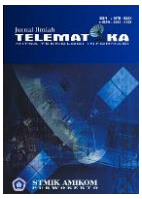


Terbit *online* pada laman web jurnal :
http://ejournal.amikompurwokerto.ac.id/index.php/telematika/

Telematika

Akreditasi KEMENRISTEKDIKTI, No. 21/E/KPT/2018

Penyelesaian *Integer Knapsack Problem* Menggunakan Algoritma *Greedy, Dynamic Programming, Brute Force* dan *Genetic*

Muhammad Abdurrahman Rois[1], Siti Maslihah[2], dan Budi Cahyono[3]

[1,2,3]Program Studi Matematika
[1,2,3]Fakultas Sains dan Teknologi
[1]Universitas Brawijaya
[2,3]UIN Walisongo Semarang
Email : roizmuhammad.math@gmail.com[1], sitimaslihah@walisongo.ac.id[2], budi.cahyono@walisongo.ac.id[3]

INFO ARTIKEL

Sejarah artikel:
Menerima 6 Juni 2019
Revisi 28 Juni 2019
Diterima 13 Agustus 2019
Online Agustus 2019

Kata kunci:
Integer knapsack problem
Greedy
Dynamic programming
Brute force
Genetic

ABSTRAK

Integer knapsack problem adalah masalah yang ada pada riset operasi di bab program bilangan bulat, dimana bertujuan untuk memaksimalkan total nilai barang ke tempat yang diinginkan dengan memiliki batasan tertentu. Keputusannya ada 2 yaitu 1 (diambil) dan 0 (tidak diambil). Data yang digunakan untuk input merupakan data hasil wawancara dengan J&T Express *drop point* Ngaliyan, dan penelitian ini terbagi menjadi beberapa penjelasan yaitu (1) *Integer knapsack problem*, (2) Penyelesaian *integer knapsack problem* menggunakan algoritma *greedy, dynamic programming, brute force, genetic* dan implementasi keempat algoritma tersebut pada *software* MATLAB *v2017a* berbasis GUI, (3) Membandingkan keempat algoritma dalam hal solusi dan waktu komputasi yang optimal. Hasil penelitian ini menghasilkan kesimpulan bahwa algoritma yang efektif dan efisien digunakan untuk data skala kecil ataupun besar adalah algoritma *dynamic programming*. Penelitian ini juga dapat memberikan wawasan tentang penyelesaian alternatif untuk memecahkan *integer knapsack problem.*

Keywords:
Integer knapsack problem
Greedy
Dynamic programming
Brute force
Genetic

ABSTRACT

Integer knapsack problem is a problem that exists in operating research in integer program chapters, which aims to maximize the total value of goods to the desired place by having certain limitations. The decision is 2, namely 1 (taken) and 0 (not taken). The data used for input is data from interviews with J&T Express drop point Ngaliyan and this research is divided into several explanations, namely (1) Integer knapsack problem, (2) Completion of integer knapsack problems using greedy algorithms, dynamic programming, brute force, genetics and implementation of these four algorithms in GUI based MATLAB v2017a software, (3) Comparing the four algorithms in terms of solutions and optimal computing time. The results of this study conclude that an effective and efficient algorithm used for small or large scale data is a dynamic programming algorithm. This study can also provide insight into alternative solutions to solve integer knapsack problems.

Korespondensi:
Telepon: +62 81914430369
E-mail:
roizmuhammad.math@gmail.com

PENDAHULUAN

Kemajuan di bidang teknologi informasi membuat dunia pada perkembangan baru dari masa ke masa termasuk pengiriman barang, pelayanan dari jasa pengiriman barang sangat penting karena membantu manusia dalam memudahkan aktifitas dan lebih menghemat waktu ataupun tenaga (Nuraeni, 2016). Proses pengiriman barang pasti mengeluarkan biaya dalam proses pengirimannya, apalagi jarak

http://dx.doi.org/10.35671/telematika.v12i2.841

antara tempat pengiriman yang satu dengan tempat pengiriman lainnya berbeda-beda dan cukup jauh. Biaya yang dikeluarkan supaya minimal dan memperoleh keuntungan maksimal, maka barang-barang yang dikirimkan perlu dipilih secermat mungkin. Karena masyarakat di zaman ini sangat antusias untuk melakukan pengiriman barang, maka banyak berdiri perusahaan jasa pengiriman barang. Jasa pengiriman barang lebih mendahulukan jarak lokasi pengiriman yang lebih jauh karena memiliki nilai yang besar. Pada jasa pengiriman barang memiliki banyak jenis paket pengiriman yang disesuaikan kebutuhan konsumen antara lain paket reguler. Paket ini salah satu paket yang paling digemari konsumen karena faktor harga yang ekonomis dibanding paket-paket lainnya. Paket reguler dapat menjangkau seluruh Indonesia dalam batas jangka waktu maksimal 7 hari. Hal ini disebabkan karena keterbatasan banyaknya kurir tidak sebanding dengan banyaknya barang yang akan dikirimkan. Konsekuensinya barang harus dikirimkan berangsur-angsur berdasarkan nilai yang lebih besar dahulu. Sehingga dengan demikian paket reguler ini sangat cocok digunakan untuk kasus *knapsack problem.*

Masalah *knapsack* atau *rucksack problem* adalah masalah optimasi kombinatorial untuk mencari solusi terbaik dari banyak solusi kemungkinan yang ada (Messac, 2015). Masalah *knapsack* muncul jika memiliki *n* buah barang yang tidak semuanya dapat dimasukkan dalam suatu tempat misalnya tas atau ransel. Sejumlah barang yang tersedia, masing-masing memiliki berat dan nilai yang berbeda-beda. Masalahnya adalah memilih barang-barang yang dibawa dengan keterbatasan kapasitas (keterbatasan tempat) agar total berat tidak melebihi kapasitas tempatnya dan nilai yang dihasilkannya sebesar mungkin (Siang, 2014). Jenis-jenis *knapsack problem* ada 3 yaitu *unbounded knapsack problem* (tidak ada batasan jumlah barang untuk setiap barang), *integer knapsack problem* (jumlah barang untuk setiap barang hanya boleh 0 atau 1), dan *fractional knapsack problem* (jumlah barang untuk setiap barang boleh pecahan). Pada penelitian ini digunakan *integer knapsack problem* yaitu semua barang diasumsikan berjumlah 1 paket barang atau unit dan tidak bisa dipecah-pecah. Masalah *knapsack* tersebut dapat dipelajari dalam bab program bilangan bulat dan terdapat di riset operasi.

Program bilangan bulat merupakan kasus khusus program linier untuk menyelesaikan masalah program linier yang penyelesaiannya harus bilangan bulat (Tarliah & Dimyati, 2016). Bentuk ini muncul karena dalam kenyataannya tidak semua variabel keputusan dapat berupa bilangan pecahan. Selanjutnya, program bilangan bulat dipelajari di riset operasi. Riset operasi/ *Operation Research* (OR) adalah aplikasi metode ilmiah untuk memecahkan persoalan dengan masukan (*input*) yang terbatas sehingga mencapai tujuan (*output*) yang optimum (Siang, 2014). OR termasuk bagian dari aplikasi matematika untuk memecahkan masalah optimasi. Menurut Zukhri (2014), optimasi adalah proses menyelesaikan masalah supaya memperoleh atau mendapatkan kondisi yang paling menguntungkan sesuai tujuan yang ingin dicapai. Pengertian menguntungkan disini yaitu melakukan pencarian nilai maksimum atau nilai minimum bergantung dengan tujuan yang ingin dicapai. Pada dasarnya masalah optimasi adalah masalah untuk mengambil keputusan, memilih yang terbaik dari berbagai pilihan berdasarkan kriteria tertentu. Kriteria secara umumnya bertujuan untuk memaksimumkan atau meminimumkan. Persoalan *knapsack problem*, khususnya *integer knapsack problem* dapat diselesaikan menggunakan menggunakan berbagai cara. Beberapa cara untuk menyelesaikan masalah *integer knapsack* diantaranya ada algoritma *greedy*, *dynamic programming*, *brute force*, dan *genetic*. Algoritma tersebut sama-sama dapat menyelesaikan permasalahan *integer knapsack* dan menghasilkan solusi optimum.

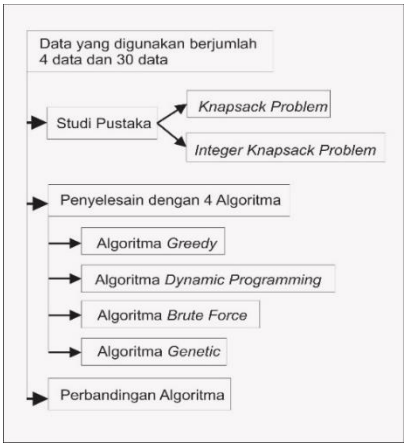
http://dx.doi.org/10.35671/telematika.v12i2.841

Algoritma *greedy* merupakan salah satu metode dari sekian banyak metode algoritma yang dapat digunakan untuk menyelesaikan permasalahan *integer knapsack*. Contoh metode algoritma lain menurut Pan & Zhang (2018) yang dapat digunakan untuk menyelesaikan permasalahan *knapsack* yaitu dengan algoritma *dynamic programming*, algoritma *brute force* dan algoritma *genetic*. Berdasarkan latar belakang tersebut peneliti tertarik untuk membahas penyelesaian persoalan *integer knapsack* dengan algoritma *greedy, dynamic programming, brute force* dan *genetic*. Intinya, untuk mengetahui algoritma yang terbaik maka dilakukan penelitian antara algoritma *greedy, dynamic programming, brute force* dan *genetic* untuk menyelesaikan permasalahan *integer knapsack.* Hasil akhir dari penelitian ini diharapkan dapat mengetahui hasil perbandingan keempat algoritma dalam hal waktu dan hasil optimumnya.

Batasan masalah untuk penelitian ini yaitu nilai merupakan biaya ongkos kirim jasa pengiriman barang, penyelesaian menggunakan metode algoritma *greedy* (konsep *greedy by profit, greedy by weight, greedy by density*), algoritma *dynamic programming,* algoritma *brute force*, dan algoritma *genetic.* Selanjutnya rumusan masalah pada penelitian ini yaitu menyelesaikan *integer knapsack problem* menggunakan algoritma *greedy, dynamic programming, brute force*, dan *genetic.* Selanjutnya, implementasi keempat algoritma dalam *integer knapsack problem* menggunakan *software* MATLAB *v2017a* berbasis *GUI*. Langkah terakhir yaitu menganalisis perbandingan keempat algoritma yang memberikan solusi optimal untuk permasalahan *integer knapsack problem* (berdasarkan hasil maksimal maupun waktu komputasi (detik) minimum yang dibutuhkan). Berdasarkan rumusan masalah tersebut maka tujuan penelitiannya adalah untuk mengetahui penyelesaian *integer knapsack problem* menggunakan keempat algoritma dengan mengimplementasikannya menggunakan *software* MATLAB *v2017a*, kemudian dibandingkan hasilnya setiap algoritma penyelesaian. Manfaat utama dari penelitian ini adalah (1) Mengenalkan *integer knapsack problem* dan penyelesaiannya, (2) Mengimplementasikan dan membandingkan keempat algoritma dari segi hasil dan waktu yang optimal, (3) Mendiskusikan arah penelitian selanjutnya untuk topik *integer knapsack problem* dan memilih algoritma mana yang efektif dan efisien untuk digunakan.

METODE PENELITIAN

Pada penelitian ini ada beberapa tahapan untuk menyelesaikan perbandingan algoritma pada masalah *integer knapsack* yang ditunjukkan pada Gambar 1 seperti berikut:



Gambar 1. Tahapan-tahapan penelitian

http://dx.doi.org/10.35671/telematika.v12i2.841

1. *Knapsack Problem*

Knapsack problem atau *rucksack problem* secara bahasa adalah masalah tempat/ ransel yang diartikan lebih lanjut yaitu masalah pengepakan. Masalah tersebut, menurut Messac (2015) merupakan masalah optimasi kombinatorial dimana harus memilih dan mencari solusi yang terbaik dari berbagai banyak solusi pilihan yang ada. Selanjutnya, menurut Juwita, Susanto, dan Halomoan (2017) tentang *knapsack problem* merupakan suatu permasalahan pemilihan barang yang akan disimpan atau dimasukkan ke suatu tempat dengan memiliki keterbatasan kapasitas. Oleh karena itu, dapat disimpulkan bahwa setiap barang memiliki berat, dan nilai yang digunakan untuk menentukan pilihan prioritasnya. Barang-barang tersebut dimasukkan ke tempat yang dipilih dengan kapasitas maksimal yang tersedia. Tempat ini memiliki berat yang membatasi jumlah barang yang dapat dimasukan, sehingga menghasilkan hasil yang optimal dan tidak melebihi kapasitas tempatnya.

2. *Integer Knapsack Problem*

Masalah *integer knapsack* adalah salah satu masalah optimasi kombinatorial yang paling banyak dipelajari. Masalah ini bertujuan untuk memaksimalkan total nilai barang yang akan dibawa/ diangkut ke tempat yang diinginkan, dan yang dimaksud kendala adalah memastikan jumlah berat kurang dari atau sama dengan kapasitas tempatnya. Keputusan pada masalah *integer knapsack* hanya ada dua pilihan untuk setiap barang, yaitu dapat dimasukkan ke dalam tempat yang diinginkan atau tidak. Setiap barang tidak dapat dimasukkan ke tempat yang diinginkan lebih dari sekali atau sebagian barang dimasukkan ke tempat yang diinginkan. Masalah *integer knapsack* dalam kehidupan sehari-hari dapat diaplikasikan di bidang enkripsi informasi, pengambilan keputusan dalam proyek-proyek teknik dan pemuatan atau pengepakan kargo. BerdasarkanModel masalah *integer knapsack* dapat dirumuskan sebagai berikut (Lin et al., 2017):

Fungsi tujuan optimal :

$$Z = \sum_{i=1}^{n} v_i y_i \qquad (1)$$

Fungsi kendala :

$$z = \sum_{i=1}^{n} w_i y_i \leq W \qquad (2)$$

$\forall y_i \in \{0,1), 1 \leq i \leq n$

Keterangan:

Z = Nilai optimum dari fungsi tujuan

z = Kendala dari fungsi tujuan

n = Jumlah objek

v_i = Nilai objek $\forall i \in \{1,2,...,n\}$

w_i = Berat objek $\forall i \in \{1,2,...,n\}$

W = Kapasitas *Knapsack*

y_i = Menunjukkan barang dimasukkan atau tidak

http://dx.doi.org/10.35671/telematika.v12i2.841

3. Penyelesaian menggunakan algoritma

a. Algoritma *greedy*

Greedy secara bahasa memiliki arti tamak atau rakus. Selanjutnya, algoritma *greedy* merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Persoalan optimasi terdiri dari dua macam yaitu maksimasi dan minimasi. Algoritma *greedy* mendapatkan solusi setiap langkah demi langkah. Pada setiap langkah, terdapat banyak pilihan yang perlu dieksplorasi. Oleh karena itu pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Pada setiap langkahnya merupakan pilihan, untuk membuat langkah optimum lokal dengan tujuan bahwa langkah selanjutnya mengarah ke solusi optimum global. Prinsip dari algoritma *greedy* menurut Ghozali, Setiawan, & Furqon (2017) adalah mengambil sebanyak mungkin apa yang dapat diperoleh sekarang. Algoritma *greedy* merupakan metode yang sering digunakan untuk menyelesaikan *integer knapsack problem. Integer knapsack problem* diselesaikan dengan algoritma *greedy* mempunyai kompleksitas waktu $O(n \log n)$ (Pan & Zhang, 2018). Metode algoritma ini tidak selalu menyelesaikan dengan hasil yang optimal, tetapi dapat menghasilkan solusi optimal lokal yang mendekati solusi optimal global dengan waktu yang cepat. Pemilihan barang yang akan dimasukkan ke dalam *knapsack* terdapat beberapa strategi dari metode algoritma *greedy* (Juwita et al., 2017) adalah:

1) *Greedy by Profit*

Setiap langkah di *knapsack problem* diisi dengan barang yang mempunyai keuntungan terbesar. Strategi ini bertujuan untuk memaksimalkan keuntungan dengan memilih barang yang paling menguntungkan terlebih dahulu. Tahap awalnya adalah mengurutkan secara menurun barang-barang berdasarkan nilai barang. Kemudian barang-barang diambil satu persatu sampai kapasitas tempatnya penuh atau sudah tidak ada yang dapat dimasukkan lagi.

2) *Greedy by Weight*

Setiap langkah di *knapsack problem* diisi dengan barang yang mempunyai berat paling ringan. Strategi ini bertujuan untuk memaksimalkan keuntungan dengan memasukkan barang sebanyak mungkin. Tahap awalnya adalah mengurutkan secara menurun barang-barang berdasarkan berat barang yang paling ringan. Kemudian barang-barang diambil satu persatu sampai kapasitas tempatnya penuh atau sudah tidak ada yang dapat dimasukkan lagi.

3) *Greedy by Density*

Setiap langkah di *knapsack problem* diisi dengan barang yang mempunyai $\frac{p_i}{w_i}$ dimana *p* adalah nilai, *w* adalah berat dan $i = (1, 2, 3, ..., n)$. Strategi ini bertujuan untuk memaksimalkan keuntungan dengan memilih barang yang mempunyai $\frac{p_i}{w_i}$ (*density*) terbesar. Tahap awalnya adalah mencari dan mengurutkan secara menurun barang-barang berdasarkan $\frac{p_i}{w_i}$ barang. Kemudian barang-barang diambil satu persatu sampai kapasitas tempatnya penuh atau sudah tidak ada yang dapat dimasukkan lagi (Kellerer, Pferschy, & Pisinger, 2004)

http://dx.doi.org/10.35671/telematika.v12i2.841

b. Algoritma *dynamic programming*

Metode *dynamic programming* adalah pendekatan umum yang muncul sebagai alat yang berguna di banyak bidang *Research Operation* (RO). Pada dasarnya, metode ini diterapkan pada masalah optimasi yang melibatkan urutan keputusan, solusi optimal dari masalah yang asli dapat ditemukan dari solusi optimal *subproblem* (Kellerer, Pferschy, & Pisinger, 2004). Definisi lain, *dynamic programming* adalah metode pemecahan dengan menguraikan solusi menjadi serangkaian langkah atau langkah-langkah sehingga solusi dari masalah dapat dilihat dari serangkaian keputusan yang saling berhubungan (Sampurno, Sugiharti, dan Alamsyah, 2018). Jadi dapat disimpulkan bahwa *dynamic programming* adalah metode pemecahan masalah dengan menguraikan solusi menjadi beberapa tahapan atau langkah-langkah sehingga solusi optimalnya dapat ditemukan dari rangkaian keputusan yang saling berkaitan. Algoritma *dynamic programming* dalam menyelesaikan masalah *integer knapsack* berdasarkan Pan & Zhang (2018) mempunyai kompleksitas waktu $O(nW)$, dimana *W* adalah koefisien dari kapasitasnya.

Pada penelitian ini, algoritma *dynamic programming* menggunakan teknik *bottom-up*. Ada tiga elemen dasar dalam teknik *bottom-up* (Kwarteng & Asante, 2017), yaitu:

a) *Substructure*

Mengurai masalah yang ada menjadi masalah yang lebih kecil dan mulai dengan permasalahan yang paling kecil.

b) Struktur tabel

Simpan jawaban (hasil) ke dalam tabel.

c) Perhitungan *bottom-up*

Prinsipnya adalah menggunakan tabel untuk menggabungkan solusi dari masalah lebih kecil yang didapatkan, kemudian untuk menyelesaikan masalah yang lebih besar, dan diproses sampai akhir hingga mendapat solusi optimum untuk menyelesaikan masalah. Gambaran solusi perhitungan *bottom-up* algoritma *dynamic programming* sebagai berikut: Masukan *n* yaitu jumlah barang, *W* yaitu kapasitas maksimum, $v = (v_1, v_2, ..., v_n)$, dan $w = (w_1, w_2, ..., w_n)$. Algoritma ini akan mengisi setiap *cell M*(*n*,*W*) mengikuti persamaan di bawah ini (Escobar, Kolar, Harb, Vinci Dos Santos, dan Valderrama, 2017):

$$M(i,j) = \begin{cases} 0, & \text{Jika } j = 0 \quad (3) \\ M(i-1,j), & \text{Jika } j < w_i \quad (4) \\ \max(M(i-1,j), M(i-1,j-w_i) + p_i), & \text{Jika} \geq w_i \quad (5) \end{cases}$$

c. *Algoritma brute force*

Brute force menurut Levitin (2012) adalah pendekatan langsung (*staraightforward)* untuk menyelesaikan masalah, biasanya langsung berdasarkan pada pernyataan masalah dan definisi dari konsep yang terlibat. Definisi lain, *Brute force* (*exhaustive search*) menurut Messac (2015) adalah pendekatan langsung (*straightforward*) yang dapat digunakan untuk memecahkan masalah diskrit skala sedikit dan menyebutkan semua kandidat yang layak atau yang diambil. Solusi terbaik adalah solusi yang optimal. Berdasarkan penelitian Pan & Zhang (2018) menyebutkan waktu yang dibutuhkan untuk menyelesaikan *integer knapsack problem* menggunakan algoritma *brute force* membutuhkan waktu yang lebih lama bahkan sangat lama sehingga terkadang menyebabkan kasus

http://dx.doi.org/10.35671/telematika.v12i2.841

time limit exceeded pada beberapa program yang ada batasan waktu kompilasi dan *runtime* program dengan kompleksitas waktunya $O(2^n)$. Menurut Levitin (2012) bahwa strategi algoritma *brute force* seringkali paling mudah untuk diterapkan dan algoritma *brute force* dapat diartikan sebagai algoritma *trial* and *error* untuk mendapatkan solusi optimalnya.

Brute force mencoba semua kemungkinan kombinasi variabel diskrit untuk menemukan hasil yang optimal (Parkinson, Balling, dan Hedengren, 2013). Prinsip – prinsip algoritma *brute force* untuk menyelesaikan permasalahan *integer knapsack* adalah:

1) Mengenumerasikan semua himpunan bagian dari solusi.
2) Mengevaluasi total keuntungan dari setiap himpunan bagian dari langkah pertama.
3) Pilih himpunan bagian yang mempunyai total keuntungan terbesar.

d. Algoritma *genetic*

Algoritma *genetic* menurut Messac (2015) merupakan algoritma komputasi yang terinspirasi oleh prinsip evolusi alami yang dijelaskan dalam teori Darwin. Selanjutnya, algoritma *genetic* menurut Syarif (2014) adalah metode yang meniru mekanisme proses evolusi dengan mengikuti prinsip seleksi alam yang dikembangkan oleh darwin. Jadi algoritma *genetic* adalah algoritma dengan mengikuti prinsip seleksi alam seperti proses evolusi yang dikembangkan oleh darwin.

Algoritma *genetic* sudah banyak diaplikasikan oleh para peneliti untuk menyelesaikan permasalahan di dunia nyata. Penelitian yang memanfaatkan algoritma *genetic* untuk mendapatkan solusi ada beberapa penelitian berikut ini, yaitu pada masalah *problem logistic*, *Vehicle Routing Problem* (VRP), *Knapsack Problem* (KP), dan masih banyak lagi (Syarif, 2014). Algoritma *genetic* berdasarkan Pan & Zhang (2018) dalam penyelesaiannya tidak stabil dan tidak dapat menjamin solusi optimalnya dan kompleksitas waktu yang dihasilkan dalam waktu *polinomial* yaitu $O(n^2)$.

Penyelesaian *integer knapsack problem* menggunakan algoritma *genetic* menurut Fanggidae dan Lado (2015) adalah sebagai berikut:

a. Membangkitkan populasi awal secara acak
b. Evaluasi Kromosom
c. *Crossover* (kawin silang)
d. Mutasi
e. Seleksi kromosom
f. *Decoding*

HASIL DAN PEMBAHASAN

Data yang digunakan untuk percobaan berjumlah 4 data dan 30 data:

1) 4 data

Berat (kg) = {1, 8, 2, 4}, dan nilai (ribuan) = {30, 152, 26, 108} dan kapasitasnya sebesar 11kg

2) 30 data

Berat (kg) = {1, 1, 1, 1, 1, 1, 1, 1, 8, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 1}, dan nilai (ribuan) = {30, 11, 11, 92, 33, 21, 15, 13, 152, 26, 13, 14, 21, 13, 13, 62, 13, 13, 27, 11, 13, 11, 9, 13, 108, 25, 25, 11, 11, 18} dan kapasitasnya sebesar 32kg.

http://dx.doi.org/10.35671/telematika.v12i2.841

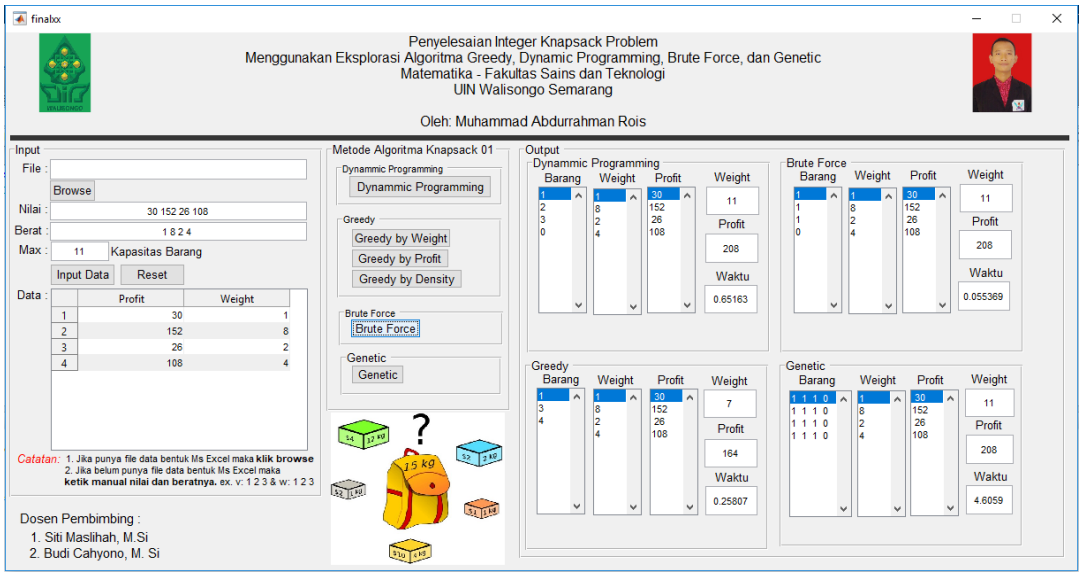
Peneliti menggunakan algoritma *greedy*, *dynamic programming*, *brute force*, dan *genetic* untuk menyelesaikan dan dibandingkan. Hasilnya sebagai berikut:

1) 4 data

Hasil penyelesaian 4 data didapatkan barang yang diambil dengan total berat dan total nilai, selanjutnya waktu komputasi yang diperlukan dalam penyelesaian di tampilkan pada Tabel 1 dan tampilan penyelesaian ditampilkan pada Gambar 2.

Tabel 1. Hasil penyelesaian 4 data dengan keempat algoritma

Metode	Nomor barang yang diambil	Total berat (kg)	Total nilai (Rp dalam ribuan)	Waktu komputasi (detik)
1. Algoritma *greedy*				
a. *Greedy by profit*	2	8	152.000	0,32447
b. *Greedy by weight*	1, 3, 4	7	164.000	0,25807
c. *Greedy by density*	1, 4	5	138.000	0,28201
2. Algoritma *dynamic programming*	1, 2, 3	11	208.000	0,65163
3. Algoritma *brute force*	1, 2, 3	11	208.000	0,055369
4. Algoritma *genetic*	1, 2, 3	11	208.000	4,6059



Gambar 2. Tampilan penyelesaian menggunakan *software* MATLAB *v2017a* berbasis *GUI*

Berdasarkan hasil penyelesaian bagian 4 data yang ditampilkan pada Tabel 1 dan Gambar 2 menunjukkan hasil yang paling optimal adalah menggunakan algoritma *dynamic programming*, *brute force*, dan *genetic*. Waktu yang paling efisien dalam penyelesaian 4 data menggunakan algoritma *brute force.*

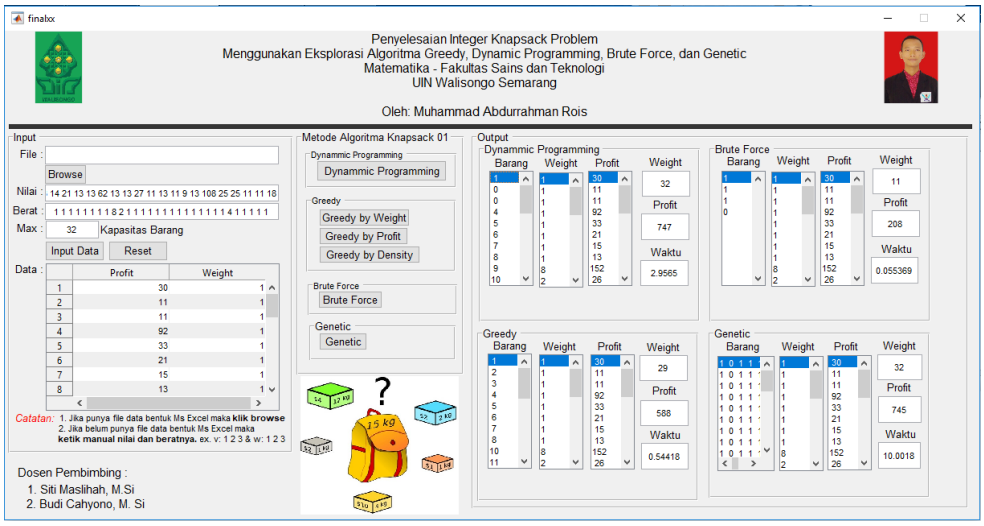
2) 30 data

http://dx.doi.org/10.35671/telematika.v12i2.841

Hasil penyelesaian 4 data didapatkan barang yang diambil dengan total berat dan total nilai, selanjutnya waktu komputasi yang diperlukan dalam penyelesaian di tampilkan pada Tabel 2 dan tampilan penyelesaian ditampilkan pada Gambar 3.

Tabel 2. Hasil penyelesaian 30 data dengan keempat algoritma

Metode	Nomor barang yang diambil	Total berat (kg)	Total nilai (Rp dalam ribuan)	Waktu komputasi (detik)
1. Algoritma *greedy*				
a. *Greedy by profit*	1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 25, 26, 27, 30	32	747.000	0,74778
b. *Greedy by weight*	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30	29	588.000	0,54418
c. *Greedy by density*	1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 25, 26, 27, 30	32	747.000	0,59587
2. Algoritma *dynamic programming*	1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 25, 26, 27, 30	32	747.000	2,9565
3. Algoritma *brute force*	1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 25, 26, 27, 30	32	747.000	6.746.795,19
4. Algoritma *genetic*	1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 16, 17, 19, 21, 24, 25, 26, 27, 28, 30	32	742.000	10,0018



Gambar 3. Tampilan penyelesaian menggunakan *software* MATLAB *v2017a* berbasis *GUI*

http://dx.doi.org/10.35671/telematika.v12i2.841

Berdasarkan hasil penyelesaian bagian 30 data yang ditampilkan pada Tabel 2 dan Gambar 3. menunjukkan hasil yang paling optimal adalah menggunakan algoritma *dynamic programming*, *brute force*, dan *genetic.* Algoritma *brute force* untuk 30 data dengan batasan waktu 100.000 detik tidak dapat diketahui karena jumlah waktu yang sangat besar tetapi untuk hasil pasti akan sama dengan hasil dari algoritma *dynamic programming*. Jadi penyelesaian waktu untuk 30 data yang belum diperoleh maka dibutuhkan peramalan untuk prediksi waktu komputasi yang diperlukan menggunakan metode peramalan *trend eksponensial*, didapatkan waktu komputasinya sebesar 6.746.795,19 detik dan algoritma *brute force* tidak efisien untuk digunakan data skala besar. Waktu yang paling efisien dalam penyelesaian 30 data menggunakan algoritma *greedy by profit*, dan *greedy by density.*

Hasil penyelesaian menggunakan data sebanyak 4 dan 30 digunakan untuk tujuan menentukan waktu komputasi yang stabil dan efisien untuk menyelesaikan masalah *integer knapsack* skala kecil ataupun besar. Berdasarkan hasil penyelesaian menggunakan empat algoritma diperoleh kesimpulan bahwa penyelesaian *integer knapsack problem* yang efektif dan efisien ditinjau dari hasil dan waktu secara umum.

KESIMPULAN DAN SARAN

1. Kesimpulan

Hasil penyelesaian *integer knapsack problem* menggunakan algoritma *greedy*, *dynamic programming*, *brute force* dan *genetic* dengan 4 data menunjukkan bahwa algoritma *dynamic programming*, *brute force* dan *genetic* menghasilkan solusi optimal. Berbeda dengan 30 data menunjukkan bahwa semua algoritma menghasilkan solusi optimal. Algoritma *greedy* dengan memiliki waktu komputasi yang baik tetapi solusi yang dihasilkan tidak selalu optimal. Algoritma *brute force* dengan data sedikit memiliki waktu komputasi yang sangat baik tetapi untuk data besar memiliki waktu yang sangat banyak, karena kompleksitas waktunya yang bersifat eksponensial. Algoritma *genetic* tidak stabil dan tidak dapat menjamin solusi yang optimal, karena dipengaruhi oleh inisialisasi awal yang acak. Jadi algoritma *dynamic programming* merupakan pilihan algoritma terbaik yang efektif dan efisien stabil dalam penyelesaian *integer knapsack problem* di skala data kecil ataupun besar.

2. Saran

Penelitian ini dari segi algoritma untuk membuat program pasti tidak 100% baik dan efektif maka dari itu penelitian selanjutnya perlu diperbaiki dan dikembangkan algoritma tersebut supaya lebih baik dan efektif untuk menyelesaikan *integer knapsack problem*. Selanjutnya, juga bisa dikembangkan dengan menggunakan algoritma lain yang dapat digunakan untuk menyelesaikan *integer knapsack problem.* Peneliti selanjutnya bisa menggunakan permasalahan yang berbeda atau sudah dikembangkan seperti *unbounded knapsack problem* guna menyesuaikan permasalahan pada realita yang ada.

DAFTAR PUSTAKA

Escobar, F. A., Kolar, A., Harb, N., Vinci Dos Santos, F., & Valderrama, C. (2017). Scalable shared-memory architecture to solve the Knapsack 0/1 problem. *Microprocessors and Microsystems*,

50, 189–201. https://doi.org/10.1016/j.micpro.2017.04.001

Fanggidae, A., & Lado, F. R. (2015). *Algoritma Genetika dan Penerapannya*. TEKNOSAIN.

Ghozali, A. E., Setiawan, B. D., & Furqon, M. T. (2017). Aplikasi Perencanaan Wisata di Malang Raya dengan Algoritma Greedy, *1*(12), 1459–1467.

Juwita, P. S., Susanto, E., & Halomoan, J. (2017). Perancangan Dan Implementasi Manajemen Daya Listrik Menggunakan Algoritma Greedy Untuk Otomatisasi Rumah. In *e-Proceeding of Engineering* (Vol. 4, pp. 1512–1519).

Kellerer, Hans, Pferschy, U., & Pisinger, D. (2004). *Knapsack Problems*. *Knapsack Problems*. Springer-Verlag Berlin Heidelberg New York. https://doi.org/10.1007/978-3-540-24777-7_9

Kwarteng, A., & Asante, B. (2017). Optimal Advertisement Placement Slot Using Knapsack Problem (A Case Study of Television Advertisement of Tv 3 Ghana). *International Journal of Engineering Research and Applications*, *07*(04), 46–62. https://doi.org/10.9790/9622-0704044662

Levitin, A. (2012). *Introduction to The Design & Analysis of Algorithms*. *Pearson* (3rd ed.). PEARSON.

Lin, B., Liu, S., Lin, R., Wu, J., Wang, J., & Liu, C. (2017). Modelling the 0-1 Knapsack Problem in Cargo Flow Adjustment. *Symmetry*, *9*(7), 118. https://doi.org/10.3390/sym9070118

Messac, A. (2015). *Optimization in Practice with MATLAB*. *Book*. Cambridge University Press.

Nuraeni. (2016). Jasa Logistik Melesat di Era e-Commerce. *Kominfo*. Retrieved from kominfo.go.id.index.php/content/detail/6707/Jasa+Logistik+Melesat+di+Era+e-Commerce+/0/sorotan_media

Pan, X., & Zhang, T. (2018). Comparison and Analysis of Algorithms for the 0/1 Knapsack Problem. *IOP Journal of Physics*, 1–8. https://doi.org/10.1088/1742-6596/1069/1/012024

Parkinson, A. R., Balling, R., & Hedengren, J. D. (2013). Optimization Methods for Engineering Design. *Brigham Young University*, 18. https://doi.org/10.1002/9780470686652.eae495

Sampurno, G. I., Sugiharti, E., & Alamsyah. (2018). Comparison of Dynamic Programming Algorithm and Greedy Algorithm on Integer Knapsack Problem in Freight Transportation. *Scientific Journal of Informatics*, *5*(1). Retrieved from http://journal.unnes.ac.id/nju/index.php/sji

Siang, J. J. (2014). *Riset Operasi Dalam Pendekatan Algoritmis* (Edisi 2). ANDI Yogyakarta.

Syarif, A. (2014). *Algoritma Genetika; Teori dan Aplikasi* (Edisi 2). Yogyakarta: Graha Ilmu.

Tarliah, D. T., & Dimyati, A. (2016). *Operations Research Model-Model Pengambilan Keputusan*. Bandung: Penerbit Sinar Baru Algensindo.

Zukhri, Z. (2014). *ALGORITMA GENETIKA: Metode Komputasi Evolusioner untuk Menyelesaikan Masalah Optimasi*. Yogyakarta: C.V. ANDI OFFSET.