Penerapan Algoritma Greedy dan Backtracking dalam Menjadwalkan Kegiatan

Christophorus Dharma Winata - 13521009
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-Mail (gmail): 13521009@std.stei.itb.ac.id

Penjadwalan aktivitas yang efisien sangat penting untuk mengoptimalkan sumber daya, meminimalkan biaya, dan meningkatkan produktivitas di berbagai domain. Makalah ini mengeksplorasi implementasi strategi algoritmik, secara khusus berfokus pada dua pendekatan: algoritma *Greedy* dan algoritma *backtracking*. Algoritma *greedy* membuat pilihan optimal secara lokal di setiap langkah, sementara algoritma *backtracking* secara sistematis menjelajahi ruang pencarian. Makalah ini mengkaji bagaimana strategi ini dapat diterapkan pada aktivitas penjadwalan, membahas keuntungan, tantangan, dan contoh praktisnya. Studi kasus dunia nyata disajikan untuk mengilustrasikan keefektifan algoritma ini dalam memecahkan masalah penjadwalan yang berbeda. Selain itu, makalah ini membahas integrasi prinsip-prinsip pemrograman dinamis ke dalam algoritma penjadwalan, memberikan perspektif holistik pada strategi algoritmik dalam kegiatan penjadwalan.

Kata kunci—jadwal; algoritma; greedy; backtracking; dinamis;

I. PENGANTAR

A. Latar Belakang

Penjadwalan aktivitas secara efisien dan efektif bersifat sangat penting dalam berbagai domain, termasuk manajemen proyek maupun organisasi, manufaktur, transportasi, dan perawatan kesehatan. Kemampuan untuk mengalokasikan sumber daya secara optimal, meminimalkan biaya, dan memastikan penyelesaian tugas tepat waktu dapat berdampak signifikan terhadap produktivitas dan keberhasilan organisasi secara keseluruhan. Namun, kegiatan penjadwalan menimbulkan tantangan yang melekat karena kompleksitas dan sifat kombinatorial dari masalah yang terlibat. Dalam penjadwalan, banyak kendala perlu dipertimbangkan, seperti sumber daya yang terbatas, hubungan prioritas, dan keterbatasan waktu. Selain itu, tujuan yang saling bertentangan sering muncul, di mana mengoptimalkan satu aspek dapat menyebabkan pertukaran di bidang lain. Misalnya, dalam penjadwalan proyek, mencapai durasi proyek tersingkat mungkin memerlukan alokasi sumber daya tambahan, yang mengakibatkan peningkatan biaya. Menyeimbangkan kendala dan tujuan ini sambil memaksimalkan efisiensi menimbulkan tantangan komputasi dan pengambilan keputusan yang signifikan.

Penerapan strategi algoritma menyediakan alat yang ampuh untuk mengatasi tantangan ini dan mengoptimalkan proses penjadwalan. Dengan memanfaatkan model matematika, struktur data, dan pendekatan sistematis, algoritma dapat secara efektif mencari solusi dalam ruang solusi yang luas dari masalah penjadwalan. Algoritma ini memungkinkan pembuat keputusan untuk membuat pilihan berdasarkan informasi, mengalokasikan sumber daya secara cerdas, dan membuat jadwal yang memenuhi tujuan yang diinginkan.

Untuk menghadapi tantangan dan peluang ini, makalah ini bertujuan untuk mengeksplorasi implementasi strategi algoritmik dalam kegiatan penjadwalan. Secara khusus, kami akan fokus pada penerapan algoritma *Greedy* dan algoritma *backtracking*, memeriksa kelebihan, tantangan, dan contoh praktisnya dalam menyelesaikan berbagai masalah penjadwalan. Dengan memahami peran strategi algoritma ini, kami dapat membuka potensinya untuk mengoptimalkan aktivitas penjadwalan dan berkontribusi pada kemajuan bidang dinamis ini.

II. STRATEGI GREEDY DALAM PENJADWALAN

A. Konsep Algoritma Greedy dan Karakteristiknya

Algoritma Greedy adalah algoritma yang memecahkan persoalan dengan pendekatan per langkah dengan prinsip mengambil langkah terbaik pada langkah tersebut dengan harapan langkah optimum yang lokal itu dapat memberikan hasil optimum yang global. Algoritma greedy merupakan algoritma yang umum digunakan untuk memecahkan persoalan optimasi.

Dalam implementasinya, algoritma greedy beberapa elemen, yaitu:

- Himpunan kandidat: pilihan-pilihan yang bisa diambil pada setiap langkah. Elemen ini umumnya diwakilkan dengan simbol C
- Himpunan solusi: himpunan pilihan yang sudah dipilih dalam algoritma. Elemen ini diwakilkan dengan simbol S
- Fungsi solusi: fungsi yang menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
- Fungsi seleksi: fungsi yang melakukan pemilihan dari himpunan kandidat. Fungsi ini yang menentukan bagaimana strategi greedy bekerja dan bersifat heuristik.

- Fungsi kelayakan: fungsi yang memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi
- Fungsi obyektif: fungsi yang memaksimumkan atau meminimumkan

B. Pengaplikasian Algoritma Greedy dalam Penjadwalan

Dalam aplikasi algoritma greedy yang berkaitan dengan penjadwalan, contohnya adalah sebagai berikut:

1) Interval Scheduling

Persoalan interval scheduling terjadi ketika diberikan suatu kumpulan kegiatan dengan waktu mulai dan waktu selesai. Dengan tidak diperbolehkan lebih dari satu kegiatan sekaligus, kita mencari bagaimana cara untuk mengikuti sebanyak mungkin kegiatan yang ada.

Dalam memecahkan persoalan ini, strategi greedy diaplikasikan dengan cara:

1. Mengurutkan aktivitas (C) berdasarkan waktu selesainya terurut dari paling awal sampai terakhir
2. Pada setiap langkah, pilih aktivitas yang waktu mulainya lebih atau sama dengan waktu selesai kegiatan yang baru dipilih (fungsi seleksi)

Elemen algoritma greedy lainnya di persoalan ini adalah:

- S = Jadwal yang sudah dipilih
- Fungsi solusi = Mencari apakah ada kegiatan lain yang dapat dipilih
- Fungsi kelayakan = Bersamaan dengan fungsi seleksi, memilih kegiatan yang waktu mulainya tidak tumpang tindih dengan kegiatan sekarang
- Fungsi obyektif = Memaksimumkan banyak kegiatan

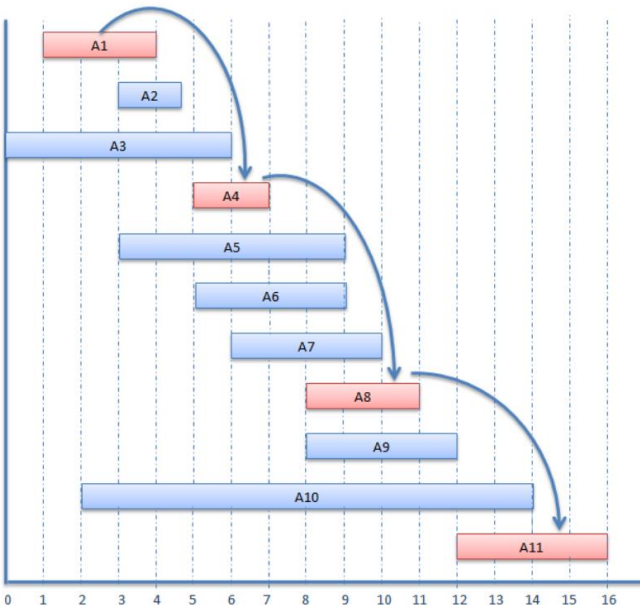


Figure 1 Visualisasi solusi algoritma greedy persoalan interval scheduling (Munir, Algoritma Greedy Bagian 1, 2023)

2) Task Scheduling

Persoalan task scheduling terjadi ketika diberikan suatu kumpulan tugas dengan deadline dan bobot dari setiap tugas. Dalam persoalan ini, kita mencari bagaimana cara mengerjakan sebanyak mungkin keuntungan dari semua tugas yang diberikan.

Solusi algoritma greedy dalam memecahkan persoalan ini adalah:

1. Mengurutkan semua tugas yang tersedia dari bobot terbesar ke terkecil
2. Memilih dari urutan tugas yang belum mencapai deadline

Elemen-elemen algoritma greedy dalam solusi tersebut adalah:

- C = Tugas yang bisa dikerjakan
- S = Tugas yang dikerjakan
- Fungsi kelayakan = Memeriksa jika tugas yang dikerjakan tidak melewati deadline
- Fungsi seleksi = pengurutan C
- Fungsi obyektif = Memaksimumkan bobot tugas yang dikerjakan
- Fungsi solusi = Memeriksa jika dari langkah yang sekarang sudah tidak ada lagi tugas yang bisa diambil.

$(p_1, p_2, p_3, p_4) = (50, 10, 15, 30)$

$(d_1, d_2, d_3, d_4) = (2, 1, 2, 1)$

Langkah	J	$F = \sum p_i$	Keterangan
0	{}	0	-
1	{1}	50	layak
2	{4,1}	50 + 30 = 80	layak
3	{4, 1, 3}	-	tidak layak
4	{4, 1, 2}	-	tidak layak

Solusi optimal: J = {4, 1} dengan F = 80.

Figure 2 Contoh kasus Job Scheduling (Munir, Algoritma Greedy Bagian 1, 2023)

Gambar di atas adalah contoh kasus penjadwalan tugas dimana tugas-tugas diberikan nilai deadline d dan masing-masing memiliki profit p. Di dalam contoh tersebut, kemungkinan pengambilan dilakukan dengan mengikuti langkah-langkah di atas.

C. Keuntungan dan Batasan Algoritma Greedy dalam Penjadwalan

Algoritma Greedy dalam implementasinya akan menghasilkan hasil yang optimal. Tetapi karena bagaimana algoritma tersebut bekerja, dimana pemilihan langkah ditentukan dari bagaimana langkah tersebut menguntungkan di saat itu saja, hasil cenderung untuk **mendekati** optimal.

Dalam kebanyakan kasus, hasil algoritma greedy sudah bisa dinilai optimal. Tetapi karena hanya sampai mendekati, disarankan untuk menggunakan cara-cara yang memberikan hasil yang lebih optimal akan lebih baik dari algoritma greedy. Kompleksitas algoritma greedy juga beragam tergantung dari kasus algoritma yang dihadapi.

III. Strategi Backtracking dalam Penjadwalan

A. Konsep Algoritma Backtracking dan Karakteristiknya

Algoritma Backtracking adalah bentuk perbaikan exhaustive search dimana semua kemungkinan solusi dieksplorasi dan dievaluasi satu persatu dengan hanya memilih kemungkinan yang mengarah ke solusi dan memangkas sisa pilihan lain.

Backtracking dapat dipandang sebagai salah satu dari kedua hal yakni:

1. Fase dalam Depth-First Search.

   Algoritma yang mencari kemungkinan pada simpul-simpul dengan prioritas kedalaman atau mencari hingga simpul yang dieksplor adalah buntu. Fase backtracking dalam algoritma ini adalah ketika proses algoritma harus secara bertahap mundur untuk mencari simpul lain.

2. Metode pemecahan masalah untuk persoalan optimasi maupun non-optimasi.

Algoritma Backtracking menyusun ruang solusi (C) dalam bentuk graf pohon berakar dimana setiap simpul adalah status solusi dan setiap cabang adalah nilai dari himpunan C.

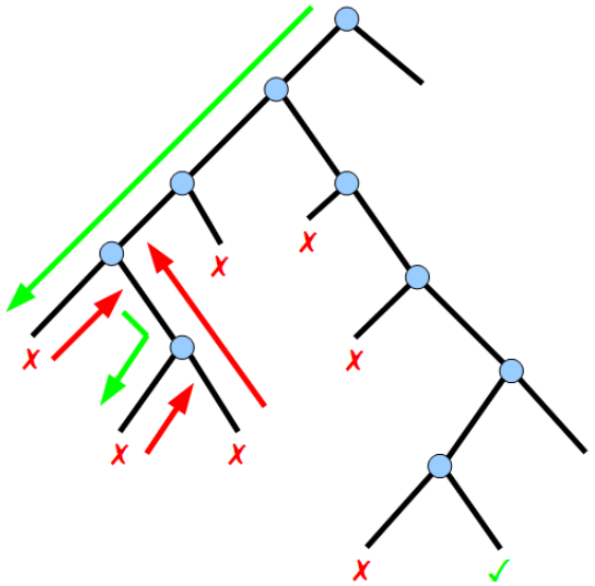


Figure 3 Visualisasi cara kerja algoritma DFS dan Backtracking (Munir, Algoritma Backtracking Bagian 1, 2023)

Pada gambar di atas, fase backtracking ditunjukkan dengan garis merah pada saat eksplorasi graf pohon. Fase tersebut dimulai ketika eksplorasi menemukan simpul daun (simpul yang tidak memiliki cabang) lalu memundurkan langkah dan mengeksplorasi cabang lain dengan langkah yang sama.

Algoritma Backtracking memiliki beberapa properti dasar:

1. Solusi persoalan
2. Fungsi pembangkit: Fungsi yang menemukan solusi yang mungkin didapat pada suatu state tertentu
3. Fungsi pembatas: Fungsi yang menentukan jika pilihan mengarah ke solusi

B. Pengaplikasian Algoritma Backtracking dalam Penjadwalan

1) N-Queens Problem

N-Queen problem adalah persoalan dimana kita harus meletakkan N buah ratu pada suatu papan catur dengan ukuran $N \times N$ dimana setiap ratu tidak saling berada di baris, kolom, maupun diagonal yang sama.

Pendekatan dari algoritma Backtracking pada kasus ini berlangsung seperti berikut:

1. Membagi setiap langkah evaluasi peletakan sebagai setiap baris di papan catur
2. Ketika berpindah ke baris berikutnya, evaluasi peletakan dilakukan jika ratu baris tersebut tidak berpotongan dengan ratu sebelumnya
3. Langkah ini dilanjutkan hingga baris ke-N
4. Jika hingga baris N ditemukan buntu (semua petak di baris N tidak layak menjadi solusi), lakukan backtracking hingga bisa ditemukan solusi lain
5. Ulangi proses dari langkah 2

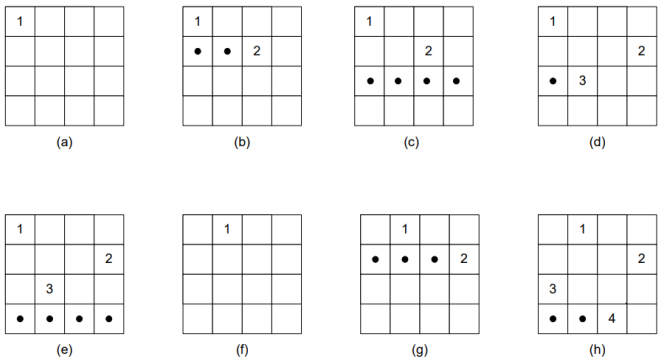


Figure 4 Visualisasi contoh algoritma backtracking dalam 4-Queens problem (Munir, Algoritma Backtracking Bagian 1, 2023)

Gambar di atas adalah visualisasi dari solusi N-Queens problem dengan menggunakan algoritma Backtracking. Pada contoh di atas, diambil sebagai contoh $N = 4$.

2) Traveling Salesperson Problem

Traveling Salesperson Problem adalah persoalan yang terkenal memiliki sifat NP-hard. Kasus ini memberikan suatu kumpulan kota dalam bentuk graf dimana semua kota saling terhubung oleh suatu garis berjarak (atau dalam graf memiliki berbobot), lalu menanyakan cara untuk melewati semua kota dengan jarak yang paling sedikit seperti seorang pedagang keliling yang ingin mencari untung.

Algoritma Backtracking dalam hal ini mengoptimasi cara kerja algoritma Bruteforce dengan mengeliminasi

kemungkinan yang tidak memberikan solusi. Langkah-langkah tersebut secara umum adalah sebagai berikut:

1. Asumsikan suatu kota sebagai titik mulai perjalanan
2. Bangkitkan kemungkinan dengan kota-kota dengan memeriksa garis-garis yang menghubungkan kota sekarang dengan kota-kota dekatnya
3. Lakukan Depth-First Search hingga dengan akhir pada kota awal
4. Jika ditemukan suatu solusi, simpan jarak dan rute solusi tersebut sebagai jawaban sementara
5. Jika melanggar batasan seperti mengunjungi kota yang sudah pernah dikunjungi atau meraih suatu titik dimana semua kemungkinan di langkah tersebut tidak valid, dilakukan backtracking ke status sebelum status tersebut
6. Jika berhasil ditemukan lagi potensi solusi, ulang proses dari langkah 2
7. Tambahkan batasan dimana jika ditemukan suatu solusi lain, lakukan perbandingan solusi tersebut dengan hasil sementara yang sudah dilakukan
8. Semua langkah ini dilakukan hingga semua kemungkinan berhasil dieksplorasi

Berikut adalah gambar contoh kasus Traveling Salesman Problem:

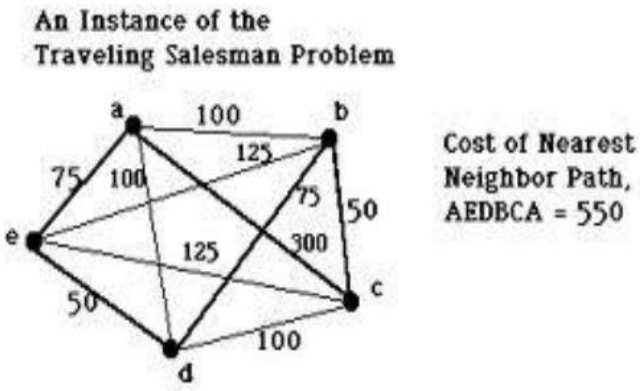


Figure 5 Contoh TSP (Munir, Graf Bagian 3, 2023)

C. Keuntungan dan Batasan Algoritma Backtracking dalam Penjadwalan

Algoritma Backtracking memberikan hasil yang baik atau mendekati baik pada ruang solusi berukuran besar. Tetapi, karena algoritma ini berprinsip dasar *Exhaustive Search*, algoritma akan berpotensi besar untuk membutuhkan kekuatan komputasi yang besar, terutama pada kasus yang kompleks.

IV. INTEGRASI DENGAN PRINSIP DYNAMIC PROGRAMMING

A. Pengertian Dynamic Programming

Dynamic Programming atau Pemrograman Dinamis adalah metode pemecahan masalah dengan cara menguraikan masalah tersebut menjadi tahapan-tahapan sehingga solusi persoalan tersebut dipandang sebagai serangkaian keputusan yang saling terkait.

Adapun pemrograman dinamis memiliki karakteristik seperti:

- Pembagian persoalan menjadi tahap-tahap dimana setiap tahap tersebut hanya diambil satu keputusan
- Masing-masing tahap memiliki sejumlah status yang mungkin terjadi di tahap tersebut. Umumnya status tersebut berupa macam-macam masukan yang bisa diterima di setiap tahap

Pemrograman dinamis juga dapat dilakukan dengan dua jenis pendekatan:

- Maju (*top-down*): Pencarian solusi yang dilakukan dari tahap 1 hingga n
- Mundur (*bottom-up*): Pencarian solusi yang dilakukan dari tahap n hingga 1

B. Penggunaan Dynamic Programming dalam Penjadwalan untuk Optimasi Algoritma Greedy dan Backtracking

Pemrograman dinamis dapat diintegrasikan dengan algoritma greedy dan backtracking untuk meningkatkan efisiensinya dan mengoptimalkan solusi untuk masalah yang kompleks.

Pemrograman dinamis dapat digunakan bersamaan dengan algoritma greedy sebagai berikut:

1. Memoisasi

   Strategi greedy menghasilkan pilihan yang optimal secara lokal tanpa memerhatikan hasil optimal yang berlaku secara global. Hal ini dapat dikembangkan dengan pemrograman dinamis dengan menyimpan hasil tersebut agar hasil optimal tersebut dapat digunakan kembali. Pendekatan seperti demikian mengurangi komputasi yang redundan dan menambah performa algoritma secara umum.
2. Pembagian tahap

   Seperti yang dijelaskan sebelumnya, pemrograman dinamis membagi masalah yang besar menjadi tahap-tahap. Kita dapat mengimplementasikan hal ini bersama dengan algoritma greedy sehingga seolah-olah kita dapat memandu cara kerja algoritma greedy untuk menghasilkan hasil yang lebih optimal dengan lebih konsisten.

Pemrograman dinamis dapat digunakan untuk optimasi algoritma backtracking sebagai berikut:

1. Pruning

   Algoritma backtracking menjelajahi ruang pencarian dengan membangun solusi secara bertahap dan membatalkan pilihan bila diperlukan. Dengan mengintegrasikan pemrograman dinamis, submasalah tertentu dapat dipangkas atau dihilangkan dari ruang pencarian berdasarkan hasil yang diperoleh dari pemrograman dinamis. Proses pemangkasan ini membantu menghindari eksplorasi yang tidak perlu dari jalur yang tidak menjanjikan, mengurangi ruang pencarian, dan meningkatkan efisiensi algoritma backtracking.

   Pemangkasan tersebut dapat dilakukan dengan cara memangkas suatu solusi dan memutuskan untuk backtracking lebih awal jika solusi yang ditemukan sekarang memberikan nilai yang lebih kecil dari hasil yang pernah ditemukan.

2. Pembagian tahap

Algoritma backtracking sering menemui subproblem yang bisa mendapatkan keuntungan dari teknik pemrograman dinamis. Dengan memecahkan submasalah ini secara optimal menggunakan pemrograman dinamis, algoritma backtracking dapat membuat keputusan selama proses eksplorasi. Hasil yang diperoleh dari pemrograman dinamis dapat memandu algoritma backtracking dalam memilih pilihan yang paling menjanjikan dan mencapai solusi optimal atau mendekati optimal secara lebih efisien.

C. Keuntungan dan Batasan penggunaan Dynamic Programming

Mengintegrasikan pemrograman dinamis dengan algoritma greedy dan backtracking memungkinkan solusi yang lebih efisien dan optimal untuk masalah yang kompleks. Teknik pemrograman dinamis seperti memoisasi, pengoptimalan submasalah, dan pemangkasan membantu mengurangi perhitungan yang berlebihan, memandu pilihan algoritma, dan meningkatkan efisiensi dan efektivitas algoritma secara keseluruhan. Integrasi ini memungkinkan kombinasi kekuatan algoritma greedy dan backtracking dengan solusi optimal yang diperoleh melalui pemrograman dinamis.

Di sisi lain dari keuntungan-keuntungan yang bisa didapat dari pemrograman dinamis. Pemrograman dinamis juga dapat memberikan batasan-batasan. Batasan tersebut berupa:

1. Penggunaan memori yang lebih banyak

Implementasi program dinamis yang umum adalah menyimpan hasil optimal pada tahap-tahap di suatu persoalan. Hasil optimal tiap tahap ini yang lalu digunakan kembali untuk mencari hasil optimal total dari suatu persoalan. Jika pada suatu persoalan didapatkan tahapan-tahapan pada jumlah yang banyak, penyimpanan hasil-hasil tersebut di memori akan menumpuk dan akan memengaruhi hasil pencarian solusi.

2. Dekomposisi tahapan

Menentukan bagaimana tahapan-tahapan masalah yang bisa didapat dari suatu persoalan berpotensi menjadi halangan jika persoalan yang ada memiliki struktur sedemikian sehingga mengidentifikasi submasalah dan hubungan dari setiap tahapan menjadi semakin sulit dan membutuhkan analisis yang teliti dan pemahaman persoalan yang matang. Dekomposisi tersebut tidak selalu bisa didapat dengan mudah dan dekomposisi yang tidak tepat akan menghasilkan solusi yang salah dan performa yang tidak optimal.

V. STUDI KASUS

Studi Kasus: Penjadwalan Tugas di Fasilitas Manufaktur

Deskripsi Masalah:

Fasilitas manufaktur menghasilkan banyak produk dengan persyaratan pemrosesan yang berbeda. Fasilitas ini memiliki suatu set mesin dan sejumlah pekerja yang tersedia. Setiap tugas memiliki waktu pemrosesan tertentu dan membutuhkan mesin dan pekerja tertentu untuk menyelesaikannya. Tujuannya adalah untuk menjadwalkan tugas dengan cara yang meminimalkan waktu produksi secara keseluruhan, dengan mempertimbangkan ketersediaan sumber daya dan kendala yang ada sebelumnya.

A. Studi kasus Algoritma Greedy

Strategi Algoritma:

Algoritma greedy digunakan untuk memecahkan masalah penjadwalan tugas. Algoritma ini mengikuti aturan "Earliest Deadline First" (EDF), di mana tugas dengan tenggat waktu paling awal dijadwalkan terlebih dahulu.

Implementasi dan Hasil:

Data produksi fasilitas manufaktur, termasuk perincian tugas, ketersediaan sumber daya, dan batasan prioritas, adalah input ke algoritma. Algoritma secara iteratif memilih tugas dengan tenggat waktu paling awal yang memenuhi ketersediaan sumber daya dan batasan prioritas. Ini memberikan tugas ke mesin dan pekerja yang sesuai, memperbarui jadwal yang sesuai. Proses ini berlanjut hingga semua tugas dijadwalkan.

Dengan menerapkan algoritma greedy, fasilitas manufaktur mencapai peningkatan yang signifikan dalam waktu produksi dan pemanfaatan sumber daya. Algoritma secara optimal menjadwalkan tugas berdasarkan tenggat waktu mereka, memastikan bahwa tugas penting diselesaikan tepat waktu. Ini meminimalkan waktu menganggur untuk mesin dan pekerja dengan mengalokasikan sumber daya secara efisien dan mempertimbangkan kendala prioritas. Jadwal yang dihasilkan memungkinkan fasilitas untuk memenuhi target produksi, mengurangi kemacetan, dan meningkatkan produktivitas secara keseluruhan.

B. Studi kasus Algoritma Backtracking

Strategi Algoritma:

Algoritma backtracking digunakan untuk memecahkan masalah penjadwalan tugas. Algoritma ini secara sistematis menjelajahi ruang pencarian dari kemungkinan jadwal tugas, membuat pilihan, dan melakukan backtracking bila diperlukan. Algoritmaini mempertimbangkan ketersediaan sumber daya, kendala prioritas, dan kriteria optimalitas untuk menemukan jadwal yang optimal atau mendekati optimal.

Implementasi dan Hasil:

Data produksi fasilitas manufaktur, termasuk perincian tugas, ketersediaan sumber daya, dan batasan prioritas, dalam algoritma ini menjadi masukan ke dalam proses pencarian. Prosesnya dimulai dengan jadwal kosong dan baik secara iteratif maupun rekursif mempertimbangkan setiap tugas dalam yang tersedia. Ia memeriksa apakah tugas dapat dijadwalkan berdasarkan ketersediaan sumber daya dan kendala prioritas. Jika bisa, tugas tersebut dijadwalkan, dan algoritma melanjutkan ke tugas berikutnya. Jika tidak, algoritma akan mundur dan mengeksplorasi pilihan alternatif hingga ditemukan jadwal yang valid.

VI. Analisis Komparatif dari Algoritma Greedy dan Backtracking

A. Komparasi algoritma berdasarkan studi kasus

Dari bagaimana kedua algoritma diimplementasikan, berikut hasil perbedaan kedua algoritma:

Tolok ukur	Greedy	Backtracking
Fleksibilitas algoritma	Perlu menentukan heuristik *EDF* untuk menentukan pendekatan pencarian hasil	Tidak perlu menentukan heuristik sehingga dapat mempertimbangkan semua variable
Kerumitan pencarian	Pendekatan greedy yang menggunakan *EDF*, mengeliminasi langkah-langkah redundan dalam pencariannya	Lebih banyak langkah yang dilakukan dengan *Exhaustive Search*
Keoptimalan hasil	Menghasilkan yang mendekati optimal karena melewatkan pertimbangan resource setiap penggunaan mesin	Hasil optimal karena menjelajah semua kemungkinan

Figure 6 Tabel perbandingan algoritma greedy dan backtracking

B. Kesimpulan Perbandingan Algoritma Greedy dan Backtracking

Berdasarkan perbandingan performa kedua algoritma di atas, algoritma greedy memberikan solusi cepat berdasarkan pilihan optimal lokal, sehingga cocok untuk masalah penjadwalan dengan kendala yang lebih sedikit dan ketika solusi mendekati optimal dapat diterima. Di sisi lain, algoritma backtracking secara sistematis menjelajahi ruang pencarian, mempertimbangkan semua kendala dan memberikan solusi optimal atau mendekati optimal dengan mengorbankan kompleksitas komputasi. Pilihan antara dua pendekatan tergantung pada persyaratan khusus, batasan, dan tujuan dari masalah penjadwalan tugas yang ada.

VII. Kesimpulan

Sebagai kesimpulan, makalah ini telah mengeksplorasi penerapan strategi algoritmik dalam kegiatan penjadwalan. Penjelasan di atas membahas beberapa pendekatan algoritmik, seperti algoritma greedy, algoritma backtracking, dan integrasi prinsip pemrograman dinamis. Studi kasus pun telah disajikan untuk menunjukkan aplikasi praktis dari strategi ini dalam berbagai skenario penjadwalan. Penjelasan di atas juga menyinggung bagaimana kita dapat menyelesaikan persoalan dengan memilih algoritma yang sesuai.

Referensi

[1] *Dynamic Programming.* (2023, May 08). Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/dynamic-programming/

[2] Munir, R. (2023). *Algoritma Backtracking Bagian 1*. Retrieved from Website Rinaldi Munir: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf

[3] Munir, R. (2023). *Algoritma Backtracking Bagian 2*. Retrieved from Website Rinaldi Munir: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian2.pdf

[4] Munir, R. (2023). *Algoritma Greedy Bagian 1*. Retrieved from Website Rinaldi Munir: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

[5] Munir, R. (2023). *Algoritma Greedy Bagian 2*. Retrieved from Website Rinaldi Munir: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

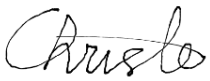
[6] Munir, R. (2023). *Graf Bagian 3*. Retrieved from Website Rinaldi Munir: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian3.pdf

[7] Munir, R. (2023). *Program Dinamis Bagian 1*. Retrieved from Website Rinaldi Munir: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf

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



Christophorus Dharma Winata / 13521009