IMPLEMENTASI ALGORITMA GENETIKA UNTUK OPTIMASI 0/1 *MULTI-DIMENSIONAL KNAPSACK PROBLEM* DALAM PENENTUAN MENU MAKANAN SEHAT

SKRIPSI

Disusun oleh :

RIZKHY AYUNING TYAS

NIM. 0810963067

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

IMPLEMENTASI ALGORITMA GENETIKA UNTUK OPTIMASI 0/1 *MULTI-DIMENSIONAL KNAPSACK PROBLEM* DALAM PENENTUAN MENU MAKANAN SEHAT

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

RIZKHY AYUNING TYAS

NIM. 0810963067

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

IMPLEMENTASI ALGORITMA GENETIKA UNTUK OPTIMASI 0/1 *MULTI-DIMENSIONAL KNAPSACK PROBLEM* DALAM PENENTUAN MENU MAKANAN SEHAT

SKRIPSI

Disusun oleh :

RIZKHY AYUNING TYAS

NIM. 0810963067

Telah diperiksa dan disetujui oleh :

Pembimbing I,	Pembimbing II,
Drs. Muh. Arif Rahman, M.Kom	**Candra Dewi S.Kom, M.Sc**
NIP. 196604231991111001	**NIP. 197711142003122001**

LEMBAR PENGESAHAN

IMPLEMENTASI ALGORITMA GENETIKA UNTUK OPTIMASI 0/1 *MULTI-DIMENSIONAL KNAPSACK PROBLEM* DALAM PENENTUAN MENU MAKANAN SEHAT

SKRIPSI

Diajukan untuk memenuhi persyaratan memperolah gelar Sarjana Komputer

Disusun oleh :

RIZKHY AYUNING TYAS

NIM. 0810960059

Skripsi ini telah diuji dan dinyatakan lulus tanggal 17 Januari 2013

Penguji I	Penguji II
Dian Eka Ratnawati, S.Si., M.Kom	**Indriati, ST., M.Kom**
NIP. 19730619 200212 2 001	**NIK. 831013 06 1 2 0035**

Penguji III

Imam Cholissodin, S.Si., M.Kom

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, MT.
NIP. 196708011992031001

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, Januari 2013

Mahasiswa,

Rizkhy Ayuning Tyas

NIM. 0810963067

ABSTRAK

Rizkhy Ayuning Tyas. 2013. Implementasi Algoritma Genetika untuk Optimasi 0/1 *Multi-Dimensional Knapsack Problem* dalam Penentuan Makanan Sehat.
Dosen Pembimbing : Drs. Muh.Arif Rahman, M.Kom. dan Candra Dewi, S.Kom.,MSc

Permasalahan yang dihadapi masyarakat sekarang adalah penyakit degenerative yang disebabkan oleh hidup yang tidak sehat. Salah satu cara hidup sehat adalah perencanaan menu makanan sehat. Menu makanan sehat merupakan permasalahan kombinatorial yang kompleks sehingga dibutuhkan metode kombinatorial yang tepat. Penelitian ini mengimplementasikan Multi-Dimensional Knapsack Problem (MDKP) dan algoritma genetika untuk alokasi penempatan perencanaan makanan yang akan dikonsumsi dalam satu hari.

Alokasi waktu makan masing-masing memiliki kapasitas dan dilakukan pembobotan sesuai dengan kebutuhan perhari berdasarkan energi yang dibutuhkan. Optimasi MDKP menggunakan algoritma genetika menggunakan pencarian heuristik dari variasi kromosom yang dihasilkan dari penyandian dengan simbol, dari masing-masing kromosom merepresentasikan variasi solusi yang berbeda-beda. Kromosom membentuk suatu menu makanan sehat untuk pagi, siang dan malam yang akan dievaluasi melalui fungsi fitness yang didapatkan dari pinalti yang dibentuk berdasarkan konstrain yang terdapat dalam MDKP.

Berdasarkan hasil pengujian didapatkan menu makanan sehat yang diperoleh dengan fitness sebesar 0.0449 pada kondisi pelatihan algoritma genetika dengan parameter probabilitas mutasi sebesar 50%, probabilitas crossover sebesar 70%, generasi sebanyak 100 generasi dan individu awal yang dibangkitkan sebanyak 7 individu.

Kata Kunci : Multi-dimensional Knapsack Problem (MDKP), Algoritma Genetika, penyusunan menu makanan

ABSTRACT

Rizkhy Ayuning Tyas. 2013. ***Implementation of Genetic Algorithm for Optimation 0/1 Multi-Dimensional Knapsack Problem to Determinate Healthy Foods.***

Advisor : Drs. Muh. Arif Rahman, M.Kom. and Candra Dewi, S.Kom., M.Sc.

The problem that society faces nowadays is a degenerative disease caused by unhealthy life style. One way to have a healthier life style is by planning menu can combining a healthy food menu. Menu planning is a complex combinatorial problem that requires a right combinatorial method. This research implements Multi-Dimensional Knapsack Problem (MDKP) and genetic algorithm to plan food that will be consumed in a day.

The allocation for each meal time has its own capacity and will be weighted according to the personal daily energy need. MDKP optimization using genetic algorithm applies a heuristic search based on chromosomal variant which encoded to a symbol that represents the variety of problem solutions. The chromosome represents a healthy diet for breakfast, lunch, and dinner furthermore it will be evaluated through fitness function that is resulted from the penalty constraint in MDKP.

From testing, the healthy food menu with the fitness of 0.0449 was obtained on genetic algorithm training with parameter mutation probability of 50%, crossover probability of 70%, the amount of generations is 100, and earlier generated individual is .

Keywords *: Multi-Dimensional Knapsack Problem, Genetic Algorithms, Determination menu diet*

KATA PENGANTAR

Puji syukur kehadirat Allah SWT, hanya dengan rahmat dan karunia yang telah diberikan kepada penulis, sehingga dapat menyelesaikan skripsi yang berjudul "Implementasi Algoritma Genetika untuk Optimasi 0/1 *Multi-Dimensional Knapsack Problem* dalam Penentuan Menu Makanan Sehat".

Skripsi ini merupakan salah satu syarat untuk memenuhi persyaratan akademis untuk menyelesaikan studi di program Sarjana Ilmu Komputer Universitas Brawijaya.

Pada kesempatan ini penulis mengucapkan banyak terima kasih atas segala bantuan dan dedikasi moral maupun material dalam rangka penyusunan skripsi ini.

1. Drs.Muh Arif Rahman,M.Kom., selaku dosen pembimbing I yang telah membimbing dengan bijaksana dan sabar dalam membimbing dengan baik penyusunan skripsi ini.
2. Candra Dewi, S.Kom., M.Sc., selaku dosen pembimbing II yang telah membimbing dengan bijaksana dan sabar dalam membimbing dengan baik penyusunan skripsi ini.
3. Drs. Marji, M.T, selaku Ketua Program Studi Informatika/Ilmu Komputer di Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
4. Dany Primanita Kartika, ST., selaku dosen pembimbing akademik atas nasehat, bimbingan, saran, dukungan yang diberikan selama penulis menuntut ilmu di Program Studi Ilmu Komputer.
5. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
6. Segenap staf dan karyawan di Program Teknik Informatika Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya yang telah banyak membantu Penulis dalam pelaksanaan penyusunan skripsi ini.
7. Segenap staf dan karyawan Jurusan Matematika Universitas Brawijaya yang telah membantu penyusunan skripsi ini.

8. Ayah, Ibu, kakak dan adik–adikku serta keluarga besarku yang tersayang, terima kasih atas dukungan dan doanya.
9. Riadlul Mu'minin, terima kasih atas dukungan dan bantuannya.
10. Zanu, Tika, Sari, Neny, Yanita, Mutya, Dinda, Rizky A., Adit, Nisa dan Tommy yang senantiasa memotivasi penulis.
11. Teman–teman Program Studi Ilmu Komputer 2008 atas segala bantuan, motivasi dan doanya.
12. Seluruh pihak yang tidak dapat disebut secara langsung yang telah memberikan bantuan demi terselesaikannya skripsi ini.

Penulis menyadari bahwa masih banyak kekurangan dalam penulisan laporan ini yang disebabkan oleh keterbatasan kemampuan dan pengalaman. Oleh karena itu, Penulis sangat menghargai saran dan kritik yang sifatnya membangun demi perbaikan penulisan dan mutu isi penelitian ini.

Penulis berharap semoga laporan ini dapat memberikan manfaat kepada pembaca dan bisa diambil manfaatnya untuk pengembangan selanjutnya.

Malang, Januari 2013

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SOURCECODE

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Hidup sehat merupakan dambaan semua masyarakat, salah satu permasalahan yang sering dihadapi masyarakat akibat hidup yang kurang sehat adalah berat badan yang tidak seimbang dan menyebabkan penyakit *degenerative* seperti obesitas. Gaya hidup sehat dapat dicapai dengan mengatur pola makan, mengatur berat badan, managemen stress, tidur yang cukup, menghindari rokok dan minuan beralkohol dan tidak terlalu banyak duduk (Eckholm, 1895).

Berat badan yang melebihi berat badan relatif disebut dengan obesitas, kondisi ini disebabkan oleh banyaknya konsumsi kalori melebihi kebutuhan kalori dari tubuh sehingga disimpan dalam bentuk jaringan lemak (Budiyanto, 2002). Obesitas selain dapat mengganggu penampilan juga menurunkan kinerja penderita obesitas yang cenderung malas untuk melakukan kegiatan sehingga mengurangi aktifitas tubuh (Misky, 2005). Obesitas ditanggulangi dengan gaya hidup sehat, terutama pola makan. Banyak cara yang dapat digunakan untuk menurunkan berat badan (diet), tapi banyak yang salah menerapkan diet yang benar.

Menu makanan yang sehat merupakan faktor penting dalam menjaga kesehatan dan mencegah berbagai macam penyakit, berdasarkan penelitian menu makanan tidak sehat adalah penyebab kedua kangker setelah asap rokok (Eckholm, 1895). Menu makanan yang sehat mengandung nutrisi makronutrien, yaitu karbohidrat, lemak dan protein untuk mendapatkan energi (Mahmud et all, 2005).

Makanan sebagai sumber tenaga mampu memberikan energi lebih dari sumber makanan lainnya, yaitu karobidrat, protein, buah-buahan dan makanan yang mengandung susu (Marshall, 2006). Sumber makanan dibagi menjadi dua, yaitu sumber makanan nabati dan hewani. Sumber makanan nabati didapatkan dari tumbuhan, sedangkan hewani didapatkan dari hewan (Suryatin, 2004).

Beberapa riset tentang menu makanan telah banyak diteliti, seperti menu makanan berbasis web untuk berbagi menu makanan, user menginputkan menu makanan yang akan dibagi dengan user lainnya. Sistem akan memberikan anjuran

dengan siapa user berbagi makanan, dengan tujuan sistem dapat mendekatkan user dengan tetangganya (Kitahara dan Kanai, 2010). Di Jepang telah terdapat riset yang mendevelop *ontology-based health support system* untuk user yang ingin menjaga gaya hidupnya tetap sehat, sistem ini mendapatkan data user menggunakan sensor dan handphone untuk memberikan saran untuk menjaga kesehatan sesuai data yang dikumpulkan oleh user (Izumi et all, 2007). Riset lainnya adalah *inferencing system* untuk menawarkan menu makanan yang sehat, sistem ini menggunakan scan dengan program OCR dari struk belanja user dan diolah dengan *inferencing system* untuk mengestimasi alternatif apa yang akan dimakan oleh user dalam satu minggu (Mankoff, et all).

Pembuatan menu makanan merupakan permasalah kombinatorial yang kompleks, sehingga dibutuhkan metode yang baik untuk membuat menu makanan. *Multi-dimensional Knapsack Problem* (MDKP), merupakan permasalahan kombinatorial yang cocok digunakan untuk membuat menu makanan menggunakan algoritma genetika (Kashima, et all., 2009). MDKP biasa disebut juga dengan *multi-constraint knapsack problem*, MKDP didefinisikan sebagai knapsack problem yang memiliki beberapa constraint seperti kandungan, ukuran, konsistensi, dll. Dalam penyelesaiannya, MDKP memiliki m knapsack dengan kapasitas W1, W2, …, Wm dan n objek yang masing-masing memiliki cost (biaya) cj, j=1, 2, .., n. MKDP akan memasukkan semua n objek tanpa melebihi kapasistas dengan cost yang minimum (Gen dan Cheng, 2000). MDKP digunakan untuk alokasi penempatan perencanaan makanan yang akan dikonsumsi dalam satu hari, dengan kapasitas W1 untuk makan pagi, W2 untuk makan siang dan W3 untuk makan malam. Dari masing-masing kapasitas maksimal perwaktu makan akan dilakkan pembobotan sesuai dengan kebutuhan perhari berdasarkan energi yang dibutuhkan. Program ini memaksimalkan kepuasan dari pelangga dengan berbagai konstrain kebutuhan makronutrien, mikronutrien, kolesterol, dan lain-lain. Dalam pengembangannya, MDKP dioptimasi menggunakan algoritma genetika (Kashima et all,2009).

Permasalahan kombinatorial yang sering diselesaikan menggunakan algoritma genetka adalah *set-covering problem, bin packing problem, minimum spanning tree problem, scheduling dan knapsack problem*.

Menu planning merupakan permasalah kombinatorial dengan permasalahan bagaimana cara mengkombinasikan menu makanan pokok, lauk nabati, lauk hewani, sayur dan buah menjadi satu menu makanan dengan konstrain yang telah ditentukan. Algoritma genetika merupakan salah satu algoritma yang sering dipakai untuk optimasi permasalahan kombinatorial. Cara optimasi algoritma genetika adalah menyimbolkan suatu solusi dari permasalahan yang ada menjadi kromosom yang selanjutnya dihitung nilai fitness dari masing-masing kromosom yang terbentuk, dari kromosom yang terbentuk akan didapatkan solusi yang berbeda-beda, kombinasi gen dalam kromosom direpresentasikan menjadi menu makanan sehat yang akan dievaluasi berdasarkan konstrain yang digunakan. Fungsi fitness didapatkan dari penalty yang digunakan algoritma genetika untuk mencari solusi yang optimal (Gen dan Cheng, 2000). Berdasarkan uraian diatas maka terpikir judul penelitian "Implementasi Algoritma Genetika untuk Optimasi 0/1 *Multi-dimensional Knapsack Problem* dalam Penentuan Menu Makanan Sehat".

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan diatas, dapat diambil rumusan sebagai berikut :

1. Bagaimana mengimplementasikan algoritma genetika untuk optimasi *multi-dimensional knapsack problem* dalam penentuan menu makanan sehat.
2. Berapa nilai probabilitas crossover dan mutasi, iterasi, serta jumlah populasi awal yang dibangkitkan pada permasalahan penentuan menu makanan sehat untuk mendapatkan solusi yang optimal.

1.3 Batasan Masalah

Batasan masalah dari permasalahan diatas adalah sebagai berikut :

1. Menu makanan digunakan untuk usia remaja dan dewasa sehat.
2. Data didapatkan dari Daftar Komposisi Bahan Makanan (DKBM) oleh persatuan ahli gizi Indonesia (PERSAGI).
3. Menu yang ditentukan terdiri dari tiga kali makan, yaitu makan pagi, siang dan malam dan tidak diperbolehkan meninggalkan salah satu dari waktu

makan tersebut dengan menu makanan yaitu menu makanan pokok, lauk nabati, lauk hewani, sayur dan buah.

4. Penentuan menu makanan berdasarkan jumlah energi user dan kandungan dari makanan dengan hasil dalam satuan gram.
5. Penentuan menu makanan hanya berdasarkan dietary menu dan tidak mempertimbangkan warna, harga, temperature, bentuk, cara penyajian dan keberadaan makanan pada musim tertentu .
6. Density menu makanan berdasarkan hasil wawancara dan studi litelatur.
7. *Crossover* yang digunakan adalah *twopoint crossover*, mutasi menggunakan mutasi biner dan seleksi dengan *roulette wheel*.
8. Nilai dari *crossover rate* dan *mutation rate* 10%, 25%, 50%, 70% dan 90%.
9. Jumlah individu yang dibangkitkan adalah 3, 5 dan 7, dengan iterasi sebanyak 25, 50 dan 100 generasi.

1.4 Tujuan

Tujuan yang ingin dicapai dalam penyusunan skripsi ini adalah :

1. Mengetahui implementasi algoritma genetika dalam penentuan menu makanan sehat.
2. Mengetahui nilai probabilitas *crossover* dan mutasi, iterasi yang dibutuhkan, serta jumlah populasi awal yang dibangkitkan dalam menentukan menu makanan sehat sehingga didapatkan solusi optimal.

1.5 Manfaat

Manfaat yang dapat diperoleh dari penyusunan skripsi ini adalah menyediakan sistem untuk perencanaan menu (menu planning) yang diharapkan dapat menghasilkan menu yang sehat sesuai dengan kebutuhan tubuh manusia dalam satu hari.

1.6 Sistematika penulisan

Metodologi yang digunakan dalam skripsi ini adalah :

1. BAB I PENDAHULUAN

   Memuat latar velakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan penelitian.

2. BAB II DASAR TEORI

   Berisi teori-teori yang menjadi landasan pelaksanaan dan penulisan penelitian ini.

3. BAB III METODOLOGI DAN PERANCANGAN

   Menerangkan metode-metode yang digunakan dan model system yang akan digunakan dalam penelitian.

4. BAB IV IMPLEMENTASI DAN PEMBAHASAN

   Menejelaskan implementasi perangkat lunak dan pembahasan hasil analisis yang dilakukan.

5. BAB V PENUTUP

   Berisi kesimpulan dari hasil yang telah dilakukan dan saran-saran mengenai hasil penelitian yang perlu dikembangkan.

BAB II
TINJAUAN PUSTAKA

2.1 Algoritma Genetika

Algoritma genetika merupakan suatu algoritma yang terinspirasi dari mekanisme biologis dengan pencarian heuristik yang memungkinkan keberagaman kromosom (Kusumadewi, 2003).

GA memungkinkan terjadinya pembentukan spesies baru atau dikenal dengan istilah *speciation*, dengan beberapa kondisi fisik yang beradaptasi dengan lingkungan. Perubahan bertahap yang terjadi tersebut adalah *co-evolution* (Suyanto, 2007).

Berikut ini merupakan beberapa istilah dalam GA, yaitu : (Suyanto, 2007).

1. *Genome* adalah kumpulan dari kromosom
2. Kromosom adalah kumpulan dari gen.
3. Gen adalah unit hereditas dan pengkodean informasi yang dapat diturunkan kepada geneprasi berikutnya.
4. *Allele* adalah setting dari gen dari pengkodean
5. *Genotype* adalah genome yang lengkap dari suatu individu dengan semua settingnya
6. *Phenotype* adalah semua individu dengan semua sifat-sifatnya

2.1.1 Struktur Umum Algoritma Genetika

Struktur umum pada algoritma genetika adalah sebagai berikut : (Kusumadewi, 2003)

1. Representasi kromosom.
2. Evaluasi dengan menghitung fitness.
3. Proses crossover untuk mendapatkan individu baru.
4. Proses mutasi yang berfungsi untuk meningkatkan variasi dalam populasi.
5. Proses seleksi untuk membentuk populasi baru.

2.1.2 Komponen-komponen Utama Algoritma Genetika

Enam komponen utama dari Algoritma Genetika adalah teknik penyandian, prosedur inisialisasi, fungsi evaluasi, seleksi, operator genetika dan penentuan parameter. Masing-masing penjelasan dari komponen-komponen utama algoritma genetika adalah sebagai berikut : (kusumadewi, 2003).

2.1.2.1 Teknik Penyandian

Teknik penyandian merupakan suatu teknik yang berupa kode pada gen dan kromosom. Teknik penyandian dalam algoritma genetika terdiri dari gen dan kromosom, dimana gen merupakan bagian dari kromosom. Gen dapat direpresentasikan dalam bentuk string, bit, pohon, array bilangan real, daftar aturan, elemen permutasi, elemen program, atau representasi lainnya yang dapat diimplementasikan untuk operator genetika.

Teknik penyandian yang digunakan adalah *Binary Encoding*, setiap gen bias berupa deretan nilai 0 dan 1 (Suyanto, 2007). Contoh dari pengkodean biner dapat dilihat pada gambar 2.1

Kromosom A	1010100110001001
Kromosom B	000010100100100

Gambar2. 1 Contoh teknik penyandian Binary Encoding

Prosedur inisialisasi adalah suatu prosedur yang harus kita lakukan dalam menginisialisasi populasi.Dalam pembuatan prosedur inisialisasi, yang harus kita perhatikan adalah masalah yang akan dipecahkan dan jenis operator genetika yang akan dimplemen-tasikan untuk menentukan ukuran populasi. Populasi dalam algoritma genetika harus dikodekan (inisialisasi), inisialisasi kromosom dilakukan secara acak akan tetapi tetap memperhatikan domain solusi dan kendala permasalahan (kusumadewi, 2003).

2.1.2.2 Fungsi Evaluasi

Fungsi evalusi merupakan suatu fungsi yang digunakan untuk mengetahui tingkat keefektifan suatu kromosom untuk diteruskan keeksistensianya. Terdapat dua hal yang harus dilakukan dalam melakukan evaluasi kromosom, yaitu :

evaluasi fungsi objektif (fungsi tujuan) dan konversi fungsi objektif ke dalam fungsi *fitness* (kusumadewi, 2003).

Dalam penetapan fungsi *fitness* digunakan suatu fungsi tertentu untuk dijadikan ukuran dari nilai fitness dari suatu kromosom. Apabila solusi yang diinginkan adalah memaksimalkan, maka fungsi fitness yang digunakan adalah f = h (Suyanto, 2007).

2.1.2.3 Density

Density merupakan proses pembobotan untuk menentukan gen yang akan dipilih dalam *fractional knapsack problem*. Pembobotan *density* berdasarkan kandungan item makanan yang akan dioptimalkan dan dipilih yang terbaik. Nilai density diambil dari penjumlahan seluruh kandungan makanan dan diambil nilai yang terbaik.

2.1.2.4 Penalty

Penalty merupakan suatu bobot yang digunakan ketika individu melakukan pelanggaran terhadap aturan. Aturan-aturan yang terdapat pada perencanaan menu makanan ini adalah :

1. **Kebutuhan Makronutrien**

   Macronutrien adalah bahan makanan sumber utama energi, terdiri dari karbohidrat, protein dan lemak. Prosentase dari makronutrien yaitu karbohidrat adalah 60%, protein adalah 15% dan lemak 25% dari total energi. *Penalty* yang digunakan adalah

A. Pagi hari, dibutuhkan banyak tenaga untuk memulai beraktifitas maka energi yang didapatkan dari karbohidrat adalah 35% energi dari total energi untuk karbohidrat, begitu juga dengan protein dan lemak.
B. Siang hari, pada saat beraktifitas tubuh membutuhkan banyak tenaga, sehingga karbohidrat, protein dan lemak yang dibutuhkan adalah 35% energi dari total energi dari masing-masing nutrisi makro.

C. Malam hari, waktu untuk beristirahat dan tidak terlalu anyak membutuhkan energi, sehingga masing-masing nutrisi makro mempunyai prosentase sebesar 30% dari total energi.

2. Kebutuhan Mikronutrien

Micronutrien merupakan zat gizi mikro yang dimiliki makanan, terdapat beberapa micronutrien yang menjadi konstrain dalam MDKP perencanaa menu makanan, yaitu :

a. Serat

Konsumsi minimal dari serat dalam sehari adalah 10 gram per 1000 kalori, dan tidak boleh melebihi 40 gram.

b. Besi

Besi adalah micronutrien yang sangat penting untuk pembentukan hemoglobin, khususnya untuk wanita usia remaja. Kebutuhan nutrisi besi dapat dilihat pada tabel 2.2. Penentuan pinalti dari besi dipengaruhi oleh jenis kelamin dan umur. Masing-masing mempunyai kebutuhan yang berbeda.

c. Fosfor

Fosfor merupakan bagian dari ATP yang digunakan untuk suplay energi kegiatan selular. Umumnya fosfor terdapat disemua makanan karena fungsinya

d. Kalsium

Kalsium memegang peranan penting dalam tulang dan gigi, selain itu juga pada peredaran darah dan jaringan tubuh. Kekurangan kalsium menyebabkan terjadinya pengeroposan tulang.

e. Vitamin A

Vitamin A berfungsi untuk memelihara kesehatan jaringan epitel, apabila kekurangan menyebabkan keratinisasi (timbul lapisan tanduk).

f. Vitamin C

Vitamin C sangat berfungsi sebagai pembentuk kolagen dan membantu penyerapan besi dalam usus. Apabila kelebihan maka vitamin C akan dibuang melalui urine.

g. Kolesterol

Jumlah konsumsi maksimal kolesterol dalam satu hari adalah 300mg perhari.

2.1.2.5 Fitness

Fungsi *fitness* digunakan untuk mengetahui bagus atau tidaknya suatu individu, fungsinya adlaah sebagai berikut

$$fitness = f = \frac{1}{1 + pinalty} \qquad (2.1)$$

2.1.2.6 Seleksi

Seleksi adalah suatu metode yang digunakan dalam algoritma genetika dimana terdapat beberapa individu untuk diseleksi menjadi dua atau 1 individu saja.

Macam-macam dari seleksi adalah *Rank-based fitness assignment, Roulete wheel selection, Stochactic universal sampling, Local selection, Truncation selection* dan *Tournament selection.*

2.1.2.7 Roulette Wheel Selection

Roulette Wheel selection merupakan seleksi yang bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling fit. Pada metode ini, individu-individu dipetakan dalam satu segmen garis secara berurutan berdasarkan pada nilai komulatif berdasarkan pada fungsi fitness masing-masing kromosom. proses yang digunakan adlaah membangkitkan sebuah bilangan random dan akan memilih individu yang sesuai dengan bilangan random tersebut. Proses ini diulang hingga diperoleh sejumlah individu yang diharapkan (kusumadewi, 2003).

Langkah-langkah dari seleksi roda roulette adalah sebagai berikut :

1. Hitung fungsi *fitness* masing-masing kromosom (fi)
2. Hitung total *fitness* dari populasi (F)
3. Hitung probabilitas masing-masing kromosom

$$pi = \frac{fi}{F} \qquad (2.2)$$

4. Hitung probabilitas komulatif

Random nilai acak (r), bila r>q1 maka dipilih kromosom pertama, bila r > q2 maka dipilih kromosom pertama dan kedua sampai didapatkan r<qk. Seleksi pemutaran roda roulette sebanyak jumlah kromosom dalam populasi.

2.1.2.8 Operator Genetika

Terdapat dua operator genetika, yaitu :

a. Operator untuk melakukan rekombinasi, yang terdiri dari :
- Rekombinasi bernilai real
- Rekombinasi bernilai biner (*crossover*)
- Crossover dengan permutasi

b. Mutasi
- Mutasi bernilai real
- Mutasi bernilai biner

2.1.2.9 Penentuan Parameter

Parameter adalah parameter kontrol algoritma genetika, seperti ukuran populasi (*popsize*), probabilitas *crossover* (pc) dan probabilitas mutasi (pm).

2.1.3 Rekombinasi

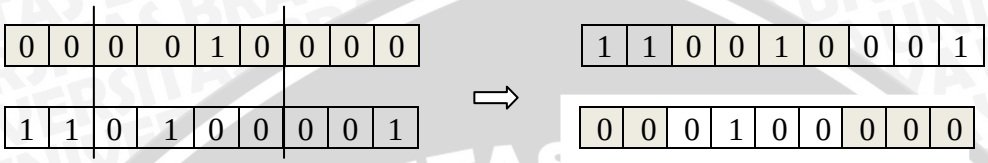
Rekombinasi adalah suatu metode yang digunakan untuk mendapatkan individu baru dengan melakkan dua orang tua, terdapat beberapa macam rekombinasi, yaitu rekombinasi diskret, rekombinasi menengah, rekombinasi garis, penyilangan satu titik, penyilangan mabyak titik, penyilangan seragam dan penyilangan dengan permutasi.

Crossover ini digunakan untuk penyilangan kromosom yang mempunyai banyak data, banyaknya titik pindah silang berpengaruh pada individu dipilih sebagai parent kemudian ditentukan titik pindah silangnya secara acak (Suyanto, 2007).

2.1.3.1 Twopoint Crossover

Twopoint crossover merupakan proses penyilangan kromosom dimana terdapat dua titik penyilangan yang nilainya dirandom antara[0,n-1] untuk

ditempatkan pada titik penyilangannya. *Crossover* dilakukan dengan membandingkan bilangan *random* dengan probablitas *crossover* (p_c) yang ditentukan sebelumnya, kemungkinan menu makanan sebanyak p_c*jumlah populasi. *Offspring* didapatkan dari segmen alternatif dari kedua *parent.* Berikut ini merupakan ilustrasi dari *multipoint crossover* : (Eiben dan Smith, 2003)



Gambar 2. 1 *Twopoint Crossover*

2.1.4 Mutasi

Mutasi merupakan suatu metode yang digunakan untuk mendapatkan individu baru sebagai anak (*offspring*) sehingga individu menjadi *heterogen.* Variable *offspring* dimutasi dengan menambahkan nilai *random* yang sangat kecil (ukuran langkah mutasi), dengan probabilitas yang rendah. Peluang mutasi (p_m) didefinisikan sebagai persentasi dari jumlah total gen pada populasi yang mengalami mutasi. Peluang mutasi mengendalikan banyaknya gen baru yang mungkin berguna tetapi tidak pernah dievaluasi.

fungsi dari mutasi adalah untuk menggantikan gen yang hilang dari populasi akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi.

2.1.4.1 Mutasi Biner

Mutasi biner adalah mutasi yang gennya berupa bilangan biner (0 atau 1). Cara sederhana untuk mendapatkan mutasi biner adalah dengan mengganti satu dengan beberapa nilai gen dari kromosom. Langkah-langkah mutasi ini adalah : (kusumadewi, 2003).

- Hitung jumlah gen pada populasi
- Pilih secara acak gen yang akan dimutasi
- Tentukan kromosom dari gen yang terpilih untuk dimutasi
- Ganti nilai gen (0 ke 1, 1 ke 0) dari kromosom yang akan dimutasi.

Pemilihan kromosom yang akan dimutasi adalah kromosom yang lebih kecil dari probabilitas mutasi (p_m), probabilitas mutasi merupakan peluang dari individu untuk mutasi. banyaknya gen yang yang diganti tidak pasti, tergantung dari jumlah random yang didapat. Berikut ini ilustrasi mutasi biner : (Eiben dan Smith, 2003)

Gambar2. 2 Mutasi Biner

2.2 Knapsack Problem

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 (Diah et all, 2010).

2.2.1 Fractional Knapsack Problem

Pada *fractional knapsack Problem*, terdapat suatu *knapsack* yang mempunyai batasan ukuran dan jumlah dari suatu item dengan perbedaan nilai dan ukurannya. Permasalahan ini memaksimalkan total nilai dari item yang ditempatkan pada *knapsack*. Item ini dapat dibagi menjadi satuan yang lebih kecil, seperti makanan dalam 100gr dapat dibagi menjadi satuan 1gr (Lemoine, 2011).

2.2.2 Multipledimensional Knapsack Problem (MDKP)

Multiple Dimensional Knapsack Problem atau dapat juga disebut sebagai *Multiconstraint knapsack Problem* atau *multiple-knapsack problem*, merupakan *knapsack problem* dengan beberapa *constraint*, seperti berat, harga, rasa, dan seterusnya. MDKP mempunyai beberapa kapasitas *knapsack* : W1, W2, W3,.., Wn dan n objek dengan masing-masing memiliki *cost* (cj), j=1,2,..,n. Berikut ini merupakan formula dari MDKP: (Gen dan Cheng, 2000)

$$\max \sum_{j=1}^{ni} c_j x_j \quad (2.3)$$

$$\text{s.t} \quad \sum_{j=1}^{ni} w_{ij} x_{ij} \leq W \qquad i=1,2,..m \quad (2.4)$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j$$

Dimana :

j = indeks item

ni = jumlah item di semua kelas

cj = *cost* item j

wij = berat item j yang berada di indeks i

xij = *feasible solution*

W = berat maksimum *knapsack*

Seperti permasalahn kombinatorial lainnya, MDKP juga mempunyai pendekatan dengan algoritma genetika sebagai solusi optimasi.

Seleksi proporsional diusulkan oleh Holland yang mengarah ke individu yang paling fit. Terdapat 2 metode popular dari jenis seleksi ini, yaitu Metode *Roulette Wheel Sampling* (RWS) dan *Stochastic Universal Sampling* (SUS) (Widodo, 2012). Untuk metode RWS akan dihitung probabilitasnya lalu akan dipilih secara random kromosom yang nantinya akan digunakan sebagai induk.

Operator *crossover* yang digunakan umumnya seksual dengan dua induk yang terseleksi x1(t) dan x2(t). Ada 3 jenis crossover penyajian biner yang sering digunakan, yaitu crossover satu titik, *crossover* dua titik dan crossover *uniform* (Widodo ,2012).

2.3 Makanan

Bahan-bahan yang ada disekitar kita digolongkan menjadi bahan yang dapat dimakan dan bahan yang tidak dapat dimakan, bahan yang dapat dimakan disebut dengan bahan makanan. Bahan makanan merupakan bahan yang masuk ke dalam tubuh melalui mulut maupun hidung dan merupakan bahan yang diperlukan untuk proses metabolisme dan pertumbuhan tubuh manusia (Suryatin, 2004).

Makanan merupakan sumber energi, nilai energi suatu makanan diperoleh dengan menghitung energi dari karbohidrat, lemak dan protein, menggunakan faktor umum untuk menghitung energi makanan adalah 4 untuk karbohidrat, 9

untuk lemak, dan 4 untuk protein, kemudian dijumlahkan. Nilai energi dinyatakan dalam kilokalori (kkal) (PERSAGI, 2005).

Gizi yang diperlukan tubuh dari makanan adalah protein, lemak, zat hidrat arang (karbohidrat), mineral dan berbagai komponen minor lainnya. Penggolongan bahan pangan di Indonesia adalah empat sehat lima sempurna, yaitu makanan pokok, lauk pauk, sayuran, buah-buahan dan susu (Buckle et all, 1987).

2.3.1 Pengelompokan Makanan

Pengelompokan makanan disesuaikan dengan pengelompokan yang digunakan dalam buku komposisi zat gizi makanan ASEAN (ASEAN Food Composition Tables). Penggunaan pengelompokan makanan tersebut dimaksudkan untuk kehar-monisan, (keselarasan) penyajian data komposisi zat gizi makanan di kawasan Asia Tenggara. Tabel-1 menunjukkan tabel pengelom-pokan makanan : (PERSAGI, 2005)

Tabel2. 1 Kelompok Makanan

NO	Kelompok Makanan	Kode
1.	Serealia dan hasil olahnya	A
2.	Umbi berpati dan hasil olahnya	B
3.	Kacang-kacangan dan hasil olahnya	C
4.	Sayuran dan hasil olahnya	D
5.	Buah dan hasil olahnya	E
6.	Daging, unggas dan hasil olahnya	F
7.	Ikan, kerang, udang dan hasil olahnya	G
8.	Telur dan hasil olahnya	H
9.	Susu dan hasil olahnya	J
10.	Lemak dan minyak	K
11.	Gula, sirup dan konfeksioneri	M

Huruf I dan L tidak digunakan untuk menghindari keliru-anggapan sebagai angka.

2.1.1 Zat Gizi yang Dicantumkan

Berbagai macam zat gizi yang dicantumkan ditunjukkan pada Tabel-2 adalah sebagai berikut : (PERSAGI, 2005).

Tabel 2.2 Zat Gizi yang Dicantumkan

NO	Zat Gizi	Satuan	Angka Desimal
1.	Air	G	1
2.	Energi	Kkal	Tidak ada
3.	Protein	G	1
4.	Lemak	G	1
5.	Karbohidrat	G	1
6.	Serat makanan	G	1
7.	Abu	G	1
8.	Kalsium	Mg	Tidak ada
9.	Fosfor	Mg	Tidak ada
10.	Besi	Mg	1
11.	Natrium	Mg	Tidak ada
12.	Kalium	Mg	Tidak ada
13.	Tembaga	mg	2
14.	Seng	mg	1
15.	Retinol (Vitamin A)	µg	Tidak ada
16.	B-karoten	µg	Tidak ada
17.	Tiamin (Vitamin B1)	mg	2
18.	Riboflavin (Vitamin B2)	mg	2
19.	Niasin	mg	1
20.	Vitamin C	mg	Tidak ada

2.3.2 Makanan Makronutrien yang dibutuhkan tubuh

Energi dalam diet diperoleh dari karbohidrat, protein, dan lemak. Ketiga zat gizi tersebut diberi nama makronutrien. Energi diperlukan untuk pertumbuhan, metabolisme, utilisasi bahan makanan, dan aktivitas. Kebutuhan energi disuplai terutama oleh karbohidrat dan lemak. Walaupun protein dalam diet dapat memberi

energi untuk keperluan tersebut, fungsi utamanya untuk menyediakan asam-amino bagi sintesis protein sel, dan hormon maupun enzim untuk mengatur metabolisme.

Susunan makanan untuk makronutrien penduduk amerika yaitu kabohidrat 46% dari total energi, lemak 42% dari total energi dan protein 12% dari total energi (Gaman dan Sherrington, 1981). Di Indonesia, susunan makanan makronutrien adalah karbohidrat 60-70%, protein 10-20% dan lemak 20-25% dari total energi (Almatsier, 2004).

Berikut ini adalah penjelasan tentang makanan makronutrien (Pudjiadi, 2000)

2.3.2.1 Karbohidrat

Karbohidrat yang penting bagi gizi ialah polisaharida (tepung), disaharida sukrosa dan laktosa, dan monosaharida glukosa dan fruktosa. Dalam diet terdapat pula sorbitol, manitol, dan silitol, bersama-sama dengan karbohidrat yang relative tidak dapat dicerna, seperti selilosa, hemilosa, dan pectin. Serat dalam makanan (dietary fiber) terdiri dari karbohidrat yang tidak dapat diserap. Serealia, sayur-mayur, buah-buahan dan kacang-kacangan merupakan sumber utama serat.

Fungsi dari karbohidrat adalah sebagai sumber energi utama, pengatur metabolisme lemak, penghemat fungsi protein, sumber energi utama bagi otak dan syaraf, simpanan sebagai glikogen dan pengatur peristaltic usus (Susanto dan Widyaningsih, 2004).

2.3.2.2 Protein

Protein yang didapat dari makanan sehari-hari harus diubah dulu menjadi asam amino sebelum dapat diserap dalam darah. Pencernaan protein dimulai dengan hidrolisasi ikatan peptidanya untuk menghasilkan asam amino. Berbagai enzim baik dari lambung (pepsin) maupun dari pancreas (tripsin, kemotripsin, dan lain-lain) diperlukan untuk proses hidrolisasi tersebut. Asam amino diserap melalui sel mukosa usus dan dibawa ke hepar melalui vena porta. Asam amino hasil hidrolisasi protein hewani dapat diserap lebih cepat dan lebih efisien jika dibandingkan dengan hasil hidrolisasi protein nabati.

Fungsi dari protein adalah sebagai zat pembangun untuk pertumbuhan dan pemeliharaan jaringan tubuh, sebagai pengatur proses dalam tubuh, sebagai sumber energi, sebagai pertahanan tubuh (sistem imun), penunjang mekanis dari kulit (kolagen), sebagai alat pengangkut dan penyimpanan (Susanto dan Widyaningsih, 2004).

2.3.2.3 Lemak

Kurang lebih 98% lemak dalam makanan sehari-hari terdiri dari trigliserida, ester daripada gliserol dalam 3 molekul asam lemak, selebihnya kolesterol, fosfolipida, sfingolipida, dan berbagai kompleks lipida yang mengandung karbohidrat (glikolipida) dan protein (lipoprotein). Kemudahan tercernanya lemak tergantung dari 3 faktor : panjangnya rantai asam lemak, derajat saturasi, dan strutura gliseridanya.

Fungsi dari lemak adalah penghasil energi, zat pembangun/ pembentuk struktur tubuh, protein sparer, penghasil asam lemak essensial, pelarut vitamin (Susanto dan Widyaningsih, 2004).

2.4 Perhitungan Energi

Perhitungan enerhi yang dibutuhkan *user* dalam sehari menggunakan metode Harris Bennedict dimana parameter yang digunakan adalah jenis kelamin, berat, tinggi, umur dan tingkat aktivitas *user* dalam satu hari. Berikut ini fungsinya, yaitu : (Buku Praktis Ahli Gizi Jurusan Gizi, 2008)

a. JENIS KELAMIN

1. Laki-laki :

   Basal Energy Expenditure (BEE) = 66.5 + 13.5 (Wt) + 5 (Ht) – 6.78 (A).

2. Wanita :

   Basal Energy Expenditure (BEE) = 655.1 + 9.66 (Wt) + 1.85 (Ht) – 4.68 (A)

   Dimana :

   Wt = Weight= berat badan sebenarnya dalam kg

   Ht = Height = tinggi badan dalam cm

   A = Age = usia dalam tahun

   BEE = Basal Energy Expenditure = Angka Metabolisme Basal (AMB)

b. Tingkat Aktifitas

Faktor koreksi BEE sesuai dengan tingkat aktifitas :

No	Tingkat Aktifitas	Kereksi BEE
1.	Bedrest	BEE x 1.2
2.	Ringan	BEE x 1.3
3.	Sedang	BEE x 1.4
4.	Berat	BEE x 1.5

2.5 Mikronutrien

Mikronutrien merupakan zat gizi yang dibutuhkan oleh tubuh dalam jumlah sedikit seperti besi, magnesium, fosfor, kalsium, vitamin C, vitamin A. Berikut ini merupakan tabel kebutuhan mikronutrien yang diperlukan manusia dalam satu hari : (Widya Karya Nasional Pangan dan Gizi, 2004)

Tabel2. 3 Angka Kecukupan Gizi 2004 bagi Orang Indonesia

NO	Kelompok Umur	Fe (mg)	Mg (mg)	Fosfor (mg)	Cal (mg)	Vit.C (mg)	Vit.A (RE)
Wanita							
1.	16-18 tahun	26	240	1000	1000	75	600
2.	19-29 tahun	26	240	600	800	75	500
3.	30-49 tahun	26	270	600	800	75	500
4.	50-64 tahun	12	270	600	800	75	500
5.	60+ tahun	12	270	600	800	75	500
Laki-laki							
6.	16-18 tahun	15	270	1000	1000	90	600
7.	19-29 tahun	13	270	600	800	90	600
8.	30-49 tahun	13	300	600	800	90	600
9.	50-64 tahun	13	300	600	800	90	600
10.	60+ tahun	13	300	600	800	90	600

2.6 Obesitas

Tingkat kesehatan gizi sesuai dengan tingkat konsumsi yang menyebabkan tercapainya kesehatan tersebut, namun apabila konsumsi gizi tidak sesuai dengan kebutuhan akan menyebabkan gizi kurang (*under nutrition*) atau kelebihan gizi (*over nutrition*). Obesitas merupakan keadaan yang melebihi dari berat badan relative seseorang akibata penumpukan gizi terutama karbohidrat, lemak, dan protein. Kondisi ini disebabkan oleh ketidakseimbangan antara konsumsi kalori dan kebutuhan energi. (Budiyanto, 2002)

Resiko dari obesitas adalah semakin tinggi tingkat obesitas maka semakin banyak pula penyakit yang terjadi. Tingkat infeksi pada penderita obesitas lebih tinggi jika dibandingan dengan yang normal, dan angka kematian yang berhubungan dengan infeksi pun demikian. Penyebab dari obesitas adalah kelebihan asupan energi dari yang seharusnya dikonsumsi, selain itu obesitas juga dipengaruhi dari hereditas, bangsa dan suku, gangguan emosi serta gangguan hormon. (Pudjiadi, 2000).

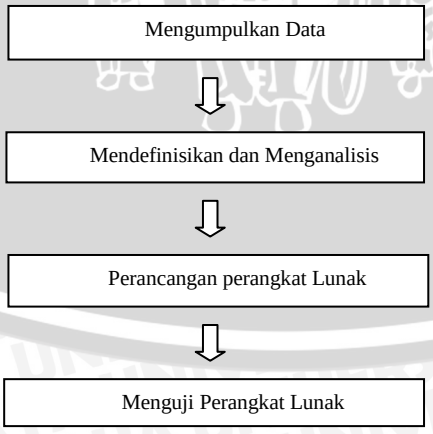
BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Pada bab ini akan dibahas metode, perancangan, dan langkah-langkah yang dilakukan dalam penelitian untuk menganalisis penerapan algoritma genetika untuk perencanaan menu.

Langkah-langkah yang dilakukan dalam penelitian ini meliputi :

1. Mengumpulkan data yang berhubungan dengan kandungan gizi dalam makanan dan algoritma genetika dari berbagai referensi.
2. Mendefinisikan dan menganalisis masalah yang dihadapi untuk mencari solusi yag tepat.
3. Membuat perancangan perangkat lunak dengan analisis terstruktur dan mengimplementasi hasil rancangan tersebut yaitu membuat perangkat lunak yang mempu menyusun menu makanan dengan menggunakan algoritma genetika.
4. Menguji perangkat lunak dan menganalisis hasil implementasi tersebut apakah sudah sesuai tujuan yang dirumuskan sebelumnya, untuk kemudian dievalusi dan disempurnakan.

Langkah-langkah penelitian ini dapat digambarkan seperti pada gambar 3.1



Gambar 3. 1 Langkah-langkah penelitian

3.1 Deskripsi sistem

Sistem perencanaan menu makanan sehat ini diharapkan mampu melakukan analisis terhadap kebutuhan gizi *user* sehingga dapat digunakan untuk solusi alternatif perencanaan menu makanan sehat sesuai dengan kebutuhan energi manusia dalam sehari berdasarkan kandungan gizi yang dibutuhkan manusia.

User hanya perlu menginputkan data diri yaitu umur, jenis kelamin, berat badan, tinggi badan, aktifitas dan makanan yang menyebabkannya alergi agar tidak muncul pada menu makanan. Sistem akan melakukan perencanaan menu sesuai dengan data yang dimasukkan oleh *user*. Data yang diinputkan oleh user akan diolah untuk digunakan pada proses selanjutnya, yaitu :

1. Kebutuhan energi dalam sehari, rumus yang digunakan adalah Metode Harris Bennedict pada bab II.
2. Menghitung pembagian bobot berdasarkan energi

   Pembagian bobot adalah 60% untuk karbohidrat, 15% untuk protein dan 25% untuk lemak. Dari masing-masing pembobotan tersebut akan dibagi berdasarkan waktu, pagi 35% dari total karbohidrat, lemak dan protein. Siang 35% dari karbohidrat, protein dan lemak. Sisanya 30% untuk malam hari. Proses ini selanjutnya digunakan untuk menentukan bobot proses fitness.

Suatu makanan mempunyai kandungan gizi yang berbeda-beda, sehingga untuk merancang suatu menu makanan diperlukan metode yang mempertimbangkan kandungan gizi serta beratnya. Algoritma genetika diharapkan dapat membuat menu makanan yang sesuai dengan konstrain standar pemenuhan gizi manusia.

Sistem ini akan menggunakan algoritma genetika, berikut ini adalah tahapan peyelesaian dalam algoritma genetika :

1. Representasi kromosom
2. Evaluasi dengan menghitung *fitness* dari masing-masing individu.
3. Proses *crossover* untuk mendapatkan individu baru.
4. Proses mutasi yang berfungsi untuk meningkatkan variasi dalam populasi.
5. Proses seleksi untuk membentuk populasi baru.

3.2 Perancangan Sistem

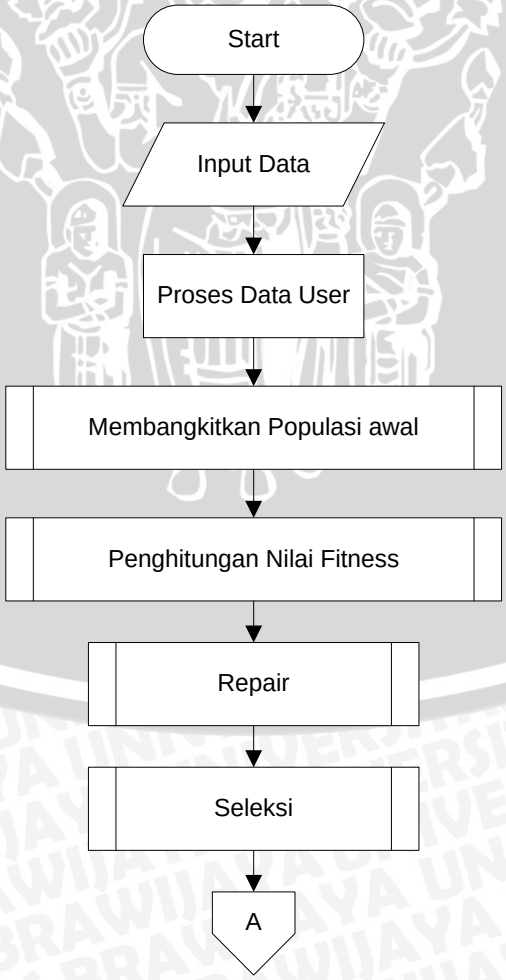
Langkah-langkah perancangan sistem menu makanan menggunakan algoritma adalah sebagai berikut

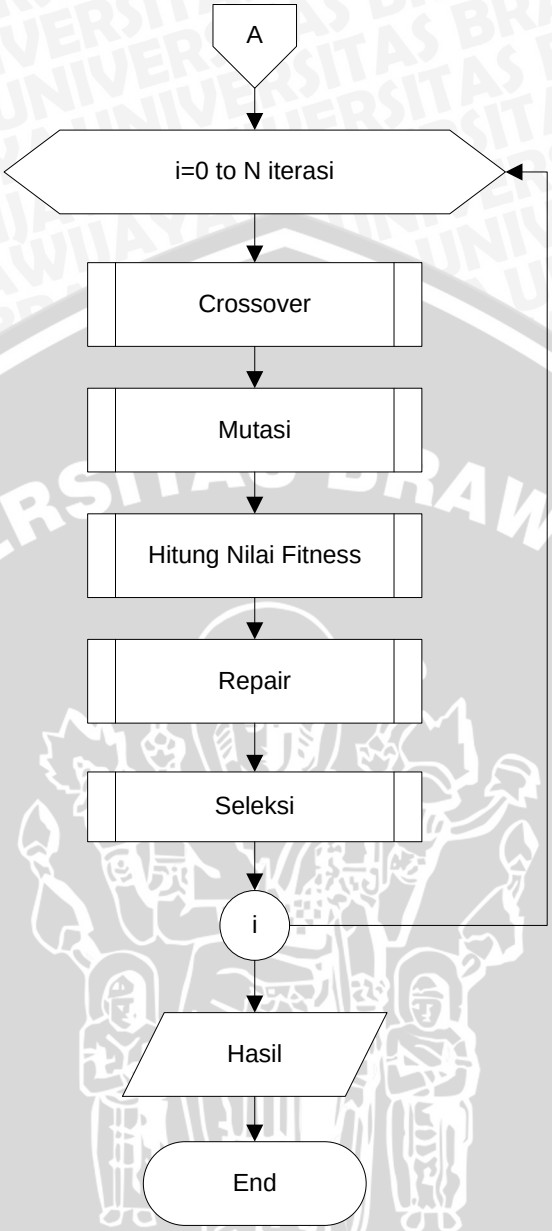
1. *User* menginputkan data jenis kelamin, umur, dan tingkat aktifitas untuk menghitung kebutuhan kalori yang dibutuhkan.
2. Setelah itu dilakukan pemrosesan data *user* untuk mengetahui energi yang dibutuhkan dalam sehari, serta mengetahui berapa jumlah berat makanan yang harus dimakan dalam sehari
3. Selanjutnya dilakukan proses *generate* kromosom sebanyak populasi yang ditetapkan. Representasi kromosom menggunakan *array-multi dimensi* (dua dimensi), dimensi pertama untuk (makanan pokok pagi, lauk nabati pagi , lauk hewani pagi, sayur dan buah pagi. Dilanjutkan untuk siang dan malam. Dimensi kedua digunakan untuk merepresentasikan indeks item makanan sejumlah n.
4. Tahap selanjutnya adalah perhitungan nilai *fitness* dimana sebelum perhitungan nilai *fitness* harus melakukan proses pembobotan *density*, pembobotan berat jenis makanan, dan pembobotan pe*nalty*. Semakin kecil *fitness* maka semakin baik suatu individu
5. Setelah perhitungan nilai *fitness* akan dilakukan proses *repair* pada setiap gen yang bernilai 1 akan tetapi tidak terpilih menjadi menu makanan, maisng-masing gen yang tidak terpilih akan diubah nilainya dari 1 menjadi 0 (dari terpilih menjadi tidak terpilih).
6. Selanjutnya adalah proses seleksi menggunakan *roulette wheel selection* untuk menentukan kromosom yang digunakan sebagai induk. seleksi menggunakan roda roulette digunakan agar dapat memberikan kesempatan bagi individu yang kurang baik terpilih menjadi *parent* untuk dilakukan proses selanjutnya.
7. Setelah terpilih 2 kromosom maka selanjutnya adalah proses *crossover*, *crossover* yang digunakan adalah *two point crossover* berdasarkan pada probabilitas *crossover*. Proses *crossover* dilakukan apabila probabilitas *crossover* lebih kecil dari nilai *random* antara (0-1).
8. Berikutnya adalah proses mutasi, mutasi yang digunakan adalah mutasi biner dengan probabilitas mutasi sebagai peluang terjadinya mutasi dalam populasi

dimana titik mutasi akan dilakukan secara random sebanyak dua nilai. Proses mutasi dilakukan apabila probabilitas mutasi lebih kecil dari nilai *random* antara (0-1).

9. Selanjutnya adalah melakukan perhitungan nilai *fitness* dari hasil proses mutasi.
10. Setelah perhitungan *fitness*, dilakukan *repair* pada individu yang tidak *valid* agar tidak terdapat 2 item makanan yang terpilih.
11. Setelah itu dilakukan seleksi untuk menentukan kromosom yang digunakan crossover. Proses 7-8 dilakukan sejumlah iterasi.
12. Tahap terakhir adalah menampilkan hasil dari proses yang diharapkan menjadi solusi dari permasalahan menu makanan.

Penyelesaian perencanaan penentuan menu makanan digambarkan pada gambar 3.2





Gambar 3. 2 Perencanaan penentuan menu makanan

3.2.1 Membangkitkan Populasi Awal

Satu kromsom memiliki banyak gen-gen yang dikodekan dengan *string* biner untuk representasi permasalahan penentuan menu makanan. *String* biner menyatakan gen dalam kromosom dipilih atau tidak (1 atau 0), masing-masing gen mempunyai indeks untuk mewakili menu makanan yang terdapat dalam kromosom.

Setiap kromosom menggunakan MDKP untuk merepresen-tasikan menu makanan yang terdiri dari menu makanan pagi, makan siang dan makan malam. Waktu makan disesuaikan dengan kebiasan makan penduduk Indonesia. Serta *fractional knapsack problem* untuk mengolah data makanan yang terpilih dari masing-masing kromosom. Langkah yang digunakan adalah membangkitkan populasi awal dengan membangkitkan bilangan biner pada setiap indeks gen n dalam kromosom. Gambar 3.3 menggambarkan bentuk gambaran dari kromosom.

Indeks Makanan		**0**	**1**	**2**	**3**	**5**	**n-1**
Makan Pagi (W1)	Makanan Pokok	0	0	0	1	1	0
	Lauk Nabati	0	1	1	0	0	1
	Lauk Hewani	0	1	1	1	0	0
	Sayur	1	0	1	0	0	0
	Buah	1	0	1	0	0	0
Makan Siang (W2)	Makanan Pokok	1	1	0	0	0	0
	Lauk Nabati	1	0	0	0	0	1
	Lauk Hewani	1	1	0	1	1	1
	Sayur	1	0	1	1	1	0
	Buah	0	1	0	1	1	1
Makan Malam (W3)	Makanan Pokok	1	0	1	1	1	0
	Lauk Nabati	0	0	0	1	0	0
	Lauk Hewani	1	1	1	1	1	0
	Sayur	0	1	0	1	0	1
	Buah	1	1	0	1	0	1

Gambar 3. 3 Gambaran kromosom

Keterangan :

Makan Pagi (W1) : menu makanan pagi dengan kapasitas maksimal bobot sebesar W1.

Makan Siang (W2) : menu makanan siang dengan kapasitas maksimal bobot sebesar W2.

Makan Malam (W3) : menu makanan malam dengan kapasitas maksimal W3.

Dimana pembagian W1, W2dan W3 bergantung pada kebutuhan energi yang dibutuhkan user dalam satu hari.

Representasi dari kromosom diatas menggunakan struktur data array multidimensi (array dua dimensi), representasinya dapat dilihat pada gambar 3.4 sebagai berikut

	0	1	2	3	4	5	6	...	N-1
0									
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									

Gambar 3. 4 Representasi kromosom menggunakan array 2 dimensi

Pada gambar representasi kromosom diatas, indeks horisontal merepresentasikan item makanan seperti nasi, bubur, soto, ayam. Indeks vertikal merepresentasikan kombinasi jenis makanan dan waktu, sehingga indeks 0 untuk makanan pokok pagi, indeks 1 untuk lauk nabari pagi, indeks 2 untuk lauk hewani pagi, indesk 3 untuk sayur pagi dan indeks 4 untuk buah pagi. Diteruskan indeks kelima sampai 9 untuk makan siang, sisanya untuk makan malam.

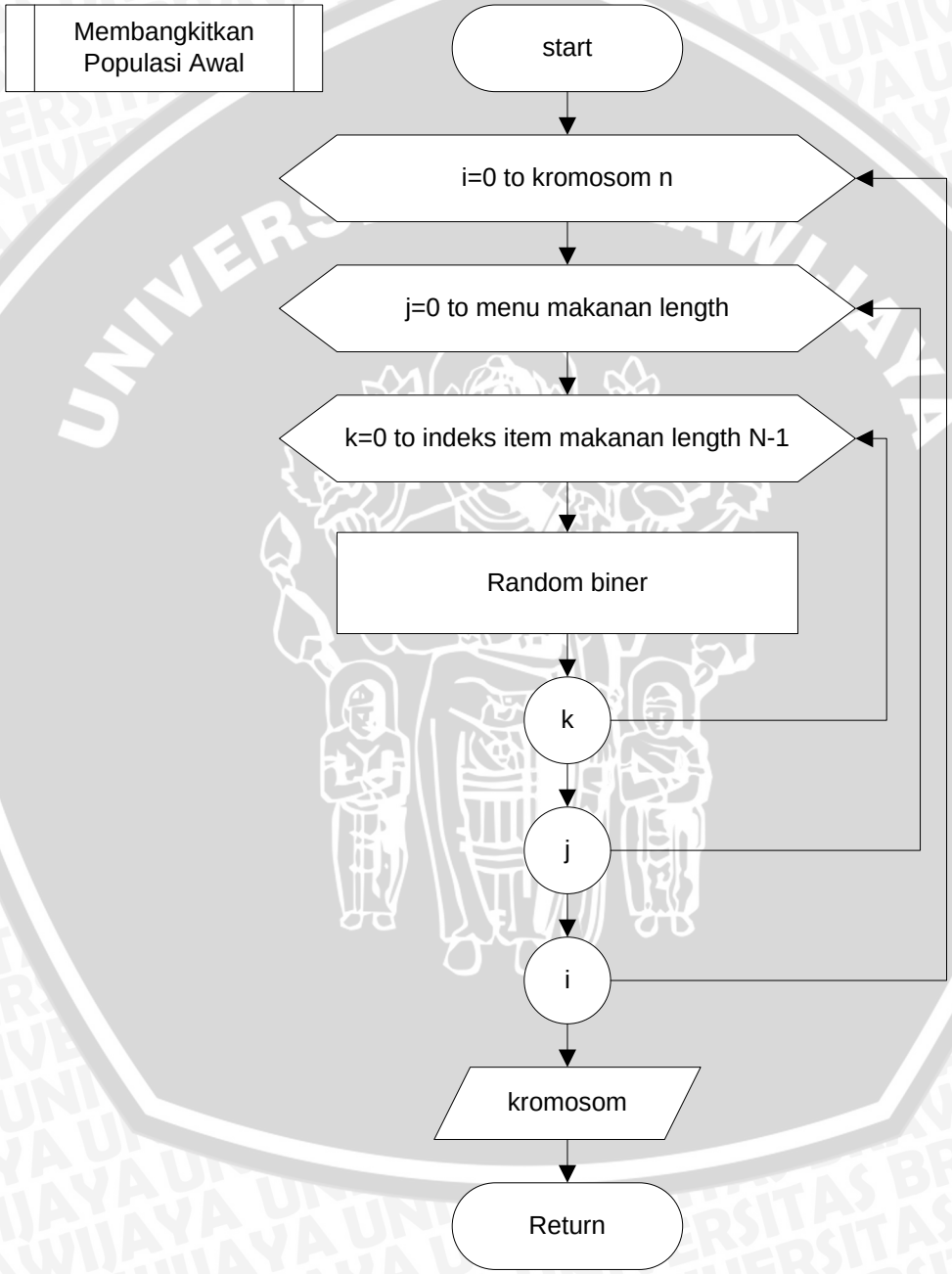
Langkah-langkah pembangkitan populasi adalah

1. Melakukan random bilangan 0 atau 1 pada gen sepanjang item makanan (N-1) pada 1 menu makanan.
2. Random diteruskan untuk menu makanan (0-14) selanjutnya hingga terbentuk satu kromosom, indeks 0 digunakan untuk menu makanan pokok pagi, indeks 1 untuk lauk nabati pagi, indeks 2 untuk lauk hewani pagi, indeks 3 untuk

sayur pagi dan indeks 4 untuk buah pagi. Selanjutnya diteruskan untuk makan siang dan malam.

3. Random berhenti apabila sudah terbentuk beberapa kromosom.

Flowchart dari proses pembangkitan populasi awal dapat dilihat pada gambar 3.5



Gambar 3. 5 Membangkitkan populasi awal

3.2.2 Perhitungan Nilai Fitness

Setelah representasi kromosom, selanjutnya adalah menghitung nilai *fitness* dari masing-masing kromosom. Tahapan perhitungan *fitness* adalah sebagai berikut :

3.2.2.1 Pembobotan density

Pembobotan *density* merupakan pembobotan yang memaksimalkan keuntungan dengan memilih objek yang memiliki keuntungan per-unit berat terbesar. Pembobotan density yaitu dengan menjumlahkan semua kandungan makanan, protein, karbo, lemak, vitamin A, vitamin C, vitamin B1, natrium, kalium, fosfor, besi, kalsium, serat dan dikurangi dengan kolesterol yang terkandung dalam makanan.

3.2.2.2 Pembobotan berat jenis makanan

Pembobotan berat jenis makanan (makanan pokok, lauk nabati, lauk hewani, sayur dan buah) menggunakan fractional knapsack problem berdasarkan kebutuhan makananya. Langkah-langkahnya adalah sebagai berikut :

1. *Random* 5 bilangan antara 15 sampai banyaknya berat makanan, berat makanan dapat dilihat pada proses data *user*. Hasil random tidak boleh melebihi atau sama dengan berat makanan.
2. *Sorting* menurun berat makanan dan berat terbesar untuk makanan pokok, selanjutnya lauk nabati, lauk hewani, sayur dan terakhir buah.

3.2.2.3 Pembobotan penalty

Dalam pembobotan *penalty* dilakukan berdasarkan kandungan dari makanan yang terpilih, makronutrien merupakan zat dalam makanan yang paling penting dibutuhkan oleh tubuh, sehingga faktor pengalinya sebesar 5. Sehingga semakin besar pelanggaran, maka penaltynya semakin besar pula. Serat dibutuhkan oleh tubuh dengan perbandingan 1000 kal minimal 10 gr serat dan tidak diperbolehkan melebihi 40gr. Sehingga apabila berat serat kurang dari seharusnya, maka faktor pengali 2 karena termasuk penalty sedang, dan apabila lebih dari 40gr maka faktor pengalinya adalah 3 karena termasuk penalty berat.

Besi, kalsium dan vitamin A dipengaruhi oleh jenis kelamin dan umur (tabel 2.2), sehingga faktor pengali yang digunakan sebesar 2 apabila kandungan melebihi seharusnya, dan faktor pengali 3 apabila kurang dari seharusnya. Penalty fosfor sama dengan jumlah selisih karena fosfor terdapat di semua makanan sehingga penalty yang digunakan sebesar selisih dari berat didapat dikurangi berat seharusnya. Vitamin C merupakan vitamin yang apabila dikonsumsi lebih dari seharusnya akan dibuang melalui urin, oleh karena itu faktor pengali hanya diberlakukan apabila kandungan vitamin C kurang dari seharusnya dan termasuk penalty yang besar. Konsumsi kolesterol memiliki jumlah maksimal konsumsi perhari yaitu 300mg, sehingga apabila kolesterol melebihi 300mg maka akan dilakukan penalty besar dengan faktor pengali sebesar 3. Dan yang terakhir adalah makanan yang sama, dalam satu hari diusahakan menu makanan tidak sama, kecuali makanan pokok. Sehingga digunakan faktor pengali sebesar 10 yang menyatakan pelanggaran untuk menu makanan yang sama adalah sangat besar. Berikut ini merupakan tabel pembobotan penalty, dapat dilihat pada tabel 3.1.

Tabel 3. 1 Pembobotan Penalty

NO	Keterangan	Fungsi	
1.	Makronutrien	$pj = \frac{tm - tn}{tm} \times 5 + 1$	(3.1)
2.	Selisih	$ss = \frac{wh - wa}{wh}$	(3.2)
3.	Serat	$pj = \begin{cases} ws < wss\ gram, ss \times 2 \\ wss \leq ws < 40, ss \times 1 + 1 \\ ws \geq 40, ss \times 3 \end{cases}$	(3.3)
4.	Besi	$pj = \begin{cases} wb < wbs, ss \times 3 \\ wb = wbs, ss \times 1 + 1 \\ wb > wbs, ss \times 2 \end{cases}$	(3.4)
5.	Fosfor	$pj = ss$	(3.5)
6.	Kalsium	$pj = \begin{cases} wk < wks, ss \times 3 \\ wk = wks, ss \times 1 + 1 \\ wk > wks, ss \times 2 \end{cases}$	(3.6)
7.	Vitamin A	$pj = \begin{cases} wa < was, ss \times 3 \\ wa = was, ss \times 1 + 1 \\ wa > was, ss \times 2 \end{cases}$	(3.7)

8.	Vitamin C	$pj = wc < wcs, ss \times 3$	(3.8)
9.	Kolesterol	$pj = wkl > 300, ss \times 5$	(3.9)
10.	Makanan yang sama selain makanan pokok	10	

Keterangan :

tm = total masing-masing makronutrienseharusnya.

tn = berat masing-masing makronutrien yang didapatkan

wh = berar seharusnya

wa = berat yang didapat

ws = berat serat yang didapatkan

wss = berat serat minimal dalam sehari

wb = berat besi yang didapat

wbs = berat besi seharusnya dalam sehari

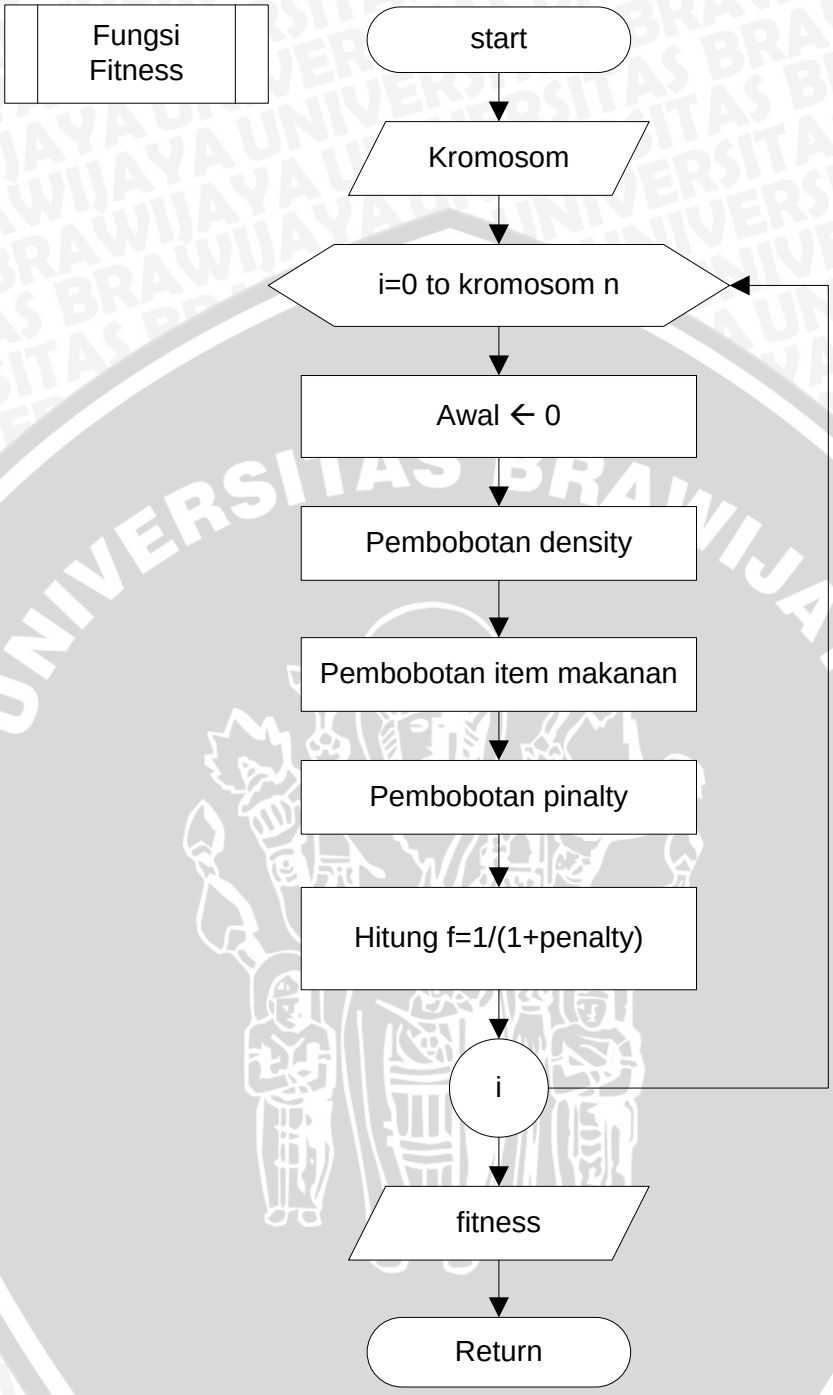
wk = berat kalsium yang didapat

wks = berat kalsium seharusnya dalam sehari

3.2.2.4 Perhitungan fitness

Perhitungan *fitness* bertujuan untuk mendapatkan *fitness* maksimal, semakin besar *penalty* yang didapatkan semakin buruk suatu individu. Fungsi *penalty* dapat dilihat pada *equation* 2.13.

Flowchart dari fungsi hitung fitness ditunjukkan pada gambar 3.6



Gambar 3. 6 Proses Hitung Fungsi Fitness

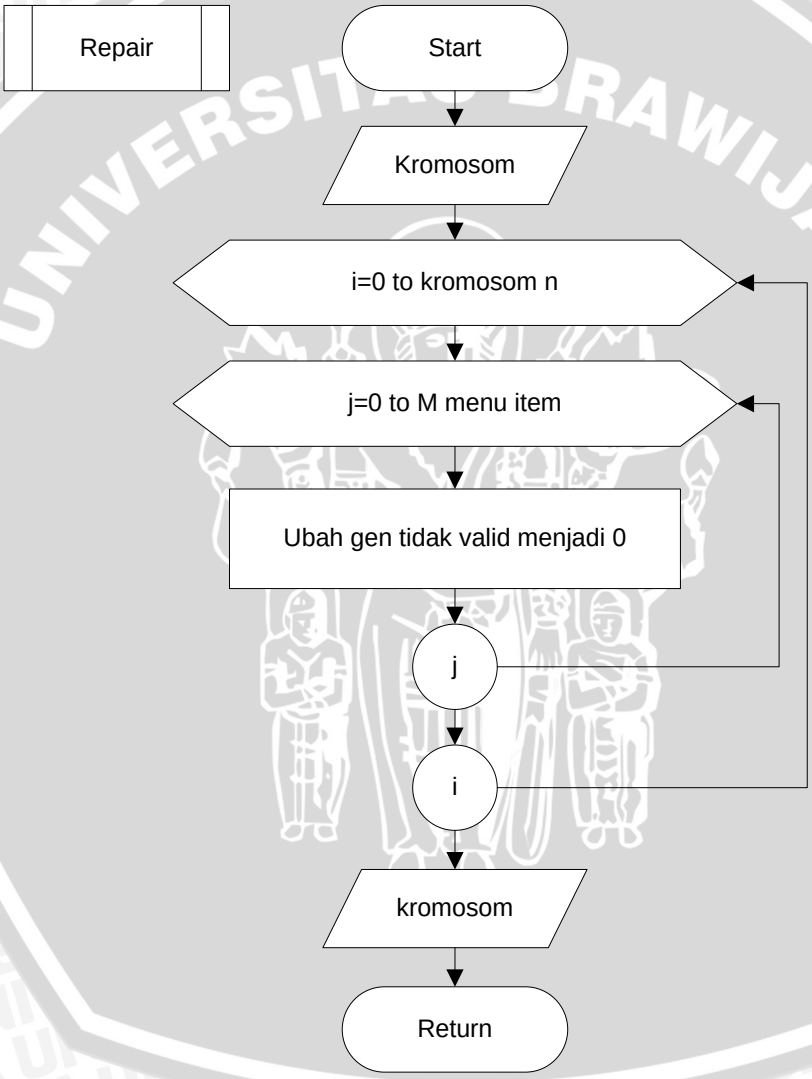
3.2.3 Repair

Proses *repair* dilakukan setelah perhitungan *fitness* dimana gen yang tidak terpilih saat proses *fitness* akan diganti dengan 0, dari 1 (terpilih) menjadi 0 (tidak terpilih) sehingga hanya satu gen yang tepilih dalam 1 menu makanan..

Langkah-langkah repair adalah sebagai berikut

1. Masing-masing kromosom dichek satu persatu untuk melihat adanya gen *invalid*.
2. Mengubah gen invalid menjadi tidak terpilih dengan nilai 0.
3. Proses diatas dilakukan sebanyak kromosom .

Flowchart proses *repair* ditunjukkan pada gambar 3.7



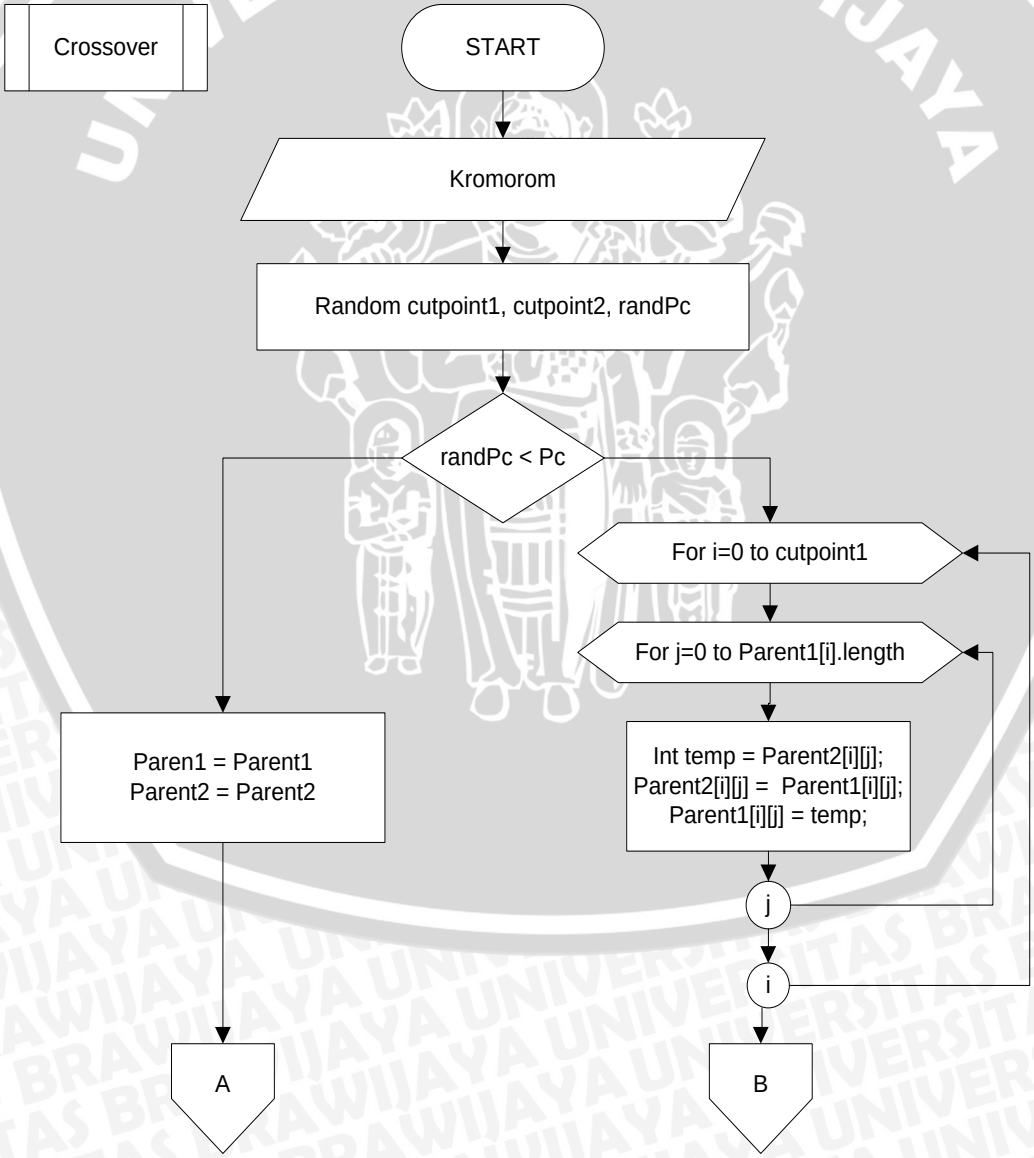
Gambar 3. 7 Flowchart Proses Repair

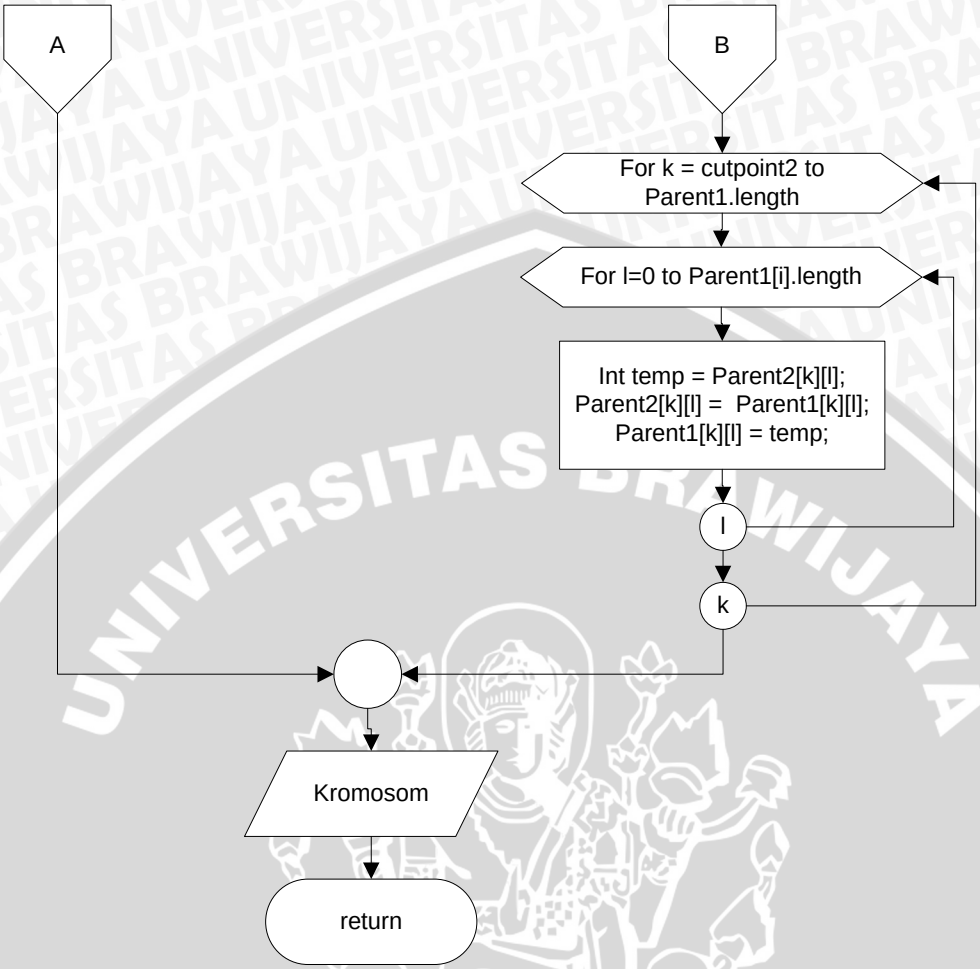
3.2.4 Crossover

Crossover yang digunakan adalah *crossover* dua titik, dimana dipilih satu titik secara *random* antara (0,n-1) sebagai titik penyilangan. Langkah *crossover* adalah sebagai berikut :

1. Merandom 2 bilangan antar 0 sampai n-1 (panjang kromosom)
2. Melakukan pertukaran pada daerah yang sama, gen kromosom1 sampai *cut-point1* ditukarkan dengan gen kromosom2, sedangkan gen dari *cutpoint2* sampai panjang kromosom akan ditukarkan seperti cutpoint1.

Flowchart *crossover* pada gambar 3.8





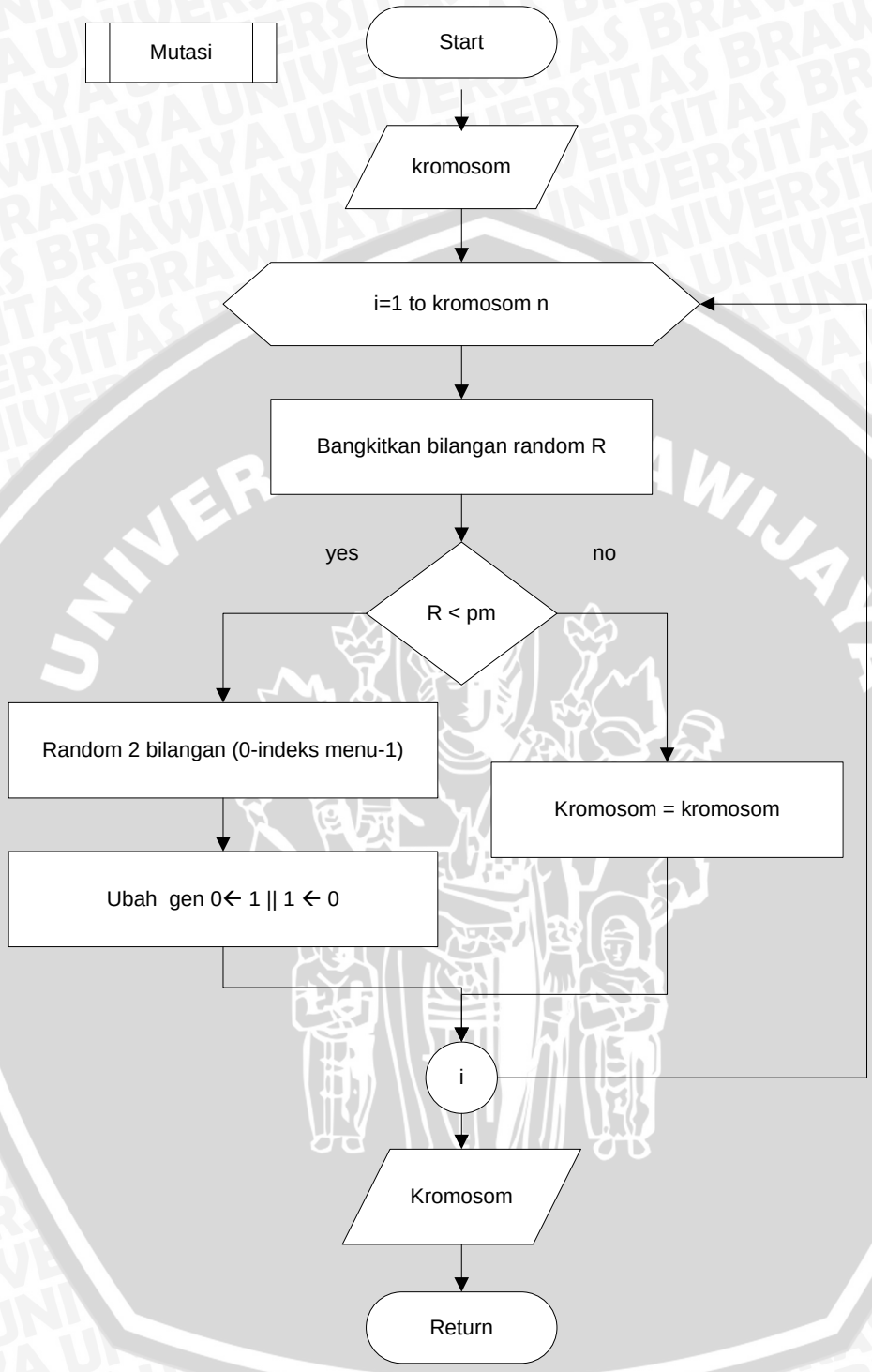
Gambar 3. 8 Proses Crossover

3.2.5 Proses Mutasi

Pada proses mutasi akan dirandom antara 0 dan 1, apabila nilai random lebih kecil dan probabilitas populasi maka mutasi akan dilakukan. Nilai probabilitas didapatkan dari fungsi *fitness* kromosom dibagi dengan jumlah *fitness* populasi. Langkah-langkah proses putasi adalah sebagai berikut :

1. Menghitung probabilitas kromosom.
2. Melakukan *random* antara 0 sampai 1.
3. Jika nilai *random* lebih kecil dari probabilitas mutasi, dilakukan random bilangan dari 0 sampai $n-_1$ untuk menempatkan posisi gen yang dimutasi. Banyak nilai random adalah 2 nilai.
4. Mengganti nilai 0 menjadi 1 atau 1 menjadi 0.

Ilustrasi dari proses mutasi dapat dilihat dari gambar 3.9



Gambar 3. 9 Flowchart Proses Mutasi

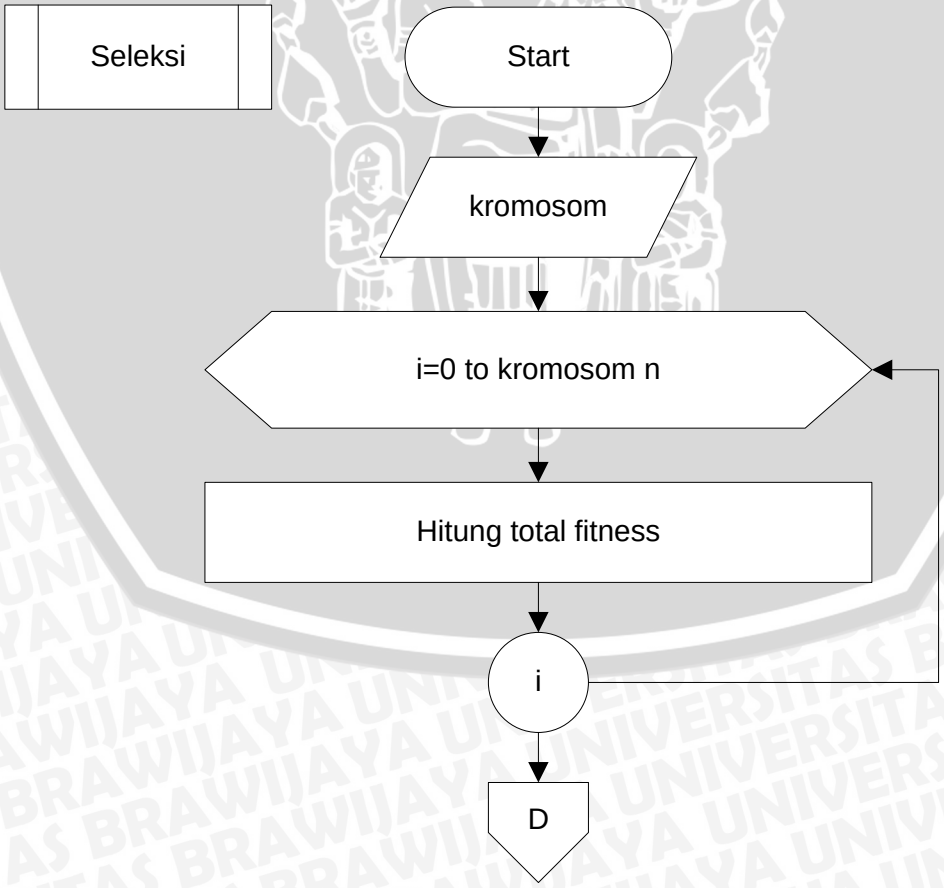
3.2.6 Seleksi

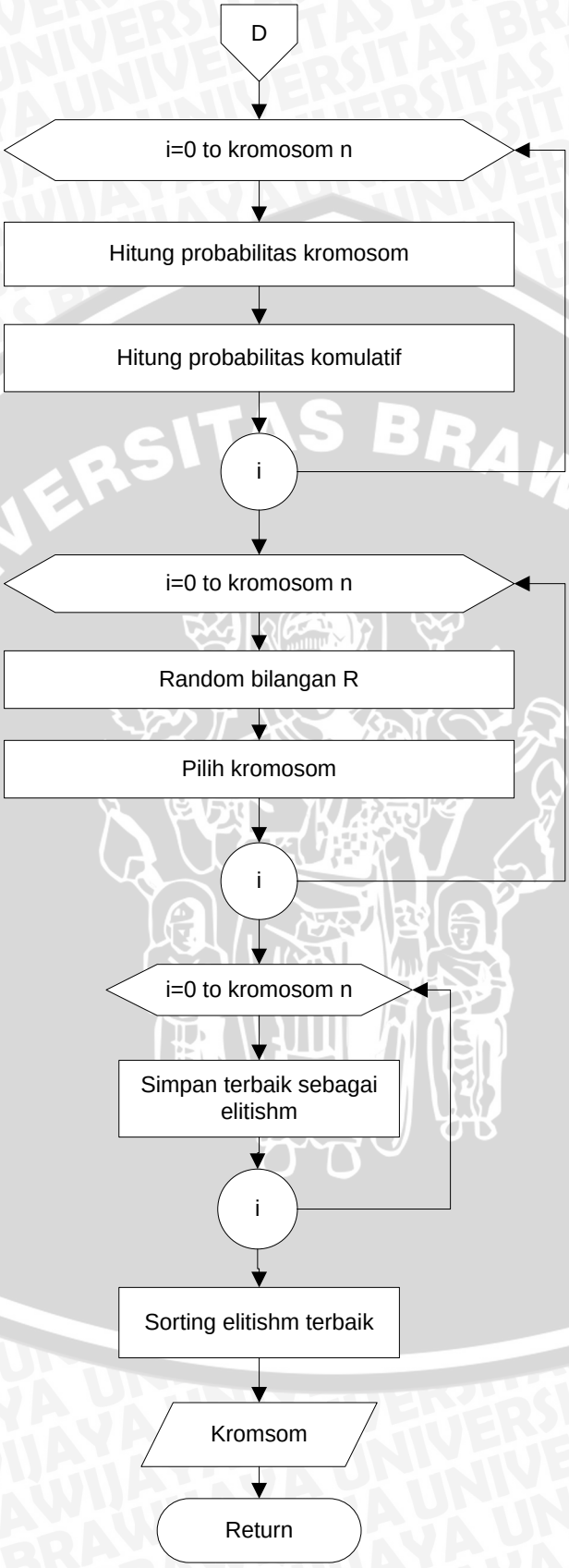
Seleksi yang digunakan adalah seleksi roda roulette (*Roulette Wheel*), metode ini sering digunakan untuk memberikan peluang reproduksi bagi semua

anggota populasi. Langkah-langkah dari seleksi *roulette wheel* adalah sebagai berikut :

1. Menghitung total *fitness* dari semua kromosom yang akan diseleksi.
2. Menghitung probabilitas masing-masing kromosom, fungsi probabilitas dapat dilihat pada equation 2.14.
3. Setelah diketahui probabilitas masing-masing kromosom, dilakukan perhitungan komulatif.
4. Melakukan *random* nilai acak (r), bila r>q1 maka dipilih kromosom pertama, bila r > q2 maka dipilih kromosom pertama dan kedua sampai didapatkan r<qk. Seleksi pemutaran roda roulette sebanyak jumlah kromosom dalam populasi.
5. Menyimpan nilai tertinggi untuk menjadi elitshm, dengan inisiasi awal elitshm = 0.

Flowchart Seleksi ditunjukkan pada gambar 3.10





Gambar 3. 10 Proses Seleksi

3.3 Perhitungan Manual

3.3.1 Proses Data User

Contoh perhitungan manual kali ini, data dari user adalah seorang wanita umur 22 tahun, berat 61 dengan tinggi 165cm dan aktivitas sedang. Langkah pertama adalah perhitungan energi sehingga digunakan metode Hariris Benedict.

Langkah pertama adalah perhitungan energi, penghitungan energi menggunakan Metode Harris Bennedict, berdasarkan data yang diinputkan oleh user dan diproses berdasarkan rumus menghasilkan :

Basal Energy Expenditure (BEE) = 655.1 + 9.66 (Berat) + 1.85 (Tinggi) – 4.68 (Umur) = 1456.31

Energi (E) = BEE x Aktivitas = 1456.31 * 1.4 = 1880.645

Setelah menghitung energi yang dibutuhkan, selanjutnya adalah pembagian kalori berdasarkan makronutrien. Perhitungan kalori berdasarkan aturan pembagian makronutrien, yaitu karbohidrat 60-70% dari total energi, protein 10-20% dari total energi dan lemak 15-25% dari total energi. perhitungannya adalah sebagai berikut :

KARBOHIDRAT = 60% x ENERGI = 1128.3870000000002

PROTEIN = 15% x ENERGI = 282.09675000000004

LEMAK = 25% x ENERGI = 470.16125000000005

Selanjutnya adalah pembagian kalori dan pembobotan makronutrien berdasarkan waktu makan, pembobotan makronutrien berdasarkan waktu makan adalah 35% dari total makronutrien untuk pagi hari karena badan membutuhkan banyak energi untuk memulai aktifitas, 35% dari total makronutrien untuk energi saat beraktifitas dan sisanya 30% untuk malam hari untuk waktu beristirahat. Perhitungan berat makanan dari masing-masing waktu makan menggunakan faktor pengali, yaitu 4 untuk karbohidrat, 4 untuk protein dan 9 untuk lemak. Faktor pengali digunakan untuk mengetahui banyaknya makanan atau berat makanan, 1 gram karbohidrat menghasilkan 4 kalori. 1 gram protein menghasilkan 4 kalori, dan 1 gram lemak menghasilkan 9 kalori. Sehingga untuk mengetahui berapa berat masing-masing makronutrien adalah dengan membagi

kalori dengan faktor pengali dari masing-masing makronuutrien. Hasil dari proses pembagian kalori dan pembobotan makronutrien diatas ditunjukkan pada Tabel 3.2. Tabel Pembagian Makronurien :

Tabel 3. 2 Tabel Pembagian Makronutrien

PAGI	**KANDUNGAN**	**TOTAL ENERGI**	**FAKTOR PENGALI**	**BERAT TOTAL**	**BERAT MAX**
	KARBOHIDRAT	98.7338625	4.000	63.704	
	PROTEIN	24.6834656	4.000	15.926	
	LEMAK	18.2840486	9.000	11.797	141.70138
SIANG	**KANDUNGAN**	**TOTAL ENERGI**	**FAKTOR PENGALI**	**BERAT TOTAL**	**BERAT MAX**
	KARBOHIDRAT	98.7338625	4.000	63.704	
	PROTEIN	24.6834656	4.000	15.926	
	LEMAK	18.2840486	9.000	11.797	141.70138
MALAM	**KANDUNGAN**	**TOTAL ENERGI**	**FAKTOR PENGALI**	**BERAT TOTAL**	**BERAT MAX**
	KARBOHIDRAT	84.6290250	4.000	54.603	
	PROTEIN	21.1572563	4.000	13.651	
	LEMAK	15.6720417	9.000	10.112	121.45832

Setelah itu dilakukan pembobotan energi (kalori) dalam satu hari, 35% utuk pagi dan siang hari dan sisanya untuk malam hari. Misal dalam perhitungan energi total dalam satu hari adalah 404.86

Tabel 3.3 Tabel pembagian kalori

WAKTU MAKAN	**TOTAL ENERGI**	**FAKTOR PENGALI**	**BERAT TOTAL**	**BERAT MAX**
KARBOHIDRAT	1880.645	35 %	141.7013	
PROTEIN	1880.645	35 %	141.7013	
LEMAK	1880.645	30 %	121.4583	404.86

3.3.2 Membentuk populasi awal

Pembentukan populasi awal dilakukan dengan merandom biner (0/1) sejumlah n baris item. Individu mempunyai 15 barin dan n kolom, dimanan baris merepresentasikan waktu makan dan jenis makanannya, sedangkan n kolom merepresentasikan menu makanan. 1 untuk diambil dan 0 tidak diambil. Misal pada baris 1 dari contoh Parent1, menu makanan yang diambil adalah pada indeks 0,1,2,3,4,5,7,8 dan 9. Begitu juga untuk baris selanjutnya sampai baris ke-15. Berikut ini contoh pembentukan populasi awal, yaitu :

Parent1

111111011100000000000000000000000000000000000000
111111110000000000000000000000000000000000000000
010111011010111011100101001101110111000110101010 0
001100111110000000000000000000000000000000000000
011001111000101100011100100011001001001111010000 0
010110011000000000000000000000000000000000000000
100110110000000000000000000000000000000000000000
100001100110100000100111110010111001100111101000 0
110100110111000000000000000000000000000000000000
010101111000110101101000010100011101110001000000 0
000001101110000000000000000000000000000000000000
010011111000000000000000000000000000000000000000
000111010001000110111100111101110010101100110100 1
110111011101000000000000000000000000000000000000
011100011111100101111010011100010101100011000000 0

Parent2

100100011000000000000000000000000000000000000000
100100110000000000000000000000000000000000000000
111011110110000100111010011101101001010001001100 1
101101111001000000000000000000000000000000000000
010110010000011101110001001011111110001000100000 0
100000010110000000000000000000000000000000000000
010000100000000000000000000000000000000000000000
101101111011000000000101001100000000000111101011
011111000010000000000000000000000000000000000000
001111101111101001010001101000100011110000000000
100010001110000000000000000000000000000000000000
000101110000000000000000000000000000000000000000
000000111111101010110000100011001100100001110111 1
001111111101000000000000000000000000000000000000
000011111100100111111011001111000111000010111000 00

Parent3

00101010001000000000000000000000000000000000000000
11111111000000000000000000000000000000000000000000
00110101110100101100010000011001011001010101011101
11011001001000000000000000000000000000000000000000
10111000011011011110010100010110011100000011100000
11011100101000000000000000000000000000000000000000
01001110000000000000000000000000000000000000000000
10100101110000011110010110100000110101110110111100
00101011011000000000000000000000000000000000000000
10010101000010111100010110011110001111100001000000
10100110011000000000000000000000000000000000000000
01100001100000000000000000000000000000000000000000
01110011101101011110000011011111101011011111010101
01000111110000000000000000000000000000000000000000
00001101001011100101100111000111001100100111000000

3.3.3 Perhitungan Nilai fitness

Perhitungan fitness berdasarkan pada individu 1 dan individu 2, individu yang dijadikan parent dapat dilihat pada gambar pada pembentukan populasi awal. Bahan makanan contoh dapat dilihat pada lampiran 1. Pertama yang harus dilakukan adalah menghitung nilai density pada masing-masing individu untuk mengetahui gen mana yang mempunyai kandungan gizi paling dibutuhkan. Perhitungan density dapat dilihat pada *equation* 2.1, 2.2 dan 2.3. berikut ini adalah contoh salah satu perhitungan density bahan makanan. Misal pada individu pertama untuk makanan pokok pada index ke-0 adlaah nasi goring, indeks ke-4 adalah nasi jagung dan indeks ke-5 adalah nasi uduk. Density dihitung dari kandungan gizi masing-masing bahan makanan.

Contoh perhitungan density bubur :

Kandungan karbohidrat = 0.16, protein = 0.013, lemak = 0.001, kalsium = 0.02, fosfor = 0.21, besi = 0.001, vitaminA = 0, vitamin = 0, Natgrium = 0, kalium = 0 dan serat = 0.002. maka perhitungannya adalah

Density = karbo + protein + lemak + kalsium + fosfor + besi + vitaminA + vitamin + natrium + kalium + serat;

Density = 0.407

Perhitungan dilakukan untuk semua bahan makanan. Hasilnya adalah sebagai berikut :

Tabel 3. 3 Density Individu 1

	Makanan	**Density**
PAGI	Makanan Pokok	1.003, 0.407, 0.97, 0.717, 1.054, 0.645, 1.821, 1.744, 0.457
	Lauk Nabati	3.635, 2.476, 2.174, 5.595, 1.962, 1.378, 2.235, 1.771
	Lauk Hewani	1.524, 4.844, 2.826, 2.846, 0.374, 2.024, 2.427, 1.52, 2.7, 2.668, 14.978, 19.4118, 11.4868, 3.4275, 5.4601, 7.6375, 2.7405, 11.2635, 18.641, 6.351, 3.363, 4.965, 4.9307, 1.927, 3.5305, 29.5015, 22.362, 27.1694
	Sayur	3.357, 5.392, 0.624, 4.317, 1.619, 1.53, 0.235
	buah	2.6965, 3.7943, 2.8925, 3.188, 13.3803, 0.392, 1.2974, 4.4203, 2.8628, 37.8843, 12.4478, 29.6266, 0.6513, 6.6778, 1.3738, 10.2856, 1.111, 6.8855, 0.9727, 0.639, 0.707, 1.413
SIANG	**Makanan**	**Density**
	Makanan Pokok	0.407, 0.717, 1.054, 1.821, 1.744
	Lauk Nabati	3.635, 5.595, 1.962, 2.235, 1.771
	Lauk Hewani	2.826, 2.846, 1.484, 2.427, 2.427, 1.52, 11.4868, 3.4275, 4.0431, 5.4601, 3.4005, 7.9305, 6.9285, 18.641, 6.351, 3.333, 4.9307, 11.2934, 1.927, 3.5305, 16.0129, 29.5015, 22.362
	Sayur	3.124, 1.873, 5.392, 0.624, 4.317, 1.53, 0.235, 1.926
	Buah	2.6965, 3.7473, 2.8925, 3.188, 13.3803, 0.392, 1.2974, 5.2247, 2.8628, 27.8055, 164.4438, 12.4478, 4.3407, 7.1994, 1.5759, 10.2856, 1.848, 1.111, 0.8004, 1.4701, 0.707
MALAM	**Makanan**	**Density**
	Makanan Pokok	0.645, 0.65, 1.744, 0.457, 1.7051
	Lauk Nabati	2.476, 1.962, 1.378, 2.235, 1.771, 1.827
	Lauk Hewani	4.844, 2.826, 2.846, 0.374, 1.367, 117.137, 14.978, 11.4868, 3.805, 3.4685, 3.4275, 3.4005, 7.9305,

		7.6375, 2.7405, 11.2635, 18.641, 6.351, 4.965, 11.2934, 11.007, 1.927, 29.5015, 41.774, 11.6805, 2.793
	Sayur	3.124, 1.873, 5.392, 1.606, 5.177, 4.317, 1.619, 1.53, 1.926
	Buah	2.6965, 3.7943, 3.7473, 13.3803, 0.392, 1.9102, 2.1682, 2.6087, 1.2974, 2.8628, 27.8055, 164.4438, 37.8843, 12.4478, 65.8993, 4.3407, 3.2568, 7.1994, 1.5759, 1.848, 1.111, 0.8004, 0.639, 0.707

Tabel 3. 4 Density Individu 2

	Makanan	**Density**
PAGI	Makanan Pokok	1.003, 0.717, 1.821, 1.744
	Lauk Nabati	3.635, 5.595, 2.235, 1.771
	Lauk Hewani	2.826, 1.524, 1.524, 2.826, 2.846, 1.484, 0.374, 2.427, 2.427, 117.137, 11.4868, 3.805, 3.4685, 4.0431, 7.9305, 7.6375, 2.7405, 11.2635, 18.641, 3.333, 4.9307, 26.5866, 16.0129, 22.362, 11.6805, 2.793
	Sayur	3.124, 3.357, 5.392, 5.177, 0.624, 4.317, 1.619, 1.926
	buah	2.6965, 3.7473, 0.7004, 13.3803, 5.2247, 4.4203, 2.8628, 27.8055, 164.4438, 37.8843, 1.8946, 3.2568, 6.6778, 1.3738, 6.815, 1.5759, 10.2856, 1.848, 4.2985, 6.8855, 2.191
SIANG	**Makanan**	**Density**
	Makanan Pokok	1.003, 1.821, 0.457, 1.7051
	Lauk Nabati	2.476, 2.235
	Lauk Hewani	2.826, 1.524, 4.844, 2.846, 1.484, 0.374, 2.024, 2.427, 1.367, 4.0431, 3.4005, 2.7405, 6.9285, 3.5305, 16.0129, 29.5015, 41.774, 11.6805, 9.1331, 2.793
	Sayur	1.873, 3.357, 5.392, 1.606, 5.177, 0.235
	Buah	3.7943, 3.7473, 0.7004, 2.8925, 3.188, 0.392, 1.9102,

		2.1682, 2.6087, 1.2974, 4.4203, 27.8055, 37.8843, 1.8946, 0.6513, 3.2568, 6.815, 4.2985, 1.111, 0.8004, 1.4701
MALAM	**Makanan**	**Density**
	Makanan Pokok	1.003, 1.054, 1.744, 0.457, 1.7051
	Lauk Nabati	5.595, 1.378, 2.235, 1.771
	Lauk Hewani	1.484, 0.374, 2.024, 2.427, 2.427, 1.367, 1.52, 2.668, 14.978, 11.4868, 3.805, 3.4005, 6.9285, 11.2635, 3.333, 3.363, 11.2934, 16.0129, 29.5015, 41.774, 11.6805, 27.1694, 9.1331, 2.793
	Sayur	3.357, 5.392, 1.606, 5.177, 0.624, 4.317, 1.619, 1.53, 1.926
	Buah	0.7004, 2.8925, 3.188, 13.3803, 0.392, 1.9102, 1.2974, 2.8628, 3.2578, 27.8055, 164.4438, 37.8843, 29.6266, 65.8993, 4.3407, 3.2568, 7.1994, 6.6778, 10.2856, 1.848, 4.2985, 0.9727, 0.707, 2.191, 1.413

Tabel 3. 5 Density Individu 3

PAGI	**Makanan**	**Density**
	Makanan Pokok	0.97, 1.054, 0.65, 1.7051
	Lauk Nabati	3.635, 2.476, 2.174, 5.595, 1.962, 1.378, 2.235, 1.771
	Lauk Hewani	1.524, 4.844, 2.846, 0.374, 2.024, 2.427, 1.367, 2.668, 14.978, 19.4118, 3.4275, 2.7405, 6.9285, 6.351, 3.363, 4.965, 26.5866, 1.927, 16.0129, 41.774, 11.6805, 27.1694, 2.793
	Sayur	3.124, 1.873, 5.392, 1.606, 4.317, 0.235
	buah	5.4805, 3.7943, 3.7473, 0.7004, 1.9102, 2.1682, 1.2974, 5.2247, 2.8628, 3.2578, 27.8055, 164.4438, 29.6266, 1.8946, 7.1994, 1.3738, 6.815, 1.848, 4.2985, 1.111, 0.707, 2.191, 1.413
	Makanan	**Density**

S I A N G	Makanan Pokok	1.003, 0.407, 0.717, 1.054, 0.645, 1.744, 1.7051
	Lauk Nabati	2.476, 1.962, 1.378, 2.235
	Lauk Hewani	2.826, 1.524, 2.846, 0.374, 2.024, 2.427, 117.137, 14.978, 19.4118, 11.4868, 3.4275, 5.4601, 3.4005, 7.6375, 3.333, 3.363, 4.9307, 26.5866, 11.007, 1.927, 16.0129, 29.5015, 22.362, 11.6805, 27.1694
	Sayur	3.357, 1.606, 0.624, 4.317, 1.53, 0.235
	Buah	5.4805, 3.7473, 2.8925, 13.3803, 1.2974, 4.4203, 2.8628, 3.2578, 27.8055, 29.6266, 1.8946, 0.6513, 7.1994, 6.6778, 1.3738, 6.815, 4.2985, 1.111, 0.8004, 1.4701, 6.8855, 1.413
M A L A M	**MALAM**	**Density**
	Makanan Pokok	1.003, 0.97, 0.645, 0.65, 0.457, 1.7051
	Lauk Nabati	2.476, 2.174, 1.771, 1.827
	Lauk Hewani	1.524, 1.524, 4.844, 1.484, 0.374, 2.024, 2.427, 1.367, 2.7, 117.137, 14.978, 19.4118, 11.4868, 7.9305, 7.6375, 6.9285, 11.2635, 18.641, 6.351, 3.333, 3.363, 4.9307, 26.5866, 11.007, 3.5305, 16.0129, 29.5015, 41.774, 22.362, 27.1694, 2.793
	Sayur	1.873, 5.177, 0.624, 4.317, 1.619, 1.53
	Buah	0.7004, 2.8925, 13.3803, 2.1682, 1.2974, 5.2247, 4.4203, 27.8055, 37.8843, 12.4478, 1.8946, 0.6513, 4.3407, 1.3738, 6.815, 1.5759, 4.2985, 1.111, 6.8855, 0.707, 2.191, 1.413

Setelah diketahui *density* masing-masing makanan, akan dilakukan *sorting* dan dipilih makanan dengan *density* tertinggi, yaitu :

Induk1

1.821, 5.595, 29.5015, 5.392, 37.8843, 1.821, 5.595, 29.5015, 5.392 164.4438, 1.744, 2.476, 117.137, 5.392, 164.4438

Induk2

1.821, 5.595, 117.137, 5.392, 164.4438, 1.821, 2.476, 41.774, 5.392, 37.8843, 1.744, 5.595, 41.774, 5.392, 164.4438

Induk3

1.7051, 5.595, 41.774, 5.392, 164.4438, 1.744, 2.476, 117.137, 4.317, 29.6266, 1.7051, 2.476, 117.137, 5.177, 37.8843

Penentuan *Penalty,* yang dilakukan pertama adalah random bilangan untuk pembobotan, melakukan lima *random* nilai antara (15-kalori max) untuk masing-masing waktu makan, dengan syarat apabila kelima random dijumlahkan tidak lebih dari kalori *max*. Setelah didapatkan lima bilangan *random* maka akan disorting dari besar ke kecil untuk ditempatkan pada masing-masing item yaitu makanan pokok, lauka nabati, hewani, sayur dan buah.

Contoh untuk nasi jagung, tempe bacem, ikan cumi basah goring, sop ayam dan alpukat, dirandom 5 nilai yaitu 18, 30, 20, 21, dan 15. Setelah itu dilakukan penjulahan total

Total = 18+30+20+21+15=104

104 melebihi maksimal total seharusnya yaitu 91. Maka *random* diulang lagi dan misalkan menghasilkan 15, 27, 16, 15 dan 18. Jumlah total bilangan adalah 91. Maka akan dilakukan *sorting* untuk diurutkan sesuai urutan makanan. Hasil dari pembobotan ini adalah

Induk=1

72	19	17	17	16
57	23	23	22	16
55	18	17	16	15

Induk=2

62	23	20	20	16
67	22	18	18	16
45	20	20	20	16

Induk=3

74	19	18	15	15
73	21	16	16	15
44	22	21	19	15

Setelah dilakukan pembobotan, langkah selanjutnya adalah menghitung nilai penalti, perhitungan nilai penalti dapat dilihat pada tabel 3.6. misalkan dihitung nilia penalty dari individu 1. Karbohidrat mempunyai fungsi penalty

$$pj = \frac{tm - tn}{tm} \times 5 + 1$$

contoh perhitungan adalah tn karbohidrat dari nasi jagung, yaitu perkalian bobot (27 g) dengan kandungan makanan yang terpilih :

Karbo = 27 * 0.272 = 7.344
Lemak = 27 * 0.272 = 0.162
Serat = 27 * 0.013 = 0.351
Besi = 27 * 0.004 = 0.108
Fosfor = 27 * 0.6300 = 17.010
Kalsium = 27 * 0.030 = 0.810
Protein = 27 * 0.028 = 0.754
Vitamin A = 27 * 0.050 = 1.350
Vitamin C = 27 * 0.001 = 0.540
Kolesterol = 27 * 0.000 = 0
Penalti makanan sama = 0

Perhitungan diatas dilakukan untuk semua gen terpilih dalam satu individu. Contoh penalti untuk karbohidrat adalah

$P_{karbohidrat}$ = ((182.011 – 38.306)/182.011)*5+1=719.526

\Perhitungan diatas dilakukan untuk semua gen yang ada, hingga didapatkan hasil sebagai berikut

Tabel 3. 6 Pinalti Individu 1

Item Penalti	Total Seharusnya	Total Didapat	Penalti
KARBO	183.36	72.226	4.0305
PROTEIN	45.84	44.26	1.1724
LEMAK	33.956	19.4067	3.1423
SERAT	18.806	5.713	2.0887
Fe	26.000	5.239	2.3955
Fosfor	600.000	729.8099	0.2163
Cal	800.000	105.8859	2.6029
Vit. A	600.000	3706.28	12.8251
Vit. C	75.000	15.2999	2.388
Kolesterol	300.000	0.0	0.0
Makanan sama	0.000	60.0	60.0

Tabel 3. 7 Pinalti Individu 2

Item Pinalty	Total Seharusnya	Total Didapat	Pinalty
KARBO	183.36	67.261	4.1659
PROTEIN	45.84	33.834	2.3096
LEMAK	33.956	20.686	2.954
SERAT	18.806	5.6049	2.1059
Fe	26.000	5.585	2.3556
Fosfor	600.000	816.36	0.3606
Cal	800.000	562.5799	0.8903
Vit. A	600.000	3701.14	12.8046
Vit. C	75.000	15.9799	2.3608
Kolesterol	300.000	0.0	0.0
Makanan sama	0.000	60.0	60.0

Tabel 3. 8 Pinalti Individu 3

Item Pinalty	Total Seharusnya	Total Didapat	Pinalti
KARBO	183.36	72.793	4.0151
PROTEIN	45.84	34.6459	2.2211
LEMAK	33.956	24.6259	2.3738
SERAT	18.806	4.941	2.2118
Fe	26.000	7.306	2.157
Fosfor	600.000	573.54	0.0441
Cal	800.000	326.44	1.7758
Vit. A	600.000	3123.45	10.4938
Vit. C	75.000	7.55	2.698
Kolesterol	300.000	0.0	0.0
Makanan sama	0.000	20.0	20.0

Total pinalti

induk 1 dengan total *penalty* 90.8617

induk 2 dengan total *penalty* 90.3073

induk 3 dengan total *penalty* 47.9905

Setelah mendapatkan hasil pinalti, selanjutnya adalah perhitungan *fitness*. Perhitungan *fitness* digunakan untuk meminimumkan penalti dengan rumus

$$fitness\ (f) = \frac{1}{1 + pinalty}$$

Sehingga didapatkan nilai *fitness*

No.	*Pinalty*	*Fitness*
1.	90.8617	0.0109
2.	90.3037	0.011
3.	47.9905	0.0204

3.3.4 Proses *Repair*

Proses *repair* dilakukan pada individu yang tidak *valid*, individu yang tidak *valid* adalah individu yang memilih lebih dari satu item makanan pada menu makanan yang sama, misal makanan pokok terpilih nasi jagung dan bubur menyebabkan individu menjadi tidak *valid*. Proses *repair* berdasarkan pada perhitungan *density*, dimana gen yang tetap bernilai 1 (terpilih) adalah gen yang mempunyai *density* paling tinggi, dan gen lainnya akan dirubah nilainya menjadi 0 (tidak terpilih).

indukRepair1

00000001000000000000000000000000000000000000000
00010000000000000000000000000000000000000000000
00000000000000000000000000000000000000001000000
00010000000000000000000000000000000000000000000
00000000000000000001000000000000000000000000000
00000001000000000000000000000000000000000000000
00010000000000000000000000000000000000000000000
00000000000000000000000000000000000000001000000
00010000000000000000000000000000000000000000000
00000000000000000010000000000000000000000000000
00000001000000000000000000000000000000000000000
01000000000000000000000000000000000000000000000
00000000000000010000000000000000000000000000000
00010000000000000000000000000000000000000000000
00000000000000000100000000000000000000000000000

indukRepair2

0000000100000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000010000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000100000000000000000000000000000000
0000000100000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000100000000
0001000000000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000
0000000100000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000100000000
0001000000000000000000000000000000000000000000000
0000000000000000100000000000000000000000000000000

indukRepair3

0000000001000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000100000000
0001000000000000000000000000000000000000000000000
0000000000000000100000000000000000000000000000000
0000000100000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000
0000000000000001000000000000000000000000000000000
0000000100000000000000000000000000000000000000000
0000000000000000000100000000000000000000000000000
0000000001000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000
0000000000000010000000000000000000000000000000000
0000010000000000000000000000000000000000000000000
0000000000000000000100000000000000000000000000000

3.3.5 Seleksi

Proses seleksi menggunakan roda roulette adalah dengan menghitung fitness dari masing-masing kromosom untuk selanjutnya dihitung *fitness* komulatif individu dari hasil sorting *fitness* masing-masing kromosom. Setelah itu dilakukan random r, bila fitness komulatif (qk) lebih kecil dari nilai random, maka individu itu yang terpilih. Berikut ini adalah proses seleksi dengan roda roulette.

Pertama yang dilakukan adalah menghitung total fitness dari masing-masing kromosom, kedua adalah menghitung probabilitas dari masing-masing kromosom, ketiga adalah menghitung probabilitas komulatif dan terakhir adalah melakukan random nilai antara (0-1) untuk menentukan kromosom yang diplih. Berikut ini merupakan contoh dari seleksi *roulette wheel* dengan fitness dari masing-masing kromosom-kromosom sebagai berikut :

fitness repair1 0.0109

fitness repair2 0.011

fitness repair3 0.0204

Setelah menghitung nilai *fitness* masing-masing, maka akan dilaku-kan perhitungan total nilai *fitness*, yaitu

F = 0.0109 + 0.011 + 0.0204 = 0.0423

Dilakukan probabilitas seleksi berdasarkan equation 2.14

$$pi = \frac{fi}{F}$$

probabilitas repair1 0.2576832151300236

probabilitas repair2 0.260047281323877

probabilitas repair3 0.48226950354609927

Langkah selanjutnya adalah menghitung komulatif (qk)

Qk_1 = 0.2576832151300236

Qk_2 = 0.5177304964539007

Qk_3 = 1.0

Setelah didapatkan qk, maka akan dibangkitkan bilangan *random* antara (0-1), misal muncul 0.5 maka kromosom yang terpilih adalah individu anak 1.2 karena 0.510 > 0.5. Proses ini dilakukan sebanyak individu yang ada, selanjutnya diulang ke proses *crossover*, mutasi, penghitungan nilai *fitness*, *repair* dan seleksi sampai sejumlah iterasi yang ditentukan.

Elitishm yang digunakan adalah individu yang terbaik bersasarkan nilai fitnessnya.

parent 1 = 0.0204 (dari fitness terbaik)

random 0.04407200498856001

parent 2 = 0.0109 ProbKom 0.2576832151300236

random 0.753800046509224

parent 3 = 0.0204 ProbKom 1.0

3.3.6 Crossover

Proses *crossover* dilakukan apabila nilai *random* (0-1) lebih kecil dari probabilitas crossover. Misal probabilitas *crossover* adalah 50% dan nilai *random* adalah 0.3, maka proses *crossover* akan dilakukan. Setelah itu dilakukan nilai *random* 2 bilangan antara (0-14) untuk menentukan titik *crossover*. Misal didapatkan bilangan random 3 dan 9. Maka gen anak akan ditukarkan pada titik tersebut. Pada iterasi pertama, individu yang terpilih sama degan fitness 0.0204 yaitu dari individu repair3. Maka tidak terdapat perbedaan parent dengan child.

Crossover2

000000000010000000000000000000000000000000000000
000100000000000000000000000000000000000000000000
000000000000000000000000000000000000000100000000
000100000000000000000000000000000000000000000000
000000000000000001000000000000000000000000000000
000000010000000000000000000000000000000000000000
000100000000000000000000000000000000000000000000
000000000000000000000000000000000000001000000000
000100000000000000000000000000000000000000000000
000000000000000001000000000000000000000000000000
000000010000000000000000000000000000000000000000
010000000000000000000000000000000000000000000000
000000000000001000000000000000000000000000000000
000001000000000000000000000000000000000000000000
000000000000000001000000000000000000000000000000

Crossover3

00000001000000000000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000000000000000000000000000000000001000000000
00010000000000000000000000000000000000000000000000
00000000000000000001000000000000000000000000000000
00000001000000000000000000000000000000000000000000
01000000000000000000000000000000000000000000000000
00000000000000010000000000000000000000000000000000
00000001000000000000000000000000000000000000000000
00000000000000000001000000000000000000000000000000
00000000010000000000000000000000000000000000000000
01000000000000000000000000000000000000000000000000
00000000000000010000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000000000000001000000000000000000000000000000

3.3.7 Mutasi

Proses mutasi tergantung pada suatu parameter yang disebut probabilitas mutasi (p_m). Misalkan individu yang diharapkan mengalami mutasi adalah 25%, apabila hasil *random* r (0-1) kurang dari 0.25 maka akan dilakukan mutasi. Misalkan hasil random adalah 0.1, karena 0.1 < 0.23 maka mutasi akan dilakukan.

Titik mutasi akan dirandom dari (0-n_{-1}) sebanyak 2 nilai random, dan misalkan didapatkan 0 dan 1 maka hasil mutasi adalah sebagai berikut :

Mutasi2

00000110001000000000000000000000000000000000000000
10010001000000000000000000000000000000000000000000
00000000000000000000000000000000000010000000100010
00000001000000000000000000000000000000000000000000
00000000000000000010000000000001000100000000000000
00000001000000000000000000000000000000000000000000
00110010000000000000000000000000000000000000000000
00000000000000000000000000001000000000100010000000
00010101000000000000000000000000000000000000000000
00000100000000000010000000000000000010000000000000
01100000100000000000000000000000000000000000000000
01010001000000000000000000000000000000000000000000
00000000000000000000010000000000000000000000000000
00001000000000000000000000000000000000000000000000
10000010000000000010000000000000000000000000000000

Mutasi3

000001110000000000000000000000000000000000000000
011100000000000000000000000000000000000000000000
000000000000000000000000000000000000001001100000
000101000100000000000000000000000000000000000000
000000010000000000010000000000000000010000000
100000011000000000000000000000000000000000000000
011001000000000000000000000000000000000000000000
000000000001000100000000000000000000010000000
001000010010000000000000000000000000000000000000
000100000000000000001000000000000010000000000000
001000000110000000000000000000000000000000000000
000000100000000000000000000000000000000000000000
000000000000000100000000000000000000100001000000
000100001010000000000000000000000000000000000000
000000000000000000100000000000100000100000000000

- PERHITUNGAN DENSITY

a. Mutasi1

1.821, 5.595, 41.774, 5.392, 37.8843, 1.821, 2.476, 117.137, 4.317, 29.6266, 1.7051, 2.235, 117.137, 5.392, 164.4438.

b. Mutasi2

1.7051, 5.595, 41.774, 4.317, 164.4438, 1.821, 5.595, 29.5015, 5.392, 164.4438, 1.744, 5.595, 4.0431, 1.606,.37.8843

c. Mutasi3

1.821, 5.595, 41.774, 5.392, 37.8843, 1.821, 2.476, 117.137, 4.317, 29.6266, 1.7051, 2.235, 117.137, 5.392, 164.4438.

- PERHITUNGAN FITNESS

a. Pembobotan

Induk1

64	22	19	18	18
69	21	17	17	17
53	20	17	16	15

Induk2

63 23 20 18 17

64 22 19 18 18

49 19 19 19 15

Induk3

73 19 17 16 16

60 22 20 20 19

46 21 20 19 15

b. Penalty dan fitness

No.	Pinalti	Fitness
1.	52.8397	0.0186
2.	71.2699	0.0138
3.	51.0191	0.0192

REPAIR

Repair2

0000000000100000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000100000
0000000100000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000
0000000100000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000000000000000000000000001000000
0001000000000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000
0000000010000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000000001000000000000000000000000000
0000100000000000000000000000000000000000000000000
0000000000000000000010000000000000000000000000000

Repair3

00000001000000000000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000100000000
00010000000000000000000000000000000000000000000000
00000000000000000010000000000000000000000000000000
00000001000000000000000000000000000000000000000000
01000000000000000000000000000000000000000000000000
00000000000000010000000000000000000000000000000000
00000001000000000000000000000000000000000000000000
00000000000000000000100000000000000000000000000000
00000000010000000000000000000000000000000000000000
00000010000000000000000000000000000000000000000000
00000000000000010000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000000000000010000000000000000000000000000000

SELEKSI

- Probabilitas

  probabilitas 0.3604651162790698

  probabilitas 0.2674418604651163

  probabilitas 0.37209302325581395

- Probabilitas Komulatif

  Probablitas Kumulatif 0.3604651162790698

  Probabilitas Kumulatif 0.627906976744186

  Probabilitas Kumulatif 1.0

- Seleksi

  parent 1 = 0.0192

  random 0.9298856666945869

  parent 2 = 0.0192 ProbKom 1.0

  random 0.5785385968814717

  parent 3 = 0.0138 ProbKom 0.627906976744186

ITERASI KEDUA

A. CROSSOVER

Induk2

0000000100000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000100000
0001000000000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000
0000000100000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000
0000000000000001000000000000000000000000000000000
0000000100000000000000000000000000000000000000000
0000000000000000000010000000000000000000000000000
0000000001000000000000000000000000000000000000000
0000001000000000000000000000000000000000000000000
0000000000000001000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000100000000000000000000000000000000

Induk3

0000000100000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000001000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000010000000000000000000000000000000
0000000100000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000100000
0001000000000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000
0000000100000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000100000
0001000000000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000

B. MUTASI

Induk2

0100000000000000000000000000000000000000000000000000
0011000100000000000000000000000000000000000000000000
0000000000000000000000000100001000000000000100000
0001000010100000000000000000000000000000000000000000
0000000000000000010100000000100000000000000000000000
0000101100000000000000000000000000000000000000000000
0101000100000000000000000000000000000000000000000000
0000000000000001010000000010000000000000000000000000
1000000100100000000000000000000000000000000000000000
0000000000000000010000000000000000000000000000000000
0000001010100000000000000000000000000000000000000000
0001101000000000000000000000000000000000000000000000
0000000001000001000000000000000000000000001000000
0011001000000000000000000000000000000000000000000000
0000011000000000010000000000000000000000000000000000

Induk3

0100001100000000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000000
0000000001000010000010000000000000000000000000000000
0101100000000000000000000000000000000000000000000000
0000000000010000001010000000000000000000000000000000
0000101100000000000000000000000000000000000000000000
0111000000000000000000000000000000000000000000000000
0000000010000000000000000000000000000000001100000
0001010001000000000000000000000000000000000000000000
1000000100000000001000000000000000000000000000000000
0000101010000000000000000000000000000000000000000000
0000000100000000000000000000000000000000000000000000
0000000000000001000000000000000000100000000100000
0001010100000000000000000000000000000000000000000000
0000000000000001001000000000000000000000001000000

C. DENSITY

Induk1 (fitness terbaik)

0.407, 5.595, 41.774, 5.392, 37.8843, 1.821, 5.595, 117.137, 4.317, 164.4438, 1.744, 5.595, 117.137, 5.392, 164.4438

Induk2

0.407, 5.595, 41.774, 5.392, 37.8843, 1.821, 5.595, 117.137, 4.317, 164.4438, 1.744, 5.595, 117.137, 5.392, 164.4438

Induk3

1.821, 5.595, 117.137, 5.392, 164.4438, 1.821, 5.595, 41.774, 5.392, 37.8843, 1.744, 1.771, 117.137, 5.392, 164.4438

- Bobot Individu

Induk=1

64	22	19	18	18
69	21	17	17	17
53	20	17	16	15

Induk=2

63	23	20	18	17
64	22	19	18	18
49	19	19	19	15

Induk=3

73	19	17	16	16
60	22	20	20	19
46	21	20	19	15

- Penalty

No.	Pinalti	Fitness
1.	90.0537	0.011
2.	89.8662	0.011
3.	90.0095	0.011

D. REPAIR

Repair2

0100000000000000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000100000
0001000000000000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000000
0000001000000000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000000
0000000000000001000000000000000000000000000000000000
0000001000000000000000000000000000000000000000000000
0000000000000000010000000000000000000000000000000000
0000001000000000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000000
0000000000000001000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000000
0000000000000000010000000000000000000000000000000000

Repair3

0000001000000000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000000
0000000000000001000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000000
0000000000000000010000000000000000000000000000000000
0000001000000000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000100000
0001000000000000000000000000000000000000000000000000
0000000000000000010000000000000000000000000000000000
0000001000000000000000000000000000000000000000000000
0000001000000000000000000000000000000000000000000000
0000000000000001000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000000
0000000000000000010000000000000000000000000000000000

E. SELEKSI HASIL MUTASI

a. Menghitung probabilitas

Probabilitas induk1, induk2 dan induk3 adalah 0.3333333333333333

b. Menghitung probabilitas komulatif

Probablitas Kumulatif 0.3333333333333333

Probabilitas Kumulatif 0.6666666666666666

Probabilitas Kumulatif 1.0

c. Seleksi

parent 1 = 0.011

random 0.15145425695608117

parent 2 = 0.011 ProbKom 0.3333333333333333

random 0.7954511853057381

parent 3 = 0.011 ProbKom 1.0

ITERASI KETIGA

A. CROSSOVER

Induk2

0100000000000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000100000
0001000000000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000
0000001000000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000001000000000000000000000000000000000
0000001000000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000
0000000100000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000001000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000

Induk3

0100000000000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000100000
0001000000000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000
0000001000000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000001000000000000000000000000000000000
0000001000000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000
0000000100000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000001000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000000000000001000000000000000000000000000000

B. MUTASI

Induk2

0010011000000000000000000000000000000000000000000
0001000000000000000000000000000000000000000000000
0000000010000000000000000010000000000000001100100
1001001000000000000000000000000000000000000000000
0000001000000000000010000000010100000000010000000
1100001110000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000
0000000000000001000000000000000010000000000001000
1000000000000000000000000000000000000000000000000
0001000000000000001100010000000000000010000000000
0110010000000000000000000000000000000000000000000
0101111000000000000000000000000000000000000000000
0100000000000001000000000000000000010000000000000
1000000110000000000000000000000000000000000000000
0100010000000000001000000000000000010000001000000

Induk3

00100110000000000000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000100000000000000000100000000000000001100100
10010010000000000000000000000000000000000000000000
00000010000000000001000000000101000000000010000000
11000011100000000000000000000000000000000000000000
01000000000000000000000000000000000000000000000000
00000000000000010000000000000000010000000000001000
10000000000000000000000000000000000000000000000000
00010000000000000011000100000000000000010000000000
01100100000000000000000000000000000000000000000000
01011110000000000000000000000000000000000000000000
01000000000000010000000000000000000010000000000000
10000000110000000000000000000000000000000000000000
01000100000000000010000000000000000010000001000000

C. DENSITY

Induk1

0.97, 5.595, 41.774, 5.392, 37.8843, 1.821, 2.476, 117.137, 3.124, 164.4438,

0.97, 5.595, 117.137, 3.124, 164.4438

Induk2

0.97, 5.595, 41.774, 5.392, 37.8843, 1.821, 2.476, 117.137, 3.124, 164.4438,

0.97, 5.595, 117.137, 3.124, 164.4438

Induk3

0.97, 5.595, 41.774, 5.392, 37.8843, 1.821, 2.476, 117.137, 3.124, 164.4438,

0.97, 5.595, 117.137, 3.124, 164.4438

D. FITNESS

- Pembobotan

Induk=3

64	22	19	18	18
69	21	17	17	17
53	20	17	16	15

Induk=3

63	23	20	18	17
64	22	19	18	18
49	19	19	19	15

Induk=3

73	19	17	16	16
60	22	20	20	19
46	21	20	19	15

- Penalty

No.	Penalti	Fitness
1.	72.4958	0.0136
2.	72.1308	0.0137
3.	70.6041	0.014

E. REPAIR

Induk2

00100000000000000000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000100000
00010000000000000000000000000000000000000000000000
00000000000000000001000000000000000000000000000000
00000010000000000000000000000000000000000000000000
01000000000000000000000000000000000000000000000000
00000000000000001000000000000000000000000000000000
10000000000000000000000000000000000000000000000000
00000000000000001000000000000000000000000000000000
00100000000000000000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000000000010000000000000000000000000000000000
10000000000000000000000000000000000000000000000000
00000000000000001000000000000000000000000000000000

Induk3

00100000000000000000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000100000000
00010000000000000000000000000000000000000000000000
00000000000000000010000000000000000000000000000000
00000010000000000000000000000000000000000000000000
01000000000000000000000000000000000000000000000000
00000000000000010000000000000000000000000000000000
10000000000000000000000000000000000000000000000000
00000000000000000010000000000000000000000000000000
00100000000000000000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000000000010000000000000000000000000000000000
10000000000000000000000000000000000000000000000000
00000000000000000010000000000000000000000000000000

F. SELEKSI

- Probabilitas

probabilitas 0.32929782082324455

probabilitas 0.33171912832929784

probabilitas 0.33898305084745767

- Probabilitas komulatif

Probablitas Kumulatif 0.32929782082324455

Probabilitas Kumulatif 0.6610169491525424

Probabilitas Kumulatif 1.0

- Seleksi

parent 1 = 0.014

random 0.25674890565673436

parent 2 = 0.0136 ProbKom 0.32929782082324455

random 0.5339181203224114

parent 3 = 0.0137 ProbKom 0.6610169491525424

G. HASIL AKHIR

Fitness terbaik didapatkan pada iterasi ke-1

00000001000000000000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000000000010000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000000000000100000000000000000000000000000000
00000001000000000000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000100000000
00010000000000000000000000000000000000000000000000
00000000000000000010000000000000000000000000000000
00000001000000000000000000000000000000000000000000
00000001000000000000000000000000000000000000000000
00000000000000010000000000000000000000000000000000
00010000000000000000000000000000000000000000000000
00000000000000000010000000000000000000000000000000

dengan bobot

73	19	17	16	16
60	22	20	20	19
46	21	20	19	15

MENU

Pagi

roti tawar, sayur tempe, hati ayam, sayur bening, Mangga gadung

Siang

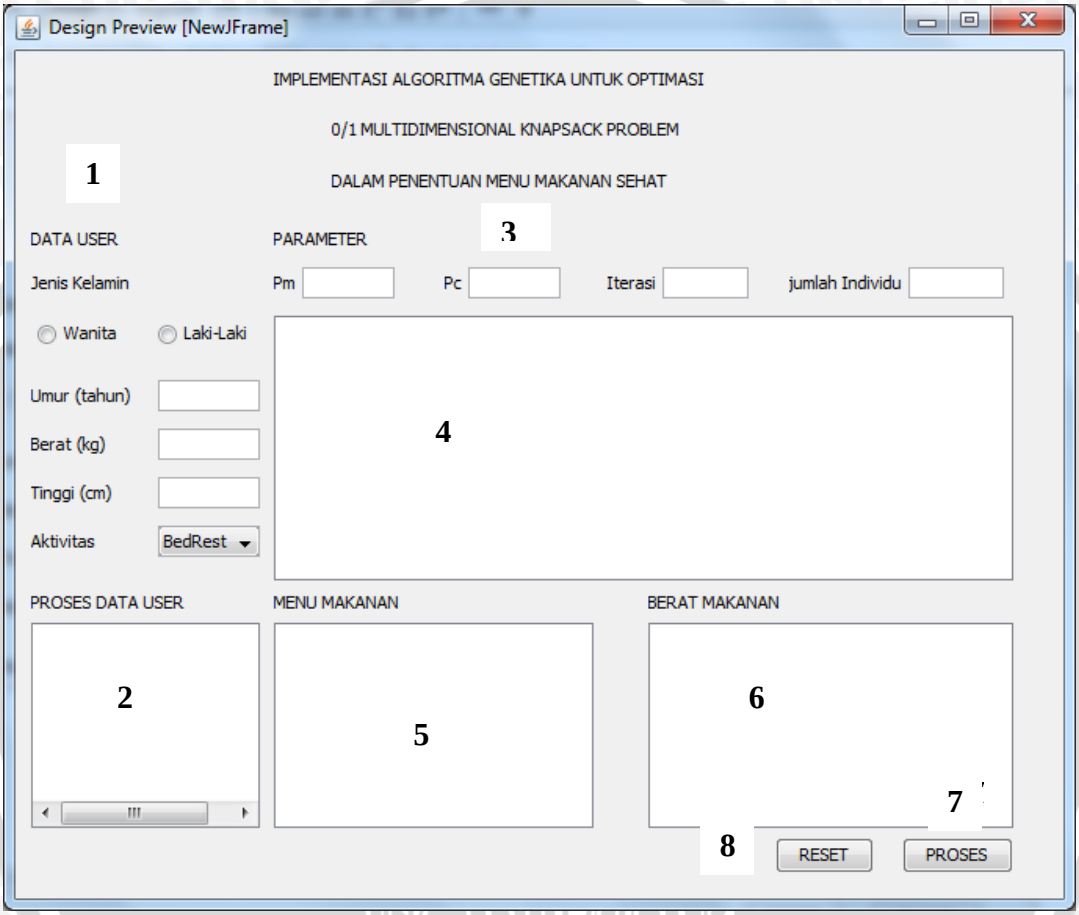
roti tawar, sayur tempe, Teri kering sekali (tawar), sayur bening, Mangga golek

Malam

roti tawar manis, sayur tahu, hati ayam, sayur bening, Mangga gadung

3.4 Rancangan Antar Muka

Rancangan antar muka (*interface*) digunakan untuk merancang interaksi antara user dan system untuk mempermudah user dalam menggunakan aplikasi menu makanan.



Keterangan :

1. Data yang harus dimasukkan oleh user untuk digunakan pada proses selanjutnya, yaitu jenis kelamin, umur, berat, tinggi, aktivitas dan data makanan alergi user.
2. Menampilkan hasil dari proses data user.
3. Parameter genetika yang perlu diinputkan.
4. Menampilkan hasil dari algortima genetika.
5. Menu makanan digunakan untuk menampilkan hasil terakhir dari menu makanan dalam satu hari.

6. Menampilkan berat dari makanan dalam satu hari.
7. Button proses digunakan untuk menjalankan proses algortima dalam menyelesaikan perencanaan menu makanan.
8. Button Reset digunakan untuk melakukan reset program.

3.5 Pengujian Tingkat Akurasi

Uji coba system untuk perencanaan menu makanan sehat ini akan melakukan evaluasi terhadap hasil fitness yang diperoleh. Pada pengujian menggunakan algoritma genetika akan diberikan nilai parameter yang berbeda-beda, yaitu probabilitas crossover, probabilitas mutasi, jumlah generasi yang dibangkitkan dan jumlah iterasi.

Tujuan dari uji coba ini adalah :

1. Untuk mengetahui hubungan probabilitas crossover dengan hasil yang didapatkan.
2. Untuk mengetahui hubungan probabilitas mutasi dengan hasil yang didapatkan.
3. Untuk mengetahui hubungan banyaknya jumlah generasi yang dibangkitkan dengan hasil yang didapat.

3.5.1 Skenario Uji Coba

Percobaan-percobaan yang dilakukan menggunakan parameter sebagai berikut:

1. Probabilitas crossover
2. Probabilitas mutasi
3. Jumlah individu baru yang dibangkitkan
4. Jumlah iterasi

Probabilitas crossover dan mutasi yang akan diuji adalah 10%, 25%, 50%, 70% dan 90%. Masing-masing dilakukan pada populasi yang memiliki individu yang dibangkitkan 5 individu dengan generasi sebanyak 100 generasi. Serta bahan yang digunakan 125 bahan makanan. Setiap pengujian dilakukan 5 kali untuk mengetahui

perubahan dari nilai fitness masing-masing kromosom hingga didapatkan hasil yang optimal dari parameter-parameter diatas.

Setelah diketahui Pm dan Pc terbaik, maka akan dilakukan pengujian untuk menentukan generasi dan individu awal yang dibangkitkan. Generasi yang diuji coba adalah 25, 50 dan 100 generasi. Sedangkan individu awal yang diuji coba adalah 3, 5 dan 7 individu awal.

Tabel 3. 9 Tabel Percobaan Probabilitas Crossover, Probabilitas Mutasi dengan 100 Generasi dan 5 Individu.

banyak generasi	**Individu Dibangkitkan**	**Probabilitas Crossover**	**Probabilitas Mutasi**				
			10	25	50	70	90
100	5	10					
		25					
		50					
		70					
		90					

Tabel 3. 10 Tabel Pengujian Jumlah Generasi dan Individu Awal

Generasi/Individu	Fitness 25 generasi	Fitness 50 generasi	Fitness 100 generasi
3 individu			
5 individu			
7 individu			

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada bab implementasi dan pembahasan, akan dibahas mengenai penerapan dan pembahasan sistem yang terlah dikembangkan serta evaluasinya. Proses yang dilakukan oleh sistem Penentuan Menu Makanan Sehat secara umum adalah melakukan pengolahan data yang dimasukkan *user*, pengambilan data dari Excel yang akan digunakan untuk proses penentuan panjang indeks menu makanan yang selanjutnya digunakan untuk membentuk kromosom individu, setelah terbentuk individu akan dilakukan perhitungan *fitness* sehingga dapat dilakukan *repair* pada gen individu yang tidak *valid*. Selanjutnya dilakukan proses genetika, yaitu seleksi untuk menentukan *parent* yang akan di *crossover* dan dimutasikan hingga terbentuk hasil menu berdasarkan metode *knapsack* dan algoritma genetika.

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan pada subbab ini meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam implementasi sistem berupa notebook dengan spesifikasi sebagai berikut :

1. Processor Intel® Core™ Duo CPU T5750 @2.00GHz
2. RAM 2048MB

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan untuk mengembangkan aplikasi Penentuan Menu Makanan Sehat adalah :

1. Sistem Operasi Microsoft Windows 7 Ultimate 32-bit (6.1, Build 7600) sebagai lingkungan kerja dari aplikasi ini.

2. Netbeans 6.8 sebagai perangkat lunak yang digunakan untuk mengembangkan aplikasi menggunakan bahasa pemrogramann Java serta J-Excel API untuk menyambungkan data pada Excel ke Java.
3. Microsoft Office Excel sebagai perangkat lunak yang digunakan sebagai tabel tempat penyimpanan menu makanan dan kandunan nutrisinya.
4. Java Development Kit 6.18

4.2 Implementasi Program

Berdasarkan analisis dan perancangan yang diuraikan pada bab 3, subbab ini akan menjelaskan implementasi perancangan yang telah dilakukan. Implementasi dalam bentuk program terbagi dalam kelas-kelas sebagai berikut :

Tabel 4. 1 Kelas-kelas yang digunakan

No	Kelas	Keterangan
1.	`ProsesDataUser` (implemetasi 4.2.1)	Implementasi perhitungan energi berdasarkan metode *Harris Bennedict*
2.	`BacaExcel` (impelemntasi 4.2.2)	Membaca data dari excel.
3.	`GenerateKromosomRandom` (implementasi 4.2.3)	Membentuk individu dan populasi
4.	`Density` (implementasi 4.2.4)	Impelementasi algoritma *Fractional Knapsack*
5.	`PenaltyMakananFitness` (implementasi 4.2.5)	Perhitungan *Penalty* dan *Fitness* pada masing-masing individu
6.	`Repair` (impelentasi 4.2.6)	Impelementasi pemilihan gen individu valid berdasarkan hasil proses *Density*
7.	`SeleksiRouletteWheel` (impelentasi 4.2.7)	Implementasi algoritma seleksi *Roulette Wheel*
8.	`CrossoverTwoPoint` (implementasi 4.2.8)	Implementasi algoritma *Crossover Two Point*

9.	`MutasiBiner` (impelentasi 4.2.9)	Implementasi algoritma Mutasi Biner

4.1.3 Implementasi Kelas ProsesDataUser

Kelas ProsesData*User* berfungsi untuk melakukan pengolahan data yang diinputkan user. Kelas ini digunakan untuk mendapatkan energi yang dibutuhkan user dalam satu hari berdasarkan umur, berat badan, jenis kelamin, berat badan, dan aktivitas. Selain itu digunakan juga untuk mengetahui bobot makanan berdasarkan Makronutrien (Karbo, Protein dan Lemak) yang dibutuhkan tubuh. Kode program kelas ProsesDataUser ditampilkan pada kode program 4.1

```
public double getEnergi(double jenisKelamin, double height, double
weight, double age, String inputAktifitas) {
        double energi = 0.0;
        if (jenisKelamin == 0) {
            energi = dataUserWanita(height, weight, age,
            inputAktifitas);
        } else {
            energi = dataUserPria(height, weight, age,
            inputAktifitas);
        }
        return energi;
    }
    public static double getAktivitas(String aktivitas) {
        double activit = 0.0;
        if (aktivitas.equals("bedrest")) {
            activit = 1.2;
        } else if (aktivitas.equals("ringan")) {
            activit = 1.3;
        } else if (aktivitas.equals("sedang")) {
            activit = 1.4;
        } else {
            activit = 1.5;
        }
        return activit;
```

```
}
public static double dataUserWanita(double height,
double weight, double age, String inputAktifitas) {
    double BEE = 655.1 + (9.66 * weight) + (1.85 *
    height) - (4.68 * age);
    double activity = getAktivitas(inputAktifitas);
    double energiTotal = BEE * activity;
    return energiTotal;
}
public static double dataUserPria(double height, double
weight, double age, String inputAktifitas) {
    double BEE = 66.5 + (13.5 * weight) + (5 * height) -
    (6.78 * age);
    double activity = getAktivitas(inputAktifitas);
    double energiTotal = BEE * activity;
    return energiTotal;
}
public static double getEnergiMaxMakronutrienKarbo(double
getEnergi) {
    double karbo = 0.6 * getEnergi;
    return karbo;
}
public static double getEnergiMaxMakronutrienProtein(double
getEnergi) {
    double protein = 0.15 * getEnergi;
    return protein;
}
public static double getEnergiMaxMakronutrienLemak(double
getEnergi) {
    double lemak = 0.25 * getEnergi;
    return lemak;
}
```

Kode Program 4. 1 Perhitungan energi user

4.2.1 Implementasi Kelas BacaExcel

Kelas BacaExcel digunakan untuk tabel data makanan yang digunakan untuk proses *Generate* kromosom, *Density*, *Penalty*, *Fitness*, *Repair* dan Seleksi. Kode program kelas BacaExcel ditampilkan pada kode program 4.2

```
public class BacaExcel {
    public static String namaExcel = "DataNutrisiFix19062012.xls";
    public static File file;
    public double[][] getSheet(int indeksMakan) throws
    IOException, BiffException {
        double[][] angka = null;
        if (indeksMakan == 0 || indeksMakan == 5 || indeksMakan ==
        10) {
            angka = getDataAngka(0);
        } else if (indeksMakan == 1 || indeksMakan == 6 ||
        indeksMakan == 11) {
            angka = getDataAngka(1);
        } else if (indeksMakan == 2 || indeksMakan == 7 ||
        indeksMakan == 12) {
            angka = getDataAngka(2);
        } else if (indeksMakan == 3 || indeksMakan == 8 ||
        indeksMakan == 13) {
            angka = getDataAngka(3);
        } else if (indeksMakan == 4 || indeksMakan == 9 ||
        indeksMakan == 14) {
            angka = getDataAngka(4);
        } else {
            angka = getDataAngka(5);
        }
        return angka;
    }
    public static String[][] readFile(int sheet) throws
    IOException, BiffException {
        file = new File(namaExcel);
        Workbook workbook = Workbook.getWorkbook(file);
```

```
        Sheet sheet_ = workbook.getSheet(sheet);
        int x = sheet_.getRows();
        int y = sheet_.getColumns();
        String[][] data = new String[x][y];
        for (int i = 0; i < x; ++i) {
            for (int j = 0; j < y; ++j) {
                String getdata = sheet_.getCell(j,
                i).getContents();
                data[i][j] = getdata;
            }
        }
        return data;
    }
    public double[][] getDataAngka(int sheet) throws IOException,
    BiffException {
        String[][] data = readFile(sheet);
        double[][] dataAngka = new double[data.length -
        1][data[0].length - 1];
        for (int i = 1; i < dataAngka.length; i++) {
            for (int j = 1; j < dataAngka[i].length; j++) {
                dataAngka[i][j] = Double.parseDouble
                (data[i][j].replaceAll(",", "."));
            }
        }
        return dataAngka;
    }
```

Kode Program 4. 2 Pengambilan data pada excel

4.2.2 Implementasi Kelas GenerateKromosoRandom

Kelas GenerateKromosomRandom merupakan kelas yang digunakan untuk membangkitkan kromosom berdasarkan panjang menu makanan yang terdapat pada Excel berbentuk array 2 dimensi, dengan gen yang berisi bilangan biner (0/1) merepresentasikan metode Knapsack dimana 0/1 merepresentasikan take it or leave it. Selain itu juga digunakan untuk menampung banyak individu dalam satu populasi

untuk digunakan pada proses selanjutnya. Kode program kelas GenerateKromosomRandom ditampilkan pada kode program 4.3.

```
public class GenerateKromosmRandom {
    public static int populasi;
    static Random random = new Random();
    public static int getMax(int[] panjang) {
        int max = 0;
        for (int i = 0; i < panjang.length; i++) {
            if (panjang[i] > max) {
                max = panjang[i];
            }
        }
        return max;
    }
    public static void setPopulasi(int jumlahPopulasi) {
        populasi = jumlahPopulasi;
    }
    public static int[][] getArray(int gMax, int[]
    panjangMsg2Menu) {
        int[][] masuk = new int[15][gMax];
        for (int i = 0; i < masuk.length; i++) {
            if (i == 0 || i == 5 || i == 10) {
                for (int j = 0; j < panjangMsg2Menu[0]; j++) {
                    masuk[i][j] = random.nextInt(2);
                }
            }
            if (i == 1 || i == 6 || i == 11) {
                for (int j = 0; j < panjangMsg2Menu[1]; j++) {
                    masuk[i][j] = random.nextInt(2);
                }
            }
            if (i == 2 || i == 7 || i == 12) {
                for (int j = 0; j < panjangMsg2Menu[2]; j++) {
                    masuk[i][j] = random.nextInt(2);
                }
            }
```

```
            if (i == 3 || i == 8 || i == 13) {
                for (int j = 0; j < panjangMsg2Menu[3]; j++) {
                    masuk[i][j] = random.nextInt(2);
                }
            }
            if (i == 4 || i == 9 || i == 14) {
                for (int j = 0; j < panjangMsg2Menu[4]; j++) {
                    masuk[i][j] = random.nextInt(2);
                }
            }
        }
        return masuk;
    }
    public ArrayList<int[][]> populasi(int max, int[] panjang) {
      ArrayList<int[][]> listArray=new ArrayList<int[][]>();
        for (int i = 0; i < populasi; i++) {
            int[][] getArray = getArray(max, panjang);
            listArray.add(getArray);
        }
        return listArray;
    }
}
```

Kode Program 4. 3 Generate kromosom random

4.2.3 Implementasi Kelas Density

Kelas Density digunakan untuk menentukan makanan apa yang akan diambil menjadi menu, density berdasarkan metode *fractional knapsack problem* ini akan menghasilkan nilai yang akan diambil dari nilai terbesar dalam satu baris. Kode program kelas Density ditampilkan pada kode program 4.4.

```
public class Density {
    public static BacaExcel baca = new BacaExcel();
    public static GenerateKromosmRandom generate = new
    GenerateKromosmRandom();
    static Random random = new Random();
```

```
public static ProsesDataUser user = new ProsesDataUser();
public double[] getTotalKandungan(int sheet) throws
IOException, BiffException {
    double[][] dataAngka = baca.getDataAngka(sheet);
    double[] jumlahTotal = new double[dataAngka.length];
    for (int i = 1; i < dataAngka.length; i++) {
        for (int j = 2; j < dataAngka[i].length - 1; j++) {
            jumlahTotal[i] += dataAngka[i][j];
        }
    }
    return jumlahTotal;
}
public ArrayList<ArrayList<Double>>
getListTotal(ArrayList<ArrayList<Integer>> getIndeksAll)
throws IOException, BiffException {
    ArrayList<ArrayList<Double>> listTotal = new
    ArrayList<ArrayList<Double>>();
    ArrayList<Double> listDoubleTotal;
    for (int i = 0; i < getIndeksAll.size(); i++) {
        int indekBarisSheet =
        generate.getKonversiWaktuMakan(i);
        double[] jumlahKandungan =
        getTotalKandungan(indekBarisSheet);
        ArrayList<Integer> listIndeksKolom =
        getIndeksAll.get(i);
        listDoubleTotal = new ArrayList<Double>();
        for (int j = 0; j < listIndeksKolom.size(); j++) {
            for (int k = 1; k < jumlahKandungan.length;
            k++) {
                if (listIndeksKolom.get(j) == (k - 1)) {
                    listDoubleTotal.add(jumlahKandungan[k]);
                }
            }
        }listTotal.add(listDoubleTotal);
    }return listTotal;
}
```

Kode Program 4. 4 Perhitungan density Fractional Knapsack

4.2.4 Implementasi Kelas PenaltyMakananFitness

Kelas PenaltyMakananFitness digunakan untuk menghitung *penalty* dan *fitness* dari individu dalam populasi. Kelas PenaltyMakananFitness mempunyai beberapa fungsi yang ditunjukkan pada tabel 4.2 Fungsi Kelas Penalti *Fitness*.

Tabel 4. 2 Fungsi Kelas Penalti Fitness

No	Method	Fungsi
1.	Double [] penPro	Digunakan untuk menghitung penalty dari Protein
2.	Double [] PenLem	Digunakan untuk menghitung penalty dari Lemak
3.	Double [] penKar	Digunakan untuk menghitung penalty dari Karbohidrat
4.	Double [] penKal	Digunakan untuk menghitung penalty dari Kalsium
5.	Double [] penFos	Digunakan untuk menghitung penalty dari Fosfor
6.	Double [] penBes	Digunakan untuk menghitung penalty dari Besi
7.	Double [] penVitA	Digunakan untuk menghitung penalty dari Vitamin A
8.	Double [] penVitC	Digunakan untuk menghitung penalty dari Vitamin C
9.	Double [] penSer	Digunakan untuk menghitung penalty dari Serat
10.	Double [] penKol	Digunakan untuk menghitung penalty dari kolesterol
11.	Double [] makanSama	Digunakan untuk menghitung penalty dari makanan yang sama
12.	Double [] totalPenalty	Digunakan untuk menghitung total dari semua penalty
13.	Double [] fitness	Digunakan untuk menghitung fitness individu

Proses perhitungan protein dihitung berdasarkan kandungan protein dari makanan yang terpilih dibandingkan dengan kandungan protein yang seharusnya. Kode penpro ditunjukkan pada kode program 4.5

```
public static double[] penPro(double[] bobotProtein, double
getBobotProtein) {
        double bs = getBobotProtein;
        double[] pen = new double[bobotProtein.length];
        for (int i = 0; i < bobotProtein.length; i++) {
            pen[i] = penaltyMakronutrien(bs, bobotProtein[i]);
        }
        return pen;
    }
}
```

Kode Program 4. 5 Perhitunga penalty protein

Setelah dilakukan perhitungan protein, dilakukan proses perhitungan *penalty* lemak. Prosesnya sama dengan protein, dengan membandingkan kandungan lemak didapat dan lemak seharusnya. Kode program ditunjukkan pada kode program 4.6.

```
public static double[] penLem(double[] bobotLemak, double
getBobotLemak) {
        double bs = getBobotLemak;
        double[] pen = new double[bobotLemak.length];
        for (int i = 0; i < bobotLemak.length; i++) {
            pen[i] = penaltyMakronutrien(bs, bobotLemak[i]);
        }
        return pen;
    }
```

Kode Program 4. 6 Perhitungan penalty lemak

Proses selanjutnya adalah proses perhitungan penalty karbohidrat, proses yang dilakukan sama dengan proses sebelumnya. Kode program ditunjukkan pada kode program 4.7.

```
public static double[] penKar(double[] bobotKarbo, double
getBobotKarbo) {
    double bs = getBobotKarbo;
    double[] pen = new double[bobotKarbo.length];
```

```
    for (int i = 0; i < bobotKarbo.length; i++) {
        pen[i] = penaltyMakronutrien(bs,
        bobotKarbo[i]);
    }
    return pen;
}
```

Kode Program 4. 7 Perhitungan penalty karbohidrat

Proses perhitungan penalty selanjutnya adalah proses perhitungan *penalty* Fosfor, proses yang dilakukan sama yaitu dengan membandingkan kandungannya. Kode program ditunjukkan pada kode program 4.8.

```
public static double[] penKar(double[] bobotKarbo, double
getBobotKarbo) {
    double bs = getBobotKarbo;
    double[] pen = new double[bobotKarbo.length];
    for (int i = 0; i < bobotKarbo.length; i++) {
        pen[i] = penaltyMakronutrien(bs, bobotKarbo[i]);
    }return pen;
}
```

Kode Program 4. 8 Perhitungan penalty kalisum

Proses perhitungan fosfor dilakukan setelah perhitungan *penalty* kalsium, cara kerja sama dengan perhitungan penalty lainnya. Kode program ditunjukkan pada kode program 4.9.

```
public static double[] penFos(double[] bobotFosfor, double age) {
        double bs = botFosSeharusnya(age);
        double[] pen = new double[bobotFosfor.length];
        for (int i = 0; i < bobotFosfor.length; i++) {
            double sel = selisih(bs, bobotFosfor[i]);
            pen[i] = sel;
        }
        return pen;
    }
```

Kode Program 4. 9 Perhitungan penalty fosfor

Setelah perhitungan penalti fosfor, dilakukan perhitungan *penalty* besi dimana cara kerja prosesnya sama dengan perhitungan *penalty* sebelumnya. Kode program ditunjukkan pada kode program 4.10.

```
public static double[] penFos(double[] bobotFosfor, double age) {
     double bs = botFosSeharusnya(age);
     double[] pen = new double[bobotFosfor.length];
     for (int i = 0; i < bobotFosfor.length; i++) {
         double sel = selisih(bs, bobotFosfor[i]);
         pen[i] = sel;
     }
     return pen;
}
```

Kode Program 4. 10 Perhitungan penalty besi

Selanjutnya adalah proses perhitungan *penalty* vitamin A dilakukan dengan membandingkan berat kandungan yang didapatkan dnegan dengan berat seharunya. Kode program ditunjukkan pada kode program 4.11.

```
public double[] penVitA(double[] bobotVitA, double age, double
jenisKelamin) {
    double[] pen = new double[bobotVitA.length];
    double bs = botVitASeharusnya(age,jenisKelamin);
    for (int i = 0; i < bobotVitA.length; i++) {
        double sel = selisih(bs, bobotVitA[i]);
        if (bs > bobotVitA[i]) {
            pen[i] = sel * 3;
        } else if (bs == bobotVitA[i]) {
            pen[i] = sel + 1;
        } else {
            pen[i] = sel * 2;
        }
    }return pen;
}
```

Kode Program 4. 11 Perhitungan penalti vitamin A

Setelah dilakukan proses perhitungan *penalty* vitamin A, dilakukan juga proses perhitungan penalti vitamin C ditunjukkan pada kode program 4.12.

```
public double [] penVitC(double [] bobotVitC, double jenisKelamin){
    double [] pen = new double[bobotVitC.length];
    double bs = botVitCSeharusnya(jenisKelamin);
    for (int i = 0; i < bobotVitC.length; i++) {
        double sel = selisih(bs, bobotVitC[i]);
        if(bobotVitC[i]<bs){
            pen[i]=sel*3;
        }else{
             pen[i]=0;
        }
    }
    return pen;
}
```

Kode Program 4. 12 Perhitungan penalty vitamin C

Setelah dilakukan proses perhitungan *penalty* Vitamin C, selanjutnya dilakukan proses perhitungan penalti serat. Proses yang dilakukan sama, yaitu dengan mengambil selisih dari kandungan yang didapatkan dengan seharusnya. Kode program ditunjukkan pada kode program 4.13.

```
  public double [] penSer(double [] bobotSerat, double getEnergi){
      double [] pen = new double[bobotSerat.length];
      double bs = botSerSeharusnya(getEnergi);
      for (int i = 0; i < bobotSerat.length; i++) {
          double sel = selisih(bs, bobotSerat[i]);
          if(bobotSerat[i]<bs){
              pen[i]=sel*3;
          }else{
              pen[i]=0;
          }
      }return pen;
  }
```

Kode Program 4. 13 Perhitungan penalty serat

Selanjutnya dilakukan perhitungan *penalty* kolesterol. Kode program *penalty* kolesterol ditunjukkan pada kode program 4.14.

```
public double [] penKol(double [] bobotKol){
    double [] pen = new double[bobotKol.length];
    double bs = 300;
    for (int i = 0; i < bobotKol.length; i++) {
        double sel = selisih(bs, bobotKol[i]);
        if(bobotKol[i]>bs){
            pen[i]=sel*5;
        }else{
            pen[i]=0;
        }
    }
    return pen;
}
```

Kode Program 4. 14 Perhitungan penalty kolesterol

Terakhir adalah perhitungan *penalty* terhadap makan yang sama dalam satu individu ditunjukkan pada kode program 4.15.

```
public static double[] makanSama(ArrayList<int[]> listIndeksPop)
{
    double[] sama = new double[listIndeksPop.size()];
    for (int i = 0; i < listIndeksPop.size(); i++) {
        int a = 0;
        int count = 0;
        if (pop[1] == pop[6]) {
            count += 1;
        }
        if (pop[1] == pop[11]) {
            count += 1;
        }
        if (pop[6] == pop[11]) {
            count += 1;
```

```
        }
        if (pop[2] == pop[7]) {
            count += 1;
        }
        if (pop[2] == pop[12]) {
            count += 1;
        }
        if (pop[7] == pop[12]) {
            count += 1;
        }
        if (pop[3] == pop[8]) {
            count += 1;
        }
        if (pop[3] == pop[13]) {
            count += 1;
        }
        if (pop[8] == pop[13]) {
            count += 1;
        }
        if (pop[4] == pop[9]) {
            count += 1;
        }
        if (pop[4] == pop[14]) {
            count += 1;
        }
        if (pop[9] == pop[14]) {
            count += 1;
        }
    }
    return sama;
}
```

Kode Program 4. 15 Perhitungan penalty makanan yang sama

Setelah didapatkan semua perhitungan *penalty*, maka selanjutnya dilakukan perhitungan total *penalty* yang merupakan penjumlahan dari semua *penalty* dalam satu individu, ditunjukkan pada kode program 4.16.

```
public static double [] totalPenalty(double[] penPro, double[]
penLem, double[] penKar, double[] penKal, double[] penFos,
double[] penBes, double[] penVitA, double[] penVitC, double[]
penSer, double[] penKol, double [] makanSama){
    double [] totPen = new double[penBes.length];
    for (int i = 0; i < totPen.length; i++) {
        totPen[i]=penKar[i]+penPro[i]+penLem[i]+penSer[i]+
        penBes[i]+penFos[i]+penKal[i]+penVitA[i]+penVitC[i]+
        penKol[i]+makanSama[i];
    }
    return totPen;
}
```

Kode Program 4. 16 Perhitungan penalty total

Setelah didapatkan *penalty* total, selanjutnya dilakukan perhitungan *fitness* yang didapatkan dari hasil total *penalty* tiap individu, fungsi fitness ditunjukkan pada *equation* 2.1. Kode program ditunjukkan pada kode program 4.17.

```
public static double [] fitness(double  [] totalPenalty){
    double [] fitnes = new double[totalPenalty.length];
    for (int i = 0; i < fitnes.length; i++) {
        fitnes[i]=1/(1+totalPenalty[i]);
    }
    return fitnes;
}
```

Kode Program 4. 17 Perhitungan fitness

4.2.5 Implementasi Kelas *Repair*

Kelas *repair* digunakan untuk memilih satu individu yang *valid* berdasarkan *density* (makanan yang terpilih). Kode program kelas *repair* dapat dilihat pada kode program 4.18.

```
ArrayList<int [][]> repair1(ArrayList<int[][]> populasi,
ArrayList<int[]> listIndeksKe2){
    ArrayList<int [][]> perbaikan1 = new ArrayList<int
    [][]>();
    for (int i = 0; i < populasi.size(); i++) {
        int [][] popo = populasi.get(i);
        int [] indeks = listIndeksKe2.get(i);
        for (int j = 0; j < popo.length; j++) {
            for (int k = 0; k < popo[j].length; k++) {
                if(k != indeks[j]){
                    popo[j][k] = 0;
                }else{
                    popo[j][k] = 1;
                }
            }
        }
        perbaikan1.add(popo);
    }
    return perbaikan1;
}
```

Kode Program 4. 18 Kelas Repair

4.2.6 Implementasi Kelas Seleksi*RouletteWheel*

Kelas Seleksi *Roulette Wheel* merupakan kelas impelementasi algoritma *Roulette*, kode program kelas seleksi dapat dilihat pada kode program 4.19.

```
  Double [] Roulette (double [] fitness){
      Random rand = new Random();
      double total=0;
      double sumProb=0;
      double [] prob = new double[fitness.length];
      double [] probKom = new double[fitness.length];
      double [] getMin = new double[fitness.length];
      double [] ParentNew = new double[fitness.length];
      double [] indeksFit = new double[fitness.length];
```

```
        double [] Seleksi = new double[fitness.length];
        for (int i = 0; i < fitness.length; i++) {
            System.out.println("fitness "+fitness[i]);
            total += fitness[i];
        }
        for (int i = 0; i < fitness.length; i++) {
            prob[i] = fitness[i]/total;
        }
        double [] ProbKumulatif = new double[fitness.length];
        ProbKumulatif[0] = prob[0];
        for(int i=1;i<fitness.length;i++){
            sumProb+=prob[i-1];
            ProbKumulatif[i]+=prob[i]+sumProb;
        }
        ParentNew[0] = getMax2(fitness);
        for (int i = 1; i < fitness.length; i++) {
            double ran = rand.nextDouble();
            for (int j = 0; j < ProbKumulatif.length; j++) {
                if (ran < ProbKumulatif[j]){
                    probKom[i]=ProbKumulatif[j];
                    ParentNew[i]=fitness[j];
                    break;
                }
            }
        }
        for (int i = 0; i < ParentNew.length; i++) {
            for (int j = 0; j < fitness.length; j++) {
                if (ParentNew[i] ==  fitness[j]){
                    Seleksi[i]=j;
                }
            }
        }
        return Seleksi;
    }
```

Kode Program 4. 19 Kelas SeleksiRouletteWheel

4.2.7 Implementasi Kelas CrossoverTwoPoint

Kelas CrossoverTwoPoint merupakan implementasi dari algoritma crossover dua titik, dimana terdapat dua titik *cutpoint* sebagai titik pertukaran. Kode program crossover dapat dilihat pada kode program 4.20.

```
ArrayList<int[][]> Crossover(double[] seleksi, ArrayList<int[][]>
repair1) {
    ArrayList<int[][]> cros = new ArrayList<int[][]>();
    Random rand = new Random();
    int parent1 = 0, parent2 = 0;
    int cutpoint1 = rand.nextInt(15);
    int cutpoint2 = rand.nextInt(15);
    while (cutpoint1 == 0) {
        cutpoint1 = rand.nextInt(15);
    }
    while (cutpoint1 == cutpoint2) {
        cutpoint2 = rand.nextInt(15);
    }
    while (cutpoint1 > cutpoint2) {
        int temp = cutpoint2;
        cutpoint2 = cutpoint1;
        cutpoint1 = temp;
    }
    double randpc = rand.nextDouble();
    int bagi = seleksi.length / 2;
    for (int b = 1; b < bagi + 1; b++) {
        parent1 = b * 2 - 1;
        parent2 = b * 2;
        int[][] child1 = repair1.get((int) seleksi[parent1]);
        int[][] child2 = repair1.get((int) seleksi[parent2]);
        if (0.1 < pc) {
            for (int j = 0; j < cutpoint1; j++) {
                for (int k = 0; k < child1[j].length; k++) {
                    int temp = child2[j][k];
                    child2[j][k] = child1[j][k];
```

```
                    child1[j][k] = temp;
                }
            }
            for (int j = cutpoint2; j < child1.length; j++) {
                for (int k = 0; k < child1[j].length; k++) {
                    int temp = child2[j][k];
                    child2[j][k] = child1[j][k];
                    child1[j][k] = temp;
                }
            }
        }
        cros.add(child1);
        cros.add(child2);
    }
    return cros;
}
```

Kode Program 4. 20 CrossoverTwoPoint

4.2.8 Implementasi Kelas MutasiBiner

Kelas mutasiBiner digunakan untuk melakukan mutasi pada individu dengan random pm (probabilitas mutasi) lebih kecil dari pm yang ditentukan. Perubahan yang dilakukan pada dua titik yang dipilih secara random, dengan mengubah gen dari 0 menjadi 1, atau sebaliknya dari 1 menjadi 0. Kode program MutasiBiner dapat dilihat pada kode program 4.21.

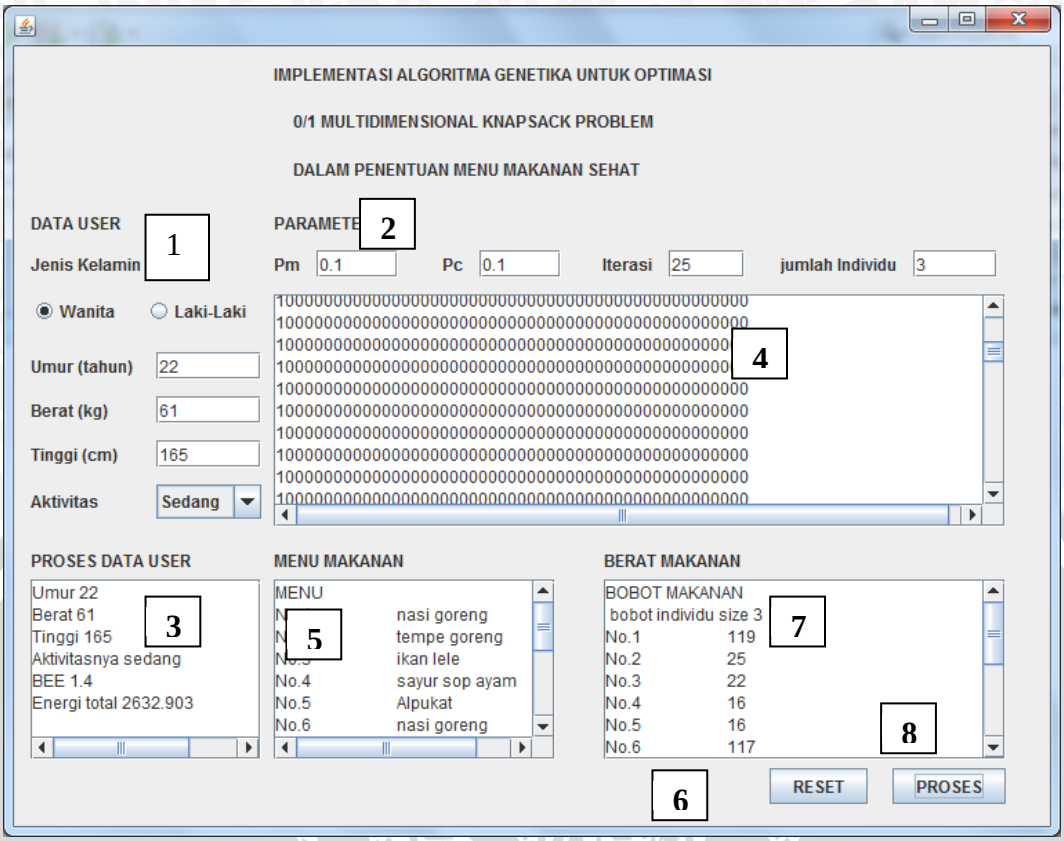
```
  ArrayList<int[][]> mutasiBiner(ArrayList<int[][]> crossover, int[]
panjang) {
      Random rand = new Random();
      ArrayList<int[][]> mutation = new ArrayList<int[][]>();
      int[] pjg = panjangIn(panjang);
      for (int i = 0; i < crossover.size(); i++) {
          double randPm = rand.nextDouble();
          if (0.1 < pm) {
              int[][] mutasi = crossover.get(i);
              for (int j = 0; j < mutasi.length; j++) {
```

```
                int randomIndeks1 = rand.nextInt(pjg[j] - 1);
                int randomIndeks2 = rand.nextInt(pjg[j] - 1);
                for (int k = 0; k < mutasi[j].length; k++) {
                    if (k == randomIndeks1) {
                        if (mutasi[j][k] == 0) {
                            mutasi[j][k] = 1;
                        } else {
                            mutasi[j][k] = 0;
                        }
                    }
                    if (k == randomIndeks2) {
                        if (mutasi[j][k] == 0) {
                            mutasi[j][k] = 1;
                        } else {
                            mutasi[j][k] = 0;
                        }
                    }
                }
            }
            mutation.add(mutasi);
        }
    }
    return mutation;
}
```

Kode Program 4. 21 Kelas MutasiBiner

4.3 Implementasi Antar Muka

Implementasi antar muka aplikasi dari sistem penentuan menu makanan adalah sebagai berikut :



Keterangan :

1. Data User digunakan untuk input data user yang dibutuhkan, yaitu jenis kelamin, umur, berat, tinggi dan aktivitas.
2. Parameter digunakan untuk input data parameter algoritma genetika, yaitu Pm, Pc, Iterasi dan Jumlah Individu.
3. TextArea yang digunakan untuk menampilkan hasil dari pengolahan data user.
4. TextArea yang digunakan untk menampilkan hasil akhir dari kromosom yang terpilih, fitness, dan didapatkan pada iterasi keberapa fitness terbaik itu didapatkan.
5. TextArea yang digunakan untuk menampilkan menu makanan yang terpilih berdasarkan hasil algoritma genetika.
6. TextArea yang digunakan untuk menampilkan bobot dari makanan yang terpilih berdasarkan hasil algoritma genetika.

7. Tombol Reset digunakan untuk melakukan proses ulang.
8. Tombol proses digunakan untuk melakukan proses dalam sistem.

4.4 Implementasi Uji Coba

Pada sub bab ini akan dibahas mengenai implementasi dari metode pengujian yang telah dilakukan oleh system dan hasil dari pengujian tersebut.

4.4.1 Skenario Uji Coba

Skenario evaluasi sistem perancangan menu makanan sehat ini menggunakan parameter Pm, Pc, Iterasi, dan Jumlah Individu. Pm dan Pc dengan prosentase 10%, 25%, 50%, 70%, 90%. Iterasi sebanyak 25, 50 dan 100 generasi dengan individu sebanyak 3, 5 dan 7 individu.

Pengujian pertama dilakukan dengan menggunakan 5 individu awal, 100 generasi dan probabilitas *crossover* dengan prosentase 10%, 25%, 50%, 70%, 90%, probabilitas mutasi dengan prosentase 10%, 25%, 50%, 70%, 90%. Pengujian pertama digunakan untuk mengetahui Pm dan Pc tertinggi yang dilihat dari nilai *fitness* yang didapatkan. Pengujian dilakukan sebanyak 5 kali, dan akan diambil nilai rata-rata untuk dianalisis.

Pengujian kedua dilakukan dengan menggunakan 3 individu awal, yakni 3,5 dan 7. Generasi awal sebanyak 3, yaitu 25, 50 dan 100 generasi. Dengan Pm dan Pc dari hasil pengujian pertama. Pengujian kedua digunakan untuk mendapatkan generasi dan individu awal terbaik berdasarkan nilai rata-rata dari 5 pengujian dan diambil nilai rata-rata terbaik berdasarkan nilai fitnessnya.

4.4.2 Hasil Uji Coba

Uji coba dilakukan berdasarkan parameter yang ditentukan sebelumnya. Tabel pengujian pertama adalah pengujian dengan banyak generasi 100. Individu yang dibangkitkan adalah 5 dengan masing-masing diuji dengan probabilitas mutasi (pm) dan probabilitas *crossover* (pc). Pm dan Pc yang digunakan adalah sebesar 10%, 25%, 50%, 70% dan 90%. Pengujian dilakukan sebanyak 5 kali pengujian, dan

diambil rata-rata dari kelima pengujian tersebut. Hasil pengujian dapat dilihat pada tabel 4.1. dan hasil pengujian individu awal dan generasi dapat dilihat pada tabel 4.2.

Tabel 4.1. Tabel Hasil Percobaan Probabilitas Crossover, Probabilitas Mutasi dengan 100 Generasi dan 5 Individu.

banyak generasi	Individu Dibangkitkan	Probabilitas Crossover	Probabilitas Mutasi					Rata-rata
			10	25	50	70	90	
100	5	10	0.03626	0.03202	0.04224	0.04036	0.03986	0.038148
		25	0.03288	0.0366	0.03872	0.0387	0.03834	0.037048
		50	0.03184	0.03138	0.0423	0.03922	0.03952	0.036852
		70	0.02882	0.03206	0.0443	0.03322	0.0392	0.03552
		90	0.02916	0.03826	0.04118	0.0361	0.04168	0.037276
Rata-rata			0.031792	0.034064	0.041748	0.03752	0.03972	

Tabel 4.2. Tabel Hasil Pengujian Jumlah Generasi dan Individu Awal

Generasi/ Individu	Fitness 25 generasi	Fitness 50 generasi	Fitness 100 generasi	Rata-rata
3 individu	0.035	0.03793333	0.04046667	0.0378
5 individu	0.039433	0.03846667	0.04363333	0.040511
7 individu	0.040167	0.04293333	0.04493333	0.042678
Rata-rata	0.0382	0.03977778	0.04301111	

4.4.3 Analisa Hasil

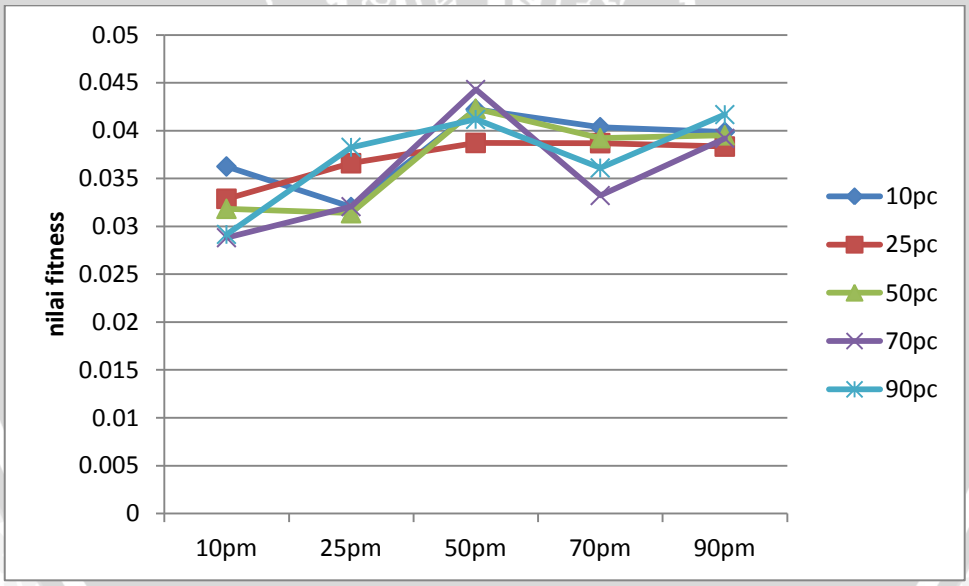
Pada penelitian ini, akan diuji pengaruh parameter algoritma genetika untuk mendapatkan fitness yang maksimal. Pengujian dilakukan dalam perencanaan menu makanan ini dilakukan dengan parameter Pm, Pc, banyak iterasi dan individu awal.

Probabilitas *Crossover* digunakan untuk mengetahui seberapa sering proses *crossover* akan terjadi diantara dua kromosom *parent* jika tidak terjadi *crossover*, maka *child* merupakan salinan dari kromosom *parent*. Sebaliknya, jika terjadi *crossover*, maka child merupakan hasil dari penyilangan kedua kromosom *parent*.

Probabilitas Mutasi digunakan untuk mengetahui seberapa sering proses mutasi akan terjadi pada kromosom *parent*. Kromosom *parent* langsung didapatkan dari *child* hasil *crossover* tanpa ada proses tambahan. jika tidak terjadi mutasi, maka *child* merupakan salinan dari kromosom *parent*. Sebaliknya, jika terjadi mutasi, maka isi gen *child* akan berubah dari 1 menjadi 0, dan sebaliknya dari 0 menjadi 1.

Jumlah individu menyatakan banyaknya individu yang akan dibentuk dalam suatu populasi. Sedangkan jumlah generasi menyatakan banyaknya iterasi yang dilakukan algoritma genetika dalam melakukan proses genetikanya.

Masing-masing hasil pengujian diambil dari 5 kali pengujian dan diambil nilai rata-rata dari kelima hasil pengujian tersebut. Pada tabel 4.1 telah dilakukan pengujian dengan generasi sebanyak 100, individu awal 5, Pm dan Pc 10%, 25%, 50%, 70% dan 90%. Hasil dari pengujian pertama dibentuk menjadi grafik untuk dianalisa, grafik percobaan pertama dapat dilihat pada Gambar 4.1.



Gambar 4. 1 Uji coba pengaruh Pm dan Pc

Berdasarkan gambar 4.1. dapat dilihat adanya kenaikan nilai *fitness* pada saat Pm 50% dan Pc 70%, Pm 50% dan Pc 50%, dan nilai *fitness* mengalami penurunan pada saat Pm dinaikkan menjadi 70% dan mengalami kenaikan lagi pada saat Pm dinaikkan menjadi 90%. Hal ini membuktikan bahwa kenaikan probabilitas mutasi

tidak menjamin mendapatkan nilai fitness yang lebih tinggi. Hal ini dikarenakan semakin tinggi Pm akan menyebabkan lebih banyak gen dan individu mengalami mutasi gen *random* (acak) yang menyebabkan *fitness* individu mengalami penurunan dikarenakan gen dalam individu tergantikan oleh gen *random* yang terlalu banyak.

Dari keterangan diatas, dapat dilihat bahwa semakin besar probabilitas *crossover*, maka nilai *fitness* akan semakin baik juga semakin buruk tergantung Pm yang dibangkitkan. Perolehan nilai *fitness* terbaik terdapat pada Pc 70 dan mengalami penurunan ketika Pc dinaikkan menjadi 90. Hal ini membuktikan bahwa kenaikan probabilitas *crossover* tidak selalu memberikan nilai *fitness* yang yang tinggi dikarenakan *crossover twocutpoint* menentukan titik potongnya dengan cara *random*.

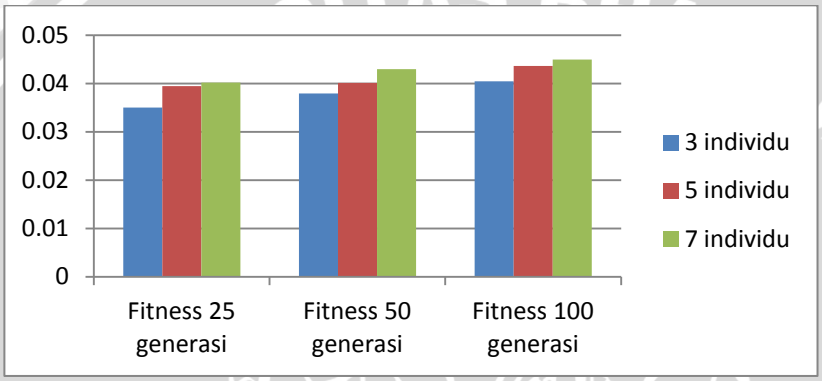
Pada pengujian yang telah dilakukan dapat diketahui bahwa grafik yang dihasilkan mengalami kenaikan dan penurunan dari setiap nilai Pm yang berbeda. Hal tersebut dikarenakan individu yang terbentuk mengalami mutasi dan *crossover* yang dilakukan secara *random*. Individu yang mengalami *crossover* ditukarkan pada dua titik potong yang didapatkan secara *random* sehingga menyebabkan nilai *fitness* menjadi lebih baik, namun ada kalanya pada kondisi tertentu nilai *fitness* menjadi lebih buruk ketika nilai *crossover* terlalu tinggi yang diakibatkan terlalu banyak gen yang ditukarkan. Sedangkan mutasi dilakukan dengan mengganti gen yang kurang valid dimana titik mutasi dibangkitkan secara *random*. Apabila probabilitas mutasi terlalu tinggi, maka semakin banyak gen *random* yang terbentuk sehingga pada kondisi tertentu nilai *fitness*.

Berdasarkan uraian diatas, dapat disimpulkan bahwa metode algoritma untuk optimasi *multi-dimensional knapsack problem* (MDKP) memperoleh hasil nilai *fitness* yang tidak menentu. Nilai fitness dipengaruhi oleh nilai parameter Pm dan Pc yang digunakan, dimana akan didapatkan nilai *fitness* yang baik atau buruk berdasarkan nilai Pm dan Pc yang dikombinasikan.

Berdasarkan penjelasan diatas, Pengujian terhadap nilai *fitness* dari setiap generasi menunjukkan peningkatan nilai *fitness* dari generasi sebelumnya. Hal ini dikarenakan seleksi yang digunakan dalam pembentukan populasi baru mengambil individu dari populasi generasi sebelumnya ditambah dengan anak dari hasi

crossover. Anak hasil mutasi kemudian diseleksi menggunakan seleksi roda roulette. Kemudian diambil individu yang akan diproses pada generasi berikutnya sejumlah individu yang telah ditentukan pada inputan awal jumlah generasi.

Setelah didapatkan probabilitas crossover dan probabilitas mutasi, selanjutnya adalah menganalisa generasi dan individu awal yang ditampilkan pada gambar 4.2. pengujian berdasarkan jumlah generasi dan jumlah individu awal ini menggunakan probabilitas mutasi dan probabilitas crossover Pm=50% dan Pc=70%. Hasil uji coba dapat dilihat pada tabel 4.2.



Gambar 4. 2 Hasil pengujian generasi dan individu awal

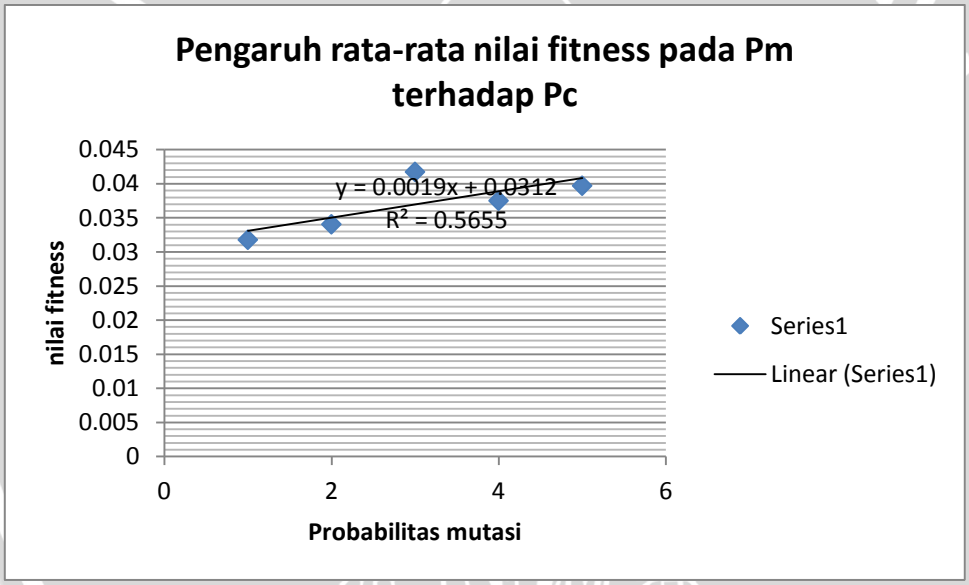
Pada gambar 4.2. dapat dilihat kenaikan nilai *fitness* pada individu dan generasi yang berbeda-beda. Pada individu 5 kenaikan pada generasi 25, 50 dan 100 generasi dapat terlihat walaupun tidak terlalu banyak.

Pengujian terhadap nilai *fitness* dari setiap generasi menunjukkan peningkat seiring dengan bertambahnya generasi yang digunakan. Hal ini dikarenakan dikarenakan seleksi yang digunakan akan menyimpan nilai *fitness* terbaik dari setiap generasi yang ada. Begitu juga untuk individu 5 dan 7. Seiring bertambahnya generasi, *fitness* yang didapatkan semakin baik. Peningkatan dari ketiga generasi tidak terlalu memberikan dampak yang besar, akan tetapi dari gambar tersebut dapat disimpulkan bahwa semakin banyak generasi semakin tinggi nilai *fitness* yang didapatkan.

Berdasarkan uji coba yang telah dilakukan, dapat diambil kesimpulan bahwa semakin besar nilai individu awal dan generasi yang dibangkitkan, semakin baik pula nilai *fitness* yang didapatkan.

4.4.3.1 Pengaruh rata-rata nilai fitness dari Probabilitas Mutasi

Pada tabel 4.1. tabel pengujian Pm dan Pc menghasilkan nilai *fitness* yang dapat diketahui nilai rata-rata dari Pm tertentu. Nilai rata-rata *fitness* yang didapatkan dari penjumlahan nilai *fitness* pada Pm yang sama dijumlahkan dengan nilai *finess* pada Pc yang berbeda kemudian dibagi dengan jumlah banyaknya Pc. Grafik *fitness* dari pengaruh rata-rata dengan nilai *fitness* dapat dilihat pada gambar 4.3.



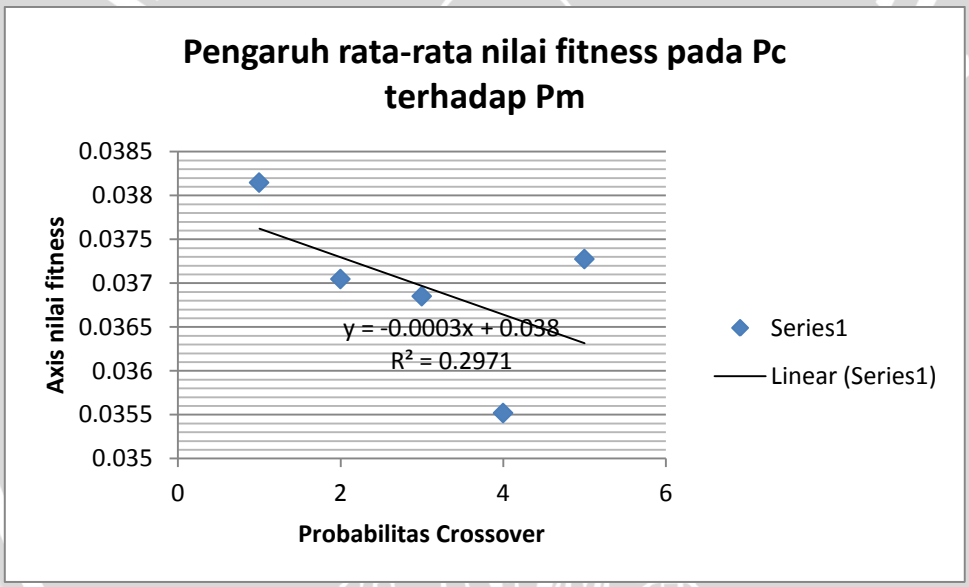
Gambar 4. 3 Gambar grafik rata-rata nilai fitness dari uji coba pada tiap Pm

Pada gambar 4.3. dapat dilihat titik-titik dimana Pm yang telah dilakukan uji coba dan diambil nilai rata-rata dari setiap nilai *fitness* yang didapatkan. Pada grafik tersebut digunakan grafik regresi yang berguna untuk mengukur pengaruh nilai rata-rata *fitness* terhadap Pm yang didapatkan. Adapun persamaan regresi yang didapatkan adalah y=0.0019x+0.0312, dimana y menyatakan rata-rata *fitness* yang didapatkan dan x menyatakan probabilitas mutasi. Pada grafik diatas dapat diketahui grafik linier

dengan nilai Pm 25%, 50%, 70% dan 90% mengalami kenaikan seiring dengan bertambahnya nilai Pm yang dimasukkan, sehingga semakin besar nilai Pm maka semakin tinggi nilai rata-rata *fitness* yang didapatkan.

4.4.3.2 Pengaruh rata-rata nilai fitness dari Probabilitas Crossover

Pada tabel 4.1. tabel pengujian Pm dan Pc menghasilkan nilai *fitness* yang dapat diketahui nilai rata-rata dari Pc tertentu. Nilai rata-rata *fitness* yang didapatkan dari penjumlahan nilai *fitness* pada Pc yang sama dijumlahkan dengan nilai *finess* pada Pm yang berbeda kemudian dibagi dengan jumlah banyaknya Pm. Grafik *fitness* dari pengaruh rata-rata dengan nilai *fitness* dapat dilihat pada gambar 4.4.



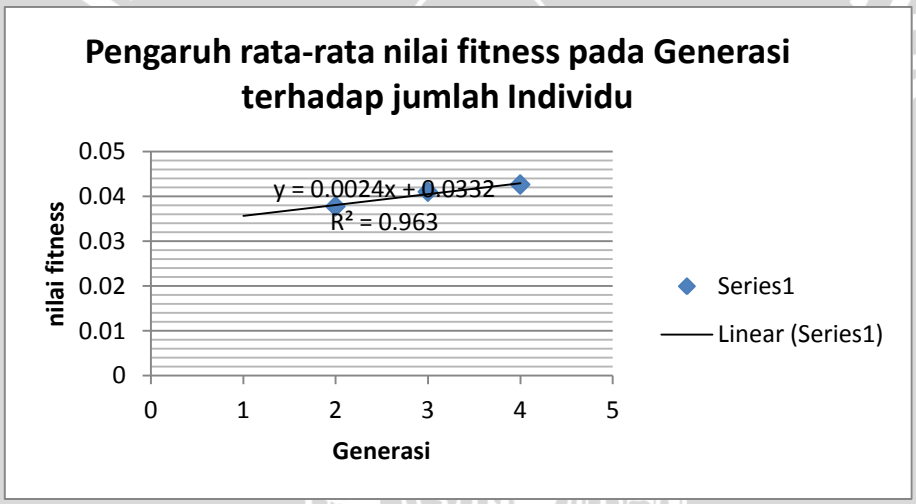
Gambar 4. 4 Gambar grafik rata-rata nilai fitness dari uji coba pada tiap Pc

pada gambar 4.4. dapat diketahui nilai rata-rata probabilitas *crossover*, pada grafik tersebut digunakan grafik regresi yang digunakan untuk mengukur pengaruh nilai rata-rata *fitness* yang didapatkan. Adapaun persamaan regresi yang didapatkan adalah y=0.0003x+0.038, dimana y menyatakan nilai rata-rata *fitness* terhadap Pc. Pada grafik tersebut dapat diketahui grafik linier dengan nilai nilai Pc 25%, 50%,

70% dan 90% mengalami penurunan, oleh sebab itu semakin tinggi nilai Pc maka semakin rendah nilai *fitness* yang didapatkan.

4.4.3.3 Pengaruh rata-rata nilai fitness dari uji coba tiap generasi

pada tabel 4.2. tabel pengujian nilai fitness berdasarkan individu awal dan generasi yang berbeda, diperoleh nilai rata-rata dari nilai fitness yang dihasilkan. Nilai rata-rata fitness yang didaptkan adalah menjumlahkan nilai fitness yang diperoleh dari setiap pengujian pada jumlah generasi yang sama dengan menambahkan dari pengujian jumlah individu yang berbeda. Adapun grafik pengaruh rata-rata dapat dilihat pada gambar 4.5.

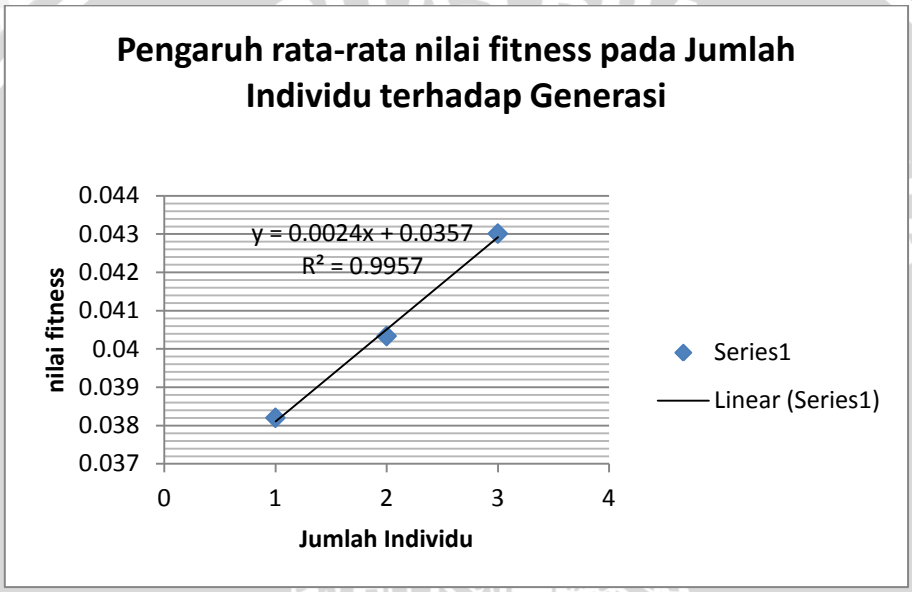


Gambar 4. 5 Grafik pengaruh rata-rata nilai fitness pada tiap generasi

Dari gambar 4.5. dapat dilihat titik-titik dimana dari jumlah generasi yang dilakukan uji coba didapatkan nilai rata-rata dari setiap jumlah generasi. Pada grafik diatas digunakan grafik regresi yang berguna untuk mengukur pengaruh jumlah generasi terhadap nilai rata-rata *fitness* yang didapatkan. Adapun persamaan regresi yang didapatkan adalah y=0.0024x+0.0332, dimana y menyatakan nilai *fitness* dan x menyatakan jumlah generasi. Sehingga dapat disimpulkan semakin banyak generasi yaitu antara 25, 50 dan 100, semakin baik nilai *fitness* yang didapatkan.

4.4.3.4 Pengaruh rata-rata nilai fitness dari uji coba tiap individu awal

Pada tabel 4.2. tabel pengujian nilai *fitness* berdasarkan individu awal dan generasi yang berbeda, diperoleh nilai rata-rata dari nilai *fitness* yang dihasilkan. Nilai rata-rata *fitness* yang didaptkan adalah menjumlahkan nilai *fitness* yang diperoleh dari setiap pengujian pada jumlah individu awal yang sama dengan menambahkan dari pengujian genrasi yang berbeda. Adapaun grafik pengaruh rata-rata dapat dilihat pada gambar 4.6.



Gambar 4. 6 Grafik pengaruh rata-rata nilai fitness pada tiap generasi

Dari gambar 4.6. dapat dilihat titik-titik dimana dari jumlah individu awal yang dilakukan uji coba didapatkan nilai rata-rata dari setiap jumlah individu awal. Pada grafik diatas digunakan grafik regresi yang berguna untuk mengukur pengaruh jumlah individu awal terhadap nilai rata-rata *fitness* yang didapatkan. Adapun persamaan regresi yang didapatkan adalah y=0.0024x+0.0357, dimana y menyatakan nilai *fitness* dan x menyatakan jumlah generasi. Dari keterangan diatas, dapat disimpulkan semakn banyak jumlah individu yaitu antara 3, 5 dan 7 individu awal, semakin tinggi nilai *fitness* yang didapatkan.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang diperoleh pada penelitian ini adalah :

1. Pada penelitian ini telah dibuat model genetika untuk permasalahan Multi-dimensinal Knapsack Problem (MDKP) untuk menentukan menu makanan sehat. Menu makanan yang terbentuk direpresentasikan dengan kromosom array dua dimensi dengan kolom merepresentasikan item makanan dan baris merepresentasikan menu makanan. Dari kromosom tersebut dihitung density masing-masing gen dan dipilih gen terbaik untuk selanjutnya dihitung nilai penalty. Setelah nilai penalty didapatkan, maka akan didapatkan nilai fitness yang merepresentasikan kondisi suatu kromosom tersebut baik atau buruk. Selanjutnya adalah seleksi kromosom menggunakan roda roulette untuk dijadikan *parent* dan dilanjutkan proses *crossover* dan mutasi sebanyak iterasi yang diinputkan oleh *user*. Setelah iterasi berhenti, maka kromosom dengan nilai *fitness* tertinggi yang akan dipilih menjadi menu.
2. Peningkatan jumlah generasi dan individu dalam mendapatkan individu terbaik menunjukkan nilai *fitness* yang semakin baik. Berdasarkan hasil uji coba, probabilitas *crossover* terbaik diperoleh pada generasi sebanyak 70%, dan Pm sebesar 50%. Probabilitas mutasi yang semakin besar akan menaikkan nilai fitness. Mutasi yang digunakan adalah mutasi biner random dan berdasarkan pada probabilitas mutasi dengan mengganti gen yang kurang dari probabilitas mutasi, serta crossover yang digunakanan adalah crossover dua titik yang ditentukan secara random berdasarkan pada probabilitas crossover yang digunakan.

5.2 Saran

Metode MDKP dan algoritma genetika cocok digunakan untuk permasalahan menu makanan, akan tetapi konstrain yang digunakan sebaiknya ditambah mencakup warna, harga, konsistensi, cara penyadian, temperatur, dan hal lain yang berhubungan dengan perencanaan menu.

DAFTAR PUSTAKA

A. E. Eiben dan J. E. Smith. 2003. *Introduction to Evolutionary Computing*. Budapest, Brisbol : Amsterdam.

Almatsier, Sunita. 2004. *Penuntun Diet Edisi Baru*. Jakarta : Gramedia Pusat.

Budiyanto, Moch Agus Krisno. 2002. *Gizi dan Kesehatan*. UMM Press : Malang.

Barbara Kurousic Seljak. *Computer-Based Dietary Menu Planning*. Proceedings of the 7th WSEAS International Conference on Evolutionary Computing, Cavtat, Croatia, June 12-14, 2006 (pp39-44).

Buku Praktis Ahli Gizi. 2008. Jurusan Gizi Politeknik Kesehatan Depkes : Malang.

Eckholm, Erick P.. 1985. *Masalah Kesehatan Lingkungan sebagai Sumber Penyakit*. PT Gramedia : Jakarta.

Gen, Mitsuo dan Cheng, Runwei. 2000. *Genetic Algorithm and Engineering Optimization*. John Willey and Sons, inc : New York.

Izumi Satoru, Kuriyama Dai, Miura Yutaro, Yasuda Naofumi, Yotyusukuraryo, Kato Yashusi dan Takahashi Kaoru. *Design and Implementation of an Ontology-based Health Support System*. Technical Report of IEICE SS, Vol. 106, No.523, pp.19-24, 2007.

Jennifer Mankoff, Gary Hsieh, Ho Chak Hung, Sharon Lee dan Elizabeth NItao. *Using Low-Cost Sensing to Support Nutritional Awareness*. Electrical Engineering and Computer Science Department, University of California at Berkeley. Berkeley, CA 94720-1770.

K.A. Buckle, R. A. Edwards, G.H. Fleet dan M. Wootton. 1987. *Ilmu Pangan*. Penerbit Universitas Indonesia : Jakarta.

Kartina Diah K. W., Mardhiah Fadhli, dan Charly SUsanto. 2010. *Penyelesaian Knapsack Problem Menggunakan Algoritma Genetika*. Seminar Nasional Informatika 22 Mei 2010 : Yogyakarta. ISSN : 1979-2328.

Kei Kitahara dan Hideaki Kanai. 2010. *A Menu-planning Support System to Facilitate Face-to-Face Interactions*. CSCW 2010, February 6-10-2010, Savannah Georgia, USA. ACM 978-1-60558-795-0/10/02.

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

Lemoin, Aaron S.. 2011. JDPET : *Java Dynamic Programming Educational Tool*. The Faculty of the Department of Computer Science San Jose State University.

Marshall, Janette. 2006. *Makanan Sumber Tenaga*. Erlangga : Jakarta.

Mien K. Mahmud, Hermana, Nils Aria Zulfianto, Rossi Rozanna, Iskari Ngadiarti, Budi Hartati, Bernadus dan Tinexcely. 2005. *Daftar Komposisi Bahan Makanan (DKBM)*. Persatuan Ahli Gizi Indonesia (PERSADI) : Jakarta.

Misky, Dudi. 2005. Terapi Diet Ideal. Progres : Jakarta.

P. M. Gaman dan K. B. Sherrington. 1981. *The Science of Food, An Introduction to Food Science, Nutrition and Microbiology, Secon Edition*. Pergamon Press : England.

Pudjiadi, Solihin. 2003. *Ilmu Gizi Klinis pada Anak Edisi Keempat*. Balai Penerbit FKUI : Jakarta.

Suryatin, Budi. 2004. *SAINS Materi dan Sifatnya*. Grasindo : Jakarta.

Suyanto. 2007. *Artificial Intelligence Searching, Reasoning, Planning dan Learning*. Informatika : Bandung.

Tomoko Kashima, Shimpei Matsumoto dan Hiroaki Ishii. *Evaluation of Menu Planning Capability Based on Mult-dimensional 0/1 Knapsack Problrm of*

Nutritional Management System. IAENG International Journal of Applied Mathematics, 39:3, IJAM_39_04. 1 August 2009.

Tri Susanto dan Tri Dewanti Widyaningsih. 2004. *Dasar-Dasar Ilmu Pangan dan Gizi*. Penerbit Akademika : Yogyakarta.

Widodo, Thomas Sri.2012. *Komputasi Evolusioner*. Graha Ilmu : Yogyakarta.