# PENYELESAIAN KNAPSACK PROBLEM MENGGUNAKAN ALGORITMA GENETIKA

**Kartina Diah KW[1)], Mardhiah Fadhli[2)], Charly Sutanto[3)]**
[1,2)]Jurusan Teknik Komputer Politeknik Caltex Riau Pekanbaru
Jl. Umban Sari No.1 Rumbai-Pekanbaru-Riau
e-mail : diah@pcr.ac.id; mardhiah@pcr.ac.id; ch4r_lys@yahoo.com

**Abstrak**
*Keterbatasan wadah yang digunakan saat memilih barang yang akan dibawa merupakan perhatian utama pada kasus distribusi dari sekian banyak barang yang harus di distribusikan, yang masing-masing memiliki berat dan harga. Permasalahan ini dinamakan Knapsack Problem. Untuk menyelesaikan masalah ini, banyak algoritma yang dapat digunakan. Salah satunya yakni Algoritma Genetika. Algoritma ini bekerja dengan sebuah populasi yang terdiri dari individu-individu, yang masing-masing individu merepesentasikan sebuah solusi yang mungkin bagi persoalan yang ada untuk selanjutnya mengalami proses seleksi, pindah silang dan mutasi sehingga didapatkan populasi baru yang memberikan solusi yang mendekati solusi optimal. Aplikasi ini dibangun dengan menggunakan bahasa C.*

***Kata kunci:*** *Knapsack Problem, Algoritma Genetika, bahasa pemrograman C*

## 1. PENDAHULUAN

### Latar Belakang

*Knapsack problem* merupakan masalah di mana orang dihadapkan pada persoalan optimasi pada pemilihan benda yang dapat dimasukkan ke dalam sebuah wadah yang memiliki keterbatasan ruang atau daya tampung. Dengan adanya optimasi dalam pemilihan benda yang akan dimasukkan ke dalam wadah tersebut diharapkan dapat menghasilkan keuntungan yang maksimum. [10]

Benda-benda yang akan dimasukkan ini masing-masing memiliki berat dan sebuah nilai yang digunakan untuk menentukan prioritasnya dalam pemilihan tersebut. Nilainya dapat berupa tingkat kepentingan, harga barang, nilai sejarah, atau yang lainnya. [8] Wadah yang dimaksud di sini juga memiliki nilai konstanta yang merupakan nilai pembatas untuk benda-benda yang akan dimasukkan ke dalam wadah tersebut sehingga harus diambil sebuah cara memasukkan benda-benda tersebut ke dalam wadah sehingga menghasilkan hasil optimum tetapi tidak melebihi kemampuan wadah untuk menampungnya.

## 2. TINJAUAN PUSTAKA

### 2.1 *Knapsack Problem*

#### 2.1.1 Pengertian *Knapsack problem* [11]

*Knapsack problem* atau *rucksack problem* adalah masalah optimasi kombinatorial. Namanya berasal dari masalah maksimasi untuk pilihan paling tepat dari barang-barang yang akan dibawa dalam sebuah tas pada sebuah perjalanan. Sejumlah barang yang tersedia ini, masing-masing memiliki berat dan nilai, yang menentukan jumlah barang yang dapat dibawa sehingga total berat tidak melebihi kapasitas tas dan dengan total nilai yang sebesar mungkin.

#### 2.1.2 Jenis-Jenis *Knapsack Problem* [6]

Terdapat beberapa variasi *Knapsack problem*:

- *0/1 Knapsack problem*
  Setiap barang hanya tersedia 1 unit, *take it or leave it*.
- *Fractional Knapsack problem*
  Barang boleh dibawa sebagian saja (unit dalam pecahan). Versi *problem* ini menjadi masuk akal apabila barang yang tersedia dapat dibagi-bagi misalnya gula, tepung, dan sebagainya.
- *Bounded Knapsack problem*
  Setiap barang tersedia sebanyak N unit (jumlahnya terbatas).
- *Unbounded Knapsack problem*
  Setiap barang tersedia lebih dari 1 unit, jumlahnya tidak terbatas.

### 2.2 Algoritma Genetika

Pada algoritma ini, teknik pencarian dilakukan sekaligus atas sejumlah solusi yang mungkin dikenal dengan istilah *populasi*. Individu yang terdapat dalam satu populasi disebut dengan istilah *kromosom*. Kromosom

ini merupakan suatu solusi yang masih berbentuk simbol. Populasi awal dibangun secara acak, sedangkan populasi berikutnya merupakan hasil evolusi kromosom-kromosom melalui iterasi yang disebut dengan istilah *generasi.* Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang disebut dengan fungsi *fitness.*

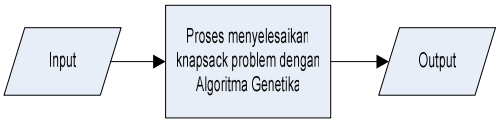
Selanjutnya konstruksi dasar dari Algoritma Genetika adalah sebagai berikut :

- Pendefinisian Kromosom
- Pendefinisian Fungsi Fitness
- Membangkitkan Sebuah Populasi Awal
- Reproduksi
- Crossover
- Mutasi
- Pelestarian

## 3. METODE PENELITIAN

### a. Perancangan Sistem

Cara kerja program secara garis besar adalah mencari nilai barang yang paling maksimum dengan berat barang tidak melebihi kapasitas yang tersedia. Pencariannya menggunakan algoritma genetika. Berikut adalah gambar blok diagram sistem:




**Gambar 3.1** Blok Diagram Sistem

### b. Perancangan Program

Perancangan ini terdiri dari perancangan penerapan algoritma genetika dalam *knapsack problem*, perancangan fungsi, dan perancangan flowchart.

**Perancangan Penerapan Algoritma Genetika dalam *Knapsack Problem***

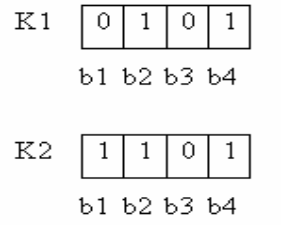
**3.2.1.1 Representasi Kromosom**

Dalam kasus *knapsack problem*, gen direpresentasikan dalam bentuk string bit. Caranya yaitu dengan memilih barang secara manual pada Tabel 3.1, di mana barang yang terpilih diberi tanda √ dan tanda x untuk barang yang tidak dipilih. Kedua tanda tersebut bisa dikodekan sebagai 1 untuk tanda √ dan 0 untuk tanda x. Dengan demikian representasi kromosom untuk masalah di atas adalah pengkodean biner dengan panjang 4 bit (sejumlah barang yang ada) [8].

**Tabel 3.1** Daftar Barang yang Bisa Dibawa Pedagang dan Pilihannya

| Kode | Nama Barang | Berat (kg) | Nilai (Rp) | Pilihan I | Pilihan II |
|---|---|---|---|---|---|
| b1 | A | 3 | 6 | x | √ |
| b2 | B | 2 | 5 | √ | √ |
| b3 | C | 5 | 9 | x | x |
| b4 | D | 4 | 8 | √ | √ |

Sehingga didapatkan kromosom seperti pada gambar berikut ini.




**Gambar 3.2** Representasi Kromosom

**Tabel 3.2** Representasi *Knapsack Problem* Jika Kapasitas Maksimum 6 kg

| Nm Brg | Berat (kg) | Nilai (Rp) | Kode Biner | Berat (kg) | Nilai (Rp) | Fit ness |
|---|---|---|---|---|---|---|
| A | 3 | 6 | (a) 0101 | 6 | 13 | 13 |
| B | 2 | 5 | (b) 1101 | 9 | 19 | -3 |
| C | 5 | 9 | | | | |
| D | 4 | 8 | | | | |

Kromosom K1 menyatakan bahwa barang-barang yang terpilih adalah b2 dan b4. Sesuai dengan Tabel 3.1, ternyata totalnya sama dengan 6 kg. Kromosom K1 ini dinyatakan *valid* karena total beratnya kurang atau sama dengan berat maksimal yang diijinkan (6 kg). Sedangkan kromosom K2 menyatakan barang-barang yang terpilih adalah b1, b2, dan b4 yang total beratnya sama dengan 9 kg. Dengan demikian, K2 dikatakan kromosom yang tidak *valid* karena memberikan total berat yang melebihi berat maksimal yang diijinkan (6 kg).

**3.2.1.2 Prosedur Inisialisasi**

Inisialisasi terhadap secara kromosom dilakukan secara acak (*random*).

**3.2.1.3 Fungsi *Fitness***

Untuk membangun fungsi *fitness* dalam kasus *knapsack problem*, bisa dimulai dengan tujuan yang ingin dicapai, yakni menemukan sejumlah barang yang total nilainya paling besar (maksimasi) dan total beratnya kurang atau sama dengan total berat yang diijinkan sehingga bisa ditulis fungsi *fitness*-nya adalah

$$f = \sum_{i=1}^{n} b_i v_i$$

batasan total beratnya $f = \sum_{i=1}^{n} b_i w_i \le w$

di mana $b_i$ bernilai 1 atau 0 yang menyatakan barang ke-*i* diambil atau tidak diambil, $v_i$ menyatakan nilai barang ke-*i* dan $w_i$ menyatakan berat barang ke-*i*. [8]

Oleh karena kapasitas maksimumnya 6 kg, maka berat pada kode biner (a) masih sama dengan (tidak melebihi) kapasitas, sehingga nilai *fitness* sama dengan nilai dari kode biner = 13, sedangkan berat pada kode biner (b) melebihi kapasitas, sehingga nilai *fitness*-nya didapat dari pengurangan kapasitas dan berat pada kode biner yakni 6 – 9 = -3.

**3.2.1.4 Seleksi** [5]

Untuk menyelesaikan masalah *knapsack problem*, digunakan metode seleksi induk yaitu seleksi berdasarkan *roulette wheel*. Langkah-langkah penyeleksiannya:

1. Hitung fungsi *fitness* untuk masing-masing kromosom *eval*($v_i$),
   i = 1, 2, ..., pop_size (ukuran populasi)
2. Hitung total *fitness* dari populasi tersebut:
   $$F = \sum_{i=1}^{pop\_size} eval(v_i)$$
3. Hitung probabilitas dari seleksi $p_i$ untuk masing-masing kromosom $v_i$,
   i = 1, 2, ..., pop_size
   $p_i = eval(v_i) / F$
4. Hitung komulatif p $q_i$ untuk masing-masing kromosom $v_i$,
   i = 1, 2, ..., pop_size
   $$q_i = \sum_{j=1}^{i} p_j$$

Proses seleksi berdasarkan atas pemutaran roulette wheel sebanyak pop_size kali; setiap kali dipilih kromosom tunggal sebagai populasi baru, dengan cara:

- Bangkitkan bilangan acak r
- Bila $r < q_1$ maka pilih kromosom pertama ($v_1$); bila tidak pilih kromosom ke-*i* $v_i$ ($2 \le i \le pop\_size$) sedemikian rupa sehingga $q_{i-1} < r \le q_i$

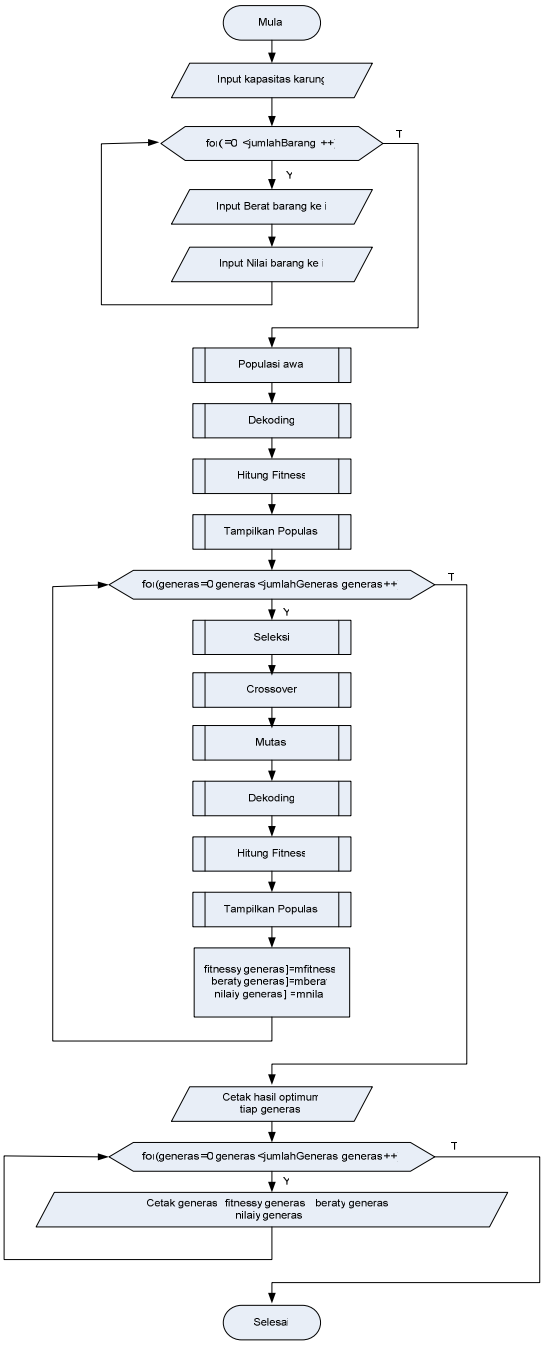
#### 3.2.1.5 Operator Genetika

Metode *crossover* yang digunakan pada kromosom berbentuk string biner ini adalah *crossover* satu titik (*one-point crossover*). Adapun mutasi yang digunakan pada masalah *knapsack problem* ini adalah mutasi yang bernilai biner.

#### 3.2.1.6 Penentuan Parameter

Parameter yang digunakan pada *knapsack problem* ini sbb:

1. Ukuran populasinya = 30.
2. Probabilitas pindah silang = 60% atau 0.6.
3. Probabilitas mutasi = 2% atau 0.02.
4. Jumlah generasi = 1000.




**Gambar 3.3** Flowchart Main Program

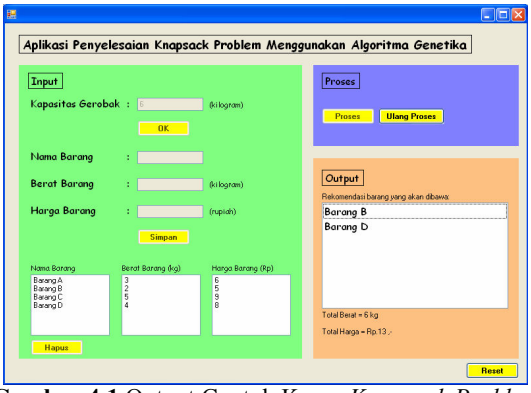
## 4. HASIL DAN PEMBAHASAN

Table 4.1 berikut adalah contoh barang yang akan diujikan dengan *knapsack problem*.

**Tabel 4.1** Tabel contoh barang

| Barang | Berat | Nilai |
|---|---|---|
| A | 3 | 6000 |
| B | 2 | 9000 |
| C | 5 | 5000 |
| D | 4 | 8000 |

Hasil pengujian dengan system akan menampilkan kombinasi barang optimum (total harga maksimum dan berat barang >= total kapasitas gerobak pengangkutnya).

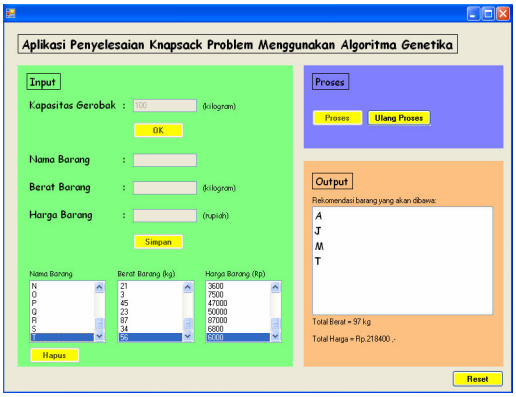



**Gambar 4.1** Output Contoh Kasus *Knapsack Problem*

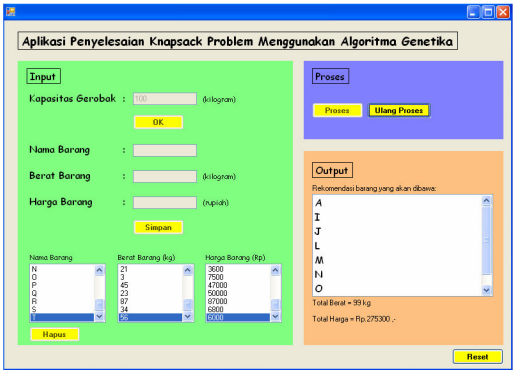
**Tabel 4.2** Data Barang

| Barang | Berat | Nilai |
|---|---|---|
| A | 2 | 5000 |
| B | 10 | 4000 |
| C | 5 | 4000 |
| D | 45 | 100000 |
| E | 23 | 68000 |
| F | 4 | 2500 |
| G | 3 | 7400 |
| H | 34 | 36000 |
| I | 7 | 1200 |
| J | 34 | 200000 |
| K | 12 | 4500 |
| L | 4 | 600 |
| M | 5 | 7400 |
| N | 21 | 3600 |
| O | 3 | 7500 |
| P | 45 | 47000 |
| Q | 23 | 50000 |
| R | 87 | 87000 |
| S | 34 | 6800 |
| T | 56 | 6000 |

Apabila diinputkan data seperti pada tabel 4.1 maka output yang dihasilkan bias jadi seperti pada gambar 4.43 atau gambar 4.44 berikut atau kemungkinan lainnya, tergantung dari kombinasi barang yang terpilih pada saat *random* (pengacakan).




**Gambar 4.2** Kemungkinan Pertama, Output dari Data Tabel 4.1




**Gambar 4.3** Kemungkinan Kedua, Output dari Data Tabel 4.2

## 5. KESIMPULAN

Setelah melakukan pengujian, maka dapat diambil beberapa kesimpulan:

1. Aplikasi ini akan menghasilkan optimasi kombinatorial yang mendekati solusi optimalnya.
2. Hasil dari program bergantung kepada penentuan parameter dan data yang diinputkan.

## 6. DAFTAR PUSTAKA

Adit279,2008,*Knapsack Problem dengan Algoritma Genetika.* http://adit279.wordpress.com/2008/12/03/knapsack-problem-dengan-algoritma-genetika/ . Diakses 13 Maret 2008, 14:56 WIB.

Desiani, Anita dan Muhammad Arhami. 2006. *Konsep Kecerdasan Buatan*. Yogyakarta: Andi Offset.

Khannedy, Eko Kurniawan. 2007. *Pemrograman C.* http://www.scribd.com/doc/5040002/Pemrograman-C-Indonesia . Diakses 13 Maret 2008, 14:51 WIB.

Kusumadewi, Sri. 2003. *Artificial Intelligence (Teknik dan Aplikasinya)*. Yogyakarta: Graha Ilmu.

Kuswadi, Son. 2007. *Kendali Cerdas, Teori dan Aplikasi Praktisnya*. Yogyakarta: Andi Offset.

Setiadi, Robert. 2008. *Algoritma itu Mudah*. Jakarta: Prima Infosarana Media.

Shrestha, Dipti dan Maya Hristakeva. *Solving the 0-1 Knapsack problem with Genetic Algorithms*. USA: Computer Science Department, Simpson College.

Suyanto. 2008. *Evolutionary Computation: Komputasi Berbasis "Evolusi" dan "Genetika"*. Bandung: Informatika.

Wibowo, Agus Urip Ari , Juni Nurma Sari dan Kori Cahyono. 2003. *Bahasa Pemrograman I*. Rumbai: Politeknik Caltex Riau.

Wicaksono, Prasetyo Andy. 2007. *Makalah IF2251 Strategi Algoritmik: Eksplorasi Algoritma Brute Force, Greedy dan Pemrograman Dinamis pada Penyelesaian Masalah 0/1 Knapsack*. Bandung: STEI, Institut Teknologi Bandung.