BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Dalam penelitian ini digunakan beberapa sumber pustaka. Pustaka yang relevan pada penelitian ini ditinjau dari sisi kasus penelitian dan metode yang digunakan. Kasus penelitian yang dilakukan adalah mengenai optimasi pada penjadwalan dengan metode AG. Review dan perbandingan dengan penelitian sebelumnya dapat dilihat pada Tabel 2.1.

Tabel 2.1 Tinjauan Pustaka

Penulis	Objek	Variabel	Metode	Output
Ahmat Josi (2017)	Penjadwalan Perkuliahan	-matakuliah -dosen -ruangan	Algoritma Genetika	Laporan jadwal yang sudah tersusun, tanpa adanya bentrok. Prosesnya cepat.
Damayanti, C. P., dkk (2017)	Penjadwalan *Cutomer Service*	-lokasi kerja -shift -pegawai	Algoritma Genetika	Dengan penerapan AG hanya memakan waktu 30% dibandingkan dengan cara manual, dihasilkan nilai *fitness* terbaik sebesar 0.822
Mauludin, S., dkk (2018)	Penjadwalan Kuliah	-jadwal -matakuliah -kelas -dosen -ruang	Algoritma Genetika	Penerapan AG dapat mencapai hasil yang optimal dengan bentrok 0 dengan rata-rata waktu 1,003 detik
Sari, Y., dkk (2018)	Penjadwalan Matakuliah	-jumlah populasi -jumlah generasi -probabilitas crossover -probabilitas mutasi	Algoritma Genetika	Pengalokasian matakuliah, dosen dan ruangan tidak ada bentrok jadwal, jika nilai fitness sama dengan 1.
Sarah Sri Vauziah, Aries Saifudin (2018)	Penjadwalan *Cleaning Service*	-jumlah iterasi -individu -probabilitas crossover -probabilitas mutasi	Algoritma Genetika	AG mampu membuat jadwal dengan baik, menghasilkan nilai fitness 100%
Pande Kadek Cahya Wisnu Murti	Penjadwalan Seminar Pra Skripsi	-status kegiatan -jam -hari	Algoritma Genetika	Implementasi AG mampu untuk mengoptimasi penjadwalan Seminar

Penelitian tentang AG sudah pernah dilakukan oleh Ahmat Josi (2017), dari Jurusan Sistem Informasi, STMIK Prabumulih, Sumatra Selatan dengan judul "Implementasi Algoritma Genetika Pada Aplikasi Penjadwalan Perkuliahan Berbasis Web Dengan Mengadopsi Model Waterfall". Pada Penelitian ini dijelaskan bahwa penjadwalan manual dengan menggunakan Microsoft Excel membutuhkan waktu berhari-hari. Dengan AG maka proses penjadwalan dapat dilakukan secara efektif, secara cepat, tanpa bentrok.

Penelitian yang dilakukan oleh Damayanti dkk (2017) dari Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya dengan judul "Implementasi Algoritma Genetika Untuk Penjadwalan *Customer Service* (Studi Kasus: Biro Perjalanan Kangoroo)". Penelitian ini melakukan uji coba sebanyak 6 kali, diantaranya pengujian berdasarkan ukuran populasi, ukuran generasi, berdasarkan perbandingan crossover dan *mutation rate*, perbaikan bobot, pengaruh konvergensi, dan waktu. Hasil dari penelitian tersebut menunjukan AG dapat mengoptimasi pembuatan jadwal dengan waktu yang dibutuhkan lebih sedikit yaitu 30% dari waktu pembuatan jadwal manual, dan dengan *fitness* terbaik sebesar 0.822368421.

Mauludin, S., dkk (2018) dari Program Studi Manajemen Informatika, FTIK, Universistas Komputer Indonesia melakukan penelitian menggunakan AG dengan judul "Optimasi Aplikasi Penjadwalan Kuliah Menggunakan Algoritma Genetik". Berdasarkan hasil penelitiannya bahwa aplikasi penjadwalan yang dibangun dengan menerapkan AG mampu

menghindari bentrok keseluruhan jadwal yang berarti sangat optimal, dan dilihat dari waktu sangat cepat dengan rata-rata 1,003 detik dari 10 kali percoboaan pembuatan jadwal.

Penelitian yang dilakukan oleh Sari, Y., dkk (2018) dari Program Studi Teknologi Informasi, Universitas Lambung Mangkurat dengan judul "Optimasi Penjadawlan Mata Kuliah Menggunakan Metode Algoritma Genetika Dengan Teknik *Tournament Selection*". Hasil dari penelitian tersebut menunjukan AG kecepatan dalam pembuatan jadwal dengan waktu sekitar 14,7 menit, dibandingakan dengan cara manual membutuhkan hingga 2 hari. Selain itu pengalokasian mata kuliah, dosen dan ruangan tidak ada bentrok jadwal jika nilai fitness sama dengan 1.

Vauziah dkk (2018) dari Program Studi Teknik Informatika Universitas Pamulang telah melakukan penelitian tentang AG dengan judul "Penjadwalan *Cleaning Service* Menggunakan Algoritma Genetika". Berdasarkan hasil penelitianya, AG mampu membuat jadwal dengan baik. Para pekerja mendapatkan hasil yang seimbang. Dengan ukuran paramater iterasi 150, individu 100, probabilitas crossover 0,9 dan probabilitas mutasi 0,04, yaitu menghasilkan nilai fitness 100%.

2.2. Penjadwalan di STMIK AKAKOM dan Secara Umum

Bagian Administrasi Akademik STMIK AKAKOM melakukan penjadwalan untuk setiap kegiatan akademiknya, khusunya pada ujian proposal Pra Skripsi, seminar Pra Skripsi, serta ujian pendadaran Skripsi. Administrasi Akademik merancang jadwal-jadwal tersebut secara manual,

dimulai dari melihat jadwal Mahasiswa yang sedang tidak dalam kegiatan perkuliahan, serta melihat agenda kegiatan akademik dan non akademik dosen yang nantinya akan menjadi pembimbing dan narasumber pada saat ujian. Dengan merancang scara manual banyak permasalahan yang biasanya terjadi, seperti memakan waktu yang relatif banyak dan adanya jadwal yang bentrok.

Menurut Del Pico (2013) penjadwalan adalah proses pengambilan keputusan yang secara logis rumit tetapi secara matematis cukup sederhana. Sebuah panduan langkah demi langkah untuk melalui suatu proyek dengan waktu yang diperlukan untuk melakukan setiap kegiatan atau aktivitas. Banyak jenis penjadwalan beberapa diantaranya rumit, dan ada yang sederhana. Berikut adalah beberapa jenis jadwal umum yang akan dijumpai.

The Checklist dianggap sebagai jenis penjadwalan tidak formal, kadang-kadang juga disebut sebagai *to-do-list*. *The Checklist* bisanya digunakan untuk penjadwalan jangka pendek, jarang lebih dari beberapa hari.

Dalam beberapa penjadwalan, *The Schedule Board* atau *Operation Board* adalah penjadwalan yang berguna. Penjadwaan ini adalah sebuah daftar tugas atau operasi kerja dengan individu yang bertanggungjawab atau tim kerja yang ditugaskan dalam beberapa hari ataupun mingguan.

Salah satu jenis jadwal yang paling umum adalah *The Bar Chart* (diagram batang). *The Bar Chart* menggunakan representasi grafik dari perencanaan kegiatan. Secara tradisional dikembangkan dengan daftar tugas-tugas kronologis di sepanjang sisi kiri dalam format kolom. Sepanjang

baris teratas adalah waktu dalam beberapa hari, minggu, atau bulan. Tugas ditampilkan secara keseluruhan, dan setiap tugas diberikan diagram batang dari awal dimulainya tugas hingga berakhirnya tugas. Kelebihan dari *The Bar Chart* yaitu mampu menampilkan setiap kegiatan dalam bentuk visual.

The Look-Ahead adalah jenis jadwal mikro yang menampilkan jadwal secara detail dari periode waktu singkat dalam jadwal. Penjadwalan ini terfokus pada suatu kegiatan yang akan dilakukan, secara terperinci. Memungkinkan tim kerja untuk menganalisa tugas dan tanggungjawab yang akan datang, dan memastikan bahwa kemajuan terpenuhi. *The Look-Ahead* biasanya dipadukan dengan penjadwalan yang lain, tidak bisa digunakan sendiri.

Jenis penjadwalan lain, yang mengambil keuntungan dari fitur linier dari beberapa kegiatan disebut penjadwalan *Line-of-Balance* atau *velocity diagram*. Jenis penjadwalan turunan dari *The Linear Schedule*. Jadwal *Line-of-Balance* menggunakan grafik visual sederhana yang mereplikasi cara kerja linear dilakukan. Penjadwalan ini mengelompokkan kemajuan kumulatif pekerjaan secara grafis terhadap waktu.

2.3. Algoritma Genetika

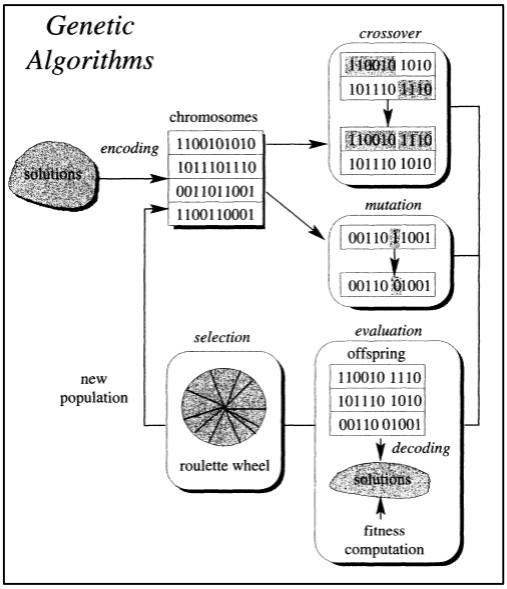
Algoritma Genetika adalah sebuah algoritma yang memiliki bentuk sederhana sehingga disebut dengan Simple GA (SGA). Pada umumnya AG digunakan untuk menyelesaikan masalah optimasi diskrit. Ciri utama dari GA adalah tidak terlalu cepat dalam menemukan solusi optimal, tetapi memiliki heuristik yang baik untuk masalah kombinatorial. Ciri lainnya

adalah GA lebih menitik beratkan pada rekombinasi atau pindah silang (Suyanto, 2008). Pada saat inisialiassi GA membangkitkan kromosom secara acak pada sejumlah individu sebagai suatu populasi. Jumlah individu dalam populasi tersebut selalu tetap selama proses evolusi. Setiap kromosom dikodekan dan kemudian dievaluasi sehingga diperoleh nilai fitness-nya dari setiap individu yang ada. Nilai-nilai fitness ini digunakan sebagai parameter dalam pemilihan orangtua. Artinya, kromosom dengan nilai fitness yang lebih besar akan memiliki peluang yang lebih besar juga untuk terpilih sebagai orang tua. *Pseudocode* dasar AG bisa dijelasakan seperti Gambar 2.1

```
Basic Genetic Algorithm
 1: initialize population
 2: repeat
 3:     repeat
 4:         crossover
 5:         mutation
 6:         phenotype mapping
 7:         fitness computation
 8:     until population complete
 9:     selection of parental population
10: until termination condition
```

Gambar 2.1 Pseudocode Umum AG (Kramer, 2017)

Secara umum Gen dan Cheng (1997) menggambarkan struktur umum algoritma genetika seperti pada Gambar 2.2.



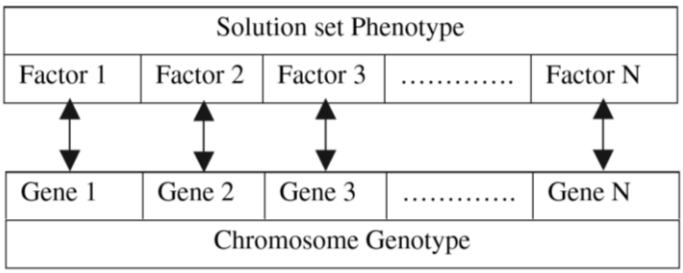
Gambar 2.2 Struktur umum AG (Gen dan Cheng, 1997)

Berdasarkan Gambar 2.2, AG dimulai dari sekumpulan solusi acak yang disebut populasi awal. Masing-masing individu yang ada dalam populasi dinyatakan sebagai kromosom yang merepresentasikan suatu solusi. Kromosom-kromosom tersebut akan mengalami iterasi yang disebut generasi. Pada setiap generasi, kromosom-kromosom tersebut dievaluasi berdasarkan nilai *fitness*-nya. Untuk menghasilkan kromosom dan generasi baru yang disebut *offspring*, dilakukan penggabungan dua kromosom dari generasi tersebut dengan menggunakan operator *crossover* dan dengan memodifikasi kromosom dengan operator mutasi. Generasi baru dibentuk melalui proses seleksi berdasarkan nilai *fitness*-nya. Populasi pada setiap generasi akan terus dijaga berada pada jumlah yang tetap.

2.3.1. Individu

Menurut Sivanandam dan Deepa (2008) setiap invidu merupakan satu calon solusi. Setiap individu berkelompok membentuk dua bentuk solusi sebagai berikut:

1. Kromosom, yang merupakan informasi dasar genetika (genotype)
2. Phenotype, yang merupakan model dari sebuah kromosom.



Gambar 2.3 Representasi Genotype dan Phenotype

Gambar 2.4 Representasi Kromosom

Sebuah kromosom terbagi menjadi gen-gen. Sebuah gen merupakan representasi dari sebuah calon solusi. Masing-masing solusi berkorespondensi dengan gen di dalam kromosom. Dalam menghitung sebuah solusi, perlu adanya representasi atau pengkodean untuk setiap individu. Pengkodean adalah proses mewakili gen individu. Proses ini dapat dilakukan dengan menggunakan bit, angka, pohon, array, daftar atau objek lainnya. Pengkodean individu tergantung pada pemecahan masalah yang akan dihadapi. Ada beberapa pengkodean yang ada dalam merepresentasikan individu, diantaranya :

1. Pengkodean Biner

   Terdiri dari angka 0 dan 1, ilustrasinya dapat dilihat pada gambar 2.5

Chromosome 1	1 1 0 1 0 0 0 1 1 0 1 0
Chromosome 2	0 1 1 1 1 1 1 1 1 1 0 0

Gambar 2.5 Pengkodean Biner

2. Pengkodean Permutasi

   Terdiri dari sebuah bilangan real/integer yang merepresentasikan kromosom secara berurutan, ilustrasi dapat dilihat pada gambar 2.6

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

Gambar 2.6 Pengkodean Permutasi

3. *Value Encoding*

   Setiap kromosom adalah serangkaian *value* (nilai) apapun yang mampu merepresaentasikan kromosomnya. Nilai dapat berupa apa saja yang terkait dengan masalah, bentuk angka, bilangan real atau karakter ke beberapa objek yang rumit. Ilustrasinya dapat dilihat pada gambar 2.7

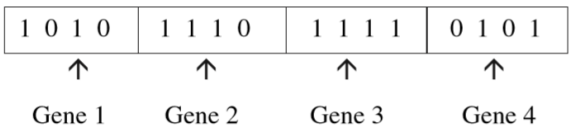
Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

Gambar 2.7 *Value Encoding*

2.3.1. Gen

Gen merupakan instruksi dasar untuk membangun sebuah algoritma genetika. Sebuah kromosom merupakan rangkaian dari gen. Gen dapat dapat mendeskripsikan solusi dari permasalahan tetapi bukanlah solusi itu sendiri. Sebuah gen terdiri dari bit string yang direpresentasikan secara biner dengan interval dari batas bawah ke batas atas atau bisa disebut dengan alel. Alel adalah nilai dari sebuah fitur dari setiap gen. Struktur masing-masing gen ditunjukkan dalam sebuah record yang memetakan genotype dengan phenotype.

Pemetaan tersebut penting untuk merubah set solusi dari model menjadi bentuk yang dapat digunakan oleh operator algoritma genetika (Sivanandam dan Deepa, 2008). Illustrasi gen dapat dilihat pada Gambar 2.8



Gambar 2.8 Representasi Gen

2.3.2. Populasi

Sebuah populasi merupakan kumpulan individu. Sebuah populasi terdiri dari sejumlah individu yang diuji dan ditentukan oleh parameter *phenotype* ditambah informasi lainnya yang ada dalam proses pencarian. Dua aspek penting populasi yang digunakan dalam algoritma genetika adalah:

1. Populasi generasi awal
2. Ukuran populasi

Ukuran populasi akan bergantung pada kerumitan permasalahan dan dalam proses inisialiasi ukuran populasi secara acak atau dipastikan sebelumnya. Secara ideal, populasi pertama harus memiliki sebuah kumpulan gen sebesar mungkin sehingga dapat mengeksplorasi keseluruhan kandidat solusi atau mengarah pada solusi. Semua kemungkinan solusi harus dimunculkan dalam populasi, yang dalam hal ini dilakukan secara acak (Sivanandam dan Deepa, 2008). Ilustrasi populasi bisa dilihat pada Gambar 2.9.

Population	Chromosome 1	1 1 1 0 0 0 1 0
	Chromosome 2	0 1 1 1 1 0 1 1
	Chromosome 3	1 0 1 0 1 0 1 0
	Chromosome 4	1 1 0 0 1 1 0 0

Gambar 2.9 Populasi

2.3.3. Fitness

Sivanandam dan Deepa (2008) menyatakan bahwa *fitness* individu dalam algoritma genetika merupakan nilai dari fungsi obyektif *phenotype*nya. Kromosom harus dikodekan dan fungsi obyektifnya harus dievaluasi terlebih dahulu sebelum dilakukan perhitungan *fitness*. *Fitness* tidak hanya mengindikasikan solusi yang terbaik, tetapi juga memperlihatkan kedekatan kromosom dengan titik optimalnya.

Suyanto (2008) mengatakan seleksi alamiah di dunia nyata, bahwa hanya individu unggul saja yang akan bertahan hidup. Sedangkan

individu berkualitas rendah akan mati atau punah. Pada AG, suatu individu dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran nilai kualitasnya. Jika tujuannya adalah memaksimalkan sebuah fungsi, maka fungsi *fitness* digunakan adalah fungsi itu sendiri, maka formula yang digunakan adalah *f* = *h* (dimana *f* adalah fungsi *fitness*). Tetapi jika tujuannya adalah meminimalkan fungsi maka fungsi *fitness* tersebut perlu dimodifikasi menjadi persamaan 2.1 sebagai:

$$f = \frac{1}{(h+\alpha)} \qquad (2.1)$$

dimana α adalah bilangan yang dianggap kecil dan disesuaikan dengan masalah yang akan diselesaikan. Sedangkan *h* adalah persamaan yang akan dibentuk sesuai dengan permasalahan yang ada. Semakin kecil nilai *h* maka semakin besar nilai *f*.

2.3.4. Seleksi

Seperti yang dikatakan oleh Sivanandam dan Deepa (2008), seleksi orang tua digunakan untuk memilih kromosom-kromosom bernilai *fitness* tinggi yang akan diberikan kesempatan reproduksi yang lebih besar. Nilai tersebut tergantung dari nilai acak yang dibangkitkan. Pemilihan kromosom-kromosom dilakukan dengan menggunakan teknik seleksi. Terdapat beberapa teknik seleksi, antara lain:

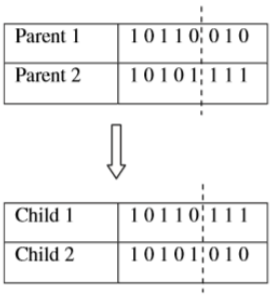
1) Roulette Wheel Selection, semakin besar nilai *fitness* dari individu semakin besar juga peluang untuk terpilihnya menjadi orang tua

2) Turnamen Selection, individu yang memiliki *fitness* terbaik akan terpilih menjadi orang tua
3) Random Selection, secara acak memilih individu untuk dijadikan orang tua

2.3.5. Rekombinasi (*Crossover*)

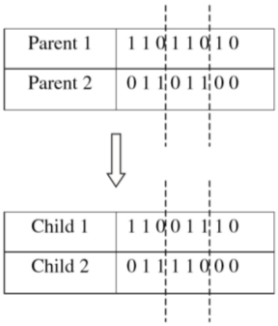
Crossover adalah proses persilangan dari dua individu (yang dianggap sebagai induk) untuk menghasilkan individu baru yang memiliki sifat dari kedua induknya (Sivanandam dan Deepa, 2008). Macam-macam *crossover* adalah sebagai berikut :

1) *Single Point*, kromosom pada dua induk ditukarkan di titik *crossover* yang diseleksi secara random, dengan sisi yang satu tetap dan yang lain ditukarkan dengan kedua kromosom pada titik *crossover* seperti pada Gambar 2.10



Gambar 2.10 *Single point crossover* (Sivanandam dan Deepa, 2008)

2) *Two Point*, dua kromosom dipotong di dua titik untuk dipertukarkan bagian tengahnya, seperti pada Gambar 2.11



Gambar 2.11 *Two Point crossover* (Sivanandam dan Deepa, 2008)

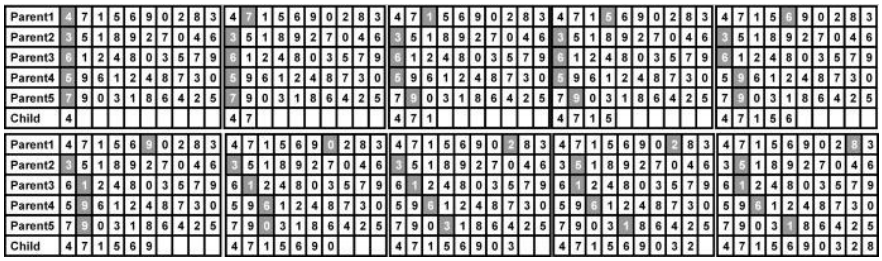
3) *Uniform*, beberapa gen pada dua kromosom induk saling dipertukarkan secara acak. Ilustrasi *uniform crossover* ditunjukkan pada Gambar 2.12

Parent 1	1 0 1 1 0 0 1 1
Parent 2	0 0 0 1 1 0 1 0
Mask	1 1 0 1 0 1 1 0
Child 1	1 0 0 1 1 0 1 0
Child 2	0 0 1 1 0 0 1 1

Gambar 2.12 *Uniform crossover* (Sivanandam dan Deepa, 2008)

4) *Multi-parent*, untuk membangun AG mungkin tidak perlu terlalu kaku dalam mengadopsi apa yang ada didunia nyata. Mungkin penggunaan lebih dari 2 kromosom orang tua dalam proses rekombinasi akan memberikan hasil yang lebih baik. Hal ini disebut sebagai rekombinasi banyak orang tua (Multi-parent) (Suyanto, 2008). Rekombinasi ini dirancang untuk menghasilkan keturunan dari banyak orang tua. Pada langkah ini, keturunan yang dihasilkan disimpan dalam arsip eksternal dan dapat digunakan pada tahap

penggantian. Dari hasil pengamatan bahwa Rekombinasi *Multi-parent* dapat memberikan hasil yang lebih baik. Kinerja algoritma tersebut tetap tergantung pada jenis masalah tertentu. Salah satu teknik rekombinasi *Multi-parent* adalah *Scanning Bassed Crossover (selanjutnya disebut SBC)*. Jika hubungan gen diisolasi dalam kromosom, dengan teknik ini mampu untuk melalui setiap urutan kromosom dari awal hingga akhir, dan mengambil nilai dari setiap orang tua sebagai kandidat (Chen dkk., 2012). Gambar 2.13 mengilustrasikan bagaimana cara kerja dari *Multi-parent Crossover*.



Gambar 2.13 *Multi-parents Crossover: SBC* (Chen dkk., 2012)

2.3.6. Mutasi

Kemampuan mewarisi sifat induknya (perubahan dalam urutan DNA) disebut mutasi. Mutasi dalam algoritma genetik, biasanya menggunakan pola *gene inversion* (pembalikan gen). Gen atau bit yang dikehendaki mutasi dibalik (dari 0 menjadi 1 dan dari 1 menjadi 0). Kondisi ini untuk menjaga agar tidak ada informasi pada kromosom yang hilang. Dengan mutasi ini setiap kromosom baru dapat diciptakan dengan melakukan modifikasi terhadap satu atau lebih gen pada kromosom yang sama. Jika tidak ada mutasi, generasi setelah *crossover* akan digunakan tanpa perubahan apa pun. Jika mutasi dilakukan, satu atau lebih bagian

dari kromosom berubah. Jika probabilitas mutasi adalah 100%, seluruh kromosom diubah, jika 0%, tidak ada yang berubah. Mutasi umumnya mencegah GA dari jatuh ke ekstrem lokal. Mutasi seharusnya tidak terlalu sering terjadi, karena dengan demikian GA sebenarnya akan berubah menjadi randomsearch (Sivanandam dan Deepa, 2008).

2.4. Optimasi

Optimasi adalah proses menyelesaikan suatu masalah tertentu supaya berada pada kondisi yang paling menguntungkan dari suatu sudut pandang. Masalah yang harus diselesaikan berkaitan erat dengan data-data yang dapat dinyatakan dalam satu atau beberapa variable. Pengertian menguntungkan, biasanya berhubungan dengan pencarian nilai minimum atau pencarian nilai maksimum, bergantung pada sudut pandang yang digunakan (Zukhri, 2014). Pada kasus optimasi dikenal dua masalah yaitu maksimasi dan minimasi. Maksimasi artinya mencari nilai maksimal dari sesuatu, sedangkan minimasi artinya mencari nilai minimal dari sesuatu (Suyanto, 2008).