci—perancangan jadwal kuliah, optimal, preferensi dosen, algoritma greedy.

I. PENDAHULUAN

Perancangan jadwal kuliah yang optimal merupakan tantangan yang kompleks dalam pengelolaan sistem pendidikan. Penjadwalan mata kuliah yang efisien dan memenuhi preferensi dosen merupakan salah satu aspek penting dalam menciptakan pengalaman belajar yang baik bagi mahasiswa. Dalam konteks ini, penerapan algoritma greedy menjadi salah satu pendekatan yang dapat digunakan untuk menghasilkan jadwal kuliah yang memenuhi preferensi dosen dengan meminimalkan tumpang tindih waktu antara mata kuliah.

Penelitian ini berfokus pada perancangan jadwal kuliah yang optimal dengan mempertimbangkan preferensi dosen. Preferensi dosen dapat meliputi waktu yang diinginkan untuk mengajar, preferensi hari dan jam kuliah, serta kebutuhan khusus lainnya. Tujuan utama dari penelitian ini adalah mengembangkan algoritma yang dapat menghasilkan jadwal kuliah yang meminimalkan tumpang tindih waktu antara mata kuliah, sehingga dosen dapat mengajar dengan efisien dan menghindari bentrok jadwal yang tidak diinginkan.

Algoritma greedy adalah pendekatan yang diadopsi dalam penelitian ini. Algoritma ini bekerja dengan memilih langkah terbaik pada setiap tahap, tanpa mempertimbangkan konsekuensi jangka panjang. Dalam konteks perancangan jadwal kuliah, algoritma greedy akan memilih slot terbaik untuk setiap mata kuliah berdasarkan preferensi dosen, dengan memperhatikan tumpang tindih waktu antara mata kuliah yang sudah dipilih sebelumnya.

Selain itu, dalam penelitian ini juga diperhatikan aspek efisiensi komputasional. Dalam perancangan jadwal kuliah yang kompleks, terdapat banyak kemungkinan kombinasi jadwal yang harus dieksplorasi. Oleh karena itu, penggunaan algoritma yang efisien secara komputasional menjadi hal yang krusial untuk menghasilkan jadwal kuliah yang optimal dalam waktu yang wajar.

Melalui penelitian ini, diharapkan dapat ditemukan solusi yang efektif dan efisien dalam perancangan jadwal kuliah yang optimal dengan mempertimbangkan preferensi dosen. Solusi tersebut dapat memberikan manfaat besar bagi pengelola sistem pendidikan dan dosen dalam mengatur jadwal kuliah yang efisien, mengurangi tumpang tindih waktu, dan meningkatkan pengalaman belajar bagi mahasiswa.

II. LANDASAN TEORI

A. Algoritma Greedy

Dikutip dari "Algoritma Greedy (Bagian 1) - Rinaldi Munir", algoritma greedy adalah sebuah pendekatan dalam pemecahan persoalan yang mengikuti prinsip "take what you can get now!" Di dalam algoritma greedy, kita memecahkan persoalan secara langkah per langkah (step by step), di mana pada setiap langkah kita mengambil pilihan yang dianggap terbaik pada saat itu tanpa memperhatikan konsekuensi ke depan. Prinsip ini didasarkan pada keyakinan bahwa dengan memilih solusi optimum lokal pada setiap langkah, kita akan mencapai solusi optimum global secara keseluruhan.

Konsep dasar algoritma greedy sangat sederhana. Pada setiap langkah, kita mencari alternatif pilihan yang tersedia dan memilih yang dianggap paling menguntungkan berdasarkan suatu kriteria atau aturan tertentu. Kriteria ini dapat berupa nilai tertinggi, biaya terendah, jarak terdekat, atau parameter lain yang relevan dengan persoalan yang sedang kita hadapi.

Namun, perlu dicatat bahwa dalam algoritma greedy, kita tidak melakukan evaluasi atau pemilihan berdasarkan konsekuensi ke depan atau dampak jangka panjang. Hal ini berarti kita tidak melihat secara keseluruhan bagaimana pemilihan pada setiap langkah akan mempengaruhi solusi secara keseluruhan. Algoritma ini beroperasi dengan asumsi bahwa pemilihan yang optimal pada setiap langkah akan menghasilkan solusi yang optimal secara keseluruhan. Oleh karena itu, dalam algoritma greedy, kita "berharap" bahwa dengan mengambil pilihan optimum lokal pada setiap langkah, kita akan mencapai solusi optimum global pada akhirnya.

Pada dasarnya, algoritma greedy sangat efisien dan mudah untuk diimplementasikan. Namun, kelemahannya terletak pada kecenderungannya menghasilkan solusi yang suboptimal atau tidak optimal secara global. Dalam beberapa kasus, solusi yang dihasilkan oleh algoritma greedy mungkin tidak mencapai solusi yang optimal secara keseluruhan. Oleh karena itu, pemilihan aturan atau kriteria dalam algoritma greedy harus dipilih dengan hati-hati, dan analisis lebih lanjut diperlukan untuk memastikan bahwa solusi yang dihasilkan memenuhi kebutuhan dan persyaratan yang diberikan.

Meskipun demikian, algoritma greedy tetap menjadi alat yang berguna dalam pemecahan berbagai persoalan yang membutuhkan pengambilan keputusan bertingkat. Dalam banyak kasus, algoritma greedy dapat memberikan solusi yang memadai dengan efisiensi yang tinggi. Namun, perlu diingat bahwa untuk persoalan yang lebih kompleks dan membutuhkan optimisasi yang lebih baik, algoritma lain yang lebih canggih dan kompleks mungkin diperlukan.

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

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.

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
```

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

Pada akhir setiap lelaran (iterasi), solusi yang terbentuk adalah optimum local. Pada akhir kalang while-do diperoleh optimum global (jika ada). Optimum global belum tentu merupakan solusi optimum (terbaik), bisa jadi merupakan solusi sub-optimum atau pseudo-optimum, dengan alasan:

1. Algoritma greedy tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada (sebagaimana pada metode exhaustive search).

2. Terdapat beberapa fungsi seleksi yang berbeda, sehingga kita harus memilih fungsi yang tepat jika kita ingin algoritma menghasilkan solusi optiamal.

Jadi, pada sebagian persoalan, algoritma greedy tidak selalu berhasil memberikan solusi yang optimal, namun sub-optimal.

Jika solusi terbaik mutlak tidak terlalu diperlukan, maka algoritma greedy dapat digunakan untuk menghasilkan solusi hampiran (approximation), daripada menggunakan algoritma yang kebutuhan waktunya eksponensial untuk menghasilkan solusi yang eksak. Misalnya mencari tur dengan bobot minimal pada persoalan TSP untuk jumlah simpul (n) yang banyak dengan algoritma brute force dibutuhkan waktu komputasi yang lama untuk menemukannya. Dengan algoritma greedy, meskipun tur dengan berbobot minimal tidak dapat ditemukan, namun solusi dengan algoritma greedy dianggap sebagai hampiran solusi optimal. Namun bila algoritma greedy dapat menghasilkan solusi optimal, maka keoptimalannya itu harus dapat dibuktikan secara matematis. Membuktikan optimalitas algoritma greedy secara matematis adalah tantangan tersendiri. Lebih mudah memperlihatkan algoritma greedy tidak selalu optimal dengan menunjukkan counter example (contoh kasus yang menunjukkan solusi yang diperoleh tidak optimal).

Contoh-contoh persoalan yang diselesaikan dengan Algoritma Greedy adalah Persoalan penukaran uang (coin exchange problem), Persoalan memilih aktivitas (activity selection problem), Minimisasi waktu di dalam system, Persoalan knapsack (knapsack problem), Penjadwalan Job dengan tenggat waktu (job schedulling with deadlines), Pohon

merentang minimum (minimum spanning tree), Lintasan terpendek (shortest path), Kode Huffman (Huffman code), Pecahan Mesir (Egyptian fraction), dan masih banyak lagi.

B. Graf Jadwal Kuliah

Graf jadwal kuliah adalah sebuah konsep yang digunakan untuk merepresentasikan hubungan antara mata kuliah, dosen, dan slot waktu dalam perancangan jadwal kuliah. Dalam graf jadwal kuliah, setiap mata kuliah direpresentasikan sebagai simpul (node) dalam graf, sedangkan hubungan antara mata kuliah tersebut ditunjukkan oleh tepi (edge) antara simpul-simpul tersebut. Graf ini memungkinkan pengelolaan yang efisien terhadap jadwal kuliah dengan memperhatikan keterkaitan antara mata kuliah, dosen yang mengampu, dan slot waktu yang tersedia.

Representasi graf dalam perancangan jadwal kuliah memungkinkan pengorganisasian yang lebih terstruktur dan pemetaan yang jelas antara setiap mata kuliah dengan dosen yang mengampunya serta slot waktu yang tersedia. Dalam graf jadwal kuliah, setiap simpul mata kuliah mengandung informasi tentang mata kuliah itu sendiri, seperti nama mata kuliah dan persyaratan atau preferensi khusus yang terkait dengan mata kuliah tersebut.

Tepi antara simpul-simpul mata kuliah menggambarkan hubungan dan keterkaitan antara mata kuliah-mata kuliah tersebut. Misalnya, sebuah tepi dapat menghubungkan dua mata kuliah yang memiliki persyaratan prasyarat, yang menunjukkan bahwa satu mata kuliah harus diambil sebelum mata kuliah lainnya. Tepi juga dapat menghubungkan mata kuliah dengan dosen yang mengampunya, menunjukkan hubungan antara mata kuliah dan pengajar yang bertanggung jawab.

Selain itu, tepi dalam graf jadwal kuliah juga digunakan untuk menggambarkan hubungan antara mata kuliah dan slot waktu yang tersedia. Tepi ini mencerminkan keterkaitan antara mata kuliah dengan jadwal waktu yang dapat digunakan untuk mengampu mata kuliah tersebut. Dengan adanya representasi ini, graf jadwal kuliah memungkinkan pengelolaan yang lebih efisien dalam menentukan alokasi waktu yang tepat untuk setiap mata kuliah, menghindari bentrok jadwal, dan memastikan bahwa semua mata kuliah dapat diampu sesuai dengan preferensi dan ketersediaan dosen.

Setiap simpul mata kuliah akan terhubung dengan simpul slot waktu yang mungkin untuk mata kuliah tersebut. Hubungan ini menunjukkan bahwa mata kuliah tersebut dapat dijadwalkan pada slot waktu yang terhubung. Misalnya, mata kuliah "Matematika" dapat terhubung dengan slot waktu "Senin 10:00-12:00", "Selasa 10:00-12:00", dan "Rabu 13:00-15:00" karena mata kuliah tersebut tersedia pada slot-slot waktu tersebut.

Dengan menggunakan konsep dan representasi graf jadwal kuliah, perancangan jadwal kuliah dapat dilakukan secara sistematis dan terstruktur. Algoritma-algoritma graf juga dapat diterapkan untuk menyelesaikan persoalan-persoalan yang terkait dengan jadwal kuliah, seperti penentuan jadwal optimal yang memenuhi preferensi dosen dan persyaratan mata kuliah. Graf jadwal kuliah memberikan visualisasi yang jelas tentang keterkaitan antara entitas-entitas dalam jadwal kuliah, sehingga memudahkan pengelolaan, pemetaan, dan pengoptimalan dalam perancangan jadwal kuliah yang efisien dan efektif.

C. Preferensi Dosen

Dalam perancangan jadwal kuliah yang optimal, penting untuk mempertimbangkan preferensi dosen. Preferensi dosen memiliki peran yang signifikan dalam menentukan keberhasilan pelaksanaan mata kuliah dan kualitas pengajaran. Dengan memperhatikan preferensi dosen, institusi pendidikan dapat menciptakan lingkungan yang lebih baik bagi dosen dalam melaksanakan tugas mengajar mereka.

Salah satu faktor yang mempengaruhi preferensi dosen adalah preferensi waktu. Setiap dosen memiliki preferensi waktu tertentu ketika mereka merasa lebih nyaman dan produktif dalam memberikan materi kuliah. Beberapa dosen mungkin lebih efektif pada pagi hari, sementara yang lain lebih baik pada siang atau sore hari. Dalam perancangan jadwal kuliah, memperhatikan preferensi waktu ini dapat membantu dosen dalam memberikan pengajaran terbaik mereka. Dosen yang dapat mengajar pada waktu yang mereka sukai cenderung lebih termotivasi dan energik dalam menyampaikan materi kuliah kepada mahasiswa.

Selain itu, jumlah jam mengajar juga menjadi faktor penting dalam preferensi dosen. Setiap dosen memiliki batasan waktu dan energi yang mereka dapatkan untuk mengajar. Beberapa dosen mungkin lebih cocok dengan jadwal mengajar yang padat, sementara yang lain membutuhkan waktu lebih luang di antara sesi mengajar. Dalam merancang jadwal kuliah, penting untuk mempertimbangkan jumlah jam mengajar yang sesuai dengan preferensi dosen. Dengan memberikan jadwal yang memperhatikan kebutuhan dosen, institusi pendidikan dapat membantu menjaga keseimbangan antara beban kerja dosen dan kualitas pengajaran yang dihasilkan.

Selain preferensi waktu dan jumlah jam mengajar, preferensi dosen terhadap hari tertentu juga perlu dipertimbangkan. Beberapa dosen mungkin memiliki preferensi terhadap hari-hari tertentu di mana mereka merasa lebih nyaman atau memiliki ketersediaan yang lebih baik. Misalnya, seorang dosen mungkin lebih memilih untuk tidak mengajar pada hari Jumat karena mereka menggunakan waktu tersebut untuk kegiatan penelitian atau kegiatan lainnya. Memperhatikan preferensi terhadap hari tertentu dapat membantu menciptakan jadwal yang sesuai dengan preferensi dosen dan memungkinkan mereka untuk memiliki keseimbangan yang baik antara tugas mengajar dan tugas lainnya.

Dalam keseluruhan, mempertimbangkan preferensi dosen dalam perancangan jadwal kuliah merupakan langkah penting untuk menciptakan lingkungan yang kondusif bagi dosen dan meningkatkan kualitas pengajaran. Dosen yang merasa dihargai dan diperhatikan preferensinya cenderung memberikan pengajaran yang lebih baik dan lebih bermakna bagi mahasiswa. Oleh karena itu, institusi pendidikan perlu mengumpulkan preferensi dosen secara sistematis dan

memadukan faktor-faktor ini dalam proses perancangan jadwal kuliah yang optimal. Dengan cara ini, kepuasan dosen dapat ditingkatkan, serta efektivitas dan efisiensi dalam pengajaran dapat dicapai.

D. Tumpang Tindih Waktu

Dalam perancangan jadwal kuliah, masalah yang sering dihadapi adalah tumpang tindih waktu antara mata kuliah yang berbeda. Tumpang tindih waktu terjadi ketika terdapat dua atau lebih mata kuliah yang memiliki jadwal yang bertabrakan, artinya mereka dijadwalkan pada waktu yang sama atau overlap. Fenomena ini dapat menimbulkan berbagai dampak negatif, terutama bagi mahasiswa yang harus menghadiri lebih dari satu mata kuliah yang bertabrakan.

Salah satu dampak negatif dari tumpang tindih waktu adalah kesulitan bagi mahasiswa dalam menghadiri lebih dari satu mata kuliah yang bertabrakan. Ketika dua mata kuliah dijadwalkan pada waktu yang sama, mahasiswa terpaksa harus memilih salah satu mata kuliah dan melewatkan yang lain. Hal ini dapat menghambat kemajuan akademik mereka, terutama jika mata kuliah yang bertabrakan merupakan mata kuliah yang penting atau prasyarat untuk mata kuliah berikutnya. Selain itu, mahasiswa juga dapat merasa tertekan dan tidak nyaman karena harus membuat keputusan sulit dalam memilih mata kuliah mana yang akan dihadiri dan mana yang harus dilewatkan.

Tumpang tindih waktu juga dapat menghambat fleksibilitas jadwal mahasiswa. Ketika terdapat banyak mata kuliah yang bertabrakan, pilihan jadwal yang tersedia untuk mahasiswa menjadi terbatas. Hal ini dapat menghambat kemampuan mahasiswa untuk mengatur jadwal mereka secara efektif, mengakibatkan kesulitan dalam menyesuaikan dengan kegiatan lain seperti pekerjaan paruh waktu, kegiatan ekstrakurikuler, atau komitmen pribadi lainnya. Ketidakfleksibelan jadwal dapat meningkatkan tingkat stres dan mengurangi kualitas hidup mahasiswa.

Dalam konteks perancangan jadwal kuliah, penting untuk menghindari atau meminimalkan tumpang tindih waktu sebanyak mungkin. Hal ini dapat dilakukan dengan memperhatikan faktor-faktor seperti ketersediaan ruang kuliah yang cukup, kebutuhan dosen, preferensi mahasiswa, dan batasan waktu yang diberikan. Penggunaan algoritma dan pendekatan yang efisien dalam perancangan jadwal kuliah dapat membantu mengurangi risiko tumpang tindih waktu. Dengan menghasilkan jadwal yang optimal dan bebas tumpang tindih waktu, mahasiswa dapat menghadiri semua mata kuliah yang mereka butuhkan tanpa hambatan.

Dalam kesimpulan, tumpang tindih waktu dalam perancangan jadwal kuliah adalah masalah yang sering dihadapi dan memiliki dampak negatif yang signifikan. Kesulitan bagi mahasiswa untuk menghadiri lebih dari satu mata kuliah yang bertabrakan dapat menghambat kemajuan akademik dan mengurangi fleksibilitas jadwal mereka. Oleh karena itu, perancangan jadwal kuliah yang efektif dan memperhatikan faktor-faktor yang relevan, seperti preferensi dosen dan mahasiswa, menjadi penting dalam mengurangi tumpang tindih waktu. Dengan demikian, institusi pendidikan dapat menciptakan lingkungan yang mendukung kemajuan akademik dan kesejahteraan mahasiswa.

E. Optimasi Jadwal Kuliah

Optimisasi jadwal kuliah adalah konsep yang penting dalam perancangan jadwal untuk mencapai jadwal yang efisien dan menguntungkan. Tujuan utama dari optimisasi jadwal kuliah adalah mencari solusi terbaik yang memenuhi berbagai kriteria, seperti meminimalkan tumpang tindih waktu antara mata kuliah, memaksimalkan preferensi dosen, atau mengoptimalkan penggunaan sumber daya yang tersedia.

Salah satu tujuan optimisasi jadwal kuliah adalah meminimalkan tumpang tindih waktu antara mata kuliah. Tumpang tindih waktu dapat menyebabkan kesulitan bagi mahasiswa untuk menghadiri lebih dari satu mata kuliah yang bertabrakan. Dengan menggunakan teknik optimisasi, perancang jadwal dapat mencari solusi yang meminimalkan jumlah mata kuliah yang bertabrakan, sehingga memungkinkan mahasiswa untuk menghadiri semua mata kuliah yang diperlukan tanpa adanya konflik waktu.

Selain itu, optimisasi jadwal kuliah juga dapat memaksimalkan preferensi dosen. Setiap dosen mungkin memiliki preferensi terhadap hari atau waktu tertentu untuk mengajar. Dengan mempertimbangkan preferensi ini, perancang jadwal dapat menyusun jadwal yang memenuhi preferensi dosen secara maksimal. Hal ini dapat meningkatkan kepuasan dan keterlibatan dosen dalam proses pengajaran, yang pada gilirannya dapat berdampak positif pada kualitas pengajaran.

Selain aspek tumpang tindih waktu dan preferensi dosen, optimisasi jadwal kuliah juga dapat berfokus pada penggunaan optimal sumber daya yang tersedia. Sumber daya ini mencakup ruang kuliah, peralatan, atau tenaga pengajar yang terbatas. Dengan mengoptimalkan penggunaan sumber daya, perancang jadwal dapat menciptakan jadwal yang efisien dan meminimalkan pemborosan sumber daya. Misalnya, mengatur jadwal kuliah sedemikian rupa sehingga ruang kuliah dimanfaatkan secara maksimal tanpa adanya jadwal kosong yang tidak produktif.

Penerapan konsep optimisasi dalam perancangan jadwal kuliah membutuhkan pendekatan yang matang dan algoritma yang efisien. Berbagai teknik optimisasi, seperti algoritma greedy, algoritma genetik, atau pemrograman linier, dapat digunakan untuk mencari solusi jadwal yang optimal berdasarkan tujuan yang ditetapkan. Selain itu, penggunaan teknologi dan perangkat lunak khusus juga dapat mempermudah proses optimisasi jadwal kuliah dengan menggabungkan faktor-faktor yang relevan, menghitung kemungkinan konflik, dan menghasilkan solusi jadwal yang memenuhi kriteria yang ditetapkan.

Dalam kesimpulan, optimisasi jadwal kuliah merupakan pendekatan penting dalam perancangan jadwal yang bertujuan untuk mencapai jadwal yang efisien dan menguntungkan. Dengan meminimalkan tumpang tindih waktu, memaksimalkan preferensi dosen, dan mengoptimalkan penggunaan sumber daya, optimisasi jadwal kuliah dapat meningkatkan efektivitas pembelajaran, kepuasan dosen, dan kesejahteraan mahasiswa.

F. Kompleksitas Algoritma

Kompleksitas algoritma memainkan peran penting dalam perancangan jadwal kuliah yang optimal. Ketika mencari solusi jadwal yang optimal, penting untuk mempertimbangkan kompleksitas waktu dan ruang algoritma yang digunakan. Kompleksitas waktu mengacu pada seberapa efisien algoritma tersebut dalam menyelesaikan permasalahan dengan memperhatikan ukuran masukan. Sedangkan kompleksitas ruang menggambarkan seberapa banyak ruang memori yang diperlukan oleh algoritma saat berjalan.

Dalam konteks perancangan jadwal kuliah, kompleksitas waktu menjadi faktor kritis karena waktu komputasi yang efisien sangat diinginkan untuk mempercepat proses pencarian solusi jadwal. Permasalahan perancangan jadwal kuliah seringkali kompleks dan dapat memiliki jumlah kombinasi yang besar, terutama jika melibatkan banyak mata kuliah dan preferensi dosen. Oleh karena itu, algoritma yang memiliki kompleksitas waktu yang tinggi dapat menghambat efisiensi pencarian solusi yang optimal. Dalam hal ini, algoritma dengan kompleksitas waktu yang rendah, seperti algoritma greedy yang dioptimalkan, dapat menjadi pilihan yang baik untuk mencapai solusi jadwal dalam waktu yang wajar.

Kompleksitas algoritma greedy dapat bervariasi tergantung pada masalah yang dihadapi. Secara umum, algoritma greedy memiliki kompleksitas waktu yang relatif rendah. Biasanya, kompleksitas waktu algoritma greedy adalah O(n log n) atau lebih rendah, di mana n adalah jumlah elemen yang harus diproses.

Selain itu, kompleksitas ruang juga perlu diperhatikan dalam perancangan jadwal kuliah. Ruang memori yang cukup besar dapat diperlukan saat algoritma bekerja dengan dataset yang besar atau ketika melakukan operasi penghitungan yang kompleks. Dalam beberapa kasus, algoritma yang membutuhkan ruang memori yang besar mungkin tidak efisien dalam penggunaan sumber daya. Oleh karena itu, perlu dipertimbangkan algoritma yang memiliki kompleksitas ruang yang lebih rendah untuk menjaga penggunaan memori yang optimal dalam perancangan jadwal kuliah.

Selain kompleksitas waktu dan ruang, faktor lain yang perlu dipertimbangkan adalah tingkat akurasi dan kualitas solusi yang dihasilkan oleh algoritma. Algoritma yang memiliki kompleksitas rendah namun menghasilkan solusi yang suboptimal mungkin tidak menjadi pilihan yang baik. Dalam perancangan jadwal kuliah, tujuan utama adalah mencapai solusi jadwal yang optimal dengan mempertimbangkan preferensi dosen, tumpang tindih waktu minimal, dan penggunaan sumber daya yang efisien. Oleh karena itu, algoritma yang memiliki kompleksitas waktu dan ruang yang moderat namun mampu memberikan solusi jadwal yang optimal dan berkualitas tinggi menjadi pilihan yang lebih baik.

Pemilihan algoritma dalam perancangan jadwal kuliah harus mempertimbangkan keseimbangan antara kompleksitas waktu, kompleksitas ruang, dan kualitas solusi yang dihasilkan. Algoritma yang dapat mengoptimalkan solusi jadwal dengan kompleksitas waktu dan ruang yang wajar akan memberikan keuntungan dalam efisiensi dan akurasi perancangan jadwal kuliah yang optimal. Oleh karena itu, pemilihan algoritma yang tepat dan penggunaan teknik optimisasi yang cermat akan memainkan peran penting dalam mencapai jadwal kuliah yang efisien, memenuhi preferensi dosen, dan memaksimalkan penggunaan sumber daya yang tersedia.

III. IMPLEMENTASI

A. Implementasi Kode

Berikut ini merupakan contoh implementasi perancangan jadwal kuliah optimal dalam bentuk program menggunakan bahasa Python.

```
class Course:
    def __init__(self, course_name, slots):
        self.course_name = course_name
        self.slots = slots

def generate_schedule(course_list):
    n = len(course_list)
    schedule = [None] * n

    for i in range(n):
        best_slot = None
        for slot in course_list[i].slots:
            is_valid_slot = True
            # jika ada slot yang tumpang tindih dengan
slot yang sudah ada, maka slot tersebut tidak valid
            for j in range(i):
                if do_slots_overlap(slot, schedule[j]):
                    is_valid_slot = False
                    break
            # jika slot valid, maka slot tersebut menjadi
slot terbaik
            if is_valid_slot:
                best_slot = slot
                break

        schedule[i] = best_slot
```

```
    optimal_schedule = []
    for i in range(n):
        if schedule[i] is not None:

optimal_schedule.append((course_list[i].course_name,
schedule[i]))
        else:

optimal_schedule.append((course_list[i].course_name,
"Tidak tersedia jadwal yang sesuai"))
    return optimal_schedule

# pengecekan apakah waktu saling tumpang tindih
def do_slots_overlap(slot1, slot2):
    if (slot1 and slot2) is not None:
        day1, time1 = slot1.split(" ")
        day2, time2 = slot2.split(" ")
        if day1 != day2:
            return False
        start_time1, end_time1 = time1.split("-")
        start_time2, end_time2 = time2.split("-")
        start1 = int(start_time1.split(":")[0]) * 60 +
int(start_time1.split(":")[1])
        end1 = int(end_time1.split(":")[0]) * 60 +
int(end_time1.split(":")[1])
        start2 = int(start_time2.split(":")[0]) * 60 +
int(start_time2.split(":")[1])
        end2 = int(end_time2.split(":")[0]) * 60 +
int(end_time2.split(":")[1])
        return start1 < end2 and start2 < end1
    return False

schedule = generate_schedule(course_list)
if schedule is not None:
    print("Jadwal Kuliah yang Optimal:")
    for course_name, slot in schedule:
        print(course_name, ":", slot)
else:
    print("Tidak ditemukan jadwal kuliah yang memenuhi
preferensi mahasiswa.")
```

Langkah-langkah pemrosesan / jalannya program adalah sebagai berikut:

1. Membuat kelas Course yang memiliki atribut course_name (nama mata kuliah) dan slots (daftar preferensi hari dan waktu).
2. Mendefinisikan fungsi generate_schedule(course_list) yang menerima daftar mata kuliah course_list sebagai argumen.
3. Menginisialisasi variabel n dengan panjang course_list dan membuat list schedule dengan panjang n yang diisi dengan nilai None. schedule akan digunakan untuk menyimpan jadwal kuliah yang optimal.
4. Melakukan iterasi untuk setiap mata kuliah dalam course_list menggunakan for i in range(n).
5. Mendefinisikan variabel best_slot yang awalnya diatur sebagai None untuk menyimpan slot waktu terbaik untuk mata kuliah saat ini.
6. Melakukan iterasi untuk setiap slot waktu dalam course_list[i].slots (preferensi hari dan waktu mata kuliah saat ini).
7. Mendefinisikan variabel is_valid_slot yang awalnya diatur sebagai True untuk menandakan bahwa slot waktu saat ini valid.
8. Melakukan iterasi untuk setiap mata kuliah sebelum mata kuliah saat ini menggunakan for j in range(i).
9. Memeriksa apakah slot waktu saat ini tumpang tindih dengan slot waktu mata kuliah sebelumnya yang sudah dipilih menggunakan fungsi do_slots_overlap(slot, schedule[j]). Jika tumpang tindih, maka is_valid_slot diatur sebagai False dan iterasi dihentikan.
10. Jika slot waktu saat ini masih valid (tidak tumpang tindih dengan mata kuliah sebelumnya), maka best_slot diatur sebagai slot waktu saat ini dan iterasi dihentikan.
11. Menyimpan best_slot ke dalam schedule pada indeks i.
12. Membuat list optimal_schedule untuk menyimpan jadwal kuliah yang optimal beserta informasi mata kuliah dan slot waktu.
13. Melakukan iterasi untuk setiap mata kuliah dalam course_list menggunakan for i in range(n).
14. Memeriksa apakah schedule[i] tidak bernilai None. Jika tidak None, maka menambahkan tuple (course_list[i].course_name, schedule[i]) ke dalam optimal_schedule. Jika None, menambahkan tuple (course_list[i].course_name, "Tidak tersedia jadwal yang sesuai").
15. Mengembalikan optimal_schedule sebagai hasil dari fungsi generate_schedule(course_list).
16. Memanggil fungsi generate_schedule(course_list) untuk membuat jadwal kuliah yang optimal berdasarkan preferensi.
17. Mengecek apakah schedule tidak bernilai None. Jika tidak None, maka mencetak jadwal kuliah yang optimal. Jika None, mencetak pesan bahwa tidak

ditemukan jadwal kuliah yang memenuhi preferensi mahasiswa.

B. *Penerapan Algoritma Greedy*

Penerapan algoritma greedy pada kode di atas terlihat dalam langkah-langkah pemilihan slot waktu terbaik untuk setiap mata kuliah. Algoritma greedy memilih slot waktu yang paling awal sesuai dengan preferensi mata kuliah pertama yang diperiksa. Kemudian, algoritma akan memeriksa preferensi mata kuliah berikutnya dan memilih slot waktu yang tidak tumpang tindih dengan mata kuliah sebelumnya.

Pada setiap iterasi, algoritma greedy mempertimbangkan slot waktu preferensi yang tersedia untuk mata kuliah saat ini. Algoritma memeriksa apakah slot waktu tersebut tumpang tindih dengan mata kuliah sebelumnya yang sudah dipilih. Jika ada tumpang tindih, slot waktu tersebut dianggap tidak valid dan algoritma melanjutkan ke slot waktu berikutnya. Namun, jika tidak ada tumpang tindih, slot waktu tersebut dianggap sebagai slot waktu terbaik dan algoritma berhenti untuk mata kuliah saat itu.

Dengan menggunakan pendekatan greedy, algoritma ini tidak mempertimbangkan solusi jangka panjang yang lebih optimal secara keseluruhan. Algoritma hanya fokus pada memilih slot waktu terbaik untuk setiap mata kuliah secara individual berdasarkan preferensi yang tersedia pada saat itu. Ini berarti bahwa algoritma tidak mengeksplorasi semua kemungkinan kombinasi jadwal yang ada dan mungkin tidak menghasilkan jadwal kuliah yang sepenuhnya optimal. dengan mata kuliah sebelumnya.

Meskipun algoritma greedy memiliki keterbatasan dalam mencari solusi yang optimal, penerapannya dalam perancangan jadwal kuliah tetap memiliki kegunaan yang signifikan. Algoritma ini dapat memberikan jadwal kuliah yang memadai dalam waktu yang cepat, terutama jika preferensi dan batasan jadwal tidak terlalu kompleks. Selain itu, algoritma greedy juga dapat digunakan sebagai langkah awal dalam proses optimisasi yang lebih kompleks, di mana solusi yang dihasilkan oleh algoritma greedy dapat menjadi titik awal untuk dioptimalkan lebih lanjut menggunakan algoritma lain.

C. *Pengujian*

Diberikan contoh kasus mata pelajaran beserta jadwal preferensi dosen sebagai berikut:

```
course_list = [
    Course("Matematika", ["Senin 10:00-12:00", "Selasa 10:00-12:00", "Rabu 13:00-15:00"]),
    Course("Bahasa Inggris", ["Senin 14:00-16:00", "Selasa 08:00-10:00", "Rabu 10:00-12:00"]),
    Course("Fisika", ["Senin 08:00-10:00", "Kamis 14:00-16:00", "Jumat 13:00-15:00"]),
    Course("Kimia", ["Selasa 07:00-09:00", "Kamis 10:00-12:00", "Jumat 10:00-12:00"]),
    Course("Biologi", ["Senin 13:00-15:00", "Rabu 14:00-16:00", "Jumat 08:00-10:00"]),
    Course("Sejarah", ["Selasa 13:00-15:00", "Rabu 15:00-17:00", "Jumat 10:00-12:00"]),
    Course("Ekonomi", ["Rabu 08:00-10:00", "Kamis 13:00-15:00", "Jumat 14:00-16:00"]),
    Course("Bahasa Indonesia", ["Senin 13:00-15:00", "Selasa 16:00-18:00", "Rabu 08:00-10:00"]),
    Course("Sosiologi", ["Rabu 10:00-12:00", "Kamis 13:00-15:00", "Jumat 15:00-17:00"]),
    Course("Geografi", ["Selasa 10:00-12:00", "Kamis 14:00-16:00", "Jumat 08:00-10:00"]),
    Course("Seni Rupa", ["Senin 15:00-17:00", "Rabu 10:00-12:00", "Kamis 14:00-16:00"]),
    Course("Komputer", ["Senin 08:00-10:00", "Selasa 10:00-12:00", "Rabu 13:00-15:00"]),
    Course("Musik", ["Senin 14:00-16:00", "Selasa 08:00-10:00", "Rabu 10:00-12:00"]),
    Course("Olahraga", ["Senin 13:00-15:00", "Rabu 14:00-16:00", "Jumat 10:00-12:00"]),
    Course("Akuntansi", ["Rabu 08:00-10:00", "Kamis 13:00-15:00", "Jumat 14:00-16:00"]),
    Course("Psikologi", ["Selasa 10:00-12:00", "Kamis 14:00-16:00", "Jumat 08:00-10:00"]),
]
```

Kode di atas menunjukkan sebuah contoh daftar mata kuliah beserta preferensi waktu kuliah untuk setiap mata kuliahnya. course_list merupakan sebuah list yang berisi objek-objek Course, yang masing-masing objek Course memiliki atribut course_name (nama mata kuliah) dan slots (preferensi waktu kuliah). Misalnya Course("Matematika", ["Senin 10:00-12:00", "Selasa 10:00-12:00", "Rabu 13:00-15:00"]): Mata kuliah "Matematika" memiliki preferensi waktu kuliah pada Senin pukul 10:00-12:00, Selasa pukul 10:00-12:00, dan Rabu pukul 13:00-15:00. Keluaran dari kasus di atas adalah seperti berikut:

```
Jadwal Kuliah yang Optimal:

Matematika : Senin 10:00-12:00
Bahasa Inggris : Senin 14:00-16:00
Fisika : Senin 08:00-10:00
Kimia : Selasa 07:00-09:00
Biologi : Rabu 14:00-16:00
Sejarah : Selasa 13:00-15:00
Ekonomi : Rabu 08:00-10:00
Bahasa Indonesia : Selasa 16:00-18:00
Sosiologi : Rabu 10:00-12:00
Geografi : Selasa 10:00-12:00
Seni Rupa : Kamis 14:00-16:00
Komputer : Tidak tersedia jadwal yang sesuai
Musik : Tidak tersedia jadwal yang sesuai
Olahraga : Jumat 10:00-12:00
Akuntansi : Jumat 14:00-16:00
Psikologi : Jumat 08:00-10:00
```

Terlihat pada hasil di atas, tidak ada jadwal yang saling tumpang tindih, namun ada beberapa mata pelajaran yang tidak memiliki jadwal yang sesuai dikarenakan pilihannya yang sudah diambil oleh amta pelajaran lain. Bagaimanapun juga, hasil di atas sudah menunjukkan hasil yang paling optimal.

D. Kompleksitas Kode

Algoritma pada kode tersebut memiliki kompleksitas yang dapat dijelaskan dari dua aspek, yaitu kompleksitas waktu dan kompleksitas ruang. Kompleksitas waktu mengacu pada jumlah operasi yang dilakukan oleh algoritma saat dijalankan, sedangkan kompleksitas ruang mengacu pada penggunaan memori oleh algoritma.

Dalam hal kompleksitas waktu, algoritma tersebut memiliki kompleksitas O(n^2). Hal ini disebabkan oleh dua buah perulangan bersarang. Perulangan pertama for i in range(n) berjalan sebanyak n kali, sedangkan perulangan kedua for j in range(i) berjalan sebanyak i kali untuk setiap iterasi perulangan pertama. Kedua perulangan ini menjalankan operasi pembandingan waktu tumpang tindih antara slot-slot kuliah. Oleh karena itu, kompleksitas waktu algoritma ini meningkat seiring dengan jumlah mata kuliah yang harus dijadwalkan.

Kompleksitas ruang algoritma ini adalah O(n), di mana n adalah jumlah mata kuliah yang ada dalam course_list. Hal ini disebabkan oleh penggunaan list schedule dengan panjang n sebagai wadah untuk menyimpan jadwal kuliah yang dihasilkan. Selain itu, list optimal_schedule juga digunakan untuk menyimpan jadwal kuliah yang optimal dengan panjang n. Sehingga penggunaan memori oleh algoritma ini sebanding dengan jumlah mata kuliah yang dijadwalkan.

Dalam hal kompleksitas waktu, algoritma ini memiliki kinerja yang cukup efisien karena berjalan dalam waktu polinomial terhadap ukuran masukan. Namun, perlu diingat bahwa kompleksitas tersebut hanya berlaku untuk algoritma yang ditunjukkan pada kode tersebut. Jika ada modifikasi atau penambahan fitur lainnya, kompleksitas algoritma dapat berubah. Oleh karena itu, penting untuk melakukan analisis kompleksitas setiap kali ada perubahan pada algoritma untuk memastikan efisiensi kinerjanya.

IV. KESIMPULAN

Dalam makalah yang telah dibuat, peneliti telah secara komprehensif membahas mengenai perancangan jadwal kuliah dan penerapan algoritma greedy dalam mencari solusi optimal. Peneliti mengidentifikasi beberapa tantangan yang sering dihadapi dalam perancangan jadwal kuliah, salah satunya adalah masalah tumpang tindih waktu antara mata kuliah. Tumpang tindih waktu terjadi ketika ada dua atau lebih mata kuliah yang memiliki jadwal yang bersinggungan, sehingga menyulitkan mahasiswa untuk menghadiri lebih dari satu mata kuliah yang bertabrakan.

Dalam konteks ini, peneliti memahami bahwa penting untuk mempertimbangkan preferensi mahasiswa, seperti preferensi waktu, jumlah jam mengajar, atau preferensi terhadap hari tertentu. Namun, dalam rangka meminimalkan tumpang tindih waktu dan mencapai jadwal kuliah yang optimal, peneliti memperkenalkan konsep optimisasi. Tujuan utama optimisasi dalam perancangan jadwal kuliah adalah mencari solusi terbaik yang memenuhi preferensi dosen, mengoptimalkan penggunaan sumber daya, dan meminimalkan tumpang tindih waktu antara mata kuliah.

Salah satu pendekatan yang peneliti terapkan adalah algoritma greedy. Algoritma greedy merupakan pendekatan heuristik yang mengambil keputusan lokal yang optimal pada setiap langkahnya dengan harapan akan menghasilkan solusi yang global secara optimal. Dalam kode yang peneliti implementasikan, peneliti menggunakan algoritma greedy untuk mencari slot terbaik untuk setiap mata kuliah berdasarkan preferensi waktu yang ada. Dalam prosesnya, peneliti mempertimbangkan slot-slot yang tersedia dan melakukan pengecekan terhadap tumpang tindih waktu dengan mata kuliah yang telah dijadwalkan sebelumnya. Dengan memilih slot terbaik yang tidak tumpang tindih, peneliti berharap dapat menghasilkan jadwal kuliah yang optimal bagi mahasiswa.

Namun, perlu diingat bahwa meskipun algoritma greedy dapat memberikan solusi yang cukup baik, tidak selalu menjamin solusi yang mutlak optimal dalam setiap situasi. Kompleksitas algoritma greedy dapat bervariasi tergantung pada jumlah mata kuliah, jumlah preferensi dosen, dan tingkat tumpang tindih waktu yang terjadi. Oleh karena itu, dalam penelitian peneliti, peneliti juga membahas mengenai kompleksitas algoritma yang digunakan dalam mencari jadwal kuliah yang optimal. Peneliti melihat aspek kompleksitas waktu dan ruang dari algoritma yang peneliti implementasikan untuk memahami kinerjanya dalam menyelesaikan masalah perancangan jadwal kuliah.

Secara keseluruhan, penelitian menunjukkan pentingnya mempertimbangkan preferensi dosen, meminimalkan tumpang tindih waktu, dan mengoptimalkan penggunaan sumber daya dalam perancangan jadwal kuliah. Meskipun algoritma greedy dapat memberikan solusi yang baik dalam banyak kasus, tetap diperlukan penelitian lebih lanjut untuk mengembangkan pendekatan yang lebih kompleks dan efisien dalam mencari solusi jadwal kuliah yang optimal. Diharapkan hasil penelitian dapat memberikan kontribusi pada pengembangan metode dan teknik dalam perancangan jadwal kuliah yang lebih baik di masa depan.

LINK VIDEO

https://youtu.be/qlP1GD7paM8

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 (Accessed: 22 May 2023).

[2] Farisi, O.I.R., Maysyaroh, S. and Dewi, E.F. (2021) Penerapan pewarnaan Graf Pada Penjadwalan mengajar Dosen Pendidikan. Available at: https://ojs.unud.ac.id/index.php/jmat/article/download/72096/40107/ (Accessed: 22 May 2023).

[3] Haryadi, D. and Jamal, A. (2015) Preferensi dosen pada proses Penjadwalan Kuliah menggunakan algoritma genetik studi kasus: Universitas al azhar Indonesia, JURNAL Al-AZHAR INDONESIA SERI SAINS DAN TEKNOLOGI. Available at: https://jurnal.uai.ac.id/index.php/SST/article/view/191 (Accessed: 22 May 2023).

[4] Wijana, K. (1970) Program bantu Penyusunan Jadwal Kuliah, Neliti. Available at: https://www.neliti.com/id/publications/172358/program-bantu-penyusunan-jadwal-kuliah (Accessed: 22 May 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

Ttd

Bernardus Willson 13521021