

# Penerapan Algoritma Genetika Dalam Penyelesaian *Boolean Satisfiability Problem*

Alfeno Muhaydein Nur  
Program Studi Sarjana Informatika  
Universitas Islam Indonesia  
Yogyakarta  
[17523079@students.uii.ac.id](mailto:17523079@students.uii.ac.id)

Taufiq Hidayat  
Program Studi Sarjana Informatika  
Universitas Islam Indonesia  
Yogyakarta  
[taufiq.hidayat@uii.ac.id](mailto:taufiq.hidayat@uii.ac.id)

**Abstract**—*Boolean Satisfiability Problem* (SAT Problem) merupakan salah satu konsep logika matematika untuk menentukan apakah suatu formula bisa disebut formula yang *satisfiable* atau *unsatisfiable* dengan memberi nilai dari setiap simbol bilangan proposisi. Sebelumnya terdapat banyak aplikasi dengan menerapkan algoritma genetika dalam penyelesaian SAT Problem. Namun belum ditemukan penerapan metode *Two Point Crossover* dan *Swap Mutation* Algoritma Genetika untuk menyelesaikan SAT Problem. Dalam penelitian ini dilakukan implementasi Algoritma Genetika dengan memodifikasi metode *crossover* menggunakan *Two Point Crossover* dan *mutasi* menggunakan *Swap Mutation* untuk menyelesaikan SAT Problem dengan aplikasi yang dibangun menggunakan bahasa pemrograman Java. Setelah dilakukan pengujian, hasil yang didapatkan adalah sistem dengan penerapan Algoritma Genetika ini mampu menyelesaikan SAT Problem dalam bentuk CNF file 50 variabel dan 80 klausa dengan waktu yang singkat dalam hitungan detik.

**Keywords**—*Algoritma Genetika, SAT Problem, CNF file.*

## I. PENDAHULUAN

Setiap masalah yang ditemukan dalam kehidupan sehari-hari memerlukan langkah-langkah atau cara dalam penyelesaiannya. Contoh seperti saat pemuda mengerjakan ujian, pemuda tersebut akan memilih mengerjakan soal mulai dari tingkat kesulitan paling ringan dahulu agar mendapatkan hasil maksimum serta dapat menyelesaikan soal dalam waktu pengerjaan yang sudah ditentukan.

Dalam teori matematika dan ilmu komputer, untuk menyelesaikan sebuah permasalahan memerlukan sebuah algoritma atau cara penyelesaian tertentu. Contoh pada dunia komputer, agar dapat menyelesaikan masalah menggunakan komputer, diperlukan sebuah Langkah-langkah perumusan penyelesaian (algoritma). Kemudian mengukur seberapa efisien suatu algoritma tersebut, dengan cara membandingkan beberapa metode algoritma yang ditemukan atau dibuat menggunakan *time complexity* dan *space complexity*. *Time complexity* yaitu mengukur seberapa lama waktu yang dibutuhkan suatu algoritma dalam menyelesaikan masalah, sedangkan *space complexity* yaitu seberapa besar memori yang digunakan dalam menyelesaikan suatu masalah[15].

*Boolean Satisfiability Problem* (SAT Problem) adalah konsep logika matematika untuk menentukan apakah suatu bilangan proposisi bersifat *satisfiable* atau *unsatisfiable* dengan memberikan nilai di setiap simbol proposisi pada logika Boolean[1]. Dengan kata lain SAT problem juga bisa diartikan sebagai salah satu logika matematika yang menentukan sebuah formula pada logika Boolean merupakan komposisi yang *satisfiable* atau *unsatisfiable*[2]. Formula yang digunakan untuk menyelesaikan SAT Problem dengan SAT Solver adalah formula proposisi dalam bentuk *Conjunctive Normal Form*.

*Conjunctive Normal Form* (CNF) adalah salah satu formula proposisi dalam bentuk normal form yang memiliki unsur berupa variabel negasi, operator or( $\vee$ ) dan operator and( $\wedge$ ). Tingkat kompleksitas untuk menyelesaikan SAT Problem adalah  $2^n$ [3]. 'n' pada logika proposisi ini memiliki artian banyaknya variabel. Dengan contoh untuk menentukan apakah logika proposisi  $A \vee B$  adalah *satisfiable*, maka dibutuhkan kompleksitas  $4^n = 4^4 = 16$ . Proposisi tersebut memiliki variabel sebanyak 4. Jika proposisi tersebut memiliki 10 variabel artinya kompleksitas proposisi tersebut adalah  $4^n = 4^{10} = 1048576$ .

SAT Problem sendiri termasuk dalam permasalahan yang tergolong *Non Polynomial Complete*[4]. Karena SAT Problem tergolong dalam permasalahan *Non Polynomial Complete*, artinya SAT Problem memiliki kompleksitas yang sangat besar. Sehingga akan sulit jika diselesaikan dengan cara perhitungan manual, dan disisi lain juga akan membutuhkan waktu yang cukup lama.

Ada dua pendekatan yang secara umum digunakan untuk memecahkan permasalahan SAT Problem, yaitu dengan menggunakan teknik optimasi dan kerangka teori genetika atau algoritma evolusioner, hal tersebut menjadi dasar dalam teori algoritma genetika[5]. Di sini metode algoritma genetika terbukti lebih cepat untuk menyelesaikan permasalahan SAT Problem dibandingkan dengan metode lainnya.

Dalam teori algoritma genetika terdapat fungsi fitness yang merupakan bagian dari proses encoding yang berfungsi untuk menentukan nilai fit dari setiap kromosom. Pada proses tersebut terdapat beberapa faktor mulai dari inisiasi, regenerasi yang di dalamnya meliputi (seleksi, crossover, mutasi), replacement dan terminasi atau perulangan[6].

Pada berbagai penelitian yang dilakukan sebelumnya sudah terdapat aplikasi SAT Solver dengan menggunakan berbagai macam algoritma antara lain: Satz[7], GRASP[8], WalkSat[9], zChaff dan mChaff[10]. Namun pada aplikasi SAT Solver tersebut memerlukan resource dan spesifikasi komputer yang besar untuk menjalankannya sekaligus dalam prosesnya membutuhkan waktu yang lama karena output solusi yang dihasilkan berupa eskak. Eskak adalah proses penyelesaian SAT Problem dengan masukan file berbentuk CNF yang saat diproses dengan menempatkan setiap nilai variabel proposisi dan harus menampilkan klausa dengan nilai akhir benar atau *satisfiable*.

Berdasarkan keterangan diatas maka makalah penelitian ini bertujuan untuk mengimplementasikan konsep Algoritma Genetika dan memodifikasi metode *crossover* dan *mutasi* dalam menyelesaikan SAT Problem dengan aplikasi yang dibangun menggunakan bahasa Java. Pada sebelumnya sudah pernah dilakukan penelitian dengan kasus SAT Problem menggunakan konsep pendekatan menggunakan Algoritma

Genetika dengan menerapkan *Single Point Crossover* dan *Bit Flip Mutation*. Sehingga pada penelitian ini diharapkan penggunaan konsep pendekatan menggunakan algoritma Genetika dengan menerapkan *Two Point Crossover* dan *Swap Mutation* mampu menyelesaikan kasus SAT Problem dengan waktu yang lebih baik dan hasil ketepatan solusi yang optimal.

## II. LANDASAN TEORI

### A. Tinjauan Pustaka

Berbagai penelitian telah dilakukan sebelumnya tentang aplikasi SAT Solver dengan menggunakan berbagai macam algoritma antara lain: Satz[7], GRASP[8], WalkSat[9], zChaff dan mChaff[10]. Terdapat pula penelitian tentang penyelesaian SAT Problem menggunakan Algoritma Genetika. Tetapi dari berbagai aplikasi SAT Solver tersebut memerlukan resource dan spesifikasi komputer yang besar untuk menjalankannya, dan dalam prosesnya membutuhkan waktu yang lama karena output solusi yang dihasilkan berupa eskak. Eskak adalah proses penyelesaian SAT Problem dengan masukan file berbentuk CNF yang saat diproses dengan menempatkan setiap nilai variabel proposisi harus menampilkan klausa dengan nilai akhir benar atau *satisfiabel*.

Berikut ini adalah beberapa penelitian yang pernah dilakukan pada sebelumnya mengenai *Boolean Satisfiability Problem*, yang mana teori ini bertujuan untuk menjadi dasar dan acuan dalam penelitian yang akan dikembangkan.

1. Berikut adalah rangkuman penelitian oleh E. Marchiori and C. Rossi[11]. Penelitian tersebut berisi tentang pengembangan SAT Solver menggunakan Algoritma Genetika dengan hasil keluarannya berbentuk eskak, bukan pendekatan. Algoritma Genetika yang digunakan dalam menyelesaikan SAT Problem pada penelitian tersebut adalah kombinasi antara Algoritma Genetika dengan Algoritma Pencarian Lokal. Pada penelitian tersebut struktur Algoritma Genetika yang digunakan adalah sebagai berikut:
  - Setiap kromosom merepresentasikan dari setiap calon solusi.
  - *Fitness* merepresentasikan jumlah klausa dengan nilai benar dalam suatu kromosom.
  - GA type merepresentasikan fungsi untuk meneruskan 2 kromosom dengan nilai *fitness* terbaik ke generasi selanjutnya.
  - *Genetic Operator* merepresentasikan fungsi *crossover* dan *mutation*.
2. Selanjutnya adalah penelitian yang dilakukan oleh Ines Lynce dan Joel Ouknine[12]. Penelitian tersebut dilakukan untuk mengatasi permasalahan dalam penyelesaian permainan Sudoku menggunakan penalaran, bukan pencarian. Dalam penelitian tersebut masalah yang diselesaikan diubah dalam bentuk *Conjunctive Normal Form* (CNF) dan kemudian diselesaikan menggunakan SAT Solver. Untuk memecahkan teka-teki sudoku tersebut, setiap masalah diselesaikan dengan empat konfigurasi yang berbeda. Pada setiap konfigurasi terdapat

teknik penyederhanaan yang paling sering digunakan, yaitu *unit propagation*. Keempat konfigurasi tersebut adalah sebagai berikut:

- Unit propagation (up).
  - *Unit propagation* dan *failed literal rule* (up+flr).
  - *Unit propagation* dan *hyper-binary resolution* (up+hyper).
  - *Unit propagation* dan *binary failed literal rule* (up+binflr).
3. Penelitian yang dilakukan oleh Dicky Puja Pratama[13] dan penelitian sejenis yang dilakukan oleh T.Hidayat dan A. Bahariyanto Irhasni[14]. Penelitian tersebut berisi tentang pengembangan SAT Solver menggunakan algoritma DPLL dengan bahasa prolog dan bahasa JAVA.
  4. Penelitian yang dilakukan oleh M. Arnesz Setiawan[15]. Penelitian tersebut berisi tentang pengembangan SAT Solver menggunakan pemrograman JAVA dan konsep pendekatan solusi menggunakan *Single Point Crossover* dan *Bit Flip Mutation* Algoritma Genetika.

### B. Dasar Teori

Pada bagian ini terdapat beberapa teori yang akan menjadi dasar dan acuan dalam penelitian yang dilakukan seperti SAT Problem, NP-Complete, Algoritma Genetika, *Conjunctive Normal Form*, Netbeans dan Java. Penjelasan lebih lengkapnya sebagai berikut:

#### 1. Boolean Satisfiability Problem (SAT Problem)

*Boolean Satisfiability Problem* (SAT Problem) adalah permasalahan tentang apakah sebuah formula logika *boolean* (proposisi) yang diberikan dengan operasi *boolean* seperti *AND*, *OR*, dan *NOT*, dan jika masing-masing variabelnya diberikan nilai *true* atau *false* maka dapat menghasilkan nilai *true*[16]. Dengan kata lain SAT Problem merupakan proses untuk mengetahui apakah sebuah formula logika *boolean* adalah sebuah formula yang *satisfiable* atau *unsatisfiable*. Yang mana sebuah formula logika *boolean* dikatakan sebagai formula yang *satisfiable* apabila terdapat rangkaian nilai kebenaran pada komponen (variabel) penyusunnya yang membuat formula menjadi benar TRUE dan sebaliknya jika komponen (variabel) penyusunnya membuat formula menjadi FALSE maka disebut *unsatisfiable*[2].

#### 2. Non Polynomial-Complete

*Non Polynomial-Complete* (NP-Complete) adalah salah satu algoritma yang termasuk dalam *Non Polynomial* (NP). Algoritma *Non Polynomial* adalah algoritma yang kompleksitas waktu kasus terburuknya tidak dibatasi oleh fungsi polinom dari ukuran masukannya[17]. Sedangkan yang dimaksud dengan NP-Complete adalah suatu permasalahan NP yang dapat dikurangi ke dalam waktu polinomial[18]. Fungsi polinom sendiri berarti persamaan yang memiliki pangkat bertingkat pada

suatu variabel. Pangkat tertinggi pada sebuah polynomial adalah derajat. Misalnya sebuah persamaan dengan suku banyak yaitu  $x^3-x+3$ , suku banyak tersebut memiliki derajat 3.

### 3. Algoritma Genetika

Algoritma Genetika adalah suatu algoritma komputer yang terinspirasi dari teori evolusi Darwin. Algoritma ini dikembangkan oleh Goldberg yang menyatakan konsep algoritma genetika adalah ilmu komputasi untuk menyelesaikan suatu permasalahan untuk mencari solusi dengan cara yang alami. Algoritma Genetika merupakan tipe algoritma evolusioner yang paling sering dan banyak diimplementasikan pada permasalahan yang kompleks seperti masalah optimasi dengan model matematikanya yang sangat kompleks. Waktu komputasi yang diperlukan Algoritma Genetika juga cenderung stabil.

Susunan terkecil dari algoritma Genetika adalah karakter, simbol, atau bilangan dari sebuah permasalahan. Susunan terkecil ini merupakan sebuah nilai yang disebut gen atau alel. Rangkaian gen atau alel merupakan penyusun suatu kromosom. Rangkaian kromosom disebut dengan populasi. Setiap kromosom merupakan representasi dari sebuah solusi.

### 4. Conjunctive Normal Form

Formula CNF (*Conjunctive Normal Form*) adalah suatu formula yang telah memenuhi syarat sebagai formula normal form. Syarat formula bisa disebut sebagai normal form adalah sebagai berikut [16]:

1. Dalam sebuah formula memiliki operator  $\wedge$ ,  $\vee$ , dan  $\sim$ .
2. Dan operator  $\sim$  hanya bisa diterapkan pada bilangan proposisi.

Formula berikut adalah contoh-contoh formula CNF yang memenuhi syarat umum digunakan:

1.  $w \wedge x$
2.  $w \wedge (y \vee \sim z)$
3.  $w \wedge x \wedge (\sim w \vee \sim x)$
4.  $(w \vee \sim x) \vee (y \vee \sim z)$
5.  $(w \vee \sim x) \wedge x \wedge (\sim w \vee \sim y)$
6.  $(w \vee \sim x) \vee (y \vee \sim z) \wedge (x \vee z)$
7.  $(w \vee \sim x) \wedge (x \vee \sim y) \wedge (y \vee \sim w)$

Dari contoh diatas seperti nomor 4 yang dimaksud dengan literal adalah a,  $\sim b$ , c, dan  $\sim d$ . Sedangkan yang dimaksud dengan klausa adalah  $(a \vee \sim b)$  dan  $(c \vee \sim d)$ .

### 5. JAVA

JAVA adalah bahasa pemrograman komputer tingkat tinggi yang berorientasi objek. Program java tersusun atas bagian-bagian yang disebut dengan kelas. Kelas terdiri dari metode-metode yang melakukan pekerjaan dan mengembalikan informasi setelah melakukan tugasnya. Pada

setiap kelas juga terdapat atribut tentang sifat-sifat yang berupa nilai atau kondisi yang dimiliki kelas tersebut.

### 6. Netbeans IDE

Netbeans adalah suatu proyek yang disponsori oleh Sun Microsystem sejak tahun 2000. Pada proyek tersebut dihasilkan dua produk yang terdiri dari Netbeans IDE dan Netbeans Platform. Netbeans IDE adalah produk yang digunakan untuk melakukan pemrograman, baik menulis kode, mengompilasi, mencari kesalahan, dan mendistribusikan program. Sedangkan Netbeans Platform adalah sebuah modul untuk kerangka awal dalam membangun aplikasi desktop berskala besar. Netbeans merupakan salah satu IDE yang dapat digunakan untuk menulis pemrograman JAVA. Disisi lain, Netbeans juga menyediakan paket *resource* yang lengkap dalam melakukan pemrograman, mulai dengan pemrograman standart, pemrograman perangkat mobile dan enterprise yang mampu beroperasi di berbagai platform.

## III. HASIL DAN PEMBAHASAN

Untuk menunjang penelitian yang akan dilakukan tentu perlu suatu metodologi. agar penyelesaian makalah dan tugas akhir berjalan dengan baik dan terstruktur. Metodologi yang digunakan adalah:

### A. Pengambilan Data

Data yang akan digunakan untuk melakukan penelitian diambil melalui internet pada tautan berikut (<https://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html>) [21]. Tautan tersebut berisi tentang file CNF yang akan digunakan untuk menyelesaikan SAT Problem dengan SAT Solver.

### B. Analisis Kebutuhan

Dalam sistem yang akan dibangun diperlukan sebuah masukan berupa data yang akan digunakan untuk menunjang proses yang akan dijalankan. Adapun proses yang dilakukan adalah sebagai berikut:

- Proses membaca data masukan logika proposisi dalam bentuk CNF (*Conjunctive Normal Form*).
- Proses inisiasi kromosom.
- Proses regenerasi
- Proses evaluasi kromosom untuk menghitung nilai *fitness*.
- Proses seleksi kromosom.
- Proses *crossover* menggunakan *Two Point Crossover*.
- Proses mutasi menggunakan *Swap Mutation*.
- Proses perulangan hingga menghasilkan kromosom terbaik.

### C. Rancangan Pemodelan

Dilakukan perancangan model Algoritma Genetika untuk menyelesaikan permasalahan SAT Problem. Misalkan terdapat sebuah formula F adalah formula dalam bentuk CNF, yaitu:  $F = w \wedge x \wedge (\sim y \vee \sim z)$

### 1. Inisiasi

Inisiasi dilakukan untuk memunculkan himpunan solusi secara acak yang terdiri atas jumlah kromosom yang disebut sebagai populasi. Misal terdapat 4 individu dalam sebuah populasi. Kemudian dalam satu individu dibentuk gen atau alel secara acak. Dalam SAT Problem gen atau alel terdiri dari 2 kemungkinan, yang pertama variabel yang bernilai 0 yang merepresentasikan dari variabel proposisi dengan nilai *false* atau yang kedua variabel yang bernilai 1 yang merepresentasikan dari variabel proposisi dengan nilai *true*. Berikut adalah contohnya:

- $K[1]=[w \ x \ y \ z]=[0 \ 0 \ 0 \ 0]$
- $K[2]=[w \ x \ y \ z]=[0 \ 0 \ 1 \ 1]$
- $K[3]=[w \ x \ y \ z]=[0 \ 1 \ 1 \ 0]$
- $K[4]=[w \ x \ y \ z]=[1 \ 1 \ 0 \ 0]$

K : Kromosom yang terdiri dari alel atau gen

### 2. Evaluasi

Pada tahap ini evaluasi kromosom bertujuan untuk menentukan nilai fungsi objektif yang didapatkan menggunakan rumus tertentu. Rumus yang digunakan untuk menyelesaikan penelitian ini.

$$F_{Objektifkromosom} = \frac{\text{JumlahKlausula}}{((w \wedge x) + (\sim x \vee \sim z))} \quad (1)$$

JumlahKlausula adalah jumlah gen/alel penyusun dalam sebuah kromosom yaitu 4.

Berikut contoh selengkapnya:

- $F_{Objektif}(K[1]) = 4 - ((0 \wedge 1) + (0 \vee \sim 1))$   
 $F_{Objektif}(K[1]) = 3$
- $F_{Objektif}(K[2]) = 4 - ((0 \wedge 0) + (\sim 1 \vee \sim 1))$   
 $F_{Objektif}(K[2]) = 3$
- $F_{Objektif}(K[3]) = 4 - ((0 \wedge 1) + (\sim 1 \vee \sim 0))$   
 $F_{Objektif}(K[3]) = 2$
- $F_{Objektif}(K[4]) = 4 - ((1 \wedge 1) + (\sim 1 \vee \sim 1))$   
 $F_{Objektif}(K[4]) = 1$

### 3. Seleksi Kromosom (Rank Based Selection)

Seleksi kromosom digunakan untuk menentukan hanya kromosom-kromosom yang memiliki kualitas terbaik yang dapat melanjutkan peranannya dalam proses berikutnya. Pada penelitian ini menggunakan *Rank Based Selection* dengan mengurutkan individu berdasarkan nilai *fitness* pada populasi dan kemudian menghitung nilai probabilitas seleksi sesuai nilai masing-masing individu. Tahapan ini sekaligus berfungsi untuk menentukan nilai *fitness*. Pada penelitian ini rumus yang digunakan untuk menentukan nilai *fitness* yaitu:

$$\text{nilaifitness} = 1/(1+F_{Objektif}) \quad (2)$$

Perlu diketahui untuk mengatasi penghitungan *error* akibat pembagian angka nol maka  $F_{Objektif}$  perlu

ditambah dengan nilai 1. Berikut contoh selengkapnya:

- $\text{nilaifitness}K[1]=1 / (1 + F_{Objektif}[1])$   
 $= 1 / (1+3)$   
 $= 0,25$
- $\text{nilaifitness}K[2]=1 / (1 + F_{Objektif}[2])$   
 $= 1 / (1+3)$   
 $= 0,25$
- $\text{nilaifitness}K[3]=1 / (1 + F_{Objektif}[3])$   
 $= 1 / (1+2)$   
 $= 0,33$
- $\text{nilaifitness}K[4]=1 / (1 + F_{Objektif}[4])$   
 $= 1 / (1+1)$   
 $= 0,5$
- $\text{totalnilaifitness} = 0,25 + 0,25 + 0,33 + 0,5$   
 $= 1,33$

Selanjutnya masuk pada proses mencari nilai probabilitas. Rumus yang digunakan dalam mencari nilai probabilitas ini adalah:

$$P[x] = \text{nilaifitness}[x] / \text{total\_fitness} \quad (3)$$

Contohnya adalah sebagai berikut:

- $P[1] = 0,25 / 1,33 = 0,187$
- $P[2] = 0,25 / 1,33 = 0,187$
- $P[3] = 0,33 / 1,33 = 0,248$
- $P[4] = 0,5 / 1,33 = 0,375$

Selanjutnya menghitung nilai kumulatif. Berikut adalah hasilnya:

- $Kum[1] = 0,187$
- $Kum[2] = P[1] + P[2] = 0,374$
- $Kum[3] = P[1] + P[2] + P[3] = 0,658$
- $Kum[4] = P[1] + P[2] + P[3] + P[4] = 1,03$

Selanjutnya membuat bilangan acak sebagai berikut:

- $A[1] = 0,58$
- $A[2] = 0,75$
- $A[3] = 0,15$
- $A[4] = 0,85$

Hasil akhir populasi setelah melalui proses tahapan diatas menjadi:

- $K[1] = K[3]$
- $K[2] = K[3]$
- $K[3] = K[1]$
- $K[4] = K[4]$

### 4. Crossover (Two Point Crossover)

Kawin silang atau *crossover* adalah proses untuk menghasilkan dua kromosom berbeda dengan menukar dua informasi dari dua induk kromosom yang berbeda. Pada tahap ini menggunakan *Two Point Crossover*. Berikut contoh selengkapnya:

Pertama buat bilangan acak sebanyak 4 sesuai dengan jumlah populasi:

- $A[1] = 0,189$
- $A[2] = 0,255$
- $A[3] = 0,755$
- $A[4] = 0,025$

Selanjutnya dilakukan persilangan antara:

- $K[1] \times K[4]$
- $K[4] \times K[1]$

Dengan membangun bilangan acak untuk titik potong atau cut point dengan (*Two Point Crossover*):

- $Cut[1] = 2$
- $Cut[2] = 2$

Proses *crossover* dengan *Two Point Crossover* sebagai berikut :

$$K[1] = [0 | 1 | 1 | 0]$$

$$K[4] = [1 | 1 | 0 | 0]$$

$$\text{Offspring}[1] = K[1] \times K[4]$$

$$= [0 | 1 | 1 | 0] \times [1 | 1 | 0 | 0]$$

$$= [1 | 1 | 0 | 0]$$

$$\text{Offspring}[4] = K[4] \times K[1]$$

$$= [1 | 1 | 0 | 0] \times [0 | 1 | 1 | 0]$$

$$= [0 | 1 | 1 | 0]$$

Contoh ilustrasi  $\text{Offspring}[1] = K[1] \times K[2]$  pada Gambar 1.

K[1]	0	1	1	0
K[4]	1	1	0	0
Offspring[1]	1	1	0	0

Gambar 1. Contoh *Two Point Crossover*

Setelah melewati proses kawin silang dengan *Two Point Crossover* maka hasil populasinya adalah sebagai berikut:

- $K[1] = [1 | 1 | 0 | 0]$
- $K[2] = [0 | 1 | 1 | 0]$
- $K[3] = [0 | 0 | 0 | 0]$
- $K[4] = [0 | 1 | 1 | 0]$

#### 5. Mutasi (Swap Mutation)

Mutasi merupakan tahap untuk menghasilkan suatu individu baru dengan cara memodifikasi gen/alel dalam satu individu. Pada tahap ini proses mutasi yang digunakan adalah *Swap Mutation*. Misalkan yang mutasi adalah gen/alel ke-2 dan gen/alel ke-4

kromosom(3) bermutasi dengan gen/alel ke-2 dan gen/alel ke-4 kromosom(4). Maka nilai gen/alel tersebut akan berganti dari 1 menjadi 0 begitu sebaliknya 0 menjadi 1.

$$\text{Kromosom}[3] = [0;0;0;0] \text{ menjadi } [0;1;0;0]$$

$$\text{Kromosom}[4] = [0;1;1;0] \text{ menjadi } [0;0;1;0]$$

Berikut contoh ilustrasi mutasi pada Gambar 2.

K[3]	0	0	0	0
K[4]	0	1	1	0
Mutasi K[3]	0	1	0	0
Mutasi K[4]	0	0	1	0

Gambar 2. Contoh *Swap Mutation*

Maka hasil kromosom setelah melewati tahap mutasi adalah sebagai berikut:

- $K[1] = [1 | 1 | 0 | 0]$
- $K[2] = [0 | 1 | 1 | 0]$
- $K[3] = [0 | 1 | 0 | 0]$
- $K[4] = [0 | 0 | 1 | 0]$

#### 6. Regenerasi

Setelah selesai melalui proses mutasi, maka hasilnya akan diregenerasi atau di tempatkan pada populasi baru yang selanjutnya akan di proses secara perulangan.

#### 7. Perulangan

Proses perulangan bertujuan untuk menentukan hasil dari proses-proses sebelumnya untuk menghasilkan kromosom dengan nilai fitness terbaik. Proses perulangan akan berhenti jika telah mencapai batas iterasi yang telah ditentukan. Dan hasilnya menjadi sebuah solusi jika bernilai *satisfiabel* dan sebuah pendekatan solusi jika proses perhitungan hingga batas iterasi menghasilkan nilai *unsatisfiable* dalam menyelesaikan Satisfiability Problem.

### D. Implementasi

Implementasi dikerjakan dengan menerapkan hasil pemodelan Algoritma Genetika pada tahapan yang dilakukan sebelumnya. Tujuannya model Algoritma Genetika yang dibuat mampu menyelesaikan kasus *Satisfiability Problem* dengan menggunakan bahasa pemrograman Java.

#### 1. Pembentukan Kelas

Pembentukan kelas terdiri dari kelas individu, kelas populasi, kelas AlgoritmaGenetika, kelas PembacaWaktuSelesai, dan kelas PembacaFormula.

#### 2. Inisiasi

Berikut adalah gambar code untuk membuat sebuah populasi sebanyak 10 individu, fungsi untuk mengisi nilai dari setiap individu dan setiap individu berisi bilangan integer 0 sampai 1 secara random:

```

demo.population.initializePopulation(10);
for(i = 0; i < population.length; i++){
    random rn = new random();
    population[i] = rn.nextInt(1);
}

```

### 3. Evaluasi Kromosom

Berikut adalah gambar code untuk menghitung fitness dari individu dan fungsi untuk memilih 2 individu terbaik untuk lanjut ke tahap berikutnya:

```

demo.population.calculateFitness();

Fittest = population.getFittest();
secondFittest = population.getSecondFittest();

```

### 4. Seleksi Kromosom

Berikutnya adalah gambar code untuk menghitung setiap fitness dari individu dan fungsi seleksi 2 individu terbaik:

```

demo.population.calculateFitness();

void selection(){
    Fittest = population.getFittest();
    secondFittest = population.getSecondFittest();
}

```

### 5. Crossover

Selanjutnya adalah gambar code untuk fungsi crossover 2 individu terbaik, memilih secara acak point crossover, dan memindahkan nilai antar induk:

```

void crossover(){
    random rn = new random();

    int crossoverPoint = rn.nextInt(population.individuals[0].geneLength);

    for (int i = 0; i < crossoverPoint; i++) {
        int temp = Fittest.genes[i];
        Fittest.genes[i] = secondFittest.genes[i];
        secondFittest.genes[i] = temp;
    }
}

```

### 6. Mutasi

Selanjutnya adalah gambar code fungsi untuk mutasi, memilih individu yang akan dimutasi, dan mengganti nilai terhadap nilai yang dimutasi:

```

void mutation(){
    random rn = new Random();

    int mutation = rn.nextInt(population.individuals[0].geneLength);

    if (Fittest.genes[mutationPoint] == 0){
        Fittest.genes[mutationPoint] = 1;
    }
    else{
        Fittest.genes[mutationPoint] = 0;
    }
    mutationPoint = rn.nextInt(population.individuals[0].geneLength);

    if (secondFittest.genes[mutationPoint] == 0){
        secondFittest.genes[mutationPoint] = 1;
    }
    else{
        secondFittest.genes[mutationPoint] = 0;
    }
}

```

### 7. Regenerasi dan Perulangan

Setelah proses mutasi hasil akan ditempatkan pada populasi baru dan akan melakukan proses perulangan hingga batas iterasi yang ditentukan. Berikut adalah gambar code untuk melakukan perulangan. Batas iterasi yang ditentukan dalam melakukan perulangan yaitu sebanyak 50 kali maka sistem akan mengulangi proses diatas 50 kali:

```

while (iteration() < 50){
    ++demo.generationCount;

    demo.selection();

    demo.crossover();

    if(rn.nextInt()%7 < 5){
        demo.mutation();
    }
}

```

### E. Pengujian

Tahapan selanjutnya adalah dilakukannya pengujian agar mengetahui sistem mampu menyelesaikan permasalahan SAT Problem. Sebuah permasalahan SAT Problem yang diujikan disimpan dalam bentuk file CNF. File CNF tersebut telah mengikuti aturan penulisan pada Kompetisi SAT dengan tautan berikut: (<http://satcompetition.org/>). Komputer yang digunakan untuk melakukan pengujian ini adalah perangkat laptop dengan processor i3-5005U @ 2.00GHz dengan memori internal 4 GigaByte.

Pada penelitian ini permasalahan yang digunakan untuk melakukan pengujian adalah sebuah *sample files*. *Sample files* merupakan file CNF yang digunakan dalam menguji sebuah SAT Solver dengan *resource* yang kecil. File CNF yang digunakan diambil dari internet pada tautan: (<https://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html>). *Sample file* yang digunakan adalah sebagai berikut:

No.	Nama	Jumlah Variabel	Jumlah Klausula
1.	Aim-50-1_6-yes1-4.cnf	50	80
2.	Simple_v3_c2.cnf	3	2

Pengujian dilakukan sebanyak 10 kali percobaan dengan *sample file* tersebut. Dan hasilnya sistem mampu menyelesaikan *sample files* tersebut dalam waktu yang cukup singkat yaitu pada kisaran 0 sampai 7 detik. Hasil lengkapnya sebagai berikut:

Tabel I. Pengujian file sampel kesatu.

No.	Nama Sampel	Waktu Penyelesaian
1.	Aim-50-1_6-yes1-4.cnf	4,38 detik
2.	Aim-50-1_6-yes1-4.cnf	4,90 detik
3.	Aim-50-1_6-yes1-4.cnf	3,91 detik
4.	Aim-50-1_6-yes1-4.cnf	4,76 detik
5.	Aim-50-1_6-yes1-4.cnf	5,87 detik
6.	Aim-50-1_6-yes1-4.cnf	6,19 detik
7.	Aim-50-1_6-yes1-4.cnf	7,09 detik

8.	Aim-50-1_6-yes1-4.cnf	6,57 detik
9.	Aim-50-1_6-yes1-4.cnf	6,27 detik
10.	Aim-50-1_6-yes1-4.cnf	5,74 detik

Tabel II. Pengujian file sampel kedua.

No.	Nama Sampel	Waktu Penyelesaian
1.	Simple_v3_c2.cnf	0,31 detik
2.	Simple_v3_c2.cnf	0,29 detik
3.	Simple_v3_c2.cnf	0,17 detik
4.	Simple_v3_c2.cnf	0,29 detik
5.	Simple_v3_c2.cnf	0,45 detik
6.	Simple_v3_c2.cnf	0,74 detik
7..	Simple_v3_c2.cnf	0,38 detik
8.	Simple_v3_c2.cnf	0,26 detik
9.	Simple_v3_c2.cnf	0,28 detik
10.	Simple_v3_c2.cnf	0,35 detik

#### IV. KESIMPULAN

Penelitian ini bertujuan agar sistem SAT Solver dengan penerapan *Two Point Crossover* dan *Swap Mutation* Algoritma Genetika menggunakan bahasa JAVA mampu menyelesaikan SAT Problem yang diujikan. Sistem ini mampu membuktikan penyelesaian SAT Problem menggunakan Algoritma Genetika dengan memodifikasi pada bagian *Two Point Crossover* dan *Swap Mutation* dengan waktu yang cukup singkat dalam hitungan detik dan menghasilkan ketepatan solusi yang lebih optimal.

Namun, masih terdapat beberapa kekurangan karena SAT Problem yang diujikan cukup sederhana. Dan kerja SAT Solver ini tentu belum bisa dibandingkan dengan mengikuti Kompetisi SAT. Dikarenakan butuh kasus yang besar serta komputer dengan spesifikasi dan performa yang bagus juga.

Pada penelitian selanjutnya, diharapkan mampu untuk mengembangkan konsep penerapan Algoritma Genetika dengan melakukan modifikasi khususnya pada tahap *crossover* dan *mutasi* agar mampu menyelesaikan kasus besar dengan waktu yang relatif singkat. Dan tentunya juga mampu diselesaikan dengan performa dan spesifikasi komputer yang sederhana dalam mencari solusi SAT Problem. Karena untuk memenuhi kebutuhan performa dan spesifikasi komputer yang tinggi juga diperlukan biaya yang besar.

#### REFERENSI

- [1] P. Kalla, "The Boolean Satisfiability ( SAT ) Problem , SAT Solver Technology , and Equivalence Verification," 2019.
- [2] M. R. A. Huth and M. D. Ryan, "Modelling and reasoning about systems," 2000.
- [3] B. Monien and E. Speckenmeyer, "Solving satisfiability in less than  $2n$  steps," *Discret. Appl. Math.*, vol. 10, no. 3, pp. 287–295, 1985, doi: 10.1016/0166-218X(85)90050-2.
- [4] S. A. Cook, "The complexity of theorem-proving procedures," *Proceedings of the Annual ACM Symposium on Theory of Computing*. pp. 151–158, 1971, doi: 10.1145/800157.805047.
- [5] N. D. Priandani and W. F. Mahmudy, "Optimasi Travelling Salesman Problem With Time Windows ( Tsp-Tw ) Pada Penjadwalan Paket Rute Wisata," *Semin. Nas. Sist. Inf. Indones.*, no. November, pp. 2–3, 2015.
- [6] A. Bhattacharjee and P. Chauhan, "Solving the SAT problem using genetic algorithm," *Adv. Sci. Technol. Eng. Syst.*, vol. 2, no. 4, pp. 115–120, 2017, doi: 10.25046/aj020416.
- [7] C. M. Li and Anbulagan, "Heuristics based on unit propagation for satisfiability problems," *Proc. Fifteenth Int. Jt. Conf. Artif. Intell.*, no. 33, pp. 366–371, 1997.
- [8] J. P. M. Silva and K. A. Sakallah, "GRASP—A New Search Algorithm for Satisfiability," *Best ICCAD*, pp. 73–89, 2003, [Online]. Available: [https://www.cs.cmu.edu/~emc/15-820A/reading/grasp\\_iccad96.pdf](https://www.cs.cmu.edu/~emc/15-820A/reading/grasp_iccad96.pdf).
- [9] M. W. Moskewicz and U. C. Berkeley, "Frequency of Dendritic Cells and their expression of Co stimulatory Molecules in Children with Autism Spectrum Disorders."
- [10] Y. Vazel, G. Weissenbacher, and S. Malik, "Boolean Satisfiability Solvers and Their Applications in Model Checking," *Proc. IEEE*, vol. 103, no. 11, pp. 2021–2035, 2015, doi: 10.1109/JPROC.2015.2455034.
- [11] E. Marchiori and C. Rossi, "A Flipping Genetic Algorithm for Hard 3-SAT Problems next section describes the genetic local search scheme," no. January 2013, 2000.
- [12] I. Lynce and J. Ouaknine, "Sudoku as a SAT problem," *9th Int. Symp. Artif. Intell. Math. ISAIM 2006*, pp. 1–9, 2006.
- [13] D. Puja Pratama, "PENYELESAIAN BOOLEAN SATISFIABILITY PROBLEM DENGAN ALGORITMA DAVIS PUTNAM LOGEMANN LOVELAND (DPLL) MENGGUNAKAN JAVA," 2018.
- [14] T. Hidayat and A. B. Irhasni, "SAT Solver dengan DPLL dalam Pemrograman Deklaratif," pp. 49–53, 2018.
- [15] T. Hidayat and M. Arnesz Setiawan, "Solusi pendekatan SAT Problem dengan Algoritma Genetika." 2019.
- [16] M. Y. Vardi, "Boolean satisfiability: Theory and engineering," *Commun. ACM*, vol. 57, no. 3, p. 5, 2014, doi: 10.1145/2578043.
- [17] O. Darmawan and N. Kusumastuti, "Konstruksi pelabelan sisi ajaib super pada graf ulat," vol. 03, no. 3, pp. 227–234, 2014.
- [18] S. Karakashian, "Introduction to NP-Complete Problems," 2008.
- [19] Friesen, J. (2017). *Data Structures and Algorithms in Java, Part 1 : Overview*.
- [20] T. Hidayat, "Logika Proposisi", Yogyakarta: Dar Firqin, 2014.
- [21] <https://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html>