

Penjadwalan Pengerjaan Tugas Mahasiswa Menggunakan Program Dinamis dengan Tenggat Lunak dan Penalti Perpindahan Konteks

Marcel Luther Sitorus - 13524063

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jalan Ganesha 10 Bandung

E-mail: sitorusmarcelluther@gmail.com , 13524063@std.stei.itb.ac.id

Abstrak—Makalah ini mengusulkan dan mengimplementasikan model program dinamis untuk menyusun jadwal pengerjaan tugas mahasiswa dengan mempertimbangkan waktu rilis, tenggat, estimasi durasi, prioritas akademik, konsekuensi keterlambatan, batas kerja berturut-turut, istirahat, dan perpindahan konteks. Tenggat diperlakukan sebagai kendala lunak: keterlambatan diberi penalti berbobot, sedangkan penyelesaian dini memperoleh imbalan terbatas untuk membentuk buffer waktu. State menyimpan kemajuan seluruh tugas, tugas terakhir, dan streak pengerjaan. Implementasi Java 21 menggunakan sparse dynamic programming, rolling layer, pengodean mixed-radix dengan fallback vektor immutable, pruning, dan parent pointer untuk merekonstruksi jadwal. Pengujian terhadap 16 skenario fungsional menunjukkan seluruh aturan terpenuhi. Pada contoh tiga tugas, seluruh tugas selesai satu jam sebelum tenggat dengan 1.037 state diekspansi dan 843 kandidat state duplikat dieliminasi. Eksperimen skalabilitas menunjukkan pertumbuhan kombinatorial: untuk sembilan tugas berdurasi dua jam, algoritma mengekspansi 177.121 state dan memerlukan median sekitar 1,06 detik. Hasil tersebut menunjukkan bahwa model exact sesuai untuk kumpulan tugas aktif berukuran kecil hingga menengah.

Kata kunci—program dinamis; penjadwalan tugas; tenggat lunak; penalti keterlambatan; penyelesaian dini; perpindahan konteks

I. PENDAHULUAN

Mahasiswa pada umumnya menerima beberapa tugas dalam periode yang saling bertumpang tindih. Setiap tugas dapat berasal dari mata kuliah dengan jumlah satuan kredit semester yang berbeda, memiliki bobot penilaian yang berbeda, membutuhkan estimasi waktu pengerjaan yang berbeda, dan menerapkan konsekuensi keterlambatan yang tidak selalu sama. Keputusan mengenai tugas mana yang dikerjakan lebih dahulu dengan demikian tidak cukup dibuat hanya berdasarkan tenggat terdekat. Tugas yang tenggatnya masih jauh dapat memiliki bobot akademik tinggi, sedangkan tugas dengan tenggat dekat dapat hanya membutuhkan durasi singkat. Selain itu, jadwal yang terlalu sering berganti tugas dapat memecah konsentrasi dan menambah waktu penyesuaian mental.

Secara abstrak, seorang mahasiswa dapat dipandang sebagai satu sumber daya yang memproses sekumpulan pekerjaan. Pemodelan ini memiliki kemiripan dengan single-machine scheduling, yaitu setiap pekerjaan hanya diproses oleh satu mesin dan pada satu waktu mesin hanya mengerjakan satu pekerjaan [3]. Akan tetapi, penjadwalan tugas mahasiswa mempunyai karakteristik tambahan yang tidak selalu muncul pada model produksi klasik. Pengerjaan dapat dipotong menjadi beberapa sesi, satu sesi minimal berdurasi satu jam, satu tugas tidak boleh dikerjakan lebih dari tiga jam berturut-turut, dan setelah batas tersebut diperlukan satu jam istirahat. Model juga perlu mempertimbangkan biaya perpindahan konteks karena perpindahan dari satu tugas ke tugas lain tidak bersifat gratis secara kognitif [9]–[11].

Pendekatan greedy seperti mendahulukan tenggat terdekat, durasi terpendek, atau prioritas terbesar dapat menghasilkan jadwal dengan cepat. Namun, keputusan lokal tersebut tidak mempertimbangkan interaksi antar-tugas dalam keseluruhan horizon. Mengutamakan tugas berprioritas tinggi tanpa memperhatikan durasi dan tenggat dapat menyebabkan tugas lain terlambat; sebaliknya, selalu memilih tenggat terdekat dapat memecah satu tugas panjang menjadi terlalu banyak blok dan memperbesar perpindahan konteks. Permasalahan penjadwalan dengan fungsi biaya tenggat pada umumnya juga memiliki kompleksitas komputasional tinggi [5]. Oleh karena itu, diperlukan metode yang mengevaluasi akibat jangka panjang dari setiap keputusan.

Makalah ini memilih program dinamis sebagai strategi algoritma. Program dinamis menyelesaikan masalah melalui kumpulan submasalah yang saling bertumpang tindih dan menggunakan prinsip optimalitas, yaitu kelanjutan optimal dari suatu keadaan hanya bergantung pada informasi keadaan tersebut, bukan pada seluruh lintasan historisnya [1], [2]. Kontribusi makalah ini bukan sekadar merangkum literatur penjadwalan, melainkan merumuskan model program dinamis khusus penjadwalan tugas mahasiswa yang menggabungkan prioritas akademik, tenggat lunak, penalti keterlambatan, reward penyelesaian dini, batas tiga jam kerja berturut-turut, istirahat wajib, dan penalti perpindahan konteks.

Kontribusi teknis utama terdiri atas empat bagian. Pertama, dirumuskan fungsi objektif yang membedakan prioritas

akademik dan konsekuensi keterlambatan, sekaligus memberi reward terbatas terhadap buffer sebelum tenggat. Kedua, dirancang state minimal yang tetap memuat informasi untuk menegakkan aturan kerja dan menghitung perpindahan konteks. Ketiga, model direalisasikan sebagai sparse dynamic programming dengan rolling layer, pengodean mixed-radix, fallback vektor immutable, dan rekonstruksi parent pointer. Keempat, implementasi divalidasi melalui pengujian fungsional, studi sensitivitas parameter, dan pengukuran pertumbuhan state.

II. LANDASAN TEORI DAN PEKERJAAN TERKAIT

A. Program Dinamis

Program dinamis merupakan teknik optimasi yang membangun solusi suatu masalah dari solusi submasalah. Dua syarat yang umum digunakan adalah optimal substructure dan overlapping subproblems [2]. Optimal substructure berarti solusi optimal dapat disusun dari solusi optimal submasalah, sedangkan overlapping subproblems berarti submasalah yang sama muncul berulang kali. Bellman memperkenalkan prinsip optimalitas sebagai dasar formulasi keputusan bertahap [1]. Dalam implementasi, nilai terbaik setiap state disimpan agar state yang sama tidak dihitung kembali melalui seluruh lintasan yang berbeda.

Pada persoalan penjadwalan, sebuah state harus merangkum seluruh informasi masa lalu yang masih memengaruhi keputusan masa depan. Apabila state terlalu sedikit, dua riwayat yang sebenarnya berbeda dapat digabung secara keliru. Apabila state terlalu banyak, kebutuhan waktu dan ruang meningkat tajam. Karena itu, rancangan state merupakan bagian terpenting dari model. Pemisahan state berdasarkan lapisan waktu juga memungkinkan penggunaan rolling layer, sehingga hanya state pada waktu sekarang dan waktu berikutnya yang harus disimpan [12].

B. Tenggat, Keterlambatan, dan Penyelesaian Dini

Dalam teori penjadwalan, completion time suatu pekerjaan dinotasikan dengan C_i dan due date dengan d_i . Tardiness didefinisikan sebagai bagian positif dari selisih completion time dan due date. Dengan definisi tersebut, pekerjaan yang selesai sebelum tenggat memiliki tardiness nol, sedangkan pekerjaan yang terlambat memperoleh biaya sesuai lama keterlambatannya [3]. Model weighted tardiness memberikan bobot berbeda pada setiap pekerjaan, sehingga keterlambatan pekerjaan penting menghasilkan biaya lebih besar. Algoritma program dinamis klasik juga telah digunakan untuk beberapa varian penjadwalan bertenggat [4].

Penelitian tentang earliness-tardiness menunjukkan bahwa penyelesaian sebelum atau sesudah due date dapat dimasukkan dalam satu fungsi objektif [6], [7]. Pada lingkungan produksi, penyelesaian terlalu awal kadang dianggap sebagai biaya inventori. Pada konteks mahasiswa, interpretasinya dibalik: penyelesaian lebih awal merupakan keuntungan karena menyediakan buffer untuk revisi, gangguan tak terduga, dan ketidakakuratan estimasi durasi. Namun, reward harus dibatasi. Tanpa batas, algoritma dapat terlalu mengutamakan tugas yang tenggatnya sangat jauh dan mengorbankan keseimbangan jadwal.

C. Perpindahan Konteks

Perpindahan konteks terjadi ketika mahasiswa berhenti mengerjakan satu tugas dan memulai tugas lain. Penelitian psikologi kognitif menunjukkan adanya switch cost yang terlihat dari peningkatan waktu respons atau penurunan akurasi setelah pergantian tugas [9], [10]. Studi pada pekerjaan pengembangan perangkat lunak juga melaporkan bahwa perpindahan tugas dapat memutus kondisi fokus dan memerlukan waktu untuk kembali memahami konteks kerja [11]. Oleh sebab itu, jumlah perpindahan tidak hanya diperlakukan sebagai preferensi tampilan jadwal, tetapi sebagai komponen biaya yang eksplisit.

Dalam literatur scheduling, gagasan serupa muncul sebagai sequence-dependent setup time atau setup cost, yaitu biaya menyiapkan mesin bergantung pada pekerjaan sebelumnya [8]. Makalah ini mengadaptasi gagasan tersebut menjadi biaya kognitif. Istirahat tidak menghapus identitas tugas terakhir. Dengan demikian, urutan A, istirahat, B tetap dihitung sebagai satu perpindahan, sedangkan A, istirahat, A tidak menimbulkan perpindahan baru.

III. FORMULASI MASALAH

A. Representasi Tugas

Terdapat n tugas yang diberi indeks i dari 1 sampai n . Setiap tugas direpresentasikan sebagai tuple A_i yang memuat identitas tugas, waktu mulai tersedia, tenggat, estimasi durasi pengerjaan, jumlah SKS mata kuliah, persentase kontribusi tugas terhadap nilai akhir, dan koefisien konsekuensi keterlambatan. Seluruh waktu dinyatakan relatif terhadap awal horizon dan dibulatkan ke slot satu jam.

$$A_i = (id_i, r_i, d_i, p_i, s_i, g_i, c_i) \quad (1)$$

Variabel r_i adalah release time, d_i adalah tenggat, p_i adalah kebutuhan jam pengerjaan, s_i adalah jumlah SKS, g_i adalah bobot tugas terhadap nilai akhir, dan c_i adalah koefisien konsekuensi keterlambatan. Persentase nilai disimpan dalam representasi konsisten, misalnya 0,20 untuk dua puluh persen. Prioritas akademik dihitung dari aturan yang diajukan sebagai berikut.

$$w_i = s_i * g_i \quad (2)$$

Pemisahan w_i dan c_i diperlukan karena keduanya mempunyai makna berbeda. Nilai w_i menunjukkan pentingnya tugas terhadap beban dan evaluasi mata kuliah, sedangkan c_i menggambarkan seberapa berat konsekuensi apabila tugas terlambat. Dua tugas dapat mempunyai prioritas akademik sama, tetapi kebijakan keterlambatan berbeda.

TABLE I. NOTASI MODEL PENJADWALAN

Simbol	Arti	Domain
r_i	Waktu tugas tersedia	bilangan bulat
d_i	Tenggat tugas	bilangan bulat
p_i	Jam pengerjaan yang diperlukan	bilangan bulat positif
w_i	Bobot prioritas akademik	bilangan nonnegatif
c_i	Konsekuensi keterlambatan	bilangan nonnegatif
C_i	Waktu tugas selesai	bilangan bulat
a_i	Progress tugas pada state	0 sampai p_i

B. Asumsi dan Aturan Penjadwalan

Model menggunakan asumsi deterministik: seluruh tugas, tenggat, release time, dan estimasi durasi diketahui ketika algoritma dijalankan. Mahasiswa hanya dapat mengerjakan satu tugas pada satu slot. Pengerjaan satu tugas bersifat preemptive dalam arti dapat dihentikan dan dilanjutkan, tetapi titik pemotongan hanya berada pada batas satu jam. Tidak ada dua tugas yang dikerjakan bersamaan, sehingga multitasking simultan tidak diperbolehkan.

Satu tugas dapat dikerjakan selama satu sampai tiga jam berturut-turut. Setelah tiga jam berturut-turut pada tugas yang sama, slot berikutnya wajib menjadi istirahat. Setelah istirahat, tugas yang sama boleh dilanjutkan. Aturan ini diperlakukan sebagai hard constraint karena jadwal yang melanggarnya tidak masuk ke ruang solusi. Sebaliknya, tenggat diperlakukan sebagai soft constraint. Tugas tetap boleh selesai setelah d_i , tetapi memperoleh penalti.

Model belum memasukkan jadwal kuliah, waktu tidur, dependensi antartugas, atau variasi produktivitas menurut waktu. Batasan tersebut dapat dimasukkan pada pengembangan berikutnya sebagai availability mask, precedence constraint, atau produktivitas per slot. Pembatasan ruang lingkup ini diperlukan agar kontribusi utama tetap terfokus pada formulasi program dinamis.

C. Fungsi Objektif

Untuk setiap tugas, keterlambatan T_i dan buffer penyelesaian B_i didefinisikan sebagai berikut.

$$T_i = \max(0, C_i - d_i) \quad (3)$$

$$B_i = \max(0, d_i - C_i) \quad (4)$$

Penalti keterlambatan memperhitungkan prioritas dan konsekuensi keterlambatan. Reward penyelesaian dini memperhitungkan prioritas, tetapi dibatasi oleh K_i sebagai buffer ideal maksimum yang masih bernilai. Tiga koefisien global digunakan: alpha untuk keterlambatan, beta untuk penyelesaian dini, dan lambda untuk perpindahan konteks.

$$\text{Late}_i = \alpha * w_i * c_i * T_i \quad (5)$$

$$\text{Early}_i = \beta * w_i * \min(B_i, K_i) \quad (6)$$

Jika S adalah jumlah perpindahan konteks, fungsi objektif total adalah meminimalkan penalti keterlambatan dan perpindahan, kemudian menguranginya dengan reward penyelesaian dini.

$$F = \sum \text{Late}_i + \lambda * S - \sum \text{Early}_i \quad (7)$$

Nilai alpha yang besar membuat algoritma sangat menghindari keterlambatan. Nilai beta yang besar mendorong pembentukan buffer sebelum tenggat. Nilai lambda yang besar menghasilkan blok pengerjaan lebih panjang dan lebih sedikit perpindahan. Pemilihan koefisien harus dilakukan pada skala yang sebanding agar salah satu tujuan tidak mendominasi secara tidak disengaja.

IV. MODEL PROGRAM DINAMIS YANG DIUSULKAN

A. Diskritisasi Waktu dan Horizon

Waktu dibagi menjadi slot satu jam $t = 0, 1, 2$, sampai H . Pada setiap slot, algoritma memilih satu tugas untuk dikerjakan atau memilih istirahat. Karena tenggat merupakan kendala lunak, H tidak boleh berhenti pada tenggat terbesar. Horizon harus cukup panjang untuk menyelesaikan seluruh pekerjaan meskipun semua tugas baru dikerjakan setelah release time terbesar.

$$H = \max r_i + \sum p_i + \sum \text{floor}(p_i / 3) \quad (8)$$

Persamaan tersebut merupakan batas atas konservatif. Suku pertama menyediakan waktu menunggu sampai semua tugas tersedia, suku kedua memuat seluruh jam kerja, dan suku ketiga menyediakan jam istirahat setelah blok tiga jam. Dalam praktik, sebagian istirahat tidak dibutuhkan apabila algoritma berganti tugas sebelum streak mencapai tiga atau ketika seluruh pekerjaan telah selesai. Kelebihan horizon hanya menambah state idle yang dapat dipangkas.

B. Definisi State

State konseptual pada waktu t dituliskan sebagai $DP[t, a, ell, q]$. Vektor a berisi progress setiap tugas, ell adalah tugas terakhir yang dikerjakan, dan q adalah jumlah jam berturut-turut pada tugas tersebut. Nilai DP menyimpan biaya minimum untuk mencapai kombinasi tersebut.

$$\text{State}_t = (a_1, a_2, \dots, a_n, ell, q) \quad (9)$$

Untuk setiap i berlaku 0 kurang dari atau sama dengan a_i dan a_i kurang dari atau sama dengan p_i . Nilai ell dapat berupa indeks tugas atau NONE jika belum pernah ada tugas yang dikerjakan. Nilai q berada pada 0 sampai 3. Setelah istirahat, q menjadi nol tetapi ell tidak dihapus. Keputusan ini penting agar A , istirahat, B tetap dikenai penalti perpindahan.

State tersebut cukup untuk keputusan masa depan. Vektor progress menentukan sisa pekerjaan, waktu t menentukan tugas yang sudah tersedia dan besarnya biaya completion, ell menentukan apakah aksi berikutnya merupakan perpindahan, sedangkan q menentukan apakah bekerja masih diperbolehkan atau wajib istirahat. Informasi lain dari lintasan masa lalu tidak memengaruhi feasibility maupun biaya lanjutan.

C. Transisi

Dari setiap state, terdapat dua kelompok aksi. Aksi pertama adalah mengerjakan tugas i selama satu jam. Aksi valid apabila t tidak lebih kecil dari r_i , progress a_i masih lebih kecil dari p_i , dan state tidak sedang mewajibkan istirahat. Jika i sama dengan ell , streak baru adalah q ditambah satu. Jika i berbeda dari ell , streak baru menjadi satu dan penalti λ ditambah, kecuali ell masih NONE.

Implementasi membedakan tiga jenis aksi, yaitu WORK, REST, dan IDLE. Apabila q sama dengan tiga, REST wajib menjadi satu-satunya aksi valid. Setelah REST, progress tidak berubah, q menjadi nol, dan ell tetap menyimpan tugas terakhir. IDLE dihasilkan ketika belum ada tugas yang tersedia atau ketika strategic idle diizinkan untuk menunggu release time tugas mendatang. Istirahat opsional juga tersedia sebagai parameter konfigurasi, tetapi dinonaktifkan pada konfigurasi bawaan. Baik REST maupun IDLE tidak menghapus ell sehingga biaya perpindahan konteks tetap dapat dihitung dengan benar.

Ketika aksi kerja menaikkan a_i dari p_i dikurangi satu menjadi p_i , tugas i selesai pada waktu C_i sama dengan t ditambah satu. Pada transisi tersebut, $Late_i$ atau $Early_i$ dihitung tepat satu kali. Setelah a_i mencapai p_i , tidak ada transisi kerja berikutnya untuk tugas tersebut sehingga biaya completion tidak mungkin dihitung ulang.

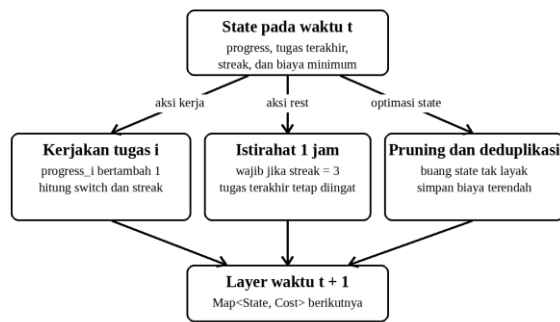


Fig. 1. Alur pembentukan satu lapisan state pada model program dinamis.

D. Inisialisasi, Kondisi Akhir, dan Rekonstruksi

State awal mempunyai seluruh progress nol, ell sama dengan NONE, q sama dengan nol, dan biaya nol. State terminal adalah state dengan a_i sama dengan p_i untuk seluruh tugas. Dari seluruh state terminal yang mungkin muncul, algoritma memilih state dengan nilai F minimum. Apabila beberapa state memiliki biaya sama, tie-breaking dapat memprioritaskan jumlah jam terlambat yang lebih kecil, jumlah perpindahan yang lebih sedikit, dan makespan yang lebih awal.

Untuk memperoleh jadwal, bukan hanya nilai optimal, setiap state yang dipertahankan perlu dikaitkan dengan parent state dan aksi pembentuknya. Setelah state terminal terbaik dipilih, lintasan ditelusuri mundur sampai state awal. Urutan terbalik tersebut kemudian dibalik dan slot berurutan dengan tugas sama digabung menjadi blok agar hasil mudah dibaca. Penyimpanan parent untuk semua lapisan meningkatkan penggunaan ruang; alternatifnya adalah menjalankan pass kedua setelah nilai optimum ditemukan.

V. REPRESENTASI DAN OPTIMASI STATE

A. Sparse Dynamic Programming dan Rolling Layer

Representasi langsung sebagai larik $DP[t][a_1][a_2]$ sampai $[a_n][ell][q]$ mengalokasikan seluruh kombinasi, termasuk state yang tidak pernah tercapai. Karena release time, batas streak, dan keterbatasan total jam membuat banyak kombinasi tidak valid, pendekatan tersebut boros. Model implementasi yang diusulkan menggunakan hash map dengan key berupa state dan value berupa biaya minimum.

Algoritma hanya membutuhkan layer waktu t untuk menghasilkan layer t ditambah satu. Oleh karena itu, dua map digunakan secara bergantian, yaitu $currentLayer$ dan $nextLayer$. Jika dua lintasan menghasilkan key yang sama, hanya lintasan berbiaya lebih rendah yang dipertahankan. Deduplikasi ini merupakan penerapan langsung prinsip optimalitas dan merupakan sumber penghematan utama dibanding enumerasi seluruh jadwal.

B. Pengodean Mixed-Radix

Vektor progress dapat dikodekan menjadi satu bilangan menggunakan basis campuran. Setiap digit a_i mempunyai basis p_i ditambah satu. Pengodean ini menghindari penggunaan array mutable sebagai key dan mempercepat hashing untuk ukuran instance moderat.

$$code(a) = a_1 + a_2(p_1+1) + \dots + a_n \text{ product}_{j<n}(p_j+1) \quad (10)$$

Pengodean bersifat satu-ke-satu selama 0 sampai p_i menjadi domain masing-masing digit. Sebelum menggunakan bilangan 64-bit, implementasi perlu memeriksa apakah hasil kali seluruh basis melebihi kapasitas. Jika terjadi overflow, key harus diganti dengan tuple immutable atau bilangan presisi arbitrer. Pengodean hanya mengubah representasi, bukan banyaknya state secara asimtotik.

C. Pruning dan Dominasi

Pruning pertama memeriksa kecukupan horizon. Implementasi menghitung jumlah jam kerja yang masih tersisa dan membandingkannya dengan sisa slot; jika q telah mencapai tiga, satu slot langsung dicadangkan untuk REST wajib. Pemeriksaan ini aman, tetapi konservatif karena belum memperkirakan seluruh istirahat wajib yang mungkin muncul pada masa depan. Pruning kedua adalah release-time pruning: tugas yang belum tersedia tidak menghasilkan transisi WORK. Pruning ketiga adalah dominance pruning untuk key identik; hanya $BestValue$ yang lebih baik yang dipertahankan berdasarkan biaya, total jam terlambat, jumlah perpindahan, dan total completion time.

Pruning yang lebih agresif harus digunakan secara hati-hati. Dua state dengan progress sama tetapi ell atau q berbeda tidak selalu dapat dibandingkan karena biaya perpindahan dan kewajiban istirahat masa depan berbeda. Menghapus salah satunya hanya berdasarkan biaya saat ini dapat merusak optimalitas. Untuk instance besar, beam search dapat digunakan sebagai mode aproksimasi dengan menyimpan sejumlah state terbaik per layer, tetapi hasilnya tidak lagi dijamin optimal.

TABLE II. KOMPONEN STATE DAN ALASAN PENYIMPANANNYA

Komponen	Fungsi	Jika dihilangkan
progress	menentukan sisa durasi seluruh tugas	state dengan sisa kerja berbeda tergabung
last task	menghitung perpindahan konteks	switch cost tidak dapat dihitung tepat
streak	menegakkan batas tiga jam	jadwal dapat melanggar aturan istirahat
time layer	release time dan completion time	deadline dan ketersediaan tidak terbaca

VI. ANALISIS KEBENARAN

A. Kecukupan State

Lemma 1. Untuk waktu t yang sama, dua lintasan yang menghasilkan progress, tugas terakhir, streak, dan biaya akumulatif yang sama memiliki himpunan aksi masa depan dan biaya tambahan yang sama. Bukti. Seluruh kendala masa depan ditentukan oleh sisa durasi, release time yang dibandingkan dengan t , aturan istirahat dari streak, serta status perpindahan dari tugas terakhir. Tidak ada informasi lintasan lain yang digunakan dalam transisi atau fungsi biaya. Dengan demikian, salah satu lintasan dapat menggantikan lintasan lain tanpa mengubah kelanjutan yang mungkin.

B. Kelengkapan Transisi

Lemma 2. Setiap jadwal valid dapat direpresentasikan sebagai lintasan dari state awal ke state terminal. Bukti. Jadwal valid terdiri atas deretan slot kerja atau istirahat. Untuk setiap slot kerja, tugas telah tersedia, belum selesai, dan tidak melanggar streak, sehingga transisi kerja tersedia. Untuk setiap slot istirahat, transisi istirahat tersedia; khusus setelah streak tiga, transisi tersebut diwajibkan. Dengan induksi terhadap jumlah slot, seluruh prefiks jadwal dapat dipetakan pada state DP.

C. Ketepatan Akumulasi Biaya

Lemma 3. Penalti keterlambatan atau reward penyelesaian dini setiap tugas dihitung tepat satu kali. Bukti. Biaya tersebut hanya ditambahkan ketika progress berubah dari p_i dikurangi satu menjadi p_i . Peristiwa itu hanya dapat terjadi satu kali karena progress tidak pernah menurun dan tugas yang selesai tidak dapat dikerjakan lagi. Penalti perpindahan juga dihitung tepat pada transisi pertama menuju tugas yang berbeda dari ell .

D. Optimalitas

Teorema 1. State terminal dengan biaya minimum yang dihasilkan recurrence merupakan jadwal optimal terhadap fungsi F . Bukti. Berdasarkan Lemma 2, setiap jadwal valid muncul sebagai suatu lintasan. Berdasarkan Lemma 3, biaya lintasan sama dengan nilai fungsi objektif jadwal. Berdasarkan Lemma 1, untuk state identik cukup dipertahankan lintasan dengan biaya terkecil, sebab seluruh kelanjutan mempunyai biaya tambahan yang sama. Dengan induksi per layer waktu, DP menyimpan biaya minimum untuk setiap state. Karena itu, minimum di antara state terminal sama dengan biaya jadwal global optimal.

VII. ANALISIS KOMPLEKSITAS

Misalkan P adalah jumlah seluruh kombinasi progress. Karena a_i memiliki p_i ditambah satu kemungkinan, P merupakan hasil kali seluruh p_i ditambah satu. Untuk setiap progress terdapat paling banyak n ditambah satu nilai tugas terakhir dan empat nilai streak. Maka jumlah state teoritis per layer adalah $O(Pn)$. Setiap state mencoba paling banyak n aksi kerja dan satu aksi istirahat.

$$P = \text{product}_{(i=1..n)} (p_i + 1) \quad (11)$$

$$\text{Waktu} = O(H * P * n^2) \quad (12)$$

$$\text{Ruang rolling layer} = O(P * n) \quad (13)$$

Batas tersebut merupakan worst case. Sparse map biasanya menyimpan jauh lebih sedikit state karena tidak seluruh kombinasi progress dapat dicapai pada setiap t . Meskipun demikian, pertumbuhan P bersifat eksponensial terhadap jumlah tugas apabila durasi dianggap bagian dari input. Hal ini selaras dengan kesulitan umum masalah scheduling bertenggat [5]. Model exact karena itu paling sesuai untuk kumpulan tugas aktif kecil sampai menengah. Penyimpanan parent pointer untuk rekonstruksi dapat meningkatkan ruang menjadi $O(H P n)$, kecuali digunakan rekonstruksi dua tahap.

VIII. PERBANDINGAN DENGAN PENDEKATAN GREEDY

Algoritma greedy memilih aksi berdasarkan informasi lokal, misalnya earliest due date, prioritas terbesar, durasi terpendek, atau rasio prioritas terhadap durasi. Kelebihannya adalah waktu komputasi kecil dan implementasi sederhana. Pada varian scheduling tertentu, aturan greedy memang optimal, tetapi optimalitas tersebut bergantung pada bentuk objektif dan kendala yang spesifik [3]. Ketika release time, penalti keterlambatan berbobot, reward penyelesaian dini, context switching, dan istirahat digabungkan, satu aturan lokal tidak cukup mewakili seluruh konsekuensi.

Sebagai contoh konseptual, memilih tugas dengan tenggat terdekat dapat menguntungkan satu tugas pendek, tetapi menyebabkan tugas berbobot tinggi kehilangan buffer dan kemudian terlambat setelah diselingi istirahat wajib. Memilih prioritas terbesar dapat menghasilkan kebalikan: tugas penting dikerjakan terlalu lama sehingga beberapa tugas kecil melewati tenggat. Selain itu, greedy yang mengevaluasi setiap slot secara terpisah cenderung melakukan perpindahan berulang saat skor urgensi berubah sedikit.

Program dinamis mengevaluasi kombinasi keputusan sampai horizon dan menggabungkan lintasan yang tiba pada state sama. Keunggulannya adalah jaminan optimalitas untuk model dan parameter yang diberikan. Kekurangannya adalah kebutuhan komputasi yang jauh lebih besar. Karena itu, greedy tetap relevan sebagai baseline eksperimen, pembentuk upper bound, atau heuristik untuk mengurutkan ekspansi state, tetapi bukan pengganti exact DP pada kontribusi utama makalah ini.

TABLE III. PERBANDINGAN PROGRAM DINAMIS DAN GREEDY

Aspek	Program dinamis	Greedy
Pandangan keputusan	global terhadap horizon	lokal pada langkah saat ini
Jaminan solusi	optimal untuk model exact	umumnya tanpa jaminan
Context switching	terintegrasi dalam state	perlu aturan tambahan
Kompleksitas	tinggi, bergantung state	rendah
Peran yang disarankan	metode utama	baseline atau heuristik

IX. IMPLEMENTASI DAN PENGUJIAN

A. Implementasi

Model direalisasikan menggunakan Java 21 dalam proyek Maven. Implementasi terdiri atas 19 berkas sumber utama yang dipisahkan menjadi representasi data, konfigurasi, mesin program dinamis, kalkulator biaya, codec progress, validator masukan, rekonstruksi jadwal, dan statistik. Assignment menyimpan identitas, SKS, bobot nilai, release time, tenggat, kebutuhan jam, konsekuensi keterlambatan, serta batas reward dini opsional. SchedulerConfig menyediakan koefisien alpha, beta, lambda, batas reward global, kebijakan strategic idle, istirahat opsional, dan override horizon.

Metode utama membentuk ProgressCodec dari seluruh kebutuhan jam, menghitung horizon, lalu memulai state dengan progress nol, lastWorkedTask bernilai NONE, dan streak nol. Nilai pada setiap lapisan disimpan dalam LinkedHashMap yang memetakan State ke BestValue. Waktu t berada pada loop luar sehingga tidak ikut dimasukkan ke key. Codec menggunakan mixed-radix berbasis long apabila hasil kali seluruh radix aman; jika terjadi potensi overflow, implementasi beralih ke VectorProgressKey immutable dengan equality berbasis isi.

Pada setiap state, scheduler mendekode progress dan menghasilkan seluruh aksi WORK yang valid. Ketika streak mencapai tiga, hanya REST wajib yang dibangkitkan. Jika tidak ada pekerjaan yang tersedia, scheduler menghasilkan IDLE; strategic idle juga dapat dipilih ketika terdapat tugas yang baru tersedia pada masa depan. Biaya perpindahan ditambahkan apabila indeks tugas baru berbeda dari lastWorkedTask. Penalti keterlambatan atau reward penyelesaian dini ditambahkan tepat ketika progress tugas mencapai requiredHours.

Deduplikasi dilakukan setiap kali beberapa lintasan mencapai State yang sama. Kandidat dipilih berdasarkan biaya objektif terendah; apabila biaya setara dalam toleransi numerik, urutan pembandingnya adalah total jam terlambat, jumlah context switch, kemudian total completion time. ParentPointer disimpan untuk setiap state terpilih pada tiap lapisan agar jadwal optimal dapat ditelusuri mundur. Hasil akhir memuat jadwal per jam, blok berurutan, status setiap tugas, nilai objektif, jumlah perpindahan, serta statistik state dan transisi.

B. Skenario Pengujian

Konfigurasi bawaan menggunakan lateWeight 10, earlyWeight 1, switchPenalty 2, dan batas reward dini 24 jam. Strategic idle diaktifkan, sedangkan istirahat opsional dinonaktifkan. Masukan dengan ID duplikat, durasi negatif, bobot nilai di luar rentang nol sampai satu, atau tenggat sebelum release time ditolak oleh validator. Suite pengujian memuat 16 metode JUnit 5 yang mencakup aturan satu sampai tiga jam, REST wajib, release time, prioritas, status dini-tepat waktu-terlambat, perilaku context switch melewati REST, deduplikasi state, rekonstruksi, dan validasi masukan.

Studi kasus utama menggunakan tiga tugas yang juga digunakan pada program contoh. Seluruh waktu dinyatakan relatif terhadap awal horizon. Nilai prioritas dihitung sebagai SKS dikalikan bobot nilai. Parameter masukan ditunjukkan pada Tabel IV.

TABLE IV. DATA STUDI KASUS UTAMA

ID	r i/d i	p_i	SKS	g i/c i
A	0/7	4	3	0,30/2,0
B	1/5	2	2	0,25/1,5
C	4/10	3	3	0,20/2,5

Selain pengujian fungsional dan studi kasus utama, dilakukan dua eksperimen tambahan. Sensitivitas lambda diuji pada dua tugas dengan alpha tetap 10 dan beta tetap 1. Skalabilitas diuji untuk satu sampai sembilan tugas, masing-masing berdurasi dua jam dan release time nol, dengan variasi tenggat, SKS, bobot nilai, dan konsekuensi yang deterministik.

C. Hasil dan Analisis Pengujian

Seluruh 16 skenario pengujian fungsional berhasil. Hasil ini memverifikasi bahwa scheduler menegakkan batas maksimum tiga jam, menyisipkan REST wajib, mempertahankan lastWorkedTask melewati REST, membedakan EARLY, ON_TIME, dan LATE, memilih prioritas berdasarkan biaya total, menolak masukan tidak valid, mempertahankan kandidat termurah untuk state duplikat, dan merekonstruksi jadwal optimal menjadi keluaran per jam maupun blok.

Pada studi kasus utama, scheduler menghasilkan urutan A, A, B, B, A, A, C, C, C. Tidak terdapat slot idle atau REST wajib karena setiap blok kerja berurutan tidak melebihi tiga jam. Seluruh tugas selesai satu jam sebelum tenggat. Rincian completion time ditunjukkan pada Tabel V.

TABLE V. HASIL STUDI KASUS UTAMA

Tugas	Slot kerja	C i/d i	Status; reward
A	0-2; 4-6	6/7	Dini 1 jam; 0,90
B	2-4	4/5	Dini 1 jam; 0,50
C	6-9	9/10	Dini 1 jam; 0,60

Jadwal tersebut memiliki tiga context switch dengan biaya total 6. Jumlah reward dini adalah 2 sehingga nilai objektif akhir menjadi 4. Scheduler mengekskansi 1.037 state, membangkitkan 1.926 transisi, dan menemukan 843 kandidat yang menuju key state yang telah ada. Pengodean mixed-radix long dapat digunakan karena produk radix masih berada dalam batas 64-bit.

Eksperimen sensitivitas memperlihatkan interaksi antara reward dini dan biaya perpindahan. Dengan α 10, β 1, dan λ 0, jadwal yang dipilih adalah A-B-A dengan dua perpindahan dan nilai objektif -4,2. Ketika λ dinaikkan menjadi 2, scheduler memilih IDLE-B-A-A, hanya melakukan satu perpindahan, dan memperoleh nilai objektif -1,6. IDLE pada awal jadwal bukan kegagalan utilisasi, melainkan keputusan global untuk menunggu B tersedia sehingga A tidak perlu dipotong menjadi dua blok. Ambang perubahan tersebut khusus untuk instance ini dan tidak berlaku universal.

Tabel VI menunjukkan pertumbuhan jumlah state dan waktu eksekusi. Setiap penambahan dua tugas meningkatkan ruang kombinasi secara tajam. Pada sembilan tugas, terdapat 177.121 state yang diekspansi dan lebih dari satu juta transisi; median waktu mencapai sekitar 1,06 detik. Sebanyak 826.632 kandidat transisi bertemu key yang sudah tersimpan, sehingga deduplikasi sparse map memberikan penghematan penting walaupun tidak mengubah kompleksitas eksponensial.

TABLE VI. PERTUMBUHAN STATE DAN WAKTU EKSEKUSI

n	State	Transisi	Median (ms)
1	2	2	0,267
3	73	132	0,910
5	1.201	3.630	11,714
7	15.289	66.304	62,568
9	177.121	1.003.770	1.058,276

X. PEMBAHASAN

Model yang diusulkan memperlihatkan bahwa aturan penjadwalan yang tampak sederhana dapat memerlukan state multidimensi. Progress seluruh tugas diperlukan karena keputusan berikutnya bergantung pada sisa durasi setiap tugas. Tugas terakhir dan streak diperlukan bukan untuk mengetahui seluruh riwayat, melainkan untuk merangkum bagian riwayat yang masih memengaruhi biaya dan feasibility. Inilah alasan state awal yang hanya menggunakan waktu dan tugas aktif belum memadai.

Penggunaan tenggat sebagai soft constraint membuat model tetap menghasilkan jadwal ketika beban tidak mungkin diselesaikan tepat waktu. Hal ini lebih realistis daripada membuang seluruh solusi yang terlambat. Koefisien konsekuensi memungkinkan perbedaan kebijakan, misalnya tugas yang langsung tidak diterima setelah tenggat dapat diberi c_i sangat besar. Namun, karena tetap soft, algoritma masih dapat memilih keterlambatan apabila alternatifnya menghasilkan total biaya lebih buruk.

Reward penyelesaian dini membedakan jadwal yang sama-sama tidak terlambat. Tanpa reward, menyelesaikan tugas satu jam sebelum tenggat dan dua hari sebelum tenggat dapat bernilai sama selama komponen lain identik. Buffer terbatas K_i menghindari dorongan yang berlebihan. Dengan demikian, model mengoptimalkan keamanan terhadap tenggat, bukan sekadar status terlambat atau tidak terlambat.

Batas tiga jam dan istirahat satu jam adalah aturan kebijakan yang dipilih dalam masalah, bukan klaim universal mengenai kapasitas belajar semua mahasiswa. Nilainya dapat diparameterkan pada pengembangan berikutnya. Begitu pula,

penalti perpindahan saat ini bersifat konstan. Model yang lebih kaya dapat menggunakan λ_{ij} agar perpindahan antara dua tugas yang sangat berbeda lebih mahal daripada perpindahan antara tugas sejenis.

Keterbatasan utama adalah ledakan state. Pengodean mixed-radix dan sparse map mengurangi overhead penyimpanan, tetapi tidak mengubah sifat kombinatorial. Untuk penggunaan semester penuh, sistem dapat memilih sejumlah tugas paling mendesak berdasarkan slack dan memasukkannya ke exact DP, sedangkan tugas lain dipertimbangkan pada horizon berikutnya. Strategi rolling-horizon tersebut mempertahankan kontribusi program dinamis pada submasalah aktif tanpa mengharuskan seluruh tugas diproses sekaligus.

Hasil eksperimen mendukung analisis tersebut. Deduplikasi menekan banyak lintasan yang berakhir pada state identik, tetapi jumlah state tetap naik dari 1.201 pada lima tugas menjadi 177.121 pada sembilan tugas. Selain itu, feasibility pruning pada implementasi baru memeriksa kecukupan sisa jam kerja dan REST yang langsung diwajibkan; pengembangan berikutnya dapat menggunakan lower bound jumlah istirahat masa depan yang lebih kuat. Diskritisasi satu jam juga membatasi resolusi jadwal, sehingga tugas berdurasi pecahan jam perlu dibulatkan atau diproses dengan slot yang lebih kecil dengan konsekuensi peningkatan state.

XI. KESIMPULAN

Makalah ini merumuskan dan mengimplementasikan penjadwalan tugas mahasiswa sebagai masalah program dinamis pada sumber daya tunggal. Setiap tugas mempunyai release time, tenggat, estimasi durasi, prioritas yang berasal dari SKS dan bobot nilai, serta konsekuensi keterlambatan. Tenggat diperlakukan sebagai soft constraint melalui weighted tardiness, sedangkan penyelesaian sebelum tenggat diberi reward terbatas. Penalti perpindahan konteks mengurangi fragmentasi, dan state streak menegakkan aturan maksimal tiga jam kerja berturut-turut serta REST satu jam.

Implementasi sparse menggunakan rolling layer, mixed-radix dengan fallback vektor immutable, deduplikasi BestValue, feasibility pruning, dan parent pointer. Seluruh 16 pengujian fungsional berhasil, sedangkan studi kasus tiga tugas menyelesaikan semua tugas satu jam sebelum tenggat. Eksperimen hingga sembilan tugas mengonfirmasi bahwa teknik sparse mengurangi pengulangan state, tetapi kompleksitas tetap tumbuh kombinatorial. Dengan demikian, kontribusi utama makalah adalah model exact yang telah direalisasikan dan divalidasi untuk mengintegrasikan aspek akademik, temporal, dan perilaku kerja pada kumpulan tugas aktif berukuran kecil hingga menengah.

LINK YOUTUBE

https://youtu.be/IcL_qq6IOGo

LINK REPOSITORY

[LutherCeltors/task-scheduling-algorithm-mls: Scheduling algorithm for every day task as a student](https://github.com/LutherCeltors/task-scheduling-algorithm-mls)

LAMPIRAN PENGUJIAN

Dengan tetap menggunakan jenis pekerjaan yang sama dengan pengujian, yaitu A sebagai tugas “Algorithm Report”, B sebagai tugas “Database Design”, dan C sebagai tugas “ML Exercise”, berikut adalah hasil penjalanan program.

```
Hour-by-hour schedule:
00-01 WORK A (Algorithm Report)
01-02 WORK A (Algorithm Report)
02-03 WORK B (Database Design)
03-04 WORK B (Database Design)
04-05 WORK A (Algorithm Report)
05-06 WORK A (Algorithm Report)
06-07 WORK C (ML Exercise)
07-08 WORK C (ML Exercise)
08-09 WORK C (ML Exercise)

Grouped blocks:
00-02 WORK A (Algorithm Report)
02-04 WORK B (Database Design)
04-06 WORK A (Algorithm Report)
06-09 WORK C (ML Exercise)

Assignment results:
A completion=6 deadline=7 status=EARLY gap=1 priority=0.900 latePenalty=0.000 earlyReward=0.900
B completion=4 deadline=5 status=EARLY gap=1 priority=0.500 latePenalty=0.000 earlyReward=0.500
C completion=9 deadline=10 status=EARLY gap=1 priority=0.600 latePenalty=0.000 earlyReward=0.600

Context switches: 3 cost=6.000
Objective: 4.000
Work=9 mandatoryRest=0 idle=0
```

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, 19 Juni 2026



Marcel Luther Sitorus 13524063

DAFTAR PUSTAKA

- [1] R. Bellman, Dynamic Programming. Princeton, NJ, USA: Princeton University Press, 1957.
- [2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 4th ed. Cambridge, MA, USA: MIT Press, 2022.
- [3] M. L. Pinedo, Scheduling: Theory, Algorithms, and Systems, 5th ed. Cham, Switzerland: Springer, 2016.
- [4] E. L. Lawler and J. M. Moore, “A functional equation and its application to resource allocation and sequencing problems,” *Management Science*, vol. 16, no. 1, pp. 77–84, 1969.
- [5] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker, “Complexity of machine scheduling problems,” *Annals of Discrete Mathematics*, vol. 1, pp. 343–362, 1977.
- [6] K. R. Baker and G. D. Scudder, “Sequencing with earliness and tardiness penalties: A review,” *Operations Research*, vol. 38, no. 1, pp. 22–36, 1990.
- [7] A. Awasthi, J. Lässig, and O. Kramer, “Common due-date problem: Exact polynomial algorithms for a given job sequence,” *arXiv:1311.2879*, 2013.
- [8] A. Subramanian and K. Farias, “Efficient local search limitation strategy for single machine total weighted tardiness scheduling with sequence-dependent setup times,” *arXiv:1501.05882*, 2015.
- [9] S. Monsell, “Task switching,” *Trends in Cognitive Sciences*, vol. 7, no. 3, pp. 134–140, 2003.
- [10] J. S. Rubinstein, D. E. Meyer, and J. E. Evans, “Executive control of cognitive processes in task switching,” *Journal of Experimental Psychology: Human Perception and Performance*, vol. 27, no. 4, pp. 763–797, 2001.
- [11] Z. S. H. Abad, M. Noaen, D. Zowghi, B. H. Far, and K. Barker, “Two sides of the same coin: Software developers’ perceptions of task switching and task interruption,” *arXiv:1805.05504*, 2018.
- [12] Q. Ma and J. Stachurski, “Dynamic programming deconstructed: Transformations of the Bellman equation and computational efficiency,” *arXiv:1811.01940*, 2018.