BAB III
METODOLOGI PENELTIAN

3.1 Metodologi Penelitian

Dalam pembuatan suatu karaya ilmiah tentu perlu suatu metodologi penelitian agar penelitian yang dilakukan berjalan dengan baik dan terarah. Metodologi penelitian yang digunakan dalam menyusun makalah ini meliputi identifikasi masalah, analisis kebutuhan, perancangan, implementasi, dan pengujian. Hal ini dilakukan agar dalam penyelesaian tugas akhir lebih mudah dan terarah. Metodologi yang digunakan antara lain:

3.2 Identifikasi Masalah

Pada tahapan ini dilakukan identifikasi kebutuhan, masalah, cara kerja, dan ruang lingkup software/perangkat lunak yang akan dibangun menggunakan cara:

1. Membaca literatur ilmiah baik artikel/jurnal, makalah, buku, ataupun penelitian-penelitian sebelumnya yang berhubungan penelitian ini.
2. Melakukan analisis terhadap permasalahan SAT *Problem* dan cara-cara penyelesaiannya.
3. Melakukan analisis terhadap Algoritma Genetika dan implementasinya untuk menyelesaikan permasalahan SAT *Problem*.
4. Melakukan analisis tentang solusi pendekatan dan implementasinya pada Algoritma Genetika agar bisa menyelesaikan permasalahan SAT *Problem*.
5. Menganalisis penggunaan bahasa pemrograman JAVA untuk mengimplementasikan pendekatan solusi SAT *Problem* menggunakan Algoritma Genetika.

3.3 Analisis Kebutuhan

Analisis kebutuhan adalah tahapan untuk melakukan proses pengumpulan data dan informasi yang akan digunakan untuk menunjang pembuatan *software*/aplikasi. Aplikasi tersebut dibuat untuk mendapatkan jawaban dari rumusan masalah yang telah dibentuk sebelumnya. Metode pengumpulan data yang dilakukan adalah dengan menggunakan metode studi pustaka. Studi pustaka adalah sebuah metode dalam pengumpulan data dengan melakukan pencarian informasi melalui media seperti artikel, buku, jurnal atau internet. Studi

pustaka yang dilakukan adalah dengan membaca dan mencari literatur dari internet mengenai sistem yang dalam penelitian ini membahas tentang SAT *Problem* dan Algoritma Genetika, serta mencari informasi mengenai perangkat keras maupun perangkat lunak yang sesuai agar sistem dapat berjalan dan berfungsi sesuai dengan yang diharapkan.

Pada tahapan ini juga akan dilakukan sebuah analisis untuk mengetahui kebutuhan apa saja yang diperlukan untuk mengembangkan sebuah software/perangkat lunak (SAT Solver) berbasis bahasa pemrograman JAVA yang dapat menyelesaikan permasalahan SAT *Problem* menggunakan Algoritma Genetika dan konsep pendekatan solusi.

3.3.1 Analisis kebutuhan fungsi

Analisis kebutuhan fungsi adalah tahapan dimana dilakukan penetapan fungsi yang dapat dilakukan oleh sistem, sehingga dapat menjawab rumusan masalah yang ada. Sistem ini nantinya akan memiliki fungsi sebagai berikut:

a. Membaca formula proposisi dalam bentuk CNF (*Conjunctive Normal Form*).
b. Menyelesaikan permasalahan SAT *Problem*.
c. Memberikan solusi pendekatan jika solusi belum ditemukan sampai batas iterasi.

3.3.2 Analisis kebutuhan Masukan

Analisis kebutuhan masukan adalah tahapan untuk menentukan masukan apa saja yang dibutuhkan. Masukan tersebut harus membuat sistem dapat menjalankan fungsi untuk menyelesaikan SAT *Problem* ini. Kebutuhan masukan dalam penelitian ini adalah sebagai berikut:

a. Masukan yang diberikan adalah sebuah *file* berisikan formula proposisi yang sesuai dengan standar CNF dan disimpan dalam format .txt.
b. Sebuah kriteria berhenti untuk membatasi seberapa banyak generasi yang dibentuk.
c. Masukan untuk menentukan probabilitas terjadinya mutasi.
d. Masukan untuk menentukan probabilitas terjadinya *crossover*.
e. Masukan untuk menentukan jumlah individu untuk sebuah populasi.
f. Masukan untuk menentukan panjangnya gen sebuah individu.

3.3.3 Analisis kebutuhan Keluaran

Pada tahapan ini akan dibuat sebuah sistem. Sistem tersebut dapat menyelesaikan masukan yang diberikan dan memberikan keluaran. Keluaran tersebut harus bisa membuktikan sistem telah berhasil menyelesaikan masukan sesuai dengan fungsi yang telah ditentukan. Kebutuhan keluaran dalam penelitian ini adalah sebagai berikut:

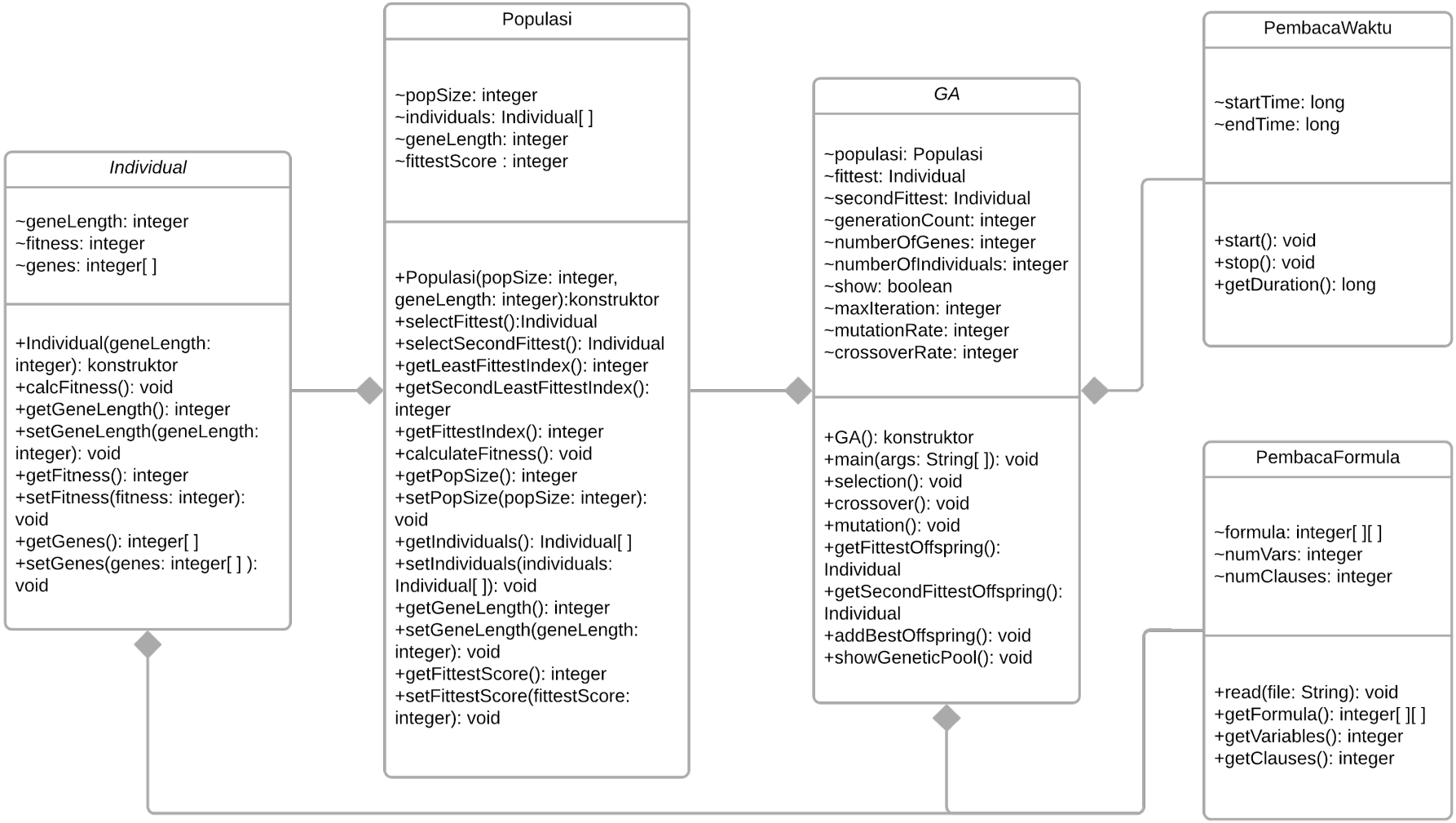
a) Sebuah keluaran string berupa "*satisfiable*" atau "Pendekatan".
b) Pemberian nilai-nilai kebenaran yang harus diberikan kepada setiap variabel proposisi.
c) Persentase jumlah klausa yang benar.

3.4 Metode Perancangan

Metode perancangan merupakan suatu cara untuk menjelaskan perancangan dari sebuah penelitian. Tahap ini merupakan tahapan penting sebelum lanjut ke tahap implementasi/pengerjaan dalam sebuah penelitian. Pada tahap ini akan didapatkan sebuah rancangan yang akan digunakan dalam tahap implementasi, sehingga bisa dengan mudah mengidentifikasi, mengatasi, dan mengevaluasi kendala-kendala dalam pengembangan aplikasi. Dalam penelitian ini rancangan dibuat adalah rancangan SAT *Solver* dengan Algortima Genetika dan konsep pendekatan solusi agar dapat menyelesaikan SAT *Problem*.

3.4.1 Diagram Kelas

Diagram kelas adalah diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam suatu sistem atau perangkat lunak yang sedang dikembangkan. Diagram kelas memberikan sebuah gambaran statis tentang sistem/perangkat lunak beserta relasi-relasi yang terdapat di dalam suatu sistem/perangkat lunak tersebut. Di dalam diagram kelas juga terdapat deskripsi dari masing-masing objek berupa properti, metode, dan relasi. Diagram kelas dalam aplikasi ini ditunjukkan pada Gambar 3.1.



Gambar 3.1 Diagram Kelas

Pada Gambar 3.1 terdapat lima kelas, yaitu GA, Populasi, Individual, PembacaFormula, dan PembacaWaktu. Adapun fungsi dari kelas-kelas tersebut akan dijelaskan sebagai berikut:

1. Kelas GA

   Kelas GA memiliki atribut dan *Method* yang berfungsi untuk mengimplementasikan Algoritma Genetika. Algoritma Genetika dalam kelas GA ini berada dalam *Method* main. Adapun atribut dan *Method* dalam Kelas GA ditunjukkan pada Tabel 3.1.

Tabel 3.1 Kelas GA

Atribut / *Method*	**Tipe Data**	**Parameter Formal**	**Keterangan**
populasi	Populasi	-	Menyimpan nilai populasi.
fittest	Individual	-	Menyimpan individu dengan nilai *fitness* terbaik.
secondFittest	Individual	-	Menyimpan individu dengan nilai *fitness* terbaik kedua.
generationCount	integer	-	Menyimpan nilai generasi dari populasi.
numberOfGenes	Integer	-	Menyimpan nilai panjangnya gen dari suatu individu.
numberOfIndividuals	Integer	-	Menyimpan nilai banyaknya individu dalam populasi.
show	Boolean	-	Menyimpan nilai proses komputasi.
maxIteration	Integer	-	Menyimpan nilai jumlah maksimum iterasi yang diinginkan.
mutationRate	Integer	-	Menyimpan nilai kemungkinan terjadinya mutasi.
crossoverRate	Integer	-	Menyimpan nilai kemungkinan terjadinya crossover.
GA	konstruktor	-	Memberi nilai pada variabel populasi dan generationCount.

main	Void	args: String[ ]	Prosedur yang berfungsi untuk menjalankan Algoritma Genetika. *Method* ini berfungsi untuk mengimplementasikan Algoritma Genetika. Keluaran dari *Method* ini akan ditampilkan di konsol.
selection	Void	-	Memilih individu dengan nilai *fitness* terbaik dan memberikan nilai pada variabel fittest dan secondFittest.
crossover	Void	-	Memberikan nilai baru pada variabel fittest dan secondFittest dengan cara crossover.
mutation	Void	-	Memberikan nilai baru pada variabel fittest dan secondFittest dengan cara mutasi.
getFittestOffspring	Individual	-	Memperbarui dan memberikan nilai pada variabel fittest.
getSecondFittestOffspring	Individual	-	Memperbarui dan memberikan nilai pada variabel secondFittest.
addBestOffspring	Void	-	Menghapus 2 individu dengan nilai *fitness* terburuk dan menggantinya dengan 2 individu terbaik dari hasil persilangan.
showGeneticPool	Void	-	*Method* untuk menunjukan proses komputasi

2. Kelas Populasi

Kelas Populasi berfungsi untuk merepresentasikan suatu populasi (kumpulan individu/kromosom). Adapun atribut dan *Method* dalam Kelas Populasi ditunjukkan oleh Tabel 3.2.

Tabel 3.2 Kelas Populasi

Atribut / *Method*	**Tipe Data**	**Parameter Formal**	**Keterangan**
popSize	Integer	-	Menyimpan nilai besarnya populasi.
individuals	Individual[ ]	-	Menyimpan list array individu
geneLength	Integer	-	Menyimpan nilai panjangnya gen.
fittestScore	integer	-	Menyimpan nilai *fitness* terbaik.
Populasi	konstruktor	popSize: integer, geneLength: integer	Membuat populasi (kumpulan individu) sebanyak popSize dan dengan panjang gen sama dengan geneLength. Kemudian mengisi gen individu tersebut secara acak.
selectFittest	Individual	-	Memberikan nilai return index individu dengan *fitness* terbaik dan perbarui urutan.
selectSecondFittest	Individual	-	Memberikan nilai return index individu dengan *fitness* terbaik kedua.
getLeastFittestIndex	Integer	-	Memberikan nilai return index individu dengan *fitness* terburuk.
getSecondFittestIndex	Integer	-	Memberikan nilai return index individu dengan *fitness* kedua terburuk.

getFittestIndex	Integer	-	Memberikan nilai return index individu dengan *fitness* terbaik.
calculateFitness	void	-	Menghitung nilai *fitness* individu.
getPopSize	Integer	-	Memberi nilai return PopSize.
setPopSize	Void	popSize: integer	Memberi nilai pada variabel PopSize.
getIndividuals	Individual[ ]	-	Memberi nilai return Individuals.
setIndividuals	Void	individuals: Individual[ ]	Memberi nilai pada variabel Individuals.
getGeneLength	Integer	-	Memberi nilai return getGeneLength.
setGeneLength	Void	-	Memberi nilai pada variabel GeneLength.
getFittestScore	Integer	-	Memberi nilai return getFittestScore.
setFittestScore	Void	fittestScore: integer	Memberi nilai pada variabel FittestScore.

3. Kelas Individual

Kelas Individual berfungsi untuk merepresentasikan suatu individu/kromosom. Adapun atribut dan *Method* dalam Kelas Individual ditunjukkan oleh Tabel 3.3.

Tabel 3.3 Kelas Individual

Atribut / *Method*	**Tipe Data**	**Parameter Formal**	**Keterangan**
geneLength	Integer	-	Menyimpan nilai panjangnya gen.
fitness	Integer	-	Menyimpan nilai *fitness*.
genes	Integer[ ]	-	Menyimpan nilai gen.
Individual	konstruktor	geneLength: integer	Membuat individu baru.

calcFitness	void	-	Menghitung nilai *fitness* suatu individu.
getGeneLength	Integer	-	Memberi nilai return geneLength.
setGeneLength	Void	geneLength: integer	Memberi nilai pada variabel geneLength.
getFitness	Integer	-	Memberi nilai return fitness.
setFitness	Void	fitness: integer	Memberi nilai pada variabel fitness.
getGenes	Integer[ ]	-	Memberi nilai return genes.
setGenes	Void	genes: integer[ ]	Memberi nilai pada variabel genes.

4. Kelas PembacaFormula

   Kelas PembacaFormula berfungsi untuk membaca masukan formula logika proposisi dalam bentuk CNF dan menyimpan nilai-nilai penting yang terdapat dari masukan tersebut. Adapun atribut dan *Method* dalam Kelas Individual ditunjukkan oleh Tabel 3.4.

Tabel 3.4 Kelas PembacaFormula

Atribut / *Method*	**Tipe Data**	**Parameter Formal**	**Keterangan**
formula	Integer[ ][ ]	-	Menyimpan variabel preposisi formula masukan dalam bentuk matriks 2D.
numVars	Integer	-	Menyimpan jumlah variabel dalam formula masukan.
numClauses	Integer	-	Menyimpan jumlah klausa dalam formula masukan.
read	void	file: String	Memberikan nilai pada variabel formula, numVars, dan numClauses dari masuka formula.

getFormula	Integer[ ][ ]	-	Memberikan nilai return formula.
getVariables	Integer	-	Memberikan nilai return variabel.
getClauses	Integer	-	Memberikan nilai return klausa.

5. Kelas PembacaWaktu

Kelas PembacaWaktu berfungsi untuk menghitung proses komputasi mulai dari awal komputasi sampai dengan akhir komputasi. Lama proses komputasi tersebut akan disimpan dalam variabel dengan tipe data long. Adapun atribut dan *Method* dalam Kelas PembacaWaktu ditunjukkan oleh Tabel 3.5.

Tabel 3.5 Kelas PembacaWaktu

Atribut / *Method*	**Tipe Data**	**Parameter Formal**	**Keterangan**
startTime	long	-	Menyimpan nilai waktu saat proses komputasi dimulai.
endTime	long	-	Menyimpan nilai waktu saat proses komputasi berakhir.
start	void	-	Memberikan nilai pada variabel startTime.
stop	void	-	Memberikan nilai pada variabel endTime.
getDuration	long	-	Memberikan nilai return selisih waktu endTime dan startTime.

3.4.2 Perancangan Masukan

Dalam tahapan perancangan masukan ini dibentuk enam buah masukan sesuai dengan analisis masukan enam masukan yaitu, formula, maxIteration, mutationRate, crossoverRate, numberOfIndividuals dan numberOfGenes. Adapun masukan dari *software*/perangkat lunak tersebut akan dijelaskan sebagai berikut:

1. formula

   Masukan ini memuat formula logika proposisi dengan standar CNF dan format .txt. Adapun standar CNF yang dimaksud terdiri dari komentar, parameter, dan variabel yang diberikan dalam bentuk integer. Baris awal sebuah masukan yang diawali oleh sebuah karakter ‘c’ berarti sebuah komentar yang tidak perlu dibaca oleh program. Komentar tersebut bisa berisi identitas dari pembuat berkas masukan tersebut maupun keterangan tambahan yang diperlukan. Baris selanjutnya adalah baris yang diawali oleh sebuah karakter ‘p’ yang berarti parameter mengenai keterangan banyaknya variabel dan klausa yang terdapat pada *file* masukan tersebut. Variabel yang diberikan pada masukan berformat CNF direpresentasikan dalam bentuk integer. *Conjunction* (∧) antar klausa direpresentasikan dengan angka 0, sedangkan *disjunction* (∨) antar literal direpresentasikan dengan spasi. Untuk variabel yang memiliki nilai negasi ditandai dengan penambahan simbol negatif (-). Contoh CNF file dapat dilihat pada Lampiran 1 dan Contoh konversi masukan logika proposisi menjadi format CNF bisa dilihat pada Tabel 3.6 berikut ini.

Tabel 3.6 konversi logika proposisi ke CNF

Keterangan	**Konversi CNF**
Logika proposisi	(a ∨ ~b) ∧ (~d ∨ a) ∧ (b ∨ c)
Setiap Variabel diubah menjadi Integer	(1 ∨ 2) ∧ (~4 ∨ 1) ∧ (2 ∨ 3)
Simbol negasi diubah menjadi negatif	(1 ∨ 2) ∧ (-4 ∨ 1) ∧ (2 ∨ 3)
Simbol disjungsi diubah menjadi spasi	(1 2) ∧ (-4 1) ∧ (2 3)
Simbol konjungsi diubah menjadi angka 0	(1 2) 0 (-4 1) 0 (2 3)
Tanda kurung dihilangkan	1 2 0 -4 1 0 2 3
Setiap klausa dipisahkan dengan baris baru	1 2 0 -4 1 0 2 3

2. maxIteration

   Masukan ini menentukan batas atas/maksimum banyaknya generasi yang akan dibentuk.

3. mutationRate

   Masukan ini menentukan kemungkinan terjadinya mutasi gen-gen penyusun individu secara acak pada tiap generasi.

4. crossoverRate

   Masukan ini menentukan kemungkinan terjadinya *crossover* terhadap individu-individu dengan nilai *fitness* terbaik tiap generasi.

5. numberOfIndividuals

   Masukan ini menentukan jumlah individu pada tiap populasi.

6. numberOfGenes

   Masukan ini menentukan panjang gen pada tiap individu sesuai dengan jumlah variabel dari formula logika proposisi tersebut. Dalam penelitian ini panjang gen tiap individu sama dengan jumlah variabel formula logika proposisi yang inigin diselesaikan.

Adapun contoh rancangan masukan ditunjukkan pada Gambar 3.2.

```
Input:
maxIteration        = 10
mutationRate        = 7
crossoverRate       = 25
numberOfIndividuals = 10
numberOfGenes       = 4

formula
 1 2 0
-4 1 0
 2 3
```

Gambar 3.2 Perancangan masukan

3.4.3 Perancangan Keluaran

Pada *software*/perangkat lunak ini keluaran yang dihasilkan terbagi menjadi dua jenis. Keluaran yang pertama adalah untuk SAT *Problem* yang solusinya sudah ditemukan sebelum batas iterasi dicapai, maka keluaran akan dalam bentuk eksak. Keluaran yang kedua adalah untuk SAT *Problem* yang solusinya belum ditemukan sampai batas iterasi dicapai, maka keluaran akan dalam bentuk solusi pendekatan.

a. SAT *Problem* yang solusinya sudah ditemukan sebelum batas iterasi dicapai

Apabila masukan yang diberikan bersifat *satisfiable* dan solusi telah ditemukan sebelum batas iterasi dicapai, maka aplikasi akan mengeluarkan teks "*Satisfiable*" beserta akurasinya diikuti dengan komposisi kromosom penyusun individu. Komposisi tersebut merepresentasikan nilai-nilai yang harus diberikan pada variabel agar logika proposisi tersebut bernilai *satisfiable*. Adapun format mengenai rancangan tersebut ditunjukkan pada Gambar 3.3.

```
Input:
maxIteration        = 10
mutationRate        = 7
crossoverRate       = 25
numberOfIndividuals = 10
numberOfGenes       = 4

formula
 1 2 0
-4 1 0
 2 3

Output:
Satisfiable

 1
 2
-3
-4

Akurasi = 100%
```

Gambar 3.3 Keluaran jika *satisfiable*

Seperti yang terlihat pada Gambar 3.3, apabila keluaran dinyatakan "*Satisfiable*", maka keluaran tersebut akan mengeluarkan akurasi sebesar 100% artinya dari tiga klausa yang ada keseluruhannya bernilai benar. Sedangkan gen penyusun individu untuk mendapatkan hasil tersebut adalah 1, 2, -3, dan -4. Hasil tersebut bermakna variabel 1 & 2 bernilai true, sedangkan variabel 3 & 4 bernilai false.

b. SAT *Problem* yang solusinya belum ditemukan sampai batas iterasi dicapai

Apabila masukan yang diberikan bersifat *satisfiable* dan solusinya belum ditemukan sampai batas iterasi dicapai atau masukan tersebut bersifat *unsatisfiable*, maka aplikasi akan mengeluarkan teks "Pendekatan" beserta akurasinya diikuti dengan komposisi kromosom penyusun individu. Komposisi tersebut merepresentasikan nilai-nilai yang harus diberikan

pada variabel agar logika proposisi tersebut bernilai pendekatan. Adapun format mengenai rancangan tersebut ditunjukkan pada Gambar 3.4.

```
Input:
maxIteration        = 10
mutationRate        = 7
crossoverRate       = 25
numberOfIndividuals = 10
numberOfGenes       = 4

formula
 1 2 0
-4 1 0
 2 3

Output:
Pendekatan

-1
 2
-3
 4

Akurasi = 66.67%
```

Gambar 3.4 Keluaran jika pendekatan

Seperti yang terlihat pada Gambar 3.4, apabila keluaran dinyatakan "Pendekatan", maka keluaran tersebut akan mengeluarkan akurasi kurang dari 100% dalam contoh ini adalah sebesar 66.67% artinya dari tiga klausa yang ada hanya dua yang bernilai benar. Sedangkan untuk gen penyusun individu untuk mendapatkan hasil tersebut adalah -1, 2, -3, dan 4. Hasil tersebut bermakna variabel 1 & 3 bernilai false, sedangkan variabel 2 & 4 bernilai true.

3.4.4 Perancangan Algoritma Genetika untuk penyelesaian SAT *Problem*

Pada tahapan ini akan dilakukan perancangan Algoritma Genetika untuk menyelesaikan SAT *Problem*. Secara singkat, istilah-istilah dan bagian Algoritma Genetika dalam penerapannya sebagai *software*/perangkat lunak SAT *Solver* dalam penelitian ini dapat dilihat pada Tabel 3.7.

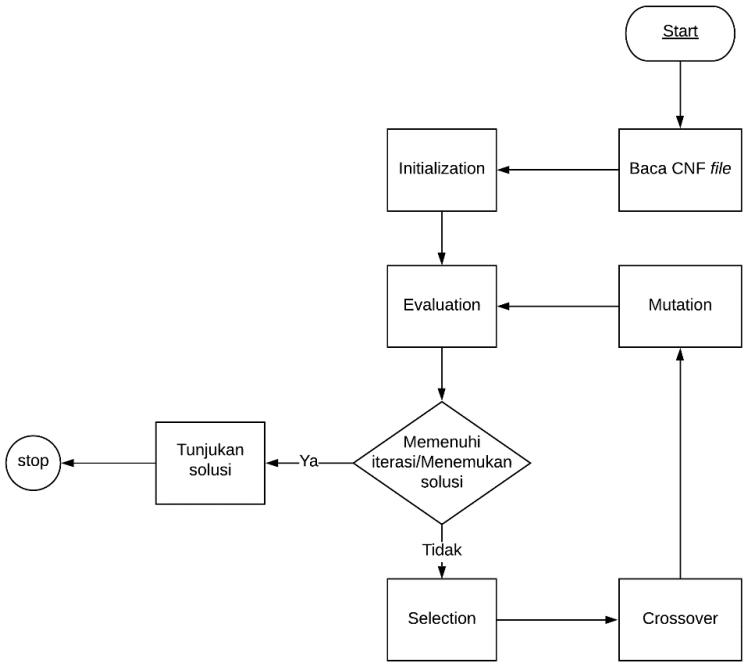
Tabel 3.7 Istilah dalam Algoritma Genetika

Nama Istilah	Penjelasan
gen	Dalam perancangan ini nilai sebuah gen merepresentasikan nilai kebenaran sebuah variabel proposisi. Nilai kebenaran adalah true yang direpresentasikan dalam bilangan integer 1 dan false yang direpresentasikan dalam bilangan integer 0.

Individu/kromosom	Dalam perancangan ini individu/kromosom adalah kumpulan gen. Individu/kromosom merepresentasikan pemberian/assignment nilai sebuah formula logika proposisi atau disebut dengan solusi. Panjangnya gen penyusun individu/kromosom sama dengan jumlah variabel proposisi.
Populasi	Populasi adalah jumlah individu/kromosom acak yang dibangkitkan dengan jumlah tertentu. Misal, kromosom 1 dan kromosom 2 dikatakan sebagai satu kesatuan populasi.
Fitness	Dalam perancangan ini nilai *fitness* sebuah individu dihitung dengan menjumlahkan klausa yang bernilai benar akibat dari pemberian/ assignment nilai dari individu tersebut.
Iterasi	Iterasi adalah bilangan yang menunjukkan tingkat generasi atau perulangan.

1. Gambaran Umum Metode

   Metode pada Algoritma Genetika memiliki beberapa tahapan dan perulangan. Dalam penelitian ini, metode pada penyelesaian SAT *Problem* memiliki beberapa tahap penyelesaian seperti yang ada pada Gambar 3.5.



Gambar 3.5 Flowchart algoritma genetika dalam penelitian ini

2. Variabel yang digunakan dalam proses Algoritma Genetika penelitian ini

a. Jumlah Generasi

Jumlah generasi adalah jumlah maksimal iterasi / paket perulangan yang diperbolehkan. Variabel ini menentukan sampai berapa kali populasi awal akan berubah, jadi juga memiliki peran yang tak kalah penting dalam menampilkan jumlah variasi individu, yang akan berpengaruh terhadap hasil Algoritma Genetika. Dalam penelitian ini dibatasi maksimal 1000 generasi untuk menyesuaikan dengan memori yang terdapat dalam komputer yang digunakan dalam pengujian penelitian ini (Ho & Ji, 2005).

b. Tingkat Persilangan (*Crossover Rate*)

Tingkat persilangan adalah peluang untuk terjadi persilangan antara sepasang individu. Dalam kenyataannya persilangan akan selalu terjadi, hanya saja jumlah gen dan gen-gen yang disilangkan akan berbeda-beda. Oleh karena itu, dalam penelitian ini Tingkat persilangan diisi dengan nilai 100, yang berarti akan selalu terjadi persilangan. Namun karena menggunakan prinsip *tournament strategy*, individidu baru/*offspring* yang terbentuk adalah dua karena hanya dua individu dengan nilai terbaik yang akan disilangkan (Hassanat, Almohammadi, Alkafaween, & Abunawas, 2019).

c. Tingkat Mutasi (*Mutation Rate*)

Variabel yang berupa angka persentase ini akan mempengaruhi seberapa banyak terjadinya mutasi dalam suatu populasi. Variabel tingkat mutasi merupakan salah satu variabel yang berbentuk peluang, artinya kemungkinan terjadinya mutasi dilihat tiap individunya. Misalkan jumlah individu dalam populasi adalah 10, tingkat mutasi diisi dengan 5 artinya peluang terjadinya mutasi untuk masing-masing individu adalah 5%. Jadi dalam contoh di atas bisa saja terjadi 0 mutasi sampai 10 mutasi. Dalam penelitian ini tingkat mutasi sebesar 25 % agar individu menjadi bervariasi (Hassanat, Almohammadi, Alkafaween, & Abunawas, 2019)dan hanya individu dengan nilai *fitness* terbaik yang mempunyai peluang untuk bermutasi, hal ini sesuai dengan prinsip *tournament strategy*.

d. Jumlah Individu per Populasi

Jumlah individu per populasi ditentukan sendiri pada awal pemrograman. Semakin banyak jumlah individu per populasi, semakin besar dan rumit komputasinya. Dalam penelitian ini ditetapkan 10 individu untuk sebuah populasi karena untuk memaksimalkan efek terjadinya *crossover* dan mutasi (Jiang, Zhang, Zhang, Ran, & Tang, 2018).

e. Jumlah Gen per Individu

Jumlah gen per individu menentukan panjangnya gen pembentuk sebuah individu. Dalam penelitian ini jumlah gen per individu sama dengan jumlah variabel dari formula logika proposisi yang ingin diselesaikan.

f. Fungsi Objektif dan Nilai *Fitness*

Fungsi objektif adalah formula yang dibuat untuk mengukur derajat kualitas individu. Formula ini yang akan dibuat untuk menghitung nilai *fitness*. Nilai *fitness* adalah bilangan yang menunjukkan kualitas individu. Semakin tinggi nilai *fitness*, semakin tinggi kualitas dan semakin tinggi pula tingkat probabilitas seleksi. Seperti yang sudah dijelaskan di atas fungsi objektif dalam penelitian ini adalah dengan menjumlahkan seluruh klausa yang bernilai benar dalam formula logika proposisi dan nilai fitness adalah jumlah seluruh klausa yang bernilai benar dalam formula logika proposisi.

3. Initialization

Populasi merupakan kumpulan beberapa individu. Semua populasi dalam Algoritma Genetika ini berasal dari satu populasi yaitu populasi awal. Solusi atau kromosom terbaik dari populasi awal ini akan dipertahankan, dan akan mengalami mengalami proses evolusi untuk mendapatkan kemungkinan solusi yang lebih baik.

Pembuatan populasi awal ini dilakukan melalui proses pemilihan secara acak dari seluruh solusi yang ada. Pemilihan acak ini menyebabkan populasi awal dari Algoritma Genetika tidak akan sama dalam setiap kali percobaan, meskipun semua nilai variabel yang digunakan sama.

Selain itu pada tahap ini akan dilakukan pengisian nilai gen sebuah individu secara acak dengan rentang bilangan integer 0 sampai 1.

4. *Evaluation*

   Untuk mengetahui baik tidaknya solusi yang ada pada suatu individu, setiap individu pada populasi harus memiliki nilai pembandingnya (*fitness cost*). Melalui nilai pembanding inilah akan didapatkan solusi terbaik dengan cara pengurutan nilai pembanding dari individu-individu dalam populasi. Solusi terbaik ini akan dipilih dan dipertahankan untuk tahap selanjutnya, sementara solusi lain diubah-ubah untuk mendapatkan solusi yang lain lagi, melalui tahap *crossover* dan mutasi (*mutation*).

5. *Selection*

   Pada tahapan ini akan dipilih dua individu dengan nilai *fitness* terbaik dan dua individu dengan nilai *fitness* terburuk. Kedua individu dengan nilai fitness terburuk akan digantikan oleh dua individu dengan nilai fitness terbaik.

6. *Crossover*

   Setelah menjalani proses seleksi, maka individu yang terpilih akan dilakukan Crossover. Tahapan ini akan menyilangkan dua individu terbaik yang ada dalam suatu populasi, untuk mendapatkan dua individu baru. Tipe crossover yang dipakai adalah *single-point crossover* supaya tidak menimbulkan perbedaan yang jauh antara offspring (anakan) dengan induk.

7. *Mutation*

   Cara lain untuk mendapatkan individu yang baru yaitu dengan mutasi. Probabilitas terjadinya mutasi gen pada suatu kromosom sangatlah kecil, karena itu dalam penerapannya pada Algoritma Genetika, probabilitasnya seringkali dibuat kecil, lebih kecil dari 1% (mutation rate). Namun dalam penelitian ini mutation rate justru diperbesar menjadi 20% hal ini dilakukan untuk menambah variasi individu. Berbeda dengan tahap C*rossover*, dimana satu individu perlu individu yang lain, pada tahap ini tidak membutuhkan individu yang lain untuk bermutasi. Dalam penelitian mutasi yang terjadi adalah pergantian nilai gen misal dari semula 1 menjadi 0 dan sebaliknya.

Setelah itu proses akan terus berulang sampai batas iterasi maksimum yang telah ditentukan atau solusi telah ditemukan. Nantinya jika solusi belum ditemukan, maka solusi yang ditawarkan adalah pendekatan (individu dengan nilai *fitness* terbaik sampai akhir iterasi).