# PENYELESAIAN *INTEGER KNAPSACK PROBLEM* MENGGUNAKAN EKSPLORASI ALGORITMA *GREEDY*, *DYNAMIC PROGRAMMING*, *BRUTE FORCE* DAN *GENETIC*

SKRIPSI
Diajukan untuk Memenuhi Sebagian Syarat
Guna Memperoleh Gelar Sarjana Sains
Dalam Ilmu Matematika

Oleh
**MUHAMMAD ABDURRAHMAN ROIS**
NIM: 1508046022

**FAKULTAS SAINS DAN TEKNOLOGI**
**UNIVERSITAS ISLAM NEGERI WALISONGO**
**SEMARANG**
**2019**

## PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama : Muhammad Abdurrahman Rois

NIM : 1508046022

Program Studi : Matematika

Fakultas : Fakultas Sains dan Teknologi

Menyatakan bahwa skripsi yang berjudul:

**PENYELESAIAN *INTEGER KNAPSACK PROBLEM* MENGGUNAKAN EKSPLORASI ALGORITMA *GREEDY*, *DYNAMIC PROGRAMMING*, *BRUTE FORCE* DAN *GENETIC***

Secara keseluruhan adalah hasil penelitian/ karya saya sendiri, kecuali bagian tertentu yang dirujuk sumbernya.

Semarang, 9 Mei 2019

Pembuat Pernyataan,


**Muhammad Abdurrahman Rois**

NIM: 150 804 6022

**KEMENTERIAN AGAMA RI**
**UNIVERSITAS ISLAM NEGERI WALISONGO**
**FAKULTAS SAINS DAN TEKNOLOGI**
Jl. Prof. Dr. Hamka (Kampus II) Ngaliyan Semarang
Telp. (024) -7601295 Fax. 7615387

## PENGESAHAN

Naskah skripsi berikut ini:

| | | |
|---|---|---|
| Judul | : | Penyelesaian *Integer Knapsack Problem* Menggunakan Eksplorasi Algoritma *Greedy*, *Dynamic Programming*, *Brute Force* dan *Genetic* |
| Penulis | : | Muhammad Abdurrahman Rois |
| NIM | : | 1508046022 |
| Program Studi | : | Matematika |

Telah diujikan dalam sidang *munaqasyah* oleh Dewan Penguji Fakultas Sains dan Teknologi UIN Walisongo Semarang dan dapat diterima sebagai salah satu syarat memperoleh gelar sarjana dalam Ilmu Matematika.

Semarang, 24 Mei 2019

**DEWAN PENGUJI**

Ketua,

**Siti Maslihah, M.Si**
NIP: 19770611 201101 2 004

Sekretaris,

**Eva Khoirun Nisa, M.Si**
NIP: 19870102 201903 2 010

Penguji I,

**Emy Siswanah, M.Sc**
NIP: 19870202 201101 2 014

Penguji II,

**Yulia Romadiastri, M.Sc**
NIP: 19810715 200501 2 008

Pembimbing I,

**Siti Maslihah, M.Si**
NIP: 19770611 201101 2 004

Pembimbing II,

**Budi Cahyono, M.Si**
NIP: 19801215 200912 1 003

**NOTA DINAS**

Semarang, 9 Mei 2019

Kepada
Yth. Dekan Fakultas Sains dan Teknologi
UIN Walisongo
Di Semarang

*Assalamu'alaikum. wr.wb.*

Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi naskah skripsi dengan:

| | | |
|---|---|---|
| Judul | : | Penyelesaian *Integer Knapsack Problem* Menggunakan Eksplorasi Algoritma *Greedy*, *Dynamic Programming*, *Brute Force* dan *Genetic* |
| Nama | : | Muhammad Abdurrahman Rois |
| NIM | : | 1508046022 |
| Program Studi | : | Matematika |

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo Semarang untuk diujikan dalam sidang *munaqasyah.*
*Wassalamu'alaikum. wr. wb.*

Pembimbing I

**Siti Maslihah, M.Si**
**NIP: 19770611 201101 2 004**

**NOTA DINAS**

Semarang, 9 Mei 2019

Kepada
Yth. Dekan Fakultas Sains dan Teknologi
UIN Walisongo
Di Semarang

*Assalamu'alaikum. wr.wb.*

Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi naskah skripsi dengan:

| | | |
|---|---|---|
| Judul | : | Penyelesaian *Integer Knapsack Problem* Menggunakan Eksplorasi Algoritma *Greedy, Dynamic Programming, Brute Force* dan *Genetic* |
| Nama | : | Muhammad Abdurrahman Rois |
| NIM | : | 1508046022 |
| Program Studi | : | Matematika |

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo Semarang untuk diujikan dalam sidang *munaqasyah.*
*Wassalamu'alaikum. wr. wb.*

Pembimbing II

**Budi Cahyono, M.Si**
**NIP: 19801215 200912 1 003**

# ABSTRAK

*Knapsack problem* merupakan bagian dari algoritma optimasi yang bertujuan untuk memaksimalkan atau meminimalkan sebuah nilai. *Knapsack* sendiri merupakan masalah optimasi kombinatorial untuk memilih barang yang harus dimasukkan sampai batas maksimum dan mendapatkan nilai yang seoptimal mungkin.

Algoritma yang digunakan pada permasalahan *integer knapsack problem* adalah algoritma *greedy, dynamic programming, brute force* dan *genetic.* Tujuan dari peneliti adalah mencari keuntungan yang maksimum pada permasalahan *integer knapsack* dengan menggunakan algoritma *greedy, dynamic programming, brute force* dan *genetic,* serta membandingkan keempat algoritma tersebut pada permasalahan *integer knapsack* dari segi hasil dan waktu komputasi (detik). Hasil penelitian bagian pertama yaitu data barang sedikit dengan 5 barang dan bagian kedua yaitu data barang banyak dengan 30 barang menunjukan: (1) Keuntungan maksimum penggunaan algoritma *greedy* yaitu pertama, pada konsep *greedy by profit* bagian pertama sebesar Rp 152.000,- dengan total berat 8 kg sedangkan waktu komputasinya 0,17667 detik dan bagian kedua sebesar Rp 747.000,- dengan total berat 32 kg sedangkan waktu komputasinya 0,34943 detik. Kedua, konsep *greedy by weight* bagian pertama sebesar Rp 164.000,- dengan total berat 7 kg sedangkan waktu komputasinya 0,2015 detik dan bagian kedua sebesar Rp 588.000,- dengan total berat 29 kg sedangkan waktu komputasinya 0,21119 detik. Ketiga, konsep *greedy by density* bagian pertama sebesar Rp 138.000,- dengan total berat 5 kg sedangkan waktu komputasinya 0,3684 detik dan bagian kedua sebesar Rp 747.000,- dengan total berat 32 kg sedangkan waktu komputasinya 0,2709 detik. (2) Keuntungan maksimum

menggunakan algoritma *dynamic programming* bagian pertama sebesar Rp 208.000,- dengan berat 11 kg sedangkan waktu komputasinya 1,0732 detik dan bagian kedua Rp 747.000,- dengan total berat 32 kg sedangkan waktu komputasinya 2,8999 detik. (3) Keuntungan maksimum menggunakan algoritma *brute force* bagian pertama sebesar Rp 208.000,- dengan total berat 11 kg sedangkan waktu komputasinya 0,066716 detik dan bagian kedua dengan batasan 100.000 detik belum didapatkan hasilnya dan dilakukan peramalan untuk waktu komputasinya didapatkan 6.746.795,19 detik. Tetapi keuntungan, total beratnya pasti sama dengan hasil pada algoritma *dynamic programming*. (4) Keuntungan maksimum menggunakan algoritma *genetic* bagian pertama sebesar Rp 56.000,- dengan total berat 3 kg sedangkan waktu komputasinya 2,9238 detik dan bagian kedua Rp 742.000,- dengan total berat 32 kg sedangkan waktu komputasinya 7,0742 detik.

Hasil di atas dapat disimpulkan pada bagian pertama dan kedua bahwa algoritma *dynamic programming* dan *brute force* menghasilkan keuntungan yang optimum, tetapi algoritma *brute force* dengan jumlah barang yang banyak waktu yang dihasilkan juga semakin lama bahkan jika dibatasi dengan jumlah waktu tertentu akan tidak ditemukan hasilnya. Jadi algoritma *brute force* tidak efektif untuk data yang banyak. Penyelesaian algoritma *dynamic programming* memiliki waktu komputasi yang lebih besar daripada algoritma *brute force* pada bagian pertama dan *greedy*. Algoritma *genetic* solusinya optimum, tetapi hasilnya tidak stabil dengan nilai yang dihasilkan pertama kali karena dipengaruhi inisialisasi kromosom yang dilakukan secara random.

**Kata kunci:** *Knapsack problem*, algoritma *greedy,* algoritma *dynamic programming,* algoritma *brute force,* algoritma *genetic*

## KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Puji dan syukur penulis panjatkan kepada Allah SWT, yang telah memberikan nikmat, hidayah, kesempatan dan kesehatan, sehingga skripsi yang berjudul "**Penyelesaian *Integer Knapsack Problem* Menggunakan Eksplorasi Algoritma *Greedy, Dynamic Programming, Brute Force* dan *Genetic***" terselesaikan dengan baik.

Shalawat dan salam tidak lupa kita haturkan kepada junjungan kita, Nabi Agung Muhammad S.A.W. di mana sampai nafas yang kita hembuskan sekarang, kita masih bisa merasakan bagaimana indahnya kebesaran agama Islam beserta ilmu dan manfaat yang dapat kita rasakan sampai saat ini.

Penyusunan skripsi ini merupakan salah satu syarat yang harus dipenuhi oleh setiap mahasiswa dalam menyelesaikan pendidikan di strata 1 (S1) di Program Studi Matematika Fakultas Sains dan Teknologi UIN Walisongo Semarang. Penyusunan skripsi ini tidak dapat penulis selesaikan tanpa adanya dukungan dan bimbingan dari berbagai pihak, baik secara langsung maupun tidak langsung.

Oleh karena itu, penulis ingin mengucapkan terima kasih kepada:

1. Dr. H. Ruswan, M.A selaku dekan Fakultas Sains dan Teknologi;
2. Emy Siswanah, M.Sc dan Siti Maslihah, M.Si selaku ketua dan sekretaris Prodi Matematika Fakultas Sains dan Teknologi UIN Walisongo Semarang;
3. Siti Maslihah, M.Si selaku Pembimbing 1 dan Budi Cahyono, M.Si selaku Pembimbing 2, dengan penuh kesabaran meluangkan waktu dan pikirannya untuk memberikan bimbingan, arahan, dan petunjuk mulai dari awal hingga selesai skripsi ini;
4. Segenap dosen Prodi Matematika dan Pendidikan Matematika yang telah memberikan kesempatan kepada penulis untuk mengikuti pendidikan, pengajaran ilmu pengetahuan, dan pelayanan yang layak selama penulis melakukan studi;
5. Segenap dosen dan karyawan Fakultas Sains dan Teknologi yang sudah memberikan ilmu dan sudah membantu dalam proses studi;
6. Segenap keluarga besar penulis, terkhusus dan teristimewa Bapak Mansur dan Ibu Muyassarah yang selalu memberi doa dan dukungan baik lahir maupun batin yang tiada terhingga;
7. Adik, dan keponakan yang sudah memberi semangat dan dukungan;

8. HMJ Matematika, senior dan junior Matematika UIN Walisongo Semarang yang selama ini memberikan banyak motivasi dan bantuan bagi penulis. Kepada UKM RISALAH (Rebana Ilmu Seni Al-Qur'an dan Tilawah) dan UKM BITA (Bimbingan Ilmu TIlawah Al-Qur'an) yang sudah memberikan banyak bantuan dan dukungan bagi penulis. Kepada teman-teman dari SENAT, BEM Fakultas Sains dan Teknologi, PMII Fakultas Sains dan Teknologi, dan keluarga besar GENBI (Generasi Baru Indonesia) tingkat UIN Walisongo, tingkat kota Semarang dan tingkat se-Jawa Tengah.
9. Teman-teman Prodi Matematika dan Pendidikan Matematika tahun 2015 – 2018, yang sudah membantu dan memberikan dukungan kepada penulis untuk menyelesaikan skripsi ini;
10. Keluarga pondok pesantren Al-Ma'rufiyyah yang sudah banyak membantu, memberikan doa sekaligus dukungan untuk menyelesaikan skripsi ini;
11. Sahabat-sahabat KKN MIT ke-VII tahun 2019 dan segenap warga di kelurahan Patemon-Gunungpati yang sudah memberikan dukungan, doa untuk kelancaran dan kemudahan dalam penyelesaian skripsi ini;
12. Rini A.W. Hartanti selaku Deputi Direktur (Kepala Divisi), Ignatius Adhi Nugroho selaku Asisten Direktur (Kepala Tim SP PUR dan KI), Kiptiah Riyanti selaku Manajer KI dan Achmad Jainuri selaku Manajer SP PUR,

Aisyah Nur Heniar dan Anggie Andeta, Utuy, Ning Widyowati, Andrian dan pegawai atau staff di Kantor Perwakilan Bank Indonesia Provinsi Jawa Tengah yang telah membantu selama proses studi S1;

Penulis berharap semoga skripsi ini dapat memberikan manfaat baik bagi semua pihak yang berkepentingan. Penulis juga menyadari bahwa dalam menyusun skripsi ini masih terdapat banyak kekurangan baik isi maupun susunannya. Oleh karena itu, penulis mengharapkan saran dan kritik demi penyempurnaan skripsi ini.

Semoga segala bantuan, dukungan dan bimbingan yang telah diberikan kepada penulis untuk menyusun skripsi dinilai oleh Allah S.W.T sebagai amal shaleh dan dibalas dengan pahala yang berlipat ganda. Aamiin.

Semarang, 15 April 2019

**Penulis**

Muhammad Abdurrahman Rois
NIM. 150 804 6022

## DAFTAR ISI


HALAMAN JUDUL ..... i
PERNYATAAN KEASLIAN ..... ii
PENGESAHAN ..... iii
NOTA DINAS ..... iv
ABSTRAK ..... vi
KATA PENGANTAR ..... viii
DAFTAR ISI ..... xii
DAFTAR TABEL ..... xv
DAFTAR GAMBAR ..... xviii
DAFTAR LAMPIRAN ..... xxi
BAB I PENDAHULUAN ..... 1
A. Latar Belakang ..... 1
B. Rumusan Masalah ..... 8
C. Tujuan Penelitian ..... 9
D. Batasan Penelitian ..... 10
E. Manfaat Penelitian ..... 10
F. Sistematika Penulisan ..... 11
BAB II TINJAUAN PUSTAKA ..... 13
A. Kajian Teori ..... 13
1. *Knapsack Problem* ..... 13

2. Pengertian Algoritma ..... 18
3. Algoritma *Greedy* ..... 21
4. Algoritma *Dynamic Programming* ..... 32
5. Algoritma *Brute Force* ..... 37
6. Algoritma *Genetic* ..... 41
B. Kajian Pustaka ..... 65
BAB III METODE PENELITIAN ..... 66
A. Jenis Penelitian ..... 66
B. Tempat dan Waktu Penelitian ..... 67
C. Prosedur Penelitian ..... 67
BAB IV HASIL DAN PEMBAHASAN ..... 72
A. Hasil ..... 72
1. Data Pengiriman Barang ..... 72
2. Program GUI untuk *Integer Knapsack Problem* . 76
3. Penyelesaian *Integer Knapsack Problem* ..... 79
a) Penyelesaian Menggunakan Algoritma *Greedy* ..... 79
b) Penyelesaian Menggunakan Algoritma *Dynamic Programming* ..... 90
c) Penyelesaian Menggunakan Algoritma *Brute Force* ..... 96

d) Penyelesaian Menggunakan Algoritma *Genetic* ..... 103
B. Pembahasan ..... 108
BAB V PENUTUP ..... 113
A. Kesimpulan ..... 113
B. Saran ..... 115
DAFTAR PUSTAKA ..... 116
LAMPIRAN ..... 120
RIWAYAT HIDUP ..... 138

## DAFTAR TABEL


| Tabel | Judul | Halaman |
|---|---|---|
| Tabel 2.1 | Kelompok Algoritma Berdasarkan Notasi Big-*O*…… | 21 |
| Tabel 2.2 | *Greedy by Profit*…… | 25 |
| Tabel 2.3 | *Greedy by Weight*…… | 25 |
| Tabel 2.4 | *Greedy by Density*…… | 26 |
| Tabel 2.5 | Hasil akhir *greedy*…… | 26 |
| Tabel 2.6 | Kondisi 1: $i = 0, v[0] = 0, w[0] = 0$ …… | 34 |
| Tabel 2.7 | Kondisi 2: $i = 1, v[1] = 3, w[1] = 2$ …… | 34 |
| Tabel 2.8 | Kondisi 3: $i = 2, v[2] = 4, w[2] = 3$ …… | 35 |
| Tabel 2.9 | Kondisi 4: $i = 3, v[3] = 5, w[3] = 4$ …… | 35 |
| Tabel 2.10 | Kondisi 5: $i = 4, v[4] = 6, w[4] = 5$ …… | 35 |
| Tabel 2.11 | Hasil *brute force*…… | 39 |
| Tabel 2.12 | Berat dan Keuntungan masing-masing barang…… | 43 |
| Tabel 2.13 | Hasil perhitungan rasio nilai/berat adalah *array* T…… | 43 |
| Tabel 2.14 | Diurutkan secara *descending* berdasarkan T…… | 43 |
| Tabel 2.15 | Urutan angka acak setiap kromosom | 50 |
| Tabel 2.16 | Hasil pengacakan allele-allele…… | 51 |
| Tabel 2.17 | Misalkan gen yang terpilih untuk dimutasi…… | 52 |
| Tabel 2.18 | Berat dan keuntungan masing-masing berat…… | 53 |
| Tabel 2.19 | Hasil perhitungan rasio nilai/ berat adalah *array* T…… | 53 |

| | | |
|---|---|---|
| Tabel 2.20 | Diurutkan secara *descending* berdasarkan T…….. | 53 |
| Tabel 2.21 | Misalkan hasil setelah 4 kali memutar 4 kali memutar roda *rolette*…….. | 61 |
| Tabel 2.22 | Hasil *decoding* yang dilakukan terhadap $v_{real}$ …….. | 63 |
| Tabel 4.1 | Laporan J&T Express *drop point* Ngaliyan kota Semarang tahun 2019… | 73 |
| Tabel 4.2 | Tabel daftar barang beserta berat dan *value/profit*…….. | 74 |
| Tabel 4.3 | Lanjutan…….. | 75 |
| Tabel 4.4 | Pengambilan barang menggunakan konsep *greedy by profit* bagian pertama…….. | 80 |
| Tabel 4.5 | Pengambilan barang menggunakan konsep *greedy by profit* bagian kedua…….. | 80 |
| Tabel 4.6 | Lanjutan…….. | 81 |
| Tabel 4.7 | Pengambilan barang menggunakan konsep *greedy by weight* bagian pertama …….. | 83 |
| Tabel 4.8 | Pengambilan barang menggunakan konsep *greedy by weight* bagian kedua. | 84 |
| Tabel 4.9 | Tabel daftar barang beserta berat, *value/ profit* dan *density*…….. | 86 |
| Tabel 4.10 | Lanjutan…….. | 87 |
| Tabel 4.11 | Pengambilan barang menggunakan konsep *greedy by density* bagian pertama…….. | 87 |

| | | |
|---|---|---|
| Tabel 4.12 | Pengambilan barang menggunakan konsep *greedy by density* bagian kedua..... | 88 |
| Tabel 4.13 | Ringkasan hasil *output* program algoritma *dynamic programming* bagian pertama..... | 94 |
| Tabel 4.14 | Ringkasan hasil *output* program algoritma *dynamic programming* bagian kedua..... | 94 |
| Tabel 4.15 | Ringkasan hasil *output* program algoritma *brute force*..... | 97 |
| Tabel 4.16 | Tabulasi data menggunakan peramalan metode trend..... | 100 |
| Tabel 4.17 | Hasil nilai peramalan..... | 102 |
| Tabel 4.18 | Hasil perhitungan bagian pertama..... | 108 |
| Tabel 4.19 | Hasil perhitungan bagian kedua..... | 109 |
| Tabel 4.20 | Waktu komputasi dalam proses penyelesaian..... | 111 |
| Tabel 5.1 | Penyelesaian algoritma *greedy*..... | 113 |
| Tabel 5.2 | Penyelesaian algoritma *dynamic programming*..... | 113 |
| Tabel 5.3 | Penyelesaian algoritma *brute force*..... | 114 |
| Tabel 5.4 | Penyelesaian algoritma *genetic*..... | 114 |

## DAFTAR GAMBAR


| Gambar | Judul | Halaman |
|---|---|---|
| Gambar 2.1 | *Flowchart* algoritma *greedy* input data awal…….. | 28 |
| Gambar 2.2 | *Flowchart* algoritma *greedy by weight*…… | 29 |
| Gambar 2.3 | *Flowchart* algoritma *greedy by profit*.. | 30 |
| Gambar 2.4 | *Flowchart* algoritma *greedy by density*…….. | 31 |
| Gambar 2.5 | *Flowchart* algoritma *dynamic programming*…….. | 36 |
| Gambar 2.6 | *Flowchart* algoritma *brute force*……. | 40 |
| Gambar 2.7 | Ilustrasi arti dari kromosom $v_1$ ……. | 44 |
| Gambar 2.8 | Ilustrasi perhitungan nilai *fitness dari* $v_1$ …… | 44 |
| Gambar 2.9 | Ilustrasi arti dari kromosom $v_2$ ……. | 45 |
| Gambar 2.10 | Ilustrasi perhitungan nilai *fitness dari* $v_2$ …… | 46 |
| Gambar 2.11 | Ilustrasi arti dari kromosom $v_3$ ……. | 46 |
| Gambar 2.12 | Ilustrasi perhitungan nilai *fitness dari* $v_3$ …… | 47 |
| Gambar 2.13 | Ilustrasi arti dari kromosom $v_4$ ……. | 47 |
| Gambar 2.14 | Ilustrasi perhitungan nilai *fitness dari* $v_4$ …… | 48 |
| Gambar 2.15 | Ilustrasi arti dari kromosom $v_5$ ……. | 49 |
| Gambar 2.16 | Ilustrasi perhitungan nilai *fitness dari* $v_5$ …… | 50 |

| | | |
|---|---|---|
| Gambar 2.17 | Ilustrasi arti dari kromosom $v_1$ ..... | 53 |
| Gambar 2.18 | Ilustrasi perhitungan nilai *fitness dari* $v_1$ ..... | 54 |
| Gambar 2.19 | Ilustrasi arti dari kromosom $v_2$ ..... | 54 |
| Gambar 2.20 | Ilustrasi perhitungan nilai *fitness dari* $v_2$ ..... | 55 |
| Gambar 2.21 | Ilustrasi arti dari kromosom $v_3$ ..... | 56 |
| Gambar 2.22 | Ilustrasi perhitungan nilai *fitness dari* $v_3$ ..... | 56 |
| Gambar 2.23 | Ilustrasi arti dari kromosom $v_4$ ..... | 57 |
| Gambar 2.24 | Ilustrasi perhitungan nilai *fitness dari* $v_4$ ..... | 58 |
| Gambar 2.25 | Ilustrasi arti dari kromosom $v_5$ ..... | 58 |
| Gambar 2.26 | Ilustrasi perhitungan nilai *fitness dari* $v_5$ ..... | 59 |
| Gambar 2.27 | *Flowchart* algoritma *genetic*..... | 64 |
| Gambar 3.1 | Langkah-langkah penelitian..... | 71 |
| Gambar 4.1 | Tampilan awal program..... | 76 |
| Gambar 4.2 | Tampilan setelan data diisi..... | 79 |
| Gambar 4.3 | Hasil perhitungan algoritma *greedy by profit* bagian pertama..... | 81 |
| Gambar 4.4 | Hasil perhitungan algoritma *greedy by profit* bagian kedua..... | 82 |
| Gambar 4.5 | Hasil perhitungan algoritma *greedy by weight* bagian pertama..... | 84 |
| Gambar 4.6 | Hasil perhitungan algoritma *greedy by weight* bagian kedua..... | 85 |
| Gambar 4.7 | Hasil perhitungan algoritma *greedy* | |

| | | |
|---|---|---|
| | *by density* bagian pertama…….. | 89 |
| Gambar 4.8 | Hasil perhitungan algoritma *greedy by density* bagian kedua…….. | 89 |
| Gambar 4.9 | Hasil perhitungan algoritma *dynamic programming* bagian pertama…….. | 95 |
| Gambar 4.10 | Hasil perhitungan algoritma *dynamic programming* bagian kedua…….. | 95 |
| Gambar 4.11 | Hasil perhitungan algoritma *brute force* bagian pertama…….. | 98 |
| Gambar 4.12 | Hasil perhitungan algoritma *genetic* bagian pertama…….. | 105 |
| Gambar 4.13 | Hasil perhitungan algoritma *genetic* bagian kedua…….. | 105 |

## DAFTAR LAMPIRAN


| Lampiran | Judul | Halaman |
|---|---|---|
| 1 | Ijin penelitian dengan karyawan J&T Express *drop point* Ngaliyan kota Semarang ..... | 120 |
| 2 | *Interview* dengan karyawan J&T Express *drop point* Ngaliyan kota Semarang ..... | 120 |
| 3 | Surat penunjukan dosen pembimbing..... | 121 |
| 4 | Hasil perhitungan algoritma *brute force* 6 barang..... | 122 |
| 5 | Hasil perhitungan algoritma *brute force* 8 barang..... | 122 |
| 6 | Hasil perhitungan algoritma *brute force* 10 barang..... | 123 |
| 7 | Hasil perhitungan algoritma *brute force* 12 barang..... | 123 |
| 8 | Hasil perhitungan algoritma *brute force* 14 barang..... | 124 |
| 9 | Hasil perhitungan algoritma *brute force 1*6 barang..... | 124 |
| 10 | Hasil perhitungan algoritma *brute force* 18 barang..... | 125 |
| 11 | Hasil perhitungan algoritma *brute force* 20 barang..... | 125 |
| 12 | Hasil perhitungan algoritma *brute force* 22 barang..... | 126 |
| 13 | Peramalan dengan metode *trend* | |

| | | |
|---|---|---|
| | *linear*…….. | 126 |
| 14 | Peramalan dengan metode *trend kuadratik*…….. | 126 |
| 15 | Peramalan dengan metode *trend eksponensial*…….. | 127 |
| 16 | *Script code* algoritma *greedy by weight* untuk KP 01…… | 127 |
| 17 | *Script code* algoritma *greedy by profit* untuk KP 01…… | 128 |
| 18 | *Script code* algoritma *greedy by density* untuk KP 01…….. | 130 |
| 19 | *Script code* algoritma *dynamic programming* untuk KP 01…… | 132 |
| 20 | *Script code* algoritma *brute force* untuk KP 01…… | 133 |
| 21 | *Script code* algoritma *genetic* untuk KP 01…….. | 135 |

# BAB I

## PENDAHULUAN

### A. Latar Belakang

Ilmu pengetahuan dalam ajaran Agama Islam kedudukannya sangat penting, hal ini terlihat dalam ayat-ayat Al-Qur'an yang memberikan tanda-tanda Ilmu Pengetahuan dan menyuruh semua manusia untuk menggali, mengkaji dan memastikan ilmu tersebut serta supaya mengambil pelajaran (hikmah) bahwa kekuasan Allah memang luas dan sempurna. Keutamaan ilmu dan ahli ilmu banyak disebutkan dalam Al-Qur'an dan As-Sunnah. Berdasarkan Tim Baitul Hikmah Jogjakarta (2014), kata ilmu sendiri dalam Al-Qur'an dengan berbagai bentuknya berulang sebanyak 854 kali. Di bawah ini salah satu ayat yang mengungkap keutamaan ilmu dan mengetahui ilmu yaitu:

شَهِدَ ٱللَّهُ أَنَّهُۥ لَآ إِلَٰهَ إِلَّا هُوَ وَٱلْمَلَٰٓئِكَةُ وَأُو۟لُوا۟ ٱلْعِلْمِ قَآئِمًۢا بِٱلْقِسْطِ ۚ لَآ إِلَٰهَ إِلَّا هُوَ ٱلْعَزِيزُ ٱلْحَكِيمُ ١٨

Artinya: "*Allah menyatakan bahwasanya tidak ada Tuhan melainkan Dia (yang berhak disembah), Yang menegakkan keadilan. Para Malaikat dan orang-orang yang berilmu (juga menyatakan yang demikian itu). Tak ada Tuhan melainkan Dia (yang berhak disembah), Yang Maha Perkasa lagi Maha Bijaksana.*" (Q.S. Ali Imron: 18)

Tanda-tanda ilmu pengetahuan yang disebutkan dalam Al-Qur'an ada banyak sekali, dan bahkan ilmuwan-ilmuwan yang menemukan terheran-heran karena Al-Qur'an turun pada abad yang belum ada teknologi canggih tetapi sudah menuliskan ilmu pengetahuan yang baru ditemukan di zaman modern. Subhanallah Maha Besar Allah S.W.T.

Kemajuan teknologi informasi saat ini menggiring dunia pada perkembangan baru dari masa ke masa termasuk pengiriman barang, pelayanan dari perusahaan jasa pengiriman barang ini sangat penting karena membantu manusia dalam memudahkan aktifitas dan lebih menghemat waktu dan tenaga (Nuraeni, 2016). Proses pengiriman barang tentu mengeluarkan biaya dalam proses pengirimannya, apalagi jarak antara tempat pengiriman yang satu dengan tempat pengiriman lainnya berbeda-beda dan cukup jauh. Agar biaya yang dikeluarkan sedikit dan memperoleh keuntungan yang maksimal, maka barang-barang yang didistribusikan sebaiknya dipilih secermat mungkin. Karena masyarakat di zaman ini sangat antusias untuk melakukan pengiriman barang, maka banyak berdiri perusahaan jasa pengiriman barang. Sebagai contoh adalah perusahaan jasa pengiriman yang baru berdiri 3 tahun lamanya dan pada

tahun 2018 mendapatkan penghargaan bergengsi yaitu *TOP BRAND AWARD.* Jasa pengiriman barang ini yaitu J&T Express.

Jasa pengiriman J&T Express berupaya untuk mendapatkan keuntungan dan juga konsumen tidak terbebani dengan biaya yang besar. Berdasarkan hasil wawancara, misalkan pengiriman barang lebih mendahulukan jarak lokasi pengiriman yang lebih jauh karena memiliki nilai/ *value* yang besar. Pada jasa pengiriman J&T Express memiliki banyak jenis paket pengiriman yang disesuaikan kebutuhan konsumen. Salah satu paket yang paling digemari konsumen yaitu paket reguler yang disebabkan faktor harga yang ekonomis dibanding paket-paket lainnya. Paket reguler dapat menjangkau seluruh Indonesia dalam batas jangka waktu 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/ *value* 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 yang harus mencari solusi terbaik dari banyak kemungkinan yang sudah ada

(Vala, Monaka, dan Pandya, 2014). 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* (0/1) 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 dan 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 (Supranto, 1988). 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 berhubungan dengan 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.

Masalah optimasi dalam kehidupan sehari-hari ternyata banyak digunakan oleh masyarakat untuk memenuhi kebutuhannya. Agama Islam menganjurkan kepada semua manusia bahwa dalam hidup tidak boleh mempunyai sikap boros yang berarti dalam hidup harus mengoptimalkan suatu pekerjaan supaya tidak mengulang lagi padahal sebenarnya

sudah bisa dikerjakan atau diambil pada pekerjaan sebelumnya yang bertujuan dapat menghemat suatu biaya atau yang lain. Di bawah ini ada ayat Al-Qur'an yang menggambarkan bahwa tidak boleh bersikap boros:

إِنَّ ٱلْمُبَذِّرِينَ كَانُوٓاْ إِخْوَٰنَ ٱلشَّيَٰطِينِ ۖ وَكَانَ ٱلشَّيْطَٰنُ لِرَبِّهِۦ كَفُورًا ٢٧

Artinya: "*Sesungguhnya pemboros-pemboros itu adalah saudara-saudara syaitan dan syaitan itu adalah sangat ingkar kepada Tuhannya*" (Q.S. Al-Isra': 27)

Masalah optimasi dalam bidang tertentu, misalnya seorang ahli Teknik Sipil ingin merencanakan membangun gedung dengan biaya minimum tetapi menginginkan kualitas dan faktor keamanan yang tinggi, dan masih banyak lagi masalah optimasi dalam kehidupan sehari-hari atau dalam bidang yang lain.

Persoalan *knapsack problem,* khususnya *integer knapsack problem* dapat diselesaikan menggunakan menggunakan berbagai cara. Beberapa cara diantaranya adalah algoritma *greedy, dynamic programming, brute force*, dan *genetic*. Algoritma tersebut sama-sama dapat menyelesaikan permasalahan *integer knapsack* dan menghasilkan solusi optimum.

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 dan Zhang (2018) yang dapat digunakan untuk menyelesaikan permasalahan *knapsack* yaitu dengan algoritma *dynamic programming*, algoritma *brute force* dan algoritma *genetic*. Dari latar belakang tersebut peneliti tertarik untuk membahas penyelesaian persoalan tersebut dengan algoritma *greedy, dynamic programming, brute force* dan *genetic.* Tegasnya, 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.

Berdasarkan masalah di atas, maka permasalahan yang digunakan dalam penelitian ini adalah permasalahan *integer knapsack* untuk mencari keuntungan maksimum di J&T Express dengan menggunakan algoritma *greedy, dynamic programming, brute force* dan *genetic.* Jadi, peneliti mengangkat judul yaitu "Penyelesaian *Integer Knapsack Problem* Menggunakan Eksplorasi Algoritma *Greedy, Dynamic Programming, Brute Force Dan Genetic"*.

## B. Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka rumusan yang akan di bahas pada penelitian ini dengan studi kasus J&T Express *drop point* Ngaliyan Semarang adalah:

1. Bagaimana penyelesaian *integer knapsack problem* menggunakan algoritma *greedy*?
2. Bagaimana penyelesaian *integer knapsack problem* menggunakan algoritma *dynamic programming*?
3. Bagaimana penyelesaian *integer knapsack problem* menggunakan algoritma *brute force*?
4. Bagaimana penyelesaian *integer knapsack problem* menggunakan algoritma *genetic*?
5. Bagaimana implementasi keempat algoritma dalam *integer knapsack problem* menggunakan *software MATLAB v2017a* berbasis *GUI*?
6. Bagaimana perbandingan keempat algoritma yang memberikan solusi optimal untuk permasalahan *integer knapsack problem* (berdasarkan hasil maksimal maupun waktu (detik) minimum yang dibutuhkan)?

## C. Tujuan Penelitian

Berdasarkan rumusan masalah di atas, maka tujuannya adalah:

1. Mengetahui penyelesaian optimum pada permasalahan *integer knapsack problem* menggunakan algoritma *greedy.*
2. Mengetahui penyelesaian optimum pada permasalahan *integer knapsack problem* menggunakan algoritma *dynamic programming.*
3. Mengetahui penyelesaian optimum pada permasalahan *integer knapsack problem* menggunakan algoritma *brute force.*
4. Mengetahui penyelesaian optimum pada permasalahan *integer knapsack problem* menggunakan algoritma *genetic.*
5. Membuat program penyelesaian menggunakan *software MATLAB v2017a* berbasis GUI.
6. Mengetahui perbandingan metode algoritma yang dilihat berdasarkan hasil dan waktu (detik) yang dibutuhkan pada algoritma *greedy,* algoritma *dynamic programming,* algoritma *brute force,* dan algoritma *genetic* dalam penyelesaian *integer knapsack problem.*

## D. Batasan Penelitian

Adapun batasan penelitian ini adalah:

1. Mencari data distribusi barang yang digunakan dalam penelitian ini adalah nama/ kode barang, harga pengiriman barang, dan berat suatu barang.
2. Studi kasus di J&T *drop point* area Ngaliyan kota Semarang dengan paket Reguler.
3. *Value/ profit* ditentukan berdasarkan biaya pengiriman.
4. Menggunakan metode algoritma *greedy* (konsep *greedy by profit, greedy by weight, greedy by density*), algoritma *dynamic programming,* algoritma *brute force,* dan algoritma *genetic.*
5. *Knapsack problem* yang digunakan adalah *integer knapsack problem.*

## E. Manfaat Penelitian

1. Bagi Peneliti

   Hasil penelitian ini dapat menambah pengetahuan dan wawasan peneliti tentang metode dalam penyelesaian *integer knapsack problem.* Selain itu, penelitian ini menjadi pengalaman berharga bagi peneliti dalam menerapkan ilmu yang diperoleh di perkuliahan.

2. Lembaga Kampus UIN Walisongo Semarang

   Hasil penelitian ini menjadi bahan informasi untuk menambah khazanah ilmu pengetahuan di Lembaga Kampus UIN Walisongo Semarang, khususnya Prodi Matematika Jurusan Matematika Fakultas Sains dan Teknologi dan dapat digunakan referensi untuk penelitian selanjutnya.

3. J&T Express

   Hasil penelitian ini menjadi bahan informasi untuk membantu pengepakan di J&T Express, sebagai alternatif untuk mendapatkan keuntungan maksimum.

## F. Sistematika Penulisan

1. Bab I (Pendahuluan)

   Bab ini membahas tentang isi keseluruhan penulisan skripsi yang terdiri dari latar belakang, rumusan masalah yang akan dimunculkan dalam pembahasan, tujuan penelitian memaparkan tujuan yang ingin dicapai oleh peneliti, manfaat penelitian, batasan masalah memaparkan tentang bagaimana masalah dibatasi supaya pembahasan tidak terlalu luas pembahasannya, dan sistematika penulisan berisi tentang apa saja yang dibahas pada masing-masing bab.

2. Bab II (Tinjauan Pustaka)

   Bab ini menjelaskan mengenai teori-teori yang berhubungan dangan penelitian.

3. Bab III (Metode Penelitian)

   Bab ini membahas mengenai metode-metode atau cara penelitian yang akan digunakan oleh peneliti.

4. Bab IV (Hasil dan Pembahasan)

   Bab ini membahas penyajian hasil penelitian yang dilakukan secara langsung oleh penulis di lapangan dan pembahasannya.

5. Bab V (Penutup)

   Bab ini adalah penutup skripsi yang terdiri dari kesimpulan hasil penelitian dan saran untuk perbaikan penelitian selanjutnya.

# BAB II

# TINJAUAN PUSTAKA

## A. Kajian Teori

### 1. *Knapsack Problem*

*Knapsack problem* atau *rucksack problem* secara bahasa adalah masalah tempat/ ransel yang diartikan lebih lanjut yaitu masalah pengepakan. Masalah tersebut, menurut Vala, Monaka, dan Pandya (2014) merupakan masalah optimasi kombinatorial dimana harus memilih dan mencari solusi yang terbaik dari berbagai banyak pilihan yang ada. Selanjutnya, menurut Juwita, Susanto, dan Halomoan (2017) *knapsack problem* merupakan suatu permasalahan pemilihan barang yang akan disimpan atau dimasukkan ke suatu tempat yang memiliki keterbatasan kapasitas. Oleh karena itu, dapat disimpulkan bahwa setiap barang memiliki berat, dan *value/ profit* 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. *Knapsack problem* atau masalah pengepakan barang di dunia nyata bisa memiliki karakteristik yang berbeda-beda dan dapat dikelompokkan menjadi 3 jenis yaitu:

a) *Unbounded Knapsack Problem*

Masalah *knapsack* tidak terbatas didefinisikan sebagai berikut: Diberikan $n$ barang misal $O = \{o_1, o_2, ...o_n\}$ adalah barang tidak ada pembatasan jumlah. Barang-barang tersebut memiliki masing-masing berat $(w_i)$ dan *value/ profit* $(p_i)$. Masalahnya adalah untuk memilih subset dari barang-barang yang ada, dan bertujuan untuk memaksimalkan berat dan nilai keseluruhannya yang tidak melebihi kapasitas tempat $(W)$. Kemudian diasumsikan bahwa semua berat dan *value/ profit* adalah positif, semua berat kurang dari atau sama dengan kapasitas tempatnya $(W)$, dan berat keseluruhan semua barang lebih dari kapasitas tempatnya $(W)$. Berdasarkan Lin et al. (2017), model masalah *knapsack* tak terbatas dapat dirumuskan sebagai berikut:

Fungsi tujuan optimal :

$$Z = \sum_{i=1}^{n} v_i x_i \qquad (2.1)$$

Fungsi kendala :

$$z = \sum_{i=1}^{n} w_i x_i \le W \qquad (2.2)$$

$\forall x_i \in Z^+, 1 \le i \le n$

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*

$x_i =$ Jumlah barang yang dimasukkan

b) *Integer Knapsack Problem*

Masalah *integer knapsack* adalah salah satu masalah optimasi kombinatorial yang paling banyak dipelajari. Masalah ini bertujuan untuk memaksimalkan total *value/ profit* barang ke tempat yang diinginkan, dan yang dimaksud kendala adalah memastikan jumlah berat kurang dari atau sama dengan kapasitas tempatnya. Keputusan dalam 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. Berdasarkan Lin et al. (2017), model masalah *integer knapsack* dapat dirumuskan sebagai berikut:

Fungsi tujuan optimal :

$$Z = \sum_{i=1}^{n} v_i y_i \qquad (2.3)$$

Fungsi kendala :

$$z = \sum_{i=1}^{n} w_i y_i \leq W \qquad (2.4)$$

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

Keterangan:

$Z =$ Nilai optimum dari fungsi tujuan

$z =$ Kendala dari fungsi tujuan

$n =$ Jumlah objek

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

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

$W =$ Kapasitas *Knapsack*

$y_i$ = Menunjukkan barang dimasukkan atau tidak

c) *Fractional Knapsack Problem*

Masalah *fractional knapsack* adalah barang yang dapat dipecah menjadi bagian yang lebih kecil, sehingga pencuri atau pembawa dapat memutuskan untuk hanya membawa sebagian kecil dari barang $i$. Sehingga, masalah *fractional knapsack* yaitu menghilangkan batasan untuk membawa barang sepenuhnya dan menjadikan barang dapat dipecah menjadi bagian yang lebih kecil. Model masalah *fractional knapsack* berdasarkan Goyal dan Parashar (2016) dapat dirumuskan sebagai berikut:

Fungsi tujuan optimal :

$$Z = \sum_{i=1}^{n} v_i u_i \qquad (2.5)$$

Fungsi kendala :

$$z = \sum_{i=1}^{n} w_i u_i \le W \qquad (2.6)$$

Dimana $0 \le u_i \le 1$

Keterangan:

$Z =$ Nilai optimum dari fungsi tujuan

$z =$ Kendala dari fungsi tujuan

$n =$ Jumlah objek

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

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

$W =$ Kapasitas *Knapsack*

$u_i =$ Menunjukkan barang-barang dimasukkan

*Knapsack problem* pada penelitian ini hanya fokus pada jenis *integer knapsack problem.* Keputusan yang diperoleh menggunakan jenis *integer knapsack problem* adalah hanya bernilai 1 dan 0. Bernilai 1 mengandung arti jika barang dipilih atau dimasukkan dan bernilai 0 mengandung arti jika barang tidak dipilih atau tidak dimasukkan.

**2. Pengertian Algoritma**

Algoritma menurut Cormen, Leiserson, Rivest, dan Stein (2001) adalah prosedur komputasi yang terdefinisi baik dengan mengambil beberapa nilai atau kumpulan nilai sebagai *input* dan menghasilkan nilai atau kumpulan nilai sebagai *output*. Selanjutnya, algoritma menurut Suarga (2015) adalah dasar pemikiran dalam menyusun penyelesaian suatu masalah dengan langkah-langkah melalui program komputer. Jadi dapat disimpulkan bahwa algoritma adalah adanya *input* dan disusun setiap langkahnya secara berurutan atau sistematis untuk menghasilkan *output* sesuai tujuan melalui program komputer. Algoritma harus menghasilkan *output* yang efektif dengan waktu yang relatif singkat dan berakhir memperoleh solusi.

a) Efisiensi Algoritma

Algoritma yang disusun untuk memecahkan masalah yang sama seringkali sangat berbeda-beda dalam efisiensi algoritmanya. Perbedaan-perbedaan ini bisa jauh lebih signifikan daripada perbedaan karena perangkat keras dan perangkat lunak.

Tujuan utama mempelajari masalah *knapsack* adalah pengembangan metode solusi, yaitu algoritma, yang menghitung solusi optimal atau perkiraan untuk setiap masalah yang diberikan. Kinerja umum komputasi menurut Kellerer, Pferschy, dan Pisinger (2004) ditentukan oleh waktu berjalan, yang diperlukan untuk memecahkan masalah yang diberikan. Aspek penting lainnya adalah tingkat kesulitan algoritma karena metode yang "mudah" akan lebih disukai dan digunakan daripada algoritma rumit yang lebih sulit untuk diterapkan.

Algoritma yang bagus adalah algoritma yang efisien. Algoritma dapat dikatakan efisien jika kebutuhan waktu untuk tahapan dalam komputasinya berjumlah sedikit. Algoritma dikatakan efisien juga berfungsi untuk membandingkan algoritma yang lebih baik untuk menyelesaikan permasalahan yang sama.

b)Notasi *Big-O*

Notasi *Big-O* adalah notasi matematika yang digunakan untuk menggambarkan tingkah laku asimtotik. Notasi *Big-O* berguna untuk menganalisa kompleksitas waktu dari suatu algoritma penyelesaian. Ada beberapa kelompok algoritma berdasarkan notasi *Big-O* yang dihasilkan dari perhitungan kompleksitas algoritma.

Algoritma yang menggambarkan tingkah laku asimptotik berdasarkan Kellerer et al. (2004) dibedakan menjadi tiga yaitu:

1. Algoritma *polinomial*, misalnya $O\left(n\right)$, $O\left(n \log n\right)$, atau $O\left(kn\right)$ untuk *k* adalah konstanta.
2. Algoritma *pseudopolinomial*, misalnya $O\left(cn\right)$, untuk *c* adalah koefisien dari kapasitas. Artinya masalah sederhana dengan jumlah barang yang sedikit akan memiliki koefisien yang banyak maka waktu berjalan lebih lama daripada algoritma polinomial.
3. Algoritma *non-polinomial*, misalnya $O\left(2^n\right)$ atau $O\left(3^n\right)$, artinya waktunya tidak terbatas (waktu yang dibutuhkan lebih lama dari algoritma *pseudopolinomial*)

Kelompok algoritma tersebut disusun berdasarkan urutan dari kompleksitas waktu terbaik kemudian menurun secara urut yang dapat dilihat pada Tabel 2.1. di bawah ini:

**Tabel 2.1.** Kelompok Algoritma Berdasarkan Notasi *Big-O*

| Kelompok Algoritma | Nama |
|---|---|
| $O(1)$ | Konstan |
| $O(\log\ n)$ | Logaritmik |
| $O(n)$ | Linier |
| $O(n \log n)$ | $n \log n$ |
| $O(n^2)$ | Kuadratik |
| $O(n^3)$ | Kubik |
| $O(2^n)$ | Eksponensial |
| $O(n!)$ | Faktorial |

### 3. Algoritma *Greedy*

*Greedy* secara harfiah memiliki arti tamak atau rakus. Selanjutnya, algoritma *greedy* merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Persoalan optimasi hanya ada dua macam yaitu maksimasi dan minimasi. Algoritma *greedy* membentuk solusi 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 harapan bahwa langkah sisanya mengarah ke solusi optimum global. Prinsip dari algoritma *greedy* menurut Ghozali, Setiawan, dan Furqon (2017) adalah mengambil sebanyak mungkin apa yang dapat diperoleh sekarang. Algoritma *greedy* merupakan metode yang sering digunakan untuk menyelesaikan *integer knapsack problem.* Pan dan Zhang (2018), menyelesaikan *integer knapsack problem* dengan algoritma *greedy* mempunyai kompleksitas waktu $O\left(n\log n\right)$. 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. Untuk memilih barang yang akan dimasukkan ke dalam *knapsack* terdapat beberapa strategi dari metode algoritma *greedy* dari Juwita, Susanto dan Halomoan (2017) adalah:

a) *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 *value*/ *profit*

barang. Kemudian barang-barang diambil satu persatu sampai kapasitas tempatnya penuh atau sudah tidak ada yang dapat dimasukkan lagi.

b) *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.

c) *Greedy by Density*

Setiap langkah di *knapsack problem* diisi dengan barang yang mempunyai $\frac{p_i}{w_i}$ dimana *p* adalah *value/ profit, w* adalah *weight* (berat) dan $i = (1, 2, 3, \ldots, n)$. 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 dan Pisinger, 2004).

Analisa contoh kasus pada *integer knapsack problem* menggunakan algoritma *greedy* untuk menentukan solusi optimumnya. Parameter *w* (berat), *p* (nilai/ keuntungan), dan *W* (kapasitas maksimum). Data yang sudah dicontohkan oleh Paryati (2009) awalnya diketahui:

$$w_1 = 6, w_2 = 5, w_3 = 10, w_4 = 5$$
$$p_1 = 12, p_2 = 15, p_3 = 50, p_4 = 10$$
$$W = 16$$

a) *Greedy by Profit*

Pertama kali yang dilakukan adalah mengurutkan secara menurun barang-barang berdasarkan *value/ profit*. Kemudian di ambil satu persatu barang sampai kapasitas maksimum tempatnya terpenuhi atau sudah tidak ada barang yang bisa dimasukkan lagi.

**Tabel 2.2.** *Greedy by Profit*

| $I$ | $w_i$ | $p_i$ | $p_i/w_i$ | Status |
|---|---|---|---|---|
| 3 | 10 | 50 | 5 | Diambil |
| 2 | 5 | 15 | 3 | Diambil |
| 1 | 6 | 12 | 2 | Tidak |
| 4 | 5 | 10 | 2 | Tidak |

b) *Greedy by Weight*

Pertama kali yang dilakukan adalah mengurutkan secara menaik barang-barang berdasarkan *weight/* beratnya. Kemudian diambil satu persatu barang sampai kapasitas maksimum tempatnya terpenuhi atau sudah tidak ada barang yang bisa dimasukkan lagi.

**Tabel 2.3.** *Greedy by Weight*

| $I$ | $w_i$ | $p_i$ | $p_i/w_i$ | Status |
|---|---|---|---|---|
| 2 | 5 | 15 | 3 | Diambil |
| 4 | 5 | 10 | 2 | Diambil |
| 1 | 6 | 12 | 2 | Diambil |
| 3 | 10 | 50 | 5 | Tidak |

c) *Greedy by Density*

Pertama kali yang dilakukan adalah mencari $\frac{p_i}{w_i}$ terbesar, kemudian diurutkan berdasarkan $\frac{p_i}{w_i}$ (*density*). Setelah itu, diambil satu persatu barang sampai kapasitas maksimum tempatnya terpenuhi atau sudah tidak ada barang yang bisa dimasukkan lagi.

**Tabel 2.4.** *Greedy by Density*

| $I$ | $w_i$ | $p_i$ | $p_i/w_i$ | Status |
|---|---|---|---|---|
| 3 | 10 | 50 | 5 | Diambil |
| 2 | 5 | 15 | 3 | Diambil |
| 4 | 5 | 10 | 2 | Tidak |
| 1 | 6 | 12 | 2 | Tidak |

Maka hasil akhir dari permasalahan di atas dan dianalisis menggunakan 3 metode dari algoritma *greedy* mendapatkan hasil data seperti di bawah ini:

**Tabel 2.5.** Hasil akhir *greedy*

| Barang | | | | *Greedy* | | |
|---|---|---|---|---|---|---|
| *I* | $W_i$ | $P_i$ | $p_i/w_i$ | *Profit* | *Weight* | *Density* |
| 1 | 6 | 12 | 2 | 0 | 1 | 0 |
| 2 | 5 | 15 | 3 | 1 | 1 | 1 |
| 3 | 10 | 50 | 5 | 1 | 0 | 1 |
| 4 | 5 | 10 | 2 | 0 | 1 | 0 |
| Total Bobot | | | | 15 | 16 | 15 |
| Total Keuntungan | | | | 65 | 37 | 65 |

Hasil akhir tersebut dapat disimpulkan bahwa total berat dan total keuntungan menggunakan algoritma *greedy* yang paling optimum sebesar 15 dan 65.

Memecahkan persoalan dengan algoritma *greedy* berdasarkan Juvianto dan Agung (2017), memerlukan elemen-elemen sebagai berikut :

a) Himpunan kandidat ($C$)

Himpunan kandidat berisi elemen-elemen pembentuk solusi. Pada setiap langkah, satu buah kandidat diambil dari himpunannya.

b) Himpunan solusi (*S*)

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

c) Fungsi seleksi

Fungsi seleksi adalah fungsi yang ada pada setiap langkah memilih kandidat yang paling memungkinkan guna mencapai solusi optimal. Kandidat yang sudah terpilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

d) Fungsi kelayakan (*Feasible*)

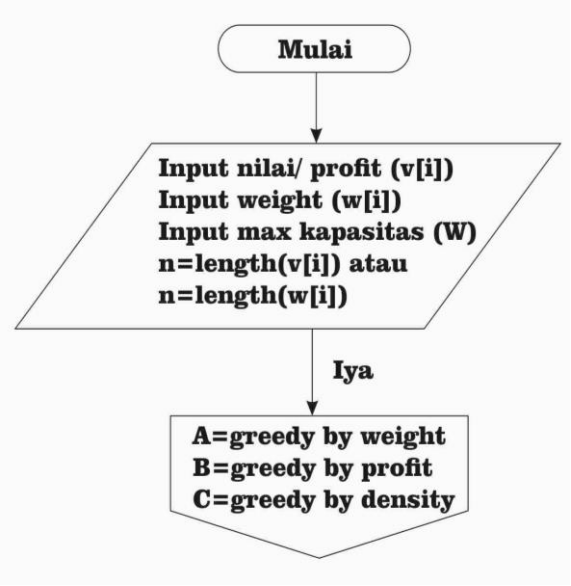
Fungsi kelayakan adalah fungsi yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak dan tidak melebihi batasan yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.

e) Fungsi objektif

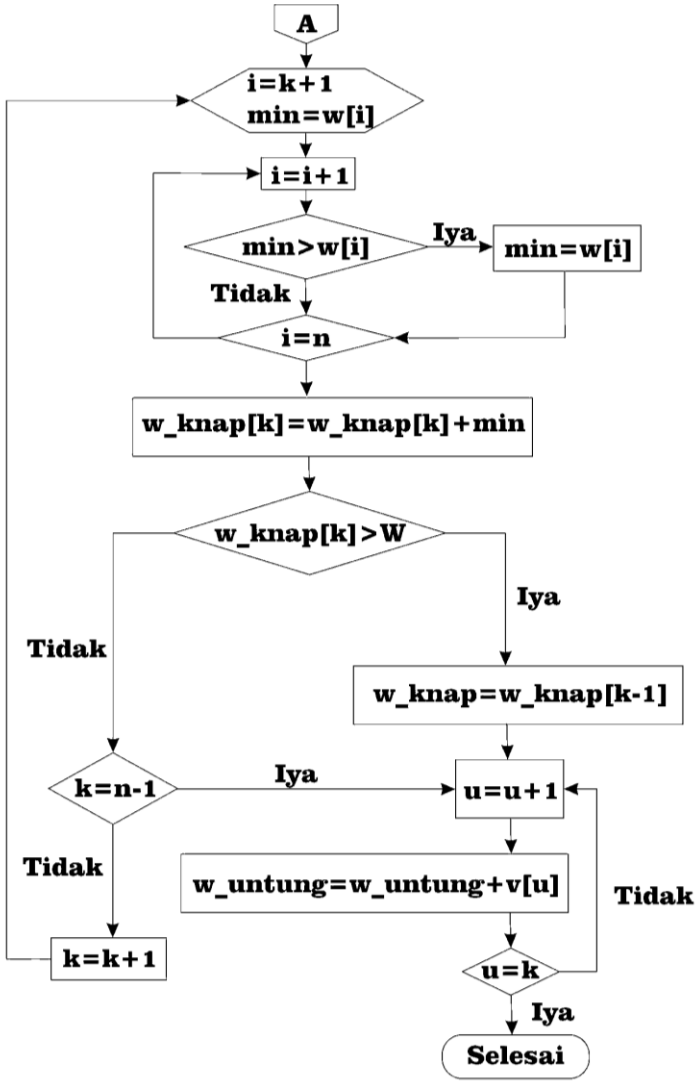
Fungsi objektif adalah fungsi yang memaksimalkan atau meminimalkan nilai solusi. Dengan kata lain, algoritma *greedy* melibatkan pencarian sebuah himpunan bagian S dari himpunan kandidat (C) yang dalam hal ini (S) harus memenuhi beberapa kriteria yang ditentukan, yaitu menyatakan solusi dan S dioptimasi oleh fungsi objektif.

Ada kalanya hasil algoritma *greedy* optimum tetapi pada sebagian masalah tidak selalu berhasil memberikan solusi yang benar benar optimum, tetapi algoritma *greedy* pasti memberikan solusi yang mendekati (*approximation*) nilai optimumnya.

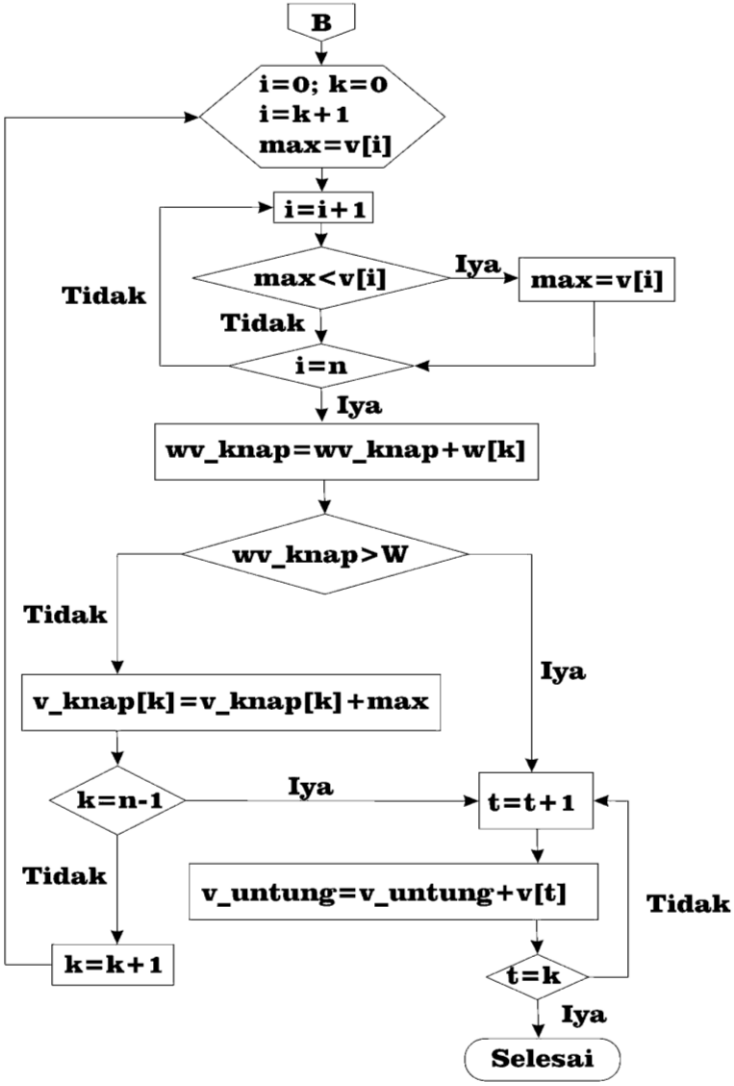
Adapun *flowchart* algoritma *greedy* untuk menyelesaikan masalah *integer knapsack* disajikan pada Gambar 2.1. – Gambar 2.4. sebagai berikut:



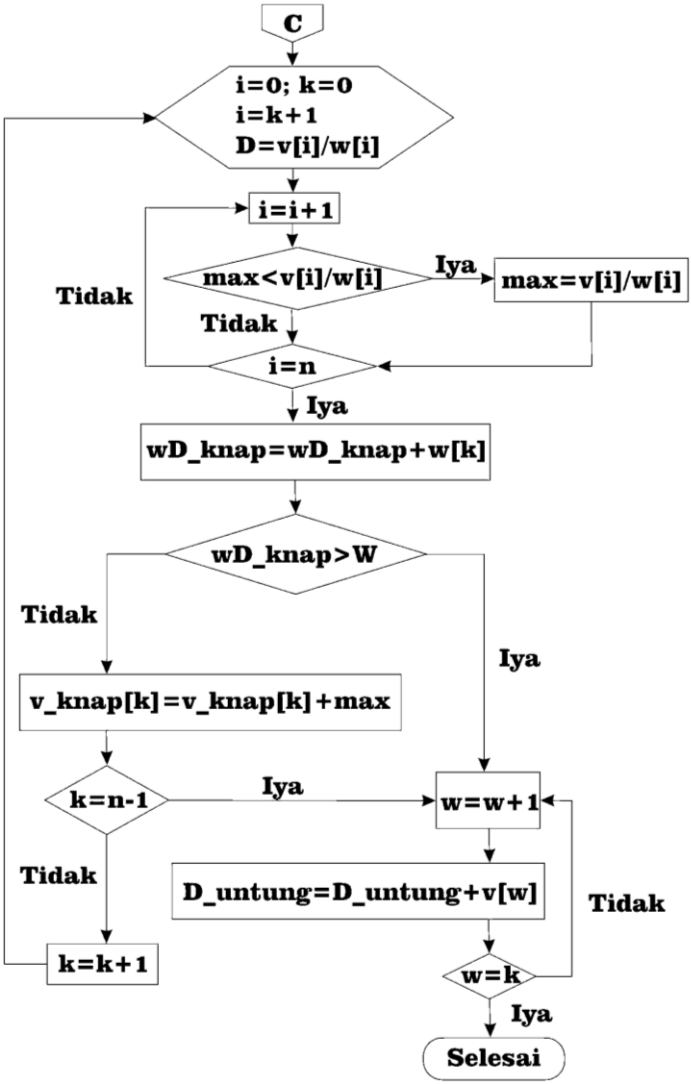

**Gambar 2.1.** *Flowchart* algoritma *greedy* input data awal




**Gambar 2.2.** *Flowchart* algoritma *greedy by weight*




**Gambar 2.3.** *Flowchart* algoritma *greedy by profit*




**Gambar 2.4.** *Flowchart* algoritma *greedy by density*

### 4. Algoritma *Dynamic Programming*

*Dynamic programming* merupakan penyelesaian masalah optimasi yang dikembangkan oleh Richar Bellman pada tahun 1952. 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 dan 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 dan 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* berdasarkan Kwarteng dan 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) :

$$M(i,j)=\begin{cases} 0, & \text{Jika } j=0 \quad (7) \\ M(i-1,j), & \text{Jika } j<w_i \quad (8) \\ \max(M(i-1,j),M(i-1,j-w_i)+p_i), & \text{Jika} \geq w_i \quad (9) \end{cases}$$

Berikut contoh dari Lu dan Ralp untuk masalah *integer knapsack* menggunakan metode algoritma *dynamic programming* yaitu *knapsack* dengan kapasitas maksimal $W=5$, ingin dimasukkan beberapa jenis barang $n=4$, masing-masing berat $w=(w_1,w_2,w_3,w_4)=(2,3,4,5)$ dan dengan *value* atau *profit* masing-masing dari barang $v=(v_1,v_2,v_3,v_4)=(3,4,5,6)$. Berapa keuntungan maksimal yang bisa didapat?

**Tabel 2.6**. Kondisi 1: $i=0, v[0]=0, w[0]=0$

| i/W | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | | | | | |
| 2 | 0 | | | | | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

for $w=0:W$ $V(0,w)=0$

for $i=1:n$ $V(i,0)$

**Tabel 2.7**. Kondisi 2: $i=1, v[1]=3, w[1]=2$

| i/W | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 3 | 3 | 3 | 3 |
| 2 | 0 | | | | | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

if $w_i \leq w$
  if $v_i + V(i-1,w-w_i) > V(i-1,w)$
$V(i,w)=v_i+V(i-1,w-w_i)$
else
$V(i,w)=V(i-1,w)$
else
$V(i,w)=V(i-1,w)$

**Tabel 2.8**. Kondisi 3: $i = 2, v[2] = 4, w[2] = 3$

| i/W | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 3 | 3 | 3 | 3 |
| 2 | 0 | 0 | 3 | 4 | 4 | 7 |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |

if $w_i \le w$
  if $v_i + V(i-1, w-w_i) > V(i-1, w)$
$V(i,w) = v_i + V(i-1, w-w_i)$
else
$V(i,w) = V(i-1, w)$
else
$V(i,w) = V(i-1, w)$

**Tabel 2.9**. Kondisi 4: $i = 3, v[3] = 5, w[3] = 4$

| i/W | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 3 | 3 | 3 | 3 |
| 2 | 0 | 0 | 3 | 4 | 4 | 7 |
| 3 | 0 | 0 | 3 | 4 | 5 | 7 |
| 4 | 0 | | | | | |

if $w_i \le w$
  if $v_i + V(i-1, w-w_i) > V(i-1, w)$
$V(i,w) = v_i + V(i-1, w-w_i)$
else
$V(i,w) = V(i-1, w)$
else
$V(i,w) = V(i-1, w)$

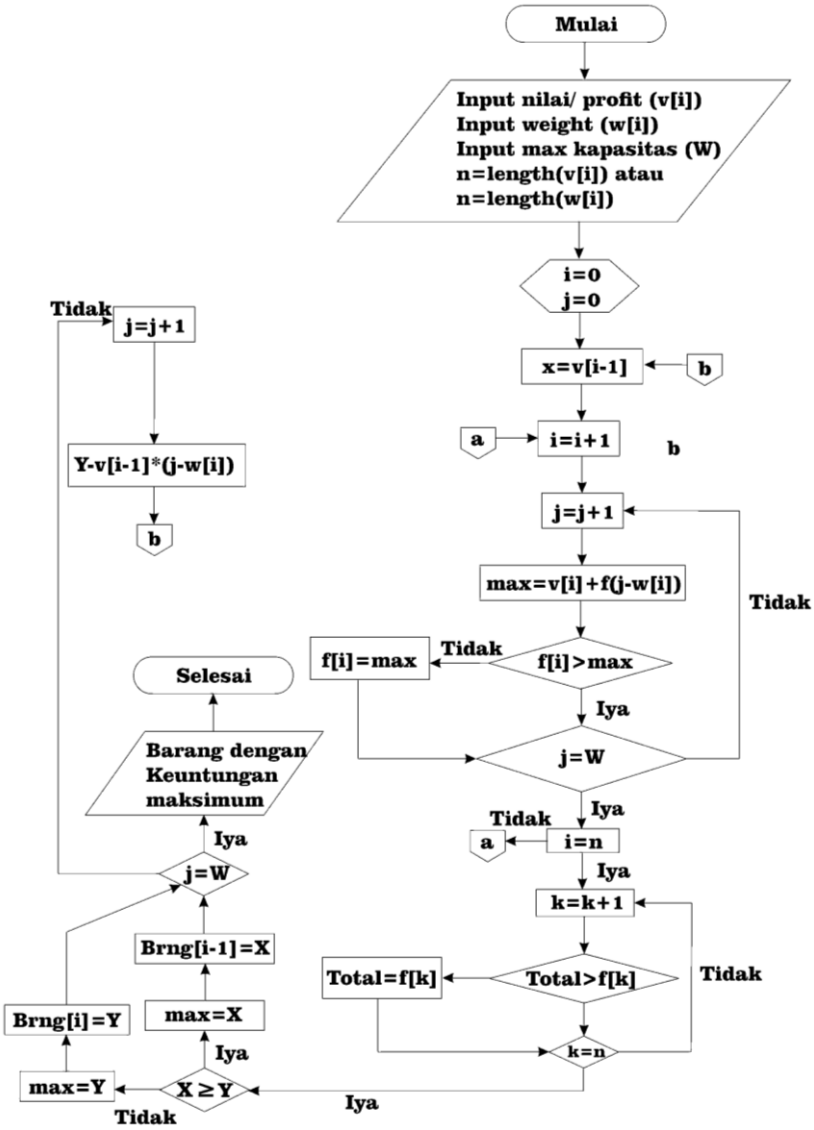
**Tabel 2.10.** Kondisi 5: $i = 4, v[4] = 6, w[4] = 5$

| i/W | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 3 | 3 | 3 | 3 |
| 2 | 0 | 0 | 3 | 4 | 4 | 7 |
| 3 | 0 | 0 | 3 | 4 | 5 | 7 |
| 4 | 0 | 0 | 4 | 4 | 5 | 7 |

if $w_i \le w$
  if $v_i + V(i-1, w-w_i) > V(i-1, w)$
$V(i,w) = v_i + V(i-1, w-w_i)$
else
$V(i,w) = V(i-1, w)$
else
$V(i,w) = V(i-1, w)$

Hasil akhir tersebut dapat disimpulkan bahwa total berat dan total keuntungan menggunakan algoritma *dynamic programming* yang paling optimum sebesar 5 dan 7.

Adapun *flowchart* algoritma *dynamic programming* untuk menyelesaikan masalah *integer knapsack* disajikan pada Gambar 2.5. sebagai berikut:




**Gambar 2.5.** *Flowchart* algoritma *dynamic programming*

**5. 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. Pan dan 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 *time limit exceeded* pada beberapa program yang ada batasan waktu kompilasi dan *runtime* program dengan kompleksitas waktunya $O\left(2^n\right)$.

Levitin (2012) menyatakan 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). Pendekatan *straightforward* pada kasus *knapsack*, membuat semua kombinasi barang yang mungkin dengan angka 1 atau 0 yang berarti diambil atau tidak untuk menentukan barang yang akan dimasukkan ke daftar barang untuk diambil atau tidak. 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.

Berikut contoh dari Levitin (2012) untuk *integer knapsack problem* menggunakan metode algoritma *brute force* yaitu *knapsack* dengan kapasitas maksimal $W = 10$, ingin dimasukkan beberapa jenis barang $n = 4$, masing-masing berat $w = (w_1, w_2, w_3, w_4) = (7,3,4,5)$ dengan keuntungan setiap barang $v = (v_1, v_2, v_3, v_4) = (42,12,40,25)$.

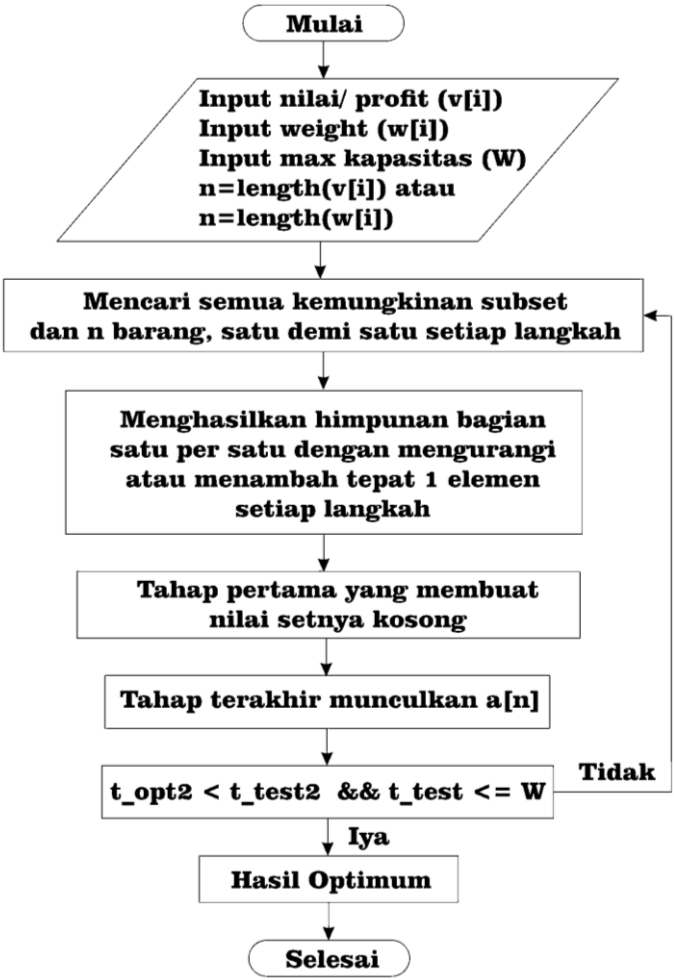
Berapa keuntungan maksimal yang bisa didapat?

**Tabel 2.11.** Hasil *brute force*

| subset | total *weight* | total *value* |
|---|---|---|
| $\varnothing$ | 0 | 0 |
| $\{1\}$ | 7 | 42 |
| $\{2\}$ | 3 | 12 |
| $\{3\}$ | 4 | 40 |
| $\{4\}$ | 5 | 25 |
| $\{1,2\}$ | 10 | 54 |
| $\{1,3\}$ | 11 | Tidak layak |
| $\{1,4\}$ | 12 | Tidak layak |
| $\{2,3\}$ | 7 | 52 |
| $\{2,4\}$ | 8 | 37 |
| **$\{3,4\}$** | **9** | **65** |
| $\{1,2,3\}$ | 14 | Tidak layak |
| $\{1,2,4\}$ | 15 | Tidak layak |
| $\{1,3,4\}$ | 16 | Tidak layak |
| $\{2,3,4\}$ | 12 | Tidak layak |
| $\{1,2,3,4\}$ | 19 | Tidak layak |

Hasil akhir tersebut dapat disimpulkan bahwa total berat dan total keuntungan menggunakan algoritma *brute force* yang paling optimum sebesar 9 dan 65.

Adapun *flowchart* algoritma *brute force* untuk menyelesaikan masalah *integer knapsack* disajikan pada Gambar 2.6. sebagi berikut:




**Gambar 2.6.** *Flowchart* algoritma *brute force*

## 6. Algoritma *Genetic*

Algoritma *genetic* menurut Messac (2015) merupakan keluarga 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* (Admi, Yun, dan Gen (2002), Admi dan Gen (2003a, 2003b)), *Vehicle Routing Problem* (VRP) ((Pankratz, 2004), (Laporte G. dan Semet., 1999)), *Knapsack Problem* (KP) (Gen dan Cheng, 2000), dan masih banyak lagi. Algoritma *genetic* berdasarkan Pan dan Zhang (2018) dalam penyelesaiannya tidak stabil dan tidak dapat menjamin solusi optimalnya dan kompleksitas waktu yang dihasilkan dalam waktu *polinomial* yaitu $O\left(n^2\right)$.

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*

Contoh untuk masalah *integer knapsack* menggunakan metode algoritma *genetic*: Terdapat *knapsack* dengan kapasitas maksimum $W = 12$, ingin dimasukkan beberapa jenis barang $n = 5$, memiliki berat $w = (w_1, w_2, w_3, w_4, w_5)$ bernilai $(8, 9, 2, 5, 2)$ dengan *value/ profit* $v = (v_1, v_2, v_3, v_4, v_5)$ bernilai $(7, 6, 8, 5, 6)$. Berapa keuntungan maksimal yang bisa didapatkan?

Perhitungan algoritma *genetic*, terlebih dahulu menginisialisasi parameter awal yang diperlukan, yaitu:

| | |
|---|---|
| Jumlah generasi | ( $jg = 1000$ ) |
| Jumlah populasi | ( $pop = 5$ ) |
| Probabilitas *crossover* | ( $pc = 0,25$ ) |
| Probabilitas mutasi | ( $pm = 0,01$ ) |

**Langkah 1: Populasi Awal**

Populasi awal berjumlah 5 buah dan dibangkitkan secara acak. Hasil dari 5 buah populasi awal seperti berikut:

$v_1 = [0\ 0\ 1\ 1\ 0]$

$v_2 = [1\ 1\ 0\ 0\ 1]$

$v_3 = [0\ 0\ 1\ 0\ 1]$

$v_4 = [0\ 1\ 1\ 1\ 1]$

$v_5 = [1\ 0\ 1\ 0\ 1]$

**Langkah 2: Evaluasi Kromosom**

Perhitungan nilai *fitness* dengan kapsitas *knapsack* $W = 12$.

**Tabel 2.12.** Berat dan keuntungan masing-masing barang

| Barang | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Nilai | 7 | 6 | 8 | 5 | 6 |
| Berat | 8 | 9 | 2 | 5 | 2 |

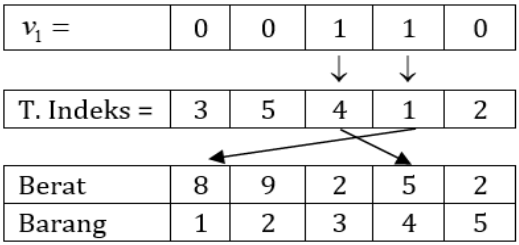
**Tabel 2.13.** Hasil perhitungan rasio nilai/berat adalah *array T*

| T. Indeks= | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| T= | $\frac{7}{8} = 0,875$ | $\frac{6}{9} = 0,667$ | $\frac{8}{2} = 4$ | $\frac{5}{5} = 1$ | $\frac{6}{2} = 3$ |

**Tabel 2.14.** Diurutkan secara *descending* berdasarkan *T*

| T. Indeks= | 3 | 5 | 4 | 1 | 2 |
|---|---|---|---|---|---|
| T= | $\frac{8}{2} = 4$ | $\frac{6}{2} = 3$ | $\frac{5}{5} = 1$ | $\frac{7}{8} = 0,875$ | $\frac{6}{9} = 0,667$ |

a. Kromosom $v_1 = [0\ 0\ 1\ 1\ 0]$

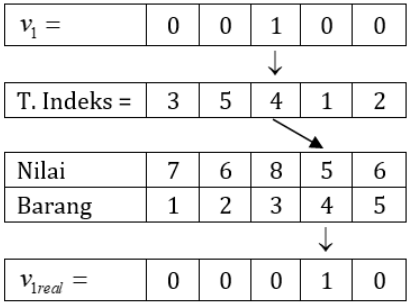



**Gambar 2.7.** Ilustrasi arti dari kromosom $v_1$

Masukan secara berurutan barang berdasarkan T. Indeks ke dalam *knapsack*:

1) Barang 4 masuk ke dalam *knapsack* = 5 kg
2) Barang 1 masuk ke dalam *knapsack* = 8 kg, jadi total 5 + 8 = 13 kg. Karena 13 kg melebihi kapasitas *knapsack* maka barang 1 dikeluarkan dari *knapsack,* maka *knapsack* = 5 kg

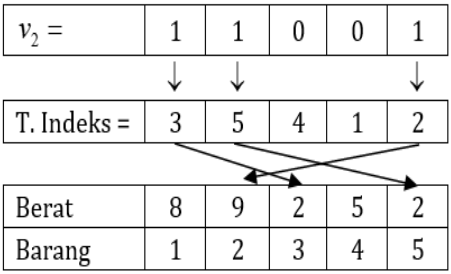
Barang terpilih untuk dimasukkan dalam *knapsack* adalah barang 4.




**Gambar 2.8.** Ilustrasi perhitungan nilai *fitness* dari $v_1$

Jadi, *fitness* dari $v_1 = 5$ dengan $v_{1real} = [0\ 0\ 0\ 1\ 0]$

b. Kromosom $v_2 = [1\ 1\ 0\ 0\ 1]$

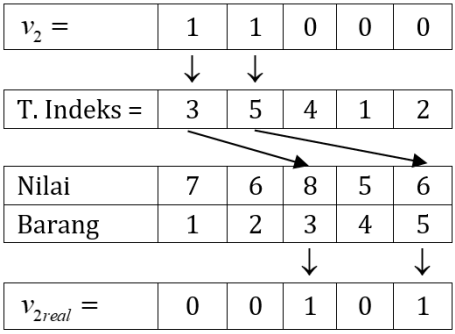



**Gambar 2.9.** Ilustrasi arti dari kromosom $v_2$

Masukan secara berurutan barang berdasarkan T. Indeks ke dalam *knapsack*:

1) Barang 3 masuk ke dalam *knapsack* = 2 kg
2) Barang 5 masuk ke dalam *knapsack* = 2 kg, jadi total 2 + 2 = 4 kg.
3) Barang 2 masuk ke dalam *knapsack* = 9 kg, jadi total 4 + 9 = 13. Karena 13 kg melebihi kapasitas *knapsack* maka barang 2 dikeluarkan dari *knapsack,* maka *knapsack* = 4 kg

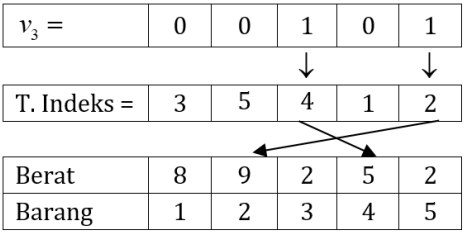
Barang terpilih untuk dimasukkan dalam *knapsack* adalah barang 3 dan 5.




**Gambar 2.10.** Ilustrasi perhitungan nilai *fitness* $v_2$

Jadi, *fitness* dari $v_2 = 8+6 = 14$ dengan $v_{2real} = [0\ 0\ 1\ 0\ 1]$

c. Kromosom $v_3 = [0\ 0\ 1\ 0\ 1]$



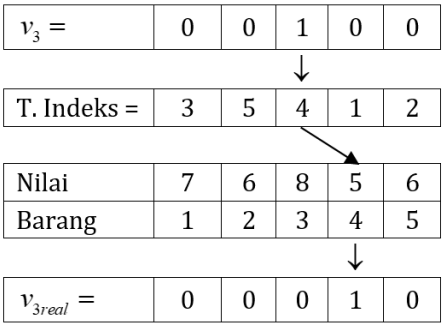

**Gambar 2.11.** Ilustrasi arti dari kromosom $v_3$

Masukan secara berurutan barang berdasarkan T. Indeks ke dalam *knapsack*:

1) Barang 4 masuk ke dalam *knapsack* = 5 kg
2) Barang 2 masuk ke dalam *knapsack* = 9 kg, jadi total 5 + 9 = 14 kg. Karena 14 kg melebihi

kapasitas *knapsack* maka barang 2 dikeluarkan dari *knapsack,* maka *knapsack* = 5 kg

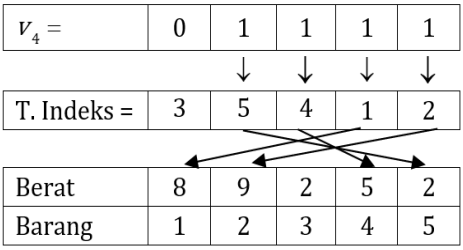
Barang terpilih untuk dimasukkan dalam *knapsack* adalah barang 4.




**Gambar 2.12.** Ilustrasi perhitungan nilai *fitness* $v_3$

Jadi, *fitness* dari $v_3 = 5$ dengan $v_{3real} = [0\ 0\ 0\ 1\ 0]$

d. Kromosom $v_4 = [0\ 1\ 1\ 1\ 1]$



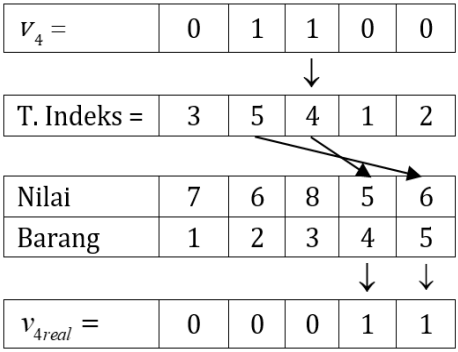

**Gambar 2.13.** Ilustrasi arti dari kromosom $v_4$

Masukan secara berurutan barang berdasarkan T. Indeks ke dalam *knapsack*:

1) Barang 5 masuk ke dalam *knapsack* = 2 kg

2) Barang 4 masuk ke dalam *knapsack* = 5 kg, jadi total 2 + 5 = 7 kg.
3) Barang 1 masuk ke dalam *knapsack* = 8 kg, jadi total 7 + 8 = 15 kg. Karena 15 kg melebihi kapasitas *knapsack* maka barang 1 dikeluarkan dari *knapsack,* maka *knapsack* = 7 kg
4) Barang 2 masuk ke dalam *knapsack* = 9 kg, jadi total 7 + 9 = 16 kg. Karena 16 kg melebihi kapasitas *knapsack* maka barang 2 dikeluarkan dari *knapsack,* maka *knapsack* = 7 kg

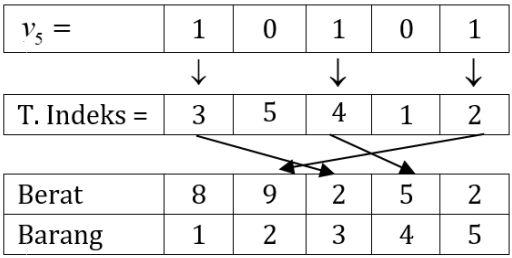
Barang terpilih untuk dimasukkan dalam *knapsack* adalah barang 5 dan 4.




**Gambar 2.14.** Ilustrasi perhitungan nilai *fitness* $v_4$

Jadi, *fitness* dari $v_4 = 2+5=7$ dengan $v_{4real} = \left[0\ 0\ 0\ 1\ 1\right]$

e. Kromosom $v_5 = [1\ 0\ 1\ 0\ 1]$

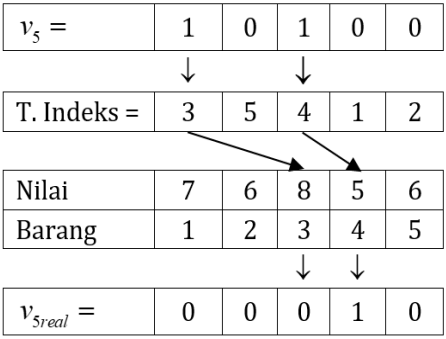



**Gambar 2.15.** Ilustrasi arti dari kromosom $v_5$

Masukan secara berurutan barang berdasarkan T. Indeks ke dalam *knapsack*:

1) Barang 3 masuk ke dalam *knapsack* = 2 kg
2) Barang 4 masuk ke dalam *knapsack* = 5 kg, jadi total 2 + 5 = 7 kg.
3) Barang 2 masuk ke dalam *knapsack* = 9 kg, jadi total 7 + 9 = 16 kg. Karena 16 kg melebihi kapasitas *knapsack* maka barang 2 dikeluarkan dari *knapsack,* maka *knapsack* = 7 kg

Barang terpilih untuk dimasukkan dalam *knapsack* adalah barang 3 dan 4.




**Gambar 2.16.** Ilustrasi perhitungan nilai *fitness* $v_5$

Jadi, *fitness* dari $v_5 = 8+5 = 13$ dengan $v_{5real} = \left[0\ 0\ 0\ 1\ 0\right]$

**Langkah 3: *Crossover* (kawin silang)**

Populasi baru hasil seleksi akan dikawin silangkan. Misalkan probabilitas *crossover* yang digunakan adalah $pc = 0,25$. Misalkan didapat urutan angka acak untuk setiap kromosom sebagai berikut:

**Tabel 2.15.** Urutan angka acak untuk setiap kromosom

| Kromosom | Nilai acak (*r*) | Kromosom terpilih |
|---|---|---|
| 1 | 0,1512 | $v_1$ |
| 2 | 0,9941 | |
| 3 | 0,9501 | |
| 4 | 0,9712 | |
| 5 | 0,2456 | $v_5$ |

Kromosom yang terpilih sebagai *parent* yang akan dikawin silangkan adalah hanya $v_1$ dan $v_5$. Selanjutnya akan dibangkitkan bilangan acak yang lain untuk memilih titik potong. Pada kasus contoh ini dibangkitkan bilangan acak (bilangan bulat dari 1 sampai 5). Anggap hasil bilangan acak yang dibangkitkan bernilai 3, maka

$$v_1 = [0\ 0\ \mathbf{1\ 1\ 0}]$$

$$v_5 = [0\ 0\ \mathbf{1\ 0\ 1}]$$

Hasil akhir kawin silang sebagai berikut:

$$v_1 = [0\ 0\ \mathbf{1\ 0\ 1}]$$
$$v_2 = [1\ 1\ 0\ 0\ 1]$$
$$v_3 = [0\ 0\ 1\ 0\ 1]$$
$$v_4 = [0\ 1\ 1\ 1\ 1]$$
$$v_5 = [1\ 0\ \mathbf{1\ 1\ 0}]$$

**Langkah 4: Mutasi**

Allele-allele yang digunakan dalam kromosom adalah hanya 1 dan 0.

**Tabel 2.16.** Hasil pengecakan allele-allele

| Allele | Pengacakan |
|---|---|
| 0 | 1 |
| 1 | 0 |

Nilai probabilitas mutasi $pm = 0,01$. Jumlah gen yang terdapat dalam satu generasi adalah $m \times pop\_size = 5 \times 5 = 25$ gen. Setiap gen memiliki peluang yang sama untuk mengalami mutasi, akan dibangkitkan bilangan acak yang mengalami kisaran $[0,1]$ sebanyak 25 buah. Gen yang mempunyai bilangan acak dimana memiliki nilai yang lebih kecil dari $pm = 0,01$ akan terpilih untuk mengalami mutasi.

**Tabel 2.17.** Misalkan gen yang terpilih untuk dimutasi

| Posisi gen | Kromosom | Gen | Bil.acak | Gen sebelum mutasi | Gen setelah mutasi |
|---|---|---|---|---|---|
| 21 | 5 | 1 | 0,0029 | 1 | 0 |

Hasil akhir mutasi sebagai berikut:

$v_1 = [0\ 0\ 1\ 0\ 1]$

$v_2 = [1\ 1\ 0\ 0\ 1]$

$v_3 = [0\ 0\ 1\ 0\ 1]$

$v_4 = [0\ 1\ 0\ 1\ 1]$

$v_5 = [0\ 0\ 1\ 1\ 0]$

**Langkah 5: Seleksi kromosom**

1. **Langkah 1**

Perhitungan nilai *fitness* dengan kapasitas *knapsack* $W = 12$.

**Tabel 2.18.** Berat dan keuntungan masing-masing barang

| Barang | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Nilai | 7 | 6 | 8 | 5 | 6 |
| Berat | 8 | 9 | 2 | 5 | 2 |

**Tabel 2.19.** Hasil perhitungan rasio nilai/berat adalah *array T*

| T. Indeks= | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| T= | $\frac{7}{8}=0,875$ | $\frac{6}{9}=0,667$ | $\frac{8}{2}=4$ | $\frac{5}{5}=1$ | $\frac{6}{2}=3$ |

**Tabel 2.20.** Diurutkan secara *descending* berdasarkan *T*

| T. Indeks= | 3 | 5 | 4 | 1 | 2 |
|---|---|---|---|---|---|
| T= | $\frac{8}{2}=4$ | $\frac{6}{2}=3$ | $\frac{5}{5}=1$ | $\frac{7}{8}=0,875$ | $\frac{6}{9}=0,667$ |

a. Kromosom $v_1 = \left[0\ 0\ 1\ 0\ 1\right]$

| $v_1$ = | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|
| | | | ↓ | | ↓ |
| T. Indeks = | 3 | 5 | 4 | 1 | 2 |
| | | | ↙ | | ↙ |
| Berat | 8 | 9 | 2 | 5 | 2 |
| Barang | 1 | 2 | 3 | 4 | 5 |

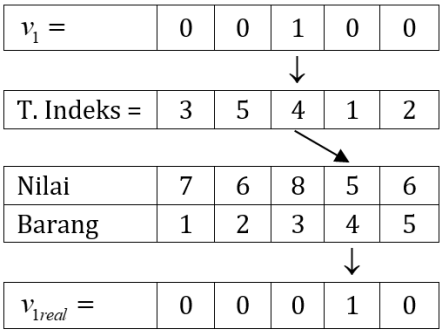
**Gambar 2.17.** Ilustrasi arti dari kromosom $v_1$

Masukan secara berurutan barang berdasarkan T. Indeks ke dalam *knapsack*:

1) Barang 4 masuk ke dalam *knapsack* = 5 kg

2) Barang 2 masuk ke dalam *knapsack* = 9 kg, jadi total 5 + 9 = 14 kg. Karena 14 kg melebihi kapasitas *knapsack* maka barang 2 dikeluarkan dari *knapsack,* maka *knapsack* = 5 kg

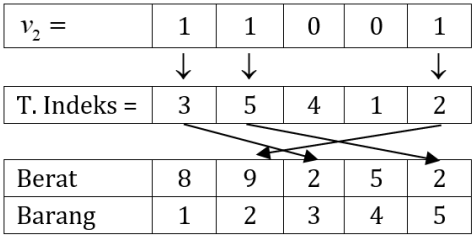
Barang terpilih untuk dimasukkan dalam *knapsack* adalah barang 4.




**Gambar 2.18.** Ilustrasi perhitungan nilai *fitness* dari $v_1$

Jadi, *fitness* dari $v_1 = 5$ dengan $v_{1real} = [0\ 0\ 0\ 1\ 0]$

b. Kromosom $v_2 = [1\ 1\ 0\ 0\ 1]$

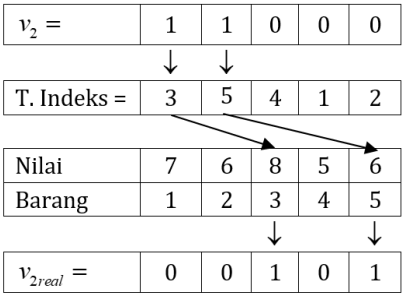



**Gambar 2.19.** Ilustrasi arti dari kromosom $v_2$

Masukan secara berurutan barang berdasarkan T. Indeks ke dalam *knapsack*:

1) Barang 3 masuk ke dalam *knapsack* = 2 kg
2) Barang 5 masuk ke dalam *knapsack* = 2 kg, jadi total 2 + 2 = 4 kg.
3) Barang 2 masuk ke dalam *knapsack* = 9 kg, jadi total 4 + 9 = 13 kg. Karena 13 kg melebihi kapasitas *knapsack* maka barang 2 dikeluarkan dari *knapsack,* maka *knapsack* = 4 kg

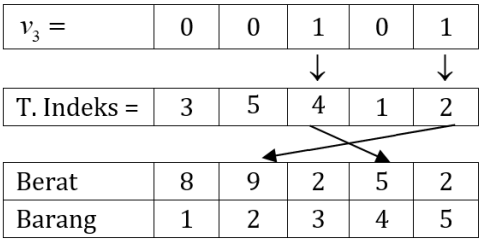
Barang terpilih untuk dimasukkan dalam *knapsack* adalah barang 3 dan 5.




**Gambar 2.20.** Ilustrasi perhitungan nilai *fitness* $v_2$

Jadi, *fitness* dari $v_2 = 8 + 6 = 14$ dengan $v_{2real} = [0\ 0\ 1\ 0\ 1]$

c. Kromosom $v_3 = [0\ 0\ 1\ 0\ 1]$

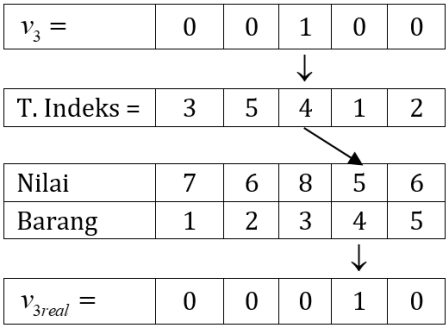



**Gambar 2.21.** Ilustrasi arti dari kromosom $v_3$

Masukan secara berurutan barang berdasarkan T. Indeks ke dalam *knapsack*:

1) Barang 4 masuk ke dalam *knapsack* = 5 kg
2) Barang 2 masuk ke dalam *knapsack* = 9 kg, jadi total 5 + 9 = 14 kg. Karena 14 kg melebihi kapasitas *knapsack* maka barang 2 dikeluarkan dari *knapsack,* maka *knapsack* = 5 kg

Barang terpilih untuk dimasukkan dalam *knapsack* adalah barang 4.




**Gambar 2.22.** Ilustrasi perhitungan nilai *fitness* $v_3$

Jadi, *fitness* dari $v_3 = 5$ dengan $v_{3real} = [0\ 0\ 0\ 1\ 0]$

d. Kromosom $v_4 = [0\ 1\ 1\ 1\ 1]$

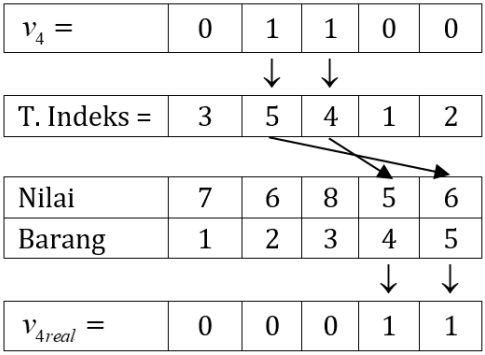
| $v_4$ = | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|
| | | ↓ | ↓ | ↓ | ↓ |
| T. Indeks = | 3 | 5 | 4 | 1 | 2 |
| Berat | 8 | 9 | 2 | 5 | 2 |
| Barang | 1 | 2 | 3 | 4 | 5 |

**Gambar 2.23.** Ilustrasi arti dari kromosom $v_4$

Masukan secara berurutan barang berdasarkan T. Indeks ke dalam *knapsack*:

1) Barang 5 masuk ke dalam *knapsack* = 2 kg
2) Barang 4 masuk ke dalam *knapsack* = 5 kg, jadi total 2 + 5 =7 kg
3) Barang 1 masuk ke dalam *knapsack* = 8 kg, jadi total 7 + 8 = 15 kg. Karena 15 kg melebihi kapasitas *knapsack* maka barang 1 dikeluarkan dari *knapsack,* maka *knapsack* = 7 kg
4) Barang 2 masuk ke dalam *knapsack* = 9 kg, jadi total 7 + 9 = 16 kg. Karena 16 kg melebihi kapasitas *knapsack* maka barang 2 dikeluarkan dari *knapsack,* maka *knapsack* = 7 kg

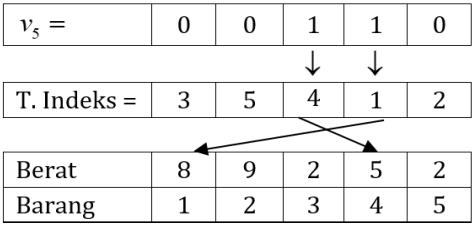
Barang terpilih untuk dimasukkan dalam *knapsack* adalah barang 5 dan 4.




**Gambar 2.24.** Ilustrasi perhitungan nilai *fitness* $v_4$

Jadi, *fitness* dari $v_4 = 5+6 = 11$ dengan $v_{4real} = \left[0\ 0\ 0\ 1\ 1\right]$

e. Kromosom $v_5 = \left[0\ 0\ 1\ 1\ 0\right]$




**Gambar 2.25.** Ilustrasi arti dari kromosom $v_5$

Masukan secara berurutan barang berdasarkan T. Indeks ke dalam *knapsack*:

1) Barang 4 masuk ke dalam *knapsack* = 5 kg

2) Barang 1 masuk ke dalam *knapsack* = 8 kg, jadi total 5 + 8 = 13 kg. Karena 13 kg melebihi kapasitas *knapsack* maka barang 1 dikeluarkan dari *knapsack,* maka *knapsack* = 5 kg

Barang terpilih untuk dimasukkan dalam *knapsack* adalah barang 4.

| $v_5 =$ | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|
| T. Indeks = | 3 | 5 | 4 | 1 | 2 |
| Nilai | 7 | 6 | 8 | 5 | 6 |
| Barang | 1 | 2 | 3 | 4 | 5 |
| $v_{5real} =$ | 0 | 0 | 0 | 1 | 0 |

**Gambar 2.26.** Ilustrasi perhitungan nilai *fitness* $v_5$

Jadi, *fitness* dari $v_5 = 5$ dengan $v_{5real} = \left[0\ 0\ 0\ 1\ 0\right]$

## 2. Langkah 2

Mencari nilai *fitness* optimal yang sudah didapatkan pada langkah sebelumnya, maka perhitungan nilai *fitness* $eval\left(v_k\right)$:

$$eval\left(v_1\right) = 5$$
$$eval\left(v_2\right) = 14$$
$$eval\left(v_3\right) = 5$$
$$eval\left(v_4\right) = 11$$
$$eval\left(v_5\right) = 5$$

**Elitisme**

Kromosom $v_2$ memiliki *fitness* tertinggi, oleh karena itu kromosom $v_2$ digandakan dan hasil penggandaannya disimpan dalam $v_1$:

$$v_1 = \left[1\ 1\ 0\ 0\ 1\right]\ \left(v_2\right)$$

**3. Langkah 3**

Perhitungan total nilai *fitness* untuk populasi menggunakan solusi *fitness* yang sudah didapatkan:

$$F = \left(5+14+5+11+5\right) = 40$$

**4. Langkah 4**

Perhitungan nilai probabilitas seleksi $p_k$ untuk setiap kromosom $v_k$:

$$p_1 = \frac{5}{40} = 0{,}125$$

$$p_2 = \frac{14}{40} = 0{,}35$$

$$p_3 = \frac{5}{40} = 0{,}125$$

$$p_4 = \frac{11}{40} = 0{,}275$$

$$p_5 = \frac{5}{40} = 0{,}125$$

**5. Langkah 5**

Perhitungan nilai probabilitas kumulatif $q_k$ untuk setiap kromosom $v_k$ :

$q_1 = 0{,}125$
$q_2 = 0{,}125 + 0{,}35 = 0{,}475$
$q_3 = 0{,}125 + 0{,}35 + 0{,}125 = 0{,}6$
$q_4 = 0{,}125 + 0{,}35 + 0{,}125 + 0{,}275 = 0{,}875$
$q_5 = 0{,}125 + 0{,}35 + 0{,}125 + 0{,}275 + 0{,}125 = 1$

Selanjutnya memutar roda roulette, karena digunakan metode elitism dalam proses algoritma *genetic* maka putaran roda dilakukan sebanyak $pop - 1$ yaitu 4 kali. Proses memutar roda *roulette* identic dengan membangkitkan sebuah nilai acak $r_k$ yang memiliki nilai kisaran $[0,1]$. Apabila $q_{k-1} \le r_k \le q_k$ , maka pilih kromosom ke-*k* sebagai induk. Misalkan setelah 4 kali memutar roda *roulette*, didapat hasil sebagai berikut:

**Tabel 2.21.** Misalkan hasil setelah 4 kali memutar roda *roulette*

| Putaran | Nilai acak $(r)$ | $q_{k-1}$ | $q_k$ | Kromosom terpilih |
|---|---|---|---|---|
| 1 | 0,416 | $q_1$ | $q_2$ | $v_2$ |
| 2 | 0,891 | $q_4$ | $q_5$ | $v_5$ |
| 3 | 0,966 | $q_4$ | $q_5$ | $v_5$ |
| 4 | 0,616 | $q_3$ | $q_4$ | $v_4$ |

Hasil akhir seleksi sebagai berikut:

$$v_1 = \left[1\ 1\ 0\ 0\ 1\right] \left(\text{hasil elitisme}\right)$$
$$v_2 = \left[1\ 1\ 0\ 0\ 1\right] \left(v_2\right)$$
$$v_3 = \left[0\ 0\ 1\ 1\ 0\right] \left(v_5\right)$$
$$v_4 = \left[0\ 0\ 1\ 1\ 0\right] \left(v_5\right)$$
$$v_5 = \left[0\ 1\ 1\ 1\ 1\right] \left(v_4\right)$$

Dengan $v_{real}$ hasil seleksi sebagai berikut

$$v_{1real} = \left[0\ 0\ 1\ 0\ 1\right] \left(\text{hasil elitisme}\right)$$
$$v_{2real} = \left[0\ 0\ 1\ 0\ 1\right] \left(v_{2real}\right)$$
$$v_{3real} = \left[0\ 0\ 0\ 1\ 0\right] \left(v_{5real}\right)$$
$$v_{4real} = \left[0\ 0\ 0\ 1\ 0\right] \left(v_{5real}\right)$$
$$v_{5real} = \left[0\ 0\ 0\ 1\ 1\right] \left(v_{4real}\right)$$

Proses seleksi setelah selesai maka selesailah perhitungan untuk 1 generasi. Proses selanjutnya adalah mengulangi lagi proses pada langkah 2 sampai proses pengecekan kondisi berhenti terpenuhi dengan menggunakan kromosom: $v_1, v_2, v_3, v_4,$ dan $v_5$.

**Langkah 8: Decoding**

*Decoding* adalah proses mendekodekan gen-gen dalam suatu kromosom agar nilainya kembali seperti semula (sebelum dilakukan *encoding*), untuk contoh perhitungan di

atas didapat hasil *decoding* yang dilakukan terhadap $v_{real}$ sebagai berikut:

**Tabel 2.22.** Hasil *decoding* yang dilakukan terhadap $v_{real}$

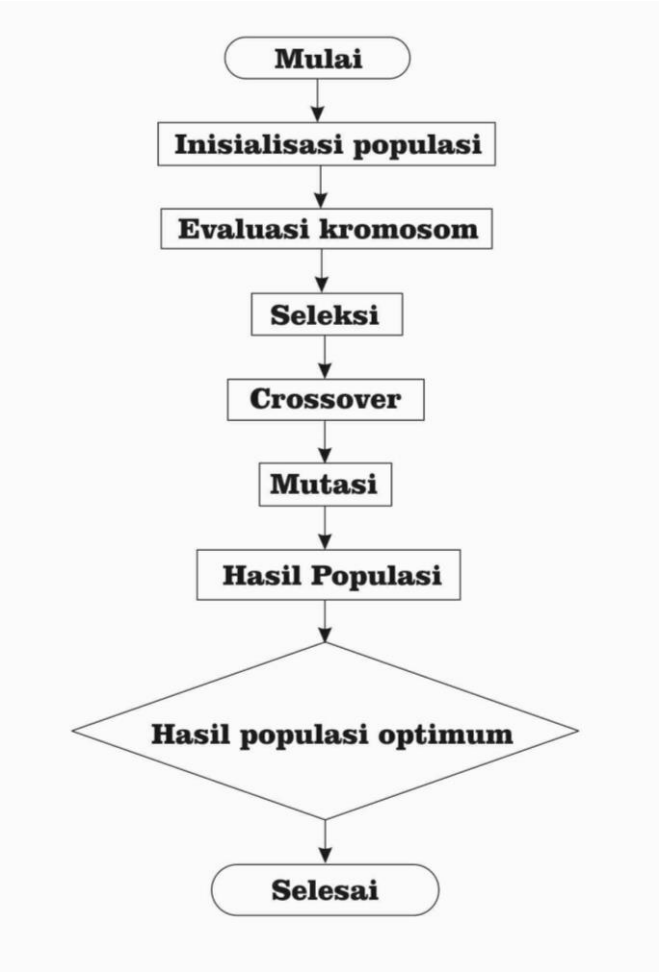
| Kromosom | Barang terpilih |
|---|---|
| $v_{1real} = [0\ 0\ 1\ 0\ 1]$ | Barang 3 dan 5 |
| $v_{2real} = [0\ 0\ 1\ 0\ 1]$ | Barang 3 dan 5 |
| $v_{3real} = [0\ 0\ 0\ 1\ 0]$ | Barang 4 |
| $v_{4real} = [0\ 0\ 0\ 1\ 0]$ | Barang 4 |
| $v_{5real} = [0\ 0\ 0\ 1\ 1]$ | Barang 4 dan 5 |

Hasil akhir tersebut dapat disimpulkan bahwa total berat dan total keuntungan menggunakan algoritma *genetic* yang dihasilkan adalah:

1. Total berat dan keuntungannya sebanyak 4 dan 14.
2. Total berat dan keuntungannya sebanyak 5 dan 5.
3. Total berat dan keuntungannya sebanyak 7 dan 11.

Jadi dapat disimpulkan dari beberapa hasil yang diperoleh menggunakan algoritma *genetic* bahwa total berat dan keuntungan yang paling optimal adalah 4 dan 14, karena memiliki keuntungan yang terbesar.

Adapun *flowchart* algoritma *genetic* untuk menyelesaikan masalah *integer knapsack* disajikan pada Gambar 2.7. sebagi berikut:




**Gambar 2.7.** *Flowchart* algoritma *genetic*

## B. Kajian Pustaka

Adapun penelitian yang relevan tentang *knapsack problem* adalah yang pertama, penelitian *knapsack problem* khususnya *integer knapsack problem* pernah dilakukan oleh Thada dan Dhaka (2014) dan disimpulkan bahwa algoritma genetika dapat memberikan hasil optimal pada *knapsack problem* dengan *software* MATLAB: GA *optimization tool*. Kedua, perbandingan algoritma pernah dilakukan yaitu penerapan algoritma *dynamic programming* dan *greedy* pada permasalahan *integer knapsack problem* (Sampurno, Sugiharti dan Alamsyah, 2018). Penelitian tersebut dapat disimpulkan bahwa algortima *dynamic programming* lebih efisien dalam perhitungan daripada algoritma *greedy.* Ketiga, penelitian oleh Pan dan Zhang (2018) tentang perbandingan algoritma untuk masalah *integer knapsack.* Penelitian tersebut disimpulkan bahwa algoritma *greedy* untuk menyelesaikan permasalahan *integer knapsack* paling cepat dibandingkan dengan algoritma yang lain dan algoritma *dynamic programming* paling optimum untuk memperoleh solusi *integer knapsack problem.* Dan masih banyak lagi penelitian terkait *integer knapsack problem*.

# BAB III

# METODE PENELITIAN

## A. Jenis Penelitian

Jenis penelitian yang digunakan dalam penyusunan skripsi ini adalah studi kasus. Studi kasus berasal dari terjemahan dalam bahasa Inggris yaitu "*A case study*" atau *Case studies*". Menurut kamus *Oxford Advanced Learner's Dictionary of Current Engglish* yang dikutip dari Prof.Dr.H. Mudjia Rahardjo, M.Si (2017) bahwa kata "kasus" yang diambil dari kata "*case*" diartikan sebagai contoh kejadian sesuatu, kondisi aktual dari keadaan atau situasi, dan lingkungan atau kondisi tertentu tentang orang atau sesuatu. Definisi tersebut dapat disimpulkan bahwa studi kasus adalah rangkaian kegiatan ilmiah yang dilakukan secara intensif dan mendalam tentang suatu program, sekelompok orang, lembaga atau organisasi untuk memperoleh pengetahuan tentang hal atau pristiwa tersebut. Selanjutnya, studi kasus di lapangan dalam penelitian ini bertujuan untuk mengumpulkan informasi yaitu mengumpulkan data nama barang, berat dan harga setiap barang di jasa pengiriman

barang dengan paket Reguler. Selanjutnya, menyelesaikan *integer knapsack* problem menggunakan 4 algoritma yaitu algoritma *greedy* (*greedy by profit, greedy by weight, greedy by desnsity*), algoritma *dynamic programming,* algoritma *brute force*, dan algoritma *genetic.*

## B. Tempat dan Waktu Penelitian

Tempat penelitian adalah di J&T *drop point* area Ngaliyan Kota Semarang. Adapun waktu penelitian pada bulan Januari 2019.

## C. Prosedur Penelitian

Adapun prosedur penelitian yang digunakan dalam mencapai tujuan penelitian yang digunakan adalah bagaimana implementasi algoritma *greedy*, *dynamic programming, brute force*, dan *genetic* dalam menyelesaikan kasus *integer knapsack problem*. Adapun langkah-langkah sebagai berikut:

1. Studi Literatur

   Langkah awal pada penelitian ini adalah melakukan studi dari berbagai literatur mengenai algoritma *greedy*, algoritma *dynamic programming*, algoritma *brute force*, algoritma *genetic.*

2. Pengumpulan Data

Data diperoleh dari wawancara dengan salah satu J&T yang ada di *drop point* area Ngaliyan kota Semarang. Mengumpulkan data pengiriman barang yang akan didistribusikan.

a) Menentukan nilai *profit* dengan mempertimbangkan jarak lokasi pengiriman.
b) Membuat tabel daftar barang beserta berat dan *profit* setiap barang.
c) Menentukan kapasitas maksimum *knapsack* yang digunakan untuk mendistribusikan barang.

3. Menggunakan konsep *greedy by profit* dalam menyelesaikan kasus *knapsack problem.*
   a) Mencari nilai keuntungan (*profit*) dari tiap-tiap barang.
   b) Barang-barang tersebut diurutkan berdasarkan *profit*-nya.
   c) Mengambil satu persatu barang berdasarkan urutan yang dapat ditampung oleh *knapsack* sampai *knapsack* penuh atau sudah tidak ada barang lagi.
4. Menggunakan konsep *greedy by weight* dalam menyelesaikan kasus *knapsack problem.*
   a) Mencari nilai berat (*weight)* dari tiap-tiap barang.

b) Barang-barang tersebut diurutkan berdasarkan *weight.*

c) Mengambil satu persatu barang berdasarkan urutan yang dapat ditampung oleh *knapsack* sampai *knapsack* penuh atau sudah tidak ada barang lagi.

5. Menggunakan konsep *greedy by density* dalam menyelesaikan kasus *knapsack problem.*

a) Mencari nilai keuntungan (*profit)* per berat (*weight*) dari tiap-tiap barang.

b) Barang-barang tersebut diurutkan berdasarkan *density.*

c) Mengambil satu persatu barang berdasarkan urutan yang dapat ditampung oleh *knapsack* sampai *knapsack* penuh atau sudah tidak ada barang lagi.

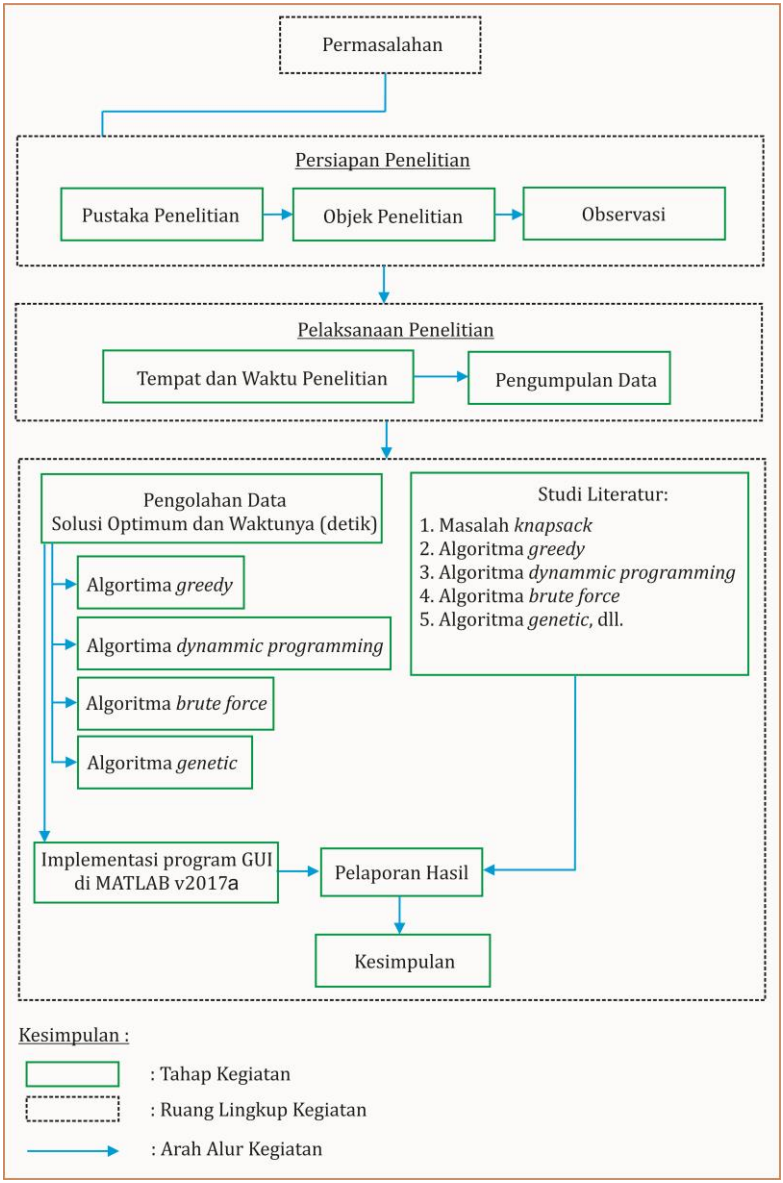
6. Menggunakan konsep *dynamic programming* metode *bottom-up* dalam menyelesaikan *knapsack problem.*
7. Menggunakan konsep *brute force* dalam menyelesaikan *knapsack problem.*
8. Menggunakan konsep *genetic* dalam menyelesaikan knapsack *problem.*
9. Menentukan waktu (detik) yang dihasilkan dari penyelesaian *integer knapsack problem* menggunakan

algoritma *greedy,* algoritma *dynamic programming*, algoritma *brute force*, dan algoritma *genetic*.

10. Pembuatan program menggunakan bahasa pemrograman MATLAB di *software* MATLAB *v2017a*. Program yang telah dibuat dijalankan menggunakan GUI.
11. Perbandingan keempat algoritma

    Membandingkan hasil optimum dari perhitungan algoritma *greedy*, algoritma *dynamic programming*, algoritma *brute force,* dan algoritma *genetic.* Membandingkan waktu (detik) yang dibutuhkan untuk menyelesaikan *integer knapsack problem* menggunakan keempat algoritma, sehingga dapat dipilih algoritma mana yang lebih baik untuk menyelesaikan pada permasalahan yang sama.

Selanjutnya diberikan skema penelitian secara sistematis setiap langkah-langkah penelitian ditunjukan pada Gambar 3.1. di bawah ini:




**Gambar 3.1.** Langkah-langkah penelitian

# BAB IV

# HASIL DAN PEMBAHASAN

## A. Hasil

### 1. Data Pengiriman Barang

Hasil pengamatan pada J&T Express *drop point* Ngaliyan Kota Semarang diperoleh data beberapa jenis barang yang diterima oleh kasir, dimana selanjutnya daftar jenis barang tersebut akan dikirimkan ke beberapa daerah yang ada di Indonesia dengan menggunakan paket reguler. Paket regular merupakan paket pilihan yang dapat menjangkau seluruh Indonesia hanya dalam jangka waktu paling cepat yaitu 3 hari dan paling lambat 7 hari, dimana dalam pengiriman barang paket reguler membutuhkan beberapa tahap dalam pengiriman. Pada penelitian ini data yang sudah diperoleh kemudian diolah dengan menggunakan 4 algoritma yaitu algoritma *greedy*, algoritma *dynamic programming*, algoritma *brute force*, dan algoritma *genetic* untuk mendapatkan jenis barang apa saja yang akan dikirimkan untuk tahap pertama.

Data yang diperoleh dari J&T Express *drop point* Ngaliyan kota Semarang sebagai berikut:

**Tabel 4.1.** Laporan J&T Express *drop point* Ngaliyan kota Semarang tanggal 12 Januari 2019

| No | Nama Barang | Berat | Biaya Pengiriman |
|---|---|---|---|
| 1 | SUB-BWI022 | 1 | Rp30.000.00 |
| 2 | SRG-UNG003 | 1 | Rp11.000.00 |
| 3 | SRG-TGL001 | 1 | Rp11.000.00 |
| 4 | PKU-RGT011 | 1 | Rp92.000.00 |
| 5 | PLM-PLM016 | 1 | Rp33.000.00 |
| 6 | SOC-BYL | 1 | Rp21.000.00 |
| 7 | JKS-JKT030 | 1 | Rp15.000.00 |
| 8 | SOC-SUJ010 | 8 | Rp13.000.00 |
| 9 | BGR-CIB030 | 2 | Rp152.000.00 |
| 10 | SRG-PML003 | 1 | Rp26.000.00 |
| 11 | SRG-KDS006 | 1 | Rp13.000.00 |
| 12 | SRG-KJN011 | 1 | Rp14.000.00 |
| 13 | JOG-PWJ009 | 1 | Rp21.000.00 |
| 14 | SRG-KNL008 | 1 | Rp13.000.00 |
| 15 | SRG-PWO011 | 1 | Rp13.000.00 |
| 16 | PNK-SKW003 | 1 | Rp62.000.00 |
| 17 | SRG-PTI003 | 1 | Rp13.000.00 |
| 18 | SRG-PDD010 | 1 | Rp13.000.00 |
| 19 | SUB-SPG012 | 1 | Rp27.000.00 |
| 20 | SRG-PKL002 | 1 | Rp11.000.00 |
| 21 | SRG-JPR006 | 1 | Rp13.000.00 |
| 22 | SRG-PUR002 | 1 | Rp11.000.00 |
| 23 | SRG-SRG008 | 1 | Rp9,000.00 |
| 24 | SRG-KNL016 | 4 | Rp13.000.00 |
| 25 | SUB-SDA | 1 | Rp108.000.00 |
| 26 | BDO-SMI035 | 1 | Rp25.000.00 |
| 27 | SOC-SRN002 | 1 | Rp25.000.00 |
| 28 | JOG-BTL004 | 1 | Rp11.000.00 |
| 29 | SRG-RMB007 | 1 | Rp11.000.00 |
| 30 | SUB-SUB012 | 1 | Rp18.000.00 |
| *n*=30 | Jumlah | 41 | Rp848,000.00 |

*Value/ profit* ditentukan berdasarkan biaya pengiriman. Biaya pengiriman juga diperoleh berdasarkan jarak kota tujuan pengiriman. Karena dalam hal ini J&T Express ingin mendahulukan kota tujuan terjauh dalam melakukan pengiriman, sehingga dalam hal ini J&T Express memperoleh *value/ profit* yang besar apabila melakukan pengiriman kota tujuan terlebih dahulu. Begitupun sebaliknya, J&T Express memperoleh *value/ profit* yang kecil apabila melakukan pengiriman kota tujuan terdekat terlebih dahulu.

Langkah pertama, yaitu membuat tabel daftar barang yang berisikan berat $w_i$ dan *value/ profit* dalam ribuan $v_i$ yang diperoleh berdasarkan pada Tabel 4.2. yaitu:

**Tabel 4.2.** Tabel Daftar Barang beserta berat dan *value/ profit*

| $i$ | No. Barang | $w_i$ | $v_i$ |
|---|---|---|---|
| 1 | 1 | 1 | 30 |
| 2 | 2 | 1 | 11 |
| 3 | 3 | 1 | 11 |
| 4 | 4 | 1 | 92 |
| 5 | 5 | 1 | 33 |
| 6 | 6 | 1 | 21 |
| 7 | 7 | 1 | 15 |
| 8 | 8 | 8 | 13 |
| 9 | 9 | 2 | 152 |
| 10 | 10 | 1 | 26 |
| 11 | 11 | 1 | 13 |
| 12 | 12 | 1 | 14 |
| 13 | 13 | 1 | 21 |
| 14 | 14 | 1 | 13 |
| 15 | 15 | 1 | 13 |

**Tabel 4.3.** Lanjutan

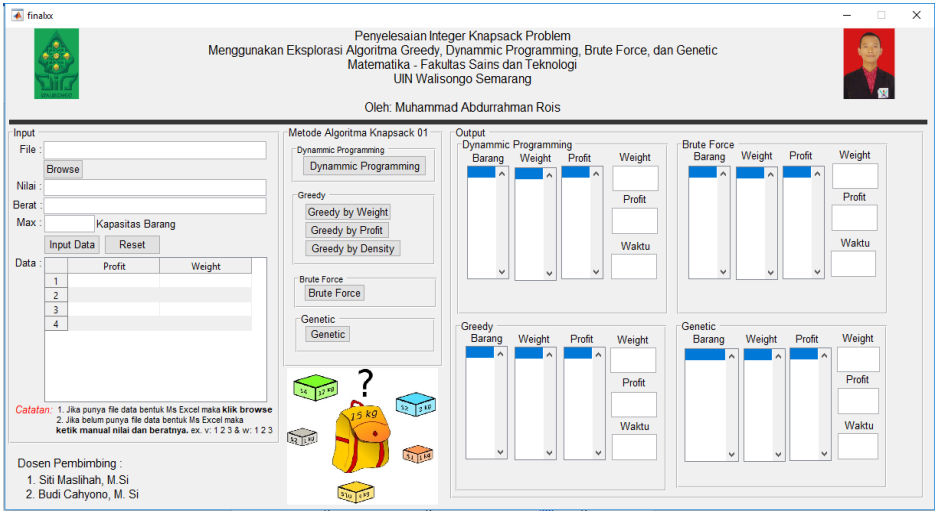
| | | | |
|---|---|---|---|
| 16 | 16 | 1 | 62 |
| 17 | 17 | 1 | 13 |
| 18 | 18 | 1 | 13 |
| 19 | 19 | 1 | 27 |
| 20 | 20 | 1 | 11 |
| 21 | 21 | 1 | 13 |
| 22 | 22 | 1 | 11 |
| 23 | 23 | 1 | 9 |
| 24 | 24 | 4 | 13 |
| 25 | 25 | 1 | 108 |
| 26 | 26 | 1 | 25 |
| 27 | 27 | 1 | 25 |
| 28 | 28 | 1 | 11 |
| 29 | 29 | 1 | 11 |
| 30 | 30 | 1 | 18 |
| $n$=30 | | $\sum_{i=1}^{n} w_i = 41$ | $\sum_{i=1}^{n} v_i = 848$ |

Langkah kedua, yaitu menentukan kapasitas maksimum dalam pengiriman tahap pertama. Pada kasus ini misalnya pada tahap pertama J&T Express hanya mengirimkan 32 kg pertahapnya atau $W = 32$. Sehingga ada batas maksimal yang harus dikirim oleh J&T Express dan mengeliminasi barang yang dianggap tidak memenuhi syarat untuk dikirim pada tahap pertama.

Langkah ketiga, yaitu memisalkan mengambil 4 barang (barang 1, 9, 10, 25) dari data penelitian dan kapasitas maksimumnya adalah 11 kg untuk diuji coba dalam hal waktu komputasi.

## 2. Program GUI untuk *Integer Knapsack Problem*

Program GUI dibuat menggunakan bantuan *software* MATLAB *vR2017a*. Program ini dibuat dengan tampilan GUI yang dapat dilihat pada Gambar 4.1. dan dibuat bertujuan untuk mempermudah dan mempercepat perhitungan pada *integer knapsack problem* dengan menggunakan algoritma *greedy, dynamic programming, brute force* dan *genetic.* Mulai menjalankan program diwali dengan membuka tampilan awal program GUI.



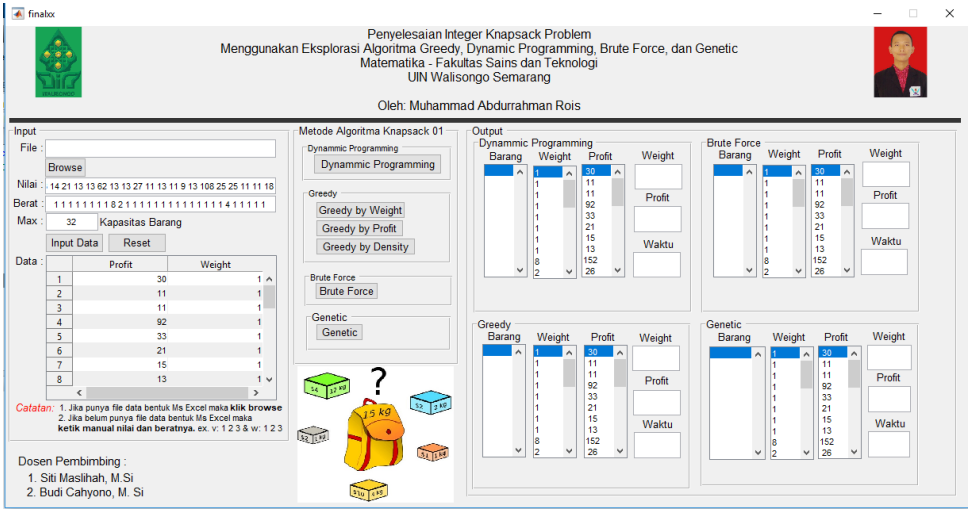

**Gambar 4.1.** Tampilan awal program

Menu yang terdapat pada Gambar 4.1. adalah sebagai berikut:

a) *Browse*, digunakan untuk menginput berat setiap barang dan keuntungan setiap barang yang sudah disimpan dalam MS. Excel bentuk format xlsx/xls.
b) Nilai, menunjukan nilai (*value/ profit*) setiap barang yang akan diangkut atau diambil.
c) Berat, menunjukan berat setiap barang yang akan diangkut atau diambil.
d) *Max*: menunjukan daya angkut maksimal untuk mengangkat barang.
e) *Input* data, digunakan untuk memproses masukan data yang sudah ada untuk ditampilkan di program.
f) *Reset*, digunakan untuk menghapus inputan dan proses dari metode yang sudah dijalankan dan kembali pada tampilan awal program dengan memasukan input data kembali di menu nilai, berat dan *max* (Untuk *browse* langsung melihat file mana yang akan digunakan).
g) *Dynamic Programming*, digunakan untuk memulai penyelesaian *integer knapsack problem* menggunakan algoritma *dynamic programming.*
h) *Greedy by Weight*, digunakan untuk memulai penyelesaian *integer knapsack problem* menggunakan algoritma *greedy* (konsep *greedy by weight*).

i) *Greedy by Profit*, digunakan untuk memulai penyelesaian *integer knapsack problem* menggunakan algoritma *greedy* (konsep *greedy by profit*).
j) *Greedy by Density*, digunakan untuk memulai penyelesaian *integer knapsack problem* menggunakan algoritma *greedy* (konsep *greedy by density*).
k) *Brute Force*, digunakan untuk memulai penyelesaian *integer knapsack problem* menggunakan algoritma *brute force.*
l) *Genetic*, digunakan untuk memulai penyelesaian *integer knapsack problem* menggunakan algoritma *genetic.*

Langkah awal yang digunakan untuk menjalankan program adalah menentukan nilai berat maksimum, berat, dan nilai (*value/ profit*) setiap barang pada kolom nilai, berat atau *browse file* jika data sudah dimiliki. Selanjutnya akan muncul tampilan program setelah diisi seperti yang ditunjukan pada Gambar 4.2. Setelah itu *klik* tombol *dynamic programming, greedy by weight, greedy by profit, greedy by density, brute force* dan *genetic* untuk memulai proses penyelesaian *integer knapsack problem.*




**Gambar 4.2.** Tampilan setelah data diisi

### 3. Penyelesaian *Integer Knapsack Problem*

#### a) Penyelesaian Menggunakan Algoritma *Greedy*

(1) *Greedy by profit*

Daftar barang-barang yang ada pada Tabel 4.2. diurutkan berdasarkan *value/ profit*-nya. Karena dalam hal ini J&T Express ingin memaksimalkan *value/ profit,* maka barang diurutkan berdasarkan *value/ profit* tertinggi. Selanjutnya, mengambil satu per satu barang berdasarkan urutan yang dapat ditampung oleh *knapsack* sampai *knapsack* penuh atau sudah tidak ada barang lagi. Peneliti mencoba menyelesaikan menjadi 2 bagian berdasarkan data penelitian untuk dianalisis.

Bagian pertama yakni menyelesaikan dengan permisalan diambil 4 data dengan berat yang berbeda (barang 1, 9, 10, 25) dan bagian kedua yakni semua barang untuk diselesaikan. Jadi diperoleh tabel sebagai berikut:

**Tabel 4.4.** Pengambilan barang menggunakan konsep *greedy by profit* bagian pertama

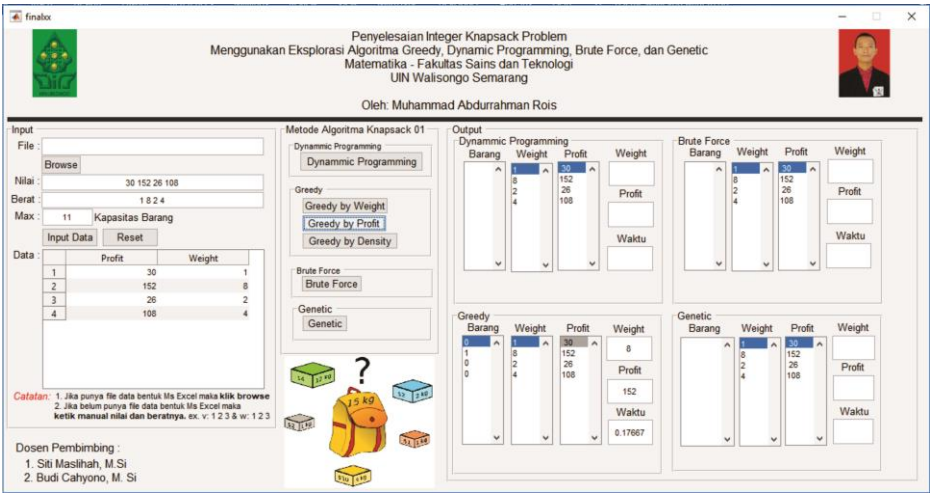
| $i$ | No. Barang | $w_i$ | $v_i$ | Frekuensi kumulatif dari $w_i$ | Status | $x_i$ | F.Kendala $(x_i \times w_i)$ | F.Objektif $(x_i \times v_i)$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 9 | 8 | 152 | 8 | Diambil | 1 | 8 | 152 |
| 2 | 25 | 4 | 108 | 12 | Eliminasi | 0 | 0 | 0 |
| 3 | 1 | 1 | 30 | 13 | Eliminasi | 0 | 0 | 0 |
| 4 | 10 | 2 | 26 | 15 | Eliminasi | 0 | 0 | 0 |
| $n=4$ | | $\sum_{i=1}^{n} w_i = 15$ | $\sum_{i=1}^{n} v_i = 316$ | | | $\sum_{i=1}^{n} x_i = 1$ | $\sum_{i=1}^{n} (x_i \times w_i) = 8$ | $\sum_{i=1}^{n} (x_i \times v_i) = 152$ |

**Tabel 4.5.** Pengambilan barang menggunakan konsep *greedy by profit* bagian kedua

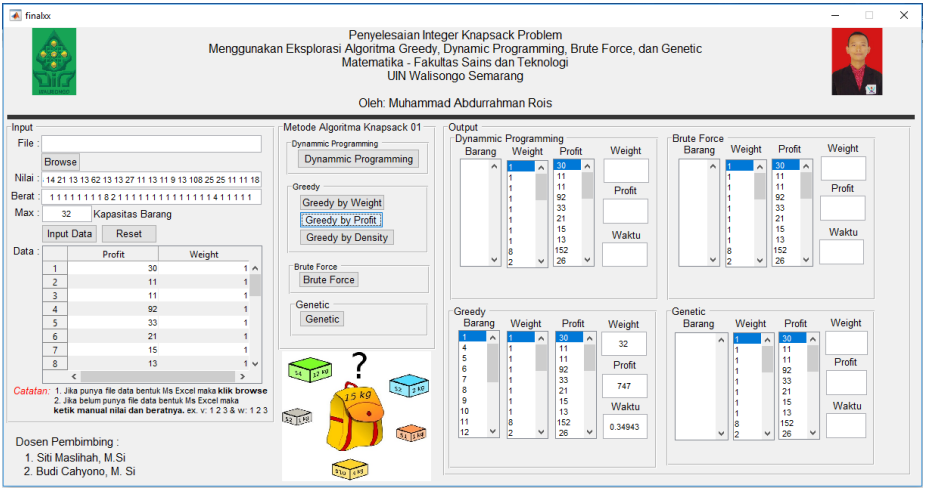
| $i$ | No. Barang | $w_i$ | $v_i$ | Frekuensi kumulatif dari $w_i$ | Status | $x_i$ | F.Kendala $(x_i \times w_i)$ | F.Objektif $(x_i \times v_i)$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 9 | 8 | 152 | 8 | Diambil | 1 | 8 | 152 |
| 2 | 25 | 4 | 108 | 12 | Diambil | 1 | 4 | 108 |
| 3 | 4 | 1 | 92 | 13 | Diambil | 1 | 1 | 92 |
| 4 | 16 | 1 | 62 | 14 | Diambil | 1 | 1 | 62 |
| 5 | 5 | 1 | 33 | 15 | Diambil | 1 | 1 | 33 |
| 6 | 1 | 1 | 30 | 16 | Diambil | 1 | 1 | 30 |
| 7 | 19 | 1 | 27 | 17 | Diambil | 1 | 1 | 27 |
| 8 | 10 | 2 | 26 | 19 | Diambil | 1 | 2 | 26 |
| 9 | 26 | 1 | 25 | 20 | Diambil | 1 | 1 | 25 |
| 10 | 27 | 1 | 25 | 21 | Diambil | 1 | 1 | 25 |
| 11 | 6 | 1 | 21 | 22 | Diambil | 1 | 1 | 21 |
| 12 | 13 | 1 | 21 | 23 | Diambil | 1 | 1 | 21 |
| 13 | 30 | 1 | 18 | 24 | Diambil | 1 | 1 | 18 |
| 14 | 7 | 1 | 15 | 25 | Diambil | 1 | 1 | 15 |
| 15 | 12 | 1 | 14 | 26 | Diambil | 1 | 1 | 14 |

**Tabel 4.6.** Lanjutan

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 16 | 15 | 1 | 13 | 27 | Diambil | 1 | 1 | 13 |
| 17 | 8 | 1 | 13 | 28 | Diambil | 1 | 1 | 13 |
| 18 | 11 | 1 | 13 | 29 | Diambil | 1 | 1 | 13 |
| 19 | 14 | 1 | 13 | 30 | Diambil | 1 | 1 | 13 |
| 20 | 17 | 1 | 13 | 31 | Diambil | 1 | 1 | 13 |
| 21 | 18 | 1 | 13 | 32 | Diambil | 1 | 1 | 13 |
| 22 | 21 | 1 | 13 | 33 | Eliminasi | 0 | 0 | 0 |
| 23 | 24 | 1 | 13 | 34 | Eliminasi | 0 | 0 | 0 |
| 24 | 2 | 1 | 11 | 35 | Eliminasi | 0 | 0 | 0 |
| 25 | 3 | 1 | 11 | 36 | Eliminasi | 0 | 0 | 0 |
| 26 | 20 | 1 | 11 | 37 | Eliminasi | 0 | 0 | 0 |
| 27 | 22 | 1 | 11 | 38 | Eliminasi | 0 | 0 | 0 |
| 28 | 28 | 1 | 11 | 39 | Eliminasi | 0 | 0 | 0 |
| 29 | 29 | 1 | 11 | 40 | Eliminasi | 0 | 0 | 0 |
| 30 | 23 | 1 | 9 | 41 | Eliminasi | 0 | 0 | 0 |
| $n=30$ | | $\sum_{i=1}^{n} w_i = 41$ | $\sum_{i=1}^{n} v_i = 848$ | | | $\sum_{i=1}^{n} x_i = 21$ | $\sum_{i=1}^{n} (x_i \times w_i) = 32$ | $\sum_{i=1}^{n} (x_i \times v_i) = 747$ |




**Gambar 4.3.** Hasil perhitungan algoritma *greedy by profit* bagian pertama




**Gambar 4.4.** Hasil perhitungan algoritma *greedy by profit* bagian kedua

Keterangan di atas dilihat dari tabel dan program GUI didapatkan: (1) Hasil maksimal pada bagian pertama dengan keuntungan optimal sebesar Rp 152.000,- dengan berat 8 kg. (2) Hasil maksimal bagian kedua dengan keuntungan optimal sebesar Rp 747.000,- dengan berat 32 kg. Sedangkan hasil dari program yang ditampilkan pada Gambar 4.3. untuk bagian pertama dan Gambar 4.4. bagian kedua menggunakan algoritma *greedy* (*greedy by profit*), dijelaskan bahwa barang yang dapat diangkut pada bagian pertama adalah hanya barang 9 dengan total berat 8 kg dan total keuntungan yang didapat Rp 152.000,- serta waktu komputasi yang dibutuhkan

0,17667 detik. Selanjutnya barang yang dapat diangkut pada bagian kedua adalah barang 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 25, 26, 27, 30 dengan total berat 32 kg dan total keuntungan yang didapat Rp 747.000,- serta waktu komputasi yang dibutuhkan 0,34943 detik.

(2) *Greedy by weight*

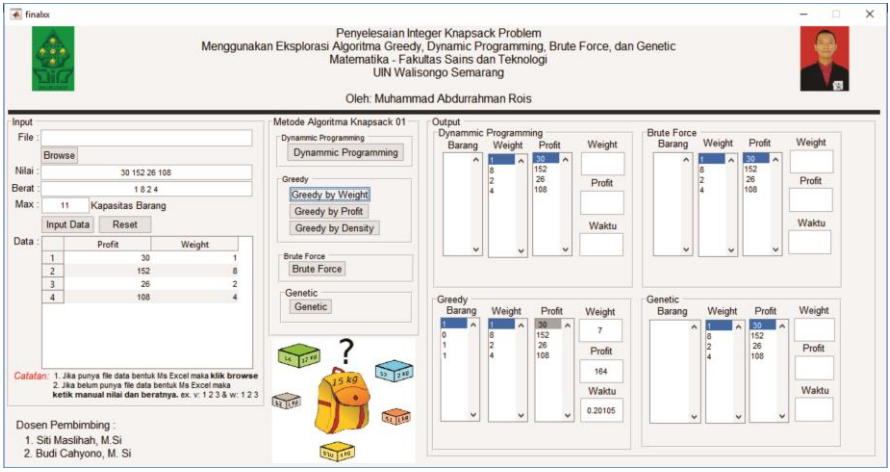
Daftar barang-barang yang ada pada Tabel 4.2. diurutkan berdasarkan ukuran *weight/* berat-nya. Karena dalam hal ini J&T Express ingin memaksimalkan keuntungan dengan mengambil barang sebanyak-banyaknya, maka barang diurutkan berdasarkan berat terkecil. Peneliti mencoba menyelesaikan menjadi 2 bagian berdasarkan data penelitian. Bagian pertama yakni memisalkan menyelesaikan dengan mengambil 4 data dengan berat yang berbeda (barang 1, 9, 10, 25) dan bagian kedua yakni semua barang untuk diselesaikan. Jadi diperoleh tabel yang dapat dilihat sebagai berikut:

**Tabel 4.7.** Pengambilan barang menggunakan konsep *greedy by weight* bagian pertama

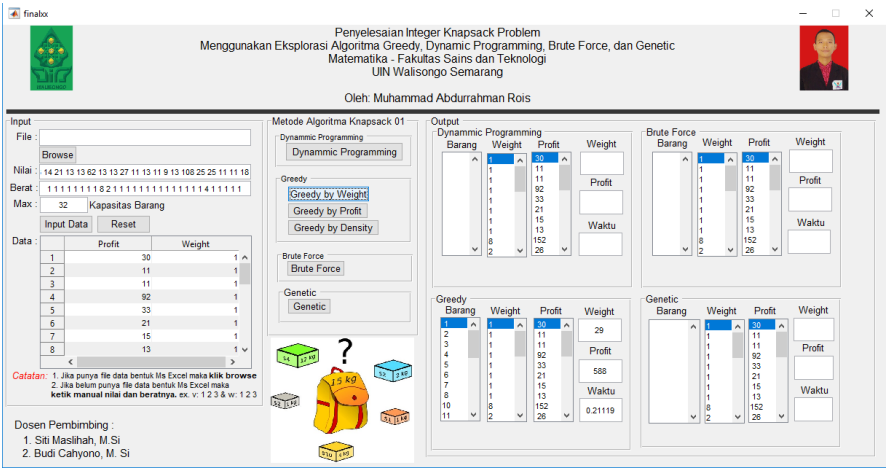
| $i$ | No. Barang | $w_i$ | $v_i$ | Frekuensi kumulatif dari $w_i$ | Status | $x_i$ | F.Kendala $(x_i \times w_i)$ | F.Objektif $(x_i \times v_i)$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 30 | 1 | Diambil | 1 | 1 | 30 |
| 2 | 10 | 2 | 26 | 3 | Diambil | 1 | 2 | 26 |
| 3 | 25 | 4 | 108 | 7 | Diambil | 1 | 4 | 108 |
| 4 | 9 | 8 | 152 | 15 | Eliminasi | 0 | 0 | 0 |
| $n=4$ | | $\sum_{i=1}^{n} w_i = 15$ | $\sum_{i=1}^{n} v_i = 316$ | 26 | 0 | $\sum_{i=1}^{n} x_i = 3$ | $\sum_{i=1}^{n}(x_i \times w_i) = 7$ | $\sum_{i=1}^{n}(x_i \times v_i) = 164$ |

**Tabel 4.8.** Pengambilan barang menggunakan konsep *greedy by weight* bagian kedua

| $i$ | No. Barang | $w_i$ | $v_i$ | Frekuensi kumulatif dari $w_i$ | Status | $x_i$ | F.Kendala $(x_i \times w_i)$ | F.Objektif $(x_i \times v_i)$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 30 | 1 | Diambil | 1 | 1 | 30 |
| 2 | 2 | 1 | 11 | 2 | Diambil | 1 | 1 | 11 |
| 3 | 3 | 1 | 11 | 3 | Diambil | 1 | 1 | 11 |
| 4 | 4 | 1 | 92 | 4 | Diambil | 1 | 1 | 92 |
| 5 | 5 | 1 | 33 | 5 | Diambil | 1 | 1 | 33 |
| 6 | 6 | 1 | 21 | 6 | Diambil | 1 | 1 | 21 |
| 7 | 7 | 1 | 15 | 7 | Diambil | 1 | 1 | 15 |
| 8 | 8 | 1 | 13 | 8 | Diambil | 1 | 1 | 13 |
| 9 | 11 | 1 | 13 | 9 | Diambil | 1 | 1 | 13 |
| 10 | 12 | 1 | 14 | 10 | Diambil | 1 | 1 | 14 |
| 11 | 13 | 1 | 21 | 11 | Diambil | 1 | 1 | 21 |
| 12 | 14 | 1 | 13 | 12 | Diambil | 1 | 1 | 13 |
| 13 | 15 | 1 | 13 | 13 | Diambil | 1 | 1 | 13 |
| 14 | 16 | 1 | 62 | 14 | Diambil | 1 | 1 | 62 |
| 15 | 17 | 1 | 13 | 15 | Diambil | 1 | 1 | 13 |
| 16 | 18 | 1 | 13 | 16 | Diambil | 1 | 1 | 13 |
| 17 | 19 | 1 | 27 | 17 | Diambil | 1 | 1 | 27 |
| 18 | 20 | 1 | 11 | 18 | Diambil | 1 | 1 | 11 |
| 19 | 21 | 1 | 13 | 19 | Diambil | 1 | 1 | 13 |
| 20 | 22 | 1 | 11 | 20 | Diambil | 1 | 1 | 11 |
| 21 | 23 | 1 | 9 | 21 | Diambil | 1 | 1 | 9 |
| 22 | 24 | 1 | 13 | 22 | Diambil | 1 | 1 | 13 |
| 23 | 26 | 1 | 25 | 23 | Diambil | 1 | 1 | 25 |
| 24 | 27 | 1 | 25 | 24 | Diambil | 1 | 1 | 25 |
| 25 | 28 | 1 | 11 | 25 | Diambil | 1 | 1 | 11 |
| 26 | 29 | 1 | 11 | 26 | Diambil | 1 | 1 | 11 |
| 27 | 30 | 1 | 18 | 27 | Diambil | 1 | 1 | 18 |
| 28 | 10 | 2 | 26 | 29 | Diambil | 1 | 2 | 26 |
| 29 | 25 | 4 | 108 | 33 | Eliminasi | 0 | 0 | 0 |
| 30 | 9 | 8 | 152 | 41 | Eliminasi | 0 | 0 | 0 |
| $n=30$ | | $\sum_{i=1}^{n} w_i = 41$ | $\sum_{i=1}^{n} v_i = 848$ | | | $\sum_{i=1}^{n} x_i = 28$ | $\sum_{i=1}^{n}(x_i \times w_i) = 29$ | $\sum_{i=1}^{n}(x_i \times v_i) = 588$ |




**Gambar 4.5.** Hasil perhitungan algoritma *greedy by weight* bagian pertama




**Gambar 4.6.** Hasil perhitungan algoritma *greedy by weight* bagian kedua

Keterangan di atas dilihat dari tabel dan program GUI didapatkan hasil maksimal bagian pertama dengan keuntungan optimal sebesar Rp 164.000,- dengan berat 7 Kg. Dan hasil maksimal bagian kedua dengan keuntungan optimal sebesar Rp 588.000,- dengan berat 29 kg. Sedangkan hasil dari program yang ditampilkan pada Gambar 4.5. untuk bagian pertama dan Gambar 4.6. untuk bagian kedua menggunakan algoritma *greedy* (*greedy by weight*), dijelaskan bahwa: (1) Barang yang dapat diangkut bagian pertama adalah barang 1, 10, 25 dengan total berat 7 kg dan total keuntungan yang didapat Rp 164.000,- serta waktu komputasi yang dibutuhkan 0,2015 detik. (2) Barang

yang dapat diangkut pada bagian kedua adalah barang 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 dengan total berat 29 kg dan total keuntungan yang didapat Rp 588.000,- serta waktu komputasi yang dibutuhkan 0,21119 detik.

(3) *Greedy by density*

*Value/ profit* perberat (*density*) dicari terlebih dahulu di setiap barang. Nilai *density* setiap barang dapat dilihat pada tabel sebagai berikut:

**Tabel 4.9.** Tabel daftar barang beserta berat, *value/ profit* dan *density*

| $i$ | **No. Barang** | $w_i$ | $v_i$ | **Density** $(D_i)$ |
|---|---|---|---|---|
| 1 | 1 | 1 | 30 | 30 |
| 2 | 2 | 1 | 11 | 11 |
| 3 | 3 | 1 | 11 | 11 |
| 4 | 4 | 1 | 92 | 92 |
| 5 | 5 | 1 | 33 | 33 |
| 6 | 6 | 1 | 21 | 21 |
| 7 | 7 | 1 | 15 | 15 |
| 8 | 8 | 1 | 13 | 13 |
| 9 | 9 | 8 | 152 | 19 |
| 10 | 10 | 2 | 26 | 13 |
| 11 | 11 | 1 | 13 | 13 |
| 12 | 12 | 1 | 14 | 14 |
| 13 | 13 | 1 | 21 | 21 |
| 14 | 14 | 1 | 13 | 13 |
| 15 | 15 | 1 | 13 | 13 |
| 16 | 16 | 1 | 62 | 62 |
| 17 | 17 | 1 | 13 | 13 |
| 18 | 18 | 1 | 13 | 13 |
| 19 | 19 | 1 | 27 | 27 |
| 20 | 20 | 1 | 11 | 11 |

**Tabel 4.10.** Lanjutan

| | | | | |
|---|---|---|---|---|
| 21 | 21 | 1 | 13 | 13 |
| 22 | 22 | 1 | 11 | 11 |
| 23 | 23 | 1 | 9 | 9 |
| 24 | 24 | 1 | 13 | 13 |
| 25 | 25 | 4 | 108 | 27 |
| 26 | 26 | 1 | 25 | 25 |
| 27 | 27 | 1 | 25 | 25 |
| 28 | 28 | 1 | 11 | 11 |
| 29 | 29 | 1 | 11 | 11 |
| 30 | 30 | 1 | 18 | 18 |
| $n=30$ | | $\sum_{i=1}^{30} w_i = 41$ | $\sum_{i=1}^{30} v_i = 848$ | $\sum_{i=1}^{30} D_i = 621$ |

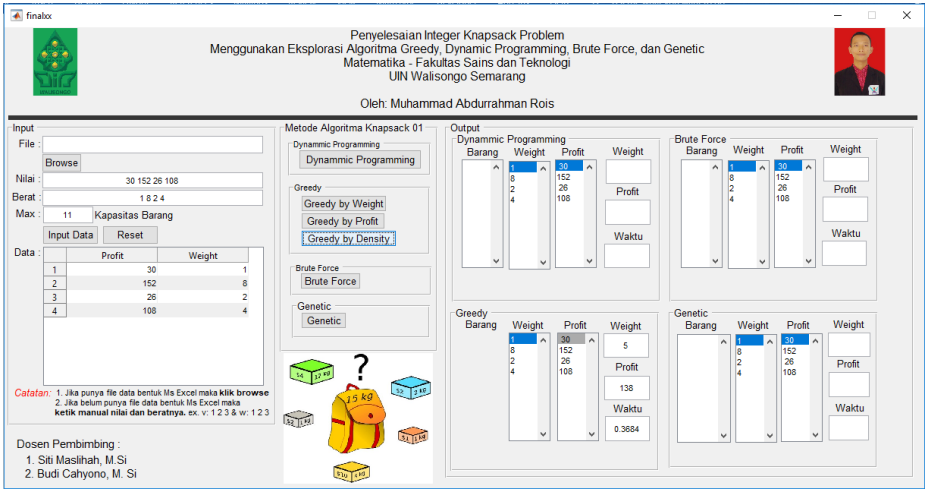
Tahap selanjutnya adalah mengurutkan daftar barang-barang yang ada pada Tabel 4.9. berdasarkan nilai *density* terbesar. Peneliti mencoba menyelesaikan menjadi 2 bagian berdasarkan data penelitian. Bagian pertama yakni menyelesaikan dengan mengambil 4 data dengan berat yang berbeda (barang 1, 9, 10, 25) dan bagian kedua yakni semua barang untuk diselesaikan. Jadi diperoleh tabel sebagai berikut:

**Tabel 4.11.** Pengambilan barang menggunakan konsep *greedy by density* bagian pertama

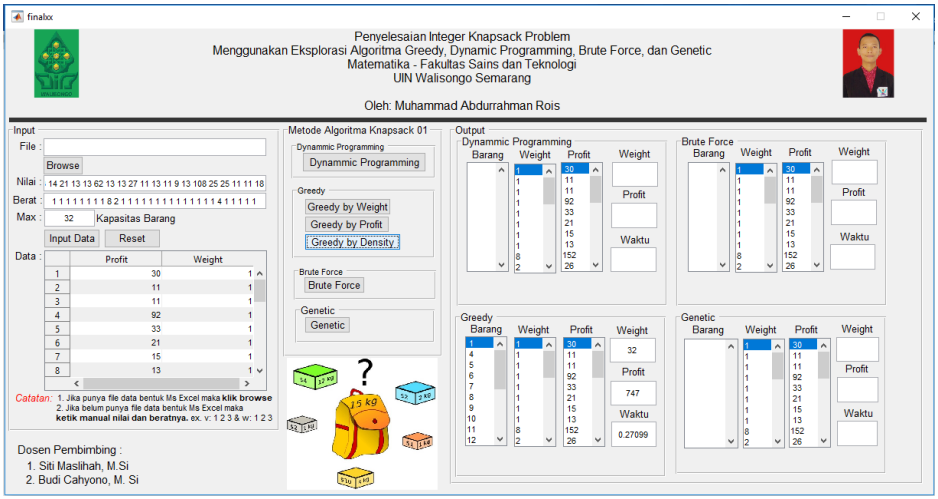
| $i$ | No. Barang | $w_i$ | $v_i$ | Density $(D_i)$ | Frekuensi kumulatif dari $w_i$ | Status | $x_i$ | F.Kendala $(x_i \times w_i)$ | F.Objektif $(x_i \times v_i)$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 30 | 30 | 1 | Diambil | 1 | 1 | 30 |
| 2 | 25 | 4 | 108 | 27 | 5 | Diambil | 1 | 4 | 108 |
| 3 | 9 | 8 | 152 | 19 | 13 | Eliminasi | 0 | 0 | 0 |
| 4 | 10 | 2 | 26 | 13 | 15 | Eliminasi | 0 | 0 | 0 |
| $n=4$ | | $\sum_{i=1}^{n} w_i = 15$ | $\sum_{i=1}^{n} v_i = 316$ | $\sum_{i=1}^{n} D_i = 89$ | | | 2 | $\sum_{i=1}^{n} (x_i \times w_i) = 5$ | $\sum_{i=1}^{n} (x_i \times v_i) = 138$ |

**Tabel 4.12.** Pengambilan barang menggunakan konsep *greedy by density* bagian kedua

| $i$ | No. Barang | $W_i$ | $V_i$ | Density $(D_i)$ | Frekuensi kumulatif dari $W_i$ | Status | $X_i$ | F.Kendala $(X_i \times W_i)$ | F.Objektif $(X_i \times V_i)$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 1 | 92 | 92 | 1 | Diambil | 1 | 1 | 92 |
| 2 | 16 | 1 | 62 | 62 | 2 | Diambil | 1 | 1 | 62 |
| 3 | 5 | 1 | 33 | 33 | 3 | Diambil | 1 | 1 | 33 |
| 4 | 1 | 1 | 30 | 30 | 4 | Diambil | 1 | 1 | 30 |
| 5 | 19 | 1 | 27 | 27 | 5 | Diambil | 1 | 1 | 27 |
| 6 | 25 | 4 | 108 | 27 | 9 | Diambil | 1 | 4 | 108 |
| 7 | 26 | 1 | 25 | 25 | 10 | Diambil | 1 | 1 | 25 |
| 8 | 27 | 1 | 25 | 25 | 11 | Diambil | 1 | 1 | 25 |
| 9 | 6 | 1 | 21 | 21 | 12 | Diambil | 1 | 1 | 21 |
| 10 | 13 | 1 | 21 | 21 | 13 | Diambil | 1 | 1 | 21 |
| 11 | 9 | 8 | 152 | 19 | 21 | Diambil | 1 | 8 | 152 |
| 12 | 30 | 1 | 18 | 18 | 22 | Diambil | 1 | 1 | 18 |
| 13 | 7 | 1 | 15 | 15 | 23 | Diambil | 1 | 1 | 15 |
| 14 | 12 | 1 | 14 | 14 | 24 | Diambil | 1 | 1 | 14 |
| 15 | 15 | 1 | 13 | 13 | 25 | Diambil | 1 | 1 | 13 |
| 16 | 8 | 1 | 13 | 13 | 26 | Diambil | 1 | 1 | 13 |
| 17 | 10 | 2 | 26 | 13 | 28 | Diambil | 1 | 2 | 26 |
| 18 | 11 | 1 | 13 | 13 | 29 | Diambil | 1 | 1 | 13 |
| 19 | 14 | 1 | 13 | 13 | 30 | Diambil | 1 | 1 | 13 |
| 20 | 17 | 1 | 13 | 13 | 31 | Diambil | 1 | 1 | 13 |
| 21 | 18 | 1 | 13 | 13 | 32 | Diambil | 1 | 1 | 13 |
| 22 | 21 | 1 | 13 | 13 | 33 | Eliminasi | 0 | 0 | 0 |
| 23 | 24 | 1 | 13 | 13 | 34 | Eliminasi | 0 | 0 | 0 |
| 24 | 2 | 1 | 11 | 11 | 35 | Eliminasi | 0 | 0 | 0 |
| 25 | 3 | 1 | 11 | 11 | 36 | Eliminasi | 0 | 0 | 0 |
| 26 | 20 | 1 | 11 | 11 | 37 | Eliminasi | 0 | 0 | 0 |
| 27 | 22 | 1 | 11 | 11 | 38 | Eliminasi | 0 | 0 | 0 |
| 28 | 28 | 1 | 11 | 11 | 39 | Eliminasi | 0 | 0 | 0 |
| 29 | 29 | 1 | 11 | 11 | 40 | Eliminasi | 0 | 0 | 0 |
| 30 | 23 | 1 | 9 | 9 | 41 | Eliminasi | 0 | 0 | 0 |
| $n=30$ | | $\sum_{i=1}^{n} w_i = 41$ | $\sum_{i=1}^{n} v_i = 848$ | $\sum_{i=1}^{n} D_i = 621$ | | | $\sum_{i=1}^{n} x_i = 21$ | $\sum_{i=1}^{n} (x_i \times w_i) = 32$ | $\sum_{i=1}^{n} (x_i \times v_i) = 747$ |




**Gambar 4.7.** Hasil perhitungan algoritma *greedy by density* bagian pertama




**Gambar 4.8.** Hasil perhitungan algoritma *greedy by density* bagian kedua

Keterangan di atas dilihat dari tabel dan program GUI didapatkan: (1) Hasil maksimal bagian pertama dengan keuntungan optimal sebesar Rp 138.000,- dengan berat 5 kg. (2) Hasil maksimal bagian kedua dengan keuntungan optimal sebesar Rp 747.000,- dengan berat 32 kg. Sedangkan hasil dari program yang ditampilkan pada Gambar 4.7. untuk bagian pertama dan Gambar 4.8. untuk bagian kedua menggunakan algoritma *greedy* (*greedy by density*), dijelaskan bahwa barang yang dapat diangkut bagian pertama adalah barang 1, 25 dengan total berat 5 kg dan total keuntungan yang didapat Rp 138.000,- serta waktu komputasi yang dibutuhkan 0,3684 detik. Selanjutnya barang yang dapat diangkut pada bagian kedua adalah barang 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 25, 26, 27, 30 dengan total berat 32 kg dan total keuntungan yang didapat Rp 747.000,- serta waktu komputasi yang dibutuhkan 0,2709 detik.

**b) Penyelesaian Menggunakan Algoritma *Dynamic Programming***

Data pengamatan yang diperoleh dari J&T Express sudah ditampilkan pada Tabel 4.1. dan selanjutnya diselesaikan menggunakan algoritma *dynamic*

*programming*. Peneliti mencoba menyelesaikan menjadi 2 bagian. Bagian pertama yakni menyelesaikan permisalan dengan mengambil 4 data dengan berat yang berbeda (barang 1, 9, 10, 25) dan bagian kedua yakni semua barang untuk diselesaikan. Langkah-langkah penyelesaiannya sebagai berikut:

(1) Struktur dari masalah

Masalah bagian pertama diketahui $n = 4$ jenis barang yang dinotasikan $i$ ($i = 1,2,3,4$) dengan bobot barang $i$ dinotasikan dengan $w_i$ sehingga $w_i = [1, 8, 2, 4]$. Masalah bagian kedua diketahui $n = 30$ jenis barang yang dinotasikan $i$ ($i = 1,2,...,30$) dengan bobot barang *i* dinotasikan dengan $w_i$ sehingga $w_i = [1, 1, 1, 1, 1, 1, 1, 1, 8, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 1]$. Kapasitas dalam menampung barang bagian pertama dinotasikan dengan $W_1 = 11$ dan kapasitas untuk menampung barang bagian kedua dinotasikan dengan $W_2 = 32$. Dimana $w_i \leq W_{1,2}$ dengan batasan $\sum_{i=1}^{n} w_i x_i \leq W_{1,2}$. Tujuan dari konsep *dynamic programming* adalah mencari keuntungan optimal yang dinotasikan dengan *Z* dari semua barang yang akan diangkut dengan keuntungan dari setiap

barang. Keuntungan bagian awal dari setiap barang dinotasikan dengan $v_i$ sehingga nilai $v_i =$ **[**30**,** 152**,** 26**,** 108**]** dan keuntungan bagian kedua nilainya $v_i =$ **[**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**]** sehingga ditulis $Z = \sum_{i=1}^{n} v_i x_i$. Selanjutnya untuk memperoleh nilai keuntungan optimal dilakukan perhitungan dengan teknik *bottom-up*.

(2) Struktur tabel

*Value/ profit,* berat dan kapasitas maksimum barang yang sudah diinput kemudian diselesaikan menggunakan algoritma ini untuk mencari nilai keuntungan setiap barang dan mengisi setiap *cell* *M*(*n*,*W*) untuk disimpan nilai keuntungannya dengan mengikuti persamaan di bawah ini:

$$M(i,j) = \begin{cases} 0, & \text{Jika } j = 0 \quad (10) \\ M(i-1,j), & \text{Jika } j < w_i \quad (11) \\ \max(M(i-1,j), M(i-1,j-w_i) + p_i), & \text{Jika} \geq w_i \quad (12) \end{cases}$$

(3) Teknik *bottom-up*

Perhitungan dimulai untuk bagian pertama yaitu dengan mencari nilai keuntungan barang ke-1 sampai barang ke-4 dan perhitungan bagian kedua yaitu mencari nilai keuntungan barang ke-1 kemudian

mencari keuntungan barang ke-2 dan seterusnya sampai barang ke-30 sesuai persamaan langkah 2. Selanjutnya, dari langkah 2 diperoleh struktur tabel berbentuk matriks yang berisi keputusan-keputusan optimal yaitu matriks *M*. Keuntungan optimal dari masalah di atas adalah nilai dari *M(n,W)* yang diilustrasikan seperti pada Tabel 4.13. untuk bagian pertama dan Tabel 4.14. untuk bagian kedua.

Adapun langkah untuk menjalankan program algoritma *dynamic programming* adalah sebagai berikut:

(1) Masukan jumlah data bagian pertama $n = 4$ dan bagian kedua $n = 30$

(2) Masukan *value/ profit* barang bagian pertama

$v_i = [30, 152, 26, 108]$

(3) Masukan *value/ profit* barang bagian kedua

$v_i = [30, 11, 11, 92, 33, 21, 15, 13, 152, 26, 13, 14, 21,$
$13, 13, 62, 13, 13, 27, 11, 13, 11, 9, 13, 108, 25, 25, 11, 11, 18]$

(4) Masukan berat barang bagian pertama

$w_i = [1, 8, 2, 4]$

(5) Masukan berat barang bagian kedua

$w_i = [1, 1, 1, 1, 1, 1, 1, 1, 8, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,$
$1, 1, 1, 4, 1, 1, 1, 1, 1]$

(6) Masukan kapasitas maksimum bagian pertama $W = 11$ dan bagian kedua $W = 32$

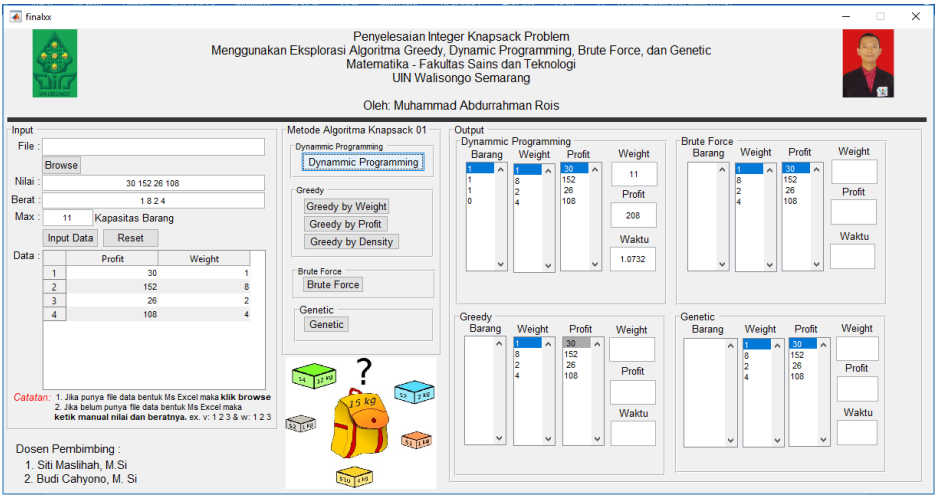
Solusi optimal permasalahan *knapsack* dihitung menggunakan algoritma *dynamic programming* disajikan tabel ringkasan hasil perhitungannya sebagai berikut:

**Tabel 4.13.** Ringkasan hasil *output* program algoritma *dynamic programming* bagian pertama

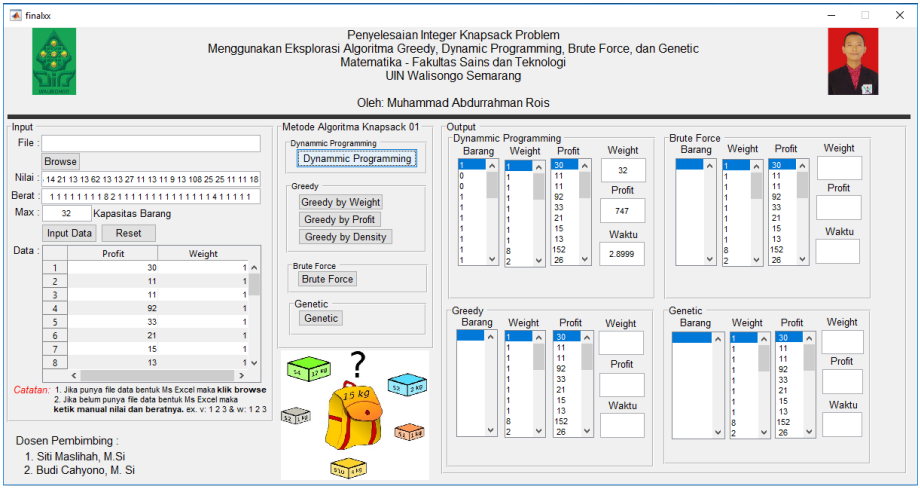
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| 2 | 0 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 152 | 182 | 182 | 182 |
| 3 | 0 | 30 | 30 | 56 | 56 | 56 | 56 | 56 | 152 | 182 | 182 | 208 |
| 4 | 0 | 30 | 30 | 56 | 108 | 138 | 138 | 164 | 164 | 182 | 182 | 208 |

**Tabel 4.14.** Ringkasan hasil *output* program algoritma *dynamic programming* bagian kedua

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| 2 | 0 | 30 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 |
| 3 | 0 | 30 | 41 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 | 52 |
| 4 | 0 | 92 | 122 | 133 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 | 144 |
| 5 | 0 | 92 | 125 | 155 | 166 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 | 177 |
| 6 | 0 | 92 | 125 | 155 | 176 | 187 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 | 198 |
| 7 | 0 | 92 | 125 | 155 | 176 | 191 | 202 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 | 213 |
| 8 | 0 | 92 | 125 | 155 | 176 | 191 | 204 | 215 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 |
| 9 | 0 | 92 | 125 | 155 | 176 | 191 | 204 | 215 | 226 | 244 | 277 | 307 | 328 | 343 | 356 | 367 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 |
| 10 | 0 | 92 | 125 | 155 | 176 | 191 | 204 | 215 | 226 | 244 | 277 | 307 | 328 | 343 | 356 | 369 | 382 | 393 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 |
| 11 | 0 | 92 | 125 | 155 | 176 | 191 | 204 | 215 | 226 | 244 | 277 | 307 | 328 | 343 | 356 | 367 | 382 | 395 | 406 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 417 |
| 12 | 0 | 92 | 125 | 155 | 176 | 191 | 204 | 215 | 226 | 244 | 277 | 307 | 328 | 343 | 356 | 367 | 382 | 357 | 370 | 383 | 396 | 409 | 420 | 431 | 431 | 431 | 431 | 431 | 431 | 431 | 431 | 431 | 431 |
| 13 | 0 | 92 | 125 | 155 | 176 | 191 | 204 | 215 | 226 | 244 | 277 | 307 | 328 | 349 | 364 | 378 | 391 | 404 | 417 | 430 | 441 | 452 | 452 | 452 | 452 | 452 | 452 | 452 | 452 | 452 | 452 | 452 | 452 |
| 14 | 0 | 92 | 125 | 155 | 176 | 191 | 204 | 215 | 226 | 244 | 277 | 307 | 328 | 349 | 364 | 378 | 391 | 404 | 417 | 430 | 443 | 454 | 465 | 465 | 465 | 465 | 465 | 465 | 465 | 465 | 465 | 465 | 465 |
| 15 | 0 | 92 | 125 | 155 | 176 | 191 | 204 | 215 | 226 | 244 | 277 | 307 | 328 | 349 | 364 | 378 | 391 | 404 | 417 | 430 | 443 | 456 | 467 | 478 | 478 | 478 | 478 | 478 | 478 | 478 | 478 | 478 | 478 |
| 16 | 0 | 92 | 154 | 187 | 217 | 238 | 259 | 274 | 288 | 301 | 314 | 339 | 369 | 390 | 411 | 426 | 440 | 453 | 466 | 479 | 492 | 505 | 518 | 529 | 540 | 540 | 540 | 540 | 540 | 540 | 540 | 540 | 540 |
| 17 | 0 | 92 | 154 | 187 | 217 | 238 | 259 | 274 | 288 | 301 | 314 | 339 | 369 | 390 | 411 | 426 | 440 | 453 | 466 | 479 | 492 | 505 | 518 | 531 | 542 | 553 | 553 | 553 | 553 | 553 | 553 | 553 | 553 |
| 18 | 0 | 92 | 154 | 187 | 217 | 238 | 259 | 274 | 288 | 301 | 314 | 339 | 369 | 390 | 411 | 426 | 440 | 453 | 466 | 479 | 492 | 505 | 518 | 531 | 542 | 555 | 566 | 566 | 566 | 566 | 566 | 566 | 566 |
| 19 | 0 | 92 | 154 | 187 | 217 | 244 | 265 | 286 | 301 | 315 | 328 | 341 | 369 | 396 | 417 | 438 | 453 | 467 | 480 | 493 | 506 | 519 | 532 | 545 | 558 | 571 | 582 | 593 | 593 | 593 | 593 | 593 | 593 |
| 20 | 0 | 92 | 154 | 187 | 217 | 244 | 265 | 286 | 301 | 315 | 328 | 341 | 369 | 396 | 417 | 438 | 453 | 467 | 480 | 493 | 506 | 519 | 532 | 545 | 558 | 571 | 582 | 593 | 604 | 604 | 604 | 604 | 604 |
| 21 | 0 | 92 | 154 | 187 | 217 | 244 | 265 | 286 | 301 | 315 | 328 | 341 | 369 | 396 | 417 | 438 | 453 | 467 | 480 | 493 | 506 | 519 | 532 | 545 | 558 | 571 | 584 | 595 | 606 | 617 | 617 | 617 | 617 |
| 22 | 0 | 92 | 154 | 187 | 217 | 244 | 265 | 286 | 301 | 315 | 328 | 341 | 369 | 396 | 417 | 438 | 453 | 467 | 480 | 493 | 506 | 519 | 532 | 545 | 558 | 571 | 582 | 593 | 606 | 617 | 628 | 628 | 628 |
| 23 | 0 | 92 | 154 | 187 | 217 | 244 | 265 | 286 | 301 | 315 | 328 | 341 | 369 | 396 | 417 | 438 | 453 | 467 | 480 | 493 | 506 | 519 | 532 | 545 | 558 | 571 | 582 | 593 | 606 | 617 | 628 | 637 | 637 |
| 24 | 0 | 92 | 154 | 187 | 217 | 244 | 265 | 286 | 301 | 315 | 328 | 341 | 369 | 396 | 417 | 438 | 453 | 467 | 480 | 493 | 506 | 519 | 532 | 545 | 558 | 571 | 582 | 597 | 608 | 619 | 630 | 641 | 650 |
| 25 | 0 | 92 | 154 | 187 | 217 | 244 | 265 | 295 | 325 | 352 | 373 | 394 | 409 | 423 | 436 | 449 | 477 | 504 | 525 | 546 | 561 | 575 | 588 | 601 | 614 | 627 | 640 | 653 | 666 | 679 | 692 | 705 | 716 |
| 26 | 0 | 92 | 154 | 187 | 217 | 244 | 269 | 295 | 325 | 352 | 377 | 398 | 419 | 434 | 448 | 461 | 477 | 504 | 529 | 550 | 571 | 586 | 600 | 613 | 626 | 639 | 652 | 665 | 678 | 691 | 704 | 717 | 730 |
| 27 | 0 | 92 | 154 | 187 | 217 | 244 | 269 | 295 | 325 | 352 | 377 | 402 | 423 | 444 | 462 | 477 | 491 | 504 | 529 | 554 | 575 | 596 | 614 | 629 | 643 | 656 | 669 | 682 | 695 | 708 | 716 | 729 | 742 |
| 28 | 0 | 92 | 154 | 187 | 217 | 244 | 269 | 295 | 325 | 352 | 377 | 402 | 423 | 444 | 459 | 473 | 486 | 504 | 529 | 554 | 575 | 596 | 611 | 625 | 638 | 651 | 664 | 667 | 690 | 703 | 716 | 729 | 742 |
| 29 | 0 | 92 | 154 | 187 | 217 | 244 | 269 | 295 | 325 | 352 | 377 | 402 | 423 | 444 | 462 | 477 | 491 | 504 | 529 | 554 | 575 | 596 | 614 | 629 | 643 | 656 | 669 | 682 | 695 | 708 | 721 | 734 | 742 |
| 30 | 0 | 92 | 154 | 187 | 217 | 244 | 269 | 295 | 325 | 352 | 377 | 402 | 423 | 444 | 462 | 477 | 491 | 504 | 529 | 554 | 575 | 596 | 614 | 629 | 643 | 656 | 669 | 682 | 695 | 708 | 721 | 734 | 747 |




**Gambar 4.9.** Hasil perhitungan algoritma *dynamic programming* bagian pertama




**Gambar 4.10.** Hasil perhitungan algoritma *dynamic programming* bagian kedua

Perhitungan di atas menggunakan algoritma *dynamic programming* dijelaskan bahwa barang yang dapat diangkut pada bagian pertama adalah barang 1, 9, 10 dan diperoleh hasil maksimal dengan keuntungan optimal sebesar Rp 208.000,- dengan berat 11 kg dan waktu komputasi 1,0732 detik. Sedangkan barang yang dapat diangkut pada bagian kedua adalah barang 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 25, 26, 27, 30 dan diperoleh hasil maksimal dengan keuntungan optimal sebesar Rp 747.000,- dengan berat 32 kg dan waktu komputasi 2,8999 detik.

**c) Penyelesaian Menggunakan Algoritma *Brute Force***

Peneliti mencoba menyelesaikan menjadi 2 bagian berdasarkan data penelitian. Bagian pertama yakni menyelesaikan dengan mengambil 4 data dengan berat yang berbeda (barang 1, 9, 10, 25) dan bagian kedua yakni semua barang untuk diselesaikan. Langkah-langkah untuk menjalankan program algoritma brute *force* adalah sebagai berikut:

(1) Masukan jumlah data bagian pertama $n = 4$ dan bagian kedua $n = 30$

(2) Masukan *value/ profit* barang bagian pertama

$v_i = [30, 152, 26, 108]$

(3) Masukan *value/ profit* barang bagian kedua

$v_i = [30, 11, 11, 92, 33, 21, 15, 13, 152, 26, 13, 14, 21,$
$13, 13, 62, 13, 13, 27, 11, 13, 11, 9, 13, 108, 25, 25, 11, 11, 18]$

(4) Masukan berat barang bagian pertama

$w_i = [1, 8, 2, 4]$

(5) Masukan berat barang bagian kedua

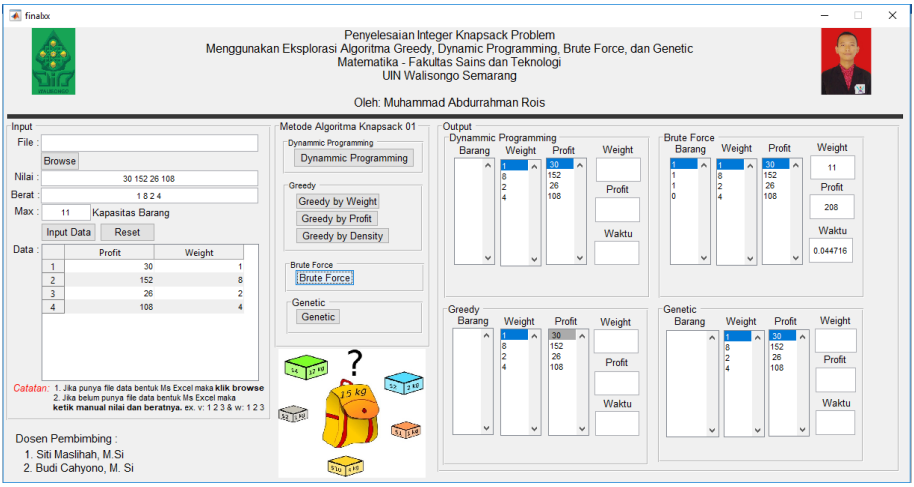
$w_i = [1, 1, 1, 1, 1, 1, 1, 1, 8, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,$
$1, 1, 1, 4, 1, 1, 1, 1, 1]$

(6) Masukan kapasitas maksimum bagian pertama $W = 11$ dan bagian kedua $W = 32$

Solusi optimal permasalahan *knapsack* dihitung menggunakan algoritma *brute force* diperoleh dari output program, untuk mempermudah membahas output dari perhitungan algoritma *brute* force. Oleh karena itu, di bawah ini disajikan ringkasan hasil perhitungannya:

**Tabel 4.15.** Ringkasan hasil output program algoritma *brute force* bagian pertama

| subset | total weight | total value |
|---|---|---|
| ∅ | 0 | 0 |
| {1} | 1 | 30 |
| {9} | 8 | 152 |
| {10} | 2 | 26 |
| {25} | 4 | 108 |
| {1,9} | 9 | 182 |
| {1,10} | 3 | 56 |
| {1,25} | 5 | 138 |
| {9,10} | 10 | 178 |
| {9,25} | 12 | Tidak layak |
| {10,25} | 6 | 134 |
| {1,9,10} | 11 | 208 |
| {1,9,25} | 13 | Tidak layak |
| {1,10,25} | 7 | 164 |
| {9,10,25} | 14 | Tidak layak |
| {1,9,10,25} | 15 | Tidak layak |




**Gambar 4.11.** *Hasil* perhitungan algoritma *brute force* bagian pertama

Hasil akhir di atas dapat disimpulkan bahwa barang yang dapat diangkut pada bagian pertama adalah barang 1, 9, 10 dan diperoleh hasil maksimal dengan keuntungan optimal sebesar Rp 208.000,- dengan berat 11 kg dan waktu komputasi 0,066716 detik. Sedangkan barang yang dapat diangkut pada bagian kedua yakni belum diperoleh hasil maksimal optimalnya, berat dan waktu. Karena data 30 barang tidak dapat diperoleh hasil karena waktu yang diperoleh sangat lama dengan batasan 100.000 detik tetapi dapat dipastikan bahwa hasil untuk keuntungan dan total berat sama dengan hasil dari *dynamic programming*, maka untuk bisa mengetahui perolehan

kira-kira waktunya dibutuhkan peramalan berdasarkan percobaan dengan data yang sudah ada dimisalkan dan diambil 6, 8, 10, 12, 14, 16, 18, 20, dan 22 barang dengan kapasitas sesuai jumlah barang yang diambil. Hasil perhitungan program algoritma *brute force* ditampilkan pada Lampiran 3. dan peramalan perkiraan waktu untuk 30 barang menggunakan metode peramalan *trend.* Penelitian yang sudah dilakukan oleh Yonhy, Goejantoro dan Wahyuningsih (2013) mengenai peramalan menggunakan metode *trend,* bahwa *trend* menggambarkan gerak data deret waktu selama jangka waktu yang panjang atau cukup lama dan berkecenderungan menuju satu arah yaitu bisa naik atau turun.

Metode *trend* yang digunakan dalam peramalan waktu ini adalah *trend linear, trend kuadratik* dan *trend eksponensial.* Langkah selanjutnya dilakukan perhitungan *Mean Square Error* (MSE) untuk mencari *trend* yang paling tepat dan memiliki kesalahan terkecil untuk dijadikan acuan peramalan. Setelah didapatkan *trend* yang paling tepat dilakukan uji *Mean Absolute Percent Error* (MAPE) untuk mengetahui ketepatan model yang didapatkan baik atau tepat dengan memiliki nilai

persentase 0%-30%. Setelah didapatkan model yang paling tepat, kemudian dilakukan peramalan data sebanyak 30 barang dengan membuat tabulasi data sebagai berikut:

**Tabel 4.16.** Tabulasi data menggunakan peramalan metode *trend*

| Barang | $Y$ | $X$ | $XY$ | $X^2$ | $X^2Y$ | $X^4$ | $logY$ | $XlogY$ |
|---|---|---|---|---|---|---|---|---|
| 6 | 0.7548 | -4 | -3.0192 | 16 | 12.0768 | 256 | -0.122168109 | 0.488672434 |
| 8 | 0.88325 | -3 | -2.64975 | 9 | 7.94925 | 81 | -0.053916354 | 0.161749062 |
| 10 | 3.5532 | -2 | -7.1064 | 4 | 14.2128 | 16 | 0.550619653 | -1.101239307 |
| 12 | 14.7866 | -1 | -14.7866 | 1 | 14.7866 | 1 | 1.169868325 | -1.169868325 |
| 14 | 60.7655 | 0 | 0 | 0 | 0 | 0 | 1.783657076 | 0 |
| 16 | 257.2313 | 1 | 257.2313 | 1 | 257.2313 | 1 | 2.410323813 | 2.410323813 |
| 18 | 1.179.1185 | 2 | 2358.237 | 4 | 4716.474 | 16 | 3.071557453 | 6.143114907 |
| 20 | 6.614.581 | 3 | 19843.743 | 9 | 59531.229 | 81 | 3.820502339 | 11.46150702 |
| 22 | 41.068.023 | 4 | 164272.092 | 16 | 657088.368 | 256 | 4.613503797 | 18.45401519 |
| **Jumlah** | **49,199.697150** | **0** | **186,703.741350** | **60.000000** | **721,642.327750** | **708.000000** | **17.243948** | **36.848275** |

Keterangan:
Y = Nilai waktu aktual
X = Periode/ waktu

Tabulasi data pada Tabel 4.16. akan ditentukan model persamaan matematika dan ketepatan model persamaan untuk metode *trend linear, trend kuadratik* dan *trend eksponensial* dimana perhitungannya ditampilkan pada Lampiran 12-14. Selanjutnya, dilakukan perhitungan MSE (Yonhy, Goejantoro dan Wahyuningsih, 2013):

$$MSE = \frac{\sum e^2}{n} \tag{4.1}$$

Keterangan:

$e$ = Galat/ *error*

$n$ = Jumlah data

(1) MSE *Trend Linear*

$$MSE = \frac{\sum e^2}{n} = \frac{881.867.468,9591260}{9} = 97.985.274,33$$

(2) MSE *Trend Kuadratik*

$$MSE = \frac{\sum e^2}{n} = \frac{378.763.989,3220020}{9} = 42.084.887,70$$

(3) MSE *Trend Eksponensial*

$$MSE = \frac{\sum e^2}{n} = \frac{306.656.950,8681270}{9} = 34.072.994,54$$

Perhitungan MSE di atas menunjukkan bahwa nilai MSE dari *trend eksponensial* merupakan yang terkecil. Jadi dapat diketahui bahwa *trend eksponensial* pada peramalan ini memiliki kecendrungan kesalahan yang paling rendah dibanding dengan *trend linear* dan *trend kuadratik.* Selanjutnya dilakukan perhitungan menggunakan MAPE berdasarkan Yonhy, Goejantoro dan Wahyuningsih (2013) untuk menguji ketepatan model yang disajikan sebagai berikut:

$$\begin{aligned} MAPE &= \left[ \frac{1}{n} \sum_{t=1}^{n} \frac{\left| Y_t - \hat{Y}_t \right|}{Y_t} \right] \times 100\% \\ &= \left[ \frac{1}{9} \left( \frac{18.044,4749}{49.199,697150} \right) \right] \times 100\% \\ &= 4.075109752 \end{aligned} \qquad (4.2)$$

Keterangan:

$Y_t$ = Nilai waktu aktual

$\hat{Y}_t$ = Nilai peramalan waktu

Hasil di atas terlihat bahwa nilai MAPE untuk model *trend eksponensial* sebesar 4,075109752% dan nilai MAPE tersebut < 30%. Jadi metode peramalan *trend eksponensial* pada masalah ini merupakan model yang baik dan tepat untuk meramalkan waktu dengan jumlah barang sebanyak 30 barang menggunakan algoritma *brute force.*

Jadi diperoleh model Y eksponensial yaitu $\hat{Y} = a \times b^{X} = 82,41271494 \times 4,112803052^{X}$ berdasarkan perhitungan yang ditampilkan pada Lampiran 14. sehingga peramalan untuk data 30 barang sebagai berikut:

**Tabel 4.17.** Hasil nilai peramalan

| Barang | Waktu Peramalan |
|---|---|
| 24 | 96.980,36 |
| 26 | 398.861,2 |
| 28 | 1.640.437,22 |
| 30 | 6.746.795,19 |

Hasil peramalan di atas, didapatkan waktu komputasi untuk 30 barang sebanyak 6.746.795,19 detik. Hasil waktu komputasi berdasarkan peramalan di atas, menunjukan bahwa semakin banyak data barang maka semakin bertambah lamanya bahkan jika dibatasi dengan waktu yang ditentukan hasil tidak akan diketahui sesuai dengan landasan teori yang sudah dijelaskan. Oleh karena itu, walaupun hasilnya selalu optimum tetapi waktunya

tidak efisien maka algoritma *brute force* tidak disarankan digunakan untuk *integer knapsack problem* dengan data kompleks.

**d) Penyelesaian Menggunakan Algoritma *Genetic***

Peneliti mencoba menyelesaikan menjadi 2 bagian. Bagian pertama yakni menyelesaikan dengan mengambil 4 data dengan berat yang berbeda (barang 1, 9, 10, 25) dan bagian kedua yakni semua barang untuk diselesaikan. Perhitungan menggunakan algoritma *genetic*, terlebih dahulu menginisialisasi parameter-parameter awal yang diperlukan, yaitu:

Jumlah generasi bagian 1 dan 2 ( $jg = 100$ )

Jumlah populasi bagian 1 dan 2 ( $pop = 4$ dan $30$ )

Probabilitas mutasi ( $pm = 0,01$ )

Populasi awal berjumlah 4 dan 30 buah barang dan dibangkitkan secara acak menggunakan program. Hasil dari 4 dan 30 buah populasi awal tampak sebagai berikut:

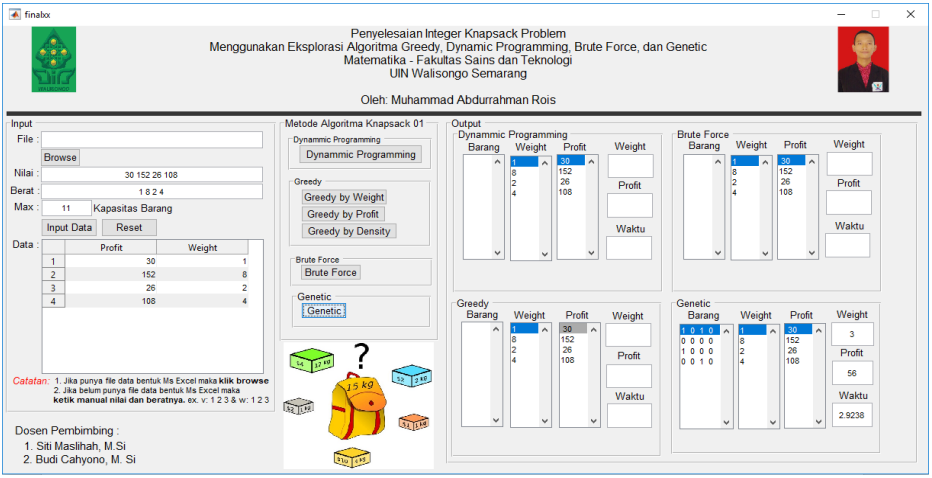
1. Bagian pertama (4 barang)

$v_1 = 0\ 0\ 0\ 1$
$v_2 = 1\ 0\ 0\ 1$
$v_3 = 1\ 0\ 1\ 1$
$v_4 = 1\ 1\ 1\ 1$

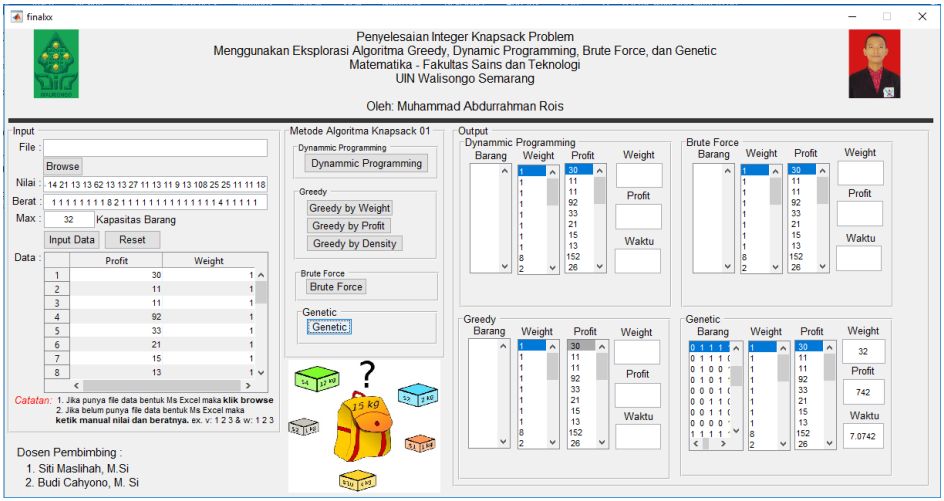
2. Bagian kedua (30 barang)

$v_1 = 0\,1\,1\,1\,0\,0\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1$
$v_2 = 1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,0\,1\,0\,0$
$v_3 = 1\,0\,1\,0\,0\,1\,0\,1\,1\,1\,0\,0\,1\,1\,0\,1\,1\,1\,0\,1\,1\,1\,1\,1\,0\,0\,0\,1\,0\,0$
$v_4 = 0\,1\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1$
$v_5 = 1\,0\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,0\,1$
$v_6 = 0\,1\,1\,1\,1\,1\,1\,0\,0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,1$
$v_7 = 0\,1\,1\,1\,0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,0\,1\,1\,0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1$
$v_8 = 0\,1\,0\,0\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,1\,0\,1\,0$
$v_9 = 0\,1\,0\,1\,1\,1\,0\,0\,1\,0\,1\,1\,1\,1\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0$
$v_{10} = 0\,0\,0\,1\,0\,0\,0\,1\,1\,0\,0\,1\,0\,0\,1\,0\,1\,1\,1\,0\,1\,1\,0\,0\,1\,1\,0\,1\,0\,1$
$v_{11} = 0\,0\,1\,1\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,1$
$v_{12} = 0\,0\,1\,1\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,0\,0\,1\,1\,0\,1\,1\,0\,1\,0\,1\,1$
$v_{13} = 0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,1\,0\,0\,0\,1\,0\,1\,0\,0\,1\,1\,1\,0\,0\,1\,0\,0\,0\,1\,0$
$v_{14} = 1\,1\,1\,1\,1\,1\,0\,0\,0\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,0\,0\,0$
$v_{15} = 1\,0\,1\,0\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,1\,1\,1\,1\,0$
$v_{16} = 0\,1\,1\,0\,1\,1\,1\,1\,1\,1\,0\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,0$
$v_{17} = 1\,1\,0\,0\,1\,0\,1\,0\,0\,0\,1\,1\,1\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,1\,1\,0\,1\,1$
$v_{18} = 1\,0\,1\,1\,1\,1\,1\,1\,1\,1\,0\,1\,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0$
$v_{19} = 1\,0\,1\,1\,1\,1\,0\,0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,1\,0\,0\,1\,0\,1\,1\,0\,0\,0\,1\,0\,0$
$v_{20} = 1\,0\,1\,0\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,0\,1\,0\,1\,1\,0$
$v_{21} = 0\,1\,0\,1\,1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,1\,0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,0$
$v_{22} = 1\,0\,1\,0\,1\,1\,0\,0\,1\,1\,0\,1\,1\,0\,1\,0\,0\,1\,1\,0\,0\,1\,0\,0\,1\,1\,1\,1\,0\,0$
$v_{23} = 1\,1\,1\,1\,0\,0\,1\,0\,1\,0\,0\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,0\,1\,0\,1$
$v_{24} = 0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,0\,1\,0$
$v_{25} = 1\,1\,1\,1\,0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1\,1\,0\,1\,0\,0\,1\,0\,0\,0\,1$
$v_{26} = 0\,0\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0$
$v_{27} = 1\,1\,1\,1\,0\,0\,1\,1\,0\,1\,1\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,1\,1\,0$
$v_{28} = 0\,0\,0\,1\,0\,1\,1\,1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,0\,1\,0\,0$
$v_{29} = 0\,1\,1\,1\,0\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1\,0$
$v_{30} = 1\,0\,1\,1\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,1\,1\,0\,0\,0\,1$

Solusi optimal permasalahan *knapsack* disajikan pada Gambar sebagai berikut:




**Gambar 4.12.** Hasil perhitungan algoritma *genetic* bagian pertama




**Gambar 4.13.** Hasil perhitungan algoritma *genetic* bagian kedua

Ringkasan hasil perhitungan dari program algoritma *genetic* disajikan sebagai berikut:

a. Bagian pertama

$v_x = 1\ 0\ 1\ 0$

**Catatan**: $v_x$ dimana $x = (1,\ 2,\ 3,\ 4)$

b. Bagian kedua

$v_{29} = 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1$

$v_{30} = 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1$

$v_y = 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1$

**Catatan**: $v_y$ dimana $y = (1,\ 2,\ 3,\ \ldots,\ \ldots.,\ 30)$ kecuali $v_{29}$

Hasil akhir tersebut dapat disimpulkan bahwa total berat dan total keuntungan menggunakan algoritma *genetic* yang dihasilkan adalah:

a. Bagian pertama

Total berat dan keuntungan dari $v_x$ sebanyak 3 kg dan Rp 56.000,-

b. Bagian kedua

Total berat dan keuntungan dari $v_{29}$ sebanyak 31 kg dan Rp 740.000,-, dari $v_{30}$ sebanyak 33 kg karena melebihi kapasitas, maka tidak dipilih. Dan selanjutnya $v_y$ sebanyak 32 kg dan Rp 742.000,-. Karena nilai paling optimum dan tidak melebihi

kapasitas terdapat pada $v_y$ maka terpilih sebagai solusi.

Hasil yang diperoleh menggunakan algoritma *genetic,* dapat disimpulkan bahwa total berat dan keuntungan yang paling optimal pada bagian pertama adalah 3 kg dan Rp56 .000,-, karena memiliki keuntungan yang terbesar. Dan hasil yang diperoleh pada bagian kedua bahwa total berat dan keuntungan yang paling optimal adalah 32 kg dan Rp 742.000,-. Hasil bagian pertama dijelaskan bahwa barang yang dapat diangkut adalah barang 1, 10 dengan total berat 3 kg dan total keuntungan yang didapat Rp 56.000,- serta waktu komputasi yang dibutuhkan 2,9238 detik. Sedangkan barang yang dapat diangkut pada bagian kedua adalah barang 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 16, 17, 19, 21, 24, 25, 26, 27, 28, 30 dan diperoleh hasil maksimal dengan keuntungan optimal sebesar Rp 742.000,- dengan berat 32 kg dan waktu komputasi 7,0742 detik.

## B. Pembahasan

Peneliti menyelesaikan masalah *integer knapsack* dari data yang diperoleh melalui pengamatan di J&T Express *drop point Ngaliyan* yang ditunjukkan pada Tabel 4.1. menggunakan 4 algoritma yaitu algoritma *greedy, dynamic programming, brute force* dan *genetic.* Penelitian yang sudah dilakukan terbagi menjadi 2 bagian yaitu bagian pertama (data kecil dengan 4 barang) dan bagian kedua (data besar dengan 30 barang).

**Tabel 4.18.** Hasil perhitungan bagian pertama

| **Bagian Pertama (Data barang sedikit)** | | | |
|---|---|---|---|
| Metode | Nomor barang yang diambil | Total berat (kg) | Total nilai/ keuntungan (Rp) |
| 1. Algoritma *greedy* | | | |
| a. *Greedy by profit* | 9 | 8 | 152.000 |
| b. *Greedy by weight* | 1, 10, 25 | 7 | 164.000 |
| c. *Greedy by density* | 1, 25 | 5 | 138.000 |
| 2. Algoritma *dynamic programming* | 1, 9, 10 | 11 | 208.000 |
| 3. Algoritma *brute force* | 1, 9, 10 | 11 | 208.000 |
| 4. Algoritma *genetic* | 1, 10 | 3 | 56.000 |

Berdasarkan hasil penyelesaian bagian pertama yang ditampilkan pada Tabel 4.18. menunjukkan hasil yang paling optimal adalah menggunakan algoritma *dynamic programming* dan *brute force* untuk diterapkan pada *integer knapsack problem* dengan jumlah barang yang sedikit.

**Tabel 4.19.** Hasil perhitungan bagian kedua

| Bagian kedua (Data barang banyak) | | | |
|---|---|---|---|
| Metode | Nomor barang yang diambil | Total berat (kg) | Total nilai/ keuntungan (Rp) |
| 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 |
| 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 |
| 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 |
| 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 |
| 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 |
| 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 |

Berdasarkan hasil penyelesaian bagian kedua yang ditampilkan pada Tabel 4.19. menunjukkan hasil Pada perhitungan algoritma *brute force* belum bisa dilihat untuk hasilnya karena waktu yang dibutuhkan sangat lama dengan batasan 100.000 detik, tetapi dapat dipastikan bahwa hasilnya akan sama dengan hasil penyelesaian algoritma *dynamic programming.* Sebenarnya kemungkinan dari beberapa jenis barang memiliki kesempatan yang sama untuk diangkut atau tidak jika memiliki berat yang sama. Jika dibandingkan dengan barang yang lain, ada beberapa barang yang memiliki bobot yang sama dengan keuntungan yang lebih besar sehingga kesempatan untuk diangkut lebih besar. Hasil yang diperoleh dari algoritma *genetic* tidak stabil karena

populasinya random setiap proses perhitungan, karena dipengaruhi oleh tahap inisialisasi kromosom yang membangkitkannya secara random/ acak. Penyelesaian pada bagian kedua, menunjukkan hasil paling optimal menggunakan algoritma *greedy* menggunakan konsep *greedy by profit* dan *greedy by density, dynamic programming* dan *brute force* untuk data besar pada masalah *integer knapsack.* Akan tetapi, algoritma *greedy* secara umum belum mendapatkan hasil yang maksimal.

Hasil dari penyelesaian pada bagian pertama dan kedua dapat disimpulkan bahwa algoritma yang menghasilkan hasil yang optimal stabil dengan data kecil ataupun data besar untuk *integer knapsack problem* adalah algoritma *dynamic programming.*

Berbeda halnya jika ditinjau dari perhitungan kompleksitas waktu komputasi (dalam detik) yang dibutuhkan keempat algoritma tersebut. Waktu komputasi yang dibutuhkan keempat algoritma tersebut untuk menyelesaikan dengan jumlah data bagian pertama sebanyak $n = 4$ dan kapasitas maksimal berat $W = 11$, sedangkan bagian kedua sebanyak $n = 30$ dan kapasitas maksimal berat $W = 32.$

**Tabel 4.20.** Waktu komputasi dalam proses penyelesaian

| Waktu Komputasi Penyelesaian | | |
|---|---|---|
| Metode | Waktu komputasi bagian pertama (detik) | Waktu komputasi bagian kedua (detik) |
| 1. Algoritma *greedy* | | |
| a. *Greedy by profit* | 0,17667 | 0,34943 |
| b. *Greedy by weight* | 0,2015 | 0,21119 |
| c. *Greedy by density* | 0,3684 | 0,2709 |
| 2. Algoritma *dynamic programming* | 1,0732 | 2,8999 |
| 3. Algoritma *brute force* | 0,066716 | 6.746.795,19 |
| 4. Algoritma *genetic* | 2,9238 | 7,0742 |

Waktu komputasi yang dihasilkan untuk menyelesaikan 4 algoritma ditampilkan pada Tabel 4.20. dan catatan untuk penyelesaian algoritma *brute force* bagian kedua dengan batasan waktu sebanyak 100.000 detik belum dapat dilihat, karena memiliki waktu komputasi sangat besar sehingga dilakukan peramalan untuk mengetahui lama waktu yang dibutuhkan untuk penyelesaian 30 barang. Oleh karena itu, pada bagian pertama algoritma *brute force* lebih efisien untuk menyelesaikan *integer knapsack problem.* Tetapi pada bagian kedua, algoritma *brute force* tidak efisien karena waktu yang dibutuhkan sangat lama. Jadi algoritma *brute force* lebih efisien jika digunakan pada data yang sedikit untuk menyelesaikan *integer knapsack problem.* Sedangkan pada bagian kedua, metode algoritma *greedy* khususnya *greedy by weight* lebih efisien untuk menyelesaikan *integer knapsack problem.*

Hasil keempat algoritma pada penelitian *integer knapsack problem*, menunjukan bahwa menyelesaikan masalah bagian pertama yaitu dengan jumlah barang sedikit ternyata algoritma *dynamic programming* dan *brute force* lebih optimal dalam memperoleh keuntungan. Sedangkan jika ditinjau dari waktu pengerjaannya pada data kecil, algoritma *brute force* memiliki kompleksitas waktu komputasi yang lebih efisien. Selanjutnya, pada masalah bagian kedua yaitu dengan jumlah barang banyak ternyata algoritma *dynamic programming* dan *brute force* lebih optimal dalam memperoleh keuntungan. Sedangkan jika ditinjau dari waktu pengerjaannya, algoritma *brute force* sangat lama untuk bagian kedua karena data barang bagian kedua banyak dan algoritma *greedy* yang memiliki kompleksitas waktu komputasi yang lebih efisien. Oleh karena itu, algoritma yang efektif dan efisien untuk diterapkan pada topik *integer knapsack problem* adalah algoritma *dynamic programming.*

# BAB V

# PENUTUP

## A. Kesimpulan

Berdasarkan data, hasil dan pembahasan yang telah dilakukan pada bab sebelumnya, maka dapat diperoleh kesimpulan dari penyelesaian *integer knapsack problem* sebagai berikut:

a) Penyelesaian *integer knapsack problem* menggunakan algoritma *greedy* yang terdiri dari 3 konsep yaitu *greedy by profit, greedy by weight* dan *greedy by density* ditampilkan pada Tabel 5.1 sebagai berikut:

**Tabel 5.1.** Penyelesaian algoritma *greedy*

| Algoritma ***Greedy*** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Metode | | Nomor barang yang diambil | | Total berat (kg) | | Total nilai/ keuntungan (Rp) | | Waktu komputasi (detik) | |
| | | Bagian pertama | Bagian kedua | Bagian pertama | Bagian kedua | Bagian pertama | Bagian kedua | Bagian pertama | Bagian kedua |
| a. | *Greedy by profit* | 9 | 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 25, 26, 27, 30 | 8 | 32 | 152.000 | 747.000 | 0,17667 | 0,34943 |
| b. | *Greedy by weight* | 1, 10, 25 | 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 | 7 | 29 | 164.000 | 588.000 | 0,2015 | 0,21119 |
| c. | *Greedy by density* | 1, 25 | 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 25, 26, 27, 30 | 5 | 32 | 138.000 | 747.000 | 0,3684 | 0,2709 |

b) *Penyelesaian integer knapsack problem* menggunakan algoritma *dynamic programming* ditampilkan pada Tabel 5.2 sebagai berikut:

**Tabel 5.2.** Penyelesaian algoritma *dynamic programming*

| Algoritma ***Dynamic Programming*** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Metode | Nomor barang yang diambil | | Total berat (kg) | | Total nilai/ keuntungan (Rp) | | Waktu komputasi (detik) | |
| | Bagian pertama | Bagian kedua | Bagian pertama | Bagian kedua | Bagian pertama | Bagian kedua | Bagian pertama | Bagian kedua |
| Algoritma *dynamic programming* | 1, 9, 10 | 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 25, 26, 27, 30 | 11 | 32 | 208.000 | 747.000 | 1,0732 | 2,8999 |

c) Penyelesaian *integer knapsack problem* menggunakan algoritma *brute force* ditampilkan pada Tabel 5.3 sebagai berikut:

**Tabel 5.3.** Penyelesaian algoritma *brute force*

| Algoritma *Brute Force* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Metode | Nomor barang yang diambil | | Total berat (kg) | | Total nilai/ keuntungan (Rp) | | Waktu komputasi (detik) | |
| | Bagian pertama | Bagian kedua | Bagian pertama | Bagian kedua | Bagian pertama | Bagian kedua | Bagian pertama | Bagian kedua |
| Algoritma *brute force* | 1, 9, 10 | 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 25, 26, 27, 30 | 11 | 32 | 208.000 | 747.000 | 0,066716 | 6.746.795,19 |

d) Penyelesaian *integer knapsack problem* menggunakan algoritma *genetic* ditampilkan pada Tabel 5.4 sebagai berikut:

**Tabel 5.4.** Penyelesaian algoritma *genetic*

| Algoritma *Genetic* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Metode | Nomor barang yang diambil | | Total berat (kg) | | Total nilai/ keuntungan (Rp) | | Waktu komputasi (detik) | |
| | Bagian pertama | Bagian kedua | Bagian pertama | Bagian kedua | Bagian pertama | Bagian kedua | Bagian pertama | Bagian kedua |
| Algoritma *genetic* | 1, 10 | 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 16, 17, 19, 21, 24, 25, 26, 27, 28, 30 | 3 | 32 | 56.000 | 742.000 | 2,9238 | 7,0742 |

e) Hasil perhitungan keempat algoritma tersebut, pada bagian pertama dapat disimpulkan bahwa algoritma *dynamic programming* dan *brute force* menghasilkan keuntungan yang optimum tetapi pada algoritma *dynamic programming* memiliki waktu komputasi yang lebih besar daripada algoritma *brute force* dan *greedy*, sedangkan pada algoritma *brute force* waktu komputasi lebih kecil dibandingkan dengan algoritma yang lain. Hasil perhitungan pada bagian kedua dapat disimpulkan bahwa algoritma *dynamic programming* menghasilkan keuntungan yang optimum tetapi pada algoritma

*dynamic programming* memiliki waktu komputasi yang lebih besar daripada algoritma *greedy* dan algoritma *brute force* optimum tetapi dengan jumlah barang yang banyak maka waktu yang dihasilkan juga semakin banyak bahkan jika dibatasi dengan jumlah waktu tertentu maka akan tidak diketemukan hasilnya, jadi algoritma *brute force* tidak efektif jika digunakan untuk data yang banyak. Penyelesaian algoritma *genetic* solusinya optimum, tetapi hasilnya tidak stabil dengan nilai yang dihasilkan pertama kali karena dipengaruhi inisialisasi kromosom yang dilakukan secara random.

## B. 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 juga dapat menambah atau merubah variabel sesuai permasalahan yang ada. Atau bisa menggunakan permasalahan yang berbeda atau sudah dikembangkan seperti *unbounded knapsack problem* guna menyesuaikan permasalahan pada realita yang ada.

## DAFTAR PUSTAKA


Cormen, T. H. *et al.* 2001. *Introduction to Algorithms*. Second Edi, *The MIT Press*. Second Edi. Cambridge, Massachusetts London, England: The MIT Press.

Escobar, F. A. *et al.* 2017. 'Scalable shared-memory architecture to solve the Knapsack 0/1 problem', *Microprocessors and Microsystems*. Elsevier B.V., 50, pp. 189–201. doi: 10.1016/j.micpro.2017.04.001.

Fanggidae, A. and Lado, F. R. 2015. *Algoritma Genetika dan Penerapannya*. TEKNOSAIN.

Ghozali, A. E., Setiawan, B. D. and Furqon, M. T. 2017. 'Aplikasi Perencanaan Wisata di Malang Raya dengan Algoritma Greedy', 1(12), pp. 1459–1467.

Goyal, S. and Parashar, A. 2016. 'A Proposed Solution to Knapsack Problem Using Branch & Bound Technique', *International Journal for Innovative Research in Multidisciplinary Field*, 2(7), pp. 240–246.

Juvianto, A. and Agung, H. 2017. 'Implementasi Algoritma Greedy pada Pencarian Langkah Optimal Permainan Mahjong Solitaire', *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 1(3), pp. 226–231. doi: https://doi.org/10.29207/resti.v1i3.58.

Juwita, P. S., Susanto, E. and Halomoan, J. 2017, 'Perancangan dan Implementasi Manajemen Daya Listrik Menggunakan Algoritma Greedy untuk Otomatisasi Rumah', in *e-Proceeding of Engineering*, pp. 1512–1519.

Kellerer, H., Pferschy, U. and Pisinger, D. 2004. *Knapsack Problems*, *Knapsack Problems*. Springer-Verlag Berlin Heidelberg New York. doi: 10.1007/978-3-540-24777-7_9.

Kwarteng, A. and 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), pp. 46–62. doi: 10.9790/9622-0704044662.

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

Lin, B. *et al.* .2017. 'Modeling the 0-1 Knapsack Problem in Cargo Flow Adjustment', *Symmetry*, 9(7), p. 118. doi: 10.3390/sym9070118.

Lu, Y. nd. *0-1 Knapsack Problem*, *University of Nebraska-Lincoln*. Available at: https://www.google.com/url?sa=t&source=web&rct=j&url=http://cse.unl.edu/~ylu/raik283/notes/0-1-knapsack.ppt&ved=2ahUKEwiX8YKPq7LfAhXWinAKHWT0CNwQFjAKegQIAhAB&usg=AOvVaw0vFxwP_t7hBszJKH4MsjZ8.

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

Nuraeni. 2016. *Jasa Logistik Melesat di Era e-Commerce*, *KOMINFO: Kategori sorotan media*. Available at: https://www.kominfo.go.id/index.php/content/detail/6707/Jasa+Logistik+Melesat+di+Era+e-Commerce+/0/sorotan_media.

Pan, X. and Zhang, T. 2018. 'Comparison and Analysis of Algorithms for the 0/1 Knapsack Problem', *IOP Journal of Phisics*, pp. 1–8. doi: 10.1088/1742-6596/1069/1/012024.

Parkinson, A. R., Balling, R. and Hedengren, J. D. 2013 'Optimization Methods for Engineering Design', *Brigham Young University*, p. 18. doi: 10.1002/9780470686652.eae495.

Paryati. 2009. 'Optimasi Strategi Algoritma Greedy untuk Menyelesaikan Permasalahan Knapsack 0-1', *Electronic Notes in Theoretical Computer Science*, 108(5), pp. 101–110. doi: 10.1016/j.entcs.2004.01.013.

Rahardjo, M. 2017. *Studi Kasus dalam Penelitian Kualitatif:*

*Konsep dan Prosedurnya*.

Ralp. nd. *1- Lecture notes of Prof. Dr. ir. RHJM (Ralph) Otten*. Available at: https://www.google.com/url?sa=t&source=web&rct=j&url=http://tuvalu.santafe.edu/~aaronc/courses/5454/csci5454_spring2013_CSL2&ved=2ahUKEwiAxNTdpLLfAhWMeisKHWiPCqgQFjAAegQIBxAB&usg=AOvVaw3jwa06ofF-nmJEqOqmv5MM&cshid=1545442437471.

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

Siang, J. J. 2014. *Riset Operasi dalam Pendekatan Algoritmis*. Edisi 2. ANDI Yogyakarta.

Suarga. 2015. *Komputasi Numerik Pemrograman MATLAB untuk Metoda Numerik*. Yogyakarta: Penerbit ANDI.

Supranto, J. 1988. *Riset Operasi untuk Pengambilan Keputusan*. Jakarta: Penerbit UI (UI-Press).

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

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

Thada, V. and Dhaka, S. 2014. 'Genetic Algorithm based Approach to Solve Non Fractional (0/1) Knapsack Optimization Problem', *International Journal of Computer Applications*, 100(15), pp. 21–26.

Tim Baitul Hikmah Jogjakarta. 2014. *Ensiklopedia Pengetahuan Al-Qur'an dan Hadits*. Kamil Pustaka.

Vala, J., Monaka, D. and Pandya, J. nd. 'Comparative Analysis of Dynamic and Greedy Approaches for Dynamic Programming', pp. 5–8. Available at: www.ijmter.com.

Yonhy, Y., Goejantoro, R. and Wahyuningsih, S. 2013. 'Metode

Trend Non Linear Untuk Forecasting Jumlah Keberangkatan Tenaga Kerja Indonesia Di Kantor Imigrasi Kelas II Kabupaten Nunukan', *Jurnal EKSPONENSIAL*, 4(1), pp. 47–54.

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

## LAMPIRAN

1. Ijin penelitian dengan karyawan J&T Express *drop point* Ngaliyan kota Semarang


2. Observasi dengan karyawan J&T Express *drop point* Ngaliyan kota Semarang

3. Surat penunjukan dosen pembimbing

KEMENTERIAN AGAMA RI
UNIVERSITAS ISLAM NEGERI WALISONGO
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Prof. Dr. Hamka Ngaliyan, Semarang 50185 Telp. 024-7601295, Fax. 024-7615387

Semarang, 8 Juni 2018

Nomor : B-2007/Un.10.8/J1/PP.00.9/06/2018
Lamp : -
Hal : **Penunjukan Pembimbing Skripsi**

Kepada Yth:
1. Siti Maslihah, M.Si
2. Budi Cahyono, M.Si

Di Semarang

***Assalamu'alaikum Wr. Wb.***

Berdasarkan hasil pembahasan usulan judul penelitian di Program Studi Matematika, maka Fakultas Sains dan Teknologi menyetujui judul skripsi mahasiswa:

Nama : Muhammad Abdurrahman Rois
NIM : 150 804 6022
Judul : **Penyelesaian *Integer Knapsack Problem* Menggunakan Eksplorasi Algoritma *Greedy, Dynamic Programming, Brute Force* dan *Genetic***

Sehubungan dengan hal tersebut kami menunjuk saudara:
1. **Siti Maslihah, M.Si** sebagai Pembimbing I
2. **Budi Cahyono, M.Si** sebagai Pembimbing II

Demikian penunjukan pembimbing skripsi ini disampaikan dan atas kerjasama yang diberikan kami ucapkan terima kasih.

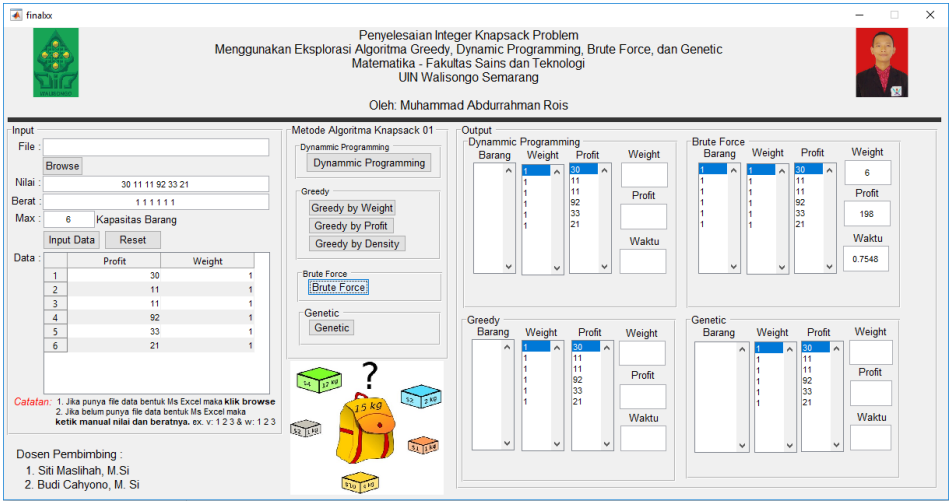
***Wassalamu'alaikum Wr. Wb.***

A.n Dekan
Ketua Program Studi Matematika

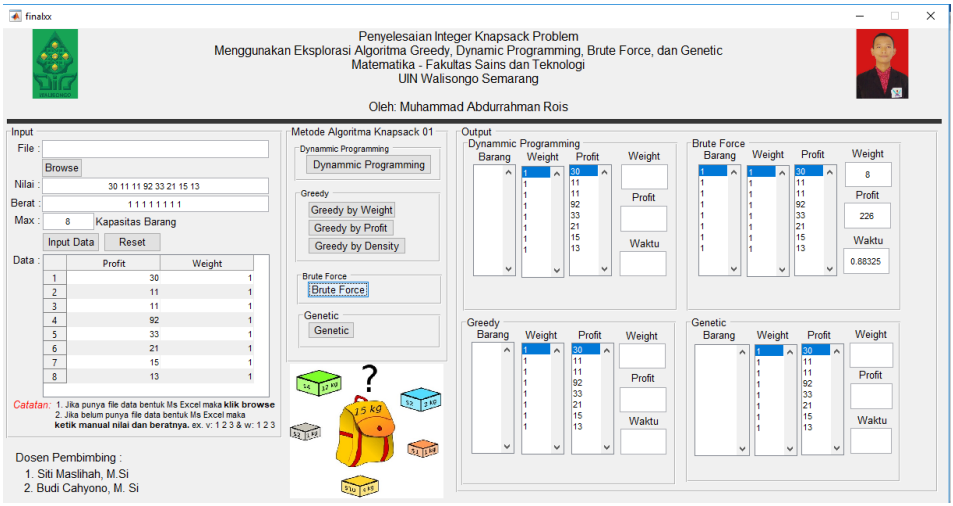
Dra. Siswanah, M.Sc
NIP. 19870202 201101 2 014

Tembusan:
1. Dekan Fakultas Sains dan Teknologi UIN Walisongo sebagai laporan
2. Mahasiswa yang bersangkutan
3. Arsip

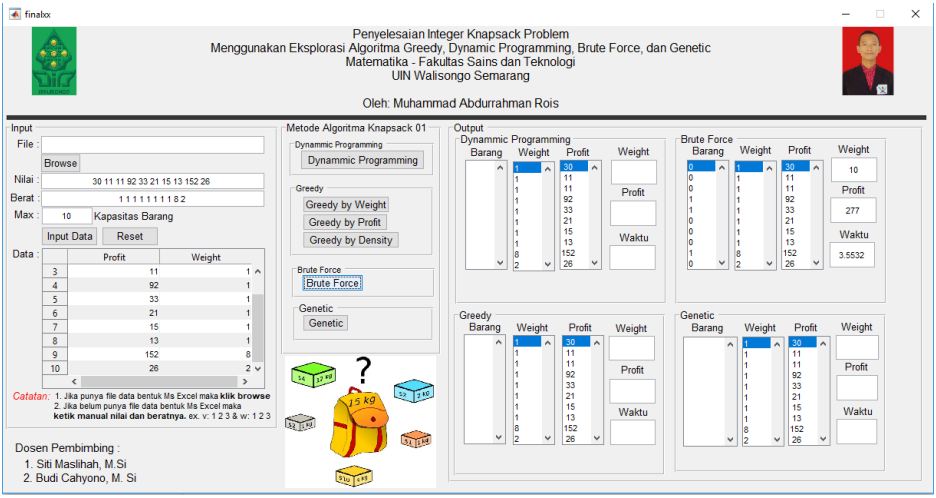
4. Hasil perhitungan algoritma *brute force* 6 barang



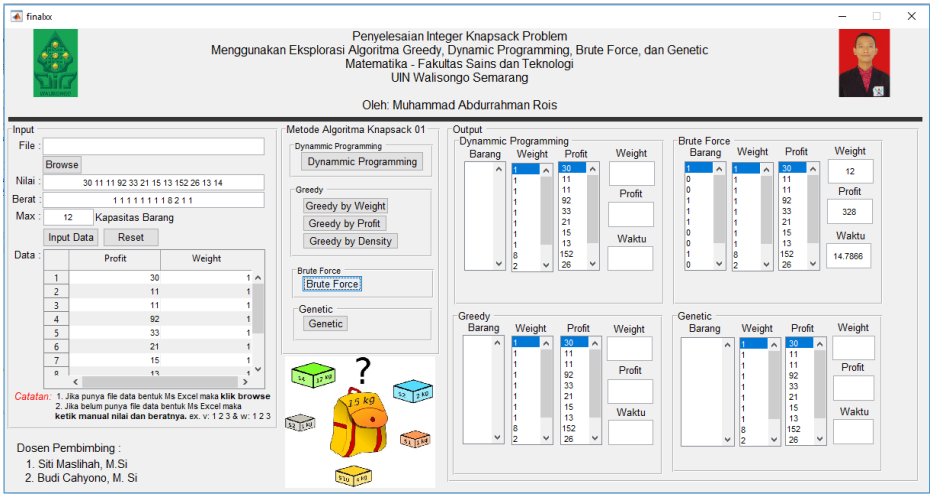

5. Hasil perhitungan algoritma *brute force* 8 barang



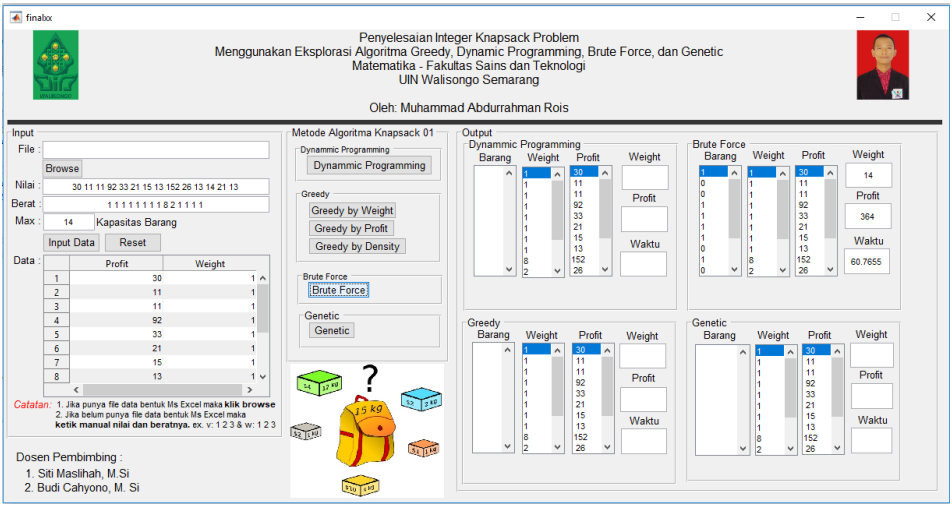
6. Hasil perhitungan algoritma *brute force* 10 barang



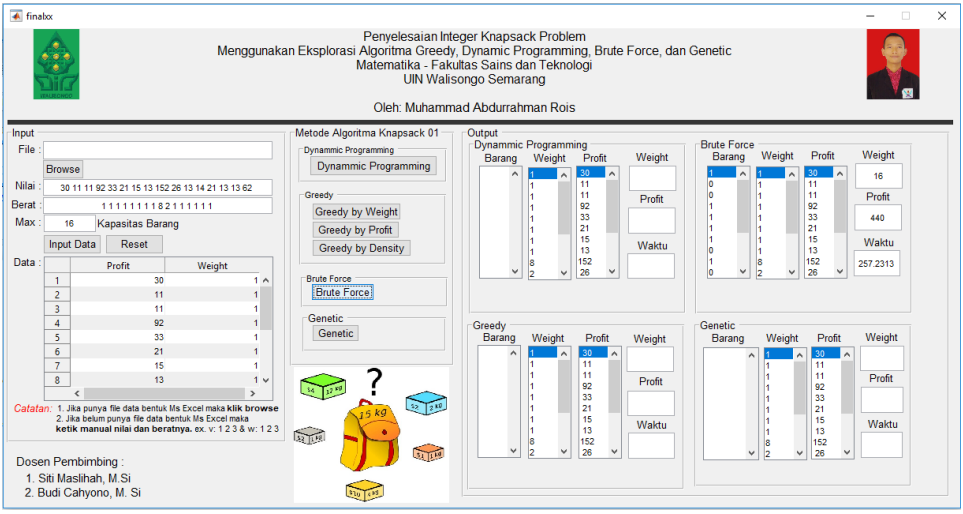

7. Hasil perhitungan algoritma *brute force* 12 barang



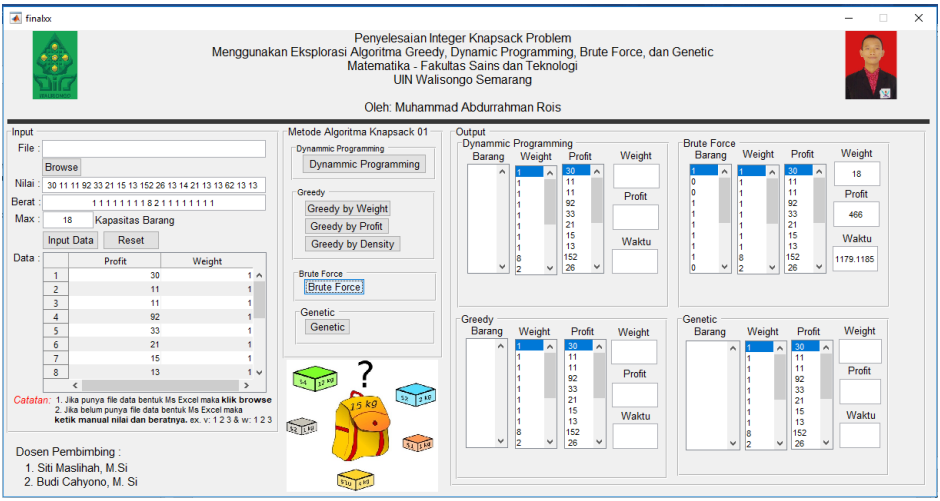
8. Hasil perhitungan algoritma *brute force* 14 barang



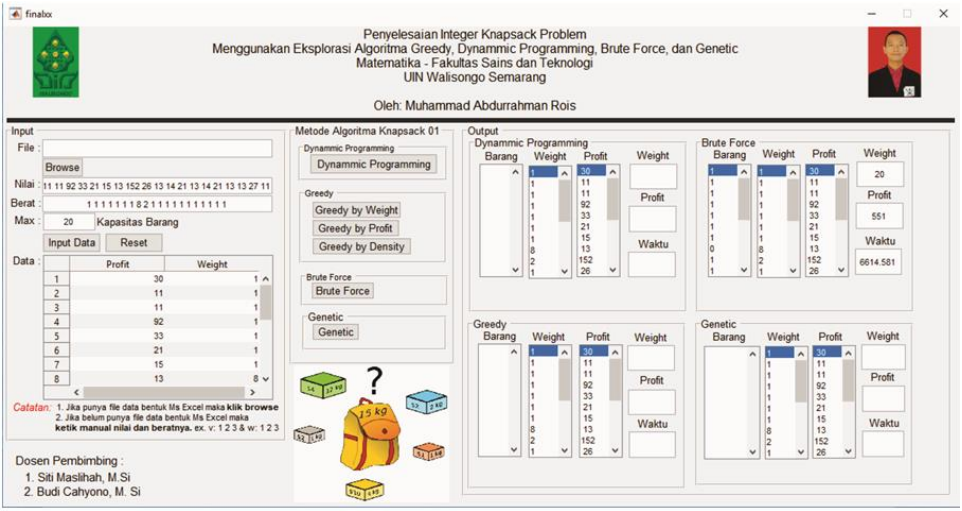

9. Hasil perhitungan algoritma *brute force* 16 barang



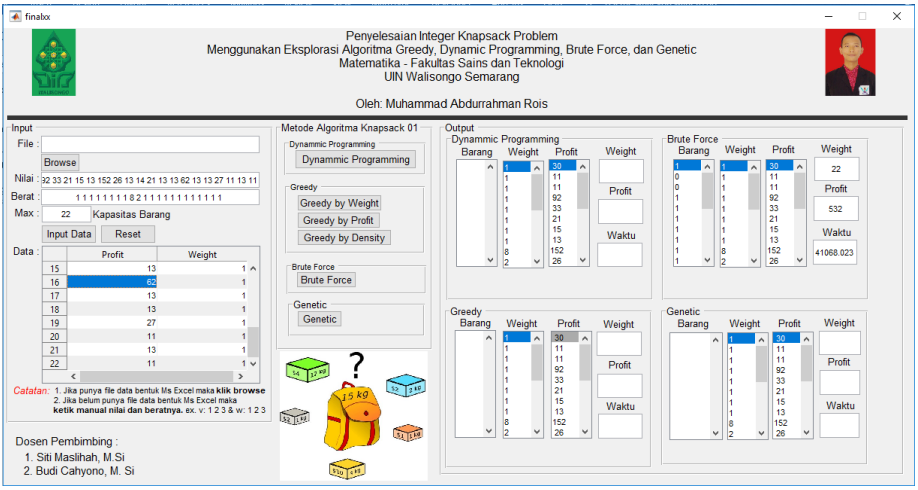
10. Hasil perhitungan algoritma *brute force* 18 barang




11. Hasil perhitungan algoritma *brute force* 20 barang



12. Hasil *perhitungan* algoritma *brute force* 22 barang




13. Peramalan dengan metode *trend linear*

| Barang | $Y$ | $X$ | $XY$ | $X^2$ | $X^2Y$ | $X^4$ | $\log Y$ | $X \log Y$ | $\hat{Y}$ | $e$ | $e^2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 0.7548 | -4 | -3.0192 | 16 | 12.0768 | 256 | -0.122168109 | 0.488672434 | -6980.283073 | 6,981.0379 | 48,734,889.7889144 |
| 8 | 0.88325 | -3 | -2.64975 | 9 | 7.94925 | 81 | -0.053916354 | 0.161749062 | -3868.554051 | 3,869.4373 | 14,972,545.0250804 |
| 10 | 3.5532 | -2 | -7.1064 | 4 | 14.2128 | 16 | 0.550619653 | -1.101239307 | -756.8250283 | 760.3782 | 578,175.0501233 |
| 12 | 14.7866 | -1 | -14.7866 | 1 | 14.7866 | 1 | 1.169868325 | -1.169868325 | 2354.903994 | -2,340.1174 | 5,476,149.4184814 |
| 14 | 60.7655 | 0 | 0 | 0 | 0 | 0 | 1.783657076 | 0 | 5466.633017 | -5,405.8675 | 29,223,403.6077518 |
| 16 | 257.2313 | 1 | 257.2313 | 1 | 257.2313 | 1 | 2.410323813 | 2.410323813 | 8578.362039 | -8,321.1307 | 69,241,216.7783044 |
| 18 | 1,179.1185 | 2 | 2358.237 | 4 | 4716.474 | 16 | 3.071557453 | 6.143114907 | 11690.09106 | -10,510.9726 | 110,480,544.1921100 |
| 20 | 6,614.581 | 3 | 19843.743 | 9 | 59531.229 | 81 | 3.820502339 | 11.46150702 | 14801.82008 | -8,187.2391 | 67,030,883.8213063 |
| 22 | 41,068.023 | 4 | 164272.092 | 16 | 657088.368 | 256 | 4.613503797 | 18.45401519 | 17913.54911 | 23,154.4739 | 536,129,661.2770550 |
| Jumlah | 49,199.697150 | 0 | 186,703.741350 | 60 | 721,642.327750 | 708 | 17.243948 | 36.848275 | 49,199.697150 | 0.0000 | 881,867,468.9591260 |

14. Peramalan dengan metode *trend kuadratik*

| Barang | $Y$ | $X$ | $XY$ | $X^2$ | $X^2Y$ | $X^4$ | $\log Y$ | $X \log Y$ | $\hat{Y}$ | $e$ | $e^2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 0.7548 | -4 | -3.0192 | 16 | 12.0768 | 256 | -0.122168109 | 0.488672434 | 4948.333495 | -4,947.5787 | 24,478,534.9417187 |
| 8 | 0.88325 | -3 | -2.64975 | 9 | 7.94925 | 81 | -0.053916354 | 0.161749062 | -886.3999088 | 887.2832 | 787,271.4038686 |
| 10 | 3.5532 | -2 | -7.1064 | 4 | 14.2128 | 16 | 0.550619653 | -1.101239307 | -4165.001191 | 4,168.5544 | 17,376,845.7079824 |
| 12 | 14.7866 | -1 | -14.7866 | 1 | 14.7866 | 1 | 1.169868325 | -1.169868325 | -4887.470351 | 4,902.2570 | 24,032,123.2116754 |
| 14 | 60.7655 | 0 | 0 | 0 | 0 | 0 | 1.783657076 | 0 | -3053.807389 | 3,114.5729 | 9,700,564.2819994 |
| 16 | 257.2313 | 1 | 257.2313 | 1 | 257.2313 | 1 | 2.410323813 | 2.410323813 | 1335.987694 | -1,078.7564 | 1,163,715.3580255 |
| 18 | 1,179.1185 | 2 | 2358.237 | 4 | 4716.474 | 16 | 3.071557453 | 6.143114907 | 8281.914899 | -7,102.7964 | 50,449,716.6903211 |
| 20 | 6,614.581 | 3 | 19843.743 | 9 | 59531.229 | 81 | 3.820502339 | 11.46150702 | 17783.97423 | -11,169.3932 | 124,755,345.0417530 |
| 22 | 41,068.023 | 4 | 164272.092 | 16 | 657088.368 | 256 | 4.613503797 | 18.45401519 | 29842.16567 | 11,225.8573 | 126,019,872.6846580 |
| Jumlah | 49,199.697150 | 0 | 186,703.741350 | 60 | 721,642.327750 | 708 | 17.243948 | 36.848275 | 49,199.697150 | 0.0000 | 378,763,989.3220020 |

15. Peramalan dengan metode *trend eksponensial*

| Barang | $Y$ | $X$ | $XY$ | $X^2$ | $X^2Y$ | $X^4$ | $\log Y$ | $X \log Y$ | $\hat{Y}$ | $e$ | $e^2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 0.7548 | -4 | -3.0192 | 16 | 12.0768 | 256 | -0.122168109 | 0.488672434 | 0.288033213 | 0.4668 | 0.2178712 |
| 8 | 0.88325 | -3 | -2.64975 | 9 | 7.94925 | 81 | -0.053916354 | 0.161749062 | 1.184623877 | -0.3014 | 0.0908262 |
| 10 | 3.5532 | -2 | -7.1064 | 4 | 14.2128 | 16 | 0.550619653 | -1.101239307 | 4.872124699 | -1.3189 | 1.7395624 |
| 12 | 14.7866 | -1 | -14.7866 | 1 | 14.7866 | 1 | 1.169868325 | -1.169868325 | 20.03808933 | -5.2515 | 27.5781402 |
| 14 | 60.7655 | 0 | 0 | 0 | 0 | 0 | 1.783657076 | 0 | 82.41271494 | -21.6472 | 468.6019148 |
| 16 | 257.2313 | 1 | 257.2313 | 1 | 257.2313 | 1 | 2.410323813 | 2.410323813 | 338.9472655 | -81.7160 | 6,677.4990223 |
| 18 | 1,179.1185 | 2 | 2358.237 | 4 | 4716.474 | 16 | 3.071557453 | 6.143114907 | 1394.023348 | -214.9048 | 46,184.0937203 |
| 20 | 6,614.581 | 3 | 19843.743 | 9 | 59531.229 | 81 | 3.820502339 | 11.46150702 | 5733.34348 | 881.2375 | 776,579.5663554 |
| 22 | 41,068.023 | 4 | 164272.092 | 16 | 657088.368 | 256 | 4.613503797 | 18.45401519 | 23580.11256 | 17,487.9104 | 305,827,011.4807140 |
| Jumlah | 49,199.697150 | 0 | 186,703.741350 | 60 | 721,642.327750 | 708 | 17.243948 | 36.848275 | 31,155.222242 | 18,044.4749 | 306,656,950.8681270 |

16. *Script code* algoritma *greedy by weight* untuk KP 01

```
%======================
%Input data
v=[30 11 11 92 33 21 15 13 152 26 13 14 21 13 14 21
13 13 27 11];
w=[1 1 1 1 1 1 1 8 2 1 1 1 1 1 1 1 1 1 1 1];
v2=[v]';
w2=[w]';
A=[w2 v2];
m=length(w);
n=length(v);
W=20;
%======================
w0=w; W0=W; n0=n; m0=m;
v0=v;
maksm= sum(w);
if W>=maksm
    disp('Semua barang terangkut');
    untung=sum(v);
    disp(['jumlah keuntungan= ', num2str(untung)]);
end
if n~=m
    disp('Error, matrik profit dan weight tidak
sama')
end
u= sort(w); %Berat sudah urut dari kecil ke besar
```

```
% menyesuaikan urutan profit dan urutan barang
for i=1:n
    for j=1:m
        a=u(i); b= w(j);
        if a==b
            V(i)=v(j);
            urutan(i)=j;
            w(j)=-b;
            break;
        end
    end
end
berat=0;i=0;
while berat<=W
    i=i+1;
    berat=berat+u(i);
end
if berat > W
    berat=berat-u(i);
    i=i-1;
end
untung= sum (V(1:i));
disp('Greedy By Weight');
disp('Hasil')
urutan=sort(urutan(1:i));
disp(['Jenis barang yang di
angkut:',num2str(urutan(1:i))])
disp(['Jumlah berat yang di angkut:
',num2str(berat)])
disp(['Jumlah keuntungan: ',num2str(untung)]);
```

17. *Script code* algoritma *greedy by profit* untuk KP 01

```
%======================
%Input data
v=[30 11 11 92 33 21 15 13 152 26 13 14 21 13 14 21
13 13 27 11];
w=[1 1 1 1 1 1 1 8 2 1 1 1 1 1 1 1 1 1 1 1];
v2=[v]';
```

```
w2=[w]';
A=[w2 v2];
m=length(w);
n=length(v);
W=20;
%======================
w0=w; W0=W; n0=n; m0=m;
v0=v;
maksm= sum(w);
if W>=maksm
    disp('Semua barang terangkut');
    untung=sum(v);
    disp(['jumlah keuntungan= ', num2str(untung)]);
    if n~=m
    disp('Error, matrik profit dan weight tidak
sama')
    end
end
w=w0; W=W0; n=n0; m=m0;
v=v0;
V= sort(v); %Nilai sudah urut dari kecil ke besar
V=V(n:-1:1);
% menyesuaikan urutan berat dan urutan barang
for i=1:n
    for j=1:m
        a=V(i); b= v(j);
        if a==b
            t(i)=w(j);
            urutan(i)=j;
            v(j)=-b;
            break;
        end
    end
end
berat=0;i=0;
while berat<=W
    i=i+1;
    berat=berat+t(i);
end
if berat > W
```

```
    berat=berat-t(i);
    i=i-1;
end
untung= sum (V(1:i));
disp('Greedy By Profit');
disp('Hasil')
urutan=sort(urutan(1:i));
disp(['Jenis barang yang di
angkut:',num2str(urutan(1:i))])
disp(['Jumlah berat yang di angkut:
',num2str(berat)])
disp(['Jumlah keuntungan: ',num2str(untung)]);
```

18. *Script code* algoritma *greedy by density* untuk KP 01

```
%======================
%Input data
v=[30 11 11 92 33 21 15 13 152 26 13 14 21 13 14 21
13 13 27 11];
w=[1 1 1 1 1 1 1 8 2 1 1 1 1 1 1 1 1 1 1 1];
v2=[v]';
w2=[w]';
A=[w2 v2];
m=length(w);
n=length(v);
W=20;
%======================
n=length(v); m=length(w);
w0=w; W0=W; n0=n; m0=m;
v0=v;
maksm= sum(w);
if W>=maksm
    disp('Semua barang terangkut');
    untung=sum(v);
    disp(['Jumlah keuntungan= ', num2str(untung)]);
end
if n~=m
    disp('Error, matrik profit dan weight tidak
sama')
end
```

```
w=w0; W=W0; n=n0; m=m0;
v=v0;
rasio=v./w
R= sort(rasio) % weight yang sudah urut dari kecil
ke besar
RR=R(n:-1:1)
% menyesuaikan urutan profit dan urutan barang
for i=1:n
    for j=1:m
        a=RR(i); b= rasio(j)
        if a==b
            V(i)=v(j);
            y(i)=w(j);
            urutan(i)=j
            rasio(j)=-b;
            break;
        end
    end
end
urutan=urutan
berat=0;i=0;
while berat<=W
    i=i+1;
    berat=berat+y(i)
end
if berat > W
    berat=berat-y(i)
    i=i-1
end
untung= sum (V(1:i));
disp('Greedy By Density');
disp('Hasil')
urutan=sort(urutan(1:i));
urutan2=urutan'
disp(['Jenis barang yang di
angkut:',num2str(urutan(1:i))])
disp(['Jumlah berat yang di angkut:
',num2str(berat)])
disp(['Jumlah keuntungan: ',num2str(untung)]);
```

19. *Script code* algoritma *dynamic programming* untuk KP 01

```
%======================
%Input data
v=[30 11 11 92 33 21 15 13 152 26 13 14 21 13 14 21 13
13 27 11];
w=[1 1 1 1 1 1 1 8 2 1 1 1 1 1 1 1 1 1 1 1];
v2=[v]';
w2=[w]';
A=[w2 v2];
m=length(w);
n=length(v);
W=20;
%======================
%inisialisasi dan proses
y=0:W;
cek=zeros(1,W+1);
barang=zeros(W+1,n);
for i=1:n
    for j=1:W+1
        selisih=y(j)-w(i);
        if selisih<0
            cek2(j)=cek(j);
            barang2(j,:)=barang(j,:);
        else
   [cek2(j) indx]=max([cek(j),v(i)+cek(selisih+1)]);
            if indx(1)==1
                barang2(j,:)=barang(j,:);
            else
                barang2(j,:)=barang(selisih+1,:);
                barang2(j,i)=1;
            end
        end
    end
    cek=cek2;
    barang=barang2;
end
[hasil_untung_DP indx]=max(cek2);
barang_DP(1,:)=barang2(indx(1),:);
urutan=[]; k=0; berat_DP=0;
```

```
for i=1:n
    if barang_DP(i)==1
        k=k+1;
        urutan(k)=i;
        berat_DP=berat_DP+w(i);
    end
end
%======================
%Hasil Algoritma DP
untung_ADP=cek2'
barang_ADP=barang2
solv=untung_ADP(W+1)
fprintf('Jumlah keuntungannya: %d', solv;
solw=berat_DP'
fprintf('Jumlah berat yang diangkut: %d', solw;
for i=1:n
    a(i)=barang_ADP(W+1,i)
end
aa=a'
fprintf('Jenis barang yang diangkut: %d', aa)
```

20. *Script code* algoritma *brute force* untuk KP 01

```
v=[30 11 11 92 33 21 15 13 152 26 13 14 21 13 14 21 13
13 27 11]
w=[1 1 1 1 1 1 1 8 2 1 1 1 1 1 1 1 1 1 1 1]
W=20
n = length(v)
%Inisialisasi
m = 0
nc = 0
s_test = zeros (n, 1)
t_test = 0
s_opt = s_test
t_opt = 0
t_opt2 =0

while(1)
```

```
    [s_test, m, nc, iad] = subfungsi(n, s_test, m, nc)
    t_test = s_test' * w
    t_test2 = s_test' * v
    if (t_opt2 < t_test2 && t_test <= W)
        t_opt=t_test
        s_opt = s_test
        t_opt2 = t_test2
    end
    if (~m)
        break
    end
end
s = s_opt
disp('Jenis barang yang diangkut: %d',s);
disp('No v(i) w(i) ambil(1)/tidak(0)');
for i = 1 : n
    fprintf ('%1d   %1d   %1d      %1d\n',
i,v(i),w(i),s(i));
end
disp('')
disp('Jadi dapat disimpulkan:')
fprintf ('Total Berat: %d\n', s'*w );
fprintf ('Total Nilai: %d\n', s'*v );
return
%++++++++++++++++++++++++++++++++++++++
function [a,m,nc,iad ] = subfungsi(n,a,m,nc)
if (~m)
    a(1:n)=0
    nc=0
    m=1
    iad=0
else
    a(1:n)=a(1:n)
    iad=1
    if (mod(nc,2)~=0)
        while (1)
            iad = iad+1
            if (a(iad-1)~=0)
                break
            end
```

```
        end
    end
    a(iad)=1-a(iad)
    nc=nc+2*a(iad)-1
    if (nc==a(n))
        m=0
    end
end
return
end
```

21. *Script code* algoritma *genetic* untuk KP 01

```
v=[30 11 11 92 33 21 15 13 152 26 13 14 21 13 14 21
13 13 27 11]
w=[1 1 1 1 1 1 1 8 2 1 1 1 1 1 1 1 1 1 1 1]
W=MaksBerat
v2=v';
w2=w';
A=[w2 v2];
m=length(w);
n=length(v);
r=W/m;
JumlahGen=n;
MaksBerat=W
%Mengatur variabel dan parameter
variabel = n;
Jumlah_populasi = n;
prob_mutasi = 0.01;
iterasi = 100;
mutasi = prob_mutasi * variabel*(Jumlah_populasi-1)
kromosom = round(rand(Jumlah_populasi,variabel))
delete('cek333.xlsx')
val2=xlswrite('cek333.xlsx',kromosom);
val2=xlsread('cek333.xlsx');
%Memulai Generasi
for generasi = 1:iterasi
    for i= 1:size(kromosom,1)
        Jumlah_v=sum(kromosom(i,:).* v);
```

```
        Jumlah_w=sum(kromosom(i,:).* w);
        if Jumlah_w <= W
            V(i)= Jumlah_v;
        else
            V(i) = 0;
        end
    end
    V
    % Mengurutkan Kromosom
    [V,index] = sort(V, 'descend')
    kromosom = kromosom(index(1:Jumlah_populasi),:)
    Jumlah_populasi_setengah =
round(Jumlah_populasi / 2);
    top_half =
kromosom(1:Jumlah_populasi_setengah,:);
    kromosom = [top_half; top_half];
    disp(['Generasi :', num2str(generasi) ,' Jumlah
harga :', num2str(V(1))])
    crossing = ceil((variabel - 1) *
rand(Jumlah_populasi_setengah,1))
    for n=1:Jumlah_populasi_setengah
kromosom(ceil(Jumlah_populasi_setengah*rand),1:cros
sing(n)) = kromosom(Jumlah_populasi_setengah +
n,1:crossing(n));
    end
    %Mutasi
    for n = 1:mutasi
        xx = ceil(Jumlah_populasi * rand);
        yy = ceil(variabel * rand);
        kromosom(xx,yy)= 1 - kromosom(xx,yy);
    end
    kromosom(Jumlah_populasi_setengah + 1: end ,:)
= top_half;
    counting(generasi) = V(1);
    if(iterasi == generasi)
        disp(['   ',num2str(Jumlah_populasi),'
',num2str(prob_mutasi), '    ',num2str(iterasi),'
' ,num2str(counting(generasi)) ])
    end
end
```

```
vx=Jumlah_v
fprintf('Jumlah keuntungannya: %d',vx);
wx=Jumlah_w
fprintf('Jumlah berat yang diangkut: %d',wx);
val2=xlsread('cek333.xlsx')
fprintf('Jenis barang yang diangkut: %d',val2);
```

## RIWAYAT HIDUP

### A. Identitas Diri

| | | | |
|---|---|---|---|
| 1. | Nama Lengkap | : | Muhammad Abdurrahman Rois |
| 2. | Tempat, Tgl.Lahir | : | Salatiga, 29 November 1996 |
| 3. | Alamat Rumah | : | Pulutan RT 03/02<br>Sidorejo Salatiga |
| 4. | No. HP | : | 081-914-430-369 |
| 5. | Email | : | roizmuhammad.math@gmail.com<br>roiz.muhammad@yahoo.co.id |

### B. Riwayat Pendidikan

1. Pendidikan Formal
   a. SD Negeri 1 Bringin
   b. SMP Negeri 16 Semarang
   c. SMA Negeri 8 Semarang
2. Pendidikan Non Formal
   a. Pondok Pesantren Al-Ma'rufiyyah
   b. Pondok Pesantren Al-Khoirot Malang

### C. Prestasi

a. Juara Komandan Peleton PBB Terbaik ORSENIK 2015
b. Juara 1 PBB ORSENIK 2015
c. Juara 1 LTUB ORSENIK 2015
d. Juara Variasi Formasi PBB ORSENIK 2015
e. Juara Umum Paskibra ORSENIK 2015
f. Juara 2 Lomba Rebana se-Kota Semarang 15 Mei 2016
g. Juara Kontributor Lomba Puisi tema "IBU" Se-Nusantara-Penerbit WA Publisher Bukit Tinggi-Sumbar- 19 Desember 2015 s/d 30 Januari 2016
h. Kontributor Lomba Cerpen&Puisi tema "Peduli Palestina" oleh Dirathree Publisher 30 November-7

Januari 2016

i. Delegasi Terpilih Pemilihan FLF 2016 Se-Nusantara "Forum Kepemudaan Berskala Nasional" Di Hotel Bumi Asih Medan-Sumatera
j. Juara 7 Lomba Essai Nasional di UIN Sunan Kalijaga Yogjakarta-13 Januari 2016
k. Juara Kontributor buku Lomba cerpen tema "Kisah Sejati" Se-Nusantara-Penerbit WA Publisher Bukit Tinggi-Sumbar- 19 Desember 2015 s/d 30 Januari 2016
l. Juara Harapan 2 Lomba Penulisan "Being Writer Being Motivation" se-Nusantara oleh Semangat Penulis
m. Lolos Non Degree Training In Country, Diklat Edutainment Nasional "Sukses Menjadi Penulis Produktif" di Hotel Aizzah Solo (8-10 November 2016)
n. Finalis 5 besar Paper Competition Pestagama 2016, Juara 4 Subtema Pertanian "Kontribusi Generasi Muda dalam Optimalisasi Sains dan Teknologi secara Berkelanjutan Menuju Indonesia Emas 2045" diselenggarakan oleh LSiS FMIPA UGM (Yogyakarta, 11-13 November 2016)
o. Pemakalah Seminar Nasional MIPA 2017 dengan Tema "Menguatkan Fundamental Research dan Pembelajaran MIPA untuk Kemanusiaan dan Peradaban"
p. Kandidat Full Funded Indonesia ASEAN SYNERGY STUDENT PROGRAM
q. Duta Sampah Kota Semarang 2018

**D. Karya Ilmiah**

a. Solusi Pertumbuhan Tanaman untuk Menghadapi Cuaca Ekstrem dengan Konsep the Power of MQMC (*Music of Quran and Music of Classic*)
b. Aplikasi Metode Simpleks Program Linier pada

Optimalisasi Pengelolaan Lahan Parkir Kawasan Fakultas Sains dan Teknologi di UIN Walisongo dengan Konsep "Super Ukhuwah"

c. Analisis Model Epidemi SIR pada Penyebaran Penyakit Tuberkulosis di Rumah Sakit Tugu Semarang dan Kontrol Optimalnya
d. Kontrol Optimal dan Model Matematika Penularan Virus Zika
e. Jaringan Jalan Kabupaten/ Kota di Provinsi Jawa Tengah Menggunakan *Minimum Spanning Tree* (MST)