OPTIMASI MODEL REGRESI MENGGUNAKAN ALGORITMA GENETIKA UNTUK MEMPREDIKSI JUMLAH MAHASISWA YANG MENGAMBIL MATA KULIAH

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Sutrisno
NIM. 145150209111002

PROGRAM STUDI INFORMATIKA/ ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

Optimasi Model Regresi Menggunakan Algoritma Genetika untuk Memprediksi Jumlah Mahasiswa yang Mengambil Mata Kuliah

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Sutrisno
NIM: 145150209111002

Skripsi ini telah diuji dan dinyatakan lulus pada
14 APRIL 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing

Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D
NIP. 19720919 199702 1 001

Mengetahui
Ketua Program Studi Informatika

Issa Arwani, S.Kom, M.Sc
NIP. 19830922 201212 1 003

PERNYATAAN ORISINALITAS

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 disitasi dalam naskah ini dan disebutkan dalam 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, 14 Maret 2016

Sutrisno

NIM: 145150209111002

KATA PENGANTAR

Puji penulis panjatkan atas kehadirat Allah SWT yang telah memberikan kenikmatan, kesehatan dan kelancaran dalam proses pembuatan skripsi ini. Sholawat serta salam penulis persembahkan kepada nabi Muhammad SAW yang telah memberikan petunjuk jalan kebenaran kepada umat manusia.

Terimakasih penulis ucapkan kepada bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku dosen pembimbing yang memberikan arahan, bimbingan dan motivasi agar skripsi ini dapat diselesaikan dengan baik dan tepat waktu. Tidak lupa rasa terimakasih penulis ucapkan kepada bapak Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer, dan bapak Issa Arwani, S.Kom, M.Sc selaku ketua program studi Informatika.

Terimakasih penulis ucapkan pula kepada orang tua yang selalu memberikan dukungan motivasi, doa dan materi kepada penulis. Terimakasih juga penulis ucapkan kepada Pasha Trylaksana Wiraharja, Garsinia Ely Riani, Elwin, Siska Puji Rahayu atas segala doa serta bantuan selama ini, dan semua pihak yang telah membantu penulis.

Penuis berharap semoga skripsi ini dapat bermanfaat bagi semua pihak, selain itu saran dan kritik yang membangun untuk perbaikan lebih lanjut sangat penulis butuhkan.

Malang, 29 Februari 2016

Penulis

ABSTRAK

Pembuatan jadwal mata kuliah dilakukan oleh pihak akademik pada setiap awal semester. Saat awal semester itulah mahasiswa akan memprogram kartu rencana studi dengan menempuh perkuliahan sesuai dengan jumlah SKS maksimal sesuai indeks prestasi yang diperoleh pada semester sebelumnya. Pembuatan jadwal membutuhkan tenaga, pikiran dan waktu yang banyak karena harus mempertimbangkan ketersediaan ruang, waktu, dosen dan perkiraan jumlah mahasiswa yang mengambil mata kuliah. Seringkali pihak akademik mengalami kesulitan memprediksi jumlah mahasiswa yang akan menempuh mata kuliah tertentu, khususnya memprediksi mahasiswa yang sebelumnya telah menempuh mata kuliah tersebut namun masih memperoleh nilai yang kurang memuaskan apakah akan memperbaiki nilainya atau tidak. Sehingga sering terjadi kekurangan kelas, karena mata kuliah yang sedang dibuka banyak diminati. Dilain sisi ada pula kelas yang masih banyak kosong, dimana mahasiswa kurang minat terhadap kelas yang dibuka. Sehingga diperlukan sebuah sistem yang mampu membantu pihak akademik dalam menentukan perkiraan kelas yang akan dibuka. Dalam optimasi menggunakan algoritma genetika, representasi kromosom yang digunakan adalah *binary*, proses *crossover* yang digunakan adalah *one cut point crossover*, proses mutasi menggunakan *random mutation*, serta seleksi menggunakan metode seleksi *elitism*. Hasil penelitian memberikan nilai parameter terbaik yaitu ukuran populasi 600, jumlah generasi 400, dengan kombinasi cr : mr berturut-turut adalah 0,4 : 0,6. Dalam pengujian yang dilakukan peneliti, hasil terbaik dari optimasi ini dibuktikan dengan nilai MSE terkecil 5,64 dengan kromosom terpilih 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 . Sehingga dapat disimpulkan model regresi linear berganda yang dibangun menggunakan algoritma genetika mampu untuk memprediksi jumlah mahasiswa pengambil mata kuliah dengan akurasi yang baik dan dengan usaha yang lebih mudah dibandingkan dengan model prediksi lainnya.

Kata kunci : optimasi, algoritma genetika, prediksi mahasiswa mengulang

ABSTRACT

Subject schedule making is conducted by academic employees in every beginning of the semester. In the beginning of the semester, the students will programmed the study program by taking the lecture based on the number of maximum SKS based on the achievement index which was gained in the previous semester. Schedule making needs power, thought, and much time since it needs to consider the availability of classroom, time, lecturer and the number of student's estimation who will take the subjects. Academic employees often find difficulties in predicting the number of those who will take the certain subjects, but they have not got the satisfying score yet, whether they will revise the score or not. Thus, it causes the lack of classroom, since the subject which is being opened has high proclivity of students in choosing the subject. On the other hand, there is opened classroom which still has high capacity because the number of the students who take the subject is lesser. Thus, it needs a system which can help the academic employees in deciding classrooms estimation which are going to be opened. In this genetic algorithm optimization, chromosomes representation using binary, crossover process using one cut point crossoer, mutation peocess using elitism method. The results of this study provide the best parameter value that is population size is 600, generation size is 400, combination cr:mr is 0,4 : 0,6. The best result of this optimization is evidence by the smallest MSE value of 5,64 with selected chromosome 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1. It can be concluded linear regression models were constructed using genetic algorithm were able to predict the number of students estioation have a good accuracy than another prediction models.

Key words: optimization, genetics algorithm, the repeating student's estimation.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR *SOURCE CODE*

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Pembuatan jadwal perkuliahan pada suatu jurusan atau fakultas dilakukan rutin setiap pergantian semester. Pembuatan jadwal mata kuliah ini merupakan hal yang penting dalam proses perkuliahan, proses penyusunan jadwal mata kuliah harus disusun dengan baik. Salah satu kriteria jadwal mata kuliah yang baik memiliki distribusi mata kuliah yang merata dan jumlah mahasiswa yang sesuai dengan kapasitas kelas untuk menghasilkan suasana perkuliahan yang kondusif.

Proses penyusunan jadwal perkuliahan bukanlah sebuah pekerjaan yang mudah. Ada banyak aspek dan kendala yang mempengaruhi dalam penyusunan jadwal mata kuliah yaitu dosen, mahasiswa, mata kuliah, ruang perkuliahan, slot waktu dan prediksi jumlah mahasiswa yang memiliki minat untuk mengambil mata kuliah tertentu. Setiap aspek tersebut memiliki keadaan yang dapat menjadi masalah dan konflik dalam proses perkuliahan. Misalnya masalah yang dihadapi adalah jumlah kelas yang dibuka tidak sesuai dengan jumlah permintaan dari mahasiswa, sehingga ada mata kuliah tertentu mengalami kekosongan karena sedikitnya minat mahasiswa yang mengambil mata kuliah tersebut. Dilain sisi ada mata kuliah yang sangat diminati oleh mahasiswa sehingga dengan sangat cepat kelas terpenuhi. Hal ini mengakibatkan tidak sesuainya persediaan jumlah kelas yang dibuka dengan jumlah mahasiswa yang akan mengambil mata kuliah.

Menentukan jumlah perkiraan atau prediksi mahasiswa yang akan mengambil suatu mata kuliah harus dilakukan pada setiap pembuatan jadwal atau saat pergantian semester. Hal ini penting untuk menentukan berapa kelas yang akan dibuka dan nantinya akan mempengaruhi pada proses pembuatan jadwal perkuliahan sesuai dengan daya tampung, jumlah peminat, ketersediaan dosen dan jam pelajaran.

Penentuan jumlah kelas yang dibuka diperlukan sebuah perhitungan dengan menggunakan sebuah metode yang mampu memprediksi perkiraan jumlah mahasiswa yang akan mengambil mata kuliah tersebut. Dari total mahasiswa yang mengambil mata kuliah tertentu terdiri dari mahasiswa yang baru yang pertama kali mengambil mata kuliah dan mahasiswa yang mengulang untuk memperbaiki nilai yang pernah diperoleh pada semester sebelumnya. Mahasiswa mengulang mata kuliah yang sebelumnya memiliki nilai C dan C+, yang berpotensi untuk mengulang pada setiap mata kuliah tersebut. Hal ini karena untuk mahasiswa yang sudah mendapatkan nilai A, B+ dan B sudah dipastikan tidak akan mengulang karena nilai mereka sudah baik dan dipastikan

lulus, sedangkan mahasiswa yang memperoleh nilai D dan E hampir dipastikan akan mengambil mata kuliah tersebut. Dari mahasiswa yang memperoleh nilai C+, C, D+, D da E tidak diketahui kapan mahasiswa tersebut akan mengulang.

Untuk dapat memberikan solusi dari permasalahan perkiraan jumlah mahasiswa yang akan mengulang mata kuliah yang sebelumnya memperoleh nilah C+, C, D+, D dan E pada setiap mata kuliah, maka ditawarkan sebuah sistem yang mampu membantu memecahkan masalah tersebut dengan menggunakan algotima genetika.

Penggunaan algoritma genetika untuk menyelesaikan permasalahan perkiraan atau prediksi telah dibahas dalam beberapa penelitian sebelumnya. Misalkan (Permatasari & Mahmudy, 2015) yang menerapkan algoritma genetika untuk melakukan prediksi konsumsi kWh listrik di kota Batu. Penelitian sejenis dilakukan pula oleh (Wahyuni, Sutojo, & Luthfiarta, 2014) yang menerapkan Naïve Bayes dan Algoritma Genetika untuk memprediksi hasil pemilu legislatif DKI Jakarta. Penelitian selanjutnya yang dilakukan oleh (Rahmi, Mahmudy, & Setiawan, 2015) yang menggunakan model regresi yang dibangun dengan menggunakan algoritma genetika untuk memprediksi harga saham berdasarkan data historis. Dari ketiga hasil penelitian tersebut menyimpulkan bahwa algoritma genetika dapat digunakan untuk memprediksi dengan cara membentuk koefisien terbaik pada analisis regresi linear berganda.

Algoritma genetika yang telah berhasil diterapkan pada berbagai masalah kompleks, maka pada skripsi ini agoritma genetika dgunakan untuk optimasi persamaan regresi linear berganda dalam menentukan jumlah mahasiswa yang mengambil mata kuliah.

1.2 Rumusan Masalah

Dari latar belakang yang telah diuraikan diatas, maka rumusan masalah yang akan dibahas sebagai berikut:

1. Bagaimana menerapkan metode algoritma genetika untuk optimasi model regresi linear berganda pada prediksi jumlah mahasiswa yang mengambil mata kuliah?
2. Bagaimana menentukan representasi kromosom terbaik untuk optimasi model regresi linear berganda menggunakan algoritma genetika dalam memprediksi jumlah mahasiswa yang mengambil mata kuliah?
3. Bagaimana parameter terbaik dalam optimasi menggunakan agoritma genetika untuk menentukan jumlah mahasiswa yang mengambil mata kuliah?

1.3 Tujuan

Adapun tujuan dari penulisan skripsi ini adalah:

1. Untuk menerapkan optimasi menggunakan algoritma genetika dalam melakukan proses prediksi jumlah mahasiswa yang mengambil mata kuliah.
2. Untuk mengetahui representasi kromosom yang terbaik dalam proses optimasi algoritma genetika dalam proses prediksi jumlah mahasiswa yang mengambil mata kuliah.
3. Untuk mengetahui parameter terbaik dalam menentukan jumlah mahasiswa yang akan mengambil mata kuliah.

1.4 Batasan Masalah

Adapun batasan masalah dalam skripsi ini yaitu:

1. Metode yang digunakan yaitu optimasi menggunakan algoritma genetika untuk menentukan model persamaan analisis regresi linear berganda.
2. Penerapan prediksi mahasiswa yang mengambil mata kuliah hanya untuk memprediksi jumlah mahasiswa yang memungkinkan mengulang pada mata kuliah tertentu, yaitu mahasiswa yang memperoleh nilai C+, C, D+, D dan E pada perkuliahan sebelumnya.

1.5 Manfaat Penelitian

Manfaat yang diperoleh dari skripsi ini yaitu:

1. Mengetahui prediksi jumlah mahasiswa yang akan mengambil mata kuliah dari mahasiswa yang sebelumnya memperoleh nilai C+, C, D+, D dan E.
2. Membantu pihak akademik dalam menentukan jumlah kelas yang akan dibuka pada tiap semesternya.

1.6 Sistematika Pembahasan

Pembuatan skripsi ini dikerjakan dengan sistematika penulisan sebagai berikut:

BAB I PENDAHULUAN

Berisi latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat dan sistematika penulisan.

BAB II LANDASAN KEPUSTAKAAN

Berisi landasan teori dan kepustakaan yang menjadi dasar dan acuan penyelesaian skripsi.

BAB III METODOLOGI PENELITIAN

Bab ini berisi metode-metode yang digunakan untuk menyelesaikan permasalahan yang sedang dibahas.

BAB IV PERANCANGAN

Bab ini berisi perancangan dan perhitungan manual algoritma yang akan digunakan untuk menyelesaikan permasalahan.

BAB V IMPLEMENTASI

Bab ini berisi implementasi program dari perancangan yang telah dibuat pada bab sebelumnya.

BAB VI PENGUJIAN DAN PEMBAHASAN

Bab ini berisi pengujian dan analisis dari hasil algoritma yang digunakan dalam pemecahan permasalahan.

BAB V KESIMPULAN DAN SARAN

Berisikan kesimpulan dan saran dari penulisan skripsi ini.

DAFTAR PUSTAKA

Berisi daftar literature ilmiah yang dijadikan acuan baik berupa buku, jurnal maupun website yang mendukung penyelesaian skripsi ini.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tinjauan Pustaka

Sejarah perkembangan algoritma genetika (*genetic algorithm*) berawal pada tahun 1960-an ketika I. Rochenberg dalam bukunya yang berjudul "*Evolution Strategies*" mengemukakan tentang evolusi komputer (*computer evolutionary*) yang kemudian dikembangkan oleh John Holland pada tahun 1970-an. John Holland menulis buku tentang algoritma genetika yang berjudul "*Adaptation in Natural and Artificial System*" yang diterbitkan pada tahun 1975.

Algoritma genetika *(Genetic Algorithm, GAs)* mampu menyelesaikan berbagai permasalahan kompleks dan dapat mengalami perkembangan seiring perkembangan teknologi (Mahmudy, 2013). GAs telah banyak digunakan untuk menyelesaikan berbagai macam kasus, terutama masalah optimasi, termasuk untuk membuat prediksi dari suatu kondisi. Pada penelitian ini akan dijabarkan beberapa referensi yang berhubungan dan relevan dengan kasus yang diangkat sebagai rujukan dalam menyelesaikan penelitian ini.

Pada penelitian sebelumnya yang pernah dilakukan dapat dilihat perbandingan objek, metode dan perbandingan dari masing-masing studi literatur skripsi peneliti. Perbandingan tersebut dapat dilihat pada Tabel 2.1 berikut.

Tabel 2.1. Perbandingan Objek dan Metode Penelitian Sebelumnya

No	Judul	Objek	Metode	Perbandingan	
				Kajian Pustaka	Skripsi Peneliti
1.	Optimasi Jaringan Syarat Tiruan dengan Algoritma Genetika untuk Peramalan Curah Hujan.	Peramalan Curah Hujan	Jaringan Syaraf Tiruan dan Algoritma Genetika	Prediksi Jaringan Syaraf Tiruan dengan Algoritma Genetika	Analisis regresi linear berganda dan algoritma genetika
2.	*Evolving RBF Neural Networks for Rainfall Prediction Using Hybrid Particle Swarm Optimization*	Peramalan Curah Hujan	Hybrid PSO (Particle Swarm Optimization) dan Algoritma Genetika	Prediksi curah hujan dengan HPSO dan Algoritma genetika	Analisis regresi linear berganda dan algoritma genetika

	And Genetic Algorithm Review				
3.	Perbandingan Metode Fuzzy dengan Regresi Linier Berganda dalam Peramalan Jumlah Produksi (Studi Kasus: Produksi Kelapa Sawit di PT. Perkebunan III (PERSERO) Medan Tahun 2011-2012)	Peramalan Jumlah Produksi	Perbandingan Metode Fuzzy dengan Regresi	Prediksi dengan membandingkan metode Fuzzy dan Regresi Linear Berganda	Analisis regresi linear berganda dan algoritma genetika

Sumber : Data diolah oleh peneliti

Pada Tabel 2.1 diatas, skripsi yang dikerjakan oleh peneliti yang berjudul "Optimasi Model Regresi Menggunakan Algoritma Genetika untuk Memprediksi Jumlah Mahasiswa yang Mengambil Mata Kuliah" dibandingkan dengan daftar referensi yang sudah pernah dilakukan penelitian sebelumnya. Dengan melakukan perbandingan ini, diharapkan dapat dilakukan kombinasi dan pembanding antara hasil penelitian yang pernah dilakukan pada Tabel 2.1 dengan penelitian ini.

2.2 Pengambilan Mata Kuliah

Mahasiswa akan mengambil suatu mata harus dilakukan setiap awal semester. Pada saat seperti itulah, mahasiswa akan melakukan program pengisian KRS (Kartu Rencana Studi) di Sistem Informasi Akademik Mahasiswa atau SIAM.

Mahasiswa yang telah menyelesaikan proses registrasi dan pembayaran finansial untuk pendidikan tiap semester akan dilanjutkan untuk melakukan pengisian KRS pada SIAM. Mahasiswa akan memilih mata kuliah, kelas, jumlah SKS maksimal dan hari perkuliahan sesuai dengan keminatan dan IP (indeks prestasi) yang diperoleh pada semester sebelumnya. Penentuan nilai IP merupakan rata-rata nilai yang diperoleh mahasiswa dalam rentang nilai 0,0-4,0. Setiap universitas akan memiliki aturan penilaian sebagai standar baku seperti yang terlihat pada Tabel. 2.2 berikut:

Tabel 2.2. Salah Satu Aturan Penilaian Indeks Prestasi

Nilai Angka (N)	Nilai Huruf	Bobot
80 < N ≤ 100	A	4
75 < N ≤ 80	B+	3,5
69 < N ≤ 75	B	3
60 < N ≤ 69	C+	2,5
55 < N ≤ 60	C	2
50 < N ≤ 55	D+	1,5
44 < N ≤ 50	D	1
0 ≤ N ≤ 44	E	0

Sumber: buku pedoman mahasiswa PTIIK Universitas Brawijaya

Dalam pemilihan kelas, mahasiswa yang akan mengambil mata kuliah harus memilih kelas yang telah disediakan atau dibuka oleh akademik dari fakultas. Idealnya jumlah mahasiswa dalam setiap kelas yang dibuka tidak lebih dari 40. Pada umumnya kelas yang dibuka ditandai dengan sebuah variabel, yaitu kelas A, B, C, D dan seterusnya.

Mahasiswa yang mengambil mata kuliah berasal dari dua jenis, yaitu mahasiswa yang baru pertama kali mengambil mata kuliah dan mahasiswa yang mengulang karena pada semester sebelumnya telah menempuh mata kuliah tersebut namun masih perlu untuk memperbaiki nilai yang diperolehnya.

2.3 Algoritma Genetika

Algoritma secara umum diartikan sebagai urutan logis dalam pengambilan keputusan untuk pemecahan suatu permasalahan. Algoritma merupakan inti dari sebuah sistem kecerdasan. Algoritma yang memiliki tingkat akurasi dan kualitas yang baik akan menghasilkan sebuah sistem yang baik pula. Dalam dunia informatika banyak dikenal berbagai macam algoritma, salah satu algoritma yang digunakan untuk membangun sebuah sistem kecerdasan adalah algoritma genetika.

Algortima genetika (AG) pertama kali dirintis oleh John Holland dari Universitas Michigan pada tahun 1970-an, AG telah diaplikasikan secara luas pada berbagai bidang. AG banyak digunakan untuk memecahkan masalah optimasi, walaupun pada kenyataannya juga memiliki kemampuan yang baik untuk masalah- masalah selain optimasi. John Holland menyatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan dalam terminologi genetika. Algoritma genetika adalah simulasi dari proses evolusi dan operasi genetika atas kromosom.

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

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

Nilai *fitness* dari suatu kromosom akan menunjukkan kualitas dari kromosom dalam populasi tersebut. Pada penelitian ini, semakin besar nilai *fitness*, maka menunjukkan bahwa kualitas dari kromosom semakin baik pula. Generasi berikutnya dikenal dengan istilah anak *(offspring)* terbentuk dari gabungan dua kromosom generasi sekarang yang bertindak sebagai induk *(parent)* dengan menggunakan operator penyilangan *(crossover)*.

Selain operator penyilangan, suatu kromosom dapat juga dimodifikasi dengan menggunakan operator mutasi. Populasi generasi yang baru dibentuk dengan cara menyeleksi nilai *fitness* dari kromosom induk *(parent)* dan nilai *fitness* dari kromosom anak *(offspring)*, serta menolak kromosom-kromosom yang lainnya sehingga ukuran populasi (jumlah kromosom dalam suatu populasi) konstan. Setelah melalui beberapa generasi, maka algoritma ini akan konvergen ke kromosom terbaik.

Algoritma Genetika banyak digunakan dalam proses optimasi. Menurut (Mahmudy, 2013) dalam proses optimasi sendiri, Algoritma Genetika memiliki berbagai keunggulan dan kelebihan yang menonjol, yaitu:

a) GA merupakan algoritma yang berbasis populasi yang memungkinkan digunakan pada optimasi masalah dengan ruang pencarian *(search space)* yang sangat luas dan kompleks.
b) Individu yang ada pada populasi bisa diletakkan pada beberapa sub-populasi yang diproses pada sejumlah komputer secara paralel. Hal ini bisa mengurangi waktu komputasi pada masalah yang sangat kompleks. Penggunaan sub-populasi juga bisa dilakukan pada hanya satu komputer untuk menjaga keragaman populasi dan meningkatkan kualitas hasil pencarian.
c) GAs menghasilkan himpunan solusi optimal yang sangat berguna pada peyelesaian masalah dengan banyak obyektif.
d) GAs dapat digunakan untuk menyelesaikan masalah yang kompleks dengan banyak variabel. Variabel tersebut bisa kontinyu, diskrit atau campuran keduanya.
e) GAs menggunakan kromosom untuk mengkodekan solusi sehingga bisa melakukan pencarian tanpa memperhatikan informasi derivatif yang spesifik dari masalah yang diselesaikan.
f) GAs bisa diimplementasikan pada berbagai macam data seperti data yang dibangkitkan secara numerik atau menggunakan fungsi analitis.

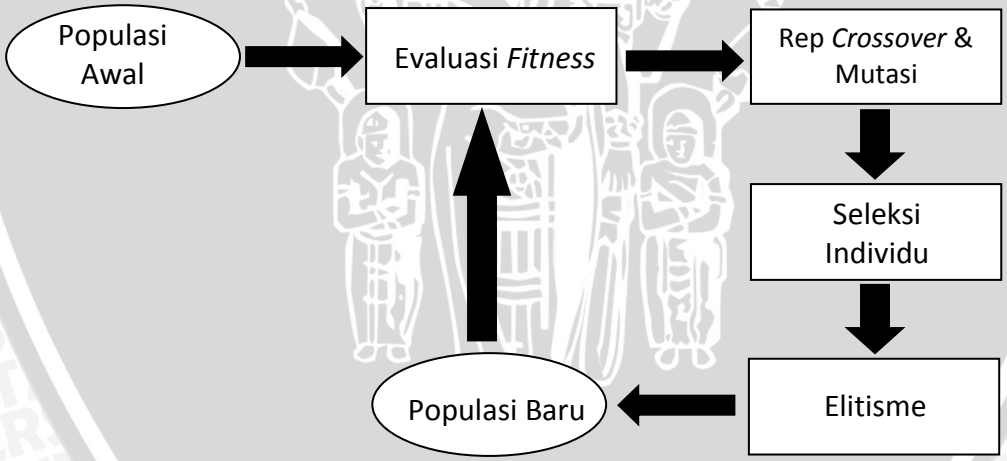
g) GAs cukup fleksibel untuk dihibridisasikan dengan algoritma lainnya. Beberapa penelitian membuktikan bahwa hybrid GAs (HGAs) sangat efektif untuk menghasilkan solusi yang lebih baik.

2.3.1 Struktur Umum Algoritma Genetika

Algoritma genetika merupakan teknik *search stchohastic* yang berdasarkan pada mekanisme seleksi alam dan genetika natural. Yang menjadi perbedaan antara algoritma genetika dengan algoritma konvensional lainnya adalah algoritma genetika menggunakan teknik pencarian secara sekaligus atas sejumlah solusi yang mungkin dikenal dengan istilah populasi.

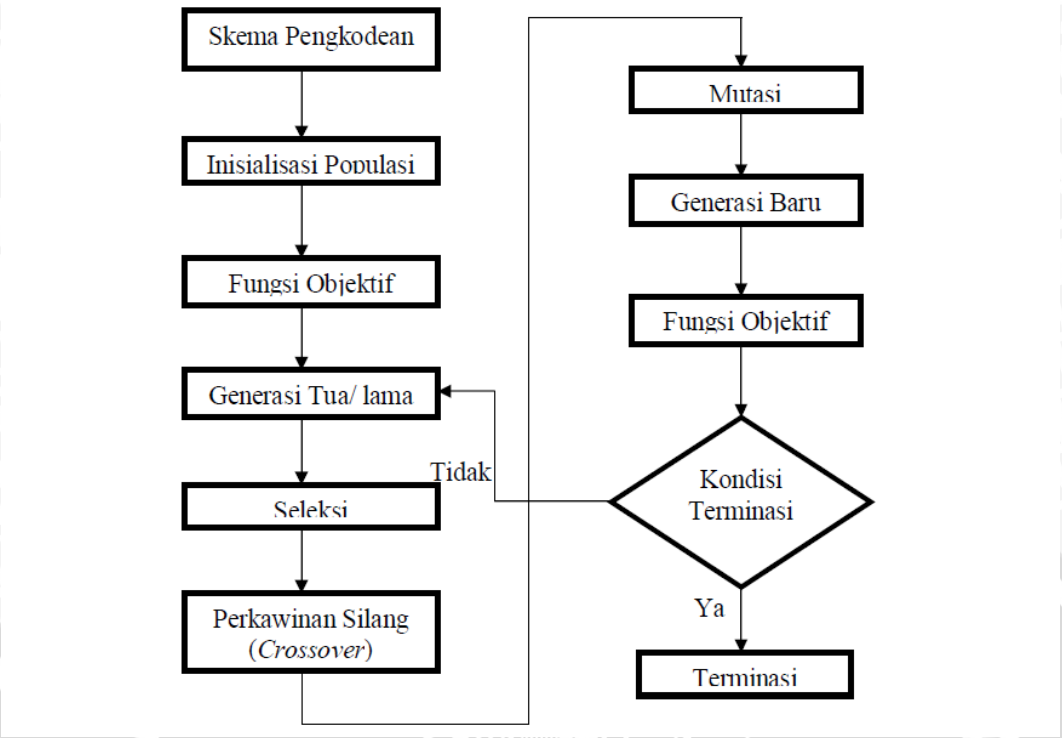
Setiap individu didalam populasi disebut kromosom, yang merepresentasikan suatu penyelesaian terhadap masalah yang ditangani. Sebuah kromosom terdiri dari sebuah *string* yang berisi berbagai simbol, dan biasanya, tetapi tidak mutlak, *string* tersebut berupa sederetan bit-bit biner "0" dan "1". Sebuah kromosom tumbuh atau berkembang biak melalui berbagai iterasi yang berulang-ulang, dan disebut sebagai generasi. Pada setiap generasi, berbagai kromosom yang dihasilkan akan dievaluasi menggunakan suatu pengukuran *fitness*.

Secara skematis, siklus algoritma genetika dapat digambarkan seperti pada Gambar 2.1 berikut:



Gambar 2.1. Siklus Algoritma Genetika

Algoritma genetika memberikan sebuah pilihan untuk penentuan nilai dari parameter yang digunakan dengan meniru proses reproduksi genetik, melalui pembentukan kromosom baru serta seleksi alami seperti yang terjadi pada makhluk hidup. Proses algoritma genetika secara umum dapat diilustrasikan dalam diagram alir pada Gambar 2.2 berikut:



Gambar 2.2. Diagram Alir Algoritma Genetika

Goldberg (1989) mengemukakan bahwa algoritma genetika mempunyai karakteristik yang perlu diketahui sehingga dapat terbedakan dari prosedur pencarian atau optimasi yang lain, yaitu:

1. Algoritma genetika dengan pengkodean dari himpunan solusi permasalahan berdasarkan parameter yang telah ditetapkan dan bukan parameter itu sendiri.
2. Algoritma genetika mencari sebuah solusi dari sejumlah individu-individu yang merupakan solusi permasalahan bukan hanya dari sebuah individu.
3. Algoritma genetika memiliki informasi fungsi objektif *(fitness),* sebagai cara untuk mengevaluasi individu yang mempunyai solusi terbaik, bukan turunan dari sebuah fungsi.
4. Algoritma genetika menggunakan aturan-aturan transisi peluang, bukan aturan deterministik.

Parameter dan variabel yang digunakan dalam algoritma genetika yaitu:

1. Fungsi *fitness* yang dimiliki oleh setiap individu untuk menentukan tingkat kesesuaian individu tersebut dengan kriteria yang dicapai.
2. Populasi jumlah individu yang dilibatkan dalam setiap generasi.
3. Peluang terjadinya persilangan *(crossover)* pada suatu generasi.
4. Peluang terjadinya mutasi pada setiap individu.
5. Jumlah generasi yang akan dibentuk menentukan lamanya penerapan algoritma genetika.

Secara garis besar struktur yang ada dari suatu algoritma genetika dapat didefinisikan dengan langkah-langkah sebagai berikut:

1. Membangkitkan populasi awal.
   Populasi yang ada pada tahap awal ini dibangkitkan secara acak 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 atau seleksi *crossover* dan mutasi. Proses ini dilakukan berulang-ulang sehingga didapatkan sejumlah 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. Evolusi solusi
   Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang dinamakan *fitness.* Nilai dari *fitness* dalam suatu kromosom menggambarkan kualitas kromosom dalam populasi tersebut. Proses ini akan melakukan evaluasi pada setiap populasi dengan menghitung nilai *fitness* setiap kromosom dan melakukan evaluasi sampai terpenuhi kriteria yang diinginkan dan berhenti. Jika kriteria berhenti belum terpenuhi maka akan dibentuk lagi generasi baru dengan mengulangi langkah 2. 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.

2.3.2 Pengkodean

Pengkodean adalah satu teknik untuk menyatakan populasi awal sebagai calon solusi suatu masalah kedalam suatu kromosom sebagai kunci pokok persoalan ketika menggunakan algoritma genetika. Berdasarkan jenis yang digunakan sebagai nilai suatu gen, metode pengkodean dapat diklasifikasikan sebagai berikut:

2.3.2.1 Pengkodean Biner

Pengkodean biner merupakan cara pengkodean yang paling umum digunakan. Pengkodean biner pertama kali digunakan dalam algoritma genetika oleh Holland. Keuntungan pengkodean ini adalah sederhana untuk diciptakan dan mudah pula untuk 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 kadang pengoreksian harus dilakukan setelah operasi *crossover* dan mutasi.

2.3.2.2 Pengkodean Bilangan Bulat

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

2.3.2.3 Pengkodean Struktur Data

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

2.3.2.4 Pengkodean Nilai Riil

Pengkodean nilai riil adalah suatu pengkodean bilangan dalam bentuk riil. Masalah optimalisasi fungsi dan optimalisasi kendala lebih tepat jika diselesaikan dengan pengkodean bilangan rill karena struktur topologi ruang genotif untuk pengodean nilai riil identic dengan ruang fenotifnya, sehingga mudah membentuk operator genetic yang efektif dengan cara memakai teknik yang dapat digunakan yang berasal dari metode konvensional.

2.3.3 Operator Algoritma Genetika

Algoritma genetika merupakan proses pencarian yang heuristik dan acak sehingga penekanan pemilihan operator yang digunakan sangat menentukan keberhasilan algoritma genetika dalam menentukan solusi optimum suatu masalah yang diberikan. Hal yang harus diperhatikan adalah menghindari terjadinya konvergensi *premature,* yaitu mencapai solusi optimum yang belum pada waktunya. Dalam arti bahwa solusi yang diperoleh adalah optimum lokal.

Operator genetika yang digunakan setelah proses evaluasi tahap pertama membentuk poulasi baru dari genetika yang ada sekarang. Operator-operator tersebut adalah operator seleksi, *crossover* dan mutasi.

2.3.3.1 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 wadah seleksi akan menerima probabilitas reproduksi yang tergantung pada nilai objektif dirinya sendiri terhadap nilai objektif dari semua individu dalam wadah seleksi tersebut. Nilai *fitness* inilah yang nantinya akan digunakan pada tahap seleksi berikutnya.

Kemampuan algoritma genetika untuk melakukan produksi kromosom yang lebih baik secara progresif tergantung pada penekanan selektif *(selective pressure)* yang diterapkan ke populasi. Penekanan selektif dapat diterapkan dalam dua cara, yaitu:

1. Membuat lebih banyak kromosom anak yang dipelihara dalam populasi dan memilih hanya kromosom-kromosom terbaik bagi generasi berikut. Walaupun orang tua dipilih secara acak, metode ini akan terus menghasilkan kromosom yang lebih baik berhubungan dengan penekakan selektif yang diterapkan pada individu anak tersebut.
2. Memilih orang tua yang lebih baik ketika membuat keturunan baru. Dengan 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 orang tua.

2.3.3.2 Perkawinan Silang *(Crossover)*

Crossover (perkawinan silang) bertujuan menambah keanekaragaman *string* dalam populasi dengan penyilangan antar-*string* yang diperoleh dari sebelumnya. Beberapa jenis *crossover* tersebut adalah:

1. *Crossover* 1-titik

   Pada *crossover* dilakukan dengan memisahkan suatu *string* menjadi dua bagian dan 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 diperlihatkan pada gambar berikut:

Tabel 2.3. Contoh *Crossover* 1-titik

Kromosom orang tua 1	**11001011**
Kromosom orang tua 2	11011111
Keturunan	**1100**1111

2. *Crossover* 2-titik

   Proses *crossover* ini dilakukan dengan memilih dua titik *crossover*. Kromosom keturunan kemudian dibentuk dengan barisan bit dari awal kromosom sampai titik *crossover* pertama disalin dari orangtua pertama, bagian dari titik *crossover* pertama dan kedua disalin dari orangtua kedua, kemudian selebihnya disalin dari orangtua pertama lagi.

Tabel 2.4. Contoh *Crossover* 2-titik

Kromosom orang tua 1	11001011
Kromosom orang tua 2	11011111
Keturunan	**11011111**

3. *Crossover* seragam

   Crossover seragam menghasilkan kromosom keturunan dengan menyalin bit-bit secara acak dari kedua orangtuanya.

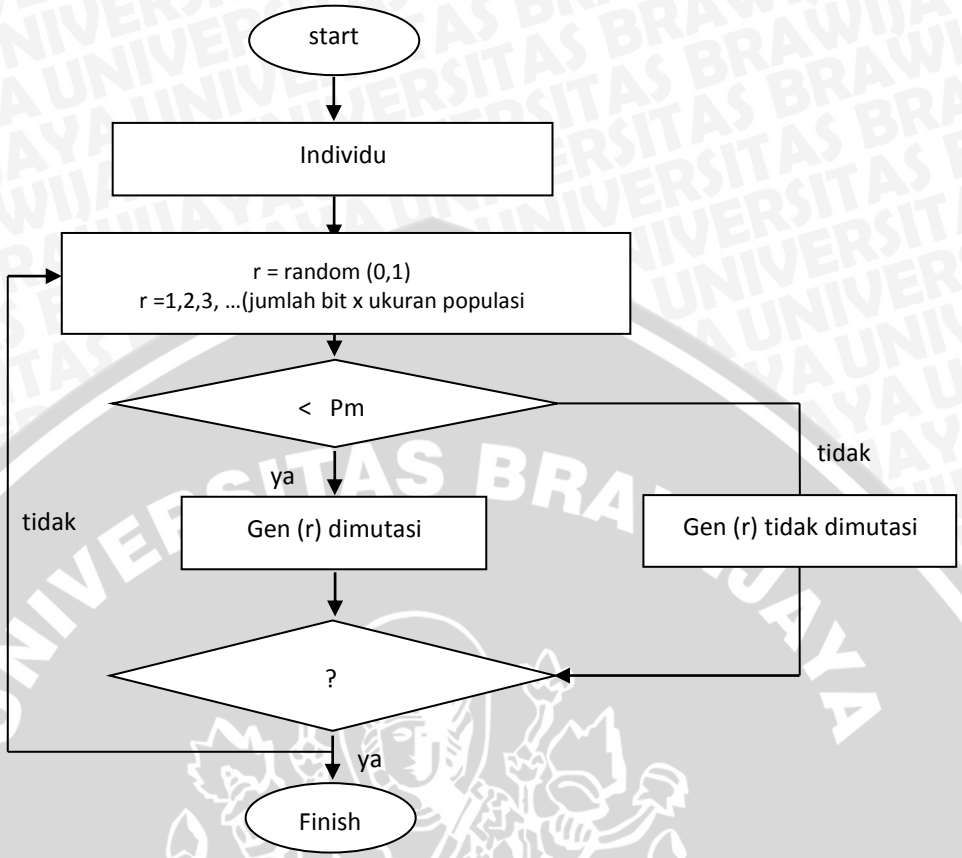
Tabel 2.5. Contoh *Crossover* Seragam

Kromosom orang tua 1	11001011
Kromosom orang tua 2	11011111
Keturunan	**11011111**

2.3.3.3 Mutasi

Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom. Operasi *crossover* yang dilakukan pada kromosom dengan tujuan untuk memperoleh kromosom-kromosom baru sebagai kandidat solusi pada generasi mendatang dengan *fitness* yang lebih baik, dan semakin lama menuju solusi optimum yang diinginkan. Akan tetapi, untuk mencapai hal ini, penekanan selektif juga memegang peranan yang penting. Jika dalam proses pemilihan kromosom-kromosom cenderung pada kromosom yang memiliki *fitness* yang tinggi saja, maka akan terjadi konvergensi *premature*. Konvergensi *premature* yaitu mencapai solusi yang optimal lokal sangat mudah terjadi.

Untuk menghindari konvergensi *premature* tersebut dan tetap menjaga perbedaan *(diversity)* kromosom-kromosom dalam populasi, selain melakukan penekanan selektif yang lebih efisien, operator mutasi juga dapat digunakan. 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. Secara skematis proses mutasi dapat digambarkan pada Gambar 2.3 berikut:



Gambar 2.3. Proses Terjadinya Mutasi

Operator mutasi merupakan operasi yang menyangkut satu kromosom tertentu. Beberapa cara operasi mutasi diterapkan dalam algoritma genetika menurut jenis pengkodean terhadap *phenotype*, antara lain:

1. Mutasi dalam pengkodean biner

   Mutasi pada pengkodean biner merupakan operasi yang sangat sederhana. Proses yang dilakukan adalah menginversi nilai bit pada posisi tertentu yang terpilih secara acak (atau menggunakan skema tertentu) pada kromosom, yang disebut *inverse* bit.

   Tabel 2.6. Contoh Mutasi pada pengkodean biner

Kromosom sebelum mutasi	1 0 0 1 0 1 1 1
Kromosom setelah mutasi	1 0 0 1 0 0 1 1

2. Mutasi dalam pengkodean permutasi

   Proses mutasi yang dilakukan dalam pengkodean biner dengan mengubah langsung bit-bit pada kromosom tidak dapat dilakukan pada pengkodean permutasi karena konsistensi urutan permutasi harus diperhatikan. Salah satu cara yang dapat dilakukan adalah

dengan memilih dua posisi *(locus)* dari kromosom dan kemudian nilainya saling dipertukarkan.

Tabel 2.7. Contoh Mutasi pada pengkodean permutasi

Kromosom sebelum mutasi	1 2 3 4 5 6 7 8 9
Kromosom setelah mutasi	1 2 7 4 6 5 8 3 9

3. Mutasi dalam pengkodean nilai
   Mutasi pada pengkodean nilai hampir sama dengan yang dilakukan pada pengkodean biner, tetapi yang dilakukan bukan menginversi nilai bit. Penerapannya bergantung pada jenis nilai yang digunakan. Sebagai contoh untuk nilai riil, proses mutasi dapat dilakukan seperti yang dilakukan pada pengkodean permutasi, dengan saling mempertukarkan nilai dua gen pada kromosom.
4. Mutasi dalam pengkodean pohon
   Mutasi dalam pengkodean pohon dapat dilakukan antara lain dengan cara mengubah operator (+, -, *, /) atau nilai yang terkandung pada suatu *verteks* pohon yang dipilih. Atau, dapat juga dilakukan dengan memilih dua verteks dari pohon dan saling mempertukarkan operator atau nilainya.

Tidak setiap gen selalu dimutasi tetapi mutasi dikontrol dengan probabilitas tertentu yang disebut dengan *mutation rate* (probabilitas mutasi) dengan notasi Pm. Jenis operator mutasi antara lain:

1. Mutasi Terarah

Mutasi terarah tergantung dari informasi gen. Informasi gen tersebut berupa nilai pelanggaran gen *(violation gen)*. Ini berarti bahwa setiap gen mempunyai peluang yang berbeda untuk terjadi mutasi. Gen yang mempunyai nilai pelanggaran yang lebih besar maka gen tersebut mempunyai peluang untuk terjadi mutasi. Mutasi ini menghubungkan nilai pelanggaran relatif (nilai pelanggaran suatu gen dibagi dengan nilai pelanggaran total suatu kromosom) dengan probabilitas terjadinya mutasi dari suatu gen pada kromosom. Hubungan tersebut dinyatakan secara matematis sebagai berikut:

$$nr(i) = \frac{n(i)}{1 + n_{total}} \quad \text{2-1}$$

$$pm(i) = \left(1 + nr(i)\right)^2 pm \quad \text{2-2}$$

Keterangan persamaan:

nr(i) : nilai pelanggaran *relative* gen ke-i.

n_{total} : nilai pelanggaran total kromosom.
pm(i) : probabilitas mutasi gen ke-i.
pm : probabilitas mutasi.

2. Mutasi Biasa

Mutasi biasa tidak tergantung dari informasi gen. setiap gen mempunyai peluang yang sama untuk terjadi mutasi.

2.3.4 Parameter Algoritma Genetika

Dalam proses pengoperasian algoritma genetik dibutuhkan 4 parameter yaitu:

2.3.4.1 Ukuran Populasi

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

2.3.4.2 Jumlah Generasi

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

2.3.4.3 Probabilitas *Crossover*

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.

2.3.4.4 Probabilitas Mutasi

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.

2.4 Penerapan Algoritma Genetika

Menurut Mahmudy (2013) penerapan algoritma genetika dapat dilakukan melalui beberapa urutan berikut:

2.4.1 Generate Populasi Awal

Untuk menerapkan algoritma genetika diawali dengan pembangkitan populasi awal yang akan menjadi kandidat dalam iterasi generasi pertama agar dapat terjadi seleksi alam. Pembangkitan dilakukkan dengan menentukan individu secara acak atau ditentukan oleh prosedur tertentu untuk mendapatkan penyelesaian dengan nilai optimal.

2.4.2 Representasi Kromosom

Menurut Mahmudy (2013), representasi kromosom adalah suatu proses pengkodean dari penyelesaian permasalahan. Solusi permsalahan harus dipetakan (*encoding*) sebagai *string* kromosom yang tersusun dari sejumlah gen yang menjadi variabel-variabel keputusan untuk mendapatkan suatu solusi. Penentuan solusi menjadi kromosom sangat menentukan kualitas dari solusi yang akan dihasilkan. Terdapat beberapa metode representasi kromosom diantaranya representasi biner, representasi *real coded*, representasi integer, representasi matriks, dan representasi permutasi.

2.4.3 *Crossover*

Crossover diawali dengan memilih dua induk (*parent*) secara acak dari populasi. *Crossover* perlu menentukan *crossover rate (cr)* sebagai pernyataan rasio *offspring* yang akan menjadi individu baru dalam populasi. *Offspring* dihasilkan dari *cr* x *popsize* (jumlah populasi). Misalkan ditentukan nilai *cr* adalah 0,5 dan *popsize* sebanyak 9, maka *offspring* yang dihasilkan adalah 0,5 x 9 = 4,5 (dibulatkan menjadi 5) yang dihasilkan dari proses *crossover*. Secara garis besar langkah *crossover* adalah sebagai berikut (Mahmudy, 2013):

1. Memilih dua buah kromosom sebagai *parent.*
2. Secara acak memilih posisi populasi dalam kromosom, agar kromosom *parent* terpisah menjadi dua *segment*.
3. Saling menukar *segment* diantara dua induk untuk menghasilkan kromosom baru yaitu kromosom anak.

2.4.4 Mutasi

Mutasi biasanya dipakai sebagai operator untuk menjaga keragaman populasi (Mahmudy, 2013). Mutasi akan menciptkan individu baru dengan cara melakukan modifikasi satu atau lebih gen dari individu yang sama. Mutasi berfungsi untuk menggantikan gen yang hilang dari populasi selama proses seleksi serta menyediakan gen yang tidak ada dalam populasi awal. Dengan adanya mutasi maka akan meningkatkan variasi dari populasi. Misalkan didalam sebuah mutasi ditentukan nilai *mutation rate (mr)* = 0.2 maka akan ada 0.2x3=0.6 (dibulatkan menjadi 1) *offspring* yang akan dihasilkan dari proses mutasi. Proses melakukan mutasi yaitu setelah melakukan tahapan *crossover.*

2.4.5 *Fitness*

Mahmudy (2013), menjelaskan bahwa fungsi *fitness* ditentukan untuk mengukur seberapa baik suatu individu. Individu dengan nilai *fitness* terbaik diakhir iterasi (generasi) akan tetap bertahan hidup dan dapat dikodekan sebagai solusi terbaik hasil dari persilangan dan mutasi. Maka semakin besar nilai fitnes semakin baik individu tersebut untuk mnejadi calon solusi.

Fungsi *fitness* ditunjukkan pada persamaan 2-3. Individu yang baik memiliki nilai *fitness* yang besar. Sedangkan fungsi hasil analisis regresi akan dikatakan baik apabila memiliki nilai yang *error* dalam jumlah sedikit. Sehingga nilai *fitness* yang baik diperoleh dari perhitungan satu per MSE *(Mean Square Error)*.

$$Fitness = \frac{1000}{\text{MSE}+1} + \frac{1}{\alpha n+1} \qquad 2\text{-}3$$

Keterangan:

MSE = merupakan *error* rata-rata kuadrat dari selisih Y dan Y'.
α = nilai alfa untuk melakukan normalisasi *fitness*.
n = banyaknya nilai 1 dalam sebuah kromosom.

2.4.6 Seleksi

Seleksi adalah tahap pemilihan individu dari himpunan populasi dan *offspring* yang dipertahankan pada generasi berikutnya (Mahmudy, 2013). Peluang suatu individu untuk hidup dengan melakukkan proses seleksi, kriteria individu yang berpeluang untuk terpilih dalam proses seleksi adalah yang memiliki *fitness* lebih besar. Metode seleksi yang sering digunakan adalah *roulette wheel, replacement, elitism* dan *binary tournament.*

2.4.6.1 *Elitism*

Metode seleksi *elitism* bekerja dengan mengumpulkan semua individu dalam populasi (*parent*) dan *offspring* dalam satu penampungan. *popsize* individu terbaik dalam penampungan ini akan lolos untuk masuk dalam generasi selanjutnya. Metode seleksi ini menjamin individu yang terbaik akan selalu lolos. Pseudo-code *elitism selection* disajikan pada Gambar 2.5 sebagai berikut (Mahmudy, 2013):

```
PROCEDURE ElitismSelection
Input:
POP: himpunan individu pada populasi
pop_size: ukuran populasi
OS: himpunan individu anak (offspring) hasil reproduksi
menggunakan crossover and mutasi
Output:
```

```
POP: himpunan individu pada populasi setelah proses seleksi
selesai
/* gabungkan individu pada POP dan OS ke dalam TEMP */
TEMP  Merge (POP,OS)
/* urutkan individu berdasarkan fitness secara ascending */
OrderAscending (Temp)
/* copy pop_size individu terbaik ke POP */
POP CopyBest (Temp, pop_size)
END PROCEDURE
```

Gambar 2.4 *Pseudocode* Elitism

2.5 Regresi Linier Berganda

Analisis regresi linier berganda merupakan hubungan secara linier antara dua atau lebih variabel *independen* (X1,X2,…,Xn) dengan variabel *dependen* (Y). Analisis ini untuk mengetahui arah hubungan antara variabel *independen* dan variabel *dependen* apakah memiliki berhubungan positif atau negatif dan untuk memprediksi nilai dari variabel *dependen* apabila variabel *independen* mengalami kenaikan atau penurunan. Data yang digunakan biasanya berskala interval atau rasio.

Persamaan regresi yang digunakan untuk membuat taksiran atau prediksi mengenai variabel *dependen* disebut persamaan regresi estimasi, yaitu suatu formula matematis yang menunjukkan hubungan keterkaitan antara satu atau beberapa variabel yang nilainya sudah diketahui dengan satu variabel lain yang nilainya belum diketahui.

Analisa regresi berganda adalah sebuah teknik statistik yang dapat digunakan untuk menganalisis hubungan linier satu variabel respond antara beberapa *variabel* predictor. Model regresi linear berganda yang melibatkan *k* buah variabel *predictor* (X_1, X_2, X_3,…. X_k dapat dilihat pada persamaan berikut (Daniel, 1989):

$$y_i = \beta_0 \sum_{j=1}^{p} \beta_j \; x_{ij} + \; \varepsilon_i \qquad 2\text{-}4$$

Taksiran persamaannya adalah

$$\hat{y} = b_0 \sum_{j=1}^{p} b_j \; x_{ij} \qquad 2\text{-}5$$

Apabila dijabarkan akan menjadi persamaan berikut.

$$\hat{y} = b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_n X_n \qquad 2\text{-}6$$

I = 1,2,…. N dan j = 1,2,…p, dengan n adalah banyaknya pengamatan dan (p+1) adalah banyaknya parameter.

Keterangan:

y = variabel respon.

ŷ = taksiran variabel respon.

x = variabel *predictor*.

β = variabel regresi.

b = taksiran variabel regresi.

ε = residual.

Didalam sebuah model regresi tedapat koefisien-koefisien. Koefisien-koefisien ini merupakan nilai duga atau perkiraan didalam model regresi untuk meramalkan kondisi sebenarnya. Koefisien pada model regresi merupakan nilai yang berpeluang terjadi pada variabel Y (variabel *dependen*/ terikat) apabila suatu nilai X (variabel *independen*/ bebas) diberikan. Menurut (Kurniawan, 2008) dalam regresi, koefisien regresi dibedakan menjadi 2 macam, yaitu:

1. Intersep *(intercept)*

   Intersep secara matematis didefinisikan sebagai titik potong antara dua garis dengan sumbu Y pada suatu diagram atau sumbu kartesius saat nilai X=0. Sedangkan definisi secara statistika adalah nilai rata-rata pada *variabel* Y apabila nilai pada variabel X bernilai 0. Dengan kata lain apabilai X tidak memberikan kontribusi, maka secara rata-rata variabel Y akan bernilai sebesar intersep. Nilai instersep hanyalah suatu konstanta yang memungkinkan munculnya koefisien lain didalam model regresi. Intersep tidak selalu dapat atau perlu untuk diinterpretasian. Apabila didalam sebuah pengamatan, pada variabel X tidak mencakup nilai 0 atau mendekati 0, maka intersep tidak memiliki makna yang berarti sehingga tidak perlu untuk diinterpretasikan.

2. *Slope*

   Secara matematis, *slope* merupakan ukuran kemiringan dari suatu garis. *Slope* adalah koefisien regresi untuk variabel X (variabel bebas). Dalam konsep statistika, *slope* merupakan suatu nilai yang menunjukkan seberapa besar kontribusi (sumbangan) yang diberikan suatu variabel X terhadap variabel Y. nilai *slope* dapat pula diartikan sebagai rata-rata pertambahan atau pengurangan yang terjadi pada variabel Y untuk peningkatan sebeasr satu variabel X.

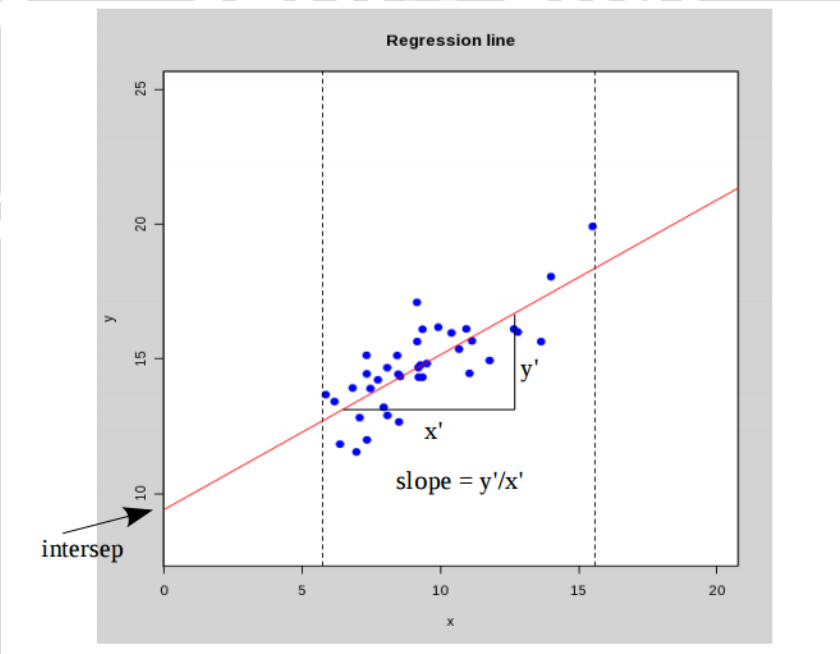
Dari Persamaan 2.6, dapat dicontohkan intersep dan *slope* yang ada pada model regresi, yaitu:

$$Y = 9{,}4 + 0{,}7*X1 = 0{,}5*X2 + \varepsilon$$

Angka 9,4 merupakan intersep, angka 0,7 dan 0,5 merupakan *slope,* sedangkan ε merupakan *error. Error* dalam statistika bukan berarti sesuatu yang rusak atau kacau, melainkan *error* adalah semua hal yang mungkin mempengaruhi variabel terikat Y yang tidak diamati oleh peneliti.

Mean Squared Error (MSE) adalah metode lain untuk mengevaluasi metode prediksi. Masing-masing kesalahan atau sisa dikuadratkan. Kemudian dijumlahkan dan ditambahkan dengan jumlah observasi. Pendekatan ini mengatur kesalahan prediksi yang besar karena kesalahan-kesalahan itu dikuadratkan. Metode itu menghasilkan kesalahan-kesalahan sedang yang kemungkinan lebih baik untuk kesalahan kecil, tetapi kadang menghasilkan perbedaan yang besar. Nilai *mean squared error (MSE)* dapat dilihat pada Persamaan 2-7 berikut (Daniel, 1989).

$$\text{MSE} = \frac{\sum e_i^2}{n} = \frac{\sum (Xi - Fi)^2}{n} \qquad \textbf{2-7}$$



Gambar 2.5. Intersep dan Slope pada Grafik

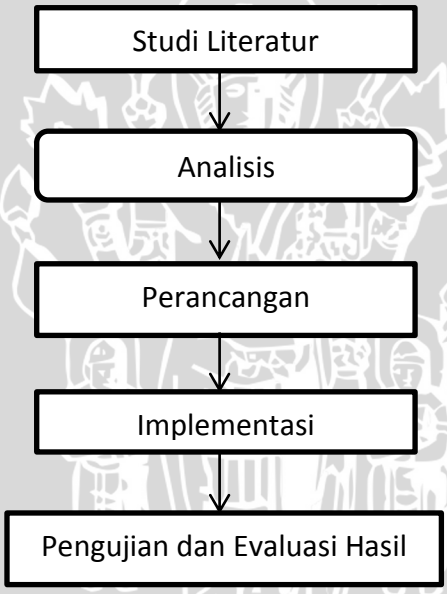
Dalam Gambar 2.5 diatas, dapat dilihat bahwa pada grafik tersebut terdapat sumbu X yang berada pada kisaran angka 5 lebih hingga angka 15 lebih. Hal ini berarti kita hanya diizinkan untuk melakukan prediksi nilai Y untuk nilai X yang berada dalam rentang tersebut. Sebab tidak memiliki dasar yang kuat untuk mengatakan bahwa hubungan variabel X dan Y tetap linier untuk titik-titik data yang mendekati angka nol. Kondisi ini berdampak terhadap interpretasi intersep. Dalam contoh grafik ini data untuk variabel X tidak memuat angka nol atau mendekati nol, intersep dikatakan tidak memiliki makna yang berarti sehingga tidak perlu untuk diinterpretasikan.

BAB 3 METODOLOGI PENELITIAN

Penelitian skripsi yang berjudul "Optimasi Model Regresi Menggunakan Algoritma Genetika untuk Memprediksi Jumlah Mahasiswa yang Mengambil Mata Kuliah" dapat dijadikan sebagai penelitian implementatif yang memberikan solusi untuk mengetahui prediksi jumlah mahasiswa yang akan mengulang pada mata kuliah tertentu. Hasil akhir penelitian ini adalah perangkat lunak yang dapat dimanfaatkan untuk memperoleh jumlah estimasi mahasiswa yang akan mengulan dengan optimasi persamaan regresi linear berganda melalui algoritma genetika.

3.1 Tahapan Penelitian

Langkah-langkah yang dilakukan dalam proses perancangan penelitian implementatif menggunakan analisis regresi yang dioptimasi menggunakan algoritma genetika dapat dilihat pada Gambar 3.1 berikut:



Gambar 3.1. Alur dan Tahapan Penelitian

Sumber : Perancangan

Dari Gambar 3.1 diatas, dapat dijabarkan bahwa alur pengerjaan penelitian ini melalui tahapan-tahapan sebagai berikut:

1. Melakukan studi literature mengenai optimasi menggunakan algoritma genetika yang berhubungan dengan prediksi menggunakan analisis regresi linear berganda.
2. Tahapan analisis membahasa tentang permasalahan yang akan diselesaikan dengan menggunakan optimasi algoritma genetika untuk memprediksi jumlah mahasiswa yang mengulang.

3. Tahapan perancangan merupakan alur penyelesaian yang dilakukan oleh algoritma genetika meliputi langkah-langkah inisiasi populasi awal, *crossover,* mutasi dan seleksi dari individu yang dihasilkan.
4. Tahap implementasi yaitu membuat sistem sesuai dengan analisis dan perancangan yang telah dilakukan pada tahapan sebelumnya.
5. Tahap pengujian dan evaluasi dilakukan terhadap hasil uji coba yang dilakukan sesuai dengan skenario pengujian yang telah ditetapkan.

3.2 Data yang Digunakan

Untuk menyelesaikan permasalahan yang akan dibahas pada penelitian ini, peneliti menggunakan data historit jumlah mahasiswa yang memperoleh nilai C+,C,D+,D dan E dari jangka waktu tertentu. Dalam kasus ini, peneliti menggunakan data hasil *generate* yang merepresentasikan kondisi realistis mahasiswa yang mengulang dalam rentang waktu 24 tahun. Dari data *generate* ini, dapat digunakan untuk membangun model persamaan yang efektif untuk memprediksi jumlah mahasiswa yang akan mengulang.

Untuk penerapan pada kondisi nyata, data hasil *generate* yang merepresentasikan kondisi realistis mahasiswa mengulang dapat diganti dengan menggunakan data sebenarnya. Sehingga hasil prediksi menggunakan analisis regresi linear berganda yang telah dioptimasi menggunakan algoritma genetika tetap akurat.

3.3 Algoritma yang Digunakan

Dalam penyelesaian permasalah untuk memprediksi jumlah mahasiswa mengulang mata kuliah menggunakan Algoritma Genetika seperti yang telah dijelaskan pada sub bab 2.3. Dalam uraian tersebut, algoritma genetika mampu memberikan hasil yang optimal untuk berbagai kebutuhan optimasi dan permasalahan yang kompleks. Diharapkan dengan menggunakan algoritma genetika dan analisi regresi linear berganda ini mampu memberikan solusi pada permasalah penentuan jumlah pengambil mata kuliah.

3.4 Kebutuhan Sistem

Sistem yang dibangun merupakan sebuah sistem berbasis perangkat lunak yang digunakan untuk memprediksi jumlah mahasiswa yang mengulang mata kuliah. Adapun rincian dari kebutuhan sistem ini adalah:

3.4.1 Lingkungan perangkat keras

Perangkat keras yang digunakan dalam mengimplementasikan algoritma genetika pada prediksi mengulang mata kuliah sebagai berikut:

1. Laptop atau komputer processor core dual core 2.53GHz.
2. Memory DDR3 2GB.
3. HDD 320 GB.

3.4.2 Lingkungan perangkat lunak

Aplikasi yang dibangun berbasis desktop ini nantinya akan mampu berjalan melalui *platform* perangkat lunak sebagai berikut:

1. Sistem operasi Windows7.
2. NetBeans IDE 8.1.

3.5 Pengujian Algoritma

Dalam proses pengujian algoritma yang digunakan pada prediksi jumlah mahasiswa pengambil mata kuliah maka dilakukan uji coba pada:

1. Uji coba menentukan populasi yang optimal.
2. Uji coba menentukan banyaknya generasi yang optimal.
3. Uji coba menentukan kombinasi *crossover rate (cr)* dan *mutation rate (mr)* yang terbaik.

3.5.1 Pengujian Ukuran Populasi

Pengujian ukuran populasi yaitu untuk mengetahui ukuran populasi yang tepat untuk menghasilkan koefisien yang terbaik. Dengan menghasilkan koefisien yang terbaik maka akan menghasilkan prediksi jumlah mahasiswa pengambil mata kuliah yang mendekati dengan jumlah sebenarnya. Dari ukuran populasi dapat mempengaruhi kemampuan dari algoritma genetika dalam melakukan perhitungan dan menemukan solusi terbaik.

Dalam perancangan pengujian ukuran populasi ini menggunakan jumlah kelipatan 100 yang dapat dilihat pada Tabel 3.1 dengan parameter pengujian sebagai berikut:

a. Ukuran populasi : 100 – 1000
b. Ukuran generasi : 10
c. *Crossover Rate* : 0,8
d. *Mutation Rate* : 0,2

Tabel 3.1. Rancangan Pengujian Ukuran Populasi

Populasi	Nilai *fitness*										Rata-rata *fitness*
	Percobaan populasi ke-										
	1	2	3	4	5	6	7	8	9	10	
100											
200											

300											
400											
500											
600											
700											
800											
900											
1000											

Sumber: Perancangan

3.5.2 Pengujian Banyaknya Generasi

Pengujian banyaknya generasi yaitu melakukan pengujian untuk mengetahui banyaknya generasi yang tepat untuk menghasilkan koefisien yang terbaik. Sehingga dengan koefisien terbaik akan menghasilkan prediksi jumlah mahasiswa pengambil mata kuliah dengan hasil yang optimal dan mendekati dengan jumlah mahasiswa pengambil yang sebenarnya.

Banyaknya generasi yang digunakan akan mempengaruhi kemampuan algoritma genetika untuk menemukan solusi yang optimal. Banyaknya generasi yang digunakan dalam pengujian ini adalah kelipatan 50 yang dapat dilihat pada Tabel 3.2. Adapun parameter yang digunakan dalam pengujian banyaknya generasi yaitu:

a. Ukuran populasi : hasil populasi terbaik dari pengujian populasi
b. Ukuran generasi : 50 – 500
c. *Crossover Rate* : 0,8
d. *Mutation Rate* : 0,2

Tabel 3.2. Rancangan Pengujian Banyaknya Generasi

Populasi	Nilai *fitness*										Rata-rata *fitness*
	Percobaan generasi ke-										
	1	2	3	4	5	6	7	8	9	10	
50											
100											
150											
200											
250											
300											
350											
400											
450											
500											

Sumber : perancangan

3.5.3 Pengujian Kombinasi *Crossover Rate (cr)* dan *Mutation Rate (mr)*

Pengujian pada kombinasi *cr* dan *mr* dilakukan untuk mengetahui nilai koefisien yang terbaik dari hasil kombinasi *cr* dan *mr.* dengan mengetahui nilai koefisien terbaik, maka akan mempengaruhi pada hasil prediksi jumlah mahasiswa yang mengambil mata kuliah. Pada pengujian ini menggunakan nilai yang berbeda pada parameter algoritma genetika (nilai *crossover rate* dan *mutation rate*). Adapun rancangan pengujian kombinasi *cr* dan *mr* disajikan pada Tabel. 3.3. untuk parameter yang digunakan dalam uji ini yaitu:

a. Jumlah populasi : hasil populasi terbaik dari pengujian populasi
b. Jumlah generasi : hasil generasi terbaik dari pengujian generasi

Tabel 3.3. Rancangan Pengujian Kombinasi *cr* dan *mr*

Kombinasi		Nilai *fitness*										Rata-rata *fitness*
		Percobaan kombinasi *cr* dan *mr* ke-										
cr	*Mr*	1	2	3	4	5	6	7	8	9	10	
1	0											
0,9	0,1											
0,8	0,2											
0,7	0,3											
0,6	0,4											
0,5	0,5											
0,4	0,6											
0,3	0,7											
0,2	0,8											
0,1	0,9											
0	1											

Sumber : Perancangan

BAB 4 PERANCANGAN

4.1 Formulasi Permasalahan

4.1.1 Deskripsi Masalah

Dalam proses penyusunan jadwal perkulihan bukanlah sebuah pekerjaan yang mudah. Ada banyak aspek dan kendala yang mempengaruhi dalam penyusunan jadwal mata kuliah yaitu dosen, mahasiswa, mata kuliah, ruang perkuliahan, slot waktu dan prediksi jumlah mahasiswa yang memiliki minat untuk mengambil mata kuliah tertentu. Setiap aspek tersebut memiliki keadaan yang dapat menjadi masalah dan konflik dalam proses perkuliahan. Misalnya masalah yang dihadapi adalah jumlah kelas yang dibuka tidak sesuai dengan jumlah permintaan dari mahasiswa, sehingga ada mata kuliah tertentu mengalami kekosongan karena sedikitnya minat mahasiswa yang mengambil mata kuliah tersebut. Dilain sisi ada mata kuliah yang sangat diminati oleh mahasiswa sehingga dengan sangat cepat kelas terpenuhi.

Menentukan jumlah perkiraan atau prediksi mahasiswa yang akan mengambil suatu mata kuliah harus dilakukan pada setiap pembuatan jadwal atau saat pergantian semester. Hal ini penting untuk menentukan berapa kelas yang akan dibuka dan nantinya akan mempengaruhi pada proses pembuatan jadwal perkuliahan.

4.1.2 Kebutuhan memprediksi jumlah mahasiswa pengambil mata kuliah

Dalam proses memprediksi jumlah mahasiswa yang mengambil mata kuliah sangat diperlukan. Untuk mengetahui berapa jumlah mahasiswa yang akan mengambil mata kuliah tersebut diperlukan sebuah perhitungan khusus dengan menggunakan sebuah metode yang mampu memprediksi perkiraan jumlah mahasiswa yang akan mengulang mata kuliah yang sebelumnya memiliki nilai C+, C, D+, D dan E yang berpotensi untuk mengulang pada setiap mata kuliah tersebut. Hal ini karena untuk mahasiswa yang sudah mendapatkan nilai A, B+ dan B sudah dipastikan tidak akan mengulang karena nilai mereka sudah baik dan dipastikan lulus, sedangkan mahasiswa yang memperoleh nilai E sudah dipastikan mengulang namun belum dapat diketahui kapan akan mengulang mata kuliah tersebut.

Untuk mahasiswa yang belum mengambil mata kuliah, dapat dilakukan perhitungan sebelumnya dengan mengetahui total mahasiswa yang ada. Karena dapat dipastikan mahasiswa yang belum mengambil mata kuliah akan mengambil mata kuliah untuk menyelesaikan jenjang pendidikannya.

Proses memprediksi jumlah mahasiswa yang mengambil mata kuliah diperlukan dengan cara mengaplikasikan metode analisis regresi linear berganda

yang seanjutnya dilakukan optimasi menggunakan algoritma genetika. Dalam metode analisis regresi linear berganda diperlukan data pada periode sebelumnya yang nantinya digunakan untuk melakukan prediksi tahun yang akan datang. Dalam kasus memprediksi jumlah mahasiswa pengambil mata kuliah ini, dapat diberikan contoh sederhana sebagai berikut:

Contoh kasus : jika diketahui mahasiswa peserta mata kuliah ABC terdiri dari mahasiswa yang baru pertama kali menempuh perkulaiahan ABC dan mahasiswa yang mengulang mata kuliah ABC. Pada akhir semester, mata kuliah ABC diketahui hasil penilaian pembelajaran seperti pada Tabel 4.1 berikut:

Tabel 4.1. Data Perolehan Nilai Mata Kuliah ABC

TMK	M(br)	M(ulg)	J(Mhs)	Perolehan Nilai							
				A	B+	B	C+	C	D+	D	E
1992	130	50	180	16	23	49	40	1	37	6	8
1993	102	43	145	19	5	34	35	16	26	4	6
1994	89	61	150	7	36	41	24	19	11	5	6
1995	91	72	163	11	17	40	38	27	16	6	8
1996	105	52	157	26	9	20	20	61	0	9	12
1997	110	66	176	14	2	58	38	4	41	8	11
1998	121	56	177	17	35	27	51	11	18	8	10
1999	124	51	175	14	27	29	31	5	52	7	10
2000	118	62	180	43	30	38	11	36	10	6	8
2001	119	53	172	38	41	20	24	26	11	6	6
2002	113	63	176	39	0	32	15	41	39	5	5
2003	104	64	168	4	23	15	37	32	46	6	6
2004	118	60	178	48	12	11	46	4	41	10	5
2005	98	76	174	29	23	31	36	13	31	8	4
2006	102	69	171	30	52	24	17	12	13	11	11
2007	109	74	183	18	49	22	43	30	0	12	9
2008	104	82	186	2	2	28	41	63	18	19	13
2009	100	75	175	3	6	48	50	26	9	22	11
2010	93	69	162	12	16	40	13	21	42	11	6
2011	114	88	202	5	10	40	48	43	39	11	6
2012	122	72	194	50	30	9	51	26	12	11	6
2013	131	78	209	47	2	47	24	35	38	10	6
2014	120	74	194	23	2	40	36	42	33	8	9
2015	110	80	190	6	9	31	43	34	45	12	9
1992	131	50	181	25	33	28	23	22	37	7	6

Tabel 4.1 menjelaskan bahwa jumlah mahasiswa yang menempuh perkuliahan ABC pada tahun mata kuliah (TMK) tertentu terdiri dari dua jenis mahasiswa, yaitu mahasiswa baru yang ditandai dengan M(br) dan mahasiswa mengulang M(ulg). Dari jumlah M(br) dan M(ulg) ini akan diketahui jumlah sebenarnya mahasiswa yang menempuh perkuliahan ABC. Tabel 4.1 selanjutnya dimodelkan menjadi model analisis regresi linear berganda. Pada analisis regresi linear berganda akan dimodelkan dengan variabel regresi seperti yang ditunjukkan pada Tabel 4.2 berikut.

Tabel 4.2. Variabel dalam Regresi Linear Berganda

Variabel	Keterangan
Y(u)	Jumlah mahasiswa yang mengulang pada tahun ajaran.
X1	Jumlah mahasiswa yang memperoleh nilai C+ pada Y(u)-1.
X2	Jumlah mahasiswa yang memperoleh nilai C+ pada Y(u)-2.
X3	Jumlah mahasiswa yang memperoleh nilai C+ pada Y(u)-3.
X4	Jumlah mahasiswa yang memperoleh nilai C pada Y(u)-1.
X5	Jumlah mahasiswa yang memperoleh nilai C pada Y(u)-2.
X6	Jumlah mahasiswa yang memperoleh nilai C pada Y(u)-3.
X7	Jumlah mahasiswa yang memperoleh nilai D+ pada Y(u)-1.
X8	Jumlah mahasiswa yang memperoleh nilai D+ pada Y(u)-2.
X9	Jumlah mahasiswa yang memperoleh nilai D+ pada Y(u)-3.
X10	Jumlah mahasiswa yang memperoleh nilai D pada Y(u)-1.
X11	Jumlah mahasiswa yang memperoleh nilai D pada Y(u)-2.
X12	Jumlah mahasiswa yang memperoleh nilai D pada Y(u)-3.
X13	Jumlah mahasiswa yang memperoleh nilai E pada Y(u)-1.
X14	Jumlah mahasiswa yang memperoleh nilai E pada Y(u)-2.
X15	Jumlah mahasiswa yang memperoleh nilai E pada Y(u)-3.

Tabel 4.2 menjelaskan variabel yang akan digunakan untuk analisis regresi dari data historis 3 tahun terakhir pada masing-masing nilai (C+, C, D+, D dan E) yang direpresentasikan dalam notasi X1, X2 dan X3. Pada Tabel 4.3 merupakan tabel yang digunakan untuk melakukan regresi linear berganda. Dalam analisis regresi linear berganda, proses penyelesaian kasus pada Tabel 4.1 dapat diselesaikan dengan cara berikut:

Tabel 4.3. Data Jumlah Mahasiswa Mengulang Mata Kuliah ABC Tahun 1993 Hingga 2015

No	Th	Y(u)	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15
1	T1	43	40	0	0	1	0	0	37	0	0	6	0	0	8	0	0
2	T2	61	35	40	0	16	1	0	26	37	0	4	6	0	6	8	0
3	T3	72	24	35	40	19	16	1	11	26	37	5	4	6	6	6	8
4	T4	52	38	24	35	27	19	16	16	11	26	6	5	4	8	6	6
5	T5	66	20	38	24	61	27	19	0	16	11	9	6	5	12	8	6
6	T6	56	38	20	38	4	61	27	41	0	16	8	9	6	11	12	8
7	T7	51	51	38	20	11	4	61	18	41	0	8	8	9	10	11	12
8	T8	62	31	51	38	5	11	4	52	18	41	7	8	8	10	10	11
9	T9	53	11	31	51	36	5	11	10	52	18	6	7	8	8	10	10
10	T10	63	24	11	31	26	36	5	11	10	52	6	6	7	6	8	10
11	T11	64	15	24	11	41	26	36	39	11	10	5	6	6	5	6	8
12	T12	60	37	15	24	32	41	26	46	39	11	6	5	6	6	5	6
13	T13	76	46	37	15	4	32	41	41	46	39	10	6	5	5	6	5
14	T14	69	36	46	37	13	4	32	31	41	46	8	10	6	4	5	6
15	T15	74	17	36	46	12	13	4	13	31	41	11	8	10	11	4	5
16	T16	82	43	17	36	30	12	13	0	13	31	12	11	8	9	11	4
17	T17	75	41	43	17	63	30	12	18	0	13	19	12	11	13	9	11
18	T18	69	50	41	43	26	63	30	9	18	0	22	19	12	11	13	9
19	T19	88	13	50	41	21	26	63	42	9	18	11	22	19	6	11	13

20	T20	72	48	13	50	43	21	26	39	42	9	11	11	22	6	6	11
21	T21	78	51	48	13	26	43	21	12	39	42	11	11	11	6	6	6
22	T22	74	24	51	48	35	26	43	38	12	39	10	11	11	6	6	6
23	T23	80	36	24	51	42	35	26	33	38	12	8	10	11	9	6	6

Sumber : Perancangan

Keterangan:

T1	:	1993	T9	:	2001	T17	:	2009
T2	:	1994	T10	:	2002	T18	:	2010
T3	:	1995	T11	:	2003	T19	:	2011
T4	:	1996	T12	:	2004	T20	:	2012
T5	:	1997	T13	:	2005	T21	:	2013
T6	:	1998	T14	:	2006	T22	:	2014
T7	:	1999	T15	:	2007	T23	:	2015
T8	:	2000	T16	:	2008			

Dari Tabel 4.3, selanjutnya dilakukan perhitungan dan analisis regresi linear berganda untuk memperoleh nilai koefisien yang nantinya digunakan untuk memprediksi jumlah mahasiswa yang akan mengulang pada mata kuliah tertentu. Hasil dari perhitungan analisis regresi linear berganda dapat dilihat pada Tabel 4.4 berikut:

Tabel 4.4. Hasil Perhitungan Menggunakan Analisis Regresi Berganda

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.89088
R Square	0.793666
Adjusted R Square	0.351523
Standard *Error*	9.031025
Observations	23

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	15	2196.041	146.4027	1.795044	0.221153
Residual	7	570.9159	81.55941		
Total	22	2766.957			

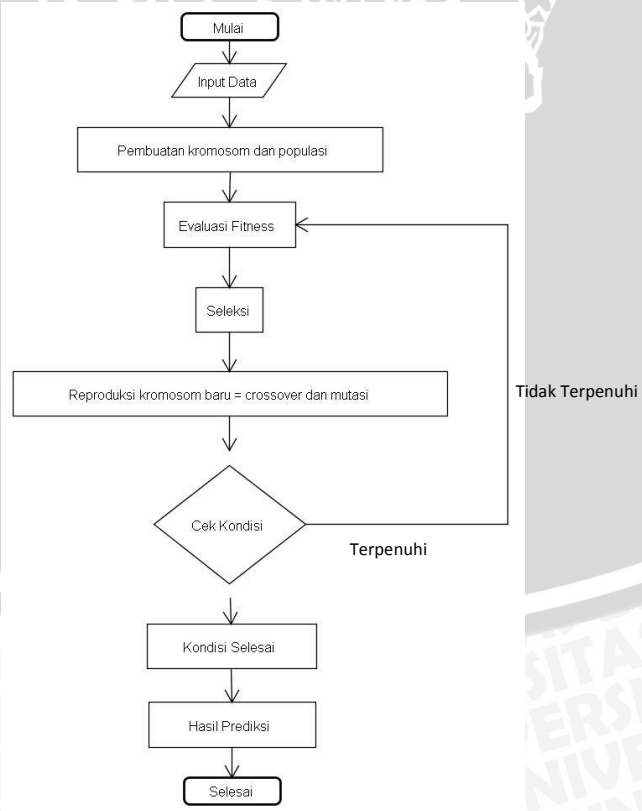
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	45.01933	16.86806	2.66891	0.03205	5.132714	84.90594	5.132714	84.90594
X1	-0.13319	0.240545	-0.5537	0.597017	-0.70199	0.435609	-0.70199	0.435609
X2	-0.00623	0.242347	-0.0257	0.980215	-0.57929	0.566831	-0.57929	0.566831
X3	-0.18959	0.205311	-0.92342	0.386513	-0.67507	0.295896	-0.67507	0.295896
X4	0.188538	0.162894	1.157425	0.285062	-0.19665	0.573721	-0.19665	0.573721
X5	0.019093	0.148446	0.128621	0.901275	-0.33193	0.370112	-0.33193	0.370112
X6	0.040936	0.15989	0.256026	0.805292	-0.33714	0.419016	-0.33714	0.419016
X7	0.067912	0.180348	0.376562	0.717654	-0.35854	0.494368	-0.35854	0.494368

X8	0.086449	0.178817	0.483449	0.643532	-0.33639	0.509284	-0.33639	0.509284
X9	0.381016	0.192932	1.974868	0.088851	-0.0752	0.837229	-0.0752	0.837229
X10	0.114349	1.151845	0.099275	0.923703	-2.60933	2.838031	-2.60933	2.838031
X11	0.95822	1.654588	0.579129	0.580652	-2.95426	4.8707	-2.95426	4.8707
X12	1.39237	1.206874	1.1537	0.286489	-1.46143	4.246172	-1.46143	4.246172
X13	0.300803	1.480049	0.203238	0.84473	-3.19896	3.800563	-3.19896	3.800563
X14	0.682181	1.460598	0.467056	0.654649	-2.77158	4.135947	-2.77158	4.135947
X15	-2.00695	1.221808	-1.6426	0.144467	-4.89606	0.882172	-4.89606	0.882172

Tabel 4.4 menjelaskan bahwa hasil analisis regresi liear berganda menghasilkan koefisien-koefisien yang dibutuhkan dalam melakukan prediksi jumlah mahasiswa mengulang. Dalam analisis regresi linear berganda pada Tabel 4.4, nilai R-Square memiliki nilai 0.793666. Hal ini menunjukkan bahwa terjadi hubungan sebesar 79% yang mempengaruhi antara variabel *independen* (X1 hingga X15) terhadap nilai Y pada variabel *dependen*.

4.2 Siklus Algoritma Genetika

Sistem yang dibangun untuk memprediksi jumlah mahasiswa mengulang mata kuliah adalah sebuah sistem yang dibangun dengan menggunakan algoritma genetika. Dalam proses memprediksi menggunakan algoritma genetika dapat digambarkan pada diagram alir pada Gambar 4.2 berikut ini.



Gambar 4.1. Diagram Algoritma Genetika

Pada Gambar 4.1 dapat dijelaskan bahwa:

1. Input Data
   Proses memasukkan data yang diperlukan oleh pengguna sistem.
2. Pembuatan kromosom dan populasi
   Pada data yang telah dimasukkan, data diolah untuk membuatnya menjadi sebuah kromosom dan populasi sehingga dapat diproses oleh algoritma genetika. Algoritma genetika akan memberikan alternatif solusi, alternatif solusi ini dikodekan dalam bentuk kromosom. Pada masing-masing kromosom berisi sejumlah gen yang mengkodekan informasi pada data yang telah dimasukkan.
3. Evaluasi *Fitness*
   Algoritma genetika selanjutnya akan melakukan evaluasi data kromosom terhadap nilai *fitness* yang digunakan.
4. Seleksi
   Pada tahapan seleksi ini bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling *fit.* Pada tahap seleksi akan menentukan individu-individu mana saja yang akan dipilih untuk mendapatkan generasi baru. Langkah pertama yang dilakukan dalam seleksi ini adalah dengan melakukan pencarian nilai *fitness.* Masing-masing individu dalam suatu kondisi seleksi akan menerima kesempatan bereproduksi yang bergantung pada nilai *fitness.* Sehingga nilai *fitness* inilah yang dipakai pada tahapan seleksi berikutnya.
5. Reproduksi kromosom baru yaitu melalui *crossover* dan mutasi
   Merupakan proses pembangkitan sejumlah populasi baru dengan menggunakan kombinasi metode *crossover rate (cr)* dan *mutation rate (mr).*
6. Cek kondisi
   Merupakan pengecekan pada kondisi yang ada saat ini, apakah kondisi sudah sesuai dengan hasil yang diharapkan. Apabila hasil dari kondisi belum sesuai, maka akan diulang kembali menuju evaluasi *fitness.*
7. Kondisi selesai
   Kondisi ini mengidentifikasikan bahwa kondisi yang dihasilkan sesuai dan hasil yang diinginkan telah terpenuhi.
8. Hasil prediksi
   Merupakan hasil solusi yang diberikan oleh sistem berupa hasil prediksi jumlah mahasiswa yang akan mengulang mata kuliah yang diprediksi.
9. Selesai

Tahapan pengerjaan oleh sistem telah selesai karena sistem telah menghasilkan hasil prediksi pada tahapan sebelumnya.

4.3 Siklus Penyelesaian Masalah Menggunakan Algoritma Genetika

Siklus penyelesaian masalah dengan menggunakan algoritma genetika dimulai dengan mendefinisikan masalah yang akan diselesaikan yang dilanjutkan dengan mengkodean kromosom, inisiasi populasi awal, reproduksi, evaluasi, dan seleksi. Dari langkah-langkah tersebut akan ada banyak parameter yang digunakan dalam algoritma genetika, yaitu individu *(popsize),* jumlah generasi, nilai *mutation rate* (*mr*) dan nilai *crossover rate*(*cr*).

Adapun siklus yang dilakukan dalam menyelesaikan permasalahan menggunakan algoritma genetika dapat dilakukan dengan langkah-langkah berikut:

1. Inisialisasi parameter algoritma genetika yang akan digunakan, inisialisasi yang digunakan dalam algoritma genetika diantaranya:
   a. Jumlah individu dalam setiap populasi *(popsize).*
   b. Jumlah generasi.
   c. *Crossover Rate (cr).*
   d. *Mutation Rate (mr).*
2. Menentukan representasi kromosom yang akan digunakan, dalam representasi kromosom dalam penelitian ini menggunakan representasi kromosom bilangan biner.
3. Membangkitkan populasi awal secara random sesuai dengan representasi kromosom yang telah dipilih sebanyak *popsize* yang diinginkan.
4. Melakukan perhitungan *fitness* dari populasi awal.
5. Melakukan reproduksi yang menghasilkan kromosom-kromosom baru dengan proses *crossover* dan mutasi.
6. Melakukan proses evaluasi dengan menggabungkan kromosom lama dan kromosom baru hasil reproduksi kemudian dilakukan perhitungan *fitness* terhadap kromosom baru.
7. Memilih individu baru yang akan dipertahankan untuk generasi selanjutnya dari kumpulan kromosom yang disebut dengan proses seleksi.

Dalam contoh perhitungan menggunakan algoritma genetika ini, inisialisasi parameter yang digunakan adalah sebagai berikut:

a. Jumlah generasi = 1
b. Ukuran populasi *(Popsize)* = 10
c. *Crossover rate (cr)* = 0.8

d. *Mutation rate (mr)* = 0.2

4.3.1 Representasi Kromosom dan Perhitungan *Fitness*

Pada prediksi mahasiswa dilakukan terhadap mahasiswa yang masih ragu-ragu, yaitu mahasiswa yang sudah mengambil matakuliah tertentu yang ditawarkan, akan tetapi mendapat nilai yang belum memuaskan yaitu nilai C+, C, D+, D dan E, sehingga masih ada kemungkinan untuk mengulang. Sedangkan mahasiswa yang belum mengambil matakuliah hampir pasti akan mengambil matakuliah tersebut.

Representasi kromosom yang akan digunakan yaitu kromosom dengan representasi bilangan biner (nilai 0 dan 1). Dalam representasi kromosom, panjang kromosom yang digunakan yaitu sepanjang 15. Contoh representasi kromosom yang dibentuk sebagai berikut:

Kromosom individu : 00000 00001 10100

Dalam representasi kromosom dengan menggunakan bilangan biner diatas menjelaskan banyaknya jumlah variabel yang mempengaruhi jumlah mahasiswa yang akan mengulang mata kuliah tertentu. Nilai 1 menyatakan bahwa variabel tersebut mempengaruhi mahasiswa untuk mengulang mata kuliah dan nilai 0 berarti variabel tersebut tidak mempengaruhi jumlah mahasiswa yang mangambil mata kuliah.

Dalam algoritma genetika, pada setiap individu dikenal pula memiliki *fitness* atau kebugaran. Nilai *fitness* pada setiap individu menunjukkan kebugarannya, semakin besar nilai *fitness* maka semakin baik pula individu tersebut. Sehingga individu yang memiliki nilai *fitness* terbesar akan memiliki peluang lebih besar untuk terpilih. Pada gen kromosom yang memiliki nilai 1, maka akan merepresentasikan faktor *dependent* (X1 hingga X15) untuk diproses dan dilibatkan dalam perhitungan untuk mencari nilai koefisien analisis regresi linear berganda. Namun jika nilai gen pada kromosom memiliki nilai 0, maka nilai faktor *dependent* (X1 hingga X15) tidak diikut sertakan dalam perhitungan mencari nilai koefisien analisis regresi linear berganda.

Untuk melakukan perhitungan nilai *fitness* dapat menggunakan *pseudocode* yang ditunjukkan pada Tabel 4.4 berikut:

Tabel 4.5. *Pseudocode* Perhitungan Nilai *Fitness*

No	*Pseudocode*
	PROCEDURE *fitness* Input: Idx — index baris yang dihitung *fitness*nya Tabel_idx — tabel himpunan *offspring*/*parent* Tabel — tabel himpunan data historis jumlah mahasiswa mengambil mata kuliah

```
Output:
For a = 0 To baris Do  // looping sebanyak data
        For int b=0 to kolom Do
                If (b==0) awal = 1*tabel_idx[idx, b]
                END IF

                ELSE
                Awal = tabel[a,b] * tabel_idx[idx,b]
                        hasilY += awal
                END ELSE
                //menghitung selisih prediksi dan kondisi nyata
                hasilE = (hasilY-tabel[a,0])^2

                //menjumlah hasil selisih tiap baris
                Sum += hasilE
        END For
        Hasil = 1/((math.sqrt(sum))/baris)
END FOR
END PROCEDURE
```

Pada Tabel 4.4 diatas, dapat dijelaskan bahwa proses perhitungan yang terjadi adalah:

1. Pada tahap awal, perlu memasukkan index kromosom dan tabel data.
2. Melakukan proses *looping* (perulangan) sebanyak jumlah baris data pada tabel untuk menghitung nilai *error*.
   a. Menghitung prediksi jumlah mahasiswa yang mengambil mata kuliah yang didapatkan dari kromosom masukan yang sesuai dengan index kromosom, dan persamaan analisis regresi linear berganda dari semua data.
   b. Menghitung selisih prediksi dari setiap data dengan jumlah pengambil mata kuliah aslinya dan dikuadradkan.
   c. Menjumlahkan nilai hasil selisih prediksi pada setiap baris data.
3. Menghitung nilai *fitness* yang didapatkan dari membagi 1 dengan akar dari hasil penjumlahan setiap data yang dibagi dengan banyaknya data.

Dalam proses menghitung nilai *fitness* dari sebuah individu, dapat dilakukan sesuai dengan Persamaan 2.3. Sebelum melakukan perhitungan pada nilai *fitness,* terlebih dahulu semua prediksi dilakukan perhitungan menggunakan persamaan regresi linier berganda untuk memperoleh nilai koefisien yang digunakan untuk mencari nilai Y' dan nilai *error*. Setelah melakukan perhitungan, barulah nilai *fitness* dapat dicari.

Perhitungan nilai *fitness* pada kromosom P1 dapat diberikan contoh perhitungan sebagai berikut:

Kromosom P1 : 00000 00001 10100

Maka kromosom diatas akan direpresentasikan pada data jumlah mahasiswa seperti pada Tabel 4.6.

Tabel 4.6. Representasi Kromosom Kedalam Data Mahasiswa

	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0
Y(u)	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15
43	40	0	0	1	0	0	37	0	0	6	0	0	8	0	0
61	35	40	0	16	1	0	26	37	0	4	6	0	6	8	0
72	24	35	40	19	16	1	11	26	37	5	4	6	6	6	8
52	38	24	35	27	19	16	16	11	26	6	5	4	8	6	6
66	20	38	24	61	27	19	0	16	11	9	6	5	12	8	6
56	38	20	38	4	61	27	41	0	16	8	9	6	11	12	8
51	51	38	20	11	4	61	18	41	0	8	8	9	10	11	12
62	31	51	38	5	11	4	52	18	41	7	8	8	10	10	11
53	11	31	51	36	5	11	10	52	18	6	7	8	8	10	10
63	24	11	31	26	36	5	11	10	52	6	6	7	6	8	10
64	15	24	11	41	26	36	39	11	10	5	6	6	5	6	8
60	37	15	24	32	41	26	46	39	11	6	5	6	6	5	6
76	46	37	15	4	32	41	41	46	39	10	6	5	5	6	5
69	36	46	37	13	4	32	31	41	46	8	10	6	4	5	6
74	17	36	46	12	13	4	13	31	41	11	8	10	11	4	5
82	43	17	36	30	12	13	0	13	31	12	11	8	9	11	4
75	41	43	17	63	30	12	18	0	13	19	12	11	13	9	11
69	50	41	43	26	63	30	9	18	0	22	19	12	11	13	9
88	13	50	41	21	26	63	42	9	18	11	22	19	6	11	13
72	48	13	50	43	21	26	39	42	9	11	11	22	6	6	11
78	51	48	13	26	43	21	12	39	42	11	11	11	6	6	6
74	24	51	48	35	26	43	38	12	39	10	11	11	6	6	6
80	36	24	51	42	35	26	33	38	12	8	10	11	9	6	6

Tabel 4.6 menjelaskan bahwa sesuai dengan kromosom P1, *variabel* X1, X2, X3, X4, X5, X6, X7, X8, X9, X12, X14 dan X15 tidak mempengaruhi jumlah mahasiswa yang megulang. Sedangkan X10, X11 dan X13 merupakan variabel yang mempengaruhi besarnya jumlah mahasiswa yang mengulang.

Dengan menggunakan uji analisis regresi linear berganda, maka dari table 4.6 dapat dilakukan pengujian analisis regresi untuk mencari nilai koefisien analisis regresi linear berganda. Dari Tabel 4.6 dapat disajikan data sesuai dengan nilai gen pada kromosom yang bernilai 1, sehingga data yang akan dilakukan pengujian dapat dilihat pada Tabel 4.7.

Tabel 4.7. Tabel Data Mahasiswa Mengulang Sesuai dengan Gen dari Kromosom P1

Y(u)	X10	X11	X13
43	6	0	8
61	4	6	6
72	5	4	6
52	6	5	8
66	9	6	12
56	8	9	11
51	8	8	10
62	7	8	10

53	6	7	8
63	6	6	6
64	5	6	5
60	6	5	6
76	10	6	5
69	8	10	4
74	11	8	11
82	12	11	9
75	19	12	13
69	22	19	11
88	11	22	6
72	11	11	6
78	11	11	6
74	10	11	6
80	8	10	9

Data Tabel 4.7 dilakukan analisis regresi linear berganda dan menghasilkan data hasil uji seperti pada Tabel 4.8.

Tabel 4.8. Hasil Pengujian Analisis Regresi Linear Berganda

Regression Statistics	
Multiple R	0.691742
R Square	0.478506
Adjusted R Square	0.396165
Standard *Error*	8.714628
Observations	23

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	3	1324.007	441.3355	5.811272	0.005405
Residual	19	1442.95	75.94474		
Total	22	2766.957			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	61.08322	6.703138	9.11263	2.3E-08	47.05339	75.11305	47.05339	75.11305
X10	0.693044	0.732201	0.946521	0.355766	-0.83947	2.225558	-0.83947	2.225558
X11	1.186695	0.591134	2.007488	0.059135	-0.05056	2.423954	-0.05056	2.423954
X13	-1.36421	0.88278	-1.545355	0.138754	-3.21189	0.483471	-3.21189	0.483471

Tabel 4.8 hasil analisis regresi linear berganda memperoleh nilai Intercept, koefisien X10, X11 dan X13. Koefisien-koefisien ini nantinya yang akan digunakan untuk mencari nilai Y' (Y estimasi). Sehingga nilai Y' dapat dicari

dengan persamaan 2.6. Uraian proses mencari nilai Y estimasi dapat dilihat pada proses berikut:

$Y' = a + b_0 + b_1X_1 + b_2X_2 + b_3$

$Y' = 61.08322 + (0.693044 * \text{X10}) + 1.186695 * \text{X11} - 1.36421 * \text{X13}$

Dari perhitungan diatas apabila diaplikasikan secara keseluruhan pada data mahasiswa, maka dapat dilihat nilai Y' estimasi sebagai berikut.

Tabel 4.9. Nilai Y' Estimasi pada Kromosom P1

Y(u)	X10	X11	X13	Y'
43	6	0	8	54.33
61	4	6	6	62.79
72	5	4	6	61.11
52	6	5	8	60.26
66	9	6	12	58.07
56	8	9	11	62.30
51	8	8	10	62.48
62	7	8	10	61.79
53	6	7	8	62.63
63	6	6	6	64.18
64	5	6	5	64.85
60	6	5	6	62.99
76	10	6	5	68.31
69	8	10	4	73.04
74	11	8	11	63.19
82	12	11	9	70.18
75	19	12	13	70.76
69	22	19	11	83.87
88	11	22	6	86.63
72	11	11	6	73.58
78	11	11	6	73.58
74	10	11	6	72.88
80	8	10	9	66.22

Setelah memperoleh nilai Y', barulah dapat diketahui nilai *error* dengan melakukan pengkuadradan pada selisih Y estimasi dan nilai Y sebenarnya. Sehingga nilai *error* yang diperoleh dapat dilihat pada Tabel 4.10 berikut.

Tabel 4.10. Nilai *Error* pada Kromosom P1

Y(u)	X10	X11	X13	Y'	*Error*
43	6	0	8	54.33	128.32
61	4	6	6	62.79	3.21
72	5	4	6	61.11	118.59
52	6	5	8	60.26	68.25
66	9	6	12	58.07	62.88
56	8	9	11	62.30	39.71
51	8	8	10	62.48	131.77
62	7	8	10	61.79	0.05
53	6	7	8	62.63	92.83
63	6	6	6	64.18	1.38
64	5	6	5	64.85	0.72

60	6	5	6	62.99	8.94
76	10	6	5	68.31	59.09
69	8	10	4	73.04	16.30
74	11	8	11	63.19	116.77
82	12	11	9	70.18	139.82
75	19	12	13	70.76	18.01
69	22	19	11	83.87	221.15
88	11	22	6	86.63	1.88
72	11	11	6	73.58	2.48
78	11	11	6	73.58	19.58
74	10	11	6	72.88	1.25
80	8	10	9	66.22	189.98
Total *Error*					1442.95

Dengan mengunakan MSE *(mean square error)* pada persamaan 2-7, dapat diperoleh nilai MSE sebagai berikut:

MSE = $\frac{1}{n}\ x\ total\ error$

MSE = $\frac{1}{23}\ x\ 1442.95$

MSE = 62.73695534.

Nilai 62.73695534, digunakan untuk menentukan besarnya *fitness* pada kromosom P1 sesuai dengan persamaan. Untuk nilai α adalah 0.1.

$$Fitness = \frac{1000}{MSE + 1} + \frac{1}{\alpha n + 1}$$

$$Fitness = \frac{1000}{62.73695534 + 1} + \frac{1}{0.1 * 3 + 1}$$

$$Fitness = 16.45863076923077$$

Dari perhitungan diatas, nilai *fitness* yang diperoleh dari kromosom P1 adalah 15.6894.

4.3.2 Inisiasi Populasi Awal

Pada proses ini yaitu proses untuk membuat populasi awal. Populasi awal didapatkan dari pembentukan kromosom sebanyak jumlah populasi yang telah ditentukan. Ukuran populasi atau *Popsize* yang sudah ditentukan sebelumnya adalah 10, sehingga satu populasi berisi kromosom *(parent)* sebanyak 10.

Pembentukan kromosom dalam bilangan biner dibangkitkan secara acak dengan melakukan *random* nilai pada rentang 0 dan 1 hingga membentuk sekumpulan gen pada kromosom dengan panjang kromosom 15 gen. Contoh pembangkitan kromosom untuk individu dapat dilihat pada Tabel 4.11 berikut.

Tabel 4.11. Pembangkitan Kromosom

Parent	**Kromosom**
P1	1 0 1 0 1 1 0 0 0 0 0 1 1 1 1
P2	0 1 0 1 0 0 0 0 1 0 0 0 0 1 1
P3	1 0 1 0 0 0 0 1 1 0 1 1 1 1 0
P4	0 0 0 1 0 0 0 1 0 0 0 0 1 1 1
P5	1 1 1 0 1 1 0 0 1 0 1 1 0 1 1
P6	1 1 1 0 0 0 1 1 0 0 1 1 1 1 0

P7	1 1 0 1 0 0 1 0 1 1 0 0 1 0 0
P8	0 1 0 1 0 1 1 1 1 1 0 1 1 1 0
P9	0 0 0 1 0 1 0 1 0 0 0 1 1 1 0
P10	0 0 0 1 0 1 1 0 1 0 0 0 1 1 0

4.3.3 Reproduksi

Pada reproduksi ini dilakukan untuk mencari hasil keturunan *(offspring)* dari setiap proses reproduksi yang terjadi pada *parent*. Tahap reproduksi ini ada dua macam yaitu akan dilakukan dengan cara *crossover* dan mutasi.

4.3.3.1 Pindah Silang *(Crossover)*

Proses pindah silang atau *Crossover* dapat dilihat pada Tabel 4.12 berikut:

Tabel 4.12. *Pseudocode* Proses *Crossover*

No	*Pseudo Code*
1	PROCEDURE *crossover*
2	
3	Input:
4	*Popsize* Ukuran populasi
5	jmlCr jumlah *offspring* hasil *crossover*
6	nMinAlfa batas bawah range alfa
7	nMaxAlfa batas atas range alfa
8	koefisien tabel himpunan populasi/*parent*
9	*offspring* tabel himpunan *offspring*/anak
10	
11	Output:
12	FOR a = 0 to jmlCr do
13	For b = 0 to kolom do
14	//membangkitkan nilai alfa sebanyak panjang kromosom
15	$randAlfa_b$ = Random(-100,100)
16	END FOR
17	
18	//memilih 2 *parent* berbeda
19	Par1 = Random (0, (*popsize* – 1))
20	Par2 = Random (0, (*popsize* - 1))
21	
22	//Proses perhitungan
23	FOR b = 0 to kolom Do
24	$Offspring_{a,b}$ = $koefisien_{par1,b}$ + $randAlfa_b$ + $koefisien_{par2,b}$ –
25	$koefisien_{par1,b}$
26	IF (a== (jmlCr – 1)) && ((a % 2) == 0) THEN
27	ELSE
28	$Offspring_{a+1,b}$ = $koefisien_{par2,b}$ + $randAlfa_b$ * $koefisien_{par1,b}$
29	– $koefisien_{par2,b}$
30	END ELSE
31	END IF
32	END FOR
33	$Offspring_{a,\ kolom}$ = *fitness* (a, *offspring*)
34	$Offspring_{a+1,\ kolom}$ =*fitness* (a+1, *offspring*)

35	a+=2
36	END FOR
37	
38	END PROCEDURE

Pada proses *crossover* yang dituliskan pada *pseudocode* Tabel 4.12, yang digunakan untuk menghasilkan *offspring* (kromosom anak) dari proses mengalikan nilai *crossover rate (cr)* dan jumlah populasi *(popsize)*. Populasi awal yang ada yaitu sebanyak 10 kromosom dengan *crossover rate* sebesar 0,5. Sehingga jumlah kromosom 10 akan dikalikan dengan 0,5 dan akan mempengaruhi jumlah anak yang dihasilkan. Dengan nilai *crossover rate* 0,5 maka *offspring* yang dihasilkan adalah 0,5 x 10 = 5 (C1-C5).

Berdasarkan Tabel 4.12 yang berisi *pseudocode* diatas, dapat dijelaskan bahwa:

1. Melakukan proses perkalian jumlah *crossover rate* dengan jumlah populasi untuk memperoleh *offspring*.
2. Dilakukan proses perulangan sebanyak jumlah *offspring.*
3. Dilakukan pembangkitan nilai random *a* pada *range* yang telah ditentukan sebanyak panjang kromosom.
4. Melakukan pemilihan *parent* sebanyak 2 *parent.*
5. Mendapatkan nilai *offspring 1* dan *offspring 2*.

Dalam melakukan *crossover* ini menggunakan model *one cut point* yang ditentukan secara *random* oleh sistem untuk posisi *one cut point.* Dalam proses *crossover* bilangan biner dilakukan dengan menggantikan kromosom 2 induk dengan posisi yang *random.* Contoh *crossover* yang dilakukan yaitu misalkan individu 1 dan individu 3 terpilih sebagai induk yang akan dilakukan *crossover.* Misalkan posisi *one cut point* adalah 8, maka dapat dilihat pada Tabel 4.13 berikut:

Tabel 4.13. Proses *Crossover*

Individu Ke	Kromosom
P1	1 0 1 0 1 1 0 0 0 0 0 1 1 1 1
P3	1 0 1 0 0 0 0 1 1 0 1 1 1 1 0
Hasil *Crossover*	
C1	1 0 1 0 1 1 0 0 1 0 1 1 1 1 0
C2	1 0 1 0 0 0 0 1 0 0 0 1 1 1 1

4.3.3.2 Mutasi *(Mutation)*

Dalam melakukan proses mutasi, metode yang digunakan adalam metode *random mutation.* Proses mutasi ini nantinya akan menghasilkan anak *(offspring)*. Jumlah anak yang dihasilkan dari proses mutasi ini sebanyak jumlah *mutation rate (mr).* Apabila nilai dari *mutation rate (mr)* adalah 0.2 maka

offspring yang dihasilakan adalah 10 x 0.2 = 2 anak. Dalam metode mutasi ini akan menambah atau mengurangi nilai gen terpilih dengan bilangan *random* yang kecil. Berikut adalah *pseudocode* dalam proses menentukan mutasi yang ditunjukkan pada Tabel 4.14.

Tabel 4.14. *Pseudocode* Proses Mutasi

No	*Pseudocode*
1	PROCEDURE mutasi
2	
3	Input:
4	jumCr //banyaknya *offspring* hasil kali
5	*popsize* dan *cr*
6	jum*Mr* // banyaknya *offspring* hasil kali
7	*popsize* dan *mr*
8	
9	//perulangan sebanyak jumlah *offspring* hasil mutasi
10	FOR a=jumCr to (jumCr+jum*Mr*)
11	//memilih induk
12	Inmut = Random(*popsize* – 1)
13	
14	//membangkitkan nilai R sebanyak panjang
15	kromosom
16	For b=0
17	$rdmR_b$ = Random (nilaiMinR, nilaiMaxR)
18	END FOR
19	
20	For b=0 to kolom
21	$Offspring_{a,b}$ = $koefisien_{inmut,b}$ + $rdmR_b$ *
22	(nilaiMax – nilaiMin))
23	END For
24	$Offspring_{a,kolom}$ = *fitness*(a, *offspring*)
25	END FOR
26	END PROCEDURE

Sumber : perancangan

Berdasarkan *pseudocode* pada Tabel 4.14 diatas, proses terjadinya sebuah mutasi dapat dijelaskan sebagai berikut:

1. Melakukan perkalian nilai *mutation rate (mr)* dengan jumlah populasi untuk mendapatkan jumlah *offspring.*
2. Melakukan proses perulangan sebanyak jumlah *offspring.*
3. Memilih induk atau *parent* sejumlah *offspring.*
4. Membangkitkan nilai *random (r)* pada rentang yang telah ditentukan sebanyak panjang kromosom.
5. Mendapatkan nilai *offspring* dari proses mutasi.

Mutasi yang digunakan dalam memprediksi jumlah mahasiswa yang mengulang mata kuliah dilakukan dengan menggunakan *random mutation*. Yaitu melakukan pergeseran kromosom secara *random*. Dalam mutasi ini, nantinya akan menggantikan nilai kromosom pada posisi tertentu dengan nilai lain pada kromosom yang sama namun posisi yang berbeda. Contoh mutasi yang digunakan adalah sebagai berikut:

Posisi mutasi 1 = 5;

Posisi mutasi 2 = 11;

Individu ke 6 = 1 1 1 0 0 0 1 1 0 0 1 1 1 1 0

Offspring C3 = 1 1 1 0 0 1 1 1 0 0 1 1 1 1 0

Pada metode *random mutation* yang dilakukan dengan menambah atau mengurangi nilai gen terpilih dengan bilangan random yang kecil. Dengan melakukan mutasi ini, maka akan diperoleh individu yang memiliki keragaman lebih bervariasi.

4.3.4 Evaluasi dan Seleksi

Untuk mendapatkah hasil yang terbaik, maka dilakukan evaluasi dan seleksi pada hasil reproduksi yang telah ada sebelumnya. Dengan menghitung nilai *fitness* dari hasil reproduksi maka akan diperoleh individu-individu pilihan yang mampu bertahan hidup. Semakin tinggi nilai *fitness,* maka akan semakin baik nilai kromosom tersebut menjadi calon solusi untuk menyelesaikan permasalahan yang ada.

Proses seleksi yang dilakukan yaitu dengan menggunakan metode *replacement selection*. Proses yang dilakukan dalam *replacement selection* yaitu dengan mengganti induk dengan *offsring*-nya (anak) yang memiliki nilai *fitness* yang lebih besar dibandingkan induknya. Proses seleksi yang dilakukan dalam algoritma genetika dapat digambarkan pada *pseudocode* yang dijelaskan pada Tabel 4.15 berikut:

Tabel 4.15. *Pseudocode* Replacement Selection

No	*Pseudocode*
1	`PROCEDURE seleksi`
2	
3	`Input :`
4	`hasilCr  //offspring hasil kali popsize dengan`
5	`cr`
6	`hasilMr  //offspring hasil kali popsize dengan`
7	`mr`
8	`koefisien  //tabel parent`
9	`asal_prnt  //asal parent yang membentuk offspring`
10	
11	`Output:`

```
12  Koefisien  //tabel populasi atau parent yang baru
13
14  //Looping sebanyak jumlah offspring
15  For a=0 to (hasilCr+hasilMr)
16       //periksa setiap perulangan kurang dari
17       jumlah crossover
18       IF a<hasilCr THEN
19            Index = asal_parent a, 0
20            Index2 = asal_parent a, 1
21
22       //cek fitness dari parent 1 lebih kecil    dari
23  parent 2
24       IF koefisien index, kolom < koefisien index2, kolom
25       THEN
26            Subseleksi (a, asal_parent a,0)
27       END IF
28       ELSE subseleksi (a, asal_parent a,1)
29       END ELSE
30  ELSE
31       Index = asal_parent a, 0
32       Subseleksi (a, asal_parent a, 0)
33  END FOR
34  END PROCEDURE
```

Sumber : perancangan

Proses seleksi yang dijabarkan dalam *pseudocode* pada Tabel 4.15 dapat dijabarkan dan diuraikan sebagai berikut:

1. Melakukan proses perulangan sebanyak jumlah *offspring.*
2. Melakukan proses pengecekan terhadap perulangan tahap awal sebanyak jumlah *offspring* dari proses *crossover.*
   a. Jika perulangan awal sebanyak jumlah *offspring* dari proses *crossover* bernilai benar, maka asal *parent* pembentuk *offspring* berjumlah 2.
      1. Melakukan pengecekan nilai *fitness* pada kedua *parent*. Jika *fitness* dari asal *parent* pertama lebih kecil daripada asal *parent* kedua maka akan bernilai benar, maka proses akan masuk ke subseleksi dengan asal *parent* pertama.
      2. Jika nilai *fitness* dari asal *parent* pertama lebih besar daripada asal *parent* kedua maka akan bernilai salah, maka proses seleksi akan masuk pada fungsi subseleksi dengan asal *parent* kedua.
   b. Jika perulangan awal sebanyak jumlah *offspring* dari *crossover* bernilai salah, maka asal *parent* pembentuk *offspring* akan berjumlah 1 dan masuk ke fungsi subseleksi dengan asal *parent*-nya.

Fungsi subseleksi yang digunakan untuk pengecekan nilai *fitness* dapat dilihat pada *pseudocode* yang disajikan pada Tabel 4.16 berikut:

Tabel 4.16. *Pseudocode* Proses Subseleksi

No	*Pseudocode*
1	PROCEDURE subseleksi
2	
3	Input:
4	*Offspring* //tabel himpunan *offspring*
5	atau anak
6	Index_offs //index *offspring*
7	Index_koef //index *parent* asal yang
8	sesuai koefisien atau
9	populasi
10	
11	//cek *fitness offspring* lebih besar dari *fitness*
12	*parent* asal
13	
14	IF $offspring_{[index\_offs,\ kolom]}$ > $koefisien_{[indek\_koef,\ kolom]}$
15	
16	//mengganti kromosom dan *fitness parent* asal
17	dengan *offspring*
18	FOR a = 0 To kolom
19	$Koefisien_{index\_koef,\ a}$ = $offspring_{index\_koef,\ a}$
20	END FOR
21	END IF
22	END PROCEDURE

Sumber : perancangan

Berdasarkan *pseudocode* pada Tabel 4.16 diatas, proses dari fungsi subseleksi yaitu diawali dengan melakukan pengecekan apakah nilai *fitness offspring* yang sesuai dengan masukan index *offspring,* bernilai lebih besar daripada nilai *fitness parent* asal. Jika bernilai benar, maka akan melakukan proses penggantian kromosom dan nilai *fitness offspring.* Penentuan nilai *fitness* dalam kromosom *parent* dan *offspring* menggunakan persamaan 2.3 dengan membagi nilai 1 dengan *mean square error (MSE)*.

Tabel 4.17. Proses Pengolahan *Offspring* Seleksi

Individu	Kromosom	*Fitness*
P1	1 0 1 0 1 1 0 0 0 0 0 1 1 1 1	17.6951
P2	0 1 0 1 0 0 0 0 1 0 0 0 0 1 1	13.1542
P3	1 0 1 0 0 0 0 1 1 0 1 1 1 1 0	23.2197
P4	0 0 0 1 0 0 0 1 0 0 0 0 1 1 1	9.8884
P5	1 1 1 0 1 1 0 0 1 0 1 1 0 1 1	31.4186
P6	1 1 1 0 0 0 1 1 0 0 1 1 1 1 0	19.2383
P7	1 1 0 1 0 0 1 0 1 1 0 0 1 0 0	19.1783
P8	0 1 0 1 0 1 1 1 1 1 0 1 1 1 0	21.4556
P9	0 0 0 1 0 1 0 1 0 0 0 1 1 1 0	13.5013

P10	0 0 0 1 0 1 1 0 1 0 0 0 1 1 0	13.4943
P11	1 0 1 0 0 0 0 1 0 0 0 1 1 1 1	16.981
P12	1 0 1 0 1 1 0 0 1 0 1 1 1 1 0	22.8174
P13	0 1 0 1 0 1 1 1 1 1 0 1 1 0 0	21.4533
P14	1 1 0 1 0 0 1 0 1 1 0 0 1 1 0	19.3291
P15	1 1 1 0 0 0 1 1 0 0 1 1 1 1 1	22.5395
P16	1 1 1 0 0 1 1 1 0 0 1 1 1 1 0	19.8347
P17	0 1 0 1 0 0 0 0 1 0 0 0 1 1 1	13.7129
P18	1 1 1 0 1 1 0 0 1 0 1 1 0 0 1	30.3539
P19	1 1 0 1 0 0 1 0 1 1 0 1 1 0 0	21.6072
P20	0 1 0 0 0 0 0 0 1 0 0 0 0 1 1	10.9467

Dari Tabel 4.17 diatas, selanjutnya dilakukan seleksi dengan melakukan *sorting* nilai *fitness* dari urutan terbesar hingga terkecil. Nilai *fitness* yang memiliki nilai tertinggi akan menjadikan individu tersebut mampu bertahan hidup dan menjadi individu yang terpilih. Hasil seleksi dapat dilihat pada Tabel 4.18 berikut:

Tabel 4.18. Hasil Seleksi *Fitness*

No	Individu	*Fitness*
1	P5	31.4186
2	P18	30.3539
3	P3	23.2197
4	P12	22.8174
5	P15	22.5395
6	P19	21.6072
7	P8	21.4556
8	P13	21.4533
9	P16	19.8347
10	P14	19.3291
11	*P6*	*19.2383*
12	*P7*	*19.1783*
13	*P1*	*17.6951*
14	*P11*	*16.9810*
15	*P17*	*13.7129*
16	*P9*	*13.5013*
17	*P10*	*13.4943*
18	*P2*	*13.1542*
19	*P20*	*10.9467*
20	*P4*	*9.8884*

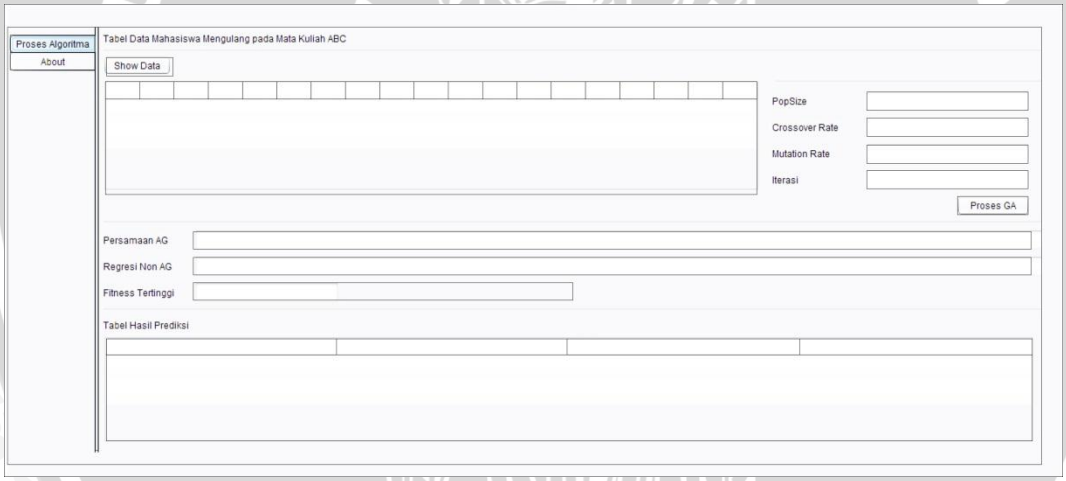
4.4 Perancangan User Interface

Perancangan *user interface* (UI) yang akan diimplementasikan dalam aplikasi ini memiliki dua halaman utama yaitu halaman home dan halaman about. Halaman home memiliki 2 tabel, yaitu tabel data historis mahasiswa yang memperoleh nilai C+, C, D+, D dan E. Tab yang kedua yaitu tab about yang berisi informasi tentang program ini.

4.4.1 Tampilan Halaman Utama

Pada tampilan halaman utama terdapat 2 tab, yaitu tab proses algoritma dan tab *about*. Pada tab proses algoritma, terdapat dua tabel dan kolom masukan dari pengguna sebagai parameter masukan dari pengguna kedalam sistem. Tabel pertama merupakan table yang berisi data mahasiswa yang akan dilakukan optimasi dan table kedua merupakan tabel hasil prediksi dan membandingkan antara nilai asli, hasil prediksi menggunakan analisis regresi linear berganda dan hasil prediksi dari proses optimasi algoritma genetika.

Rancangan halaman utama dari sistem ini dapat dilihat pada Gambar 4.2 berikut ini:

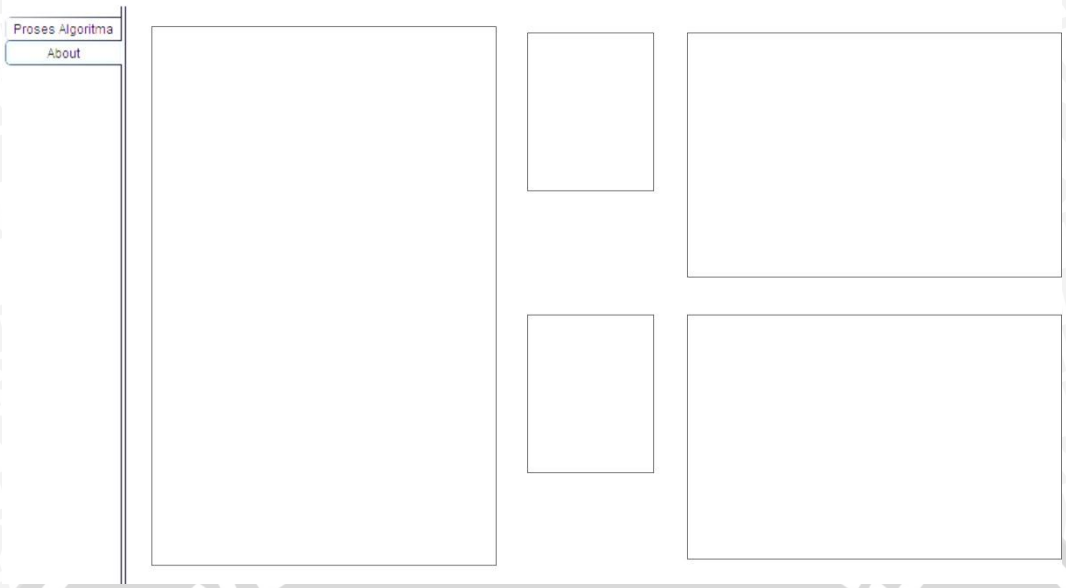


Gambar 4.2. Rancangan Halaman Utama

4.4.2 Tampilan Tab About

Tab about ini digunakan untuk menampilkan profil dan penjelasan singkat tentang aplikasi yang dibangun. Dalam tab ini, terdapat instruksi singkat yang dapat membantu pengguna dalam mengoperasikan aplikasi. Dengan adanya tab About, diharapkan dapat membantu pengguna yang mengalami kesulitan dalam menjalankan aplikasi ini.

Rancangan tampilan tab about dapat dilihat pada Gambar 4.3 berikut:

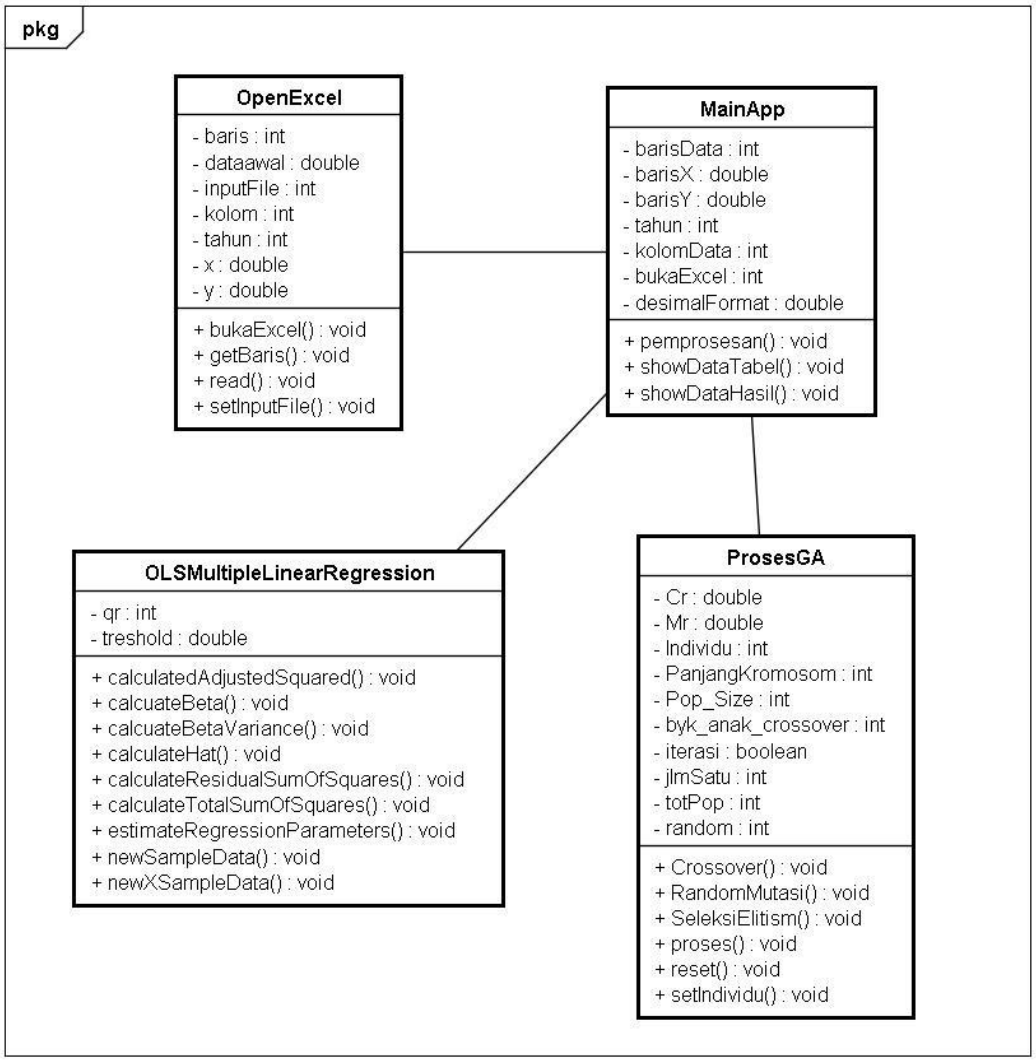


Gambar 4.3. Rancangan Halaman About

BAB 5 IMPLEMENTASI

5.1 Struktur *Class*

Dalam sebuah sistem, sebuah struktur kelas merupakan sebuah gambaran dari sistem yang dibangun. Didalam struktur kelas terdapat atribut dan tipe-tipe data yang digunakan dalam implementasi aplikasi. Dalam aplikasi yang dibangun untuk memprediksi jumlah mahasiswa yang mengulang dapat dilihat pada Gambar 5.1 berikut:



Gambar 5.1. Desain Struktur *Class Diagram*

Pada struktur kelas diagram diatas, terdapat *field* dan *method* yang digunakan untuk melakukan proses prediksi jumlah mahasiswa yang mengulang. Dalam proses optimasi menggunakan algoritma genetika, terdapat proses-proses yang digunakan seperti menentukan inisialisasi individu, melakukan *crossover,* mutasi, mencari nilai *fitness,* melakukan *sorting* pada nilai *fitness* untuk menentukan kromosom terbaik yang nantinya terpiih sebagai solusi terbaik.

5.2 Potongan *Source Code* Utama

Implementasi program untuk memprediksi jumlah mahasiswa yang mengulang dengan menggunakan analisis regresi linear berganda dibangun dengan menggunakan bahasa pe*mr*ograman Java. Dalam proses implementasi terdapat beberapa potongan *source code.* Potongan *source code* yang ada dalam program antara lain:

5.2.1 Formulasi Data Mahasiswa

Formulasi data mahasiswa digunakan untuk menampilkan dan mengolah data awal untuk dilakukan analisis regresi. Data mahasiswa disimpan dalam sebuah berkas excel dengan nama Mhs.xls. Potongan kode yang digunakan untuk memproses data mahasiswa ditampilkan pada *Source Code* 5.1 berikut:

***Source Code* 5.1. Formulasi Data Mahasiswa**

No	*Source Code*

1	public void read() throws IOException {
	File inputWorkbook = new File(inputFile);
	Workbook w;
	try {
	w = Workbook.getWorkbook(inputWorkbook);
	// Get the first sheet
	Sheet sheet = w.getSheet(0);
	// Loop over first 10 column and lines
	baris=sheet.getRows();
	baris=baris-1;
	kolom=sheet.getColumns();
	dataawal = new double[baris][kolom];
	y = new double[baris];
	x = new double[baris][kolom-3];
	
	for (int i = 1; i < sheet.getRows(); i++) {
	for (int j = 0; j < sheet.getColumns(); j++) {
	Cell cell = sheet.getCell(j, i);
	CellType type = cell.getType();
	if (type == CellType.LABEL) {
	//dataawal[i][j]=Integer.parseInt(cell.getContents());
	//System.out.println("I got a label "+ cell.getContents());
	
	}
	
	if (type == CellType.NUMBER) {
	//System.out.println("I got a number "+ cell.getContents());
	
	if(j==2) {
	y[i-1]=Integer.parseInt(cell.getContents());
	}
	if(j>2) x[i-1][j-3]=Integer.parseInt(cell.getContents());
	dataawal[i-1][j]=Integer.parseInt(cell.getContents());
	}

```
36          }
37        }
38      } catch (BiffException e) {
39        e.printStackTrace();
40      }
41    }
42    public static void coba() throws IOException{
43      OpenExcel test = new OpenExcel();
44        test.setInputFile("Mhs.xls");
45        test.read();
46    }
```

5.2.2 Regresi Linear Berganda

Data formulasi mahasiswa yang telah disimpan pada berkas sebelumnya akan diambil dan lakukan analisis regresi linear berganda untuk mencari koefisien-koefisien dalam menentukan prediksi jumlah mahasiswa yang mengulang mata kuliah. Dalam proses mencari nilai koefisien pada analisis regresi linear berganda dapat dilihat pada *Source Code* 5.2 berikut:

***Source Code* 5.2. Proses Regresi Linear Berganda**

No	*Source Code*
1	`public RealMatrix calculateHat() {`
2	`  RealMatrix Q = qr.getQ();`
3	`  final int p = qr.getR().getColumnDimension();`
4	`  final int n = Q.getColumnDimension();`
5	`  Array2DRowRealMatrix augI = new Array2DRowRealMatrix(n, n);`
6	`  double[][] augIData = augI.getDataRef();`
7	`  for (int i = 0; i < n; i++) {`
8	`  for (int j =0; j < n; j++) {`
9	`  if (i == j && i < p) {`
10	`  augIData[i][j] = 1d;`
11	`  } else {`
12	`  augIData[i][j] = 0d;`
13	`  }`
14	`  }`
15	`  }`
16	`  return Q.multiply(augI).multiply(Q.transpose());`
17	`  }`
18	
19	`  public double calculateResidualSumOfSquares() {`
20	`  final RealVector residuals = calculateResiduals();`
21	`  // No advertised DME, args are valid`
22	`  return residuals.dotProduct(residuals);`
23	`  }`
24	
25	
26	`  public double calculateRSquared() {`
27	`  return 1 - calculateResidualSumOfSquares() / calculateTotalSumOfSquares();`
28	`  }`
29	
30	`  @Override`
31	`  protected RealMatrix calculateBetaVariance() {`

```
32      int p = getX().getColumnDimension();
33      RealMatrix Raug = qr.getR().getSubMatrix(0, p - 1 , 0, p - 1);
34      RealMatrix Rinv = new LUDecomposition(Raug).getSolver().getInverse();
35      return Rinv.multiply(Rinv.transpose());
36    }
```

5.2.3 Pembangkitan Generasi Awal

Pembangkitan generasi awal dilakukan dengan melakukan inisialisasi nilai kromosom secara acak sebanyak jumlah *Popsize* yang dimasukkan oleh pengguna pada menu utama. Proses pembangkitan populasi awal dapat dilihat pada *Source Code* 5.3.

***Source Code* 5.3. Pembangkitan Generasi Awal**

No	*Source Code*
1	System.out.println("\n1. inisialisasi individu :");
2	if(!iterasi){
3	for (int i = 0; i < Pop_Size; i++) {
4	for (int j = 0; j < PanjangKromosom; j++) {
5	int BatasBawahRand = 0, BatasAtasRand = 1;
6	Individu[i][j] = Integer.to*String*(random.nextInt(BatasAtasRand -
7	BatasBawahRand + 0) + BatasBawahRand);
8	}
9	}
10	}

5.2.4 *One Cut Point Crossover*

Jenis *crossover* yang digunakan dalam penelitian ini adalah jenis *one-cut point crossover.* Cara yang digunakan dalam metode ini yaitu dengan memilih dua kromosom yang akan dijadikan induk secara acak dan kemudian ditentukan titik potong diantara gen pada kromosom secara acak lalu dilakukan penukaran pada gen-gen setelah posisi titik potong antara *parent 1* dengan *parent* kedua. Jumlah *offspring* yang dihasilkan dari proses *crossover* sejumlah *Popsize x cr* yang dimasukkan oleh pengguna. Metode *crossover* yang digunakan dapat dilihat pada *Source Code* 5.4 berikut.

Source Code* 5.4. *One Cut Point Crossover

No	*Source Code*
1	// Proses *crossover* dilakukan sebanyak (n_*crossover*)
2	int n_*crossover* = (int) Math.ceil((double) byk_anak_*crossover* / 2); // dibagi 2, krn 1
3	kali proses *crossover* akan menghasilkan 2 anak
4	System.out.println("Proses *crossover* dilakukan sebanyak = " + n_*crossover*);
5	
6	*String*[][] Hasil*Crossover* = new *String*[byk_anak_*crossover*][PanjangKromosom];
7	*String*[][] Hasil*Crossover*Temp = new *String*[2][PanjangKromosom];
8	int loop_anak_*crossover* = 0;
9	// membuat *offspring crossover* sebanyak byk_anak_*crossover*
10	for (int i = 0; i < n_*crossover*; i++) {
11	//menampung hasil *Crossover*
12	Hasil*Crossover*Temp = MhsGA.*Crossover*(Individu);

```
13
14          for (int j = 0; j < PanjangKromosom; j++) {
15             Hasil*Crossover*[loop_anak_*crossover*][j] = Hasil*Crossover*Temp[0][j];
16          }
17
18          loop_anak_*crossover* = loop_anak_*crossover* + 1;
19
20          if ((loop_anak_*crossover* + 1) > byk_anak_*crossover*) {
21             break;
22          }
23
24          for (int j = 0; j < PanjangKromosom; j++) {
25             Hasil*Crossover*[loop_anak_*crossover*][j] = Hasil*Crossover*Temp[1][j];
26          }
27
28          loop_anak_*crossover* = loop_anak_*crossover* + 1;
29       }
```

5.2.5 Proses *Random Mutation*

Metode yang digunakan dalam melakukan proses mutasi adalah *random mutation.* Dalam proses *random mutation* dilakukan dengan menambah atau mengurangi gen terpilih dengan bilangan random yang kecil (Mahmudy, 2013). Peluang individu untuk melakukan proses mutasi ditentukan berdasarkan nilai *mutation rate (mr)* yang dimasukkan oleh pengguna pada halaman utama aplikasi. Setelah itu sistem melakukan pemilihan salah satu gen dari kromosom itu untuk dilakukan mutasi. Dari hasil mutasi akan menghasilkan *offspring* baru dengan jumlah sebanyak *Popsize x mr.* Dalam setiap proses mutasi hanya akan menghasilkan satu buah *offspring*. Proses melakukan *random mutation* dapat dilihat pada *Source Code* 5.5.

Source Code* 5.5. Proses *Random Mutation

No	*Source Code*
1	int n_mutasi = byk_anak_mutasi;
2	System.out.println("Proses mutasi dilakukan sebanyak = " + n_mutasi);
3	
4	*String*[][] HasilMutasi = new *String*[byk_anak_mutasi][PanjangKromosom];
5	*String*[][] HasilMutasuTemp = new *String*[1][PanjangKromosom];
6	
7	// membuat *offspring* mutasi sebanyak byk_anak_mutasi
8	for (int i = 0; i < n_mutasi; i++) {
9	//menampung hasil Random Mutasi
10	Hasil*Crossover*Temp = MhsGA.RandomMutasi(Individu);
11	
12	for (int j = 0; j < PanjangKromosom; j++) {
13	HasilMutasi[i][j] = Hasil*Crossover*Temp[0][j];
14	}
15	}

5.2.6 Menentukan Nilai *Error* dan *Fitness*

Nilai *error* diperoleh dari perhitungan dengan mengkuadradkan selisih dari nilai Y estimasi dengan nilai Y sebenarnya. Nilai Y estimasi diperoleh dengan memasukkan gen kromosom yang bernilai 1 dan dicari koefisien pada analisis regresi pada prediksi jumlah mahasiswa yang mengulang. Nilai *error* kemudian dijumlahkan untuk mencari total *error* dari kromosom yang digunakan dalam perhitungan. Setelah didapatkan nilai total *error*, barulah nilai *fitness* dari kromosom dapat dicari. Fungsi *fitness* yang digunakan adalah 1 dibagi dengan *mean square error (MSE)*. Perhitugan total *error* dan *fitness* dapat dilihat pada *Source Code* 5.6.

Source Code* 5.6. Menentukan Nilai *Error* dan *Fitness

No	*Source Code*
1	OLSMultipleLinearRegression reg = new OLSMultipleLinearRegression();
2	System.out.println("");
3	reg.newSampleData(y,x);
4	System.out.println("banyak hasil : "+reg.calculateBeta().getDimension());
5	System.out.println("nilai ke 16 (x16): "+reg.calculateBeta().getEntry(15));
6	System.out.println("semua nilai: "+reg.calculateBeta());
7	System.out.println("sum eror: "+reg.calculateResidualSumOfSquares());
8	System.out.println("sum eror dibagi n:
9	"+(reg.calculateResidualSumOfSquares()/barisData));
10	System.out.println("fitnes :
11	"+(1/((reg.calculateResidualSumOfSquares()/barisData))));

BAB 6 PENGUJIAN DAN PEMBAHASAN

Berdasarkan sub bab 3.5, pengujian algoritma dilakukan sesuai dengan perancangan yang meliputi pengujian ukuran populasi, pengujian banyaknya generasi dan pengujian kombinasi *crossover rate (cr)* dan *mutation rate (mr).* Tahapan-tahapan pengujian dilakukan secara bertahap sesuai dengan urutan pada perancangan sub bab 3.5. Proses pengujian secara detail dapat dilihat pada penjelasan berikut:

6.1 Hasil Pengujian Ukuran Populasi

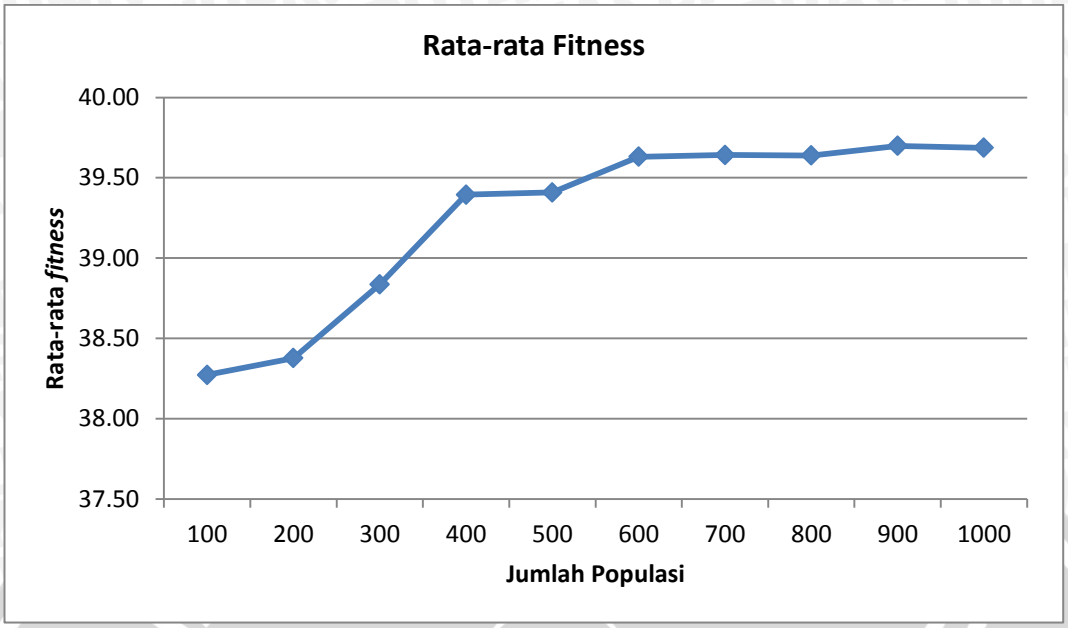
Uji coba ini bertujuan untuk melihat pengaruh dari ukuran populasi atau *Popsize* terhadap perubahan pada nilai *fitness.* Pada pengujian ukuran populasi ini menggunakan parameter banyaknya generasi, *crossover rate,* dan *mutation rate* yang telah ditetapkan pada parameter sebelumnya. Sedangkan pada nilai *popsize* akan diganti secara bertahap dari 50 hingga 500 individu dengan kelipatan 50.

Data historis mahasiswa mengulang yang digunakan pada pengujian ini menggunakan formulasi data hasil *generate* yang merepresentasikan kondisi realistis selama 24 tahun. Jumlah generasi yang digunakan adalah 10 hingga 100 generasi. *Crossover rate* yang digunakan adalah 0.8 dan *mutation rate* adalah 0.2. Setiap *popsize* yang dimasukkan akan diuji sebanyak 10 kali agar dapat diketahui rata-rata *fitness* tersebut. Dari pengujian ini akan diketahui nilai *popsize* yang optimal untuk menyelesaikan malasah pemodelan analisis regresi linear berganda untuk jumlah mahasiswa yang mengulang. Hasil dari pengujian *popsize* dapat dilihat pada Tabel 6.1 berikut:

Tabel 6.1. Hasil Uji Coba *Popsize*

Popu-lasi	Nilai *fitness* Percobaan populasi ke-										Rata-rata *fitness*
	1	2	3	4	5	6	7	8	9	10	
100	37.8819	37.9105	37.5369	39.9395	38.8435	38.0819	39.595	38.4122	37.6175	36.9111	38.273
200	38.0682	39.765	38.8826	39.4389	38.0906	38.4532	36.9324	37.8551	38.6723	37.6175	38.37758
300	38.2171	40.0263	38.0682	39.4388	39.9113	38.7837	38.7593	37.9079	38.9684	38.2871	38.83681
400	39.0074	39.9113	38.2871	39.0021	40.0969	39.7428	38.6911	39.6895	39.6247	39.8917	39.39446
500	38.7259	39.2762	38.5232	39.9412	40.0498	39.9412	39.8917	39.4389	38.3617	39.9412	39.4091
600	39.7291	39.595	39.4855	40.2823	38.5236	40.2823	39.7528	39.9113	39.765	38.9871	39.6314
700	39.4855	40.0498	40.0498	39.7036	39.9412	39.9395	38.407	38.8754	40.0584	39.9113	39.64215
800	40.0584	39.0074	40.0498	39.0021	38.7837	39.9412	39.7036	39.6247	40.2823	39.9395	39.63927
900	39.9113	40.0584	40.2823	39.9395	39.7528	39.4855	40.0584	38.7837	39.0074	39.7036	39.69829
1000	40.2823	38.5232	40.0498	40.2823	39.9113	38.0906	39.765	39.7428	40.2823	39.9412	39.68708

Hasil pengujian pada *popsize* atau ukuran populasi pada tabel 6.1 diperoleh nilai rata-rata *fitness* terbaik pada ukuran populasi 900. Pada populasi 900, nilai rata-rata *fitness* telah mencapai titik optimum. Dari nilai rata-rata *fitness* dapat disajikan dalam bentuk grafik pengujian. Grafik pengujian dapat dilihat pada Gambar 6.1.



Gambar 6.1. Grafik pengujian Ukuran Populasi

Dari gambar 6.1 dapat dijelaskan bahwa pada populasi 100 menjadi populasi yang memiliki nilai rata-rata *fitness* yang terendah, hal ini dikarenakan populasi dengan jumlah 100 belum mencakup secara optimal daerah pencarian pada algoritma genetika. Pada populasi 600 telah terjadi konvergensi hingga populasi 1000. Namun pada populasi 900 menjadi populasi terbaik karena memiliki nilai rata-rata *fitness* yang paling besar, selanjutnya pada populasi 1000 tidak terjadi perbedaan rata-rata nilai *fitness* yang jauh. Hal ini menandakan bahwa telah terjadi konvergensi pada populasi 600 hingga 1000.

Banyaknya populasi yang digunakan akan mempengaruhi lamanya komputasi yang diperlukan, namun dengan populasi yang besar akan menghasilkan nilai *fitness* yang lebih tinggi dan menghasilkan solusi yang semakin baik pula. Pada penelitian sebelumnya yang dilakukan oleh Fakhiroh, D, Mahmudy, WF & Indriati (2015), menghasilkan pengujian dengan pola yang sama dimana mulai terjadi kovergensi pada ukuran populasi tertentu.

6.2 Hasil Pengujian Banyaknya Generasi

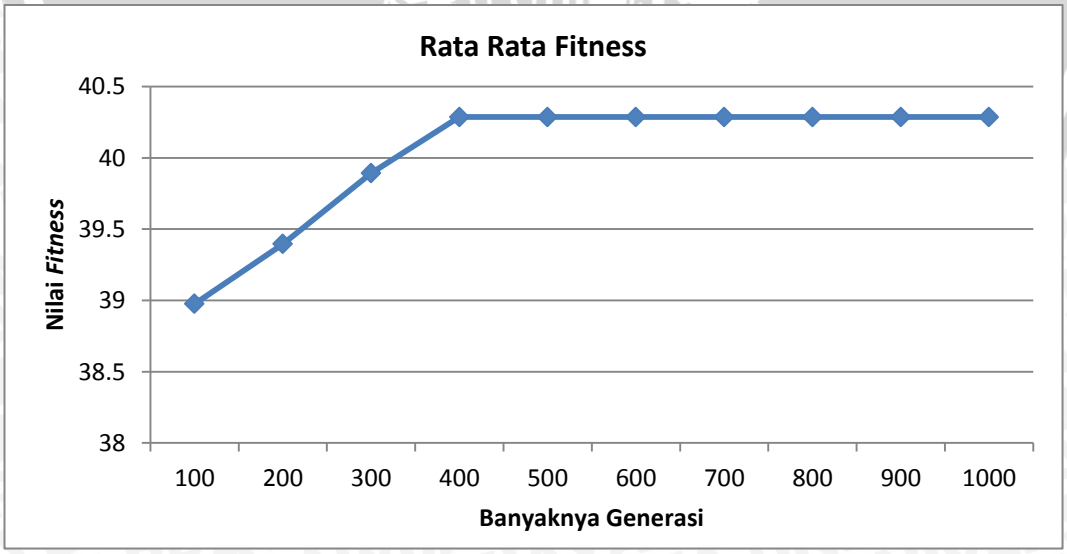
Pada pengujian ini dilakukan untuk mengetahui banyaknya generasi terhadap perubahan nilai *fitness.* Parameter algoritma genetika seperti nilai *popsize, crossover rate,* dan *mutation rate* menggunakan nilai tetap selama proses pengujian ini. Banyaknya generasi yang akan diuji akan diganti secara bertahap dengan kelipatan 100 mulai dari 100 generasi hingga 1000 generasi.

Pada pengujian ini menggunakan data historis jumlah mahasiswa yang memperoleh nilai C+, C, D+, D dan E sebagai masukan system. *Popsize* yang digunakan dalam pengujian ini adalah 100 individu, dengan *crossover rate* sebesar 0.8 dan *mutation rate* 0.2. Setiap generasi akan melalui 10 kali percobaan kemudian dihasilkan rata-rata *fitness* dari percobaan tersebut. Dari hasil pengujian ini akan diketahui banyaknya generasi yang optimal untuk mengetahui faktor yang mempengaruhi jumlah mahasiswa yang mengulang dari model persamaan analisis regresi linear berganda untuk memprediksi jumlah mahasiswa yang mengulang. Hasil pengujian dari uji coba banyaknya generasi dapat diihat pada Tabel 6.2.

Tabel 6.2. Hasil Pengujian Banyaknya Generasi

Generasi	Nilai *fitness* Percobaan generasi ke-										Rata-rata *fitness*
	1	2	3	4	5	6	7	8	9	10	
100	38.0682	38.6911	39.9113	38.5232	38.407	39.7428	40.0584	38.0906	38.7837	39.4855	38.97618
200	38.8754	38.407	39.0021	39.4855	39.7528	39.4855	40.2823	39.9113	38.6911	40.0584	39.39514
300	39.9412	40.2861	40.0498	38.0682	40.2861	39.7428	40.0584	40.2861	39.9113	40.2861	39.89161
400	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861
500	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861
600	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861
700	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861
800	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861
900	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861
1000	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861	40.2861

Rata-rata *fitness* yang diperoleh dari hasil pengujian banyaknya generasi dapat direpresentasikan dalam bentuk grafik untuk melihat hasil pengujian generasi secara visual. Grafik pengujian banyaknya generasi dapat dilihat pada Gambar 6.2 berikut.



Gambar 6.2. Grafik Pengujian Banyaknya Generasi

Dari Gambar 6.2, grafik hasil pengujian banyaknya generasi 400 menunjukkan angka *fitness* yang konstan. Dengan melakukan pengujian sebanyak generasi 400, sudah tidak didapatkan nilai *fitness* yang lebih variatif dikarenakan telah terjadi konvergensi. Generasi 400-1000 telah memiliki rata-rata nilai *fitness* yang sama, yaitu 40,2861.

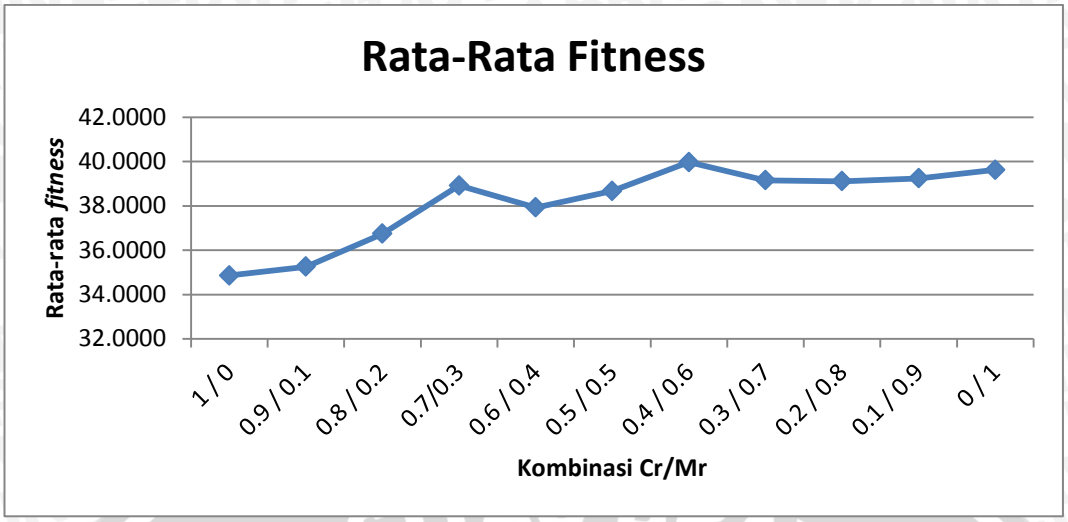
6.3 Hasil Pengujian Kombinasi *Crossover Rate (cr)* dan *Mutation Rate (mr)*

Pengujian untuk kombinasi nilai *crossover rate (cr)* dan *mutation rate (mr)* dilakukan untuk mendapatkan kombinasi yang paling optimal sehingga nilai *fitness* yang diperoleh juga optimal. Parameter algoritma seperti jumlah generasi dan *popsize* telah ditetapkan sesuai dengan data hasil pengujian terbaik pada proses sebelumnya. Untuk proses *crossover rate* dan *mutation rate* akan mengalami pergantian kombinasi secara bertahap dari 0 hingga 1 dengan kelipatan 0.1 setiap perubahan. Hasil pengujian kombinasi *crossover rate* dan *mutation rate* dapa dilihat pada Tabel 6.3 berikut.

Tabel 6.3. Hasil Pengujian Kombinasi *Crossover Rate (cr)* dan *Mutation Rate (mr)*

Kombinasi		Nilai *fitness* Percobaan kombinasi *cr* dan *mr* ke-										Rata-rata *fitness*
cr	*Mr*	1	2	3	4	5	6	7	8	9	10	
1	0	35.0666	35.1666	29.1413	31.6704	36.3151	38.4689	34.3541	33.1078	39.3519	35.9482	34.85909
0,9	0,1	39.0148	38.5683	33.4044	35.6994	33.8017	28.9858	33.2385	36.6327	35.8987	37.2881	35.25324
0,8	0,2	38.5516	35.8577	36.9508	38.7055	33.3586	35.024	40.1925	37.3007	35.2977	36.2001	36.74392
0,7	0,3	40.2854	35.345	39.7159	40.2861	40.1794	40.2861	40.0634	35.6984	40.0285	37.3007	38.91889
0,6	0,4	35.396	33.4662	36.9235	40.2861	39.3519	37.3007	38.5964	38.8551	39.0148	40.0634	37.92541
0,5	0,5	40.2861	40.2854	37.3007	40.0634	36.6401	40.2854	38.6011	35.8316	37.1395	40.2861	38.67194
0,4	0,6	40.2854	40.2861	40.1925	39.7159	39.7159	40.2861	40.0634	40.2861	38.6011	40.2861	39.97186
0,3	0,7	39.7159	38.0496	39.0148	37.3007	38.8551	39.0148	40.2861	38.7055	40.2854	40.2861	39.1514
0,2	0,8	38.6011	40.0634	40.2861	40.2854	38.5942	40.2861	40.0634	35.172	37.446	40.2854	39.10831
0,1	0,9	40.2861	37.3007	40.2861	39.0148	35.345	40.2861	40.2861	40.2861	40.2854	39.0148	39.23912
0	1	40.2854	40.2861	40.1925	35.345	40.2861	38.8551	40.1925	40.2861	40.2861	40.1925	39.62074

Dari hasil pengujian Kombinasi *Crossover Rate (cr)* dan *Mutation Rate (mr)* pada Tabe 6.3 dapat disajikan dalam bentuk grafik untuk menggambarkan fluktuasi dari hasil pengujian kombinasi *crossover rate* dan *mutation rate* terhadap nilai rata-rata *fitness.* Grafik hasil pengujian dapat dilihat pada Gambar 6.3 berikut:



Gambar 6.3. Grafik Hasil Pengujian Kombinasi *Crossover Rate* (*cr*) dan *Mutation Rate* (*mr*)

Dari Gambar 6.3 diatas dapat diketahui bahwa kombinasi dari *crossover rate* dan *mutation rate* dapat diketahui bahwa kombinasi *cr* = 0.4 dan *mr* = 0.6 menjadi kombinasi terbaik dari hasil pengujian yang dilakukan.

6.4 Hasil Pengujian Perbandingan Hasil Prediksi dari Analisis Regresi dan Optimasi Algoritma Genetika

Pengujian dari hasil prediksi jumlah mahasiswa dengan menggunakan metode analisis regresi linear berganda dan hasil optimasi menggunakan algoritma genetika ini menggunakan beberapa parameter yang berbeda. Seperti jumlah individu, kombinasi *cr* dan *mr*, serta banyaknya iterasi yang digunakan.

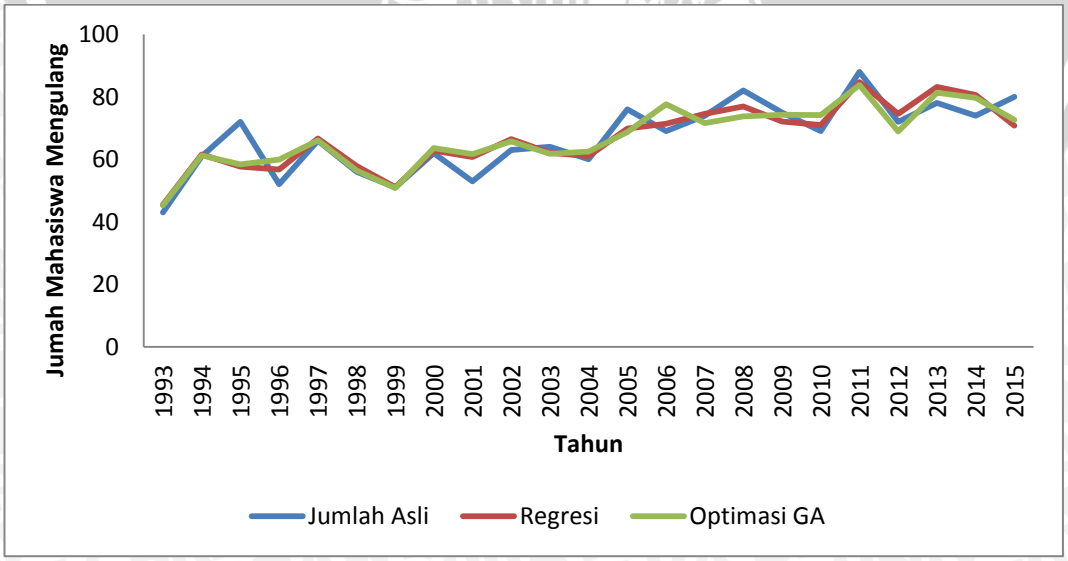
Pengujian dengan jumlah populasi sebanyak 50 populasi, *crossover rate* 0.4, *mutation rate* 0.6 serta iterasi sebanyak 1 kali menghasilkan hasil seperti pada Tabel 6.4 berikut ini.

Tabel 6.4 Perbandingan Hasil Hasil Pengujian ke 1

Tahun	Jumlah Asli	Hasil Prediksi Regresi	Hasil Optimasi Algoritma Genetika	*Error* Analisis Regresi	*Error* Optimasi Algoritma Genetika
1993	43.0	6.43839876	6.43839876	6.43839876	6.43839876
1994	61.0	2.11586116	2.11586116	2.11586116	2.11586116
1995	72.0	191.020041	191.020041	191.020041	191.020041
1996	52.0	23.13032836	23.13032836	23.13032836	23.13032836
1997	66.0	0.07458361	0.07458361	0.07458361	0.07458361
1998	56.0	3.24828529	3.24828529	3.24828529	3.24828529
1999	51.0	0.61921161	0.61921161	0.61921161	0.61921161
2000	62.0	2.47779081	2.47779081	2.47779081	2.47779081
2001	53.0	61.99515169	61.99515169	61.99515169	61.99515169

2002	63.0	14.22496656	14.22496656	14.22496656	14.22496656
2003	64.0	5.16107524	5.16107524	5.16107524	5.16107524
2004	60.0	1.14233344	1.14233344	1.14233344	1.14233344
2005	76.0	50.68585636	50.68585636	50.68585636	50.68585636
2006	69.0	5.32271041	5.32271041	5.32271041	5.32271041
2007	74.0	0.11985444	0.11985444	0.11985444	0.11985444
2008	82.0	30.26980324	30.26980324	30.26980324	30.26980324
2009	75.0	6.70913604	6.70913604	6.70913604	6.70913604
2010	69.0	4.82109849	4.82109849	4.82109849	4.82109849
2011	88.0	10.84516624	10.84516624	10.84516624	10.84516624
2012	72.0	8.34285456	8.34285456	8.34285456	8.34285456
2013	78.0	33.31021225	33.31021225	33.31021225	33.31021225
2014	74.0	32.94989604	32.94989604	32.94989604	32.94989604
2015	80.0	81.53728804	81.53728804	81.53728804	81.53728804
Jumlah *error*				570.917	576.562

Pada hasil pengujian Tabel 6.4, diperoleh selisih MSE sebesar 5,64 lebih baik pada analisis regresi linear berganda, individu yang terpilih adalah P89 dengan kromosm 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1. Pada kromosom P89 hanya menggunakan 13 variabel independen (nilai X) untuk menghasilkan hasil analisis yang sama akuratnya dengan regresi linear berganda. Apabila dari data pada Tabel 6.4 disajikan dalam bentuk grafik, maka akan terlihat seperti pada Gambar 6.4 berikut. Pada Gambar 6.4 terdapat tiga parameter yaitu jumlah asli, jumlah yang diperoleh dari hasil analisis regresi dan jumlah yang diperoleh dari hasil optimasi algoritma genetika. Pada sumbu X (horizontal) merupakan tahun perkuliahan. Pada bidang vertical (sumbu Y) terdapat parameter jumlah mahasiswa yang mengulang.



Gambar 6.4 Grafik Perbandingan Hasil Hasil Pengujian ke 1

Selanjutnya dilakukan pengujian lebih lanjut untuk membandingkan hasil yang diperoleh dari hasil analisis regresi dan optimasi agoritma genetika terhadap jumlah asli mahasiswa yang mengulang mata kuliah. Pada pengujian kedua, menggunakan parameter jumlah populasi 100, *crossover rate 0.5, mutation rate 0,5* dan dilakukan iterasi sebanyak tiga kali. Hasil dari pengujian kedua dapat diihat pada Tabel 6.5 berikut.

Tabel 6.5 Perbandingan Hasil Hasil Pengujian ke 1

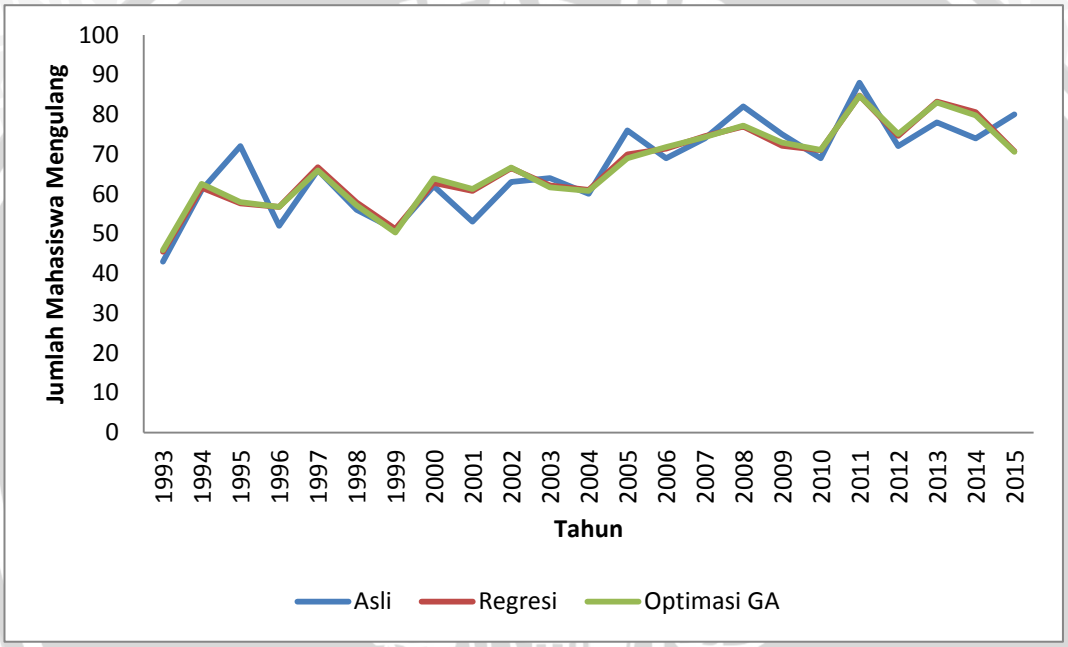
Tahun	Jumlah Asli	Hasil Prediksi Regresi	Hasil Optimasi Algoritma Genetika	*Error* Analisis Regresi	*Error* Optimasi Algoritma Genetika
1993	43	45.4855	45.8633	6.178	8.198
1994	61	61.5776	62.563	0.334	2.443
1995	72	57.6434	57.9549	206.112	197.265
1996	52	56.7298	56.7054	22.371	22.141
1997	66	66.7029	65.9294	0.494	0.005
1998	56	57.8664	57.1126	3.483	1.238
1999	51	51.153	50.3084	0.023	0.478
2000	62	62.7527	63.9037	0.567	3.624
2001	53	60.7494	61.2244	60.053	67.641
2002	63	66.4702	66.6068	12.042	13.009
2003	64	62.1132	61.6653	3.560	5.451
2004	60	61.0202	60.7605	1.041	0.578
2005	76	69.8995	68.9727	37.216	49.383
2006	69	71.3606	71.7842	5.572	7.752
2007	74	74.5188	74.317	0.269	0.100
2008	82	76.9488	77.1884	25.515	23.151
2009	75	72.1463	72.9206	8.144	4.324
2010	69	70.9974	71.0495	3.990	4.200
2011	88	84.7389	84.676	10.635	11.049
2012	72	74.5987	75.1163	6.753	9.711
2013	78	83.2054	83.0001	27.096	25.001
2014	74	80.5943	79.7806	43.485	33.415
2015	80	70.7272	70.597	85.985	88.416
Jumlah *Error*				570.917	578.575

Hasil optimasi menggunakan Algortima Genetika pada Tabel 6.5 memperlihatkan hasil yang hampir sama akuratnya dengan jumlah asli. Pada hasil pengujian optimasi algoritma menggunakan kromosom pada individu 138 dengan gen kromosom 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1. Hal ini menunjukkan bahwa tidak semua faktor *independen* (X1 hingga X15) mempengaruhi jumlah prediksi.

Variabel X5 dan X6 tidak digunakan dalam proses melakukan prediksi jumlah mahasiswa yang mengulang mata kuliah.

Apabila dari data pada Tabel 6.5 disajikan dalam bentuk grafik, maka akan terlihat seperti pada Gambar 6.5 berikut. Pada Gambar 6.5 terdapat tiga parameter yaitu jumlah asli, jumlah yang diperoleh dari hasil analisis regresi dan jumlah yang diperoleh dari hasil optimasi algoritma genetika. Pada sumbu X (horizontal) merupakan tahun perkuliahan. Pada bidang vertical (sumbu Y) terdapat parameter jumlah mahasiswa yang mengulang.

Dari Tabel 6.5 diatas, setelah disajikan dalam bentuk grafik dapat dilihat pada Gambar 6.5 berikut.



Gambar 6.5 Grafik Perbandingan Hasil Hasil Pengujian ke 1

Hasil pengujian hasil 1 dan hasil 2 memperlihatkan bahwa nilai *error* pada analisis regresi linear berganda dan hasil optimasi algoritma genetika menunjukkan bahwa jumlah *error* pada regresi linear berganda lebih kecil dibandingkan dengan hasil optimasi menggunakan algoritma genetika. Semakin besar nilai *error* yang dihasilkan, maka akan memperkecil hasi *fitness* pada setiap individu.

Perlu diperhatikan bahwa pada hasil prediksi menggunakan agoritma genetika tidak menggunakan semua variabel *independen* untuk proses perhitungannya. Sehingga meminimalkan jumlah variabel yang digunakan untuk menghasilkan prediksi yang lebih akurat.

BAB 7 KESIMPULAN DAN SARAN

7.1 Kesimpulan

Kesimpulan yang diperoleh dari hasil implementasi dan pengujian yang telah diakukan dalam penelitian ini adalah:

1. Penerapan metode algoritma genetika dalam melakukan optimasi model regresi linear berganda dengan melakukan pengkodean kromosom secara *binary, crossover* menggunakan *one cut point crossover,* metode permutasi yang digunakan adalah *random mutation* dengan metode seleksi *elitism.*
2. Representasi kromosom terbaik yang dihasilkan dari optimasi model regresi linear berganda menggunakan algoritma genetika adalah 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 dengan selisih *error* sebesar *5*,64 lebih baik pada analisis menggunakan regresi linear berganda dibandingkan dengan hasil otimasi menggunakan algoritma genetika. Namun, pada agoritma genetika tidak menggunakan semua variabel *independen* (variabel X) sebagai variabel yang mempengaruhi prediksi analisis regresi linear berganda. Hanya variabel X yang memiliki kromosom 1 yang digunakan dalam proses perhitungan prediksi jumlah mahasiswa yang mengulang.
3. Parameter yang dihasilkan untuk populasi terbaik yang terpilih adalah 600 dengan kombinasi *crossover rate* 0.4 dan *mutation rate* 0.6 dan banyaknya generasi sebesar 400 generasi.

7.2 Saran

Saran yang dapat diberikan untuk mengembangkan penelitian ini agar menjadi lebih baik adalah:

1. Perlu melakukan uji coba pada algoritma evolusi lain. Sehingga diharapkan memperoleh variasi hasil yang lebih banyak dan memungkinkan menghasilkan hasil yang lebih baik.
2. Perlu melakukan uji kombinasi *crossover rate, mutation rate* yang lebih kompleks dan lebih lengkap untuk menemukan kombinasi nilai *cr* dan *mr* yang paling baik.

DAFTAR PUSTAKA

Arianti, N. (2005). Optimasi Jaringan Syarat Tiruan dengan Algoritma Genetika untuk Peramalan Curah Hujan. IPB Repository.

Daniel, W. W. (1989). Statistika Non Parametrik Terapan. Jakarta: Gramedia.

Goldberg, D.E. (1989). Genetic Algorithm in Search, Optimization and Machine Learning. New York: Addison Weasley Publishing Company Inc.

Kurniawan, D. (2008). Regresi Linier.

Kustiyo, Aziz., Buono, Agus,. & Apriyanti, Novi. (2006). Optimasi Jaringan Syarat Tiruan dengan Algoritma Genetika untuk Peramalan Curah Hujan. IPB Repository Jurnal Ilmu Komputer (74).

Mahmudy, W. F. (2013). Algoritma Evolusi. Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang.

Mahmudy, W. F., Marian, R. M., & Luong, L. H. (2014). Hybrid Genetic Algorithms for Part Type Selection and Machine Loading Problems with Alternatif Production Plans in Flexible Manufacturing System. ECTI Transactions On Computer And Information Technology (ECTI-CIT), 8(1), 80-93.

Permatasari, A. I., & Mahmudy, W. F. (2015). Pemodelan Regresi Linear dalam Konsumsi Kwh Listrik di Kota Batu Menggunakan Algoritma Genetika. DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya, 5(14).

Rahmi, A., Mahmudy, W. F., & Setiawan, B. D. (2015). Prediksi Harga Saham Berdasarkan Data Historis Menggunakan Model Regresi yang Dibangun dengan Algoritma Genetika. DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya, 5(12).

Wahyuni, D. T., Sutojo, T., & Luthfiarta, A. (2014). Prediksi Hasil Pemilu Legislatif DKI Jakarta Menggunakan Naïve Bayes dengan Algoritma Genetika Sebagai Fitur Seleksi.

Wati, S. E., Sebayang, D., & Sitepu, R. (2013). Perbandingan Metode Fuzzy dengan Regresi Linier Berganda dalam Peramalan Jumlah Produksi (Studi Kasus: Produksi Kelapa Sawit di PT. Perkebunan III (PERSERO) Medan Tahun 2011-2012). Saintia Matematika, 1(3), 273–284.

Wu, Jiansheng., Long, Jin., & Liu, Mingzhe. (2012). Evolving RBF Neural Networks for Rainfall Prediction using Hybrid Particle Swarm Optimization And Genetic Algorithm. Neurocomputing Vol 148. pp 136–142