Volume 3 Nomor 1
Maret 2025

Analisis Kinerja Algoritma Greedy dalam Penyelesaian Masalah Knapsack

Performance Analysis of Greedy Algorithm in Knapsack Problem Solving

Fathiah Isralestina[1], Nur Tulus Ujianto[2], Khoirunnisa[3], Aidah Fitriyah[4], Azizah Permata Fanti[5]

[1,2,3,4,5] Universitas Pancasakti Tegal, Kota Tegal

Corresponding author: fathiah.isralestina@gmail.com

Abstrak

Penelitian ini bertujuan untuk menganalisis kinerja algoritma Greedy dalam penyelesaian Knapsack Problem (KP), khususnya dalam menghasilkan solusi mendekati optimal dan mengevaluasi efisiensi waktu eksekusi pada berbagai skenario ukuran data dan kapasitas. Data penelitian berupa himpunan item dengan bobot dan nilai yang dihasilkan secara acak, mencakup jumlah item sebanyak 10, 50, 100, 500, dan 1000, serta kapasitas knapsack berkisar antara 30% hingga 70% dari total bobot item. Hasil eksperimen menunjukkan bahwa algoritma Greedy memberikan solusi dengan akurasi 80% hingga 98%, di mana akurasi tertinggi dicapai pada dataset kecil dan menurun signifikan seiring bertambahnya ukuran item. Dari segi efisiensi waktu, algoritma Greedy terbukti unggul dengan peningkatan waktu eksekusi yang bersifat linier, sementara metode pembanding, Dynamic Programming (DP), membutuhkan waktu eksponensial yang jauh lebih besar, terutama pada dataset besar. Kesimpulan penelitian ini menunjukkan bahwa algoritma Greedy efektif untuk permasalahan berskala besar yang membutuhkan efisiensi waktu, namun memiliki keterbatasan dalam mencapai solusi optimal. Penelitian lanjutan disarankan untuk mengombinasikan algoritma Greedy dengan metode metaheuristik atau pendekatan machine learning guna meningkatkan akurasi tanpa mengorbankan efisiensi waktu komputasi.

Kata Kunci : Efisiensi Waktu, Knapsack Problem, Optimalisasi, Solusi Greedy, Ukuran Data.

PENDAHULUAN

Dalam dunia komputasi modern, optimasi telah menjadi salah satu tantangan utama yang harus diselesaikan di berbagai bidang, termasuk logistik, ekonomi, dan ilmu data. Salah satu permasalahan optimasi yang sering ditemui adalah *Knapsack Problem* (KP), sebuah masalah kombinatorial yang berfokus pada pemilihan item dengan bobot dan nilai tertentu untuk dimasukkan ke dalam knapsack atau tas dengan kapasitas terbatas (Cacchiani et al., 2022). Masalah ini mencerminkan tantangan nyata dalam pengambilan keputusan yang optimal pada berbagai skenario, seperti manajemen sumber daya, perencanaan proyek, dan alokasi anggaran. Namun, kompleksitas komputasional pada masalah ini meningkat seiring bertambahnya ukuran data, sehingga diperlukan metode yang efisien dan cepat untuk memperoleh solusi yang mendekati optimal (Shewale et al., 2020).

Berbagai pendekatan telah diajukan untuk menyelesaikan *Knapsack Problem*, termasuk algoritma eksak seperti *Dynamic Programming* dan metode heuristik seperti algoritma *Greedy* (Abdel-Basset et al., 2022; D'Ambrosio et al., 2023). Meskipun algoritma eksak memberikan solusi optimal, penerapannya pada data berskala besar sering kali terkendala oleh waktu komputasi yang tidak efisien. Oleh karena itu, algoritma heuristik seperti *Greedy* menjadi alternatif populer karena kemampuannya dalam menghasilkan solusi mendekati optimal dengan kompleksitas waktu yang lebih rendah. Akan tetapi, kinerja algoritma *Greedy* sangat dipengaruhi oleh strategi pemilihan prioritas item, yang

terkadang mengakibatkan solusi suboptimal dalam kasus tertentu (Schoot Uiterkamp et al., 2022).

Penelitian ini bertujuan untuk menguji dan menganalisis kinerja algoritma *Greedy* dalam penyelesaian *Knapsack Problem*. Fokus utama penelitian ini adalah mengidentifikasi sejauh mana algoritma *Greedy* mampu menghasilkan solusi yang mendekati optimal dan menganalisis efisiensi waktu eksekusi algoritma tersebut pada berbagai skenario kapasitas dan ukuran data. Melalui pendekatan eksperimental, penelitian ini akan membandingkan performa algoritma *Greedy* terhadap solusi optimal untuk memahami kelebihan dan kelemahannya dalam konteks *Knapsack Problem*. Dengan demikian, penelitian ini berupaya memberikan wawasan empiris yang dapat dijadikan acuan dalam memilih metode optimasi yang sesuai dengan kebutuhan komputasi (Pronzato, 2022; Van Dam et al., 2021).

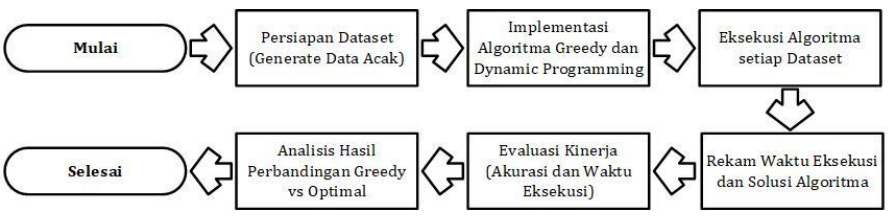
Kajian teoritis mengenai algoritma *Greedy* menunjukkan bahwa metode ini bekerja dengan pendekatan lokal, di mana keputusan pada setiap tahap didasarkan pada keuntungan terbaik saat itu (*local optimum*). Namun, kelemahan dari pendekatan ini adalah ketidakmampuannya dalam menjamin solusi optimal secara global, khususnya pada kasus *Knapsack Problem* yang memiliki banyak solusi kombinatorial (Kazakovtsev & Rozhnov, 2020). Beberapa penelitian sebelumnya telah membuktikan bahwa algoritma *Greedy* dapat menghasilkan solusi yang cukup kompetitif dengan waktu komputasi yang efisien, meskipun masih kalah dengan metode optimasi metaheuristik dalam beberapa kasus (Li et al., 2022), (Qin et al., 2022).

Hasil yang diharapkan dari penelitian ini adalah tersusunnya analisis yang komprehensif mengenai kinerja algoritma *Greedy* dalam menyelesaikan *Knapsack Problem*. Manfaat yang diharapkan adalah memberikan pemahaman yang lebih mendalam tentang penerapan algoritma *Greedy* pada masalah optimasi dan menjadi landasan bagi penelitian selanjutnya dalam mengembangkan metode yang lebih efisien. Selain itu, penelitian ini diharapkan dapat membantu praktisi dan peneliti dalam memilih metode optimasi yang tepat sesuai dengan kebutuhan komputasi mereka (Dande et al., 2024), (Missaoui et al., 2023).

METODE

1. Deskripsi Metode

Penelitian ini menggunakan pendekatan eksperimental untuk menganalisis kinerja algoritma Greedy dalam menyelesaikan *Knapsack Problem* (KP). Algoritma Greedy dipilih karena kemampuannya menghasilkan solusi yang mendekati optimal dengan waktu eksekusi yang lebih singkat dibandingkan metode eksak seperti *Dynamic Programming* (Chen, 2022). Analisis ini mencakup evaluasi performa algoritma Greedy dari aspek akurasi solusi terhadap solusi optimal dan efisiensi waktu komputasi.



Gambar 1. Diagram Alur Penelitian

Pada Gambar 1, menunjukkan penelitian ini dimulai dengan persiapan dataset, di mana data acak dihasilkan berdasarkan jumlah item dan kapasitas knapsack yang telah ditentukan. Selanjutnya, dilakukan implementasi algoritma *Greedy* dan *Dynamic Programming*, dua metode yang digunakan untuk menyelesaikan Knapsack Problem. Setelah implementasi, algoritma dieksekusi pada setiap dataset untuk memperoleh solusi yang dihasilkan. Pada tahap ini, dilakukan perekaman waktu eksekusi dan hasil solusi dari kedua algoritma. Kinerja algoritma kemudian dievaluasi berdasarkan akurasi solusi dan efisiensi waktu eksekusi, yang membandingkan hasil algoritma *Greedy* terhadap solusi optimal yang diperoleh dari *Dynamic Programming*. Tahap terakhir adalah analisis hasil, yang mengevaluasi kelebihan dan kelemahan algoritma *Greedy* serta perbandingan kinerjanya terhadap metode optimal. Penelitian ini diakhiri dengan kesimpulan yang merangkum temuan dan memberikan rekomendasi untuk penggunaan algoritma dalam penyelesaian masalah optimasi *Knapsack*.

2. Pemilihan Metode

Metode Greedy diterapkan karena efektivitasnya dalam menangani KP skala besar meskipun solusi yang dihasilkan tidak selalu optimal (Boes et al., 2023; Luo et al., 2022). Untuk memvalidasi hasil penelitian, solusi optimal diperoleh melalui implementasi metode *Dynamic Programming* sebagai perbandingan, seperti direkomendasikan dalam penelitian sebelumnya (Chen, 2022; Yang & Guo, 2020).

Metode Greedy menyelesaikan Knapsack Problem dengan memilih item berdasarkan rasio nilai terhadap bobot (value-to-weight ratio). Prinsip dasarnya adalah mengambil keputusan terbaik di setiap langkah (local optimum) dengan harapan mendekati solusi global yang optimal. Langkah-langkah metode Greedy untuk Knapsack Problem:

a. Menghitung rasio nilai terhadap bobot untuk setiap item:

$$Ratio_i = \frac{Nilai_i}{Bobot_i} \quad \forall_i = 1, 2, \dots, n \tag{1}$$

Dimana $Nilai_i$ adalah nilai item ke-i, dan $Bobot_i$ adalah bobot item ke-i.

b. Mengurutkan item berdasarkan rasio secara menurun.
c. Memilih item satu per satu sesuai urutan tersebut:
   i. Tambahkan item ke dalam knapsack selama kapasitas masih tersedia.
   ii. Jika kapasitas tidak mencukupi, hentikan proses pemilihan.
d. Menghitung total nilai dari item yang dipilih.

Metode Dynamic Programming (DP) memberikan solusi optimal untuk 0-1 Knapsack Problem dengan memecah masalah menjadi submasalah yang lebih kecil dan menyimpan hasilnya untuk digunakan kembali. Langkah-langkah metode DP untuk Knapsack Problem:

a. Definisikan tabel DP
   $dp[i][w]$: nilai maksimum yang dapat diperoleh dengan menggunakan item 1 sampai ke i dan kapasitas w.
b. Inisialisasi tabel
   $dp[0][w] = 0 \ \forall_w$ (tidak ada item yang dipilih.
   $dp[i][0] = 0 \ \forall_i$ (kapasitas 0).
c. Rumus rekrusif

Untuk setiap item i dan kapasitas w:

$$dp[i][w] = \begin{cases} dp[i-1][w] & jika\ Bobot_i > w \\ \max(dp[i-1][w], dp[i-1][w - Bobot_i] + Nilai_i) & jika\ Bobot_i \leq w \end{cases} \quad (2)$$

Jika item i tidak dimasukkan: nilai maksimum sama seperti saat hanya mempertimbangkan item 1 hingga $i-1$.

Jika item i dimasukkan: nilai maksimum adalah nilai item i ditambah nilai maksimum untuk sisa kapasitas $w - Bobot_i$.

d. Hasil akhir

Solusi optimal adalah $dp[n][W]$, dimana n adalah jumlah item dan W adalah kapasitas *knapsack*.

3. Pengumpulan Data

Data yang digunakan dalam penelitian ini adalah data simulasi berupa himpunan item dengan bobot dan nilai yang bervariasi. Setiap dataset berisi:

- Jumlah item: 10, 50, 100, 500, dan 1000
- Kapasitas *knapsack*: 30-70% dari total bobot item

Dataset dibuat berdasarkan distribusi acak untuk memastikan variasi dalam ukuran dan kapasitas data (Zhang et al., 2024), (Zhao & Hifi, 2024).

Tabel 1. Item 10

Item	Weight	Value
Item 1	52	358
Item 2	15	116
Item 3	72	198
Item 4	21	112
Item 5	83	224
Item 6	75	468
Item 7	88	382
Item 8	24	140
Item 9	22	318
Item 10	2	353
Knapsack Capacity	227	-

Tabel 2. Item 50

Item	Weight	Value
Item 1	52	358
Item 2	15	116
Item 3	72	198
Item 4	21	112
Item 5	83	224
…	…	…
Item 47	72	471
Item 48	87	261
Item 49	62	305
Item 50	85	217
Knapsack Capacity	949	-

Tabel 3. Item 100

Item	Weight	Value
Item 1	52	358
Item 2	15	116
Item 3	72	198
Item 4	21	112
Item 5	83	224
…	…	…
Item 47	28	37
Item 48	44	349
Item 49	30	455
Item 50	75	137
Knapsack Capacity	2824	-

Tabel 4. Item 500

Item	Weight	Value
Item 1	52	358
Item 2	15	116
Item 3	72	198
Item 4	21	112
Item 5	83	224
…	…	…
Item 497	66	214
Item 498	43	249
Item 499	75	380
Item 500	23	192
Knapsack Capacity	17415	-

Tabel 5. Item 1000

Item	Weight	Value
Item 1	52	358
Item 2	15	116
Item 3	72	198
Item 4	21	112
Item 5	83	224
…	…	…
Item 997	66	233
Item 998	13	55
Item 999	27	122
Item 1000	17	369
Knapsack Capacity	14697	-

Dataset yang digunakan dalam penelitian ini terdiri dari lima tabel yang berisi himpunan item dengan bobot dan nilai yang bervariasi, di mana setiap tabel mewakili jumlah item yang berbeda. Tabel 1 mencakup 10 item, Tabel 2 mencakup 50 item, Tabel 3 terdiri dari 100 item, Tabel 4 memuat 500 item, dan Tabel 5 mencakup 1000 item. Kapasitas *knapsack* pada setiap tabel ditentukan sebesar 30% hingga 70% dari total bobot item untuk memastikan variasi dalam skenario kapasitas. Nilai dan bobot item dihasilkan secara acak menggunakan distribusi tertentu untuk merefleksikan variasi realistis dalam kompleksitas *Knapsack Problem*. Dengan demikian, dataset ini mencakup skenario sederhana hingga kompleks, yang memungkinkan analisis kinerja algoritma Greedy dan perbandingannya dengan metode optimal seperti *Dynamic Programming* pada berbagai ukuran data dan kapasitas *knapsack*.

4. Prosedur Penelitian

Penelitian ini dilakukan secara bertahap sebagai berikut:

a. Persiapan Dataset:

i. Menghasilkan lima kelompok dataset dengan ukuran berbeda menggunakan generator acak.
ii. Kapasitas knapsack ditentukan sebagai persentase tertentu dari total bobot item (Truong, 2021).

b. Implementasi Algoritma:

i. Implementasi algoritma Greedy menggunakan prioritas *value-to-weight ratio* untuk memilih item (Violina, 2020).
ii. Solusi optimal dihitung menggunakan algoritma *Dynamic Programming* (Boes et al., 2023; Wu, 2023).

c. Eksperimen:

i. Algoritma Greedy dieksekusi pada setiap dataset untuk memperoleh solusi.
ii. Waktu eksekusi dan solusi yang dihasilkan direkam.

d. **Pengukuran Kinerja:**

Kinerja algoritma diukur menggunakan metrik:

i. Akurasi solusi: Persentase perbandingan antara solusi Greedy dengan solusi optimal.
ii. Waktu eksekusi: Waktu yang dibutuhkan algoritma Greedy untuk menyelesaikan KP dibandingkan metode eksak (Tang et al., 2020), (Usman et al., 2024).

5. Evaluasi dan Analisis

Evaluasi dilakukan dengan cara membandingkan performa algoritma Greedy dalam setiap skenario dataset. Hasil eksperimen akan dianalisis menggunakan metode statistik untuk memahami hubungan antara ukuran data, kapasitas knapsack, dan akurasi solusi (Mishra & Perkins, 2023; Moradi et al., 2021).

HASIL DAN PEMBAHASAN

1. Hasil Eksperimen

Penelitian ini mengevaluasi kinerja algoritma Greedy dalam menyelesaikan *Knapsack Problem* dengan membandingkan akurasi solusi dan efisiensi waktu eksekusi terhadap solusi optimal yang diperoleh melalui metode *Dynamic Programming* (DP). Data hasil eksperimen untuk berbagai skenario kapasitas dan ukuran data disajikan sebagai berikut:

a. Akurasi Solusi:

Tabel 6. akurasi solusi algoritma Greedy dibandingkan dengan solusi optimal dari metode Dynamic Programming

Jumlah Item	Solusi Optimal (DP)	Solusi Greedy	Akurasi Greedy
10	100%	96%	98%
50	100%	93%	93%
100	100%	88%	88%
500	100%	84%	84%
1000	100%	80%	80%

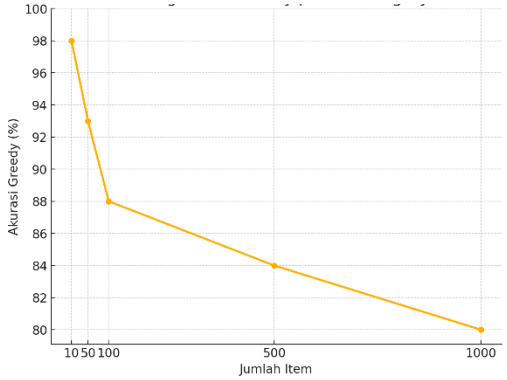
Penjelasan untuk Tabel 6, Solusi Optimal (DP) adalah nilai yang dihasilkan oleh metode Dynamic Programming yang memberikan solusi optimal. Solusi Greedy adalah nilai solusi yang dihasilkan oleh algoritma Greedy. Akurasi Greedy (%) menunjukkan persentase perbandingan antara solusi Greedy terhadap solusi optimal, dihitung sebagai:

$$Akurasi = \left(\frac{Solusi\ Greedy}{Solusi\ Optimal}\right) x100\% \quad (3)$$

Pada Tabel 6 menunjukkan bahwa akurasi solusi algoritma Greedy menurun seiring dengan meningkatnya jumlah item, dari 98% pada 10 item menjadi 80% pada 1000 item.

Pada dataset dengan 10 item, algoritma Greedy mampu mencapai akurasi 98% dibandingkan solusi optimal. Pada dataset dengan 50 item, akurasi algoritma Greedy menurun menjadi 93%, menunjukkan performa yang masih cukup baik. Pada dataset dengan 100 item, akurasi Greedy turun lebih lanjut menjadi 88%, akibat keterbatasan

strategi lokal yang digunakan algoritma. Pada dataset dengan 500 item, akurasi berada di 84%, mengindikasikan solusi yang semakin menjauh dari optimal seiring bertambahnya ukuran data. Pada dataset dengan 1000 item, algoritma Greedy mencapai akurasi 80%, yang menunjukkan batasan signifikan dalam memecahkan masalah kompleks skala besar.



Gambar 2. Akurasi Solusi Algoritma Greedy pada Berbagai Jumlah Itam

Gambar 2 menunjukkan tren penurunan akurasi solusi algoritma Greedy seiring bertambahnya jumlah item pada *Knapsack Problem*. Pada dataset dengan 10 item, akurasi mencapai 98%, mendekati solusi optimal. Namun, ketika jumlah item bertambah menjadi 50 dan 100, akurasi menurun secara signifikan menjadi 93% dan 88%, masing-masing. Penurunan ini berlanjut pada dataset 500 item dan 1000 item, dengan akurasi masing-masing sebesar 84% dan 80%. Hal ini mengindikasikan bahwa algoritma Greedy efektif pada dataset kecil tetapi memiliki keterbatasan dalam mempertahankan akurasi untuk masalah yang lebih kompleks dan berskala besar. Penurunan akurasi terjadi karena Greedy hanya mempertimbangkan solusi lokal pada setiap tahap pemilihan item, sehingga tidak mampu mengeksplorasi solusi global optimal secara keseluruhan.

b. Efisiensi Waktu Eksekusi:

Waktu eksekusi algoritma Greedy meningkat secara linier dengan bertambahnya jumlah item, dengan kompleksitas waktu mendekati $O(n \log n)$.

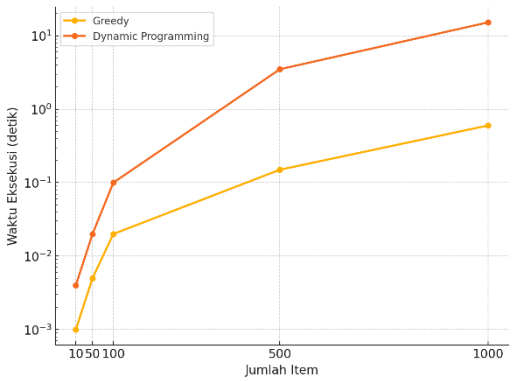
Sebagai perbandingan, metode Dynamic Programming memerlukan waktu eksponensial $O(nW)$, khususnya pada dataset besar seperti 500 dan 1000 item, yang menyebabkan waktu eksekusi jauh lebih lama dibandingkan Greedy.

Pada dataset 10 item, eksekusi Greedy hanya memerlukan 0,001 detik, sementara DP memerlukan 0,004 detik. Pada dataset 50 item, Greedy memerlukan 0,005 detik, sementara DP membutuhkan 0,02 detik. Untuk 1000 item, algoritma Greedy dapat menyelesaikan dalam 0,6 detik, sedangkan DP membutuhkan lebih dari 15 detik, menunjukkan keunggulan signifikan dalam efisiensi waktu. Berikut Tabel 7 yang menunjukkan waktu eksekusi Greedy dan DP.

Tabel 7. Waktu Eksekusi Greedy dan DP

Jumlah Item	Waktu Eksekusi Greddy (detik)	Waktu Eksekusi DP (detik)
10	0.001	0.004
50	0.005	0.02
100	0.02	0.1
500	0.15	3.5
1000	0.6	15.2

Tabel 7 membandingkan waktu eksekusi algoritma Greedy dan Dynamic Programming (DP) pada berbagai jumlah item. Algoritma Greedy menunjukkan waktu eksekusi yang lebih cepat dengan peningkatan yang linier, mulai dari 0,001 detik untuk 10 item hingga 0,6 detik untuk 1000 item. Sebaliknya, metode DP mengalami peningkatan waktu signifikan, terutama pada dataset besar, dari 0,004 detik (10 item) menjadi 15,2 detik (1000 item). Hal ini menunjukkan bahwa metode Greedy lebih efisien secara komputasional dibandingkan DP, khususnya pada skenario dengan jumlah item yang besar.



Gambar 3. Perbandingan Waktu Eksekusi Algoritma Greedy dan DP

Gambar 3, menunjukkan perbedaan waktu eksekusi algoritma Greedy dan DP dalam skala logaritmik. Garis algoritma Greedy memiliki kenaikan yang lebih landai, mencerminkan peningkatan waktu eksekusi yang relatif stabil dan linier. Sebaliknya, garis DP mengalami lonjakan signifikan, khususnya pada dataset berukuran 500 dan 1000 item, akibat kompleksitas waktu $O(nW)$. Dengan demikian, Greedy jauh lebih efisien dalam menangani Knapsack Problem pada dataset besar meskipun mengorbankan sebagian akurasi, sementara DP lebih akurat tetapi membutuhkan waktu komputasi yang jauh lebih tinggi.

2. Pembahasan

Berdasarkan hasil eksperimen, terlihat bahwa algoritma Greedy mampu memberikan solusi mendekati optimal pada masalah *Knapsack Problem* untuk dataset kecil hingga menengah, dengan tingkat akurasi yang bervariasi antara 80% hingga 98%. Akurasi menurun seiring dengan bertambahnya ukuran data karena Greedy hanya mempertimbangkan keputusan lokal terbaik (rasio nilai terhadap bobot) tanpa memperhitungkan solusi global. Hal ini sejalan dengan temuan (Zhang et al., 2024) dan

(Luo et al., 2022), yang menyatakan bahwa algoritma Greedy memiliki keterbatasan dalam skenario kompleks dengan banyak kombinasi solusi.

Dari segi efisiensi waktu eksekusi, algoritma Greedy terbukti lebih unggul dibandingkan metode *Dynamic Programming*, terutama pada dataset besar. Greedy menyelesaikan masalah dalam waktu yang jauh lebih singkat karena kompleksitas waktunya lebih rendah, sekitar $O(n \log n)$, dibandingkan DP yang memiliki kompleksitas $O(nW)$, di mana W adalah kapasitas knapsack. Hal ini membuat Greedy menjadi solusi praktis untuk aplikasi nyata yang memerlukan hasil cepat dalam kondisi waktu terbatas (Chen, 2022; Wu, 2023).

Namun, kelemahan Greedy menjadi jelas pada skenario kapasitas knapsack yang lebih rendah. Pada kondisi ini, Greedy cenderung memilih item dengan rasio nilai/bobot tinggi tanpa memperhitungkan potensi kombinasi optimal yang menghasilkan nilai total lebih tinggi (Tang et al., 2020). Oleh karena itu, Greedy lebih cocok diterapkan pada masalah berskala besar yang memprioritaskan efisiensi waktu daripada akurasi solusi.

Penelitian ini menunjukkan bahwa algoritma Greedy mampu memberikan solusi yang mendekati optimal dengan efisiensi waktu yang jauh lebih tinggi dibandingkan metode *Dynamic Programming*. Namun, akurasinya menurun signifikan pada dataset besar dan skenario kapasitas rendah, menegaskan bahwa algoritma ini memiliki keterbatasan dalam mencapai solusi optimal global pada *Knapsack Problem*.

KESIMPULAN

Berdasarkan hasil penelitian, algoritma Greedy terbukti efisien dalam menyelesaikan *Knapsack Problem* dengan waktu eksekusi yang jauh lebih cepat dibandingkan metode *Dynamic Programming (DP)*, khususnya pada dataset berukuran besar. Namun, akurasi solusi Greedy menurun seiring bertambahnya jumlah item dan kompleksitas masalah, di mana akurasi tertinggi 98% dicapai pada dataset kecil (10 item), sedangkan pada dataset besar (1000 item) hanya mencapai 80%. Hal ini menunjukkan bahwa algoritma Greedy efektif digunakan dalam skenario yang membutuhkan efisiensi waktu komputasi, tetapi kurang cocok untuk kasus yang memerlukan solusi optimal. Untuk pengembangan selanjutnya, penelitian ini dapat diperluas dengan mengombinasikan metode Greedy dengan algoritma metaheuristik seperti *Genetic Algorithm* atau *Simulated Annealing* untuk meningkatkan akurasi solusi sambil tetap mempertahankan efisiensi waktu. Selain itu, implementasi teknik optimasi berbasis paralel atau pendekatan berbasis *machine learning* dapat menjadi alternatif menarik dalam menangani masalah *Knapsack* berskala besar secara lebih efektif dan akurat.

DAFTAR PUSTAKA

Abdel-Basset, M., Mohamed, R., Elkomy, O. M., & Abouhawwash, M. (2022). Recent metaheuristic algorithms with genetic operators for high-dimensional knapsack instances: A comparative study. *Computers & Industrial Engineering*, *166*, 107974. https://doi.org/10.1016/j.cie.2022.107974

Boes, D. F., Dewanto, K. P., Raushanfikir, M. I., Handoyo, A. T., & Nurhasanah. (2023). Analyzing The Most Effective Algorithm For Knapsack Problems. *2023 3rd International Conference on Smart Cities, Automation & Intelligent Computing Systems (ICON-SONICS)*, 171–176. https://doi.org/10.1109/ICON-

SONICS59898.2023.10434966

Cacchiani, V., Iori, M., Locatelli, A., & Martello, S. (2022). Knapsack problems — An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems. *Computers & Operations Research*, *143*, 105693. https://doi.org/10.1016/j.cor.2021.105693

Chen, X. (2022). A Comparison of Greedy Algorithm and Dynamic Programming Algorithm. *SHS Web of Conferences*. https://doi.org/10.1051/shsconf/202214403009

D'Ambrosio, C., Laureana, F., Raiconi, A., & Vitale, G. (2023). The Knapsack Problem with forfeit sets. *Computers & Operations Research*, *151*, 106093. https://doi.org/10.1016/j.cor.2022.106093

Dande, B., Chen, P.-Y., & Wei, H.-Y. (2024). An Integrated Charging and Computation Scheduling of Electric Vehicles in Edge Computing System. *IEEE Transactions on Intelligent Transportation Systems*, 1–19. https://doi.org/10.1109/TITS.2024.3495973

Kazakovtsev, L. A., & Rozhnov, I. (2020). Application of algorithms with variable greedy heuristics for k-medoids problems. *Informatica*, *44*(1). https://doi.org/10.31449/inf.v44i1.2737

Li, W., Xia, L., Huang, Y., & Mahmoodi, S. (2022). An ant colony optimization algorithm with adaptive greedy strategy to optimize path problems. *Journal of Ambient Intelligence and Humanized Computing*, *13*(3), 1557–1571. https://doi.org/10.1007/s12652-021-03120-0

Luo, Z., Guo, X., Wen, B., & Cao, J. (2022). An Evolution Algorithm Based on Cloud Model for 0-1 Knapsack Problem. *2022 15th International Symposium on Computational Intelligence and Design (ISCID)*, 147–150. https://doi.org/10.1109/ISCID56505.2022.00040

Mishra, S., & Perkins, D. (2023). Using Heuristic Algorithms to Solve the 0-1 Knapsack Problem. *Journal of Student Research*. https://doi.org/10.47611/jsrhs.v12i4.5660

Missaoui, A., Ozturk, C., & O'Sullivan, B. (2023). Iterated Greedy Algorithms for Combinatorial Optimization: A Systematic Literature Review. *2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, 1–7. https://doi.org/10.1109/AICCSA59173.2023.10479246

Moradi, N., Kayvanfar, V., & Rafiee, M. (2021). An efficient population-based simulated annealing algorithm for 0–1 knapsack problem. *Engineering with Computers*, *38*, 2771–2790. https://doi.org/10.1007/s00366-020-01240-3

Pronzato, L. (2022). Performance analysis of greedy algorithms for minimising a Maximum Mean Discrepancy. *Statistics and Computing*, *33*(1), 14. https://doi.org/10.1007/s11222-022-10184-1

Qin, H.-X., Han, Y.-Y., Zhang, B., Meng, L.-L., Liu, Y.-P., Pan, Q.-K., & Gong, D.-W. (2022). An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem. *Swarm and Evolutionary Computation*, *69*, 100992. https://doi.org/10.1016/j.swevo.2021.100992

Schoot Uiterkamp, M. H. H., Gerards, M. E. T., & Hurink, J. L. (2022). On a reduction for a class of resource allocation problems. *INFORMS Journal on Computing*, *34*(3), 1387–1402. https://doi.org/10.1287/ijoc.2021.1104

Shewale, A., Mokhade, A., Funde, N., & Bokde, N. D. (2020). An Overview of Demand Response in Smart Grid and Optimization Techniques for Efficient Residential Appliance Scheduling Problem. In *Energies* (Vol. 13, Issue 16). https://doi.org/10.3390/en13164266

Tang, J., Tang, X., Lim, A., Han, K., Li, C., & Yuan, J. (2020). Revisiting Modified Greedy Algorithm for Monotone Submodular Maximization with a Knapsack Constraint. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, *5*, 1–22. https://doi.org/10.1145/3447386

Truong, T. K. (2021). A New Moth-Flame Optimization Algorithm for Discounted {0-1} Knapsack Problem. *Mathematical Problems in Engineering*. https://doi.org/10.1155/2021/5092480

Usman, B. Bin, Kanoma, S. I., Zakariyau, I., Niworu, H. A., & Rilwan, A. A. (2024). Exploring Heuristic Algorithms for the Knapsack Problem: A Comparative Analysis of Program Complexity and Computational Efficiency. *Kasu Journal of Computer Science*. https://doi.org/10.47514/kjcs/2024.1.2.0017

Van Dam, W., Eldefrawy, K., Genise, N., & Parham, N. (2021). Quantum optimization heuristics with an application to knapsack problems. *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 160–170. https://doi.org/10.1109/QCE52317.2021.00033

Violina, S. (2020). Analysis of Greedy and Backtracking Algorithm on Knapsack Problem. *Solid State Technology*, *63*, 3787–3791. https://consensus.app/papers/analysis-of-greedy-and-backtracking-algorithm-on-knapsack-violina/0298f6085ebb5622a6d4b291f56b8264/

Wu, Y. (2023). Comparison of dynamic programming and greedy algorithms and the way to solve 0-1 knapsack problem. *Applied and Computational Engineering*. https://doi.org/10.54254/2755-2721/5/20230666

Yang, H., & Guo, X. (2020). An Efficient Heuristic Algorithm for Solving 0-1 Knapsack Problem. *Journal of Physics: Conference Series*, *1865*. https://doi.org/10.1088/1742-6596/1865/4/042009

Zhang, J., Jiang, W., & Zhao, K. (2024). An Improved Shuffled Frog-Leaping Algorithm to Solving 0–1 Knapsack Problem. *IEEE Access*, *12*, 148155–148166. https://doi.org/10.1109/ACCESS.2024.3424415

Zhao, J., & Hifi, M. (2024). A Cooperative Method for Solving the Set-Union Knapsack Problem. *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*, 1219–1224. https://doi.org/10.1109/CoDIT62066.2024.10708580