EVALUASI KINERJA *GENETIC ALGORTIHM* (GA) DENGAN STRATEGI PERBAIKAN KROMOSOM STUDI KASUS *MULTI-CHOICE MULTI-DIMENSIONAL KNAPSACK PROBLEM*

(Skripsi)

Oleh

RANI CAHYANI

JURUSAN ILMU KOMPUTER

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS LAMPUNG

2016

ABSTRACT

EVALUATION PERFORMANCE OF GENETIC ALGORITHM (GA) WITH REPAIRING STRATEGY IN MULTI-CHOICE MULTI-DIMENSIONAL KNAPSACK PROBLEM

By

RANI CAHYANI

The knapsack problem is a combinatorial optimization problem where one has to maximize the benefit of objects in knapsack without exceeding its capacity. Generally, Knapsack Problem can be divided into four types, one of which is a Multi-choice Multi-dimensional Knapsack Problem (mmKP). Some previous research has been developed to search the optimal values using various methods. Ant Colony Optimization (ACO) Algorithm is used in a case study on research mmKP (Sohel et al, 2010). Modified Method of Reactive Local Search. research Hify et al, 2007 and is effective to obtain the optimum value. One other heuristic method is quite popular is the Genetic Algorithm (GA). Genetic algorithms (GA) is an optimization technique for searching very large spaces that models the role of the genetic material in living organisms.

However, the issue of Knapsack has a function obstacles that will divide the space into two solutions : feasible and not feasible solutions. There are some coping strategies constraints that have been conducted in various studies.This research aims to implement GA with chromosome repair strategy on a case study Multi-choice Multi-dimensional Knapsack Problem, in addition this research added local optimum search methods. The method developed is tested on some test data test (test problem) from the OR Library. This method of performance will be evaluated by comparing performance with strategy GA penalty.

Keywords: *Combinatorial Optimization, Genetic Algorithm* (GA), *Multi-Choice Multi-dimensional Knapsack Problem, Repairing Strategy*

ABSTRAK

EVALUASI KINERJA GENETIC ALGORTIHM (GA) DENGAN STRATEGI PERBAIKAN KROMOSOM STUDI KASUS MULTI-CHOICE MULTI-DIMENSIONAL KNAPSACK PROBLEM

Oleh

RANI CAHYANI

Knapsack Problem merupakan persoalan optimasi kombinatorial yang bertujuan untuk pengoptimalan jumlah nilai prioritas dari beberapa barang yang akan di-*packing* dalam suatu *knapsack* tanpa melebihi kapasitasnya. Secara umum *Knapsack Problem* dibedakan menjadi 4 tipe, salah satunya adalah *Multi-choice Multi-dimensional Knapsack Problem* (mmKP). Beberapa penelitan terdahulu telah dikembangkan untuk pencarian nilai optimal menggunakan berbagai metode. Algoritma *Ant Colony Optimization* (ACO) digunakan dalam studi kasus mmKP pada penelitian (Sohel *et al*, 2010). Metode *Modified Reactive Local Search.* pada penelitian Hify *et al*, 2007 dan dinilai cukup efektif untuk mendapatkan nilai optimum. Salah satu metode heuristik lain yang cukup populer adalah *Genetic Algorithm* (GA). GA merupakan salah satu teknik pencarian nilai optimasi yang mengadapatasi dari proses evolusi biologi.

Persoalan *Knapsack* memiliki fungsi kendala yang akan membagi ruang solusi menjadi dua, yaitu solusi layak dan tidak layak. Terdapat beberapa strategi penanganan kendala yang telah dilakukan di berbagai penelitian. Penelitian ini bertujuan untuk mengimplementasikan GA dengan strategi perbaikan kromosom pada studi kasus *Multi-choice Multi-dimensional Knapsack Problem,* selain itu ditambahkan metode pencarian optimum lokal. Metode yang dikembangkan diujicobakan pada beberapa data tes uji (*test problem*) dari OR *Library*. Kinerja metode ini akan dievaluasi dengan membandingkan dengan kinerja GA dengan strategi *penalty*.

Kata Kunci : Optimasi Kombinatorial, Algortima Genetika, *Multi-Choice Multi-Dimensional Knapsack Problem,* Strategi Perbaikan Kromosom.

EVALUASI KINERJA GENETIC ALGORTIHM (GA) DENGAN STRATEGI PERBAIKAN KROMOSOM STUDI KASUS MULTI-CHOICE MULTI-DIMENSIONAL KNAPSACK PROBLEM

Oleh

RANI CAHYANI

Skripsi

Sebagai Salah Satu Syarat untuk Memperoleh Gelar

SARJANA KOMPUTER

pada

Jurusan Ilmu Komputer

Fakultas Matematika dan Ilmu Pengetahuan Alam

JURUSAN ILMU KOMPUTER

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS LAMPUNG

2016

Judul Skripsi	:	**EVALUASI KINERJA GENETIC ALGORTIHM (GA) DENGAN STRATEGI PERBAIKAN KROMOSOM STUDI KASUS MULTI-CHOICE MULTI-DIMENSIONAL KNAPSACK PROBLEM**
Nama Mahasiswa	:	Rani Cahyani
No. Pokok Mahasiswa	:	1217051054
Jurusan	:	Ilmu Komputer
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam

MENYETUJUI

1. Komisi Pembimbing

Dr. Eng. Admi Syarif
NIP. 19670103 199203 1 003

Aritoteles, S.Si., M.Si
NIP. 19810521 200604 1 002

2. Mengetahui
Ketua Jurusan Ilmu Komputer
FMIPA Universitas Lampung

Dr. Ir. Kurnia Muludi, M.S.Sc
NIP. 19640616 198902 1 001

MENGESAHKAN

1. Tim Penguji

   Ketua : **Dr. Eng. Admi Syarif** .....

   Sekretaris : **Aristoteles, S.Si., M.Si.** .....

   Penguji
   Bukan Pembimbing : **Rico Andrian, S.Si., M.Kom.** .....

2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Prof. Warsito, S.Si., D.E.A., Ph.D.
NIP. 19710212 199512 1 001

Lulus Ujian Skripsi Tanggal: **23 Desember 2016**

PERNYATAAN

Saya yang bertanda tangan di bawah ini, menyatakan bahwa skripsi saya yang berjudul "Evaluasi Kinerja Genetic Algorithm (GA) dengan Strategi Perbaikan Kromosom Studi Kasus *Multi-Choice Multi-Dimensional Knapsack Problem*" merupakan karya saya sendiri dan bukan karya orang lain. Semua tulisan yang tertuang di skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung. Apabila dikemudian hari terbukti skripsi saya merupakan hasil penjiplakan atau dibuat orang lain, maka saya bersedia menerima sanksi berupa pencabutan gelar yang telah saya terima.

Bandar Lampung, 23 Desember 2016

Rani Cahyani
NPM. 1217051054

RIWAYAT HIDUP

Penulis dilahirkan pada 20 Juni 1994 di Bandar Lampung, sebagai ketiga dari lima bersaudara dengan Ayah bernama Indra Cahya dan Ibu bernama Iryani. Penulis menyelesaikan pendidikan Sekolah Dasar (SD) di SD Negeri 1 Karang Anyar tahun 2006, menyelesaikan Sekolah Menengah Pertama (SMP) di SMP Negeri 10 Bandar Lampung tahun 2009, kemudian melanjutkan jenjang Sekolah Menengah Atas (SMA) di SMA Negeri 9 Bandar Lampung dan lulus di tahun 2012.

Pada tahun 2012, penulis terdaftar sebagai mahasiswa Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung melalui jalur SNMPTN Undangan. Pada bulan Januari-Februari 2015, penulis melakukan kerja praktik di PT Tunas Dwipa Matra Lampung. Pada bulan Juni-September 2015, penulis melakukan Kuliah Kerja Nyata selama 60 hari di Desa Marga Kencana Kecamatan Daya Murni Kabupaten Tulang Bawang Barat.

Selama perkuliahan, penulis mengikuti beberapa organisasi, yaitu Natural, Organisasi Jurnalis FMIPA dan menjabat sebagai Sekretaris Bidang Sirkulasi dan Periklanan kepengurusan periode 2013-2014. Organisasi Himakom, penulis menjabat sebagai Sekretaris Bidang Usaha kepengurusan periode 2014-2015.

PERSEMBAHAN

Perjuangan merupakan Pengalaman Berharga yang Dapat Menjadikan Kita Manusia yang Berkualitas

Teruntuk Keluargaku, Ibu, Bapak, Abang dan Adik yang sangat kucintai, kupersembahkan skripsi ini

Terimakasih untuk kasih sayang, perhatian, pengorbanan, usaha, dukungan moril maupun materi, motivasi dan do'a-do'a yang tiada henti untuk kesuksesanku....

Teuntuk sahabat dan teman-teman tersayang,

Terimakasih untuk canda tawa, tangis dan perjuangan yang telah kita lewati bersama....

Teuntuk kamu yang masih setia denganku,

Terimakasih untuk kesabaran dan dukungan selama ini, wève been through it all together....

MOTTO

“Maka sesungguhnya bersama kesulitan ada kemudahan. Sesungguhnya bersama kesulitan ada kemudahan. Maka apabila engkau telah selesai (dari sesuatu urusan), tetaplah bekerja keras (untuk urusan yang lain). Dan hanya kepada Tuhanmulah engkau berharap.”

(Q.S.Al-Insyirah:6-8)

"Pendidikan merupakan senjata paling ampuh yang bisa kamu gunakan untuk merubah dunia"

(Nelson Mandela)

SANWACANA

Assalamualaikum wr. wb.

Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT, yang maha pengasih lagi maha penyayang, yang telah memberikan inspirasi berpikir serta motivasi untuk berkarya dalam menyelesaikan amanah ini. Selama studi sampai dengan penulisan skripsi ini, penulis banyak menerima bantuan baik secara langsung maupun tidak langsung. Oleh karena itu, penulis menyampaikan rasa terimakasih yang sebesar-besarnya kepada:

1. Bapak Dr. Eng. Admi Syarif, selaku Pembimbing I atas segala saran, bimbingan, kepercayaan serta berbagi pengalamannya untuk kerja-keras dalam hidup ini dan juga motivasi untuk belajar dan terus belajar.
2. Bapak Aristoteles, S.Si., M.SI. Selaku Pembimbing II dan Pembimbing Akademik atas segala waktu yang telah diberikan untuk diskusi, bimbingan serta pengarahannya selama ini.
3. Bapak Rico Andrian, S.Si., M.Kom selaku pembahas sekaligus sekretaris jurusan matematika yang telah memberikan kritik dan saran dalam skripsi ini, serta pengarahan dan bimbingan selama ini.
4. Dr. Ir. Kurnia Muludi selaku Ketua Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung.

5. Bapak Prof. Warsito, S.Si., DEA., Ph.D selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung.
6. Seluruh staff dosen jurusan Ilmu Komputer : Dwi Sakethi, M.Kom. Astria Hijriani, S.Kom., Machudor Yusman, M.Kom, Rangga Firdaus, S.Kom., M.Kom., Ossy Dwi Endah Wulansari, M.T., Febi Eka Febriansyah, ST., Anie Rose Irawati, ST, M.Cs., Tristiyanto, Ph.D, M.Kom, Favorisen Rossy King, S.Kom., M.Si., serta Zainudin S.Kom atas bimbingan, ilmu pengetahuan, motivasi serta nasehatnya selama ini.
7. Kedua Orang Tua tercinta Mamak, Bapak atas segala kasih sayang dan cinta kasih yang senantiasa tercurah.
8. Abang Chepy, Abang Riko, Rina, dan Nita yang sangat penulis cintai atas kebersamaan, kasih sayang, doa, kepercayaan serta kegembiraan yang diberikan.
9. Muhammad Rahman Koestanto sebagai teman seperjuangan terbaik atas segala dukungan dan bantuannya.
10. *Partner* Skripsi Dian Anggraini dan para *Supporter* Skripsi Nila Liliani Prihatin, Esti Putri Cindona, Nurul Hamidah atas bantuan dan kerjasamanya, semoga kita bisa menjadi saudara yang baik.
11. Afriska Amidya, Cindy Covia Saris, Rizki Aptriani, Haryati, Beta Yolanda, Iluh, Deby Aryandi yang tercinta atas segala bantuan dan nasehatnya serta kebersamaannya selama masa kuliah.
12. Seluruh rekan-rekan Himakom Kepengurusan 2011-2012 dan 2012-2013 atas ilmu, Pengalaman, Motivasi Organisasi serta pengetahuannya selama ini, yang telah melebur menjadi sebuah kekeluargaan yang hakiki, Yakin Usaha Sampai.

13. Seluruh teman seperjuangan ILKOM 2012 atas kerjasama, keceriaan dan semangat bersama dalam menjalankan masa kuliah di Ilkom, Fmipa, Unila tercinta.
14. Pemerintah Indonesia melalui Program Pendidikan Beasiswa Bidik Misi yang mampu mewujudkan mimpi banyak keluarga Indonesia untuk memperbaiki kualitas hidup.

Demikian uncapan terimakasih penulis sampaikan kepada semua pihak, hanya Allah SWT yang dapat membalas dan memberi rahmat-Nya atas segala usaha dan bantuan yang telah diberikan kepada penulis.

Bandar Lampung, 22 Desember 2016

Rani Cahyani

DAFTAR ISI

Halaman

DAFTAR ISI ix

DAFTAR GAMBAR x

DAFTAR TABEL xiii

DAFTAR LAMPIRAN xiv

BAB I PENDAHULUAN 1

1.1. Latar Belakang 1
1.2. Rumusan Masalah 4
1.3. Batasan Masalah 4
1.4. Tujuan 4
1.5. Manfaat 4

BAB II TINJAUAN PUSTAKA 5

2.1. Persoalan *Knapsack* 5
2.2. Jenis Knapsack Problem 6
2.2.1. 0/1 *Knapsack Problem* 6
2.2.2. *Bounded Knapsack Problem* 7
2.2.3. *Multi-choice Knapsack Problem* 8
2.2.4. *Multi-dimensional Knapsack Problem* 9
2.2.5. *Multi-choice Multi-dimensional Knapsack Problem* 10
2.3. Metode Penyelesaian *Knapsack Problem* 11
2.4. Algoritma Genetika 13
2.4.1. Struktur Umum GA 14

2.4.2. Pengkodean....16
2.4.3. Operator Genetika 17
2.4.3.1. Seleksi 17
2.4.3.2. Pindah Silang (*Crossover*) 19
2.4.3.3. Mutasi....21
2.4.4. Parameter Genetika 23
2.5. Pencarian Optimum Lokal (*Local Search*) 24
2.6. Strategi Penanganan Kendala 25

BAB III METODOLOGI PENELITIAN....28

3.1. Waktu dan Tempat Penelitian....28
3.2. Lingkungan Pengembangan....28
3.3. Tahapan Penelitian....28
3.3.1. Tahap Pertama : Studi Literatur 30
3.3.2. Tahap Kedua : Penyusunan Algortima....32
3.3.3. Tahap Ketiga : Pengembangan Program 38
3.3.3.1. Representasi Kromosom 38
3.3.3.2. *Decode*....38
3.3.3.3. *Crossover* 41
3.3.3.4. Mutasi....43
3.3.3.5. *Pencarian Optimum Lokal (Local Search)*) 44
3.3.3.6. Evaluasi *(*Strategi *Repair)* 45
3.3.4. Tahap Keempat : Pengujian dan Eksperimen....46

BAB IV HASIL DAN PEMBAHASAN 47

4.1. Hasil Penelitian 47

BAB V PENUTUP....55

5.1. Kesimpulan 55
5.2. Saran 55

DAFTAR PUSTAKA

LAMPIRAN

DAFTAR GAMBAR

Gambar Halaman

2.1 Contoh *Crossover* 1-titik 19

2.2 Contoh *Crossover* 2-titik 20

2.3 Contoh *Crossover n*-titik..... 20

2.4 Ilustrasi Metode Pembalikan 21

2.5 Ilustrasi Metode Penyisipan..... 22

2.6 Ilustrasi Metode Pemindahan 22

2.7 Ilustrasi Metode Penukaran 22

2.8 Ilustrasi Metode Pengantian 22

2.9 *Local Search* 24

2.10 Ruang Kelayakan Solusi..... 25

3.1 Alur Penelitian..... 28

3.2 Diagram Alir GA..... 31

3.3 Diagram alir *hr*GA..... 32

3.4 Diagram Alir *Crossover* 33

3.5 Diagram Alir Mutasi..... 34

3.6 Diagram Alir *Merge-Sort* 35

3.7 Diagram Alir *Local Search*..... 36

3.8 Kromosom 37

3.9 *Decoding* Kromosom 28

3.10 Contoh *Crossover* 2-titik..... 40

3.11 Ilustrasi Metode Penukaran 42

4.1 Perbandingan waktu *r*GA dan *hr*GA 49

4.2 Nilai *Error* 52

4.3 Nilai *Profit* Generasi I02 53

4.4 Nilai *Profit* Generasi I06 53

DAFTAR TABEL

Tabel Halaman

2.1 Istilah dalam Metode GA 14

3.1 *Decoding* Kromosom..... 39

4.1 Hasil Data Penelitian mmKP 47

4.2 Perbandingan Hasil Optimasi mmKP..... 50

4.3. Perbandingan Hasil Strategi Penanganan Kendala..... 53

DAFTAR LAMPIRAN

Lampiran Halaman

1. Hasil rGA 59

2. Hasil hrGA.....60

3. Nilai *Error* 61

BAB I
PENDAHULUAN

1.1. Latar Belakang

Optimasi adalah proses menyelesaikan suatu masalah tertentu supaya berada pada kondisi yang paling menguntungkan dari suatu sudut pandang (Zukri, 2014). Persoalan optimasi dikelompokan menjadi empat, yaitu optimasi tanpa pembatas, optimasi dengan pembatas, optimasi dengan multi fungsi tujuan dan optimasi kombinatorik (Syarif, 2014). Beberapa persoalan yang termasuk kelompok optimasi kombinatorik adalah *Set Covering*, *Bin Packing*, penjadwalan, penugasan, *Knapsack Problem* dan sebagainya (Gen dan Cheng, 2000).

Persoalan *Knapsack* (*Knapsack Problem* : KP) pertama kali diteliti oleh Dantzig pada akhir tahun 1950an. Menurut Gupta, 2013 KP termasuk ke dalam permasalahan optimisasi dan kombinatorik. Dalam permasalahan kombinatorik, KP termasuk ke dalam persoalan *NP-Hard Combinatorial*, dimana algoritma penyelesaiannya memiliki kompleksitas non-polinomial dengan skala ruang solusi yang sangat besar dan peningkatan koefisien secara eksponensial (Pisinger, 1995, Martello, 2000). Sebagai salah satu persoalan optimasi kombinatorik, *Knapsack* dapat digambarkan sebagai persoalan maksimasi keuntungan nilai objek-objek yang dimasukkan dalam sebuah *Knapsack* dengan kapasistas tertentu tanpa melebihi kapasitasnya (Mahajan dan Chopra, 2012).

Beberapa macam permasalahan di dalam dunia nyata yang termasuk ke dalam *KP* yaitu, penganggaran modal, pemilihan proyek, pemuatan kargo, pemotongan stok,

Bin P*acking*, *Material Incision* dan perencanaan ekonomi (Gupta, 2013). Terdapat beberapa macam tipe dari *Knapsack Problem* diantaranya 0/1 *Knapsack Problem* (0/1 KP) (Astuti, 2015), *Bounded Knapsack Problem* (bKP), *Multiple Knapsack Problem* (mKP) (Elizabeth dkk, 2014), *Multi-dimensional Knapsack Problem* (mdKP) (Varnamkhasti, 2012), *Multi-choice Multi-dimensional Knapsack Problem* (mmKP) (Zennaki, 2013). Metode penyelesaian KP yang sudah ditawarkan diantaranya *Branch and Bound* (Fukunaga, 2011), *Dynamic Programming*, *Ant Colony* (Rahman dkk, 2010), *Heuristic* (Shojaei dkk, 2011), dan *Genetic Algorithm* (GA) (Gupta, 2013).

GA merupakan salah satu evolusi/perkembangan dunia komputer dalam bidang kecerdasan buatan (*Artificial Intelligence*). Menurut John McCarthy, 1956, AI bertujuan untuk mengetahui dan memodelkan proses– proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia. GA menjadi salah satu metodologi dalam teknik *soft computing*, sebuah teknik inovasi baru dalam membangun sistem cerdas. Metodologi-metodologi yang digunakan dalam *Soft Computing* diantaranya adalah sistem *Fuzzy* (mengakomodasi ketidaktepatan) dengan terapannya Logika *Fuzzy* (*Fuzzy Logic*), Jaringan Syaraf (menggunakan pembelajaran) dengan Jaringan Syaraf Tiruan (*Neurall Network*), *Probabilistic Reasoning* (mengakomodasi ketidakpastian), *Evolutionary Computing* (optimasi) dengan GA (Kusumadewi, 2003).

GA menjadi salah satu metodologi dalam teknik *soft computing* untuk membangun sistem cerdas. GA dinilai mempunyai hasil yang optimal untuk banyak persoalan, hal ini telah dibuktikan bahwa GA dapat menghasilkan

himpunan solusi optimal yang sangat berguna dengan banyak objek. Kekuatan utama GA adalah kemampuannya untuk menyelesaikan masalah kompleks dalam waktu yang relatif cepat (Mahmudy, 2014).

Penerapan GA terdapat dalam penelitian Sulistiyorini 2015 untuk permasalahan optimasi distribusi barang dua tahap. GA diterapkan pada kasus distribusi barang dua tahap untuk mengoptimasi rute distribusi barang untuk mendapatkan biaya distribusi minimum. GA mampu menyelesaikan permasalahan tersebut dengan memberikan solusi berupa pencarian jalur distribusi yang tepat dan menghitung biaya total minimum dari satu jenis barang dengan distribusi barang dua tahap.

Menurut Shojaei, 2011 dalam *A Parameterized Compositional Multi-dimensional Multiple-choice Knapsack Heuristic for CMP Run-time Management*, melakukan penelitian untuk meningkatkan kualitas *run-time* pada *chip multi-processor* secara heuristik. Aplikasi yang berjalan pada sebuah sistem *multi-processor paltform* membutuhkan sumber daya (aspek *multi-dimensional*) dimana setiap aplikasi memiliki banyak pilihan konfigurasi (aspek *multi-choice*). Tujuan penelitiannya adalah untuk mencari konfigurasi optimal dari total aplikasi yang berjalan.

Penelitian yang akan dilakukan bertujuan untuk mengevaluasi kinerja GA pada studi kasus *Multi-choice Multi-dimensional Knapsack Problem* dengan menggunakan Dataset Benchmark dari *OR-Library*.

1.2 Rumusan Masalah

Rumusan masalah yang akan diselesaikan pada penelitian ini adalah bagaimana

mengimplementasikan dan mengevaluasi efektivitas GA dengan strategi perbaikan kromosom sehingga dapat menghasilkan nilai optimum yang optimal pada studi kasus *Multi-choice Multi-dimensional Knapsack Problem* (mmKP*)*.

1.3 Batasan Masalah

Batasan masalah penelitian ini adalah sebagai berikut :

1. Penelitian ini akan mengevaluasi kinerja GA pada strategi perbaikan kromosom dengan membandingkan hasil dan waktu komputasi.
2. Penelitian ini akan dilakukan pada studi kasus mmKP.
3. Representasi kromosom adalah dengan representasi permutasi.
4. Bahasa pemrograman yang digunakan adalah Matlab.

1.4 Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Mengimplementasikan GA dengan strategi perbaikan kromosom untuk menyelesaikan mmKP.
2. Mengevaluasi kinerja (efektifitas dan efisiensi) GA.

1.5 Manfaat

Manfaat dari hasil penelitian ini adalah sebagai berikut :

1. Menambah pustaka aplikasi GA untuk berbagai persoalan di dunia nyata.
2. Memberikan informasi mengenai efisiensi dan efektivitas kinerja GA untuk menyelesaikan mmKP.
3. Sebagai sumber informasi bagi peneliti yang ingin menyelesaikan mmKP

BAB II
TINJAUAN PUSTAKA

2.1 Persoalan *Knapsack*

Persoalan *Knapsack* (*Knapsack Problem* : KP) merupakan salah satu permasalahan optimasi kombinatorial, dimana algoritma penyelesaiannya memiliki kompleksitas non-polinomial dengan skala ruang solusi yang sangat besar dan peningkatan koefisien secara eksponensial (Pisinger, 1995, Martello, 2000). Permasalahan inti dari KP adalah bagaimana cara menentukan pemilihan barang dari sekumpulan barang dimana setiap barang (*item*) mempunyai nilai berat (*weight)* dan nilai keuntungan (*profit)* masing-masing, sehingga dari pemilihan barang tersebut didapatkan keuntungan (*profit)* yang maksimum tanpa melebihi kapasitas *Knapsack*, dengan kata lain telah mencapai *optimal packing*.

KP merupakan salah satu permasalahan yang memiliki satu fungsi tujuan dan satu / lebih fungsi kendala (*constraint*). Kombinasi dari barang terpilih akan menentukan hasil optimasi. Contoh sederhana dari permasalahan KP tersebut diantaranya seperti pemuatan barang kontainer (*cargo loading*), pemotongan stok, persoalan bagaimana seorang pengusaha yang memiliki modal mendapatkan keuntungan dengan investasinya, persoalan bagaimana seorang pendaki dapat mengisi tasnya dengan kebutuhan pendakiannya dengan berat yang minimum dan fungsi yang maksimal dan sebagainya (Astuti, 2015).

2.2 Jenis Knapsack Problem

Terdapat beberapa macam tipe *Knapsack Problem* (Martello, 1990), diantaranya adalah sebagai berikut :

2.2.1 *0/1 Knapsack Problem* (0/1 KP)

Knapsack 0-1 merupakan persoalan pemilihan memilih *n* barang (*item*) untuk dikumpulkan dengan batas kapasitas *c* dan mendapatkan nilai keuntungan (*profit*) yang maksimum tanpa melebihi kapasitas *c*. Contoh persoalannya seperti bagaimana menetukan 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. Berikut model matematis dari *0/1* KP :

i = indeks barang (*item)*

n = jumlah barang (*item)*

p_i = keuntungan *(profit)* dari barang (*item)* ke-*i*

w_i = berat *(weight)* dari barang (*item)* ke-*i*

c = kapasitas *Knapsack*

$$\textbf{max} \ : z = \sum_{i=1}^{n} p_i x_i \tag{2.1}$$

$$\textbf{s.t} \quad \sum_{i=1}^{n} w_i x_i \leq c \tag{2.2}$$

$$x_i = 0 \ atau \ 1, i \in N = (1, \dots, n) \tag{2.3}$$

$$x_i = \begin{cases} 1 \text{ ; jika barang ke} - j \text{ di kelompok ke} - i \text{ terpilih} \\ 0 \text{ ; barang ke} - j \text{ di kelompok} - i \text{ tidak terpilih} \end{cases}$$

2.2.2 *Bounded Knapsack Problem (BKP)*

Bounded Knapsack Problem merupakan persoalan pemilihan yakni memilih *n* barang (*item*) yang memiliki berat (*weight*) dan keuntungan (*profit*) dengan batas kapasitas (*c).* Setiap barang yang tersedia sebanyak *n* namun jumlahnya terbatas (*b*). Fungsi tujuannya adalah untuk mendapatkan nilai keuntungan yang maksimum tanpa melebihi kapasitas. Terdapat juga tipe *Unbounded Knapsack Problem,* perbedaannya pada stok barang (*item*) jumlahnya tidak terbatas. Berikut diberikan tipe-tipe *n item* dan *Knapsack,* dengan:

i = indeks kelompok (*class)*

n = jumlah barang (*item)*

p_i = *k*euntungan / *profit* dari barang (*item)* ke- *i*

w_i = berat *(weight)* dari barang (*item)* ke- *i*

b_i = batas atas yang tersedia dari barang (*item)* tipe - *i*

c = kapasitas *Knapsack*

Pilih sejumlah $x_i = (1,...,n)$ dari masing-masing tipe barang sehingga:

$$\textbf{max} \ : z = \sum_{i=1}^{n} p_i x_i \tag{2.4}$$

$$\textbf{s.t} \quad \sum_{i=1}^{n} w_i x_i \ \leq c \tag{2.5}$$

$$0 \leq x_i \leq b_i \ \text{ dan integer, } i \in N = (1, \dots, n) \tag{2.6}$$

$$x_i = \begin{cases} 1 \text{ ; jika barang ke} - j \text{ di kelompok ke} - i \text{ terpilih} \\ 0 \text{ ; barang ke} - j \text{ di kelompok} - i \text{ tidak terpilih} \end{cases}$$

2.2.3. *Multi-choice Knapsack Problem* (mcKP)

Multichoice Knapsack Problem (mcKP) merupakan persoalan pemilihan yakni memilih *n* item yang memiliki berat (*weight*) dan keuntungan (*profit*) dengan kapasistas (*c)* dalam tiap kelompok (*i*). Fungsi tujuannya adalah untuk mendapatkan nilai keuntungan yang maksimum tanpa melebihi kapasitas, dengan memilih tepat satu item dari setiap kelas. Persoalan dalam kehidupan sehari-hari diantaranya *Menu Planning* dan *Capital Budgeting* (Chang, 2011). Diberikan tipe-tipe *n* barang (*item)* dan *Knapsack*, dengan:

i = indeks kelompok (*class)*

j = indeks barang (*item)*

m = jumlah barang (*item)*

n = jumlah kelompok (*class*)

p_i = keuntungan (*profit)* dari barang (*item)* ke- *i*

w_i = berat (*weight)* barang (*item)* ke- *i*

c = kapasitas *Knapsack*

Pilih sejumlah x_{ij} = (1,...,*n*) dari masing-masing tipe *item* sehingga:

$$\textbf{max} : z = \sum_{i=1}^{n} . \sum_{i=1}^{m} p_i \, x_{ij} \tag{2.7}$$

$$\textbf{s.t} \quad \sum_{i=1}^{n} w_i x_{ij} \leq c_i \, , j \in M = \{1, \dots, m\} \tag{2.8}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1, i \in N = \{1, \dots, n\} \tag{2.9}$$

$$x_{ij} = \begin{cases} 1 \, ; \text{jika barang ke} - j \text{ di kelompok ke} - i \text{ terpilih} \\ 0 \, ; \text{ barang ke} - j \text{ di kelompok} - i \text{ tidak terpilih} \end{cases}$$

2.2.4 *Multi-dimensional Knapsack Problem* (mdKP)

Multi-dimensional Knapsack Problem merupakan persoalan pemilihan yakni memilih *n* item yang memiliki berat (*weight*) dan nilai keuntungan (*profit*) dengan kapasistas (*c)* dalam tiap kelas (*m*). Fungsi tujuan mdKP sama halnya dengan fungsi tujuan mcKP yakni memilih tepat satu barang (*item*) dari setiap kelompok, dengan barang yang sama akan dikelompokan menjadi satu kelompok. Contoh persoalan mdKP dianataranya *Project Selection*, *Cutting Stock*, *Cargo Loading*, dan *Combinatorial Auctions (Chang, 2011).* Berikut model matematis mdKP, diberikan tipe-tipe *n item* dan *Knapsack*, dengan:

i = indeks kelompok (*class)*

j = indeks barang (*item)*

n = jumlah kelompok (*class*)

w_{ij} = *weight* kelompok (*class*) ke- *i* barang *(item*) ke- *j*

c_i = kapasitas *Knapsack* ke kelompok-*i*

p_j = keuntungan *(profit)* barang *(item)* ke-*j*

Pilih sejumlah x_{ij} = (1,...,*n*) dari masing-masing tipe *item* sehingga:

$$\textbf{max} \ : z = \sum_{j=1}^{n} p_j x_{ij} \tag{2.10}$$

$$\textbf{s.t} \qquad \sum_{i=1}^{n} w_{ij} x_j \ \leq c_i \ , i = \{1, \dots, m\} \tag{2.11}$$

$$x_j \ \in \{0,1\}, j = \{1, \dots, n\}.$$

$$x_{ij} = \begin{cases} 1 \ ; \text{jika barang ke} - j \text{ di kelompok ke} - i \text{ terpilih} \\ 0 \ ; \ \text{barang ke} - j \text{ di kelompok} - i \text{ tidak terpilih} \end{cases}$$

2.2.5 *Multi-choice Multi-dimensional Knapsack Problem* (mmKP)

MMKP merupakan salah satu jenis *KP* yang memiliki tingkat kompleksitas lebih tinggi dibanding dengan tipe permasalahan *KP* lainnya. Permasalahan mmKP merupakan gabungan dari tipe permasalahan *Multichoice Knapsack Problem* (mcKP) dan *Multidimensional Knapsack Problem* (mdKP). Pada dasarnya permasalahan *Knapsack* adalah bagaimana cara pemilihan barang yang memiliki berat dan nilai keuntungan sedemikian rupa, dimana barang yang terpilih nantinya memiliki nilai keuntungan optimal dan berat yang tidak melebihi kapasitas maksimal. Implementasi mmKP diantaranya dalam permasalahan *Chip Multiprocessor Run-time Resource Management, Global Routing of Wiring in Circuits* dan permasalahan lainnya dalam *Service Level Agreement* dan model dari *allocation resource (Hifi, 2004).* Berikut merupakan model matematis dari mmKP:

I = indeks kelompok (*class)*
j = indeks barang (*item)*
k = sumber daya material / *resources*
n = jumlah kelompok (*class*)
l_i = jumlah barang (*item*)
v_{ij} = nilai solusi barang ke-*j* di kelompok ke-*i/ value of the solution*
R_k = pembatas bahan *k* / *resource constraint*
r_{ijk} = kebutuhan *resources k* untuk memproduksi barang *j di* kelompok *i*

Pilih sejumlah $x_i = (1,...,n)$ dari masing-masing tipe barang sehingga:

$$\textbf{max} : z = \sum_{i=1}^{n} . \sum_{j=1}^{l_i} x_{ij}\, v_{ij} \tag{2.12}$$

$$\textbf{s.t} \quad \sum_{i=1}^{n} . \sum_{j=1}^{l_i} x_{ij}\, r_{ijk} \leq R_k \tag{2.13}$$

$$1 \leq k \leq m,\ x_{ij} \in \{0,1\},\ \sum_{j=1}^{l_i} x_{ij} = 1 \quad \forall\, i$$

$$x_{ij} = \begin{cases} 1\ ; \text{jika barang ke} - j \text{ di kelompok ke} - i \text{ terpilih} \\ 0\ ;\ \text{barang ke} - j \text{ di kelompok} - i \text{ tidak terpilih} \end{cases}$$

2.3 Metode Penyelesaian *Knapsack Problem*

Beberapa teknik atau metode telah digunakan untuk menyelesaikan persoalan *Knapsack*, diantaranya adalah *Branch and Bound*, *Dynamic Programming*, S*tate Space Relaxations*, *Pre-prosecessing*, GA dan sebagainya. Algoritma *Branch and Bound* juga merupakan metode pencarian di dalam ruang solusi secara sistematis. Ruang Solusi diorganisasikan ke dalam pohon ruang status. Pembentukan pohon ruang status pada Algoritma *Branch and Bound* berbeda dengan pembentukan pohon pada algoritma runut-balik. Bila pada algoritma runut-balik ruang solusi dibangun secara *Depth-First Search* (DFS), maka pada Algoritma *Branch and Bound* ruang solusi dibangun dengan skema *Breadth-First Search* (BFS) (Shena, 2007).

Strategi algoritma lainnya yang dapat digunakan salah satunya adalah *Brute Force.* Algoritma ini dapat diibaratkan sebagai algoritma *trial* and *error* yang tidak memiliki sistematika yang baik dalam penyelesaikan berbagai kasus. Strategi algoritma tersebut tidak efisien dan membutuhkan waktu yang lebih lama sehingga terkadang menyebabkan kasus *Time Limit Exceeded* pada beberapa kesempatan yang membatasi waktu kompilasi dan *run-time* program. Pendekatan Algoritma *Brute Force* adalah *Straight-forward* dimana pada kasus *Knapsack*, semua kombinasi barang yang mungkin dengan penanda 1/0 (*true*/*false*, ambil/tidak) untuk menentukan barang tersebut akan dimasukkan ke dalam list barang yang diambil atau tidak.

Algoritma *Greedy* merupakan alternatif lain untuk memecahkan permasalahan *Knapsack*. Algoritma ini sering digunakan untuk memecahkan masalah optimasi dalam berbagai kasus seperti pada pohon bentangan terpendek (*Minimum Spanning Tree*). Secara umum teknik ini menggunakan *heuristic* untuk mencari solusi *sub-optimum* sehingga diharapkan solusi optimum.

Dynamic programming adalah metode pemecahan masalah dengan menguraikan solusi menjadi sekumpulan langkah (*step*) atau tahapan (*stage*) sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan. Pada pemrograman dinamis, rangkaian keputusan yang optimal dibuat menggunakan prinsip optimalitas. Prinsip Optimalitas mengandung ide bahwa jika solusi total optimal, maka bagian solusi sampai tahap ke-*k* juga optimal. Oleh karena itu, *Dynamic Programming* merupakan salah satu cara penanganan kasus *Knapsack* yang membutuhkan optimalisasi hasil.

Genetic Algorithm (GA) adalah teknik pencarian di dalam ilmu komputer untuk menemukan penyelesaian perkiraan untuk optimisasi dan masalah pencarian. GA merupakan kelas khusus dari algoritma *evolusioner* menggunakan teknik yang terinspirasi oleh biologi evolusioner seperti warisan, mutasi, seleksi alam dan rekombinasi (atau *crossover*). Namun karena penerapan GA dalam penyelesaian *Knapsack Problem* secara *Heuristik* yakni dengan dengan bilangan random yang bersifat *probabilistik*. Solusi yang didapatkan oleh GA dalam beberapa kasus persoalan belum menemukan hasil yang optimum dibanding dengan algoritma pencari lain. Namun hal inilah yang menjadi peluang banyak peneliti untuk melakukan penelitian untuk sejumlah persoalan dengan GA.

2.4 Algoritma Genetika

GA merupakan suatu metode pencarian heuristik yang dikembangkan berdasarkan gagasan evolusi seleksi alamiah pada Teori Darwin (Sutojo dkk, 2011 dan Zukhri, 2014). GA ini pertama kali dikenalkan oleh John Holland (1975) pada bukunya yang berjudul "*Adaptation in Natural and Artificial Systems*" (Purnomo, 2014) dan dikembangkan oleh muridnya David Goldberg (1989). Prinsip GA diinspirasi oleh prinsip yang ada pada teori genetika pada ilmu biologi sehingga proses yang terjadi pada GA sama dengan proses yang terjadi pada evolusi biologis (Suyanto, 2005).

GA adalah algoritma yang berusaha menerapkan pemahaman mengenai evolusi alamiah pada tugas-tugas pemecahan-masalah (*problem solving*). Pendekatan yang diambil oleh algoritma ini adalah dengan menggabungkan secara acak berbagai pilihan solusi terbaik dalam suatu kumpulan, untuk mendapatkan generasi solusi terbaik berikutnya yaitu pada suatu kondisi yang memaksimalkan nilai solusi individu atau nilai *fitness*. Generasi ini akan merepresentasikan perbaikan-perbaikan pada populasi awalnya, dengan melakukan proses ini secara berulang, algoritma ini diharapkan dapat mensimulasikan proses evolusioner. Pada akhirnya, akan didapatkan solusi-solusi yang paling tepat bagi permasalahan yang dihadapi. Untuk menggunakan GA, solusi permasalahan direpresentasikan sebagai kromosom. Tiga aspek yang penting untuk penggunaan GA:

1. Definisi fungsi *fitness*
2. Definisi dan implementasi representasi genetik
3. Definisi dan implementasi operasi genetik

Jika ketiga aspek di atas telah didefinisikan, GA akan bekerja dengan baik.

2.4.1 Struktur Umum GA

GA memberikan suatu pilihan bagi penentuan nilai parameter dengan meniru cara reproduksi genetik, pembentukan kromosom baru serta seleksi alami seperti yang terjadi pada makhluk hidup. Istilah dalam metode GA yang digunakan merupakan adaptasi istilah dalam proses evolusi biologi. Tabel 2.1 berikut menjelaskan istilah dalam metode GA.

Tabel 2.1 Istilah dalam metode GA

Individu/Kromosom	Kandidat solusi dari masalah yang akan diselesaikan
Populasi	Himpunan dari kromosom
Gen	Bagian dari kromosom (satu kromosom terdiri dari beberapa gen)
Orangtua (*Parent*)	Kromosom yang terpilih untuk proses reproduksi (baik itu crossover maupun mutation)
Anak (*Offspring*)	Keturunan dari hasil reproduksi
Fitness	Suatu nilai yang melambangkan kualitas suatu kromosom
Crossover	Proses reproduksi yang dilakukan dengan proses perkawinan silang antar dua kromosom
Mutasi	Proses reproduksi yang dilakukan dengan memodifikasi gen yang ada di dalam kromosom

GA merupakan algoritma stokastik. *Random* atau ketidakberaturan merupakan peran utama dalam GA. Pada proses *crossover* dan mutasi memerlukan suatu parameter yang dibangun secara acak. GA selalu memilih atau mempertimbangkan populasi solusi. Pada tiap tahap GA, populasi solusi selalu direkombinasi dan dievaluasi untuk mendapatkan solusi yang lebih baik (Malinda,2015). Pada aplikasinya, GA biasa digunakan untuk memperoleh solusi optimal ataupun solusi pendekatan dari persoalan optimasi (minimisasi atau maksimasi) yang mempunyai banyak sekali solusi yang mungkin (*feasible solution*).

Secara umum struktur dari suatu GA dapat didefinisikan dengan langkah-langkah sebagai berikut:

1. Membangkitkan populasi awal

Populasi awal ini dibangkitkan secara random sehingga didapatkan solusi awal. Populasi itu sendiri terdiri atas sejumlah kromosom yang merepresentasikan solusi yang diinginkan.

2. Membentuk generasi baru

Untuk membentuk generasi baru, digunakan operator reproduksi / seleksi, *crossover* dan mutasi. Proses ini dilakukan berulang-ulang sehingga didapatkan jumlah kromosom yang cukup untuk membentuk generasi baru dimana generasi baru ini merupakan representasi dari solusi baru. Generasi baru ini dikenal dengan istilah anak (*offspring*).

3. Evaluasi solusi

Pada tiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang dinamakan *fitness*. Nilai *fitness* suatu kromosom menggambarkan kualitas kromosom dalam populasi tersebut. Proses ini akan mengevaluasi setiap populasi dengan menghitung nilai *fitness* setiap kromosom dan mengevaluasinya sampai terpenuhi kriteria berhenti. Bila kriteria berhenti belum terpenuhi maka akan dibentuk lagi generasi baru. Beberapa kriteria berhenti sering digunakan antara lain: berhenti pada generasi tertentu, berhenti setelah dalam beberapa generasi berturut-turut didapatkan nilai *fitness* tertinggi tidak berubah, berhenti dalam *n* generasi tidak didapatkan nilai *fitness* yang lebih tinggi, dan berhenti dalam batasan waktu tertentu.

2.4.2 Pengkodean

Pengkodean adalah suatu teknik untuk menyatakan populasi awal sebagai calon solusi suatu masalah ke dalam suatu kromosom. Berdasarkan jenis simbol yang digunakan sebagai nilai suatu gen, metode pengkodean dapat diklasifikasikan sebagai berikut

1. Pengkodean biner merupakan cara pengkodean yang paling umum digunakan karena pengkodean biner yang pertama kali digunakan dalam algoritma genetik oleh Holland. Keuntungan pengkodean ini adalah sederhana untuk diciptakan dan mudah dimanipulasi. Pengkodean biner memberikan banyak kemungkinan untuk kromosom walaupun dengan jumlah nilai-nilai yang mungkin terjadi pada suatu gen yang sedikit (0 dan 1). Di pihak lain, pengkodean biner sering tidak sesuai untuk banyak masalah dan terkadang pengoreksian harus dilakukan setelah operasi *crossover* dan mutasi.

2. Pengkodean bilang riil adalah suatu pengkodean bilangan dalam bentuk riil. Masalah optimalisasi fungsi dan optimalisasi kendala lebih tepat jika diselesaikan dengan pengkodean bilangan riil. Pengkodean bilangan riil memiliki struktur topologi ruang genotif identik dengan ruang fenotifnya, sehingga mudah membentuk operator genetik yang efektif dengan cara memakai teknik yang dapat digunakan yang berasal dari metode konvensional.

3. Pengkodean bilangan bulat adalah metode yang mengkodekan bilangan dalam bentuk bilangan bulat. Pengkodean ini baik digunakan untuk masalah optimisasi kombinatorial.

4. Pengkodean struktur data adalah model pengkodean yang menggunakan struktur data. Pengkodean ini digunakan untuk masalah kehidupan yang lebih kompleks seperti perencanaan jalur robot, dan masalah pewarnaan *Graph*.

2.4.3 Operator Genetika

GA merupakan proses pencarian yang heuristik dan acak sehingga penekanan pemilihan operator yang digunakan sangat menentukan keberhasilan GA dalam menemukan solusi optimum suatu masalah yang diberikan. Hal yang harus diperhatikan adalah menghindari terjadinya konvergensi *premature*, yaitu mencapai solusi optimum yang belum waktunya, dalam arti bahwa solusi yang diperoleh adalah hasil optimum lokal. Operator genetik yang digunakan setelah proses evaluasi tahap pertama membentuk populasi baru dari generasi sekarang. Operator-operator tersebut adalah operator seleksi, *crossover* dan mutasi.

2.4.3.1 Seleksi

Seleksi bertujuan memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling *fit*. Langkah pertama dalam seleksi ini adalah pencarian nilai *fitness*. Masing-masing individu dalam suatu ruang seleksi akan menerima probabilitas reproduksi yang senilai pada nilai objektif dirinya sendiri terhadap nilai objektif dari semua individu. Nilai *fitness* inilah yang nantinya akan digunakan pada tahap seleksi berikutnya (Kusumadewi, 2003).

Proses seleksi dilakukan secara *random,* sehingga tidak ada jaminan bahwa suatu indvidu yang bernilai *fitness* tertinggi akan selalu terpilih. Walaupun individu bernilai fitness tertinggi terpilih, mungkin saja individu tersebut akan rusak (nilai

fitnessnya menurun) karena proses pindah silang (*crossover*). Oleh karena itu, untuk menjaga agar individu bernilai *fitness* tertinggi tersebut tidak hilang selama evolusi, maka perlu dibuat satu atau beberapa *copy*-nya. Prosedure ini dikenal sebagai elitisme. *Elitism* merupakan suatu prosedur untuk menduplikat kromosom yang memiliki nilai *fitness* terbaik. Tujuan *elitism* adalah agar kromosom dengan nilai terbaik akan tetap terpilih dan tidak mengalami kerusakan (Sutojo dkk, 2011). Beberapa metode seleksi yang terkenal ialah *Roulette Wheel Selection*, *Rank Base Selection* dan *Tournament Selection* (Syarif, 2014). Pada proses ini individu yang diseleksi berdasarkan pada nilai *fitness.* Individu-individu yang lolos membentuk generasi baru sebanyak populasi awal

Kemampuan GA untuk memproduksi kromosom yang lebih baik secara progresif dipengaruhi pada penekanan selektif yang diterapkan ke populasi. Penekanan selektif dapat diterapkan dalam dua cara. Cara pertama adalah membuat lebih banyak kromosom anak yang dipelihara dalam populasi dan memilih hanya kromosom terbaik bagi generasi berikut. Walaupun orang tua dipilih secara acak, metode ini akan terus menghasilkan kromosom yang lebih baik berhubungan dengan penekanan selektif yang diterapkan pada individu anak tersebut.

Cara lain menerapkan penekanan selektif adalah memilih orang tua yang lebih baik ketika membuat keturunan baru. Metode ini, hanya kromosom sebanyak yang dipelihara dalam populasi yang perlu dibuat bagi generasi berikutnya. Walaupun penekanan selektif tidak diterapkan ke level keturunan, metode ini akan terus menghasilkan kromosom yang lebih baik, karena adanya penekanan selektif yang diterapkan ke orangtua.

2.4.3.2 Pindah silang (*Crossover)*

Pindah silang (*Crossover*) adalah operator dari GA yang melibatkan dua induk untuk membentuk kromosom baru, yang bertujuan menambah keanekaragaman *string* dalam populasi dengan penyilangan antar-*string* yang diperoleh dari kromosom orang tua (*parent*) sebelumnya. Pindah silang menghasilkan titik baru dalam ruang pencarian yang siap untuk diuji. Operasi ini tidak selalu dilakukan pada semua individu yang ada. Proses *crossover* dilakukan pada setiap individu dengan probabilitas *crossover* yang ditentukan. Prosedur pemilihan individu yang akan melalui proses *crossover* adalah dengan dibangkitkan bilangan random 0 < 1 sebanyak ukuran populasi, nilai probabilitas yang kurang dari sama dengan nilai P_c, dipilih sebagai kromosom orang tua (*parent*) untuk menghasilkan kromsom keturunan (*offspring*). Terdapat beberapa jenis *crossover* diantaranya sebagai berikut:

1. *Crossover* 1-titik

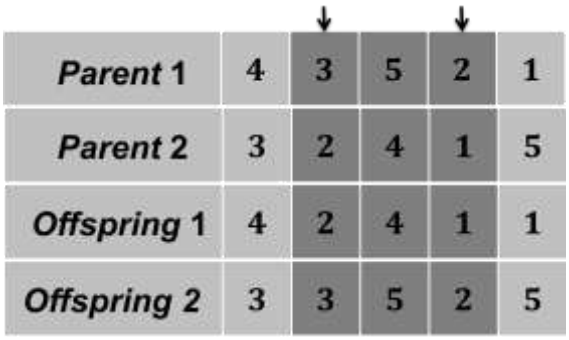
Proses pada *Crossover* 1-titik dilakukan dengan memisahkan *string* kromosom menjadi dua bagian, selanjutnya salah satu bagian dipertukarkan dengan salah satu bagian dari string yang lain yang telah dipisahkan dengan cara yang sama. Proses yang demikian dinamakan operator *crossover* satu titik seperti berikut:

			↓		
Parent 1	4	3	5	2	1
Parent 2	3	2	4	1	5
Offspring 1	4	3	4	2	1
Offspring 2	3	2	5	1	5

Gambar 2.1 Contoh *Crossover* 1-titik

2. *Crossover* 2-titik

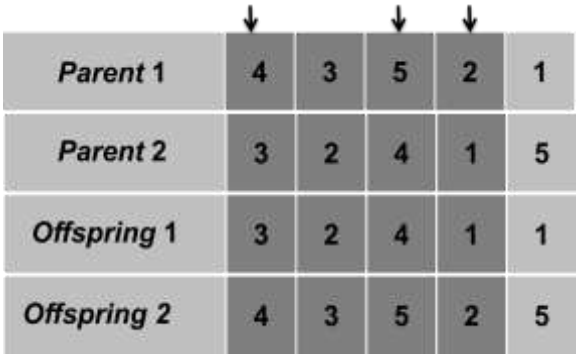
Prosedur *crossover* 2 titik (*Two-point Crossover*) diawali dengan ditentukan dua titik *crossover* secara acak, kemudian gen diantara kedua titik *crossover* tersebut di tukar / pindah silang antar kromosom orang tua (*parent*) yang terpilih. Selebihnya kromosom anak (*offspring*) disalin dari kromosom orang tua (*parent*). Berikut contoh prosedur *Two-point Crossover* :



Gambar 2.2 Contoh *Crossover* 2-titik

3. *Crossover n*- Titik

Pada *crossover* banyak titik, *m* posisi penyilangan k_i =(k=1,2,...*n*-1, i=1,2...,*m*) dengan *n*= panjang kromosom diseleksi secara random dan tidak diperbolehkan ada posisi yang sama, serta diurutkan naik. Variabel - variabel ditukar antar kromosom pada titik tersebut untuk menghasilkan kromosom anak (*offspring)*.



Gambar 2.3 Contoh *Crossover n*-titik

2.4.3.3 Mutasi

Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom. Proses mutasi dalam sistem biologi berlangsung dengan mengubah isi *allele* gen pada suatu *locus* dengan *allele* yang lain. Proses mutasi ini bersifat acak sehingga tidak selalu menjamin bahwa setelah proses mutasi akan diperoleh kromosom dengan *fitness* yang lebih baik. Operator mutasi merupakan operasi yang menyangkut satu kromosom tertentu. Prosedur pemilihan individu yang akan melalui proses mutasi adalah dengan dibangkitkan bilangan random 0 < 1 sebanyak ukuran populasi, nilai probabilitas yang kurang dari sama dengan nilai P_m maka dipilih sebagai kromosom orang tua (*parent*) untuk menghasilkan kromosom keturunan (*offspring*). Beberapa cara operasi mutasi diterapkan dalam GA antara lain *Flip Mutation* (Metode Penggantian), *Swap Mutation* (Metode Penukaran), *Inversion Mutation* (Metode Pembalikan), *Insertion Mutation* (Metode Penyisipan). Berikut contoh metode mutasi :

1. Metode pembalikan (*Inversion Mutasi*)

Metode ini dilakukan dengan mengambil substring secara acak yang terletak diantara dua titik pada kromosom, selanjutnya substring dilakukan *invers*. Ilustrasi terdapat pada Gambar 2.4.

Parent	4	3	5	2	1
Offspring	4	2	5	3	1

Gambar 2.4 Ilustrasi Metode Pembalikan

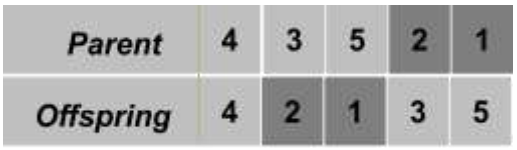
2. Metode Penyisipan (*Insertion Mutation*)

Metode ini akan memilih satu gen yang akan disisipkan pada posisi yang dipilih secara acak. Ilustrasi pada Gambar 2.5.

Gambar 2.5 Ilustrasi Metode Penyisipan

3. Metode Pemindahan (*Displacement Mutation*)

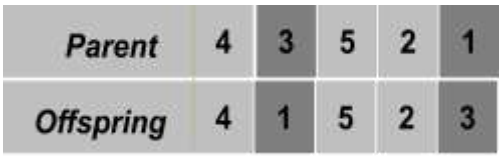
Metode ini akan mengambil dua gen yang akan disisipkan pada posisi yang dipilih secara acak. Ilustrasi metode ini pada Gambar 2.6.



Gambar 2.6 Ilustrasi Metode Pemindahan

4. Metode Penukaran (*Swap Mutation*)

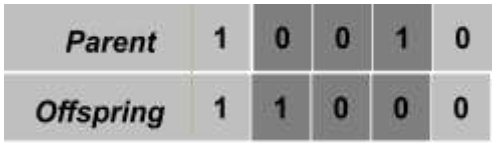
Metode ini memilih dua gen secara acak, selanjutnya dua gen akan saling dipertukarkan. Metode ini diilustrasikan pada Gambar 2.7.



Gambar 2.7 Ilustrasi Metode Penukaran

5. Metode Penggantian (*Flip Mutation*)

Metode ini mengganti nilai gen yang dipilih, apabila gen bernilai satu akan diganti menjadi nol begitu pula sebaliknya. Ilustrasi metode ini pada Gambar 2.8.



Gambar 2.8 Ilustrasi Metode Penggantian (Syarif, 2014)

2.4.4 Parameter Genetik

Pengoperasian GA dibutuhkan 4 parameter (Juniawati, 2003) sebagai berikut :

1. Probabilitas Persilangan (*Crossover Probability*)

   Menunjukkan kemungkinan *crossover* terjadi antara 2 kromosom. Jika tidak terjadi *crossover* maka keturunannya akan sama persis dengan kromosom orangtua, tetapi tidak berarti generasi yang baru akan sama persis dengan generasi yang lama. Jika probabilitas *crossover* 100% maka semua keturunannya dihasilkan dari *crossover*. *Crossover* dilakukan dengan harapan bahwa kromosom yang baru akan lebih baik.

2. Probabilitas Mutasi (*Mutation Probability*)

   Menunjukkan kemungkinan mutasi terjadi pada gen-gen yag menyusun sebuah kromosom. Jika tidak terjadi mutasi maka keturunan yang dihasilkan setelah *crossover* tidak berubah. Jika terjadi mutasi bagian kromosom akan berubah. Jika probabilitas 100%, semua kromosom dimutasi. Jika probabilitasnya 0%, tidak ada yang mengalami mutasi.

3. Jumlah Individu

   Menunjukkan jumlah kromosom yang terdapat dalam populasi (dalam satu generasi). Jika hanya sedikit kromosom dalam populasi maka GA akan mempunyai sedikit variasi kemungkinan untuk melakukan *crossover* antara orangtua karena hanya sebagian kecil dari *search space* yang dipakai. Sebaliknya jika terlalu banyak maka GA akan berjalan lambat.

4. Jumlah Populasi

   Menentukan jumlah populasi atau banyaknya generasi yang dihasilkan, digunakan sebagai batas akhir proses seleksi, persilangan dan mutasi.

2.5 Pencarian optimum lokal (*Local Search*)

Pencarian optimum lokal (*Local Search*) merupakan metode tambahan yang dilakukan dalam proses pencarian nilai optimum menggunakan GA, atau dikenal dengan GA *Hybrid*. GA *hybrid* merupakan kombinasi metode-metode heuristik lain ke dalam GA dengan harapan mampu meningkatkan kinerja GA (Syarif, Wamiliana & Junaidi. 2007). GA hybrid merupakan gabungan dari GA dengan pencarian lokal (local search) yaitu *best improvement search* untuk menghasilkan solusi yang lebih optimal. Pada prinsipnya hibridisasi ini diharapkan mampu memberikan solusi lain yang lebih baik disekitar lokal optimum yang diberikan oleh GA atau dikenal dengan istilah (*local search*).

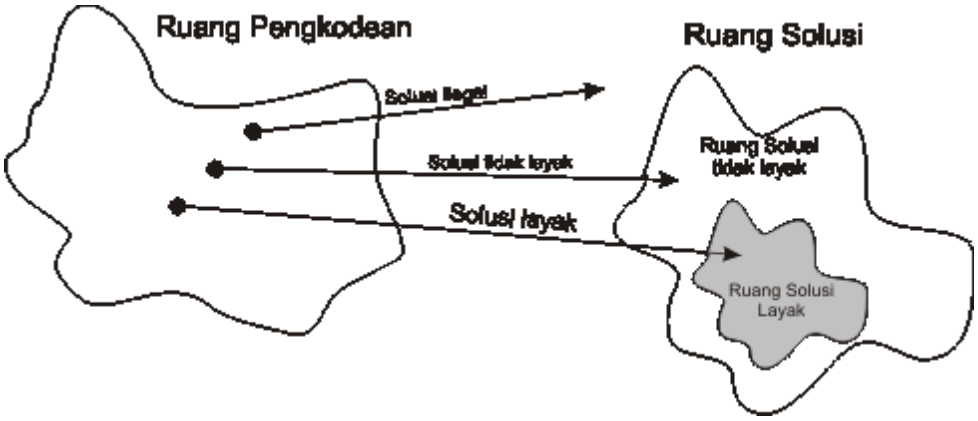
Local search (LS) merupakan salah satu metode dalam *metaheuristic method*. Metode ini dimulai dengan memberikan *initial solution,* pada setiap iterasi *heuristic* mengubah solusi dibagian akhir berdasarkan *neighbour*. Metode *local search* yang diterapkan dalam penelitian ini adalah dengan mengambil 10 gen terbaik (memiliki nilai optimum tertinggi) di generasi baru yang merupakan gen hasil seleksi dari proses *crossover*, mutasi dan perbaikan. Proses *local search* dari gen yang terpilih tersebut adalah dengan mengubah tepat satu isi *allele* gen pada suatu *locus* dengan *allele* yang lain yang secara acak. Gambar 2.9 menampilkan contoh perubahan gen dalam proses *Local Search*.

Parent	4	3	5	2	1
Offspring	4	2	1	3	1

Gambar 2.9 *Local Search*

2.6 Strategi Penanganan Kendala

Persoalan optimasi memiliki fungsi pembatas. Fungsi pembatas tersebut akan membagi kromosom menjadi beberapa bagian, yaitu kromosom layak, kromosom tidak layak, dan kromosom ilegal. Kromosom layak merupakan kromosom yang memenuhi fungsi persoalan. Kromosom tidak layak merupakan kromosom yang berada di luar ruang kelayakan.



Gambar 2.10 Ruang Kelayakan Solusi

Kromosom yang tidak layak biasanya ditemukan pada persoalan optimasi berkendala yang direpresentasikan dengan persamaan atau tidak persamaan. Kendala pada persoalan membuat ruang tidak layak. Pada persoalan optimisasi berkendala, solusi maksimum biasanya berada diantara ruang solusi layak dan tidak layak. Kromosom ilegal merupakan kromosom yang tidak merepresentasikan solusi persoalan. Kromosom ilegal biasanya bermula dari tenik operasi genetika (*crossover* atau mutasi) yang digunakan. Operasi genetika terkadang akan menghasilkan *offspring* yang ilegal (Gen dan Cheng, 2000). Kromosom yang tidak layak dan ilegal perlu dilakukan evaluasi dan perbaikan.

Ada beberapa metode yang digunakan untuk menangani fungsi pembatas tersebut. Metode tersebut diantaranya adalah sebagai berikut:

a. Penolakan Kromosom (*Rejecting*)

Metode ini akan membuang kromosom yang tidak layak. Kelemahan dari metode ini adalah saat individu yang dihasilkan pada generasi pertama tidak layak semua, maka semua individu akan tertolak semua (Zukri, 2014).

b. Perbaikan Kromosom (*Repairing*)

Metode perbaikan akan membuat prosedur perbaikan untuk kromosom yang tidak layak atau ilegal. Metode perbaikan kromosom diklasifikasikan menjadi dua, yaitu prosedur yang hanya mengevaluasi kromosomnya tanpa merubah dan prosedur yang akan merubah kromosom menjadi kromosom layak. Letak kesulitan metode ini adalah membuat prosedur perbaikannya itu sendiri. Metode perbaikan biasa digunakan pada persoalan kombinatorial yang menghasilkan *offspring* ilegal. Metode akan mempertahankan kromosom yang ilegal dan merubah kromosom tersebut tersebut menjadi kromosom legal. Orvosh dan Davis (1994) telah membuktikan banyak penyelesaian persoalan kombinatorik relatif lebih mudah dengan perbaikan kromosom tidak layak atau ilegal dan metode perbaikan kromosom ini lebih baik dari metode penolakan dan pemberian penalti (Syarif, 2014 dan Zukri, 2014).

c. Modifikasi Operator Genetika

Metode ini akan menyediakan representasi kromosom khusus atau operasi genetika yang mampu menghasilkan kromosom yang layak. Metode ini akan selalu menjaga kromosom pada ruang solusi layak (Syarif, 2014).

d. Pemberian Penalti (*Penalty strategy*)

Metode dengan pemberian nilai penalti menganggap semua kromosom yang telah dibangkitkan akan memberi solusi layak. Metode ini biasanya diaplikasikan pada persoalan optimasi berkendala. Beberapa persoalan optimasi berkendala akan menghasilkan ruang tidak layak yang besar sehingga apabila pencarian hanya pada ruang layak, maka ruang pencarian akan menjadi terbatas. Oleh karena itu, ruang pencarian metode ini berada pada ruang tidak layak. Ada dua teknik untuk menghitung nilai fungsi evaluasi untuk metode pinalti, yaitu, teknik penambahan nilai pinalti pada fungsi tujuan atau objektif dan teknik pengalian fungsi tujuan dengan fungsi penalti (Syarif, 2014).

BAB III
METODE PENELITIAN

3.1 Waktu Dan Tempat Penelitian

Penelitian dilaksanaan di lingkungan Program Studi Ilmu Komputer Fakultas Matematika Dan Ilmu Pengetahuan Alam Universitas Lampung. Waktu penelitian dilakukan di Semester Genap tahun ajaran 2015/2016.

3.2 Lingkungan Pengembangan

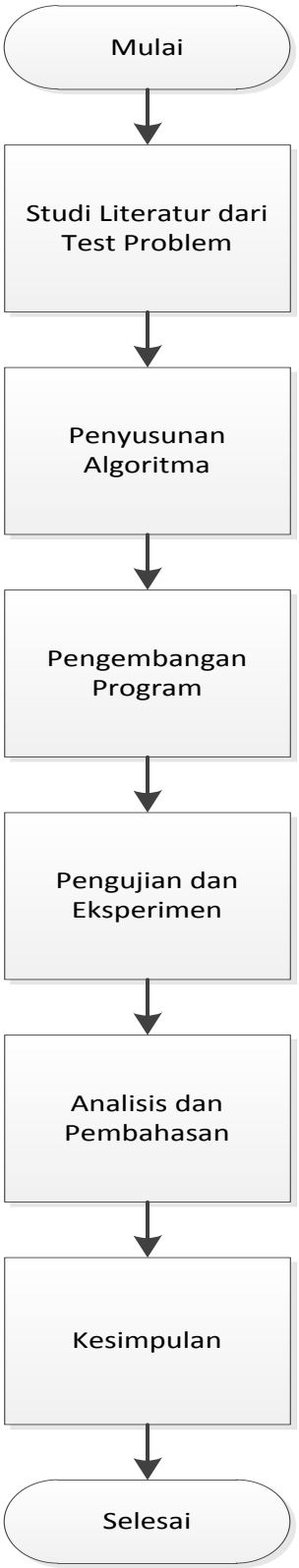
Lingkungan pengembangan yang akan digunakan pada penelitian ini adalah sebagai berikut:

a. *Software* : Ubuntu 14.04 LTS, Matlab R2016a

b. *Hardware* : PC HP (Processor intel ® Core ™ i5-3470S CPU @2.90 GH_Z x 4, RAM 4 GB

3.3 Tahapan Penelitian

Tahapan penelitian yang akan dilakukan terdapat beberapa langkah yaitu studi literatur, penyusunan algoritma, pengembangan program, pengujian dan eksperimen, analisis dan pembahasan dan kesimpulan. Tahapan penelitian terdapat pada Gambar 3.1.



Gambar 3.1 Alur Penelitian

3.3.1 Tahap Pertama : Studi Literatur

Penelitian persoalan *Knapsack* telah banyak dilakukan untuk menemukan solusi optimal. Beberapa metode telah diaplikasikan. Beberapa penelitian persoalan *Knapsack* yang telah dilakukan diantaranya sebagai berikut :

a. **Performance Analysis of Genetic Algorithm for Solving the Multiple-Choice Multi-Dimensional Knapsack Problem. (Moinur, Ahmed, 2009)**

   Penelitian ini menyajikan *Genetic Algorithm* (GA) untuk memecahkan *Multiple-choice Multi-dimensional Knapsack Problem* (mmKP), sebuah permasalahan optimisasi kombinatorial *NP-Hard*. Performa dari GA telah diuji menggunakan Benchmark *Problem Instances*. Penelitian diawali dengan inisiasi kromosom yang direpresentasikan ke dalam tiga-*dimensional array* yakni ([ukuran kromosom],[nomor grup], [no item]). Selanjutnya ditentukan nilai *fitness*, seleksi grup, penyilangan kromosom (crossover), mutasi, dan seleksi elitism. Hasil perbandingan dengan algoritma lain menunjukkan performa GA bukanlah algoritma yang terbaik untuk permasalahan mmKP.

b. **Solving the Multi-dimensional Multi-choice Knapsack Problem with the Help of Ants. (Iqbal, Bari dan Sohel, 2010)**

   Penelitian ini menggunakan *Ant Colony Optimization* (ACO) untuk mencari solusi optimal pada permasalahan mmKP. Ide dasar dari ACO adalah untuk memodelkan permasalahan pencarian, ketika M*inimum Cost Path* dalam sebuah graph dicari, kecerdasan *ants* mencari *path* terbaik. Penelitian menunjukkan bahwa ACO menjadi algoritma yang terbaik dalam kualitas

pencarian solusi dan solusi optimal terdekat dengan waktu yang cepat, setelah dibandingkan dengan algoritma lainnya. mmKP merupakan sebuah permasalahan optimasi diskret yang merupakan variasi dari permasalahan 0/1 Knapsack dan permasalahan *NP-hard*.

c. A Fast and Scalable Multi-Dimensional Multiple-Choice Knapsack Heuristic. (Shojaei, Basten, Geilen dan Davoodi, 2013)

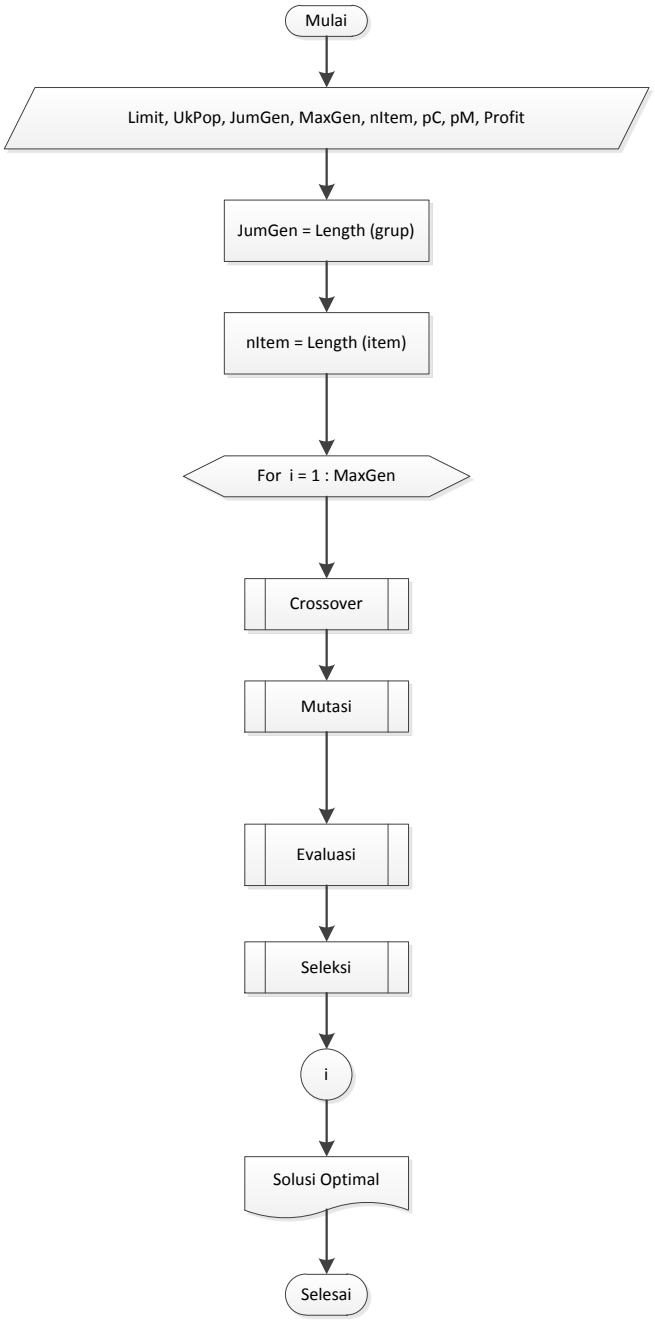
Penelitian ini membahas tentang M*ultidimensional Knapsack Problem*, menyajikan beberapa teori dan kesimpulan mengenai struktur dan mengevaluasi perbedaan *Integer Linear Programming* (ILP) berdasarkan metaheuristik dan kolaborasi keduanya.

d. Genetic Algorithms: Concepts, Design for Optimization of Process Controllers. (Malhotra, Singh dan Singh, 2011)

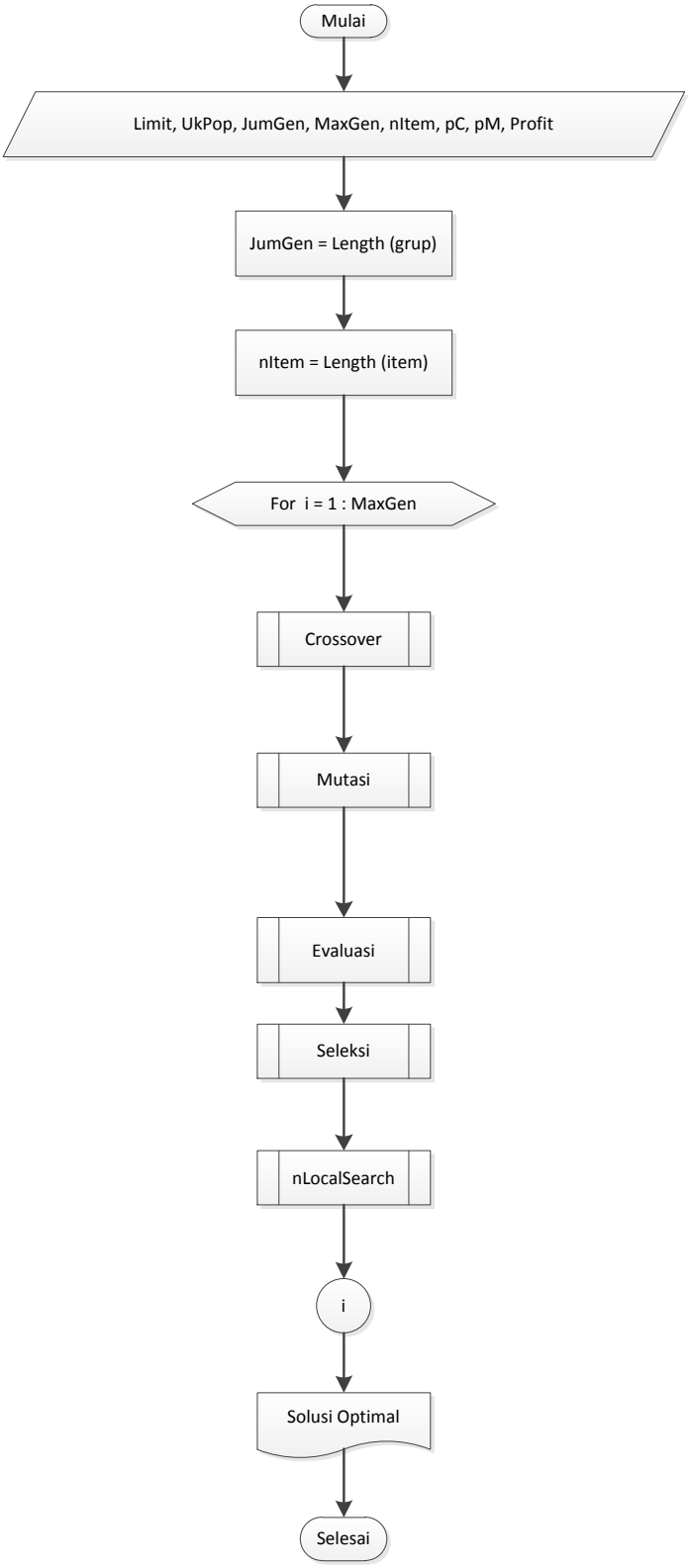
Penelitian ini membahas konsep dan desain prosedur GA sebagai alat optimasi menggunakan operator alami. GA diterapkan untuk kontrol torsi langsung motor induksi, kontrol kecepatan turbin gas, kontrol kecepatan motor DC servo untuk optimalisasi parameter control Simulasi dilakukan dalam paket Simulink MATLAB. Hasil simulasi menunjukkan optimasi yang lebih baik dari pengendali algoritma genetika hybrid dari *Fuzzy standalone* dan pengendali konvensional.

3.3.2 Tahap Kedua : Penyusunan Algoritma

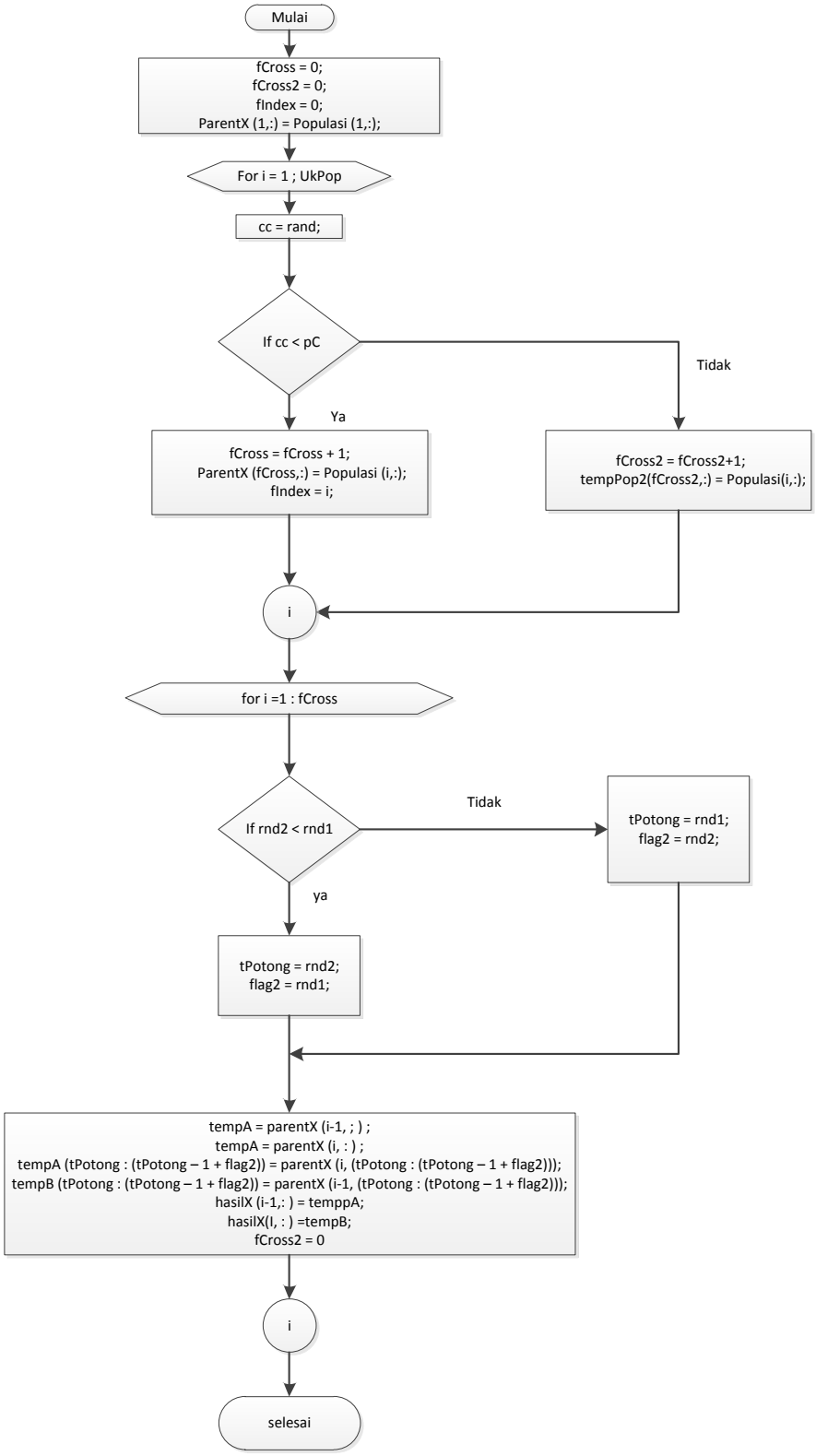
Algoritma yang dipakai dalam penelitian ini akan ditunjukkan pada *Flowchart* sebagai berikut :



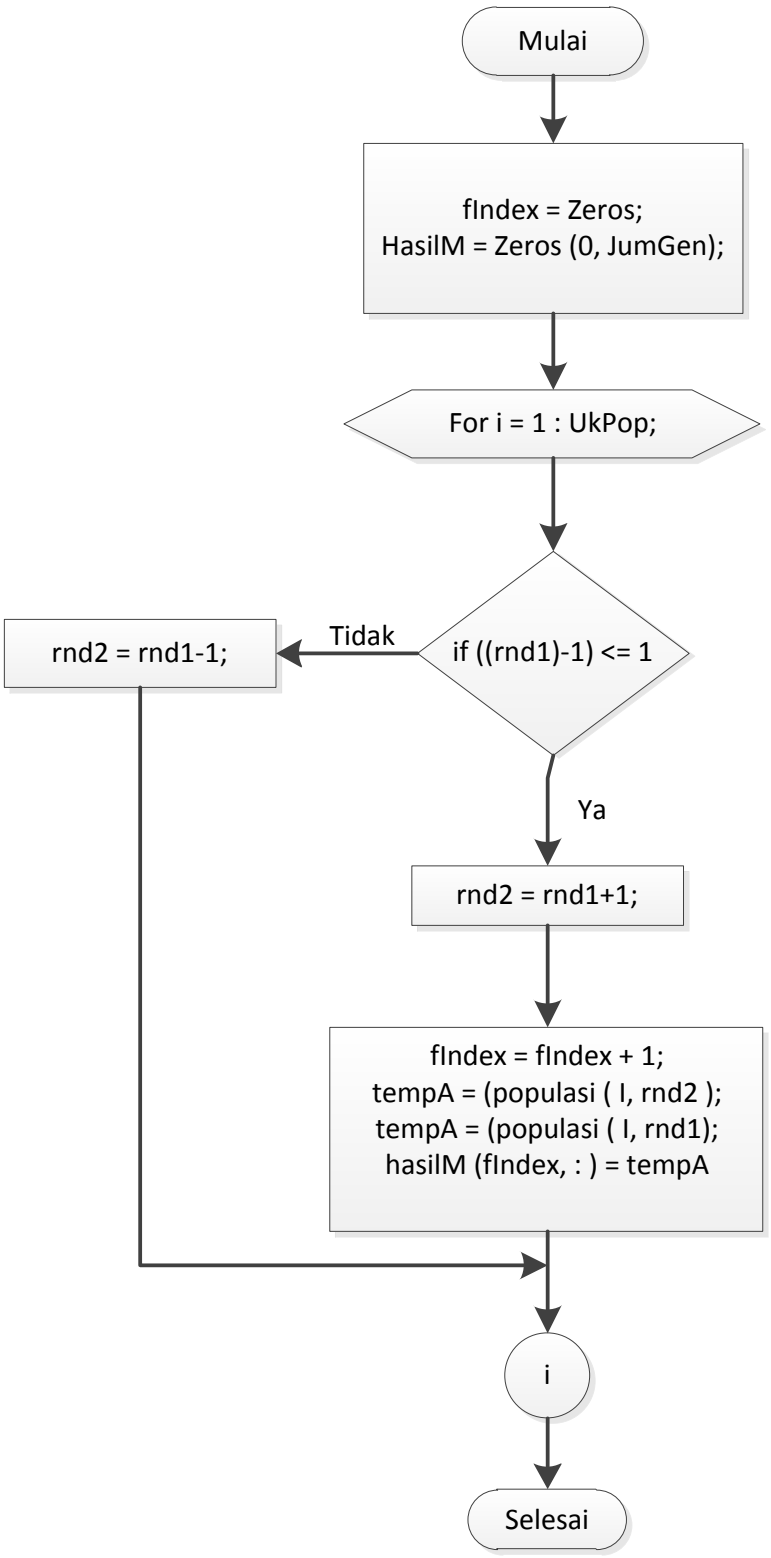
Gambar 3.2 Diagram Alir *r*GA



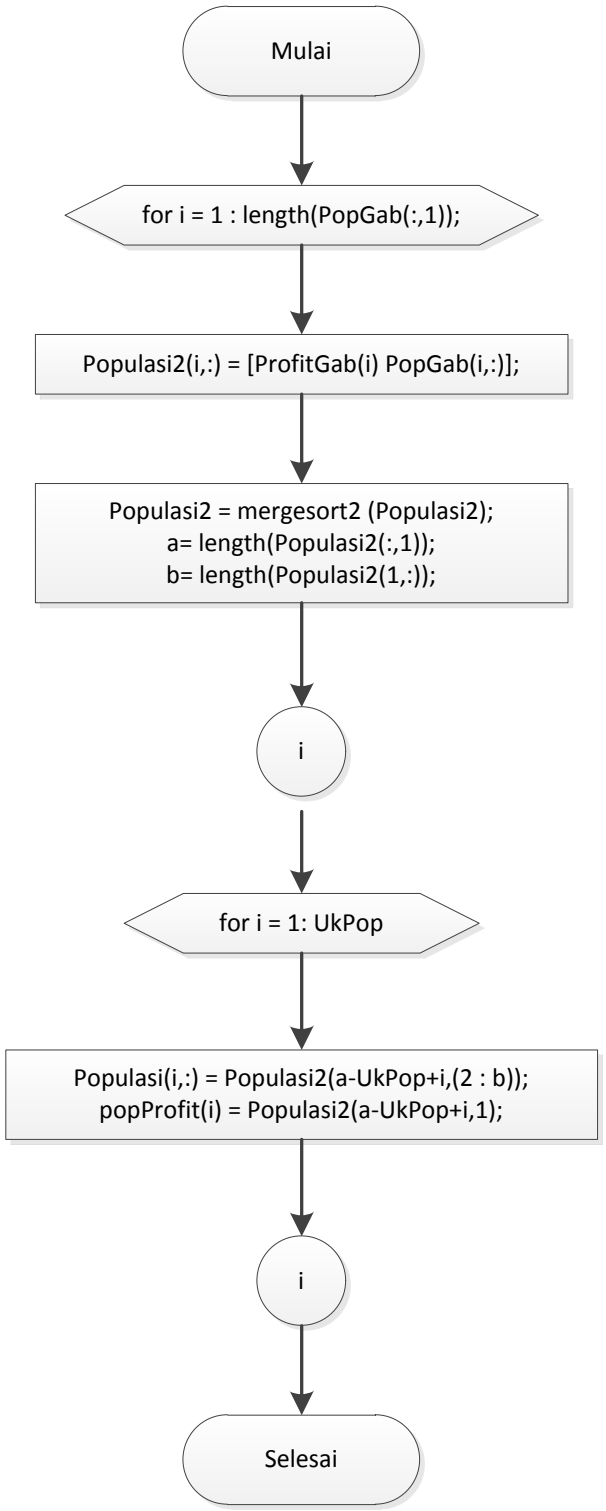
Gambar 3.3 Diagram Alir *hrGA*



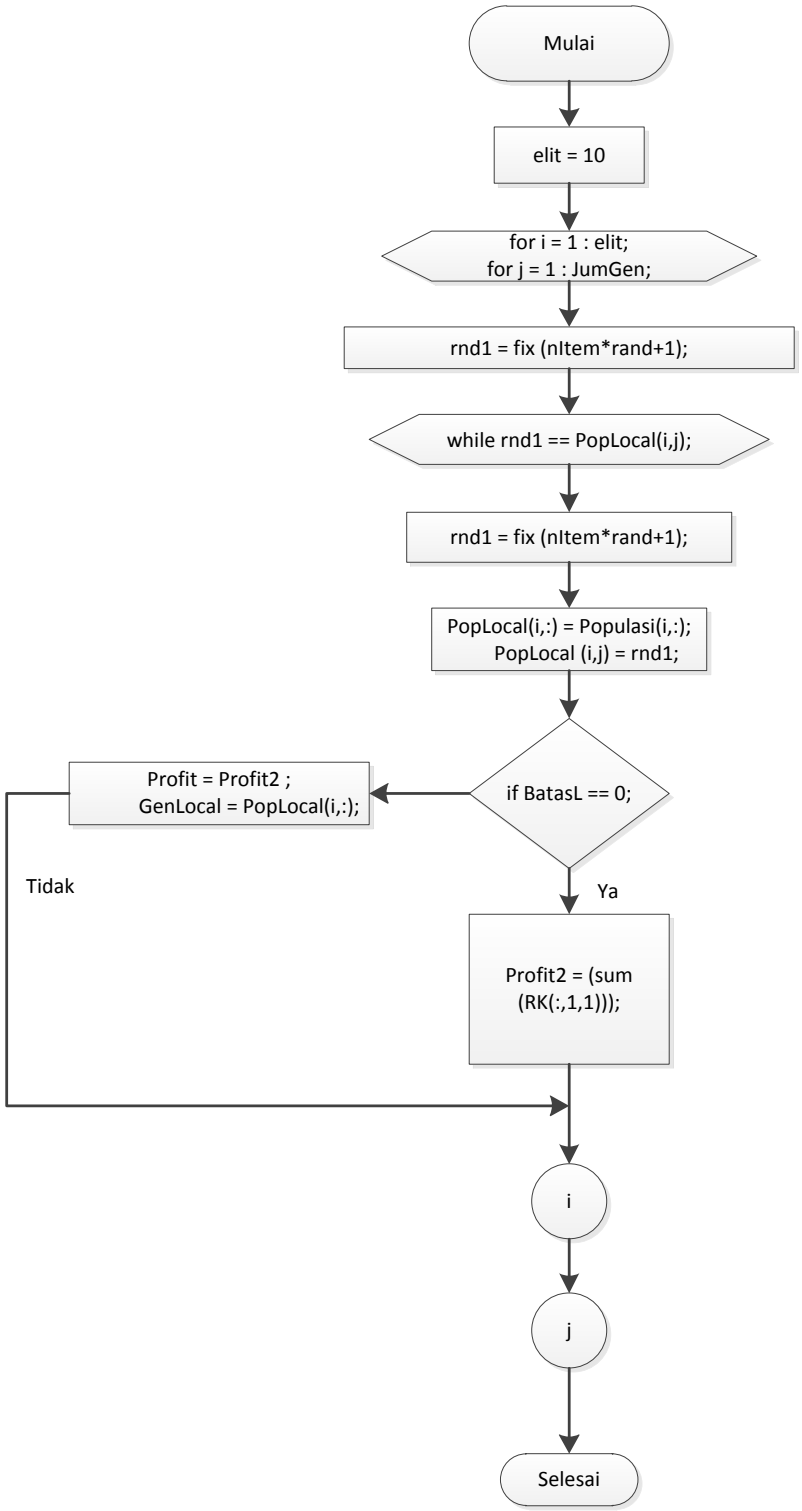
Gambar 3.4 Diagram Alir *Crossover*



Gambar 3.5 Diagram Alir Mutasi



Gambar 3.6 Diagram Alir *Merge-Sort*



Gambar 3.7 Diagram Alir *Local Search*

3.3.3 Tahap Ketiga : Pengembangan Program

3.3.3.1 Representasi Kromosom

Kromosom adalah kumpulan dari gen-gen. Untuk persoalan mmKP dalam penelitian ini, jumlah gen dalam kromosom sama dengan jumlah barang (*item- j*) yang di setiap kelompok (*class-i*). Langkah awal adalah dengan dibangkitkan kromosom-kromosom yang berada pada ruang solusi layak (*feasible*), sebanyak ukuran populasi yang diinginkan. Ukuran populasi berbanding lurus dengan jumlah barang (*item*) di setiap kelompok (*class*). Semakin banyak jumlah kelompok (*class*) maka semakin banyak ukuran populasi yang ditentukan. Berikut algoritma pembangkitan kromosom random.

```
function populasi =
InisiasiPopulasi2(UkPop,JumGen,nItem)

populasi = fix(nItem*rand(UkPop,JumGen)+1);
```

Sebagai contoh, anggap bahwa kromosom berikut merupakan salah satu kromosom dari *n* ukuran populasi yang dibangkitkan melalui proses pembangkitan kromosom diatas, pada Gambar 3.8 berikut :

4	3	5	4	1

Gambar 3.8 Kromosom

3.3.3.2 *Decode*

Proses ini ditandai dengan menghitung nilai solusi (*profit*) dan kebutuhan sumber daya barang (*resource item*) setiap gen agar tidak melebihi batasan sumber daya (*resource constraint*) setiap dataset. Dibawah ini adalah representasi dataset Instances 01 (I01). I01 memiliki 5 *class* (*i*), 5 *item* (*j*), dan 5 *resources* (*k*) untuk setiap barang-*j di* setiap *class-i,* dan batasan setiap kebutuhan sumber daya (R_k) sebesar 25.

I01					
	5	5	5		
	25	25	25	25	25
1					
7.00	1	3	1	1	6
17.00	1	4	9	9	3
25.00	4	3	9	8	2
35.00	4	5	8	0	6
36.00	6	8	3	0	7
2					
9.00	0	0	4	4	2
10.00	0	0	1	8	7
10.00	1	1	6	0	6
39.00	9	1	2	2	4
44.00	8	7	0	8	2
3					
15.00	2	0	5	5	5
19.00	2	3	2	6	2
20.00	3	1	6	4	7
44.00	6	7	5	6	9
50.00	9	5	9	2	2
4					
5.00	0	1	3	8	0
25.00	2	2	7	0	8
32.00	5	5	6	1	9
37.00	6	3	6	9	1
37.00	7	9	7	2	3
5					
24.00	4	0	7	0	2
30.00	4	8	9	0	0
32.00	5	2	7	2	0
43.00	5	5	9	5	2
44.00	9	2	2	2	3

Keterangan :

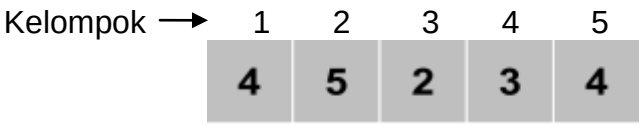
: *class/kelas* (*i*)

: nilai barang*/value item* (*j)*

: kebutuhan/resource (*k*)

: Batasan kebutuhan/ *resource constraint*(R_k)

Mekanismenya adalah dengan memilih satu item di setiap kelas, dimana nilai total r_{ijk} terpilih, tidak melebihi nilai $R_{k.}$ Selanjutnya batasan ini akan menjadi nilai *fitness* yang menjadi tolak ukur layak atau tidak layak suatu kromosom random yang dibangkitkan. Berikut contoh satu kromosom yang dibangkitkan random serta representasinya dalam dataset I01.



Gambar 3.9 *Decoding* Kromosom

Indeks *locus* pada kromosom merepresentasikan indeks *class* dalam dataset, dan nilai *allele* dalam gen merepresentasikan indeks *item* di setiap *class*. Barang (*item*) terpilih dari contoh kromosom tersebut adalah : *item* ke 4 di *class* ke-1, *item* ke 5 di *class* ke-2, *item* ke 3 di *class* ke-3, *item* ke 2 di *class* ke-4, *item* ke 4 di *class* ke-5. Perhitungan jumlah *resource consumption* (r_{ijk}) setiap *item*, dimana total nilai r_{ijk} tidak melebihi nil*ai* R_k setiap *class*, ditampilkan pada Tabel 4.2 berikut :

Tabel 4.2. *Decoding* Kromosom.

	4	**5**	2	3	**4**
r_{14}	4	5	8	0	6
r_{25}	8	7	0	8	2
r_{32}	2	3	2	6	2
r_{43}	5	5	6	1	9
r_{54}	5	5	9	5	2
Rk	24	25	25	20	21

Kromosom yang memiliki nilai tidak lebih sama dengan R_k maka dinyatakan sebagai kromosom yang layak, dan akan mengikuti alur GA selanjutnya untuk seleksi nilai *profit* tertinggi. Berikut merupakan prosedur *decode* yang digunakan :

```
function RK = generateRK (i,JumGen,kk,Populasi)
for j = 1 : JumGen;
    RK(j,:,1) = (kk((Populasi(i,j)),:,j));
end;
function batas = cek_layak(RK,nItem,Limit)
batas = 0;
for h = 1 : nItem;
    sumK = (sum (RK(:,h+1,1)));
        if sumK > Limit
           batas = 1;
           break
        end;
end;
RK = generateRK (i,JumGen,kk,Populasi);
    BatasLimit = cek_layak(RK,nItem,Limit);
    if BatasLimit == 0;
        Profit = (sum (RK(:,1,1)));
        Status = 'Layak';
    else
       [Populasi, Profit] =
repair(nItem,JumGen,Populasi,RK,BatasL,Limit,kk,i);
        Status = 'Layak';
    end;
```

3.3.3.3 *Crossover*

Metode *crossover* dalam pengujian ini adalah *Two-point Crossover*. Prosedurnya adalah dengan melakukan pindah silang gen kromosom *parent* diantara dua titik *crossover* yang ditentukan secara random untuk menghasilkan kromosom *offspring*. Nilai *profit* yang diharapkan dari kromosom *offspring* hasil *crossover* dapat lebih optimum. Berikut merupakan prosedur *crossover* yang digunakan.

		↓		↓	
***Parent* 1**	4	3	5	2	1
***Parent* 2**	3	2	4	1	5
***Offspring* 1**	4	2	4	1	1
***Offspring* 2**	3	3	5	2	5

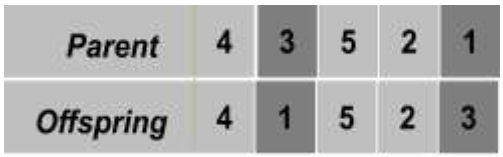
Gambar 3.10 Contoh *Crossover* 2-titik

Berikut merupakan potongan program proses *crossover* yang dilakukan pada penelitian :

```
function HasilX = xOver3(UkPop,JumGen,Populasi,pC)
ParentX (1,:) = Populasi (1,:);
for i = 1 : UkPop;
    cc = (rand);

    if cc < pC
       fCross = fCross + 1;
       ParentX (fCross,:) = Populasi (i,:);
       fIndex = i;
    end;
end;
if mod(fCross , 2) == 1;
    if fIndex == UkPop
        fIndex2 = fix ((rand)*(length(tempPop2(:,1)))+1);
        ParentX(fCross+1,:) = tempPop2(fIndex2,:);
    else
    ParentX(fCross+1,:) = Populasi(fIndex+1,:);
    fCross = fCross + 1;
    end;
end;
if mod(fCross , 2) == 1;
    if fIndex == UkPop
        fIndex2 = fix ((rand)*(length(tempPop2(:,1)))+1);
        ParentX(fCross+1,:) = tempPop2(fIndex2,:);
    else
    ParentX(fCross+1,:) = Populasi(fIndex+1,:);
    fCross = fCross + 1;
    end;
end;
fCross2 = zeros;
for i = 1 : fCross;
    fCross2 = fCross2+1;
    if fCross2 == 2;

rnd1 = int8((rand*JumGen));
rnd2 = int8((rand*JumGen));
        tempA = ParentX(i-1,:);
        tempB = ParentX(i,:);
        tempA(tPotong:(tPotong-1 +flag2)) =
      ParentX(i,(tPotong:(tPotong-1 +flag2)));
        tempB(tPotong:(tPotong-1 +flag2)) = ParentX(i-
1,(tPotong:(tPotong-1 +flag2)));
        HasilX(i-1,:)=tempA;
        HasilX(i,:)=tempB;
        fCross2 = 0;

    end;
end;
```

3.3.3.4 Mutasi

Metode mutasi dalam pengujian ini adalah *Swap Mutation*. Metode ini memilih dua gen secara acak, selanjutnya dua gen akan saling dipertukarkan. Berikut merupakan prosedur mutasi yang digunakan.



Gambar 3.11 Ilustrasi Metode Penukaran

```
function HasilM = Mutation_swap(UkPop,JumGen,Populasi,pM)
fIndex = zeros;
HasilM = zeros (0,JumGen);
for i = 1 : UkPop;
    cc = (rand);
    if cc < pM
       rnd1 = int8((rand*JumGen));
       rnd2 = int8((rand*JumGen));

       while rnd2 == rnd1
          rnd2 = int8((rand*JumGen));
       end;
      if rnd1 == 0;
         rnd1 = 1;
      end;
      if rnd2 == 0;
         rnd2 = 1;
      end;

    if (Populasi(i,rnd1)) == (Populasi(i,rnd2))
        if ((rnd1)-1) <= 1
            rnd2 = rnd1+1;
        else
            rnd2 = rnd1-1;
        end;
    end;
fIndex = fIndex + 1;
tempA = Populasi(i,:);
tempA (rnd1) = Populasi (i,rnd2);
tempA (rnd2) = Populasi (i,rnd1);
HasilM (fIndex,:) = tempA;
    end;
end;
```

3.3.3.5 Pencarian Optimum Lokal (*Local Search)*

Setelah kromosom melewati proses crossover dan mutasi, kromosom akan diurutkan secara *descending* (menurun) berdasarkan nilai optimum. Selanjutnya akan dipilih 10 kromosom terbaik berdasarkan nilai *profit* tertinggi untuk dilakukan proses *Local Search*, yakni dengan memilih satu gen kromosom secara random dan mengganti nilai gen tersebut secara random. Berikut prosedur *Local Search.*

Parent	4	3	5	2	1
Offspring	4	2	1	3	1

Gambar 3.12 *Local Search*

```
if Local_On == 1 ;
nLocal = 10;

PopLocal(1:elit,:) = Populasi (UkPop-elit+1:UkPop,:);
Profit = 0;

for i = 1 : elit;
for j = 1 : JumGen;
       rnd1 = fix (nItem*rand+1);
       while rnd1 == PopLocal(i,j);
           rnd1 = fix (nItem*rand+1);
       end;
       PopLocal(i,:) = Populasi(UkPop-elit+i,:);
       PopLocal (i,j) = rnd1;
       BatasL = cek_layak(RK,nItem,Limit);

       if BatasL == 0;
       Profit2 = (sum (RK(:,1,1)));
        if Profit2 >= Profit;           %TAG
            Profit = Profit2 ;
            GenLocal = PopLocal(i,:);
        end;
       else
       Profit2 = 0;
       PopLocal(i,:) = 0;
       end;
       a = [PopLocal(i,:) Profit2];

end;
```

3.3.3.6 Evaluasi (Strategi *Repair*)

Pada proses evaluasi dilakukan pemilihan. Meskipun kromosom awal dibangkitkan pada ruang solusi yang layak, namun setelah melewati proses mutasi dan *crossover* ada kemungkinan kromosom tersebut menjadi tidak layak. Oleh karena itu, setiap kromosom yang diketahui memiliki *resource* yang melebihi kapasitas R_k dilakukan perbaikan kromosom yang disebut dengan prosedur *Repairing Strategy*. Berikut merupakan algoritma yang digunakan untuk prosedur *Repairing Strategy*.

```
while BatasLimit > 0
    a = fix (nItem*rand+1);
    Populasi(i,a) = fix (nItem*rand+1);
    for g = 1 : JumGen;
        RK(g,:,1) = (kk((Populasi(i,g)),:,g));
    end;
    Profit = (sum (RK(:,1,1)));
    for h = 1 : nItem;
        sumRK = (sum (RK(:,h+1,1)));
            if sumRK <= Limit
                batas2 = batas2 + 1;
                if batas2 == nItem;
                    BatasLimit = 0;
                end;
            else
                BatasLimit = 1;
            end;
    end;
    batas2 = 0;
    a = fix (nItem*rand+1);
end;
```

Gambar 3.13 menampilkan ilustrasi strategi perbaikan kromosom pada alur penelitian

Perbaikan 1	4	4	3	1	3	Tidak Layak
Perbaikan 2	4	5	3	1	2	Tidak Layak
Perbaikan 3	4	5	3	1	3	Tidak Layak
Perbaikan n	4	5	3	2	4	Layak

Gambar 3.13 Ilustrasi perbaikan kromosom

3.3.4 Tahap Keempat : Pengujian dan Eksperimen

Pada tahap eksperimen, dilakukan tahap perancangan desain eksperimental yang akan dilakukan dalam penelitian. Dataset yang digunakan merupakan dataset *Benchmark* mmKp yang diunduh melalui alamat *website* kan yang diunduh dari alamat *website* *http://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/MMKP/.*

Dataset digolongkan menjadi 3 tipe berdasarkan jumlah *class* (*i*) yakni I01-I06 digolongkan ke dalam data kecil, I07-I09 data medium, dan I10-I13 digolongkan ke dalam data besar. Penentuan nilai parameter GA merupakan salah satu hal penting dalam strategi pencarian solusi yang baik. Probabilitas Crossover (*Pc*) yang digunakan dalam penelitian ini sebesar 0.4 dan Probabilitas Mutasi (*Pm*) sebesar 0.2. Berikut representasi dataset dan parameter GA yang digunakan dalam penelitian ini pada Tabel 3.1 :

Tabel 3.1 Dataset *Benchmnark* mmKP

No.	Dataset	Representasi Data				Parameter	
		i	*j*	*k*	R_k	*uk_pop*	*max_gen*
1	I01	5	5	5	25	50	1000
2	I02	10	5	5	50	50	1500
3	I03	15	10	10	75	50	2000
4	I04	20	10	10	100	100	1000
5	I05	25	10	10	125	100	1500
6	I06	30	10	10	150	100	2000
7	I07	100	10	10	500	200	1000
8	I08	150	10	10	750	200	1500
9	I09	200	10	10	1000	250	2000
10	I10	250	10	10	1250	300	1000
11	I11	300	10	10	1500	300	1500
12	I12	350	10	10	1750	450	2000
13	I13	400	10	10	2000	500	2000

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan dari penelitian ini adalah sebagai berikut :

1. Metode GA dengan strategi perbaikan kromosom memerlukan waktu yang lebih lama, namun hasil komputasi lebih unggul dibanding GA strategi penalti.
2. Metode *hr*GA memiliki nilai optimal lebih baik dibanding *r*GA.
3. Penentuan kombinasi parameter GA dapat mempengaruhi pencapaian nilai optimum suatu data.

5.2 Saran

Untuk penelitian selanjutnya, GA dapat diimplementasikan pada berbagai tipe *Knapsack Problem* yang lain, dengan penanganan kendala yang lainnya, yaitu strategi penalti, strategi perbaikan kromosom dan strategi perbaikan kromosom yang cukup efektif . Penelitian selanjutnya juga disarankan untuk membandingkan penggunaan operator dan parameter GA yang digunakan untuk menemukan solusi dan representasi yang tepat dari suatu persoalan.

DAFTAR PUSTAKA

Akbar,Md. M., Rahman, M. S., Kaykobad, M., Maning, E. G., dan Shojaei, G. C., 2004. Solving the Multidimensional Multiple-choice Knapsack Problem by Constructing Convex Hulls. *Elsevier*.

Astuti, D.A., 2015. Evaluasi Kinerja Genetic Algorithm (GA) dengan Strategi Perbaikan Kromosom: Studi Kasus Knapsack Problem. *Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Lampung*.

Chang, HF., Liang, CH. 2011. A New Heuristic for Solving the Multi-choice Multi-dimensional Knapsack Problem. *Digital Library.* Vol.35. 4. Pp708-717.

Elizabeth M. Hilliard, Amy Greenwald, Victor Naroditskiy. *An Algorithm for the Penalized Multiple Choice Knapsack Problem. Creative Commons Attribution Non-Commercial License. doi:10.3233/978-1-61499-419-0-1025*

Fanggidae, A. 2015. *Algoritma Genetika dan Penerapannya*. Yogyakarta. Teknosain.

Fukunaga, A. 2011. A Branch and Bound Algorithm for Hard Multiple Knapsack Problem. *Vannals of Operations Research.* Vol. 184, 1, pp 97-119.

Gen, Mitsuo dan Runwei Cheng. 2000. Genetic Algorithms and Engineering Optimization. Canada: A Wiley-Interscience Publication
Goldberg, D.E.. 1989. Genetic Algorithms in Search Optimizatiuon, and Machine Learning. Reading, MA:Addison Wesley.

Gupta. M. A Fast and Effecient Genetic Algorithm to Solve 0-1 Knapsack Problem *International Journal of Digital Aplication dan Contemporary Research Volume 1 Issue 6*. 2013.

Hifi, M., Michrafy, M., dan Sbihi, A., 2005. Algorithms for the Multiple-Choice Multi-Dimensional Knapsack Problem. *LaRIA, Laboratoire de Recherche en Informatique d'Amie ns, 5 rue du Moulin Neuf, 80000 Amien.* France.

Juniawati, 2003. Impelementasi Algoritma Genetika Untuk Mencari Volume Terbesar Bangun Kotak Tanpa Tutup Dari Suatu Bidang Datar Segi Empat. *Jurnal Ilmiah*. Surabaya, Indonesia : Universitas Surabaya.

Kusumadewi, S. 2003. *Artificial Intelligence (Teknik dan Aplikasinya).* Yogyakarta: Graha Ilmu.
Mahajan, Ritika dan Sarvesh Chopra. 2012. Analisys of 0/1 Knapsack Problem using Deterministic and Probabilistic Techniques. *Second International Conference on Advanced Computing dan Communication Techniques. IEEE*. 150.
Mahmudy, W.F. 2014. *Algoritma Evolusi.* Universitas Brawijaya.

Malhotra, R., Singh, N., Sing, Y., 2011. Genetic Algorithms: Concepts, Design for Optimization of Process Controllers. *Computer and Information Science, Canadian Center of Science and Education,* Vol. 4, No. 2, 29.

Malinda, R., 2015. Implementasi Genetic Algorithm Berbasis Strategi Penalti pada Knapsack Problem. *Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Lampung.*
Martello, S; Toth, P. 1990. An Exact Algorithm for Large Unbounded Knapsack Problems. *Operations research letters,* 9.1: 15-20.

Martello, S., & Toth, P. 1997. Upper Bounds and Algorithms for Hard 0-1 Knapsack Problems. *Operations Research*, 45(5), 768-778.

Martello, S., Pisinger, D., & Toth, P. (1999). Dynamic programming and Strong Bounds for the 0-1 Knapsack Problem. *Management Science*, 45(3), 414-424.

Martello, S., Pisinger, D. and Toth P., 2000. New Trends in Exact Algorithms for the 0-1 Knapsack Problem. *European Journal of Operational Research*., pp. 325-332.
M. Moser, D. P. Jokanovi´c and N. Shiratori. 1997. An Algorithm for The Multidimesional Multiple-Choice Knapsack Problem. *IEICE Transactions on Fundamentals of Electronics*. vol. 80, No 3., pp.582-589.

Shena, P. A. *Pemecahan Masalah Knapsack dengan Algoritma Branch And Bound*. Bandung : *Institut Teknologi Bandung.*

Pisinger, D., 1995. Algorithm for Knapsack Problem, *PhD Thesis, The University of Copenhagen,* Universitet Sparken 1, DK-2100 Copenhagen, Denmark.

Purnomo, H.D., 2014. *Cara Mudah Belajar Metode Optimasi Metaheuristik Menggunakan Matlab*. Yogyakarta: Gava Media.

Rahman, K. M dan Syed,i. A., 2009. Performance Analysis of Genetic Algorithm for Solving the Multiple-Choice Multi-Dimensional Knapsack Problem, *International Journal of Recent Trends in Engineering, Vol 2.*

Sbihi, A. D., Mustapha, M., dan Hifi, M., 2007. A Reactive Local Search-Based Algorithm for the Multiple-Choice Multi-Dimensional Knapsack Problem. *Computational Optimization and Applications*.

Sbihi, A., Michrafy, M., dan Hifi, M., 2005. A Reactive Local Search-Based Algorithm for the Multiple-Choice Multi-Dimensional Knapsack Problem. *Computational Optimization and Applications*.

Iqbal, S., Faizulbari, M.d., Rahman, S. 2010. Solving the Multi-dimensionala Multi-choice Knapsack Problem with the Help of Ants. ANTS Conference. Pp.312-323.

Shojaei, H., Basten, T., Geilen, M.C.W., Davoodi, A., 2013. A Fast and Scalable Multi-dimensional Multiple-choice Knapsack Problem. *ACM Transactions on Design Automation of Electronic Systems, Vol. 18, No.4.*

Sulistiyorini , R dan Wayan, F. M., 2015. Penerapan Algoritma Genetika Untuk Permasalahan Optimasi Distribusi Barang Dua Tahap, *Repository Jurnal Mahasiswa Ptiik Universitas Brawijaya,* Vol. 5, No. 12.

Suyanto. 2005. *Algoritma Genetika Dalam Matlab*. Yogyakarta : Penerbit Andi.

Suyanto. 2007. *Artificial Intellegence*. Bandung : Penerbit Informatika

Suyanto,T. 2008. *Evolutionary Computation. Komputasi berbasis Evolusi dan Genetika*. Bandung. Penerbit Informatika Bandung.

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

T. Sutojo, S.Si., M.Kom. 2011. *Kecerdasan Buatan*. ANDI : Yogyakarta

Varnamkhasti, M. J., 2012. Overview of the Algorithms for Solving the Multidimensional Knapsack Problems. Advanced *Studies in Biology*, Vol. 4, 2012, no. 1, 37 – 47.

Zennaki, M. 2013., A New Hybrid Algorithm for the Multiple-Choice Multi-Dimensional Knapsack Problem, *Wseas Transactions on Information Science and Applications*.

Zukri, Zainudin. 2014. *Algoritma Genetika: Metode Komputasi Evolusioner untuk Menyelesaikan Masalah Optimasi*. Yogyakarta: Penertbit Andi Yogyakarta.