ISSN : 1978-6603

JURNAL Ilmiah SAINTIKOM Sains dan Komputer

# IMPLEMENTASI ALGORITMA GREEDY UNTUK MENYELESAIKAN MASALAH KNAPSACK PROBLEM

**Dian Rachmawati [#1], Ade Candra [#2]**
[#1,2] Program Studi Ilmu Komputer, Universitas Sumatera Utara
Jl. Alumni no.9 Medan
Email : dee230783@gmail.com[#1],

**Abstrak**

Optimisasi merupakan metode pemecahan masalah maksimasi ataupun minimalisasi. Optimisasi sangat bermanfaat untuk meningkatkan performa dan produktifitas kinerja. Transportasi merupakan salah satu bidang yang sangat erat kaitannya dengan optimisasi. Masalah pengangkutan barang dengan kapasitas alat pengangkut yang terbatas dan keinginan untuk memperoleh keuntungan yang maksimum adalah masalah klasik di dunia transportasi. Masalah ini sering juga dianalogikan dengan menggunakan teori Knapsack Problem. Algoritma Greedy yang diimplementasikan ke dalam perangkat lunak dapat digunakan untuk menyelesaikan Knapsack Problem pada dunia transportasi dengan waktu yang lebih singkat dibandingkan dengan menggunakan perhitungan manual dan algoritma Brute Force.

***Kata Kunci** : Greedy, Optimisasi, Algoritma, Knapsack Problem.*

***Abstract***

Optimization is a method that can solve maximization or minimization problem. Optimization is very useful for improving the performance and productivity. Transportation is one of the example area that is very closely related to optimization. Transporting of goods conveyance which has limited capacity and the desire to obtain the maximum profit is a classic problem in the world of transportation. This problem is often also analogous to the Knapsack Problem. Greedy algorithms which implemented in software can be used to solve the Knapsack Problem on the world of transportation. Using the software can reduce time a compared to using manual calculations and Brute Force algorithm.

***Keywords** : Greedy, Optimization, Algorithm, Knapsack Problem, Brute Force.*

## PENDAHULUAN

Persoalan transportasi merupakan kasus khusus dari optimisasi. Ada banyak masalah di bidang transportasi yang terkait dengan optimisasi. Salah satu permasalahan di bidang transportasi yang muncul adalah bagaimana suatu perusahaan mengatur produk apa yang harus diangkut agar memperoleh keuntungan yang maksimal, sementara perusahaan sendiri memiliki problematika / kendala yaitu kapasitas angkut dari kendaraan yang sangat terbatas.

Persoalan optimisasi transportasi di atas sering dianalogikan sebagai *Knapsack* Problem. *Knapsack* memiliki arti karung/ kantung. Karung mempunyai kapasitas muat yang sangat terbatas. Berhubung kapasitas muat terbatas maka barang-barang yang dapat dimasukkan ke dalam karung hanya sampai batas kapasitas muatan maksimum karung saja. Sehingga jika dihubungkan dengan transportasi maka karung akan mewakili alat transportasi sebagai media pengangkut. Banyak tahapan yang dapat dilakukan untuk menyelesaikan masalah tersebut salah satunya adalah algoritma Brute Force. Akan tetapi algoritma ini tidak terbukti dapat menyelesaikan permasalahan ini dengan efisien. Sehingga untuk menyelesaikan permasalahan ini akan digunakan algoritma Greedy dengan meninjau dari 3 sisi yaitu : Greedy by Weight, Greedy by Profit dan Greedy by Density.

## TINJAUAN PUSTAKA

### 1. Knapsack Problem

Knapsack adalah tas atau karung. Karung digunakan untuk memuat sesuatu. Dan tentunya tidak semua objek dapat ditampung di dalam karung tersebut. Karung tersebut hanya dapat menyimpan beberapa objek dengan total ukurannya (weight) lebih kecil atau sama dengan ukuran kapasitas karung. Ilustrasi permasalahan dapat dilihat pada gambar 1. Pada gambar 1 terlihat terdapat sebuah tas berkapasitas 15 kg. Dan terdapat 5 barang dengan berat dan keuntungannya masing – masing. Yang menjadi persoalan adalah barang mana saja yang harus dimasukan ke dalam tas.


Gambar 1. Ilustrasi Permasalahan Knapsack Problem

Knapsack Problem secara matematis dapat ditulis sebagai berikut :
Diberikan bobot knapsack adalah M. Diketahui n buah objek yang masing-masing bobotnya adalah $w_1$, $w_2$, …, $w_n$. Tentukan nilai bi sedemikian sehingga

$$M = b_1w_1 + b_2w_2 + \ldots + b_nw_n$$

yang dalam hal ini, bi bernilai 0 atau 1. Jika $b_i$ = 1, berarti objek i dimasukkan ke dalam *knapsack*, sebaliknya jika $b_i$ = 0, objek i tidak dimasukkan. Berhubung nilai $b_i$ 0 dan 1 maka masalah ini sering juga disebut sebagai *Knapsack* 0/1.

Dalam teori algoritma, persoalan knapsack termasuk ke dalam kelompok NP-complete. Persoalan yang termasuk NPcomplete tidak dapat dipecahkan dalam orde waktu polinomial.

### 2. Algoritma Brute Force

Brute Force adalah sebuah pendekatan langsung (straight forward) untuk memecahkan suatu masalah, yang biasanya didasarkan pada pernyataan masalah (problem statement) dan definisi konsep yang dilibatkan. Pada dasarnya algoritma Brute Force adalah alur penyelesaian suatu permasalahan dengan cara berpikir yang sederhana dan tidak membutuhkan suatu permikiran yang lama.

Sebenarnya, algoritma Brute Force merupakan algoritma yang muncul karena pada dasarnya alur pikir manusia adalah Brute Force (langsung/to the point).

Beberapa karakteristik dari algoritma Brute Force dapat dijelaskan sebagai berikut.

a. Membutuhkan jumlah langkah yang banyak dalam menyelesaikan suatu permasalahan sehingga jika diterapkan menjadi suatu algoritma program aplikasi akan membutuhkan banyak memori.
b. Digunakan sebagai dasar dalam menemukan suatu solusi yang lebih efektif.
c. Banyak dipilih dalam penyelesaian sebuah permasalahan yang sederhana karena kemudahannya.
d. Pada banyak kasus, algoritma ini banyak dipilih karena hampir dapat dipastikan dapat menyelesaikan banyak persoalan yang ada.
e. Digunakan sebagai dasar bagi perbandingan keefektifan sebuah algoritma.

Dalam beberapa kasus tertentu algoritma Brute Force hampir sama dengan Exhaustive Search. Exhausitve Search yang merupakan teknik pencarian solusi secara Brute Force pada masalah yang melibatkan pencarian elemen dengan sifat khusus. Biasanya elemen tersebut berada di antara objek-objek kombinatorik seperti permutasi, kombinasi, atau himpunan bagian dari sebuah himpunan. Berdasarkan definisi ini, maka dapat ditarik kesimpulan bahwa Exhaustive Search adalah Brute Force juga. Oleh karena itu Exhaustive Search adalah salah satu implementasi dari Brute Force dalam kasus pencarian. Masalah–masalah dalam *Exhaustive Search* dengan penerapan algoritma Brute Force dapat dijelaskan sebagai berikut.

a. Enumerasi setiap solusi yang mungkin dengan cara yang sistematis.
b. Evaluasi setiap kemungkinan solusi yang ditemukan satu per satu, meskipun terdapat beberapa kemungkinan ditemukannya solusi yang tidak layak atau bahkan terdapat kemungkinan–kemungkinan solusi terbaik yang telah ditemukan dan dievaluasi.
c. Bila pencarian sudah sampai pada tujuan, maka pilih solusi yang terbaik.

**3. Algoritma Greedy**

Algoritma Greedy merupakan algoritma yang lazim untuk memecahkan persoalan optimasi meskipun hasilnya tidak selalu merupakan solusi yang optimum. Sesuai arti harafiah, Greedy berarti tamak. Prinsip utama dari algoritma ini adalah mengambil sebanyak mungkin apa yang dapat diperoleh sekarang. Untuk memecahkan persoalan dengan algoritma Greedy, kita memerlukan elemen-elemen sebagai berikut.

a. Himpunan Kandidat (C)

   Himpunan ini berisi elemen-elemen pembentuk solusi.
b. Himpunan Solusi, (S)

   Himpunan ini berisi kandidat yang terpilih sebagai solusi persoalan. Dengan kata lain, himpunan solusi adalah himpunan

bagian dari himpunan kandidat.

c. Fungsi Seleksi
   Fungsi seleksi merupakan fungsi yang ada pada setiap langkah memilih kandidat yang paling memungkinkan guna mencapai solusi optimal.
d. Fungsi Kelayakan (Feasible)
   Fungsi kelayakan adalah fungsi yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak dan tidak melanggar batasan atau constraints yang ada.
e. Fungsi Objektif
   Fungsi objektif adalah fungsi yang memaksimumkan atau meminimumkan nilai solusi.

Skema umum algoritma Greedy adalah sebagai berikut:

a. Inisialisasi S dengan kosong.
b. Pilih sebuah kandidat C dengan fungsi seleksi.
c. Kurangi C dengan kandidat yang sudah dipilih dari langkah (b) di atas.
d. Periksa apakah kandidat yang dipilih tersebut bersama-sama dengan himpunan solusi membentuk solusi yang layak atau feasible (dengan fungsi kelayakan).
e. Periksa apakah himpunan solusi sudah memberikan solusi yang lengkap serta optimal (dengan fungsi objektif).

**4. Greedy dalam Knapsack Problem**

Pada penyelesaian Knapsack Problem terdapat 3 jenis algoritma Greedy yang dapat digunakan yaitu :

A. Greedy By Weight

Pada setiap langkah pilih objek yang mempunyai berat teringan. Mencoba memaksimumkan keuntungan dengan memasukkan sebanyak mungkin objek ke dalam *knapsack*.

Pertama kali yang dilakukan adalah program mengurutkan secara menaik objek-objekberdasarkan weightnya. Kemudian baru diambil satu-persatu objek yang dapat ditampung oleh *knapsack* sampai knapsack penuh atau sudah tidak ada objek lagi yang bisa dimasukkan.

B. Greedy by profit

Pada setiap langkah, pilih objek yang mempunyai keuntungan terbesar. Mencoba memaksimumkan keuntungan dengan memilih objek yang paling menguntukan terlebih dahulu.

Pertama kali yang dilakukan adalah program mengurutkan secara menurun objek-objek berdasarkan profitnya. Kemudian baru diambil satu-persatu objek yang dapat ditampung oleh *knapsack* sampai *knapsack* penuh atau sudah tidak ada objek lagi yang bisa dimasukkan.

C. Greedy By Density

Pada setiap langkah *knapsack* di isi dengan objek yang mempunyai $p_i/w_i$ terbesar, dimana p adalah keuntungan dan w adalah berat barang. Mencoba memaksimumkan keuntugan dengan memilih objek yang mempunyai density per unit berat terbesar.

Pertama kali yang dilakukan adalah program mencari nilai profit per berat tiap – tiap unit (density) dari tiap- tiap objek. Kemudian objek-objek tersebut diurutkan berdasarkan density-nya. Kemudian baru diambil satu-persatu objek yang dapat ditampung oleh *knapsack* sampai knapsack

penuh atau sudah tidak ada objek lagi yang bias dimasukkan.

**CONTOH PENERAPAN GREEDY DI DALAM TRANSPORTASI**

Pada tabel 1 terdapat sebuah alat angkut dengan dengan kapasitas 100 kg terdapat 4 buah barang dengan ukuran sebagai berikut :

| Barang | Weight | Profit | Density |
|---|---|---|---|
| 1 | 23 | 67 | 2.913043478 |
| 2 | 45 | 86 | 1.911111111 |
| 3 | 67 | 139 | 2.074626866 |
| 4 | 39 | 42 | 1.076923077 |

Tabel 1. Contoh Weight, Profit dan Density

Maka dengan menggunakan algoritma Greedy diperoleh Tabel 2 sebagai berikut :

| | GBW | GBP | GBD |
|---|---|---|---|
| | 1 | 1 | 1 |
| | 0 | 0 | 0 |
| | 0 | 1 | 1 |
| | 1 | 0 | 0 |
| **Keuntungan** | 109 | 206 | 206 |

Tabel 2. Penyelesaian contoh kasus pada tabel 1.

Keterangan :
GBW : Greedy By Weight
GBP : Greedy By Profit
GBD : Greedy By Density
0 : Barang tidak diangkut
1 : Barang diangkut

**ANALISIS DAN IMPLEMENTASI**

Berdasarkan teori dan contoh algoritma greedy dalam menyelesaikan *Knapsack* Problem maka *pseudocode* algoritma greedy adalah sebagai berikut :

```
function Knapsack(input C : himpunan_objek, K : real) → himpunan_solusi

{ Menghasilkan solusi persoalan knapsack dengan algoritma greedy yang menggunakan
strategi pemilihan objek berdasarkan profit (pi),weight(wi),density (pi/wi). Solusi
dinyatakan sebagai vektor X = x[1], x[2], …, x[n].

Asumsi: Untuk Greedy by profit seluruh objek sudah terurut berdasarkan nilai p_i yang menurun.
Untuk Greedy by weighted seluruh objek sudah terurut berdasarkan nilai w_i yang menaik.
Untuk Greedy by density seluruh objek sudah terurut berdasarkan nilai p_i/w_i yang menurun.
}

Deklarasi
  i, TotalBobot : integer
  Avalaible : boolean
  x : himpunan_solusi

Algoritma:
  for i ← 1 to n do
    x[i] ← 0    { inisialisasi setiap status pengambilan objek i dengan 0 }
  endfor

  i ← 0
  TotalBobot ← 0
  Available ← true
  while (i ≤ n) and (Available) do
   { cek objek ke-i }
   i ← i + 1
   if TotalBobot + w[i] ≤ K then
     { masukkan objek Ci ke dalam knapsack }
     x[i] ← 1
     TotalBobot ← TotalBobot + w[i]
   else
        Available ← false
```

x[i] ← 0

*{ objek Ci tidak dimasukkan ke dalam knapsack }*

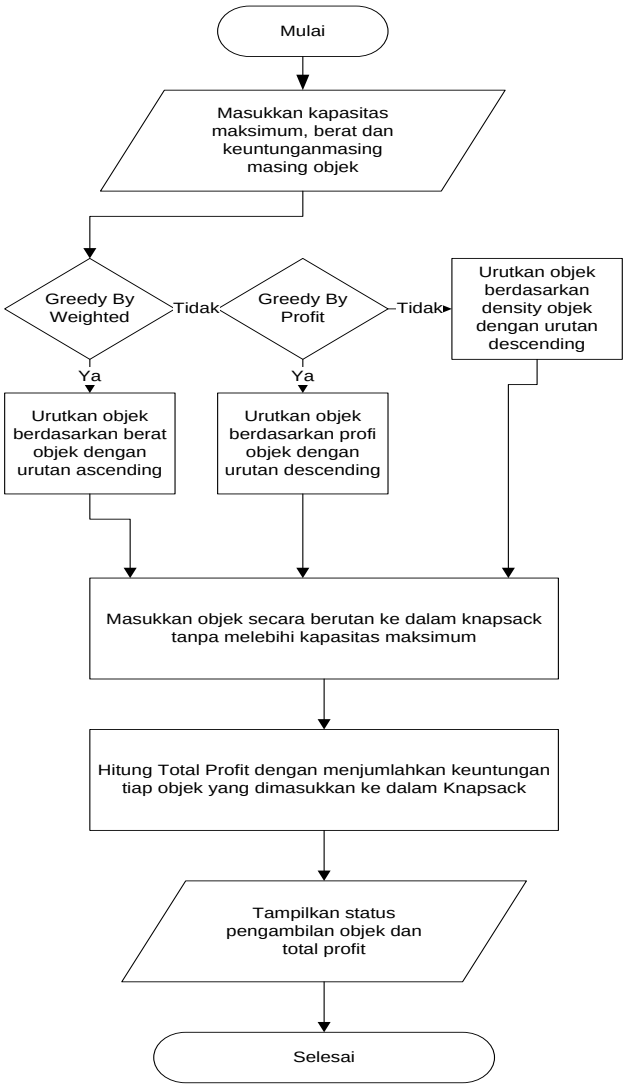
endif

endwhile

*{ i > n or not Available }*

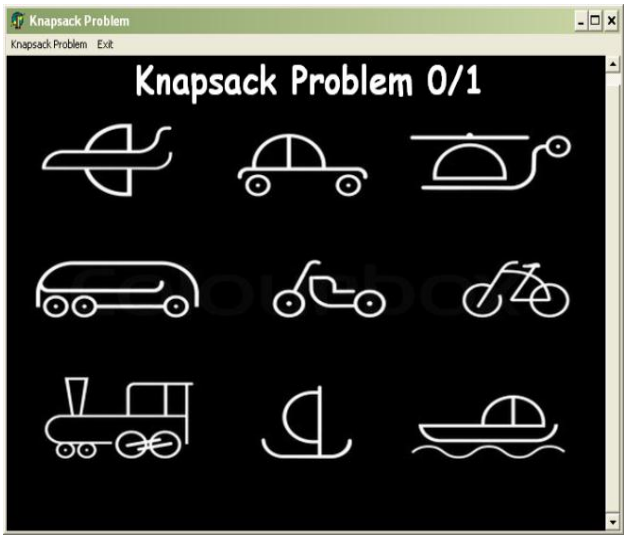
return x

Gambar 2 menggambar Flowchart sistem secara umum untuk implementasi algoritma Greedy pada Knapsack problem




Gambar 2. Flowchart Sistem implementasi algoritma Greedy untuk menyelesaikan Knapsack Problem

Gambar 3 dan gambar 4 merupakan tampilan awal aplikasi. Tampilan awal aplikasi ini berisi 2 menu utama yaitu Knapsack Problem dan Exit. *Knapsack* Problem memiliki 3 sub menu penyelesaian yaitu : Greedy by Weigh, Greedy by Profit dan Greedy by Density.

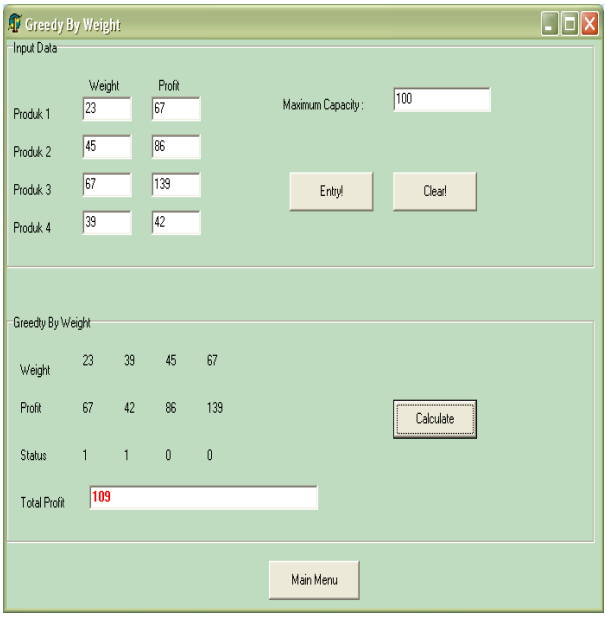



Gambar 3. Tampilan Awal Program

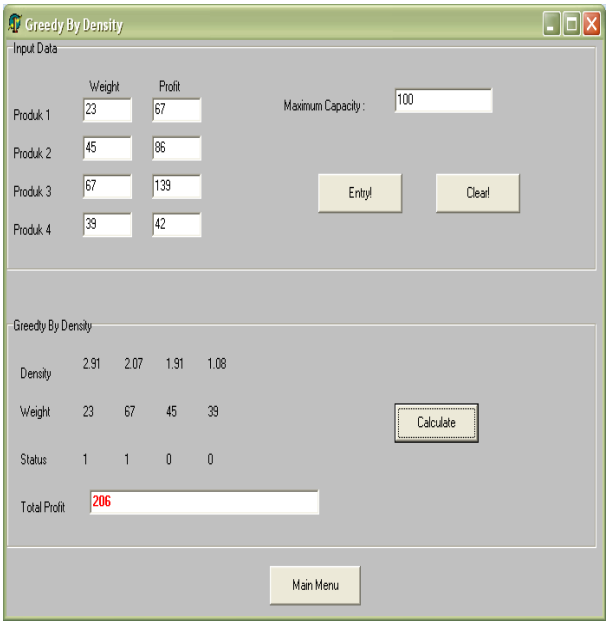

Gambar 4. Tampilan Menu dan Sub Menu

Gambar 5, 6 dan 7 menampilkan implementasi Algoritma Greedy pada *Knapsack* Problem dengan inputan berat dan keuntungan seluruh item barang / objek dan

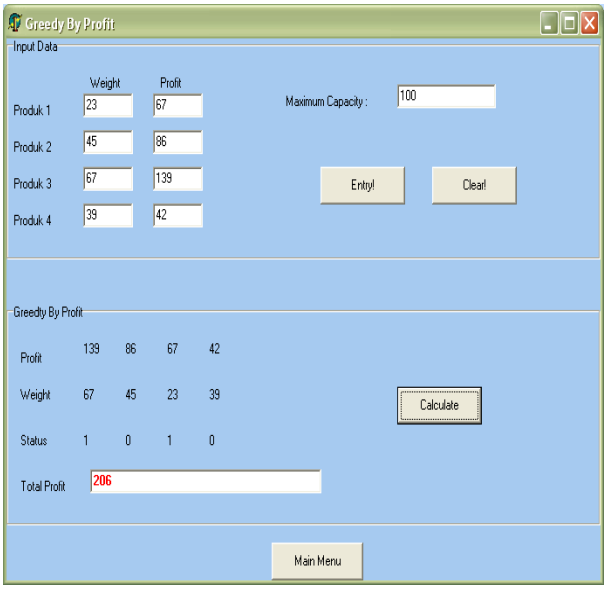
kapasitas maksimum alat angkut.




Gambar 5. Penyelesaian dengan Greedy by Weight




Gambar 7. Penyelesaian dengan Greedy by Density




Gambar 6. Penyelesaian dengan Greedy by Profit

## SIMPULAN

Dari analisis dan implementasi dapat diambil kesimpulan bahwa hasil perhitungan sistem yang mengimplementasikan algoritma greedy ke dalam Knapsack problem sama dengan hasil perhitungan manual. Performansi Greedy by Weight lebih rendah dibandingkan Greedy by Profit ataupun Density untuk memperoleh keuntungan yang maksimal.

Sistem ini cocok untuk kasus yang item barang / objek nya harus diambil keseluruhan (1) atau tidak diambil sama sekali (0) lebih sering disebut Knapsack Problem 0/1. Sistem ini tidak dapat menangani kasus dimana item barang/objek nya dapat diambil sebagian.

**DAFTAR PUSTAKA**

Cormen, Thomas H. 2001. Introduction to Algorithms,2nd Edition. Cambridge: The MIT Press.

David Pisinger. 1995. Algorithm For Knapsack Problems. Denmark : Copenhagen.

Karinda P. N. et. Al. 2013. Menyelesaikan Persoalan Optimisasi Terpadu untuk Menyelesaikan Persoalan Transportasi dan Inventori. FMIPA UNRI Pekan Baru.

Paryati. 2009. Optimasi Strategi Algoritma Greedy untuk Menyelesaikan Permasalahan Knapsack 0 - 1, Seminar Nasional Informatika 2009 (semnasIF 2009) ISSN: 1979-2328

Pinandito, Aryo. Design and Analysis of Greedy Algorithm. PTIIK Universitas Brawijaya.

Rinaldi Munir. 2005. Strategi Algoritmik, Sekolah Teknik Informatika dan Elektro, Institut Teknologi Bandung. Yogyakarta : UPN ”Veteran”.