p-ISSN : 2443-2210
e-ISSN : 2443-2229

Jurnal Teknik Informatika dan Sistem Informasi
Volume 4 Nomor 3 Desember 2018

Aransemen Nada Alto, Tenor, dan Bass Menggunakan Algoritma Genetika

http://dx.doi.org/10.28932/jutisi.v4i3.864

Hendry Setiawan[#1], Windra Swastika [#2], Ossie Leona [#3], Oesman Hendra Kelana[#4]

[#]*Teknik Informatika,Universitas Ma Chung*
Villa Puncak Tidar N-01 Malang

[1]hendry.setiawan @machung.ac.id

[2]windra.swastika@machung.ac.id

[3]ossieleona@yahoo.co.id

[4]oesman.hendra@machung.ac.id

Abstract — **Harmony is the key element in producing a piece of music. When a song will be sung by the choir, the song should be arranged in a division of soprano, alto, tenor, and bass voice parts first. Unfortunately, not everyone has the ability to determine the harmonic notes division. There are theories to make a harmonic division of voice parts. Based on the theories, an application for notes arrangement can be developed. The application development uses Application Programming Interface (API) JFugue that can represent music as programming script. Genetic Algorithm is used to implement the application. Chromosome in Genetic Algorithm contains the music notes that have translated into the value of algorithmic music and its duration. Selection is done by giving a reward for the correct notes arrangement and punishment for the wrong notes arrangement. The testing result of this application is already showing a good result. The results of fitness function are in accordance with the verification result from the expert. The average percentage of truth note arrangement is above 98%.**

Keywords— **Genetic Algorithm, Harmonization, Music.**

I. PENDAHULUAN

Dalam dunia musik, harmoni atau keselarasan nada merupakan elemen utama dalam menghasilkan sebuah karya musik. Harmoni merupakan rangkaian yang terbentuk dari penyusunan atau penggabungan nada-nada dengan mengikuti teori musik yang ada. Ketika nada-nada dirangkai menjadi sebuah kesatuan yang selaras, musik akan menjadi sebuah karya yang harmonis dan indah.

Dalam bidang seni suara, pembagian suara merupakan hal yang perlu diperhatikan untuk menciptakan harmonisasi nada yang baik. Setiap pria dan wanita memiliki jenis sura yang berbeda-beda yang dibedakan menjadi tenor, baringtone, bass, sopran, mezzo sopran, dan alto [1].Jenis-jenis suara utama yang digunakan dalam paduan suara adalah sopran, alto, tenor, dan *bass*. Sopran merupakan jenis suara tinggi pada penyanyi wanita. Pada umumnya penyanyi sopran akan mengikuti nada asli dari sebuah lagu. Alto merupakan jenis suara rendah penyanyi wanita. Tenor merupakan jenis suara tinggi pada penyanyi pria. Suara tenor lebih tinggi dari suara *bass* dan lebih rendah dari suara sopran dan alto. Sedangkan *bass* merupakan jenis suara terendah penyanyi pria.

Ketika sebuah lagu akan dinyanyikan oleh paduan suara, lagu tersebut harus diaransemen dalam pembagian suara sopran, alto, tenor, dan *bass* terlebih dahulu. Sayangnya, tidak semua orang memiliki kemampuan dalam menentukan nada pada pembagian suara. Bahkan seorang penyanyi belum tentu memiliki kemampuan untuk menentukan nada dalam pembagian suara secara otomatis menggunakan *feeling*. Pembagian suara memiliki teori agar nada dapat selaras, sehingga dapat dibuat sebuah aplikasi untuk aransemen nada dalam pembagian suara.

Aransemen nada alto, tenor, dan *bass* memiliki variasi yang beragam, sehingga memungkinkan untuk menerapkan algoritma pencarian. Pencarian nada alto, tenor, dan *bass* berdasarkan nada sopran dapat dilakukan menggunakan teknik optimasi. Teknik optimasi merupakan sebuah cara yang dilakukan untuk memberikan sebuah rekomendasi hasil terbaik dari sebuah permasalahan. Teknik optimasi mampu memberikan banyak manfaat dalam pengambilan keputusan sehingga dapat diterapkan pada berbagai permasalahan di berbagai bidang. Salah satu algoritma optimasi yang dikembangkan oleh John Holland adalah Algoritma Genetika yang pertama kali didasarkan dengan menggunakan representasi string biner [2]. Algoritma ini memodelkan proses genetika biologi dalam melakukan optimasi proses pencarian solusi terbaik dari suatu permasalahan. Imlementasi Algoritma Genetika untuk pencarian komposisi musik dalam not balok yang dilakukan dengan mengkombinasikan beberapa operator seperti mutasi dan seleksi. Untuk mutasi dilakukan dengan beberapa variasi yaitu dilakukan dengan perubahan tone untuk satu oktaf, perubahan satu tone, serta pertukaran not yang berdekatan. Untuk seleksi dilakukan dengan eliminasi solusi

yang memiliki nilai fitness terendah serta membatasi adanya duplikasi solusi [3]. Implementasi Algoritma Genetika yang lain dalam pengenalan *speech*, algoritma ini dapat menyelesaikan permasalahan nonlinier, diskrit dan permasalahan dengan batasan tertentu untuk DTW [4].

Penelitian yang telah dilakukan dalam pencarian komposisi musik memiliki kemiripan dengan pencarian nada yang didasarkan pada input yang berupa nada sopran. Selanjutnya dikembangkan sebuah aplikasi berbasis *desktop* untuk aransemen nada sopran, alto, tenor, dan *bass*. Aplikasi ini dibuat dengan mengimplementasikan Algoritma Genetika. Melalui aplikasi ini diharapkan pengguna mendapatkan pembagian suara yang harmonis dari sebuah lagu.

II. TINJAUAN PUSTAKA

A. *Not Angka*

Notasi musik dapat dituliskan menggunakan not balok maupun not angka. Not angka merupakan penulisan notasi musik dengan menggunakan angka dan beberapa tanda sebagai simbolnya. Beberapa aturan dalam penulisan not angka antara lain:

- Nada C D E F G A B digambarkan oleh angka 1 2 3 4 5 6 7 yang dibaca do re mi fa sol la si.
- Angka 0 menggambarkan tanda istirahat.
- Garis miring ke kanan (/) pada angka menggambarkan setengah nada lebih tinggi. Sedangkan garis miring ke kiri (\ ) pada angka menggambarkan setengah nada lebih rendah.
- Tanda titik di bagian atas angka menggambarkan satu oktaf lebih tinggi. Tanda titik di bagian bawah angka menggambarkan satu oktaf lebih rendah. Sedangkan tanda titik saja menggambarkan perpanjangan nada.
- Garis mendatar di bagian atas angka menggambarkan pembagian nilai-nilai ketukan dari nada.

B. *Tanda Birama*

Tanda birama merupakan tanda yang terdiri dari dua angka yang menggambarkan tempo dalam bermain musik, misalnya 4/4, 2/4, dan 3/4. Tanda birama terletak pada garis paranada dan diletakkan di awal lagu atau awal bar. Angka penyebut pada tanda birama menyatakan jumlah ketukan dalam sebuah bar, untuk angka bagian pembilang dari tanda birama menyatakan nilai not yang dianggap dalam satu ketuk [5].

C. *Nilai Not dan Tanda Istirahat.*

Masing-masing not dapat memiliki nilai yang berbeda-beda dan bentuk yang berbeda-beda, hal itu tergantung pada perbedaan ketukan. Ragam dari not yaitu dengan nilai *whole note*, *half note*, *quarter note*, *eighth note*, dan *sixteenth note*. Untuk tanda istirahat / *rest* memiliki panjang ketukan yang mirip serupa dengan not yaitu *whole rest*, *half rest*, *quarter rest*, *eighth rest*, dan *sixteenth rest*. Untuk birama 4/4 dengan nilai *whole note*, maka not tersebut akan memiliki nilai sebesar 4 ketuk. Hal ini juga berlaku untuk *whole rest* pada birama 4/4, maka dalam satu birama akan tanpa not/istirahat.

D. *Chord*

Chord merupakan sekumpulan nada dengan pola tertentu. Setiap nada memiliki frekuensi masing-masing sehingga setiap *chord* dengan susunan nada yang berbeda akan memiliki karakteristik frekuensi yang berbeda pula. Proses klasifikasi *chord* didasarkan pada interval antara nada–nada pembentuknya. Interval dalam musik merupakan jarak antara dua nada. Jenis *chord* yang paling umum dan paling sederhana terdiri dari tiga buah nada, yaitu yang dikenal dengan sebutan triad. Jenis triad yang secara umum digunakan yaitu major dan minor triad [6].

E. *Representasi Chord dan Nada Dalam JFugue*

JFugue merupakan sebuah *open-source Java Application Programming Interface* (API) yang digunakan untuk merepresentasikan music ke dalam Bahasa pemrograman. Fitur *string* yang dimiliki oleh JFugue digunakan untuk menuliskan nada, chord, ketukan, alat musik, dan *track* dari musik Nada dalam JFugue direpresentasikan oleh karakter C, C#/Db, D, E, F, F#/Gb, G, G#/Ab, A, A#/Bb, dan B. Tanda # menyatakan bahwa nada sebelum tanda tersebut mengalami kenaikan sebesar ½, sedangkan tanda b menyatakan nada sebelum tanda tersebut mengalami penurunan sebesar ½. Sedangkan tanda istirahat direpresentasikan oleh karakter R. Penulisan nada dalam JFugue juga dapat direpresentasikan menggunakan angka yang disebut dengan *algorithmic music*. Sebuah *algorithmic music* dilambangkan dengan angka di dalam tanda kurung siku, misalnya [60]. Nilai dari *algorithmic music* telah merepresentasikan nada dan oktaf, sehingga nilai oktaf tidak perlu dispesifikasikan lagi. Nilai dalam *algorithmic music* bertipe *byte* dengan rentang 0 sampai 127, ditunjukkan oleh Tabel I.

TABEL I

NILAI ALGORITHMIC MUSIC DALAM JFUGUE

Oktaf	C	C#/Db	D	…	G	G#/Ab	A	A#/Bb	B
0	0	1	2	…	7	8	9	10	11
1	12	13	14	…	19	20	21	22	23
2	24	25	26	…	31	32	33	34	35
3	36	37	38	…	43	44	45	46	47
4	48	49	50	…	55	56	57	58	59
5	60	61	62	…	67	68	69	70	71
6	72	73	74	…	79	80	81	82	83
7	84	85	86	…	91	92	93	94	95
8	96	96	97	…	102	103	104	105	106
9	108	109	110	…	115	116	117	118	119
10	120	121	122	…	127				

Dalam JFugue oktaf nada direpresentasikan oleh nilai angka 0-10. Penulisan G4 memiliki makna nada G pada oktaf ke-4. Secara default okataf nada pada JFugue akan mengarah ke oktaf ke-5, bila terdapat penulisan nada tanpa oktaf.

F. Chord Dalam JFugue

Beragam jenis chord telah tersedia dalam Jfugue diantaranya chord major, chord minor, chord augmented, chord diminished, dan lain-lain. JFugue merepresentasikan musik dalam sebuah objek bertipe string yang berisi instruksi dan keterangan dari musik yang akan dimainkan. Objek tersebut dinamakan Music String. Music String mengandung sebuah pattern yang akan dimainkan melalui metode Player.play(). Jenis chord beserta kode dan intervalnya seperti pada Tabel II. Root merepresentasikan nada dasar dari chord, sedangkan intervals merepresentasikan jarak antara komposisi nada pada chord.

TABEL II

CHORD DALAM JFUGUE

No	*Common Name*	JFugue *Name*	*Intervals* (0 = *root*)
1	Major	Maj	0, 4, 7
2	Minor	Min	0, 3, 7
3	Augmented	Aug	0, 4, 8
4	Diminished	Dim	0, 3, 6
5	7th(dominant)	Dom7	0, 4, 7, 10
6	Major 7th	Maj7	0, 4, 7, 11
7	Minor 7th	Min7	0, 3, 7, 10
8	Suspended 4th	Sus4	0, 5, 7
9	Suspended 2th	Sus2	0, 2, 7
10	6th (major)	Maj6	0, 4, 7, 9
11	Minor 6th	Min6	0, 3, 7, 9
12	9th (dominant)	Dom9	0, 4, 7, 10, 14
13	Major 9th	Maj9	0, 4, 7, 11, 14
14	Minor 9th	Min9	0, 3, 7, 10 ,14
15	Diminished 7th	Dim7	0, 3, 6, 9
16	Add9	Add9	0, 4, 7, 14
17	Minor 11th	Min11	0, 7, 10, 14, 15, 17
18	11th (dominant)	Dom11	0, 7, 10, 14, 17
19	13th (dominant)	Dom13	0, 7, 10, 14, 16, 21
20	Minor 13th	Min13	0, 7, 10, 14, 15, 21
21	Major 13th	Maj13	0, 7, 11, 14, 16, 21
22	7-5(dominant)	Dom7<5	0, 4, 6, 10
23	7+5(dominant)	Dom7>5	0, 4, 8, 10
24	Major 7-5	Maj7<5	0, 4, 6, 11
25	Major 7+5	Maj7>5	0, 4, 8, 11
26	Minor major 7	Minmaj7	0, 3, 7, 11
27	7-5-9(dominant)	Dom7<5<9	0, 4, 6, 10, 13
28	7-5+9(dominant)	Dom7<5>9	0, 4, 6, 10, 15
29	7+5-9(dominant)	Dom7>5<9	0, 4, 8, 10, 13
30	7+5+9(dominant)	Dom7>5>9	0, 4, 8, 10, 15

Penggunaan Music string Cmaj dilakukan dalam memainkan *chord* C-major. Penulisan *Music String* Cmaj akan memiliki hasil yang sama dengan penulisan *Music String* C+E+G. JFugue akan mengisi komposisi nada lainnya secara otomatis ketika *chord* dimainkan.

G. Durasi Dalam JFugue

Penulisan kode durasi dalam JFugue diletakkan setelah nilai oktaf atau setelah nama nada. Durasi dilambangkan dengan sebuah karakter yang dijabarkan pada tabel III. Apabila durasi tidak dituliskan, maka durasi secara default bernilai ¼.

TABEL III

DURASI DALAM JFUGUE

No	Duration	Character
1	Whole = 1	w
2	Half = 1/2	h
3	Quarter = 1/4	q
4	Eighth= 1/8	i
5	Sixteenth = 1/16	s
6	Thirty-second = 1/32	t
7	Sixty-fourth = 1/64	x
8	One-twenty-eight = 1/128	o

H. Tempo Dalam JFugue

Dalam permainan musik, penggunaan tempo untuk mengatur kecepatan sebuah lagu yang akan dimainkan. Tempo mengindikasikan jumlah ketukan pada setiap menit (beats per minute). Semakin besar nilai dari tempo, maka semakin cepat lagu dimainkan. Sebaliknya, semakin kecil nilai dari tempo, maka semakin lambat lagu dimainkan. Beberapa jenis tempo yang disediakan oleh library JFugue ditunjukkan oleh Tabel IV.

TABEL IV

TEMPO DALAM JFUGUE

Tempo	Beats Per Minutes (BPM)
Grave	40
Largo	45
Larghetto	50
Lento	55
Adagio	60
Adagietto	65
Andante	70
Andantino	80
Moderato	95
Allegretto	110
Allegro	120
Vivace	145
Presto	180
Pretissimo	220

I. Harmoni Musik

Harmoni berperan penting sebagai dasar pengetahuan dan keterampilan dalam membuat sebuah komposisi musik.

Penggunaan harmoni musik yang baik dapat membuat musik menjadi menjadi indah dan enak untuk didengar. Harmoni ini juga memiliki peran dalam pembentukan aransemen nada alto, tenor, dan bass.

J. Jenis Suara Dalam Paduan Suara

Sopran merupakan suara penyanyi yang tertinggi pada klasifikasi vokal dalam musik klasik barat. Saat ini, istilah sopran digunakan untuk penyanyi wanita yang memiliki *range vocal* sopran. Standar dari *range vocal* sopran adalah mulai dari C4 dengan nilai *algorithmic music* sebesar 48 hingga satu setengah oktaf ke atas yaitu A5 dengan nilai *algorithmic music* sebesar 69.

Alto merupakan jenis suara yang lebih rendah dari sopran dan lebih tinggi dari tenor. Suara alto memiliki tingkat ambitus terendah dari seorang wanita. Standar dari *range vocal* alto adalah mulai dari F3 dengan nilai *algorithmic music* sebesar 41 hingga D5 dengan nilai *algorithmic music* sebesar 62. Suara alto memiliki ciri-ciri yaitu rendah, berat, dalam, dan berwibawa.

Tenor merupakan jenis suara tinggi pada pria. Kata tenor berasal dari bahasa latin *tenere* yang memiliki arti menahan. Standar dari *range vocal* tenor adalah mulai dari B2 dengan nilai *algorithmic music* sebesar 35 hingga G4 dengan nilai *algorithmic music* sebesar 55.

Bass merupakan jenis suara rendah pada pria. Suara *bass* dapat dihasilkan oleh suara manusia ataupun alat musik. Suara *bass* berfungsi sebagai dasar dari lagu. Standar dari *range vocal bass* adalah mulai dari E2 dengan nilai *algorithmic music* sebesar 28 hingga C4 dengan nilai *algorithmic music* sebesar 48

K. Aturan Dalam Menyusun Aransemen Nada

Dalam menyusun aransemen nada sopran, alto, tenor dan *bass,* terdapat banyak variasi yang dapat digunakan. Akan tetapi, proses penyusunan aransemen tersebut memiliki teorinya sendiri. Menurut [7], aturan dalam menyusun aransemen nada sopran, alto, tenor dan *bass* adalah sebagai berikut:

- Pergerakan melodi dari *bass* yang baik adalah berlawanan atau menahan pergerakan melodi dari sopran. Apabila pergerakan melodi sopran pada awal birama naik, maka sebaiknya pergerakan melodi *bass* pada awal birama turun atau menahan nada.
- Urutan nada dari tinggi ke rendah adalah sopran, alto, tenor, dan *bass.* Dalam hal ini tidak boleh terjadi *overlapping.*
- Interval dari nada sopran dan alto tidak boleh melebihi satu oktaf.
- Interval dari nada alto dan tenor tidak boleh melebihi satu oktaf.
- Untuk hasil yang harmonis, diusahakan secara vertikal mengandung komposisi nada *chord* yang lengkap.
- Perlu memperhatikan aspek vertikal di setiap waktu. Artinya apabila ditarik garis lurus ke atas, nada-nadanya merupakan isi dari *chord* yang ditentukan.
- Pengulangan nada diprioritaskan untuk nada dasar *chord*, agar tidak menimbulkan interpretasi *chord* lainnya

L. Algoritma Genetika

Algoritma Genetika terinspirasi oleh evolusi dari populasi yang melibatkan kromosom dan gen serta mekanisme dalam bentuk rekombinasi dan mutasi. Sebuah populasi tersusun dari sejumlah individu yang memiliki keunikan kode yang dinamakan kromosom [8]. Algoritma Genetika menggunakan mekanisme pencarian acak dan terstruktur untuk mencari solusi optimal dari suatu permasalahan. Algoritma ini akan melakukan pencarian hasil terbaik dalam setiap generasi. Strategi yang digunakan dalam Algoritma Genetika ini adalah melakukan operasi rekombinasi/*crossover* dan mutasi secara berulang-ulang pada kandidat solusi agar dihasilkan solusi yang lebih baik. Parameter pengukuran baik tidaknya solusi yang dihasilkan akan terlihat pada fungsi evaluasi/ *fitness*. Algoritma Genetika akan melakukan 5 fungi secara umum yang melibatkan insialisasi, *crossover*, mutasi, rating atau perhitungan fungsi evaluasi / fungsi fitness dan seleksi [9]. Inisialisasi awal untuk membentuk kromosom dilakukan secara random dengan menyebarkan nilai solusi pada batasan domain yang ada.

Proses seleksi dilakukan setelah mengetahui nilai *fitness* dari masing-masing kromosom maupun *offspring* yang terbentuk yang bertujuan untuk memilih individu/kromosom yang akan diproses melalui operasi *crossover* maupun mutasi. Kromosom yang baik merupakan calon induk / *parent* akan menghasilkan keturunan/offspring yang baik. Kromosom dengan nilai fitness baik memiliki probabilitas terpilih lebih besar untuk menjadi parent. Beberapa proses seleksi yang sering digunakan diantaranya seleksi mesin roulette, seleksi ranking, turnamen seleksi, dan sebagainya [10].

Operasi *crossover* dilakukan untuk memperoleh solusi lain yang merupakan keturunan/*offspring* dari solusi yang telah ada sebelumnya / kromosom (*parent*). Untuk itu dibutuhkan probabilitas *crossover* yang terletak pada range 0 hingga 1 untuk mengendalikan frekuensi terpilihnya *parent.* Dengan pembangkitan nilai acak antara 0 hingga 1 pada masing-masing kromosom, maka nilai probabilitas crossover akan memilih kromosom dengan nilai acak yang kurang darinya sebagai *parent*. Selanjutnya pasangan parent yang terpilih akan dipotong pada bagian tertentu dan dipertukarkan hingga membentuk *offspring* yang baru.

Operasi mutasi dilakukan untuk menghasilkan offspring baru dengan cara memutasi gen dari kromosom yang terpilih. Mekanisme pemilihan dilakukan dengan membangkitkan secara acak suatu nilai dengan rentan 0 hingga 1 untuk masing-masing gen dalam kromosom dan membandingkan dengan probabilitas mutasi. Pengaturan

probabilitas mutasi pada nilai 0 hingga 1 mengendalikan frekuensi terpilihnya gen yang akan dimutasi. Gen yang memiliki nilai acak kurang dari probabilitas mutasi akan mengalami mutasi.

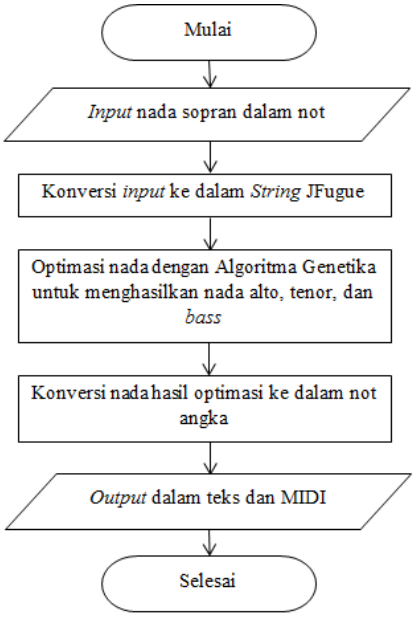
Berikut pseudocode untuk Algoritma Genetika [11].

```
Input : Ukuran Populasi, Pcrossover, Pmutasi
Output : Solusi terbaik (kromosom terbaik)
1.  populasi←Insialisasi populasi (random)
2.  Evaluasi (populasi)
3.  Sbest ←Solusi terbaik(populasi)
4.  While ¬stopkriteria do
5.    parent←cari parent(populasi);
6.    offspring ←0;
7.    Foreach parent1 dan parent2 do
8.     offspring1,offspring2←crossover(parent1,parent2,Pcros);
9.     offspring←mutasi(offspring1,Pmutasi);
10.   offspring←mutasi(offspring2,Pmutasi);
11.  End
12.  Evaluasi(offspring);
13.  Sbest←best(offspring);
14.  Populasi←ganti(populasi,offspring)
15. End
16. Return Sbest
```

Kode program 1

III. RANCANGAN

Rancangan aplikasi terdiri dari lima langkah utama. Pertama, *user* dapat memberikan *input* nada sopran dalam not angka dengan mengikuti beberapa aturan yang telah ditetapkan. Kedua, *input* dari *user* akan dikonversi ke dalam *String* JFugue. Ketiga, *input* yang telah dikonversi dalam *String* JFugue akan dioptimasi dengan menggunakan Algoritma Genetika. Keempat, setelah proses optimasi selesai, nada hasil optimasi dikonversikan dalam bentuk not angka. Kelima, *output* dapat ditampilkan dalam bentuk teks dan dapat dimainkan dengan format MIDI. Penjelasan detail mengenai langkah-langkah tersebut dijabarkan pada subbab selanjutnya. Rancangan aplikasi ditunjukkan oleh Gambar 1.



Gambar 2.Alur Pengerjaan

A. *Desain Algoritma Genetika*

Diawali dengan inisialisasi populasi awal secara acak sebagai representasi solusi dari permasalahan. Kromosom menyimpan nilai *algorithmic music* dan durasi dari setiap nada pada lagu. Nilai pada kromosom didapatkan secara *random* tetapi tidak menyimpang dari aturan aransemen nada yang ada. Representasi kromosom ditunjukkan oleh Tabel V.

TABEL V

REPRESENTASI KROMOSOM

S	Nilai Not 1	Nilai Not 2	Nilai Not 3	Nilai Not 4	…
A	Nilai Not 1	Nilai Not 2	Nilai Not 3	Nilai Not 4	…
T	Nilai Not 1	Nilai Not 2	Nilai Not 3	Nilai Not 4	…
B	Nilai Not 1	Nilai Not 2	Nilai Not 3	Nilai Not 4	…
Durasi	Durasi Not 1	Durasi Not 2	Durasi Not 3	Durasi Not 4	…

Evaluasi yang melibatkan setiap kromosom dalam satu populasi dihitung berdasarkan beberapa komponen [12], yaitu:

- Terdapat *punishment* apabila pergerakan melodi suara sopran pada setiap awal bar sejajar dengan pergerakan melodi suara *bass.*

$$f_1 = n_1 \times 0{,}25 \quad (1)$$

Dimana n_1 menyatakan jumlah bar

p-ISSN : 2443-2210
e-ISSN : 2443-2229

Contoh kromosom 1 yang terdapat pada Tabel VI yang selanjutnya dihitung dengan perhitungan f_1.

TABEL VI

KROMOSOM 1

S	67	64	**64**	**65**	62	**62**	**60**	62	64	65	67	67	67
A	64	60	60	62	49	59	55	55	60	62	64	64	64
T	60	55	55	57	55	55	52	53	55	55	50	60	60
B	48	48	**48**	**55**	55	**55**	**48**	50	52	53	52	52	48
Durasi	32	32	64	32	32	64	32	32	32	32	32	32	64

Terdapat dua pergerakan sejajar antara melodi suara sopran dan *bass* pada awal bar, sehingga nilai

$$f_1 = 2 \times 0{,}25 = 0{,}5$$

- Terdapat *punishment* apabila terjadi *overlapping* di antara susunan nada sopran, alto, tenor, dan *bass*.

$$f_2 = n_2 \times 0{,}5 \quad (2)$$

Dimana n_2 menyatakan jumlah not yang overlap

Contoh kromosom 2 yang terdapat pada Tabel VII yang selanjutnya dihitung dengan perhitungan f_2.

TABEL VII

KROMOSOM 2

S	67	64	64	65	**62**	62	**60**	62	64	65	**67**	67	67
A	64	60	60	62	**49**	59	**55**	55	60	62	**64**	64	64
T	60	55	55	57	**55**	55	**52**	53	55	55	**50**	60	60
B	48	48	48	55	**55**	55	**55**	50	52	53	**52**	52	48
Durasi	32	32	64	32	32	64	32	32	32	32	32	32	64

Terdapat tiga kejadian *overlapping* pada susunan nada tenor dan *bass* di awal bar ke-3, sehingga nilai

$$f_2 = 3 \times 0{,}5 = 1{,}5$$

- Terdapat *punishment* apabila *interval* pada setiap nada sopran dan alto melebihi satu oktaf.

$$f_3 = n_3 \times 0{,}5 \quad (3)$$

Dimana n_3 menyatakan jumlah not

Contoh kromosom 3 yang terdapat pada Tabel VIII yang selanjutnya dihitung dengan perhitungan f_3.

TABEL VIII

KROMOSOM 3

S	67	64	64	65	**62**	62	60	62	64	65	67	67	67
A	64	60	60	62	**49**	59	55	55	60	62	64	64	64
T	60	55	55	57	55	55	52	53	55	55	50	60	60
B	48	48	48	55	55	55	55	50	52	53	52	52	48
Durasi	32	32	64	32	32	64	32	32	32	32	32	32	64

Terdapat satu susunan dimana *interval* nada sopran dan alto melebihi satu oktaf (satu oktaf memiliki 12 nada), sehingga nilai $f_3 = 1 \times 0{,}5 = 0{,}5$

- Terdapat *punishment* apabila *interval* pada setiap nada alto dan tenor melebihi satu oktaf.

$$f_4 = n_4 \times 0{,}5 \quad (4)$$

Dimana n_4 menyatakan jumlah not

Contoh kromosom 4 yang terdapat pada Tabel IX yang selanjutnya dihitung dengan perhitungan f_4.

TABEL IX

KROMOSOM 4

S	67	64	64	65	62	62	60	62	64	65	67	67	67
A	64	60	60	62	49	59	55	55	60	62	**64**	64	64
T	60	55	55	57	55	55	52	53	55	55	**50**	60	60
B	48	48	48	55	55	55	55	50	52	53	52	52	48
Durasi	32	32	64	32	32	64	32	32	32	32	32	32	64

Terdapat satu susunan dimana *interval* nada alto dan tenor melebihi satu oktaf, sehingga nilai

$$f_4 = 1 \times 0{,}5 = 0{,}5$$

- Terdapat *punishment* apabila setiap susunan nada secara vertikal tidak merepresentasikan suatu *chord*.

$$f_5 = n_5 \times 0{,}5 \quad (5)$$

Dimana n_5 menyatakan jumlah not

Contoh kromosom 5 yang terdapat pada Tabel X yang selanjutnya dihitung dengan perhitungan f_5.

TABEL X

KROMOSOM 5

S	67	64	64	65	62	62	60	62	64	65	67	67	67
A	64	60	60	62	49	59	55	55	60	62	64	64	64

T	60	55	55	57	55	55	52	53	55	55	50	60	60
B	48	48	48	55	55	55	55	50	52	53	52	52	48
Durasi	32	32	64	32	32	64	32	32	32	32	32	32	64

Terdapat dua susunan nada sopran, alto, tenor, dan *bass* yang tidak merepresentasikan sebuah *chord*, sehingga nilai $F_5 = 2 \times 0{,}5 = 1$ Nilai *algorithmic music* 62 merepresentasikan nada D, 49 merepresentasikan nada C#, dan 55 merepresentasikan nada G. Kombinasi nada D-C#-G tidak merepresentasikan suatu *chord*. Nilai *algorithmic music* 67 merepresentasikan nada G, 64 dan 52 merepresentasikan nada E, 50 merepresentasikan nada D. Kombinasi nada G-E-D tidak merepresentasikan suatu *chord*.

- Terdapat *punishment* apabila terdapat lebih dari dua nada yang sama dalam satu aspek vertikal

$$F_6 = n_6 \times 0{,}25 \quad (6)$$

Dimana n_6 menyatakan jumlah not

Contoh kromosom 6 yang terdapat pada Tabel XI yang selanjutnya dihitung dengan perhitungan f_6.

TABEL XI

KROMOSOM 6

S	67	64	64	65	**62**	62	60	62	64	65	67	67	67
A	64	60	60	62	**55**	59	55	55	60	62	64	64	64
T	60	55	55	57	**55**	55	52	53	55	55	50	60	60
B	48	48	48	55	**55**	55	55	50	52	53	52	52	48
Durasi	32	32	64	32	32	64	32	32	32	32	32	32	64

Terdapat satu kejadian dimana susunan nada sopran, alto, tenor, dan *bass* memiliki lebih dari dua nada yang sama, sehingga nilai $f_6 = 1 \times 0{,}25 = 0{,}25$

- Terdapat *reward* apabila komposisi nada secara vertikal merepresentasikan suatu *chord* dengan komponen yang lengkap.

$$f_7 = n_7 \times 0{,}5 \quad (7)$$

Dimana n_7 menyatakan jumlah not

Contoh kromosom 7 yang terdapat pada Tabel XII yang selanjutnya dihitung dengan perhitungan f_7.

TABEL XII

KROMOSOM 7

S	**67**	**64**	**64**	65	62	**62**	**60**	62	**64**	65	67	**67**	**67**
A	**64**	**60**	**60**	62	55	**59**	**55**	55	**60**	62	64	**64**	**64**
T	**60**	**55**	**55**	57	55	**55**	**52**	53	**55**	55	50	**60**	**60**
B	**48**	**48**	**48**	55	55	**55**	**55**	50	**52**	53	52	**48**	**48**
Durasi	32	32	64	32	32	64	32	32	32	32	32	32	64

Terdapat delapan kejadian dimana susunan nada sopran, alto, tenor, dan *bass* merepresentasikan suatu *chord* dengan komponen yang lengkap, sehingga nilai $f_7 = 8 \times 0{,}5 = 4$ Susunan nilai 67-64-60-48 merepresentasikan nada G-E-C-C yang merupakan komponen lengkap dari *chord* Cmajor. Susunan nilai 64-60-55-48 merepresentasikan nada E-C-G-C yang merupakan komponen lengkap dari *chord* Cmajor. Susunan nilai 62-59-55 merepresentasikan nada D-B-G yang merupakan komponen lengkap dari *chord* Gmajor. Susunan nilai 64-60-55-52 merepresentasikan nada E-C-G-E yang merupakan komponen dari *chord* Cmajor.

- Fungsi *fitness* total.

$$\max imize\ f_{total} = \frac{f_7 + 0{,}001}{\frac{(f_1 + f_2 + f_3 + f_4 + f_5 + f_6)}{6} + 0{,}01} \quad (8)$$

Persentase banyaknya not benar dalam sebuah lagu didapatkan dengan cara membagi jumlah not yang benar dengan jumlah not pada lagu, kemudian dikalikan dengan 100%. Nilai persentase tersebut digunakan untuk mempermudah penilaian pada proses pengujian dan verifikasi. Rumus dari perhitungan persentase banyaknya not yang benar adalah sebagai berikut:

$$\%truemusicnotes = \frac{\sum true\ music\ notes}{\sum music\ notes\ in\ a\ song} \times 100\% \quad (9)$$

Langkah awal dari proses *crossover* adalah dengan menentukan sebuah bilangan *random* antara 0 sampai 1 untuk setiap kromosom. Apabila bilangan *random* tersebut nilainya lebih kecil dibandingkan dengan nilai probabilitas *crossover*, maka akan dilakukan proses *crossover*. Sebaliknya jika bilangan *random* tersebut nilainya lebih besar dibandingkan dengan nilai probabilitas *crossover*, maka proses *crossover* tidak dilakukan. Proses *crossover* pada aplikasi ini adalah dengan memotong kromosom pada nada alto, tenor, dan *bass* menjadi dua bagian dengan *single one cutting point* setengah dari panjang kromosom seperti pada Gambar 2, kemudian menukar salah satu bagian dengan kromosom induk lainnya [13].

p-ISSN : 2443-2210
e-ISSN : 2443-2229

Kromosom Induk 1

A	64	60	60	62	59	59	55	55	60	62	64	64	64
T	60	55	55	57	55	55	52	53	55	55	60	60	60
B	48	48	48	55	55	55	48	50	52	53	52	52	48

Kromosom Induk 2

A	64	64	60	62	62	59	55	59	60	62	60	60	60
T	52	55	55	59	55	55	52	53	52	50	52	52	52
B	48	48	48	55	47	47	48	50	48	47	47	48	48

Hasil *Crossover* 1 (Ambil bagian kiri dari induk 1 dan bagian kanan dari induk 2)

A	64	60	60	62	59	59	55	59	60	62	60	60	60
T	60	55	55	57	55	55	52	53	52	50	52	52	52
B	48	48	48	55	55	55	48	50	48	47	47	48	48

Hasil *Crossover* 2 (Ambil bagian kiri dari induk 2 dan bagian kanan dari induk 1)

A	64	64	60	62	62	59	55	55	60	62	64	64	64
T	52	55	55	59	55	55	52	53	55	55	60	60	60
B	48	48	48	55	47	47	48	50	52	53	52	52	48

Gambar 2. Proses *Crossover*

Setelah proses *crossover*, langkah selanjutnya dari Algoritma Genetika adalah proses mutasi. Proses mutasi pada aplikasi ini bertujuan untuk memperbaiki not yang salah. Langkah awal dari proses mutasi adalah dengan menentukan sebuah bilangan *random* antara 0 sampai 1 pada setiap not yang salah pada masing-masing kromosom. Apabila bilangan *random* tersebut nilainya lebih kecil dibandingkan dengan nilai probabilitas mutasi, maka akan dilakukan proses mutasi pada not tersebut. Sebaliknya, jika bilangan *random* tersebut nilainya lebih besar dibandingkan dengan nilai probabilitas mutasi, proses mutasi tidak dilakukan. Proses mutasi pada aplikasi ini adalah dengan mengganti not yang salah dengan not baru yang sesuai dengan teori musik yang ada seperti pada Gambar 3

S	67	64	64	65	62	62
A	64	64	60	**62** **r = 0.81**	62	59
T	52	55	**43** **r = 0.07**	59	55	55
B	48	48	36	47	**50** **r = 0.02**	47
Durasi	32	32	64	32	32	64

Hasil Mutasi:

S	67	64	64	65	62	62
A	64	64	60	**62**	62	59
T	52	55	**55**	59	55	55
B	48	48	36	47	**47**	47
Durasi	32	32	64	32	32	64

Gambar 3. Proses Mutasi

Proses seleksi dilakukan setelah proses crossover dengan melakukan perhitungan nilai *fitness* dan nilai persentase not yang benar dari individu-individu baru hasil *crossover* dan mutasi. Kromosom-kromosom tersebut kemudian akan diurutkan berdasarkan nilai *fitness*-nya secara *descending*. Kromosom-kromosom yang telah diurutkan berdasarkan nilai *fitness*-nya akan diambil mulai indeks pertama hingga jumlah populasi pada populasi awal. Kromosom yang terpilih untuk menjadi generasi berikutnya adalah kromosom dengan nilai *fitness* terbesar.

Proses tersebut dijalankan hingga mencapai batas generasi maksimum. Setelah batas generasi maksimum terpenuhi, maka akan dievaluasi nilai *fitness* terbaik dari kromosom yang ada. Kromosom dengan nilai *fitness* terbesar merupakan solusi optimal yang diambil menjadi *output* dari aplikasi.

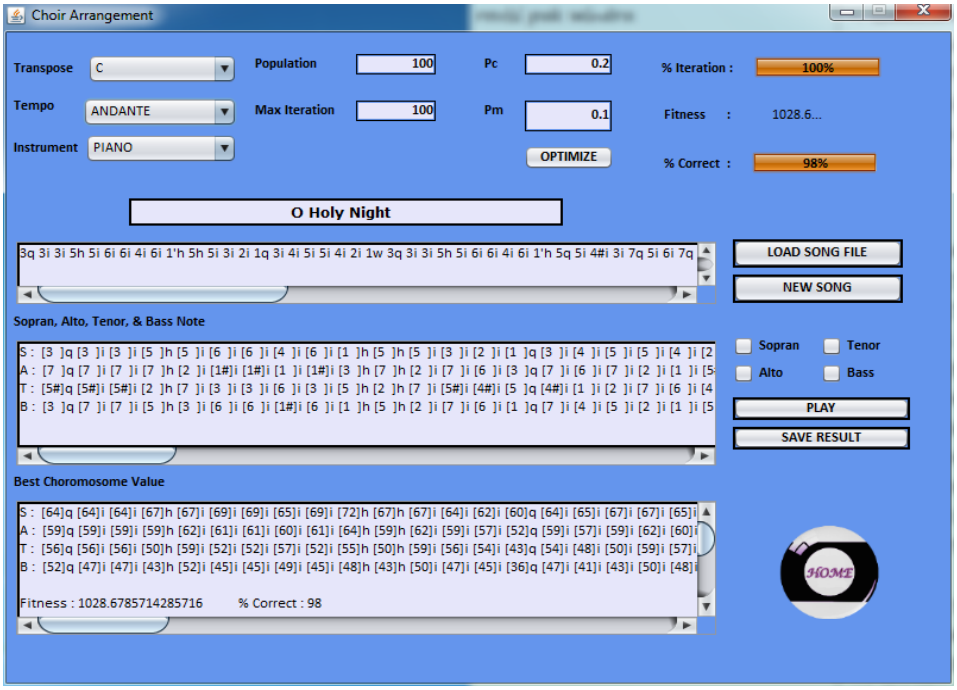
IV. HASIL DAN PEMBAHASAN

A. Pengujian untuk Penentuan probabilitas Crossover, Mutasi dan Generasi

Lagu-lagu yang digunakan untuk pengujian dipilih berdasarkan tingkat kesulitan lagu dan jumlah not yang dimiliki. Lagu-lagu dengan tingkat kesulitan rendah yaitu Tanah Air dan Bangun Pemuda Pemudi. Lagu-lagu dengan tingkat kesulitan sedang yaitu All of Me dan Indonesia Jaya. Sedangkan lagu-lagu dengan tingkat kesulitan tinggi yaitu Gaudeamus Igitur dan O Holy Night. Jumlah not yang dimiliki lagu-lagu tersebut bervariasi. Lagu terpendek adalah Gaudeamus Igitur yang memiliki 47 not, sedangkan lagu terpanjang adalah Indonesia Jaya yang memiliki 128 not.

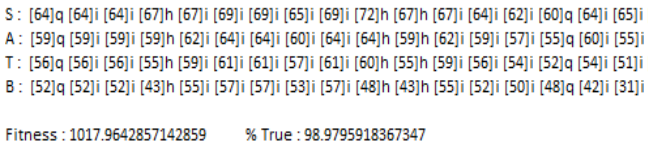
Hasil pengembangan perangkat lunak dalam aransemen alto, tenor, dan bass berisi pengaturan input Algoritma Genetika, input nada, dan beberapa fitur pendukung. Untuk input Algoritma Genetika, pengaturan terkait input probabilitas crossover, probabilitas mutasi, jumlah populasi, serta maksimum iterasi. Input nada secara langsung dituliskan pada aplikasi ataupun load nada, sedangkan hasil aransemsen alto, tenor, dan bass tersimpan dalam format .txt. Beberapa fitur pendukung diantaranya pemilihan chord dalam transpose, pemilihan tempo, serta alat musik dipersiapkan untuk memperdengarkan aransemen alto,tenor, dan bass. Pada lagu O, Holy Night hasil aransemen alto, tenor, dan bass dengan kromosom terbaiknya dapat dilihat pada Gambar 4. Pemberhentian dari proses iterasi diperoleh dengan 2 kriteria yaitu: jumlah maksimum iterasi atau seluruh not benar (yang ditunjukkan oleh % *correct*).

JuTISI
Jurnal Teknik Informatika dan Sistem Informasi



Gambar 4. Tampilan program

Kromosom terbaik dari akan menyimpan nilai *algorithmic music* dari nada sopran, alto, tenor, dan bass beserta durasi masing-masing not tampak pada Gambar 5.



Gambar 5. Kromosom terbaik

Proses pengujian ini bertujuan untuk menentukan jumlah generasi terbaik dengan pertimbangan generasi berikutnya sudah tidak mengalami perubahan nilai not benar yang berarti. Selanjutnya juga dilakukan pencarian untuk menentukan nilai probabilitas *crossover*, serta nilai probabilitas mutasi yang paling optimal sebagai nilai acuan untuk pengujian lagu lainnya yang tampak pada Tabel XIII.

TABEL XIII

PENCARIAN PC, PM, DAN GENERASI TERBAIK

Pc	Pm	Lagu	Generasi				
			25	50	100	200	250
0,5	0,1	Gaudeamus Igitur	83%	95%	97%	97%	97%
		O Holy Night	84%	91%	97%	98%	98%
		All of Me	87%	93%	97%	97%	97%
		Tanah Air	87%	95%	98%	98%	98%
		Bangun Pemudi Pemuda Indonesia Jaya	83%	97%	97%	97%	98%
		Rata-Rata	84%	93%	97%	97%	98%
0,2	0,1	Gaudeamus Igitur	82%	95%	97%	97%	97%
		O Holy Night	84%	95%	98%	98%	98%
		All of Me	89%	93%	97%	98%	98%
		Tanah Air	87%	94%	100%	100%	100%
		Bangun Pemudi Pemuda Indonesia Jaya	89%	90%	98%	98%	98%
		Rata-Rata	85%	94%	97%	97%	97%
0,75	0,3	Gaudeamus Igitur	86%	93%	98%	98%	98%
		O Holy Night	93%	95%	97%	97%	97%
		All of Me	89%	91%	98%	98%	98%
		Tanah Air	93%	95%	97%	97%	97%
		Bangun Pemudi Pemuda Indonesia Jaya	91%	95%	97%	98%	98%
		Rata-Rata	91%	94%	97%	98%	98%

p-ISSN : 2443-2210
e-ISSN : 2443-2229

		Pemudi Pemuda Indonesia Jaya	89%	93%	97%	97%	97%
		Rata-Rata	91%	93%	97%	98%	98%
0,1	0,05	Gaudeamus Igitur	53%	81%	94%	97%	97%
		O Holy Night	63%	87%	98%	98%	98%
		All of Me	61%	85%	97%	97%	97%
		Tanah Air	59%	83%	97%	97%	97%
		Bangun Pemudi Pemuda Indonesia Jaya	65%	87%	98%	98%	98%
			59%	89%	97%	97%	97%
		Rata-Rata	60%	85%	97%	97%	97%

Dari data hasil pengujian apabila dilihat dari hasil penilaian dan waktu yang diperlukan aplikasi, maka nilai parameter yang terbaik adalah Pc = 0,2; Pm = 0,1. Jumlah iterasi yang paling efektif untuk digunakan adalah 100 karena rata-rata pada iterasi ke-100, aplikasi telah mendapatkan nilai yang paling optimal sehingga iterasi selanjutnya tidak menunjukkan perbaikan solusi yang signifikan dari proses optimasi.

B. Hasil Pengujian dari Program dan Verifikasi Pakar

Hasil pengujian dari program dengan formula (9) dibandingkan dengan verifikasi dari pakar tampak pada Tabel XIV.

TABEL XIV

VERIFIKASI PAKAR

No.	Judul Lagu	% Not Benar	% Nilai Verifikasi
1	Gaudeamus Igitur	97,9	95,7
2	O Holy Night	98,9	96,9
3	All of Me	98,6	95,8
4	Tanah Air	98,1	96,2
5	Bangun Pemudi Pemuda	98,5	95,4
6	Indonesia Jaya	98,4	96,8
	Rata-Rata	98,4	96,1

Nilai persentase not benar dari program telah mendekati nilai verifikasi yang diberikan oleh pakar, sehingga dapat disimpulkan bahwa fungsi *fitness* yang digunakan pada program telah sesuai. Hasil tersebut juga menunjukkan bahwa Algoritma Genetika dapat diimplementasikan dengan baik pada *music programming.* Aransemen dan nilai *fitness* yang dihasilkan aplikasi bervariasi, bergantung pada nilai parameter-parameter Algoritma Genetika yang digunakan yaitu jumlah generasi, jumlah generasi maksimal, probabilitas *crossover*, dan probabilitas mutasi.

V. SIMPULAN

Simpulan dari aransemen nada alto, tenor, dan *bass* ini adalah: .

- Algortima Genetika telah berhasil melakukan pencarian nada alto, tenor, dan bass.
- Nilai *fitness* dan nilai persentase not benar yang diperoleh aplikasi telah mendekati nilai yang telah diverifikasi oleh pakar.
- Nilai probabilitas *crossover* dan probabilitas mutasi yang optimal adalah 0.2 dan 0.1.
- Jumlah iterasi maksimal yang efektif adalah 100.
- Rata-rata persentase kebenaran aransemen nada yang dihasilkan aplikasi adalah sebesar 98,44%.
- Hasil aplikasi dan hasil verifikasi memiliki rata-rata perbedaan 2 not.

DAFTAR PUSTAKA

[1] A. Saputra, S. Suwarno,& L. Chrisantyo, "Klasifikasi Suara Manusia Ke Dalam Sopran, Mezzo Sopran, Alto, Barinton, Bass Dengan Self Organizing Map," *Informatika*, vol. 11, no. 1, pp. 23-27, 2015.

[2] O Kramer, *Genetic Algorithm Essentials*, Springger, pp. 6,2017

[3] D. Matic, "A Genetic Algorithm For Composing Music," *Yugoslav Journal of Operation Research*, vol. 20, no. 1, pp. 157-177, 2010.

[4] Z. Beukhellat and A. Belmehdi, "Genetic Algorithms in Speech Recognition Systems," in *International Conference on Industrial Engineering and Operations Management*, Istambul, Turkey, 2012.

[5] Liliana, S. Tedjokusuma ,& R. Alamveta, "Aplikasi Penulisan Notasi Balok dari File MIDI Monophonic," *Jurnal Infromatika*, vol. 10, no. 2, pp. 101-106, 2009.

[6] L. K. timotius & A. Prayoga, "Sistem Pengenalan Chord pada File Music Digital dengan Menggunakan Pitch Class Profiles dan Hidden Markov Model," *Techne Jurnal Ilmiah Elektronika*, vol. 9, no. 2, pp. 133-143, 2010.

[7] I. B. Linggono, Seni Musik Non Klasik Jilid 3, Jakarta: Departemen Pendidikan Nasional, 2008.

[8] H. S. I. Harba & E. S. I. Harba, "Voice Recognition with Genetic Algorithms," *International Journal of Modern Trends in Engineering and Research*, vol. 2, no. 12, pp. 144-155, 2015.

[9] V. Anderling, O. Andreasson, C. Olsson, S. Pavlov, C. Svensson, & J. Wikner, Generation of music through Genetic Algorithms, Goteborg,Sweden: Chalmers University of Technology, 2014.

[10] T. F. Ramadonna, A. Sivia and C. , "Perbandingan Algoritma Genetika dan TSP Untuk Optimalisasi Jaringan Akses Fiber To the Home," *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 3, no. 2, pp. 344-353, 2017.

[11] J. Brownlee, *Clever Algorithms : Nature-Inspired Programming Recipes*. Jason Brownlee, Australia, 2011.

[12] H. Setiawan, O. Leona, W. Swastika and O. H. Kelana, "Desain Aransemen Suara pada Algoritma Genetika," *Seminar Nasional Teknologi Informasi, Komunikasi dan Aplikasinya*, vol. 4, pp. 200-203, 2017.

[13] A. Ball, N. Paul, S. Chuakraborty and S. Sen, "Voice Matching Using Genetic Algorithm," *International Journal of Advanced Computer Research*, vol. 4, no. 1, pp. 305-312, 2014.