JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 4, No. 1 Tahun 2026
ISSN 3025-0919 (media online)
Hal 105-115

Pendekatan Dynamic Programming pada Penentuan Urutan Pembelajaran Optimal Berdasarkan Beban Kognitif Mahasiswa

Joseph Eduard Uly Loni[1], M Dhafa Adjie Saputra[2], Yohannes[3]

[1,2,3] Program Studi Informatika, Fakultas Ilmu Komputer dan Rekayasa, Universitas Multi Data Palembang, Palembang, Indonesia
Email: [1]josepheduardulyloni_2327250080@mhs.mdp.ac.id,
[2]mdhafaadjiesaputra_2327250081@mhs.mdp.ac.id, [3]yohannesmasterous@mdp.ac.id

Abstrak−Penelitian ini bertujuan untuk mengoptimalkan prioritas belajar mahasiswa menjelang ujian akhir semester (UAS) dengan memformulasikan masalah pemilihan mata kuliah sebagai *0/1 knapsack problem*. Data diperoleh dari dua dataset sintetis yang menggabungkan karakteristik mahasiswa (tingkat stres, kelelahan, efisiensi belajar, performa) dan mata kuliah (tingkat kesulitan, urgensi jadwal ujian) menjadi nilai prioritas (*priority score*) serta alokasi waktu belajar sebagai bobot. Eksperimen dilakukan pada 10 kapasitas waktu belajar berbeda (4–28 jam) untuk membandingkan algoritma *Dynamic Programming* (DP) dan *Greedy*. Hasil menunjukkan bahwa DP selalu menghasilkan solusi optimal atau superior dengan total nilai prioritas lebih tinggi atau sama dibandingkan *Greedy* pada seluruh kapasitas, dengan *optimality gap* tertinggi sebesar 9,87% pada kapasitas 16 jam. Meskipun waktu eksekusi DP lebih lambat (*mikrodetik*), perbedaannya tidak signifikan untuk ukuran data kecil. Disimpulkan bahwa pendekatan DP efektif untuk merekomendasikan kombinasi mata kuliah prioritas secara optimal. Penelitian selanjutnya disarankan menggunakan data nyata dan algoritma optimasi lain seperti *branch and bound* atau algoritma genetika.

Kata Kunci: *Knapsack Problem*; *Dynamic Programming*; beban kognitif; urutan pembelajaran optimal; kapasitas kognitif.

Abstract−*This study aims to optimize student learning priorities prior to the final semester examination (UAS) by formulating the course selection problem as a 0/1 knapsack problem. Data were obtained from two synthetic datasets that combine student characteristics (stress level, fatigue, learning efficiency, performance) and course characteristics (difficulty level, exam schedule urgency) into a priority score, with study time allocation as the weight. Experiments were conducted on 10 different study time capacities (4–28 hours) to compare the Dynamic Programming (DP) and Greedy algorithms. The results show that DP consistently produces optimal or superior solutions, achieving a total priority value higher than or equal to Greedy across all capacities, with the highest optimality gap of 9.87% at a capacity of 16 hours. Although DP has a slower execution time (microseconds), the difference is negligible for small dataset sizes. It is concluded that the DP approach is effective for optimally recommending priority course combinations. Future research is advised to use real-world data and other optimization algorithms such as branch and bound or genetic algorithms.*

Keywords: *Knapsack Problem; Dynamic Programming; Cognitive Load; Optimal Learning Sequence; Cognitive Capacity.*

1. PENDAHULUAN

Periode ujian akhir semester (UAS) merupakan fase kritis bagi mahasiswa, di mana kemampuan mereka dalam mengatur dan memprioritaskan waktu belajar sangat menentukan keberhasilan akademik. Menjelang UAS, mahasiswa umumnya dihadapkan pada beberapa mata kuliah yang harus dipelajari dengan waktu yang terbatas. Tidak jarang, mahasiswa merasa kebingungan dalam menentukan urutan prioritas mata kuliah yang harus dipelajari terlebih dahulu (Robiatul Adawiyah, 2017). Permasalahan ini semakin kompleks karena adanya berbagai faktor yang saling terkait, seperti tingkat kesulitan mata kuliah, urgensi jadwal ujian, serta kondisi internal mahasiswa seperti stres, kelelahan, efisiensi belajar, dan performa akademik sebelumnya (Praseno, 2024). Sweller (1988) dalam teori beban kognitifnya menjelaskan bahwa desain pembelajaran yang efektif harus meminimalkan beban kognitif ekstra (*extraneous load*) dan mengoptimalkan beban kognitif yang konstruktif, sehingga kapasitas memori kerja dapat digunakan secara maksimal untuk pemrosesan informasi yang mendalam. Kim et al. (2025) memberikan kajian kritis terkini tentang *cognitive load theory* (CLT) yang dikemukakan oleh Sweller (1988), dengan diskusi mendalam mengenai implikasi teori ini dalam konteks pembelajaran di perguruan tinggi. Karya ini telah

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 4, No. 1 Tahun 2026
ISSN 3025-0919 (media online)
Hal 105-115

memperoleh lebih dari 12.000 tampilan dan terindeks dengan baik, sehingga sangat valid sebagai referensi tentang beban kognitif mahasiswa. Kombinasi dari berbagai faktor ini seringkali membuat mahasiswa mengambil keputusan yang kurang optimal, sehingga berpotensi menurunkan pencapaian akademik mereka.

Dalam upaya memahami dan mengatasi permasalahan pendidikan yang kompleks, berbagai penelitian telah menerapkan pendekatan *data mining* dan algoritma optimasi. Misalnya, Samaray (2022) menggunakan algoritma *Rough Set* untuk memprediksi hasil belajar mahasiswa berdasarkan kehadiran, nilai tugas, nilai UTS, dan nilai UAS, dan menemukan bahwa nilai UAS merupakan atribut yang paling berpengaruh. Sementara itu, Raschintasofi et al. (2022) menerapkan algoritma *K-Nearest Neighbor* (KNN) untuk menganalisis tingkat pemahaman mahasiswa dalam pembelajaran daring. Widyawati et al. (2022) menggunakan metode *Naive Bayes* untuk memprediksi tingkat stres mahasiswa selama perkuliahan *hybrid*. Penelitian-penelitian tersebut menunjukkan bahwa pendekatan komputasional sangat potensial untuk mengatasi masalah edukasi, namun belum ada yang secara khusus menyelesaikan masalah prioritas belajar menjelang UAS dengan mempertimbangkan keterbatasan waktu belajar sebagai kapasitas utama.

Secara fundamental, pemilihan mata kuliah prioritas dapat dipandang sebagai masalah optimasi kombinatorial, yaitu menentukan kombinasi mata kuliah yang memberikan nilai prioritas tertinggi dengan keterbatasan waktu belajar. Dalam ranah ilmu komputer dan riset operasi, permasalahan ini dikenal sebagai *Knapsack Problem* (Ilham & Saputra, 2023). *Knapsack problem* bertujuan memilih sekumpulan item (mata kuliah) dengan bobot (alokasi waktu belajar) dan nilai (skor prioritas) ke dalam wadah berkapasitas tertentu (total waktu belajar efektif) sehingga total nilai maksimum (Guntara et al., 2023). Untuk menyelesaikan *0/1 knapsack problem* (setiap mata kuliah dipilih seluruhnya atau tidak sama sekali), terdapat beberapa algoritma. Algoritma *Greedy* sederhana dan cepat namun tidak selalu optimal (Prasha et al., 2024). Ilham & Saputra (2023) menerapkan algoritma *Greedy* untuk seleksi penerimaan mahasiswa baru dengan hasil cepat, tetapi belum menjamin optimalitas global. Penelitian Rizal (2024) secara langsung mengimplementasikan algoritma greedy untuk optimalisasi penjadwalan mata pelajaran pada sistem informasi akademik berbasis web, dengan tingkat keberhasilan pengujian Black Box mencapai 100% dan *User Acceptance Testing* (UAT) sebesar 89,2%. Sebaliknya, *Dynamic Programming* (DP) memberikan jaminan solusi optimal dengan memecah masalah menjadi sub masalah (Riyanti et al., 2023). Penelitian Guntara et al., (2023) juga menunjukkan bahwa algoritma genetika dapat digunakan untuk optimasi penjadwalan sidang tugas akhir, namun DP lebih terstruktur untuk masalah *knapsack*. Selain itu, Wee et al. (2024) mengembangkan dashboard analitik pembelajaran untuk memahami proses berpikir siswa, yang sejalan dengan kebutuhan dashboard untuk dosen dalam memonitor prioritas belajar mahasiswa.

Fokus penelitian ini terletak pada perumusan nilai (*value*) mata kuliah yang tidak hanya berdasarkan data statis mata kuliah, tetapi juga data dinamis perilaku mahasiswa per minggu (stres, kelelahan, efisiensi belajar, performa). Pendekatan ini memungkinkan rekomendasi belajar yang personal dan adaptif. Penelitian ini juga membandingkan kinerja DP dengan algoritma *Greedy* menggunakan metrik seperti efisiensi belajar (nilai per jam) dan *optimality gap*. Dengan demikian, tujuan penelitian ini adalah merumuskan model optimasi pemilihan mata kuliah prioritas menggunakan DP untuk *0/1 knapsack* problem serta membandingkan kinerjanya dengan algoritma *Greedy* dalam konteks prioritas belajar mahasiswa menjelang UAS.

2. METODE

2.1 Desain Penelitian

Penelitian ini menggunakan pendekatan eksperimen komputasional (*computational experiment*) dengan desain *comparative study*. Tujuan dari pendekatan ini adalah untuk membandingkan kinerja dua algoritma optimasi (*Dynamic Programming* dan *Greedy*) dalam menyelesaikan masalah pemilihan prioritas belajar mahasiswa yang diformulasikan sebagai *0/1 knapsack problem*. Eksperimen dilakukan pada data sintetis yang merepresentasikan kondisi belajar mahasiswa dan karakteristik mata kuliah. Pendekatan ini dipilih karena memungkinkan pengendalian variabel, pengulangan eksperimen, serta pengukuran kuantitatif terhadap metrik kinerja seperti total nilai prioritas, *optimality gap*, efisiensi, dan waktu eksekusi.

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 4, No. 1 Tahun 2026
ISSN 3025-0919 (media online)
Hal 105-115

2.2 Dataset

Penelitian ini menggunakan dua dataset sintetis yang bersumber dari platform publik Kaggle. Dataset pertama, *Student Learning Trajectory*, merekam perilaku belajar mahasiswa secara longitudinal selama 16 minggu, mencakup berbagai indikator seperti tingkat stres, kelelahan, efisiensi belajar, dan performa akademik. Dataset kedua, *Course Dataset Synthetic*, berisi karakteristik 10 mata kuliah umum di kurikulum program studi Teknik Informatika, meliputi tingkat kesulitan dasar, kebutuhan matematika dan pemrograman, serta kedalaman konsep. Kedua dataset tersebut digabungkan melalui proses *expanded dataset* untuk menghasilkan data per mahasiswa per minggu per mata kuliah.

2.2.1 Dataset Student Learning Trajectory

Dataset ini merekam perilaku belajar mahasiswa secara longitudinal selama 16 minggu. Terdiri dari 9.776 baris data dengan 13 atribut. Seluruh nilai telah dinormalisasi ke rentang tertentu sesuai dengan proses pembangkitan data sintetis. Atribut pada dataset dapat dilihat pada Tabel 1.

Tabel 1. Atribut pada Student Learning Trajectory Dataset

No	Nama Atribut	Deskripsi
1	*student_id*	Identifikasi unik mahasiswa (0 – 609)
2	*week*	Minggu ke- perkuliahan (1 – 16)
3	*study_hours*	Rata-rata jam belajar per hari dalam minggu tersebut
4	*sleep_hours*	Rata-rata jam tidur per hari
5	*stress_level*	Tingkat stres mahasiswa (skala 0–100, semakin tinggi semakin stres)
6	*attendance_rate*	Tingkat kehadiran dalam perkuliahan (0–1)
7	*screen_time*	Rata-rata waktu penggunaan layar (jam/hari)
8	*caffeine_intake*	Asupan kafein (skala 0–4)
9	*learning_efficiency*	Efisiensi belajar (0–100)
10	*fatigue_index*	Indeks kelelahan mahasiswa (0–100)
11	*quiz_score*	Nilai kuis (0–100)
12	*assignment_score*	Nilai tugas (0–150)
13	*performance_index*	Indeks performa akademik gabungan (0–120)

2.2.2 Dataset Course Dataset Synthetic

Dataset ini berisi karakteristik 10 mata kuliah yang umum ditemui di kurikulum teknik informatika. Data disajikan dalam 10 baris dengan 10 atribut yang dapat dilihat pada Tabel 2.

Tabel 2. Atribut pada Course Dataset Synthetic

No	Nama Atribut	Deskripsi
1	*subject_name*	Nama mata kuliah
2	*course_type*	Jenis mata kuliah: *Calculation* (perhitungan) atau *Theory* (teori)
3	*mathematical_requirement*	Tingkat kebutuhan matematika (skala 0–10)
4	*programming_requirement*	Tingkat kebutuhan pemrograman (0–10)
5	*conceptual_depth*	Kedalaman konsep (0–10)
6	*practical_workload*	Beban kerja praktik (0–10)
7	*base_course_difficulty*	Tingkat kesulitan dasar mata kuliah (0–10)

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 4, No. 1 Tahun 2026
ISSN 3025-0919 (media online)
Hal 105-115

8	*focus_requirement*	Tingkat kebutuhan fokus belajar (0–10)
9	*exam_day*	Hari pelaksanaan ujian (1–5, ditambahkan dari jadwal tetap)
10	*days_until_exam*	Jumlah hari tersisa hingga ujian (6–11, hasil dari *exam_day* + 6)

2.2.3 Dataset Hasil Penggabungan

Kedua dataset di atas digabungkan melalui proses *expanded dataset* untuk setiap mahasiswa, setiap minggu, dan setiap mata kuliah yang diambil. Proses ini menghasilkan data per mahasiswa per minggu per mata kuliah dengan total baris sesuai jumlah mahasiswa × jumlah minggu × jumlah mata kuliah yang diambil (7–10 mata kuliah per mahasiswa). Atribut tambahan yang dihasilkan dapat dilihat pada Tabel 3.

Tabel 3. Atribut pada Dataset Hasil Penggabungan

No	Nama Atribut	Deskripsi
1	*allocated_study_hours*	Alokasi waktu belajar efektif untuk mata kuliah tertentu pada minggu tersebut (dalam jam)
2	*final_difficulty*	Tingkat kesulitan akhir yang dipengaruhi oleh kondisi mahasiswa
3	*urgency_bonus*	Bonus urgensi berdasarkan kedekatan hari ujian
4	*priority_score*	Skor prioritas akhir (0–100) sebagai nilai (*value*) dalam *knapsack*

Dari dataset final ini, penelitian memfokuskan pada minggu ke-15 (minggu terakhir sebelum UAS) untuk satu mahasiswa dengan ID = 1 yang memiliki 10 mata kuliah terdaftar. Data inilah yang menjadi masukan utama eksperimen *knapsack*.

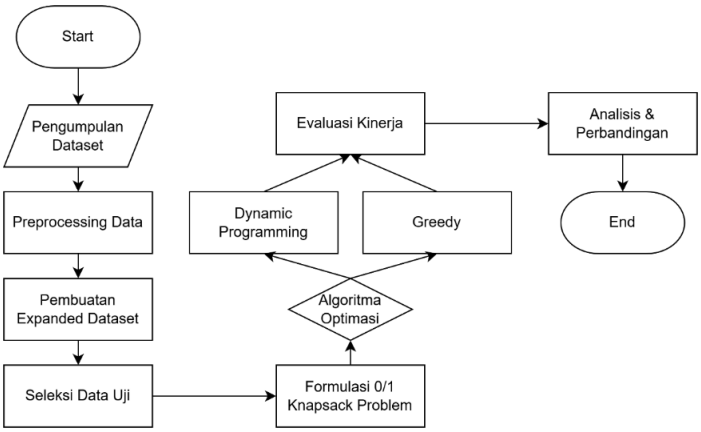
2.3 Tahap Penelitian

Tahapan penelitian disusun secara sistematis sebagai berikut:

a. Pengumpulan data: Mengumpulkan dataset *Student Learning Trajectory* dan *Course Dataset Synthetic* dari platform publik seperti *Kaggle*.
b. Prapemrosesan data: Normalisasi fitur mahasiswa (*stress_level, fatigue_index, learning_efficiency, sleep_hours, performance_index) dan fitur mata kuliah (base_course_difficulty, focus_requirement, mathematical_requirement, programming_requirement, conceptual_depth, practical_workload*) ke rentang 0–100 menggunakan *MinMaxScaler*.
c. Pembuatan *expanded dataset*: Menggabungkan data mahasiswa dan mata kuliah dengan aturan: setiap mahasiswa mengambil 7–10 mata kuliah secara acak, kemudian untuk setiap minggu dihitung alokasi jam belajar (*allocated_study_hours*) berdasarkan rasio *raw effort* dan kapasitas belajar mingguan. Selanjutnya dihitung *priority_score* sebagai nilai prioritas untuk setiap kombinasi mahasiswa–minggu–mata kuliah.
d. Seleksi data uji: Memfilter data pada minggu ke-15 (UAS) dan memilih satu mahasiswa (*student_id*=1) yang memiliki 10 mata kuliah.
e. Formulasi *knapsack*: Menentukan bobot (*weights*) sebagai *allocated_study_hours* (dibulatkan ke atas) dan nilai (*values*) sebagai *priority_score* (dibulatkan ke bilangan bulat). Kapasitas ditentukan sebagai variabel eksperimen dengan 10 nilai: {4, 8, 10, 12, 16, 18, 20, 22, 24, 28} jam.
f. Implementasi algoritma: Menerapkan *Dynamic Programming* (*DP*) dan algoritma *Greedy* untuk setiap nilai kapasitas.
g. Evaluasi: Menghitung metrik kinerja untuk setiap kapasitas, meliputi total nilai prioritas, *optimality gap*, pemanfaatan kapasitas, efisiensi, cakupan mata kuliah, dan waktu eksekusi.

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 4, No. 1 Tahun 2026
ISSN 3025-0919 (media online)
Hal 105-115

h. Analisis perbandingan: Membandingkan hasil *DP* dan *Greedy* serta menyajikan dalam bentuk tabel dan grafik.



Gambar 1. Alur Tahapan Penelitian

2.4 Metode yang digunakan

Penelitian ini memformulasikan masalah pemilihan prioritas belajar mahasiswa sebagai *0/1 Knapsack Problem.* Setiap mata kuliah direpresentasikan sebagai *item* dengan bobot w_i(alokasi waktu belajar) dan nilai v_i(skor prioritas). Kapasitas total waktu belajar dinyatakan sebagai *C.* Tujuan utama adalah memilih *subset* mata kuliah yang memaksimalkan total nilai tanpa melampaui kapasitas. Masalah optimasi *0/1 knapsack problem* dapat diformulasikan dengan Persamaan (1), dengan kendala yang dinyatakan pada Persamaan (2):

$$\max \sum_{i=1}^{n} v_i x_i \quad (1)$$

$$\sum_{i=1}^{n} w_i x_i \leq C, x_i \in \{0,1\} \quad (2)$$

Di mana *n* adalah jumlah mata kuliah, dan $x_i = 1$ jika mata kuliah ke-*i* dipilih, serta 0 jika tidak dipilih.

2.4.1 Algoritma Dynamic Programming (DP)

Algoritma DP dipilih karena mampu menghasilkan solusi optimal global untuk *0/1 knapsack* dengan ukuran data yang relatif kecil ($n = 10, C \leq 28$). DP membangun tabel *dp[i][c]* yang menyimpan nilai maksimum yang dapat dicapai menggunakan i mata kuliah pertama dengan kapasitas *C*. Dengan rumus pada persamaan (3), relasi rekurensi yang digunakan adalah:

$$dp[i][c] = \begin{cases} dp[i-1][c], & jika\ w_i > c \\ \max(dp[i-1][c], dp[i-1][c-w_i] + v_i), & jika\ w_i \leq c \end{cases} \quad (3)$$

Nilai optimal akhir diperoleh dari *dp[n][C]*. Selanjutnya, proses penelusuran balik dilakukan untuk menentukan mata kuliah yang terpilih berdasarkan Persamaan (4).

$$Jika\ dp[i][c] \neq dp[i-1][c]\ maka\ mata\ kuliah\ i\ diambil, lalu\ c = c - w_i \quad (4)$$

Proses ini diulang hingga seluruh item diperiksa untuk menentukan status pengambilan dari setiap mata kuliah secara formal, keputusan pemilihan ini disimbolkan dengan variabel biner x_i yang didefinisikan pada Persamaan (5):

$$x_i = \begin{cases} 1, & jika\ dp[i][c] \neq dp[i-1][c] \\ 0, & jika\ dp[i][c] = dp[i-1][c] \end{cases} \quad (5)$$

Dengan pembaruan kapasitas $c \leftarrow c - w_i$, jika $x_i = 1$. Proses ini dilakukan dengan menurunkan indeks i hingga seluruh elemen telah dievaluasi. Menurut Harefa et al. (2025), *dynamic*

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 4, No. 1 Tahun 2026
ISSN 3025-0919 (media online)
Hal 105-115

programming terbukti efektif dalam optimasi sistem *multistage* dengan fluktuasi permintaan, mampu menekan biaya produksi dan persediaan, serta mengungguli metode konvensional seperti EOQ karena sifatnya yang adaptif.

2.4.2 Algoritma *Greedy*

Menurut Roihan et al. (2022), algoritma *greedy* merupakan salah satu metode yang paling populer dalam menyelesaikan persoalan optimasi, yaitu persoalan yang menuntut pencarian solusi optimum (terbaik). Algoritma ini sederhana dan fleksibel, bekerja dengan selalu mengambil penyelesaian sementara atau lokal yang terbaik pada setiap langkah, tanpa memikirkan pengaruhnya terhadap penyelesaian secara keseluruhan. Dima et al. (2025) dalam tinjauan literaturnya juga menegaskan bahwa algoritma greedy sangat unggul dalam hal kecepatan komputasi dan kemudahan implementasi, meskipun tidak selalu menjamin solusi global optimal.

Algoritma *Greedy* diimplementasikan sebagai pembanding dengan strategi memilih item berdasarkan nilai tertinggi (*value-based greedy*). Langkah-langkahnya adalah sebagai berikut: pertama, semua mata kuliah diurutkan menurun berdasarkan nilai v_i kedua, dilakukan iterasi dari nilai tertinggi ke terendah; ketiga, suatu mata kuliah dimasukkan ke dalam solusi jika bobotnya tidak melebihi kapasitas yang tersisa. Tidak ada proses *backtracking* atau perbaikan. Dengan rumus pada persamaan (6), total nilai *Greedy* dinyatakan sebagai:

$$Total_{greedy} = \sum_{j \in S_{greedy}} v_j \; dengan \; \sum_{j \in S_{greedy}} w_j \leq C \qquad (6)$$

Di mana S_{greedy} adalah himpunan mata kuliah yang terpilih melalui strategi *Greedy*. Algoritma ini dipilih karena kompleksitasnya yang rendah ($O(n \log n)$) dan mudah diimplementasikan, meskipun tidak menjamin optimalitas.

2.4.3 Paramater dan Nilai yang Digunakan

Berdasarkan hasil *expanded dataset*, bobot w_i diperoleh dari nilai *allocated_study_hours* yang dibulatkan ke atas (*ceil*) menjadi bilangan bulat. Nilai v_i diperoleh dari *priority_score* yang dibulatkan ke bilangan bulat terdekat (*round*). Kapasitas C ditentukan melalui 10 skenario eksperimen, yaitu $C \in \{4,8,10,12,16,18,20,22,24,28\}$. Dengan rumus pada persamaan (7), efisiensi belajar didefinisikan sebagai total nilai prioritas per satuan waktu belajar:

$$Efisiensi = \frac{\sum v_i}{\sum w_i} \qquad (7)$$

Metrik ini digunakan dalam evaluasi untuk mengukur kualitas solusi selain dari total nilai absolut.

2.5 Teknik Evaluasi

Evaluasi kinerja dilakukan dengan membandingkan hasil DP dan *Greedy* menggunakan metrik berikut:

Tabel 4. Teknik Evaluasi

No.	Nama Metriks	Deskripsi
1	Total Nilai Prioritas	Jumlah v_i dari mata kuliah terpilih
2	*Optimality Gap*	Persentase selisih antara nilai *Greedy* dan DP terhadap nilai DP
3	Pemanfaatan Kapasitas	Persentase kapasitas yang terpakai
4	Efisiensi	Rasio total nilai prioritas terhadap total bobot terpilih
5	Cakupan Mata Kuliah	Jumlah mata kuliah yang berhasil dipilih
6	Waktu Eksekusi	Waktu yang diukur dalam mikrodetik

Semua metrik dihitung untuk setiap nilai kapasitas dan disajikan dalam bentuk tabel serta grafik perbandingan. Tidak dilakukan validasi silang karena data bersifat sintetis dan penelitian bersifat eksploratif komparatif.

3. ANALISIS DAN PEMBAHASAN

Pada bagian ini disajikan hasil dari seluruh rangkaian eksperimen yang telah dilakukan. Data yang digunakan berasal dari proses *expanded dataset* pada minggu ke-15 (UAS) untuk satu

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 4, No. 1 Tahun 2026
ISSN 3025-0919 (media online)
Hal 105-115

mahasiswa dengan ID = 1, yang memiliki 10 mata kuliah. Setiap mata kuliah memiliki bobot (*allocated_study_hours*) dan nilai (*priority_score*) seperti ditunjukkan pada Tabel 5.

Tabel 5. Data Mata Kuliah Mahasiswa

No	Mata Kuliah	Bobot(jam)	Nilai(*priority_score)*	Hari Ujian	Sisa Hari
1	Algoritma	9	87	1	7
2	Sistem Operasi	4	37	2	8
3	Struktur Data	8	63	3	9
4	Basis Data	2	11	2	8
5	Machine Learning	10	85	4	10
6	Statistika	3	28	4	10
7	Kalkulus	6	50	5	11
8	Rekayasa Perangkat Lunak	2	11	5	11
9	Kecerdasan Buatan	9	87	1	7
10	Jaringan Komputer	2	14	3	9

Eksperimen dilakukan terhadap 10 nilai kapasitas berbeda: 4, 8, 10, 12, 16, 18, 20, 22, 24, dan 28 jam. Pada setiap kapasitas, algoritma DP dan *Greedy* dijalankan, lalu dievaluasi berdasarkan total nilai prioritas, *optimality gap*, pemanfaatan kapasitas, efisiensi, cakupan mata kuliah, serta waktu eksekusi.

3.1 Hasil Eksperimen

Hasil menunjukkan bahwa pada seluruh kapasitas yang diuji, algoritma *Dynamic Programming* (DP) selalu menghasilkan total nilai prioritas yang lebih tinggi atau sama dibandingkan algoritma *Greedy*. Selisih terbesar terjadi pada kapasitas 16 jam, yaitu sebesar 9,87%, diikuti kapasitas 8 jam (3,08%), 24 jam (0,44%), dan 28 jam (0,77%). Pada kapasitas lainnya, kedua algoritma mencapai nilai yang sama. Tabel 6 merangkum perbandingan nilai prioritas dan *optimality gap*.

Tabel 6. Perbandingan Nilai Prioritas dan *Optimality Gap*

Kapasitas	*DP Priority*	*Greedy Priority*	*Optimality Gap*
4	37	37	0.00
8	65	63	3.08
10	87	87	0.00
12	115	115	0.00
16	152	137	9.87
18	174	174	0.00
20	188	188	0.00
22	211	211	0.00
24	225	224	0.44
28	261	259	0.77

Tabel 7 merangkum perbandingan pemanfaatan kapasitas

Tabel 7. Perbandingan Pemanfaatan Kapasitas

Kapasitas	*DP Hours Used*	*Greedy Hours Used*	*DP Util. (%)*	*Greedy Util. (%)*
4	4	4	100	100
8	7	8	87.5	100
10	9	9	90	90
12	12	12	100	100
16	16	15	100	93.8
18	18	18	100	100

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 4, No. 1 Tahun 2026
ISSN 3025-0919 (media online)
Hal 105-115

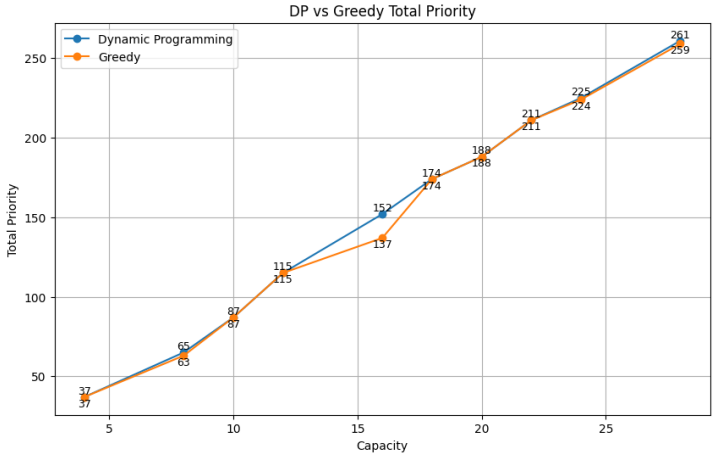
20	20	20	100	100
22	22	22	100	100
24	24	24	100	100
28	28	28	100	100

Hasil dari eksekusi memperlihatkan bahwa DP dan *Greedy* umumnya memanfaatkan hampir seluruh kapasitas waktu belajar. Namun, pada kapasitas 8 jam, DP hanya menggunakan 87,5% kapasitas karena memilih dua mata kuliah dengan bobot kecil yang memberikan nilai total lebih tinggi, sedangkan *Greedy* menggunakan 100% kapasitas tetapi dengan nilai total lebih rendah. Pada kapasitas 16 jam, DP memanfaatkan kapasitas penuh (100%), sementara *Greedy* hanya mencapai 93,8%.

3.2 Pembahasan

3.2.1 Perbandingan Total Nilai Prioritas

Berdasarkan Tabel 4, algoritma DP selalu menghasilkan total nilai prioritas yang lebih tinggi atau sama dengan algoritma *Greedy* pada semua kapasitas. Pada kapasitas 8, 16, 24, dan 28, *Greedy* gagal mencapai solusi optimal, dengan *optimality gap* tertinggi sebesar 9,87% pada kapasitas 16. Hal ini menunjukkan bahwa meskipun *Greedy* sederhana dan cepat, ia tidak dapat diandalkan untuk menghasilkan solusi optimal ketika kombinasi item dengan bobot kecil namun nilai sedang lebih menguntungkan.



Gambar 2. Perbandingan total nilai prioritas

Pada kapasitas 16, DP memilih tiga mata kuliah (Algoritma 9 jam + Sistem Operasi 4 jam + Statistika 3 jam) dengan total nilai 152, sedangkan *Greedy* memilih Algoritma (9 jam, nilai 87) dan Kalkulus (6 jam, nilai 50) dengan total nilai 137. *Greedy* mengabaikan Sistem Operasi (nilai 37) dan Statistika (nilai 28) karena setelah mengambil Algoritma (9 jam) dan Kalkulus (6 jam), kapasitas tersisa hanya 1 jam, tidak cukup untuk mata kuliah lain. Sebaliknya, DP memilih kombinasi yang lebih padat nilai.

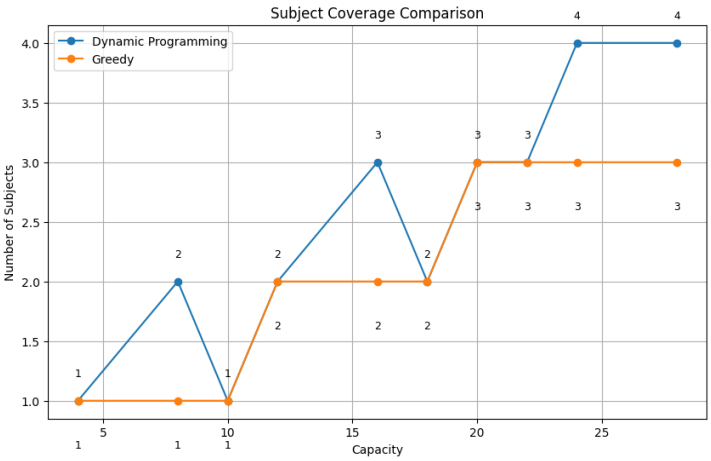
3.2.2 Pemanfaatan Kapasitas dan Efisiensi

Pada hampir semua skenario (kecuali kapasitas 8 untuk DP dan kapasitas 8,16 untuk *Greedy*), kedua algoritma mencapai pemanfaatan kapasitas 100% atau di atas 90%. Efisiensi (total nilai per jam) DP cenderung stabil di sekitar 9,3–9,7, sedangkan *Greedy* lebih fluktuatif. Pada kapasitas 8, efisiensi *Greedy* turun hingga 7,88 karena hanya memilih satu mata kuliah (Struktur Data, bobot 8, nilai 63), sementara DP memilih dua mata kuliah (Sistem Operasi bobot 4 dan Statistika bobot 3) dengan total nilai 65.
Temuan ini menegaskan bahwa DP tidak hanya optimal dalam total nilai, tetapi juga lebih efisien dalam memanfaatkan setiap jam belajar.

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 4, No. 1 Tahun 2026
ISSN 3025-0919 (media online)
Hal 105-115

3.2.3 Cakupan Mata Kuliah

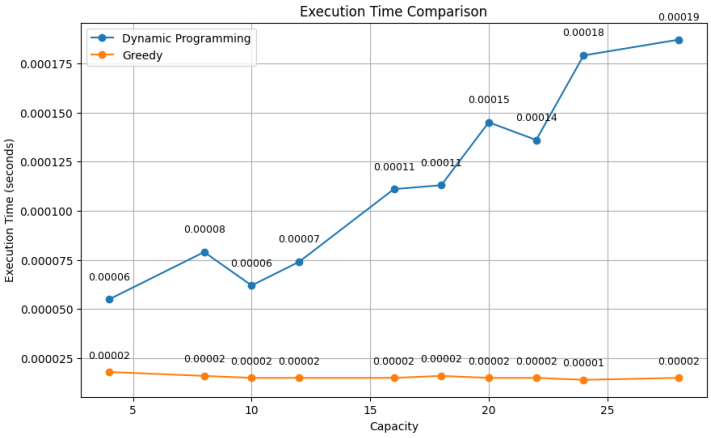
DP cenderung memilih lebih banyak mata kuliah (*coverage* lebih tinggi) dibandingkan *Greedy*, terutama pada kapasitas sedang (8, 16, 24, 28). Misalnya pada kapasitas 28, DP memilih 4 mata kuliah, sementara *Greedy* hanya 3 mata kuliah. Hal ini penting dalam konteks akademik karena mahasiswa perlu mempersiapkan lebih dari satu mata kuliah secara merata.



Gambar 3. Cakupan Mata Kuliah

3.2.4 Waktu Eksekusi

Waktu eksekusi DP selalu lebih lambat daripada *Greedy*, dengan perbedaan faktor sekitar 2–10 kali. Namun, karena skala data sangat kecil (n=10, C≤28), waktu absolut DP masih dalam orde mikrodetik (maksimum 170 µs), sehingga perbedaan ini tidak signifikan untuk penggunaan praktis. Untuk sistem *real-time* dengan data ribuan mahasiswa, peningkatan waktu DP perlu dipertimbangkan, namun dalam konteks satu mahasiswa dan sepuluh mata kuliah, DP tetap layak.



Gambar 4. Perbandingan Waktu Eksekusi

Dari hasil eksperimen, dapat disimpulkan bahwa:

a. *Dynamic Programming* memberikan solusi optimal atau mendekati optimal untuk semua kapasitas, sehingga direkomendasikan untuk sistem rekomendasi prioritas belajar yang membutuhkan akurasi tinggi.
b. Algoritma *Greedy* dapat digunakan jika kecepatan menjadi prioritas utama dan toleransi terhadap kesalahan (gap hingga 10%) dapat diterima, misalnya pada skenario awal penjadwalan dengan kapasitas besar.

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 4, No. 1 Tahun 2026
ISSN 3025-0919 (media online)
Hal 105-115

c. Kapasitas optimal bagi mahasiswa pada studi kasus ini adalah 18–22 jam per minggu, karena pada rentang tersebut DP dan *Greedy* menghasilkan solusi optimal dengan pemanfaatan kapasitas penuh dan efisiensi tertinggi (9,67 dan 9,59).
d. Penelitian ini juga membuktikan bahwa formulasi *knapsack* dengan bobot berupa alokasi waktu belajar dan nilai berupa prioritas yang mempertimbangkan faktor kelelahan, urgensi, serta performa mahasiswa merupakan pendekatan yang valid untuk membantu mahasiswa dalam menentukan prioritas belajar menjelang UAS.

4. KESIMPULAN

Berdasarkan hasil eksperimen dan pembahasan, dapat disimpulkan bahwa formulasi pemilihan prioritas belajar mahasiswa sebagai *0/1 knapsack problem* dapat diselesaikan secara optimal menggunakan algoritma *Dynamic Programming* (DP), yang terbukti selalu menghasilkan total nilai prioritas lebih tinggi atau sama dengan algoritma *Greedy* pada seluruh skenario kapasitas, dengan *optimality gap* tertinggi mencapai 9,87% pada kapasitas 16 jam. Meskipun DP memiliki waktu eksekusi lebih lambat dibandingkan *Greedy*, perbedaan tersebut tidak signifikan untuk ukuran data yang kecil (n=10, C≤28). Penelitian ini juga menegaskan bahwa pemformulasian pemilihan mata kuliah sebagai *0/1 knapsack problem* secara tidak langsung mengakomodasi prinsip manajemen beban kognitif, karena pembatasan kapasitas waktu belajar (C) berfungsi sebagai batas atas akumulasi beban kognitif mahasiswa. Dengan menerapkan DP, mahasiswa dapat memilih kombinasi mata kuliah yang tidak hanya optimal secara nilai prioritas, tetapi juga menghindari kelebihan beban kognitif (*cognitive overload*) yang sering terjadi menjelang UAS. Sebagai saran untuk penelitian selanjutnya, perlu dilakukan eksperimen pada dataset yang lebih besar (lebih banyak mahasiswa dan mata kuliah), integrasi dengan data nyata dari sistem informasi akademik, serta penerapan metode optimasi lain seperti *branch and bound* atau algoritma genetika untuk perbandingan yang lebih luas.

REFERENCES

Dima, J., Hamzah, M. S., Tallo, C. G., & Ariswati, D. Y. (2025). Tinjauan Literatur tentang Pemanfaatan Algoritma Greedy untuk Pencarian Jalur Terpendek. 7.

Gelar Guntara, Rangga, Rizki Nugraha, M., Prasetyo, Y., & Aprilia, R. (2023). Implementasi Algoritma Genetika Untuk Aplikasi Penjadwalan Sidang Tugas Akhir Berbasis Web. Jurnal Minfo Polgan, 12(2), 2224–2232. https://doi.org/10.33395/jmp.v12i2.13206

Harefa, F. M., Lubis, N. U., Pasaribu, A. U. K., Nasution, A., & Ginting, S. S. B. (2025). Penerapan Program Dinamik Untuk Optimasi Perencanaan Produksi Dan Persediaan. Jejak Digital: Jurnal Ilmiah Multidisiplin, 2(1), 1641–1650. https://doi.org/10.63822/7yh2g598

Ilham, M. F. N., & Saputra, A. (2023). Seleksi Penerimaan Mahasiswa Baru Dengan Metode Pemecahan Masalah Algoritma Greedy Menggunakan Python. Jurnal Rekayasa Teknologi Informasi (JURTI), 7(1), 32–38. https://doi.org/10.30872/jurti.v7i1.9566

Kim, M., Duncan, C., Yip, S., & Sankey, D. (2025). Beyond the theoretical and pedagogical constraints of cognitive load theory, and towards a new cognitive philosophy in education. 56(7), 662–673.

Praseno, I. (2024). Learning Analytics untuk Meningkatkan Kualitas Pendidikan di Indonesia: Sebuah Kajian Pustaka (Learning Analytics to Improve the Quality of Education in Indonesia: A Literature Review).

Prasha, A., Ourizqi Rachmadi, C., Garin Raditya, N., Mutiara, S., & Sari, A. (2024). Implementasi Algoritma Greedy dan Dynamic Programming untuk Masalah Penjadwalan Interval dengan Model Knapsack. Format Jurnal Ilmiah Teknik Informatika, Vol. 13, No. 2, 15. https://doi.org/10.22441/format.2024.v13.i2.005

Raschintasofi, M., Khumairo, N., Rasywir, E., & Feranika, A. (2022). Analisis Tingkat Pemahaman Mahasiswa Universitas Dinamika Bangsa Dalam Pembelajaran Daring Menggunakan Algoritma K-Nearst Neighbor. Jurnal Manajemen Teknologi Dan Sistem Informasi (JMS), 2(1), 69–77. https://doi.org/10.33998/jms.2022.2.1.29

Rizal, M. F. (2024). Penerapan Algoritma Greedy untuk Optimalisasi Penjadwalan Mata Pelajaran pada Sistem Informasi Akademik SMK Negeri 5 Kendal Berbasis Web.

Robiatul Adawiyah. (2017). Analisis Tingkat Stres Mahasiswa dalam Menghadapi Penyusunan Skripsi [S1, Universitas Muhammadiyah Yogyakarta]. http://repository.umy.ac.id/handle/123456789/12924

Roihan, A., Nasution, K., & Siambaton, Mhd. Z. (2022). Implementasi Algoritma Greedy Kombinasi dengan Perulangan pada Aplikasi Penjadwalan Praktikum. sudo Jurnal Teknik Informatika, 1(2), 42–50. https://doi.org/10.56211/sudo.v1i2.8

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 4, No. 1 Tahun 2026
ISSN 3025-0919 (media online)
Hal 105-115

Samaray, S. (2022). Implementasi Algoritma Rough Set dengan Software Rosetta untuk Prediksi Hasil Belajar. Jurnal Eksplora Informatika, 11(1), 57–66. https://doi.org/10.30864/eksplora.v11i1.498

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. Cognitive Science, 12(2), 257–285. https://doi.org/10.1016/0364-0213(88)90023-7

Wee, L. K., Tan, D., Clemente, F. J. G., & Esquembre, F. (2024). Easy JavaScript Simulation (EJSS) Data Analytics for Singapore. Journal of Physics: Conference Series, 2693(1), 012017. https://doi.org/10.1088/1742-6596/2693/1/012017

Widyawati, N., Khasanah, M., Muttaqin, Rasywir, E., & Feranika, A. (2022). Prediksi Tingkat Stress Pada Mahasiswa Universitas Dinamika Bangsa Jambi Dalam Melakukan Perkuliahan Metode Hybrid Menggunakan Algoritma Naive Bayes. Jurnal Manajemen Teknologi Dan Sistem Informasi (JMS), 2(1), 99–109. https://doi.org/10.33998/jms.2022.2.1.44