Sergeev Nikita AI assignment 2 report

Introduction:

In this assignment, I wrote a program for constructing accompaniment for the given melody. To solve this problem, I used an evolutionary algorithm. The first thing to do in this program is to find tonic key of the given melody to contruct the accompaniment using it. After this I'm appling evolutionary algorithm to generate a solution.

Python requirements:

In my solution I'm using python 3.10 with the following libraries:

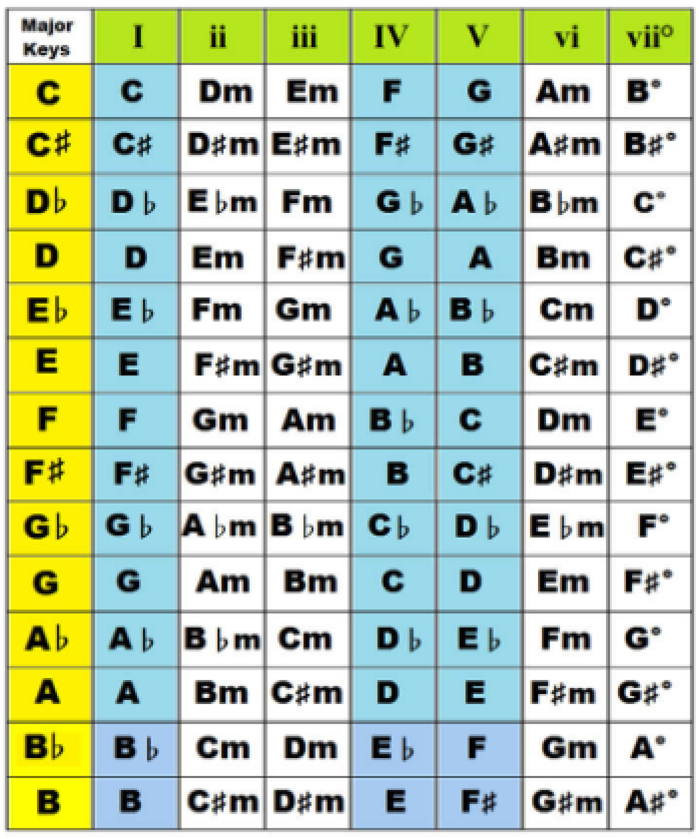
1. NumPy for the mathematical operations
2. Mido library for working with the midi files
3. random library for working with the random variables in the evolutionary algorithm
4. typing library to use advanced type hints in the function for clear and readable code

Required work with the music theory:

Before running an evolutionary algorithm, we should work with the initial melody to take some information from it. We should find the tonic key of the initial melody. And from this tonic key, we will construct a list of step chords. And to construct an accompaniment we will use only these step chords.
Tonic key can found in 2 ways:

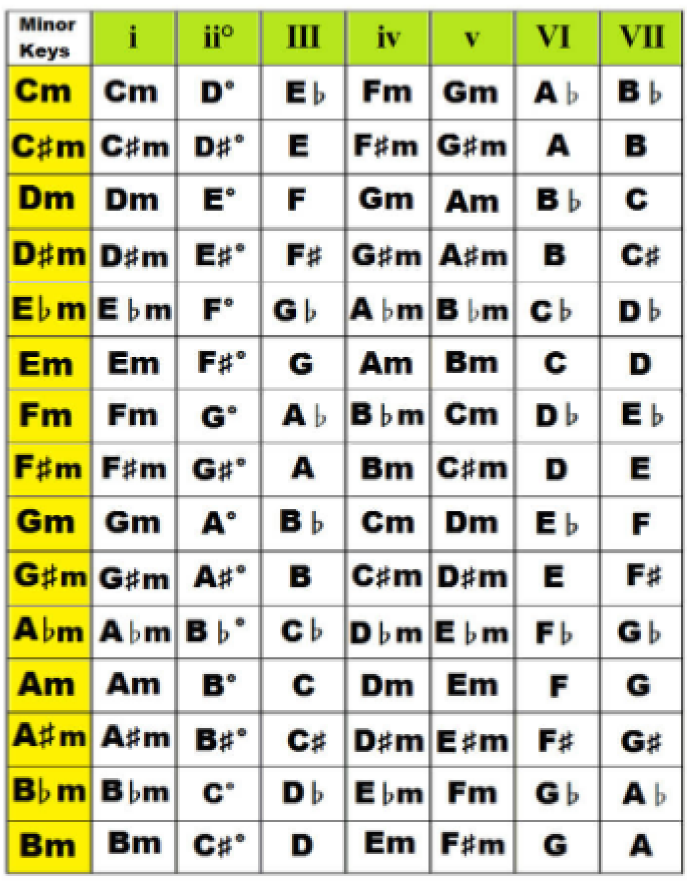
1. by methods decribed in the assignment:
   - find candidate tonic keys by fitting in 1 of gammas
   - use considering first and last note of the melody metric for determining best tonic key
   - use considering gamma notes repetitions metric for all candidate keys for determining best tonic key
   - based on this metrics find the tonic key of the melody
2. find tonic key of the melody using python library music21 (easy way)
   In my solution I used both of this ways and after that comparing their results.

And then we will get tonic key of the melody we can construct step chords. We chose chord according to table which will fit for our chosen tonic key, according to the table given in the assignment in further we will generate our accompaniment using only these step chords , because these chords will fit best for our tonic key of the given melody.



Major Keys	I	ii	iii	IV	V	vi	vii°
C	C	Dm	Em	F	G	Am	B°
C♯	C♯	D♯m	E♯m	F♯	G♯	A♯m	B♯°
D♭	D♭	E♭m	Fm	G♭	A♭	B♭m	C°
D	D	Em	F♯m	G	A	Bm	C♯°
E♭	E♭	Fm	Gm	A♭	B♭	Cm	D°
E	E	F♯m	G♯m	A	B	C♯m	D♯°
F	F	Gm	Am	B♭	C	Dm	E°
F♯	F♯	G♯m	A♯m	B	C♯	D♯m	E♯°
G♭	G♭	A♭m	B♭m	C♭	D♭	E♭m	F°
G	G	Am	Bm	C	D	Em	F♯°
A♭	A♭	B♭m	Cm	D♭	E♭	Fm	G°
A	A	Bm	C♯m	D	E	F♯m	G♯°
B♭	B♭	Cm	Dm	E♭	F	Gm	A°
B	B	C♯m	D♯m	E	F♯	G♯m	A♯°

Figure 6. Major keys



Minor Keys	i	ii°	III	iv	v	VI	VII
Cm	Cm	D°	E♭	Fm	Gm	A♭	B♭
C♯m	C♯m	D♯°	E	F♯m	G♯m	A	B
Dm	Dm	E°	F	Gm	Am	B♭	C
D♯m	D♯m	E♯°	F♯	G♯m	A♯m	B	C♯
E♭m	E♭m	F°	G♭	A♭m	B♭m	C♭	D♭
Em	Em	F♯°	G	Am	Bm	C	D
Fm	Fm	G°	A♭	B♭m	Cm	D♭	E♭
F♯m	F♯m	G♯°	A	Bm	C♯m	D	E
Gm	Gm	A°	B♭	Cm	Dm	E♭	F
G♯m	G♯m	A♯°	B	C♯m	D♯m	E	F♯
A♭m	A♭m	B♭°	C♭	D♭m	E♭m	F♭	G♭
Am	Am	B°	C	Dm	Em	F	G
A♯m	A♯m	B♯°	C♯	D♯m	E♯m	F♯	G♯
B♭m	B♭m	C°	D♭	E♭m	Fm	G♭	A♭
Bm	Bm	C♯°	D	Em	F♯m	G	A

By this table we get first note of the chord, in step chord list. And by the shown below triads we can construct the chord itself.

```
major_triad = [0, 4, 7]
minor_triad = [0, 3, 7]
dim_triad = [0, 3, 6]
```

And only these seven chords, located in the desired row will be used in the resulted melody.

Evolutionary algorithm description:

My evolutionary algorithm consists of several functions to run the evolution of the genomes and find the best genome for the current task. In my case genome - is just a list of chords, that represent accompaniment for the given melody. Population, in this case, is a list of genomes or, in other words, a 2-dimensional list of chords. Further, I describe all the functions I used in my evolutionary algorithm and how they works.

Generate genome:

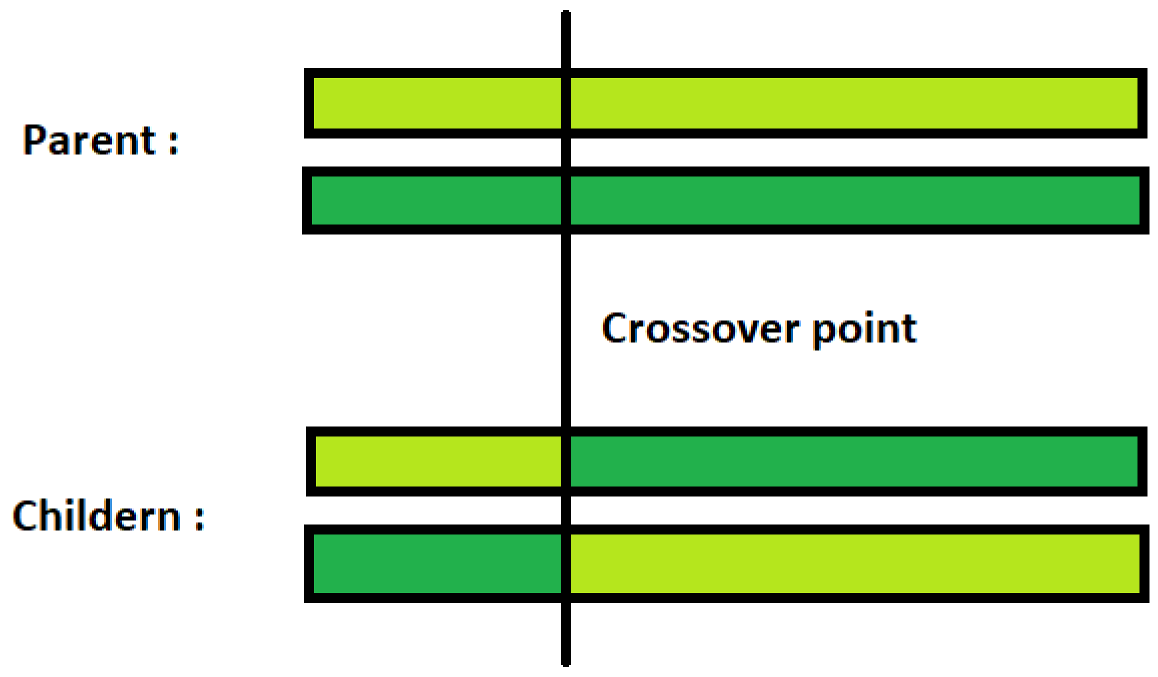
This is function used to generate genome. In this function, we just chose k random chords from the step chord list.

Generate population:

Function to generate population. We generate k genomes, and they are constructing population.

Crossover:

Function to crossover between 2 chosen genomes. In it we randomly generate a number in the range of length of the genome - p, and then we form 2 new genomes which are a combination of initial genomes. The part from 0 to p of the first genome with the remaining part of the second, and part from 0 to p of the second genome with the remaining part of first. This is one point crossover function and it's result can be represented as shown in the picture below

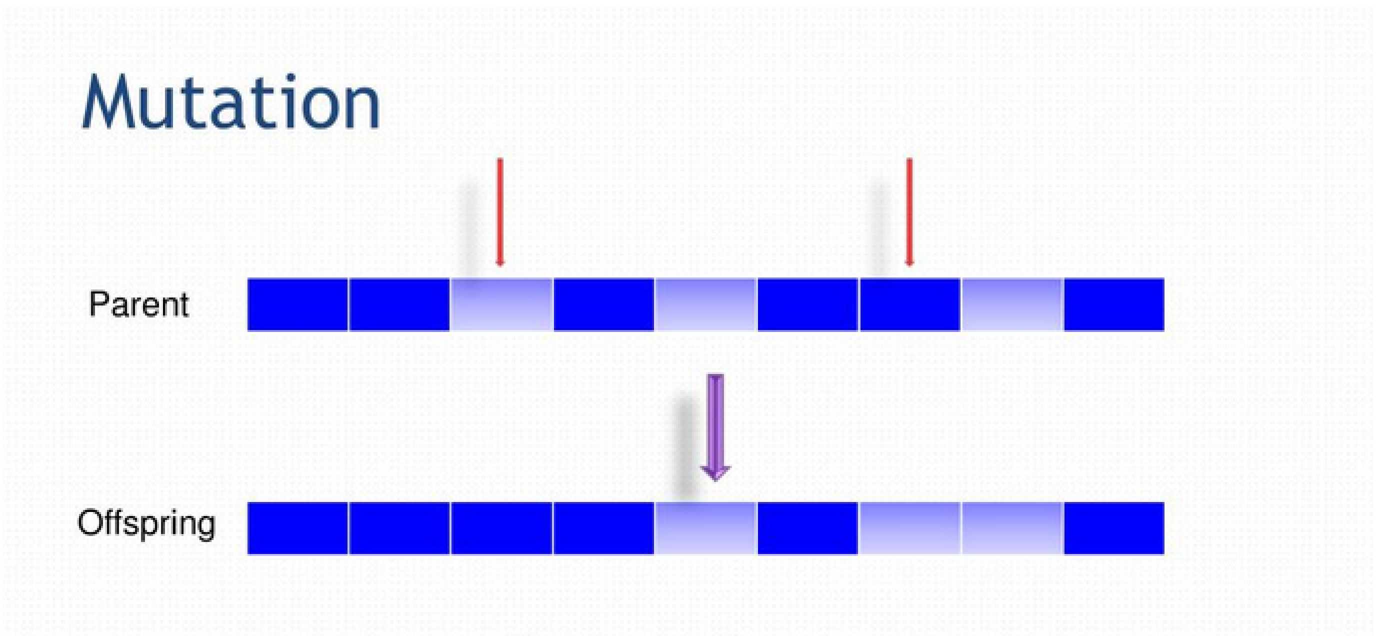


Mutation:

mutation function. In it we chose the number of mutations to perform, and then generate 2 random variables

1. position of the element in the genome

2. probability to perform mutation
   If the second variable is greater than the given mutation probability we replace the genome chosen by the random variable with some random from step chord.
   In the image below shown graphical representattion of the mutation, if we have 2 possible values in the genome(blue or white). And in our case we just chose randomly 1 of 7 possible new chords



   Default values:

- for number of mutations: 10
- for probability of mutation: 0.5

Fitness function:

In the fitness function, I'm considering the patterns of chords sequences which sounds good with our tonic key.
II-S-VI-T-III-D-VII

1. D-T
2. S-T
3. S-D-T
4. bad: D - S
5. It's bad to move into dominant decreasing side
   and according to these pattern I'm calculating value of the fitness function

- first and seconds patterns: fitness += 40
- third pattern: fitness += 35
- fourth pattern: fitness -= 70
- fifth pattern: fitness -= 20
  Also, I decided to consider that chord repetition is bad, but not that bad as the incorrect patterns above, so in this case, I'm substituting 10 from the fitness function value(fitesss -= 10)

Selection

in this function, we chose 2 genomes of parents for further crossover, and the probability of being selected as a parent depends on its fitness score.

Run evolution:

in this function, we run our evolution. We firstly declare the number of generations to choose the best genome. Then we generate a population of genomes and times equal generations number we generate next-generation according to the following algorithm:

- sort our population by fitness function result of the genomes
- take 2 best genomes
- and population size/2 do next:
- select 2 parents
- crossover them

- after this mutate them
- and append them to the next generation
  and after this algorithm, we will get the next population with the same number of genomes. This new population should be better than the previous in terms of the fitness function of genomes.

Default values for this function:

- number of generations: 100
- number of genomes in 1 population: 100

Result:

Using all the functions described above, we will get the best population. After this, we can choose the genome with the best fitness value from the final population, and this genome will be our accompaniment for the given melody.