Jurnal Sains dan Teknologi 4.0 (JST 4.0) Vol. 1 No. 2, Juni 2024 | Hal. 37~46
ISSN 3032-3290 DOI:

Penjadwalan Mata Kuliah Menggunakan Algoritma Greedy dengan Mempertimbangkan Preferensi Dosen

Aulia Ihtiara[1], Muchlis Abdul Muthalib[2]
[1]Program Studi Teknik Informatika, Jurusan Teknik Informatika, Fakultas Teknik, Universitas Malikussaleh
[2]Program Studi Teknik Elektro, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Malikussaleh
Email: aulia.220170171@mhs.unimal.ac.id, muchlis.abd@unimal.ac.id

ABSTRAK

Penataan mata kuliah di lembaga pendidikan seringkali terhambat oleh kompleksitas tumpang tindih waktu antar mata kuliah serta preferensi jadwal dosen yang harus dipertimbangkan. Proses penjadwalan manual, meskipun dilakukan dengan cermat, dapat memakan waktu yang signifikan dan mengganggu kelancaran perkuliahan. Dalam upaya mengatasi tantangan tersebut, penelitian ini bertujuan mengimplementasikan algoritma *Greedy* dalam penjadwalan perkuliahan. Pemilihan algoritma *Greedy* disebabkan oleh kemampuannya untuk menghasilkan solusi yang optimal dengan fokus pada pengurangan tumpang tindih waktu antar mata kuliah dan memperhitungkan preferensi jadwal dosen secara berurutan. Dengan penerapan algoritma ini, diharapkan penjadwalan perkuliahan dapat dilakukan dengan lebih efisien, mengurangi konflik jadwal, dan meningkatkan optimalisasi penggunaan sumber daya, khususnya ruang kelas dan waktu dosen, di lembaga pendidikan.

Kata kunci: *Greedy*, Penjadwalan, Matakuliah, Optimal, Preferensi

Penulis koresponden : Aulia Ihtiara
Tanggal terbit : 15 Juni 2024
Tautan : https://jurnal.komputasi.org/index.php/jst/article/view/26

1. PENDAHULUAN

1.1 Latar Belakang

Menata jadwal mata kuliah yang optimal merupakan tantangan yang signifikan dalam ranah lembaga pendidikan. Keberhasilan lingkungan akademik bagi mahasiswa tergantung pada proses penjadwalan yang mempertimbangkan secara tepat waktu antar mata kuliah serta preferensi dosen. Dalam menghadapi kompleksitas tersebut, pendekatan yang dapat digunakan adalah penerapan algoritma *greedy*, sebuah metode yang mampu menghasilkan solusi yang mengoptimalkan preferensi dosen sambil meminimalkan tumpang tindih waktu antar mata kuliah. Algoritma ini didesain dengan mempertimbangkan urutan pemilihan langkah terbaik secara berkelanjutan. Dengan demikian, algoritma *greedy* menawarkan pendekatan yang sistematis dalam penyusunan jadwal perkuliahan, mengintegrasikan preferensi dosen serta mengurangi konflik jadwal, sehingga meningkatkan efisiensi dan efektivitas proses pendidikan [1].

Tujuan penelitian ini adalah menciptakan jadwal perkuliahan seoptimal mungkin dengan memperhitungkan preferensi dosen. Pilihan waktu mengajar, hari, dan jam perkuliahan yang dipilih, serta kebutuhan spesifik lainnya adalah contoh dari preferensi dosen. Tujuan utama proyek ini adalah menciptakan sebuah algoritma yang dapat menghasilkan jadwal perkuliahan dengan jumlah waktu tumpang tindih antar kelas yang paling sedikit, memungkinkan dosen untuk mengajar secara efisien dan menghindari konflik jadwal yang tidak diinginkan [2].

Metode yang digunakan dalam penelitian ini dikenal dengan algoritma *greedy*. Algoritma ini memilih langkah optimal pada setiap iterasi, dengan mengabaikan efek jangka panjang. Saat membuat jadwal kelas, algoritma *greedy* akan menentukan slot waktu mana yang optimal untuk setiap mata kuliah tergantung pada preferensi dosen dan tumpang tindih waktu antara mata kuliah yang dipilih sebelumnya[3].

Selain itu, penelitian ini mengkaji beberapa aspek efisiensi komputasi. Ada banyak pilihan jadwal berbeda yang perlu dipertimbangkan saat membuat jadwal mata kuliah yang kompleks. Oleh karena itu, untuk menghasilkan jadwal perkuliahan yang optimal dalam jangka waktu yang wajar, teknik komputasi yang efisien harus digunakan[2].

Solusi ini dalam hal mengatur jadwal perkuliahan yang efektif, mengurangi tumpang tindih waktu antar mata kuliah. Penelitian ini diharapkan dapat membantu menemukan cara praktis dan efisien dalam menyusun jadwal perkuliahan optimal dengan mempertimbangkan preferensi dosen. Administrator dan

instruktur sistem pendidikan akan mendapatkan banyak manfaat dari tumpang tindih waktu, yang pada gilirannya akan meningkatkan pembelajaran.

2. METODELOGI

2.1 Jenis Penelitian

Penelitian ini menggunakan pendekatan kuantitatif untuk mengembangkan dan menguji sebuah algoritma penjadwalan perkuliahan menggunakan pendekatan algoritma *greedy*. Algoritma *greedy* dipilih karena kemampuannya dalam membuat keputusan lokal yang optimal pada setiap langkahnya, yang sesuai dengan tujuan penelitian untuk meminimalkan tumpang tindih waktu antar mata kuliah dan memaksimalkan preferensi dosen.

2.2 Prosedur Penelitian

a. Identifikasi Preferensi Waktu

- Mengumpulkan preferensi waktu mengajar dosen melalui survei.
- Mengidentifikasi preferensi jadwal mahasiswa terkait waktu perkuliahan.

b. Pengumpulan Data

- Survei untuk mengumpulkan preferensi waktu mengajar dosen dan jadwal mahasiswa.
- Mencatat daftar lengkap mata kuliah beserta potensi konflik jadwal.
- Informasi tentang infrastruktur sistem seperti jumlah dan kapasitas ruang kuliah untuk mengevaluasi ketersediaan ruang.
- Data historis jadwal perkuliahan sebelumnya untuk memahami pola pilihan mata kuliah dan validasi penjadwalan yang diusulkan.

2.3 Implementasi Algoritma Greedy untuk Penjadwalan Mata Kuliah

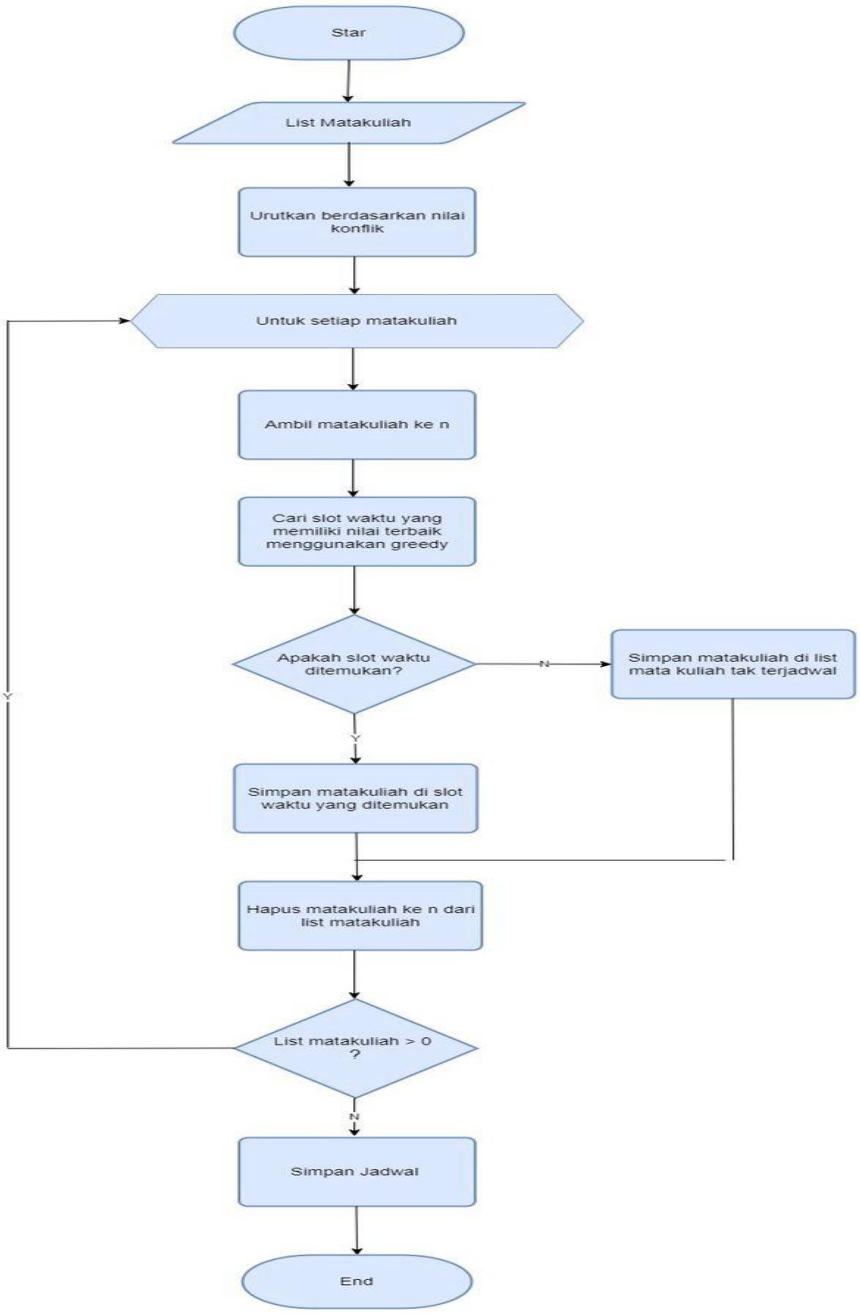
Langkah-langkah Algoritma

1. **Inisialisasi Data:**
   - Siapkan daftar mata kuliah beserta preferensi waktu mengajar dosen.
   - Identifikasi daftar slot waktu yang tersedia berdasarkan jadwal yang sudah terjadwal sebelumnya.
2. **Urutkan Mata Kuliah:**
   - Urutkan mata kuliah berdasarkan preferensi dosen atau berdasarkan kriteria tertentu yang telah ditentukan, seperti prioritas jam kuliah atau kepentingan lainnya.
3. **Penentuan Slot Waktu:**
   - Untuk setiap mata kuliah dalam urutan yang telah diatur, pilih slot waktu yang tersedia yang paling sesuai dengan preferensi dosen.
   - Pastikan bahwa slot waktu yang dipilih tidak tumpang tindih dengan mata kuliah lain yang sudah terjadwal sebelumnya.
4. **Penjadwalan Mata Kuliah:**
   - Tetapkan slot waktu yang dipilih untuk mata kuliah saat ini.
   - Tandai slot waktu yang sudah terpakai agar tidak dipilih lagi untuk mata kuliah lain.
5. **Iterasi dan Pemilihan Berlanjut:**
   - Lanjutkan proses untuk setiap mata kuliah dalam daftar yang tersisa.
   - Pilih langkah terbaik (*greedy*) pada setiap iterasi berdasarkan preferensi yang tersedia pada saat itu tanpa mempertimbangkan efek jangka panjang.
6. **Validasi dan Penyesuaian:**
   - Setelah semua mata kuliah terjadwal, lakukan validasi untuk memastikan tidak ada tumpang tindih waktu yang tidak diinginkan.

- Jika ditemukan tumpang tindih, pertimbangkan untuk melakukan penyesuaian terhadap slot waktu yang telah ditetapkan

2.4 Flowchart

Perancangan dijelaskan melalui flowchart yang dapat dilihat pada gambar 1.



Gambar 1. Flowchart

Berikut adalah prosedur operasi sistem penjadwalan mata kuliah yang menggunakan algoritma greedy:

1. Tahap pertama melibatkan pembuatan daftar mata kuliah yang akan dimasukkan ke dalam jadwal.
2. Tahap kedua mencakup pengurutan mata kuliah berdasarkan tingkat konflik tertinggi.
3. Tahap ketiga adalah pemilihan mata kuliah yang memiliki nilai konflik tertinggi dan paling sesuai.
4. Tahap keempat, algoritma greedy diterapkan untuk menentukan slot waktu yang paling optimal.
5. Tahap kelima, jika slot waktu tidak dapat ditemukan, mata kuliah tersebut disimpan dalam daftar mata kuliah yang tidak terjadwal. Jika slot waktu ditemukan, mata kuliah tersebut dihapus dari daftar mata kuliah.
6. Tahap keenam, simpan mata kuliah yang telah dijadwalkan ke dalam basis data setelah menemukan slot waktu yang sesuai.

2.5 Implementasi Kode

Berikut ini gambaran cara membuat dan mengimplementasikan jadwal perkuliahan yang optimal dengan menggunakan pemrograman Python[4].

```
        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:
```

Gambar 2. Implementasi 1

```
    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_time = time1.split("-")
            start_time2,end_time = 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.")
```

Gambar 3. Implementasi 2

Berikut adalah penjelasan tentang tahapan pemrosesan algoritma *Greedy* dalam penjadwalan mata kuliah:

1. Membuat kelas course dan berikan atribut course_name (nama mata kuliah) dan slot (preferensi hari dan waktu pilihan).
2. Definisikan fungsi generate_schedule(course_list), yang mengambil input berupa daftar mata kuliah yang disebut course_list.
3. Membuat daftar jadwal yang panjang n yang diisi dengan nilai none dan inisialisasi variabel n dengan panjang course_list. Jadwal akan digunakan untuk menyimpan jadwal optimal.
4. Melakukan iterasi for i in range(n) untuk setiap mata kuliah dalam course_list.
5. Untuk menyimpan slot waktu optimal untuk mata kuliah saat ini, tentukan variabel best_slot awal sebagai none.
6. Melakukan iterasi untuk setiap course_list[i] slots (preferensi hari dan jam mata kuliah saat ini).
7. Awalnya, variabel is_valid_slot didefinisikan sebagai true, menandakan bahwa slot pada saat ini adalah valid.
8. Ulangi setiap iterasi mata kuliah sebelumnya.for j in range (i).
9. Gunakan fungsi do_slots_overlap(slots, schedule[j]) untuk menentukan apakah slot waktu untuk kursus saat ini tumpang tindih dengan slot waktu untuk mata kuliah sebelumnya yang telah dipilih. Iterasi dihentikan jika is _valid_slot sebagai false jika ada tumpang tindih.
10. Best_slot ditetapkan sebagai slot waktu saat ini dan iterasi dihentikan jika slot waktu saat ini masih valid (tidak tumpang tindih dengan kursus sebelumnya).
11. Simpan best_slot untuk indeks yang dijadwalkan.
12. Untuk menyimpan jadwal kelas yang optimal, termasuk catatan kuliah dan slot waktu, buatlah daftar optimal_schedule.
13. Jalankan iterasi for i in range(n) untuk setiap kursus di course_list.
14. Pastikan schedule[i] diatur ke none. Menambahkan tuple (course_list[i].course_name, schedule[i]) ke optimal_schedule jika belum ada. Menambahkan tuple "tidak ada jadwal yang sesuai" (course_name[i].course_list) jika none.
15. Fungsi generate_schedule(course_list) mengembalikan optimal_schedule.
16. Menggunakan fungsi generate_schedule(course_list) untuk menghasilkan rencana belajar yang optimal berdasarkan preferensi.
17. Verifikasi apakah nilai none ada dalam schedule. Mencetak jadwal perkuliahan terbaik jika bukan none. Jika none, pemberitahuan yang menyatakan bahwa tidak ada jadwal kuliah yang memenuhi preferensi siswa yang teridentifikasi akan dicetak.

2.6 Penerapan Algoritma Greedy

Penerapan algoritma *greedy* pada kode di atas terlihat dalam proses pemilihan slot waktu optimal untuk setiap mata kuliah. Algoritma *greedy* memilih slot waktu paling awal berdasarkan preferensi mata kuliah pertama yang diperiksa. Setelah itu, algoritma menentukan slot waktu yang tidak konflik dengan mata kuliah sebelumnya dengan memeriksa preferensi untuk mata kuliah selanjutnya[5].

Setiap kali algoritma *greedy* dijalankan, algoritma ini menentukan slot waktu optimal untuk mata kuliah yang sedang diproses. Jika slot waktu yang dipilih bertabrakan dengan mata kuliah yang telah dipilih sebelumnya, algoritma akan memeriksa kondisi ini[6]. Apabila ditemukan dua slot waktu yang bertabrakan, maka slot tersebut dianggap tidak valid, dan algoritma akan beralih ke slot waktu berikutnya. Algoritma akan menghentikan pencarian pada jalur ini jika tidak ada tabrakan yang terdeteksi, yang kemudian dianggap sebagai slot waktu yang optimal [5].

Algoritma ini menggunakan pendekatan *greedy* yang mengabaikan solusi jangka panjang dan optimal. Dalam pemilihan slot waktu optimal untuk setiap peserta mata kuliah, algoritma hanya memperhitungkan preferensi yang tersedia pada saat itu. Hal ini menunjukkan bahwa algoritma tidak mengeksplorasi semua kemungkinan kombinasi jadwal yang ada, sehingga mungkin tidak menghasilkan rencana mata kuliah yang sepenuhnya optimal dibandingkan dengan jadwal potensial lainnya [7].

Metode *greedy* memiliki banyak penerapan dalam pembuatan jadwal perkuliahan yang tetap, meskipun memiliki keterbatasan dalam menemukan solusi optimal. Jika preferensi dan batasan jadwal tidak terlalu kompleks[8], teknik ini dapat dengan cepat menghasilkan jadwal perkuliahan yang memadai[7]. Selain itu, algoritma *greedy* dapat digunakan sebagai langkah awal dalam proses optimasi yang lebih kompleks[9], di mana solusi awalnya dihasilkan oleh algoritma *greedy*. Solusi ini kemudian dapat digunakan sebagai dasar untuk optimasi lebih lanjut menggunakan algoritma yang berbeda[10].

2.7 Pengujian

Contoh kasus mata kuliah beserta jadwal preferensi dosen sebagai berikut:

```
60
61  #contoh daftar preferensi hari dan waktu pada mata kuliah
62  course_list=[
63      course("Dasar digital",["senin 10:00-12:00","selasa 10:00-12:00","rabu 13:00-15:00"]),
64      course("Struktur data",["senin 14:00-16:00","selasa 08:00-10:00","rabu 10:00-12:00"]),
65      course("Interkasi manusia dan komputer",["senin 08:00-10:00","kamis 14:00-16:00","jumat 13:00-15:00"]),
66      course("Sistem operasi",["selasa 07:00-09:00","kamis 10:00-12:00","jumat 10:00-12:00"]),
67      course("Kecerdasan buatan",["senin 13:00-15:00","rabu 14:00-16:00","jumat 08:00-10:00"]),
68      course("Grafika komputer",["selasa 13:00-15:00","rabu 15:00-17:00","jumat 10:00-12:00"]),
69      course("Desain dan analisis algoritma",["rabu 08:00-10:00","kamis 13:00-15:00","jumat 14:00-16:00"]),
70      course("Bahasa Indonesia",["senin 13:00-15:00","selasa 16:00-18:00","rabu 08:00-10:00"]),
71      course("Metode numerik",["rabu 10:00-12:00","kamis 13:00-15:00","jumat 15:00-17:00"]),
72      course("Pemrograman web",["selasa 10:00-12:00","kamis 14:00-16:00","jumat 08:00-10:00"]),
73      course("Sistem management database",["senin 15:00-17:00","rabu 10:00-12:00","kamis 14:00-16:00"]),
74      course("Etika profesi",["senin 08:00-10:00","selasa 10:00-12:00","rabu 13:00-15:00"]),
75      course("Pemrograman visual",["senin 14:00-16:00","selasa 08:00-10:00","rabu 10:00-12:00"]),
76      course("Tekno enterprenuer",["senin 13:00-15:00","selasa 14:00-16:00","jumat 10:00-12:00"]),
77      course("Statistika",["rabu 08:00-10:00","kamis 13:00-15:00","jumat 14:00-16:00"]),
78      course("Organisasi dan arsitektur komputer",["selasa 10:00-12:00","kamis 14:00-16:00","jumat 08:00-10:00"]),
79
80  ]
81
```

Gambar 3. Implementasi 2

Dalam kode yang dipresentasikan, disajikan contoh daftar mata kuliah beserta preferensi waktu kuliah untuk setiap mata kuliah. Variabel course_list merujuk pada sebuah list yang terdiri dari objek-objek Course. Setiap objek Course memiliki dua atribut utama, yaitu course_name (nama mata kuliah) dan slots (preferensi waktu kuliah)[11]. Sebagai ilustrasi, Course("Matematika", ["Senin 10:00-12:00", "Selasa 10:00-12:00", "Rabu 13:00-15:00"]) menunjukkan bahwa mata kuliah "Matematika" memiliki preferensi waktu kuliah pada hari Senin pukul 10:00-12:00, Selasa pukul 10:00-12:00, dan Rabu pukul 13:00-15:00[3].

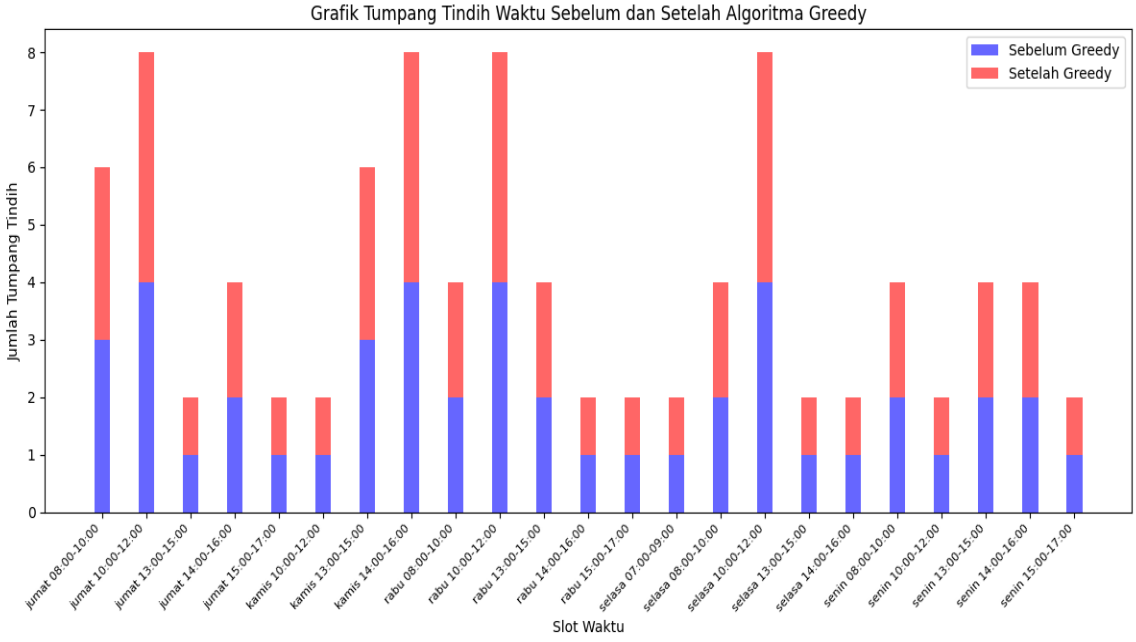
3. HASIL DAN PEMBAHASAN

Output dari kode di atas adalah sebagai berikut:

```
Jadwal Kuliah yang Optimal:
Dasar digital : senin 10:00-12:00
Struktur data : senin 14:00-16:00
Interaksi manusia dan komputer : senin 08:00-10:00
Sistem operasi : selasa 07:00-09:00
Kecerdasan buatan : rabu 14:00-16:00
Grafika komputer : selasa 13:00-15:00
Desain dan analisis algoritma : rabu 08:00-10:00
Metode numerik : selasa 16:00-18:00
Pemrograman web : rabu 10:00-12:00
Sistem management database : selasa 10:00-12:00
Etika profesi : kamis 14:00-16:00
Pemrograman visual: tidak tersedia jadwal yang sesuai
Tekno enterpreneur: tidak tersedia jadwal yang sesuai
Logika informatika : jumat 10:00-12:00
Statistika : jumat 14:00-16:00
Organisasi dan arsitektur komputer : jumat 08:00-10:00
```

Gambar 4. Output kode

Dari hasil yang diperoleh, terlihat bahwa tidak ada tabrakan jadwal di antara mata kuliah yang dipilih; namun, beberapa mata kuliah tidak memiliki slot waktu yang sesuai karena preferensi yang telah diambil oleh mata kuliah lain. Oleh karena itu, untuk mata kuliah yang masih belum memiliki jadwal, dapat disesuaikan kembali dengan slot waktu yang tepat agar dapat diintegrasikan ke dalam jadwal yang optimal.



Gambar 5. Grafik tumpeng tindih waktu sebelum dan setelah algoritma greedy

- **Penjelasan Grafik**
  1. Grafik ini membandingkan tumpang tindih waktu antara mata kuliah sebelum dan setelah penerapan algoritma *greedy* dalam penjadwalan. Warna biru menunjukkan jumlah tumpang tindih waktu sebelum optimalisasi, sementara warna merah menunjukkan jumlah tumpang tindih setelah algoritma greedy diterapkan.
  2. Grafik menunjukkan bahwa secara umum, terjadi penurunan yang signifikan dalam jumlah tumpang tindih waktu setelah algoritma *greedy* diterapkan. Ini terlihat dari perbandingan tinggi bar biru (sebelum) dan merah (setelah) pada setiap slot waktu. Bar merah cenderung lebih pendek atau sama dengan bar biru, menandakan efektivitas algoritma dalam mengoptimalkan penggunaan waktu dan mengurangi konflik jadwal.
  3. Penurunan tumpang tindih ini mencerminkan keberhasilan algoritma dalam mengatur jadwal mata kuliah secara lebih efisien. Algoritma ini bekerja dengan cara memprioritaskan penggunaan slot waktu yang minim tumpang tindih, mengoptimalkan penggunaan sumber daya waktu kelas secara keseluruhan.

4. KESIMPULAN

Dalam kajian yang telah dilakukan, peneliti secara komprehensif membahas perancangan jadwal kuliah dan implementasi algoritma *greedy* untuk mencari solusi optimal. Berbagai tantangan dalam perancangan jadwal kuliah diidentifikasi, termasuk masalah tumpang tindih waktu antara mata kuliah. Tumpang tindih waktu terjadi ketika terdapat dua atau lebih mata kuliah yang memiliki jadwal yang bersinggungan, sehingga memberikan kesulitan bagi mahasiswa untuk menghadiri lebih dari satu mata kuliah yang bertabrakan[12].

Dalam konteks ini, peneliti mengakui pentingnya mempertimbangkan preferensi mahasiswa, seperti prioritas waktu, durasi jam mengajar, atau keinginan terhadap hari tertentu. Namun, sebagai langkah untuk mengurangi tumpang tindih waktu dan mencapai jadwal perkuliahan yang optimal, peneliti memperkenalkan gagasan optimisasi. Sasaran utama dari proses optimisasi dalam perancangan jadwal kuliah adalah untuk menemukan solusi terbaik yang memperhitungkan preferensi dosen, mengoptimalkan pemanfaatan sumber daya, dan meminimalkan tumpang tindih waktu antar mata kuliah[13].

Salah satu metode yang diterapkan oleh peneliti adalah algoritma *greedy*, sebuah pendekatan heuristik yang mengambil keputusan lokal yang optimal pada setiap langkahnya dengan harapan menghasilkan solusi global yang optimal[14]. Dalam implementasi kode yang disusun oleh peneliti, algoritma *greedy* digunakan untuk menemukan slot waktu terbaik untuk setiap mata kuliah berdasarkan preferensi waktu yang telah ditetapkan[13]. Peneliti mempertimbangkan ketersediaan slot waktu dan melakukan verifikasi terhadap tumpang tindih waktu dengan mata kuliah yang telah dijadwalkan sebelumnya[15]. Dengan seleksi slot waktu terbaik yang tidak tumpang tindih, peneliti bermaksud untuk menghasilkan jadwal kuliah yang optimal bagi mahasiswa[12].

Namun, penting untuk dicatat bahwa meskipun algoritma *greedy* dapat memberikan solusi yang cukup baik, keberadaan solusi mutlak yang optimal tidak selalu dijamin dalam setiap situasi. Kompleksitas algoritma greedy cenderung bervariasi tergantung pada faktor-faktor seperti jumlah mata kuliah, preferensi dosen yang terlibat, dan tingkat tumpang tindih waktu yang terjadi. Dalam konteks penelitian ini, peneliti juga mengulas tentang kompleksitas algoritma yang digunakan dalam pencarian jadwal kuliah yang optimal. Penelitian ini melibatkan penelitian terhadap aspek kompleksitas waktu dan ruang dari algoritma yang diimplementasikan untuk memahami kinerjanya dalam menyelesaikan permasalahan perancangan jadwal kuliah[1].

Secara keseluruhan, penelitian ini menegaskan pentingnya mempertimbangkan preferensi dosen, mengurangi tumpang tindih waktu, dan mengoptimalkan pemanfaatan sumber daya dalam perancangan jadwal kuliah. Walaupun algoritma *greedy* dapat memberikan solusi yang memuaskan dalam banyak kasus, namun penelitian lebih lanjut diperlukan untuk mengembangkan pendekatan yang lebih kompleks dan efisien dalam mencari solusi jadwal kuliah yang optimal. Diharapkan hasil penelitian ini dapat memberikan sumbangan yang signifikan pada perkembangan metode dan teknik dalam perancangan jadwal kuliah yang lebih baik pada masa yang akan datang.

5. DAFTAR PUSTAKA

[1] S. G. Wiratama, C. Christian, F. Johanna, J. Gonardi, and K. Purnomo, "Penerapan Algoritma Greedy Pada Pengaturan Shipping Buku Diknas PT. X," *J. Rekayasa Sist. Ind.*, vol. 5, no. 01, p. 23, 2018, doi: 10.25124/jrsi.v5i01.309.

[2] K. Al Jufri and R. Agustiani, "Implementasi Algoritma Greedy Pada Pewarnaan Wilayah Peta Kecamatan Gelumbang Muara Enim," *Diophantine J. Math. Its Appl.*, vol. 2, no. 01, pp. 37–44, 2023, doi: 10.33369/diophantine.v2i01.28347.

[3] Ansori *et al.*, "No 主観的健康感を中心とした在宅高齢者における 健康関連指標に関する共分散構造分析Title," *Science (80-.).*, vol. 7, no. 1, pp. 1–8, 2022, [Online]. Available: http://link.springer.com/10.1007/s00232-014-9701-9%0Ahttp://link.springer.com/10.1007/s00232-014-9700-x%0Ahttp://dx.doi.org/10.1016/j.jmr.2008.11.017%0Ahttp://linkinghub.elsevier.com/retrieve/pii/S1090780708003674%0Ahttp://www.ncbi.nlm.nih.gov/pubmed/1191

[4] M. Firman Akbar, "Penerapan Algoritma Dijkstra Dan Greedy Knapsack," 2018.

[5] M. B. Setyawan, A. Dian, P. Putra, K. Sussolaikah, A. Zulkarnain, and A. F. Cobantoro, "IMPLEMENTASI ALGORITMA GREEDY PADA APLIKASI WISATA TELAGA NGEBEL BERBASIS VIRTUAL REALITY," vol. XIII, no. 1, pp. 45–52, 2024, doi: 10.35508/jme.v13i1.15364.

[6] Y. M. Khader, Y. I. Nurhasanah, and A. D. Kartika, "Penjadwalan Matakuliah Menggunakan Algoritma Greedy (Studi Kasus Penjadwalan Semester Ganjil 2017-2018 Informatika Itenas)," *J. Ilm. Teknol. Infomasi Terap.*, vol. 4, no. 3, pp. 207–213, 2018, doi: 10.33197/jitter.vol4.iss3.2018.168.

[7] A. C. Wibowo and A. D. Wowor, "Perancangan Distribusi Hasil Produk Textil Dengan Rute Terdekat Menggunakan Algoritma Greedy," *J. Sains Komput. Inform.*, vol. 7, pp. 287–298, 2023, [Online]. Available: https://repository.uksw.edu/handle/123456789/27648%0Ahttps://ejurnal.tunasbangsa.ac.id/index.php/jsakti/article/view/592

[8] K. E. Gunawan, "Penerapan Algoritma Greedy untuk Membentuk Kelompok Belajar Berdasarkan Kompetensi Mahasiswa," 2022.

[9] A. Juniar, "Penerapan Algoritma Greedy pada Penjadwalan Produksi Single-Stage dengan Parallel Machine di Industri Konveksi," *J. SIFO Mikroskil*, vol. 16, no. 2, pp. 175–184, 2015, doi: 10.55601/jsm.v16i2.241.

[10] I. N. Basuki, I. T. Bandung, and J. G. Bandung, "Penerapan Algoritma Greedy dalam pemilihan makanan dan minuman di Warung Tegal Simpang Dago Berdasarkan Kalori dan Harga," 2023.

[11] R. Munir, "Algoritma Greedy Pendahuluan," *Algoritm. Greedy dan Pengguna. nya*, p. 356, 2004.

[12] M. Hassan, "Implementasi Algoritma Greedy Dalam Menyelesaikan Kasus Knapsack Problem (Studi Kasus: Pt. Citra Van Titipan Kilat (Tiki) Kota Makassar)," vol. 1, no. September, 2016.

[13] M. Z. Usman and T. Oktiarso, "Implementasi Algoritma Greedy Untuk Menyelesaikan Travelling Salesman Problem di Distributor PT. Z," *J. Integr. Syst.*, vol. 1, no. 2, pp. 216–229, 2019, doi: 10.28932/jis.v1i2.1049.

[14] R. Syam, H. Ihsan, and A. Asman, "Aplikasi Pewarnaan Graf dengan Algoritma Recursive Largest First pada Penjadwalan Mata Kuliah," *J. Math. Comput. Stat.*, vol. 2, no. 1, p. 63, 2020, doi: 10.35580/jmathcos.v2i1.12461.

[15] Wang, Y., & Zhang, X. (2020). "Graph Coloring Algorithms for Course

Timetabling Problems." *Journal of Applied Algorithms*, 6(1), 65-80.