BITI 3143

EVOLUTIONARY COMPUTING

PROJECT TITLE:

COMPOSING MUSIC FOR BEGINNER BY USING GENETIC ALGORITHM

LECTURER:

TS. DR. ZERATUL IZZAH MOHD YUSOH

2 BITI S1G1 (2021/2022)

NAME	MATRIC NUMBER
SYAKIRAH HANIM BINTI ZULKERNAIN	B032010116
NURSYAZA NISA BINTI ARFARIZAL	B032010244
MUHAMMAD ZUL HANIF BIN NORDIN	B032020021
MUHAMMAD ANASS BIN SULAIMAN	B032010124

Individual Task Distribution

Type of task	Task	Member's Name
Algorithm Design	• Find sample data • Design crossover strategies • Design mutation	• Nursyaza Nisa • Syakirah Hanim
	• Creating fitness function	• Nursyaza Nisa • Syakirah Hanim • Muhammad Zul Hanif • Muhammad Anass
Coding	• Code C++ GA	• Muhammad Zul Hanif
Report	• Description problem to solve	• Syakirah Hanim
	• Description problem instances	• Nursyaza Nisa
	• Details description about GA	• Syakirah Hanim • Nursyaza Nisa • Muhammad Anass
	• Analysis	• Muhammad Anass

DESCRIPTION OF THE PROBLEM TO SOLVE

Music is an art that combined notes and octaves in music arrangement. This knowledge is particularly important, so that the quality of the outcome is great when played by using any instruments. Refer to the Table 1, there are a lot of combination of notes and octaves that present in music, hence it can lead over choice phenomena when writing music especially for people who lack in this knowledge. Because of this, chances for producing bad music arrangement are high. The example of bad and good music arrangement is shown in Figure 1 and Figure 2.

Notes/Octaves	1	2	3	4	5	6	7	8
C	C1	C2	C3	C4	C5	C6	C7	C8
C#	C#1	C#2	C#3	C#4	C#5	C#6	C#7	C#8
D	D1	D2	D3	D4	D5	D6	D7	D8
D#	D#1	D#2	D#3	D#4	D#5	D#6	D#7	D#8
E	E1	E2	E3	E4	E5	E6	E7	E8
F	F1	F2	F3	F4	F5	F6	F7	F8
F#	F#1	F#2	F#3	F#4	F#5	F#6	F#7	F#8
G	G1	G2	G3	G4	G5	G6	G7	G8
G#	G#1	G#2	G#3	G#4	G#5	G#6	G#7	G#8
A	A1	A2	A3	A4	A5	A6	A7	A8
A#	A#1	A#2	A#3	A#4	A#5	A#6	A#7	A#8
B1	B1	B2	B3	B4	B5	B6	B7	B8

Table 1: Combination notes and octaves

Notes	**C#1**	**G7**	**A3**	**E5**	**C1**	**D#6**	**G2**	**F#7**	**B4**	**A#7**	**F3**	**A1**	**G5**	**B8**	**A2**
Freq	34	3136	220	659	32	1245	98	2960	496	3729	174	55	784	7905	110

Figure 1: Bad music arrangement

Notes	**F#4**	**G#4**	**G#4**	**G#4**	**G#4**	**F4**	**F4**	**F#4**	**A3**	**A3**	**F#3**	**A3**	**A3**	**F#3**	**F2**
Freq	369	415	415	415	415	349	349	369	220	220	185	220	220	185	87

Figure 2: Good music arrangement

According to the problem, the goal of this project is to produce a good music arrangement based on frequency but at the same time have more combination notes and octaves in each line.

DESCRIPTION OF THE PROBLEM'S INSTANCES

This project has been developed with the evolutionary algorithm (EA) to compose music, making it understandable not just for the expert music but also for the beginners. The examples or data of this project has been researched (refer to table 1) before conducting the project. By using Genetic Algorithm, we will be produced 1 bar line or a strip with each strip having 4 notes only. Hence, 16 strips (genes) of 16 different or same notes from different or same octaves will be in 1 chromosome. To evaluate the bar line produced, the smoothness and the melody is used as an indicator of the objectives of this project, which is to produce good music that smooth, blends naturally and at the same time, generating a composition music that is easy for the understanding of music beginners.

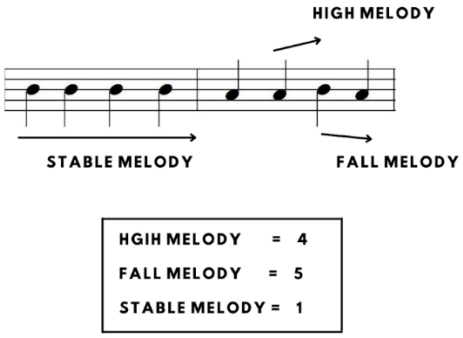


Figure 3: A strip consisting of 4 notes only

As we can see, there are 3 types of melody in the music composition which are: high melody, fall melody and stable melody with the assigned weight of the melody 4, 5, 1, respectively. It is significant to assign the right weight for each type of melody as it will affect the music composition. For example, the stable melody has the least assigned weight to lessen the notes from the same octaves as it will produce a dull and uninteresting music.

The evaluation of composed music will be also in terms of the smoothness which is the measurement of how pleasant the combination of notes is to the ear. Smoothness will be based on the difference in frequency of one note to another note in a strip. A combination of notes that has the least total difference between notes in a strip will have a lower smoothness, making it more pleasant to the ear. Therefore, we can conclude we must control the total difference of frequency in a strip and the total assigned weight of the melody to generate a good music composition.

DETAIL DESCRIPTION ON THE GA'S DESIGN INCLUDING

a. Chromosome Representation

The representation is a starting point to build genetic algorithm model. In this project, we have data of frequency and notes' name based on octave. The frequency and notes' name will be stored in arrays form. Figure 4 shows the arrays of frequency and notes name.

```
const int FREQUENCY[12][8] = { {32,65,130,261,523,1047,2093,4168},
                               {34,69,138,277,554,1109,2217,4435},
                               {36,73,146,293,587,1175,2349,4699},
                               {38,77,155,311,622,1245,2489,4978},
                               {41,82,164,329,659,1319,2637,5274},
                               {43,87,174,349,698.1397,2794,5588},
                               {46,92,185,369,739,1480,2960,5920},
                               {49,98,196,392,784,1568,3136,6272},
                               {52,104,208,415,830,1661,3322,6645},
                               {55,110,220,440,880,1760,3520,7040},
                               {58,116,233,466,932,1865,3729,7459},
                               {61,123,246,493,987,1976,3951,7905} }; //{ {},{},{},{},{},{},{},{},{},{},{},{} }; template

const string NOTES[12][8] = { {" C1"," C2"," C3"," C4"," C5"," C6"," C7"," C8"},
                              {"C#1","C#2","C#3","C#4","C#5","C#6","C#7","C#8"},
                              {" D1"," D2"," D3"," D4"," D5"," D6"," D7"," D8"},
                              {"D#1","D#2","D#3","D#4","D#5","D#6","D#7","D#8"},
                              {" E1"," E2"," E3"," E4"," E5"," E6"," E7"," E8"},
                              {" F1"," F2"," F3"," F4"," F5"," F6"," F7"," F8"},
                              {"F#1","F#2","F#3","F#4","F#5","F#6","F#7","F#8"},
                              {" G1"," G2"," G3"," G4"," G5"," G6"," G7"," G8"},
                              {"G#1","G#2","G#3","G#4","G#5","G#6","G#7","G#8"},
                              {" A1"," A2"," A3"," A4"," A5"," A6"," A7"," A8"},
                              {"A#1","A#2","A#3","A#4","A#5","A#6","A#7","A#8"},
                              {" B1"," B2"," B3"," B4"," B5"," B6"," B7"," B8"} };
```

Figure 4: Arrays of frequency and notes

To make as the representation, we already chose frequency (integer form) to generate randomly from frequency array for each gene of a chromosome. This can help to evaluate each chromosome later.

The genotype (frequency of notes) will come out together with phenotype (note name) in console. The phenotype can help us to track and check the pattern of arrangement in chromosome later.

To store the frequency that randomly generated from frequency array, we need 16 genes for a chromosome to be initialized. The representation of chromosome as shown at Figure 4.

69	174	6	261	61	49	61	65	329	293	784	466	65	196	104	311

C#2	F3	B1	C4	B1	G1	B1	C2	E4	D4	G5	A#4	C2	G3	G#2	D#4

Figure 4: Representation of chromosome based on frequency of combinations of notes and octaves with the phenotype

b. Fitness Function

Fitness function is used to evaluate the performance of each chromosome. To build the fitness function, we already considered the smoothness only for this project. Smoothness is particularly important since it can determine the quality of variation of notes and octaves in a line. To determine the smoothness of a line, frequency of each note based on octaves is important to know first. The table of frequency can be referred at Table 2.

NOTES/ OCTAVES	1	2	3	4	5	6	7	8
C	32	65	130	261	523	1047	2093	4168
C#	34	69	138	277	554	1109	2217	4435
D	36	73	146	293	587	1175	2349	4699
D#	38	77	155	311	622	1245	2489	4978
E	41	82	164	329	659	1319	2637	5274
F	43	87	174	349	698	1397	2794	5588
F#	46	92	185	369	739	1480	2960	5920
G	49	98	196	392	784	1568	3136	6272
G#	52	104	208	415	830	1661	3322	6645
A	55	110	220	440	880	1760	3520	7040
A#	58	116	233	466	932	1865	3729	7459
B	61	123	246	493	987	1976	3951	7905

Table 2: Frequency of notes based on octaves

From the Table 2, we can calculate the smoothness from one combination of notes and octaves to other. The example of calculation can be referred below.

Example of frequency randomly generated by computer:

69	174	65	261	61	49	61	65	329	293	784	466	65	196	104	311

$$Smoothness = |frequency[index1] - frequency[index2]|$$
$$= |69 - 174|$$

$$= |-105|$$
$$= 105$$

The simple calculation gave the idea for creating formula of smoothness for a chromosome. The formula is as below:

$$Smoothness = \sum(|frequency(x) - frequency(x-1)|$$

To evaluate in finding the best chromosome, fitness function is needed. Since we have the smoothness formula, it must be divided into total frequency of the chromosome. The formula of fitness function is:

$$f(x) = \frac{smoothness}{\sum frequency\ for\ a\ chromosome}$$

If the fitness function is low, then the arrangement of notes and octaves in a chromosome is smooth.

c. Strategy Of Parent Selection, Crossover, Mutation and Survival Selection

The tournament selection method is used for parent selection. Based on their fitness function, two players are chosen at random to compete. The winner is chosen to be one of the parents. the selection is repeated until both parents are chosen and parent 1 is not equal to parent 2. Tournament selection is used because it is easier to compute and provides a more equitable chance of being selected as a parent to chromosomes with higher fitness values. More chromosomes with lower fitness values are selected in a tournament with a larger selection size.

Crossover is carried out using a one-point crossover. The generation of crossover point is at random between 0 and the maximum gene length. The genes on the parent chromosome are divided into two segments before and after the crossover point. As shown in the figure below, each child will inherit one segment from each of the parents.

Parent 1	B2	C#2	A1	D#4	D#4	B2	A#2	A2	B3	B3	C5	A4	E5	F5	E5	G#5

Parent 2	B2	C#2	A1	B3	D#4	D#4	D#5	D5	B3	D#4	C5	G#5	E5	F5	E5	G#5

Crossover Point: 6

Children 1	B2	C#2	A1	D#4	D#4	B2	D#5	D5	B3	D#4	C5	G#5	E5	F5	E5	G#5

Children 2	B2	C#2	A1	B3	D#4	D#4	A#2	A2	B3	B3	C5	A4	E5	F5	E5	G#5

Figure 5: One-Point Crossover

The mutation method is the swap mutation. In swap mutation, two random genes in a chromosome are selected randomly. Their values are swapped to create a mutated chromosome, as shown in the figure below.

Prob: 0.6, No Mutation

Children 1	D5	D5	F5	F#5	F#5	F#5	F#5	F#5	A4	A4	D4	D4	D#5	F5	F#5	F#5

Prob: 0.2, Mutation happens

Children 2	D5	D5	F5	F5	F#5	F#5	F#5	A4	A4	A4	D4	D5	D#5	F5	F5	F#5

After Mutation

Childre n 2	D5	D5	F5	F5	F#5	D#5	F#5	A4	A4	A4	D4	D5	F#5	F5	F5	F# 5

Figure 6: Swap Mutation

The survival selection is the process of reducing the number of chromosomes. When the crossover is implemented, the parents will produce the same number of offspring. As a result, the population doubles, but we are unable to increase our population size. Hence, we must eliminate half of the population by using survival selection in which all children will replace the parent. The children always replace the parents in this method.

d. Termination Strategies

The algorithm terminates when it reaches a fixed maximum generation. The number of generations was manually selected. Once the system has reached the specified number of generations, the software will be automatically terminated.

e. Measurement Indices

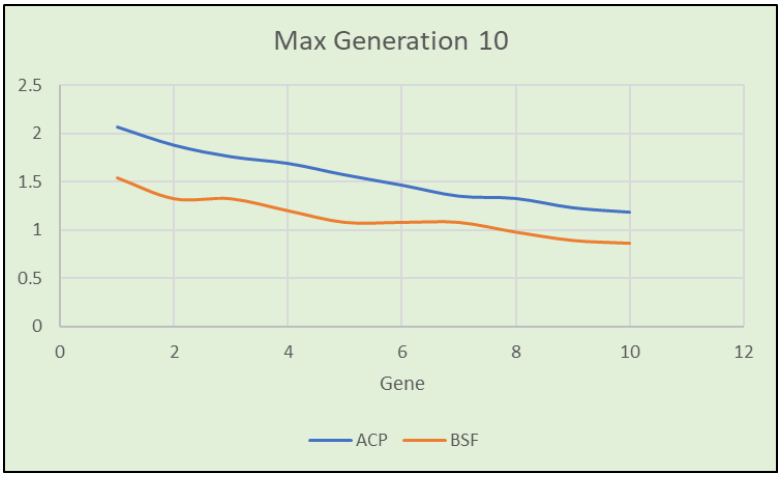
The algorithm is tested using measurement indices. The only measurement indices used in our project are the fitness values. To choose the optimal chromosome, we applied best fitness and average fitness. The population's best individual is represented by its best fitness value, and its average fitness is the average of all its fitness values. Depending on the fitness function, the best fitness value may be either lower or higher than the average fitness value.

RESULT ANALYSIS

The analysis of the results is based on the output of the algorithm. The algorithms are run a total of 5 times. The values for maximum generation are inputted differently for each run. The results of the average fitness value and the best fitness value are then compared but with different parameters which are mutation probability, cross-over probability, maximum generation and population size.

10 Generations

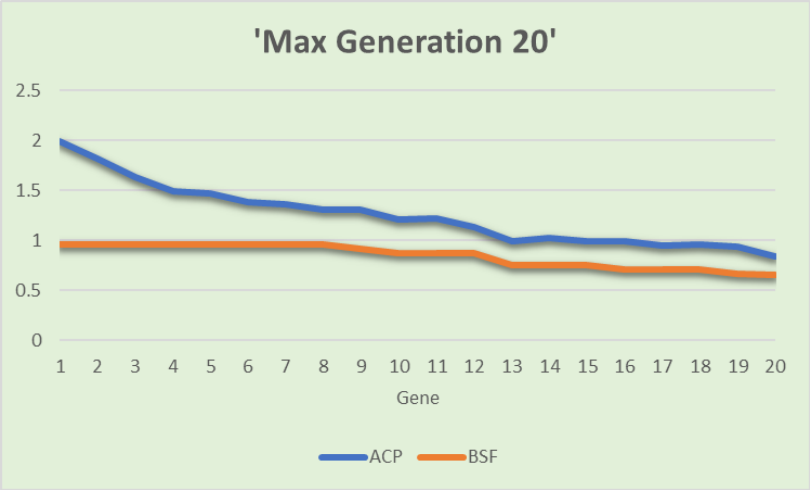
GENE	Population size	Cross-over probability	Mutation probability	Max generation
16	30	0.8	0.3	10



Figures above show the results of the 10 generations with 30 population size. As we can see there is still a gap between the average fitness and best fitness at the end of generations. Hence, the solution is possibly not the best. The peak decreases as it slowly nears the best fitness which is located at for the 1.18046 which is at the 10^{th} generation. The produced results from 10 generations are proved to be not satisfactory yet, so we extend the generations to 20 to 50 as shown below to search for a better result.

20 Generation

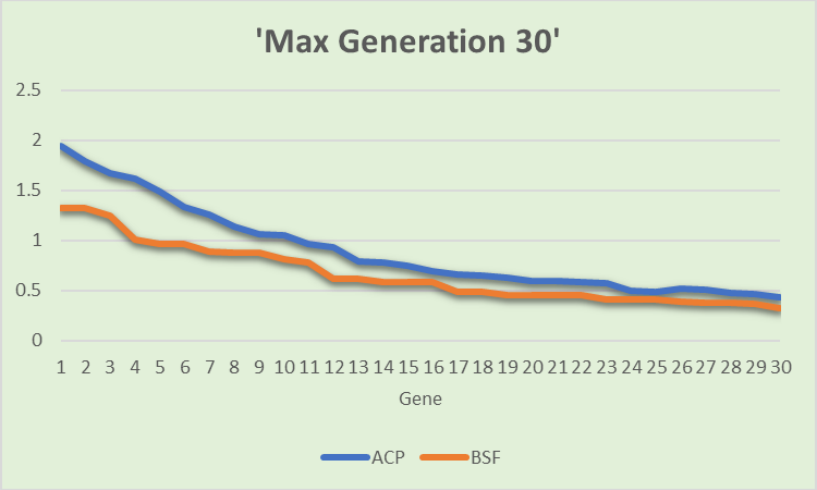
GENE	Population size	Cross-over probability	Mutation probability	Max generation
16	30	0.8	0.3	20



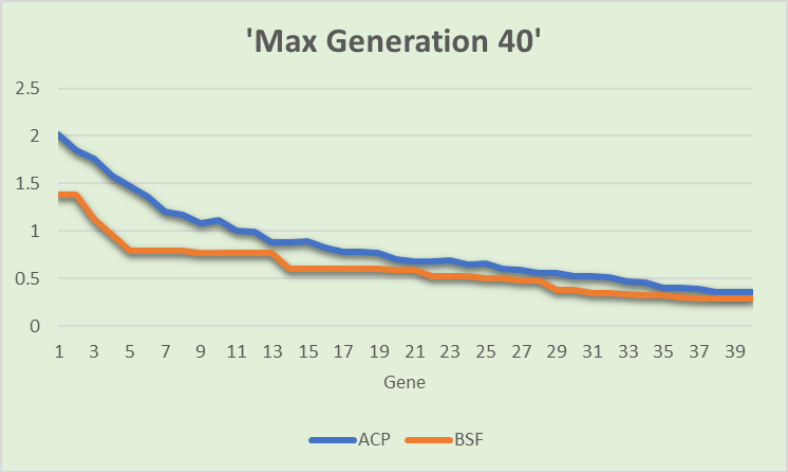
Figures above show the results of the 20 generations with 30 population size. The peak decreases as it slowly nears the best fitness which is located at for the 1.843091which located at the 20th generation. As we can see there is still a gap between the average fitness and best fitness, but it slowly nears the best fitness at the end of the generation. It is shown that the results are better compared to the previous generation. The peak decreases as it slowly nears the best fitness which is located at for the 1.843091 which located at the 20th generation.

30 and 40 Generations

GENE	Population size	Cross-over probability	Mutation probability	Max generation
16	30	0.8	0.3	30



GENE	Population size	Cross-over probability	Mutation probability	Max generation
16	30	0.8	0.3	40

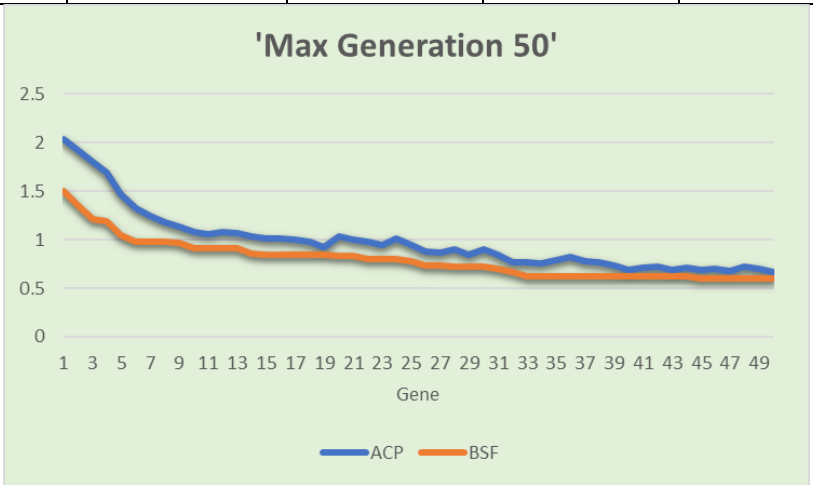


Figures above show results of using 30 and 40 generations with a population size of 30. As we can see, from generation 1 to 5, it has the most changes of fitness function value. The best fitness value decreases in value and affects all the generation throughout generation 5 until 40. From generation 5 to 40, the graph plot does not have bigger changes which approximately 0.02 value changes from generation 5. For the average, the fitness value is not consistent from the first generation until the last generation that we recorded. The drastic change that we can record is between generation 1 until 7 which there was approximately 1 drop fitness value. As the conclusion, the fitness value drops every generation that has evolved. All above the graph did evolve since most of them have changing of its best and average fitness value. But it does

not have a lot of fitness value changes that we can conclude it might not have the best generation evolving.

50 Generation

GENE	Population size	Cross-over probability	Mutation probability	Max generation
16	30	0.8	0.3	50



Figures above show the results for 50 generations with 30 population size. What we observed from all the generations that we used, 50 generations have the best evolving graph and it meets our project requirement to the best. As we can see, 50 generations have the best evolving graph that can conclude it has the best generation evolution. They have the biggest changes when the maximum generation is at 40 to 45 which Is where the line for both fitness starts to converge. This biggest change of graph value decreasing is considered the best evolution of generation.