Interactive Evolution for Melody Generation

Noemi Canovi, Roberto Mazzaro
Michele Presutto, Olga Zaghen

INTRODUCTION

Interactive Evolution for Melody Generation

Aim:

Generating melodic and rhythmic lines

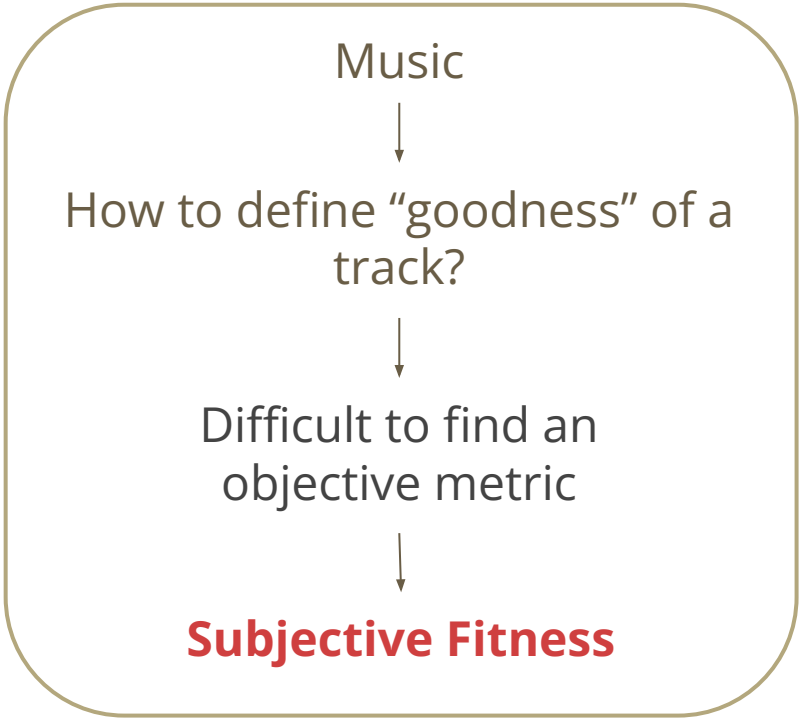
How:

Using evolutionary algorithms and interactive fitness

Introduction

GA

- A Genetic Algorithm (GA) evolves the population of tracks
- The fitness leads the evolution process



Introduction

Subjective Fitness

PRO:

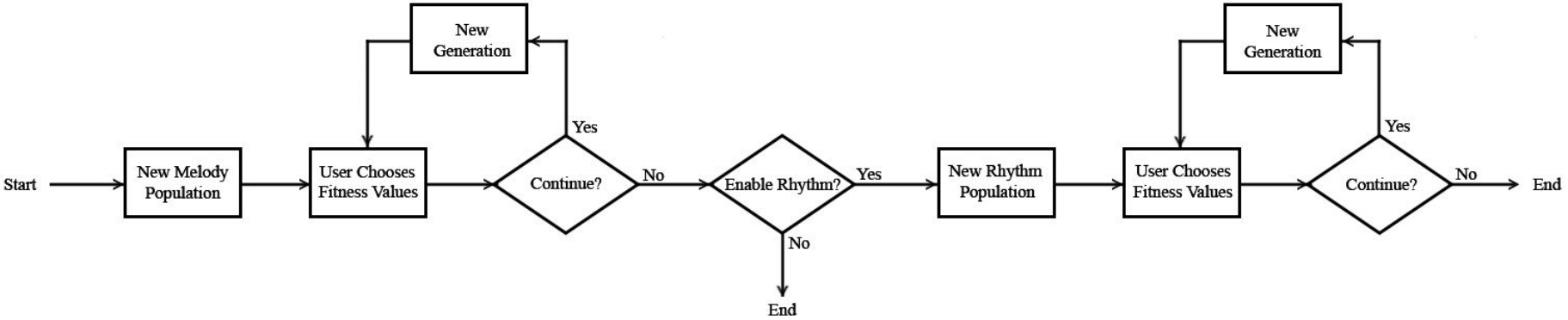
- no complex metric
- the algorithm will suit the user taste

CON:

- user's fatigue

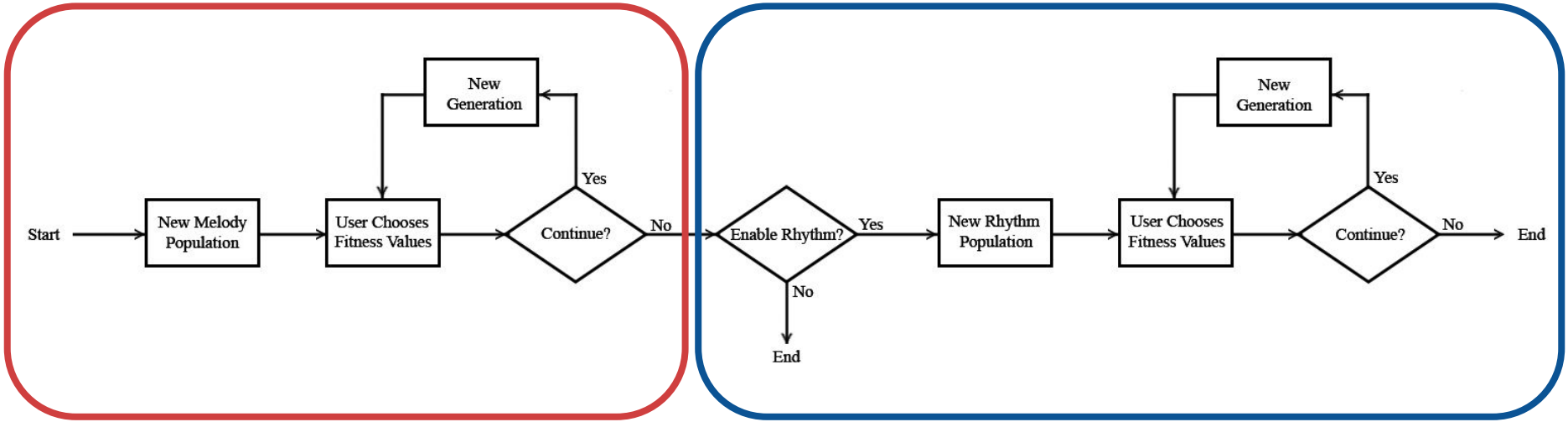
Example of fitnesses chosen by the user

Introduction
Main Steps



Introduction

Main Steps



Melody Evolution

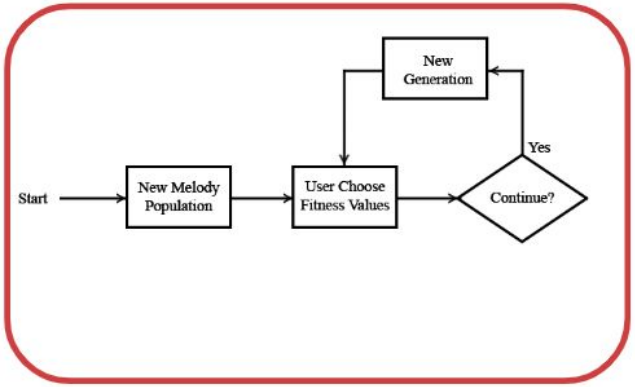
Rhythm Evolution

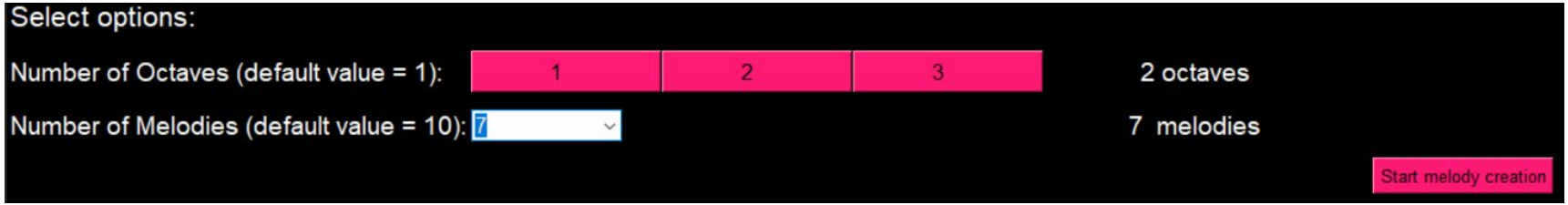
Introduction

Melody Evolution: Initial Frame

First interface where the user can choose:

- number of octaves
- population size





Example of values chosen by the user

Introduction

Melody Evolution: Generation

The **melodies** are

- **created**
- **converted** to listenable files (.wav)
- **presented** to the user

Introduction

Melody Evolution: Evaluation

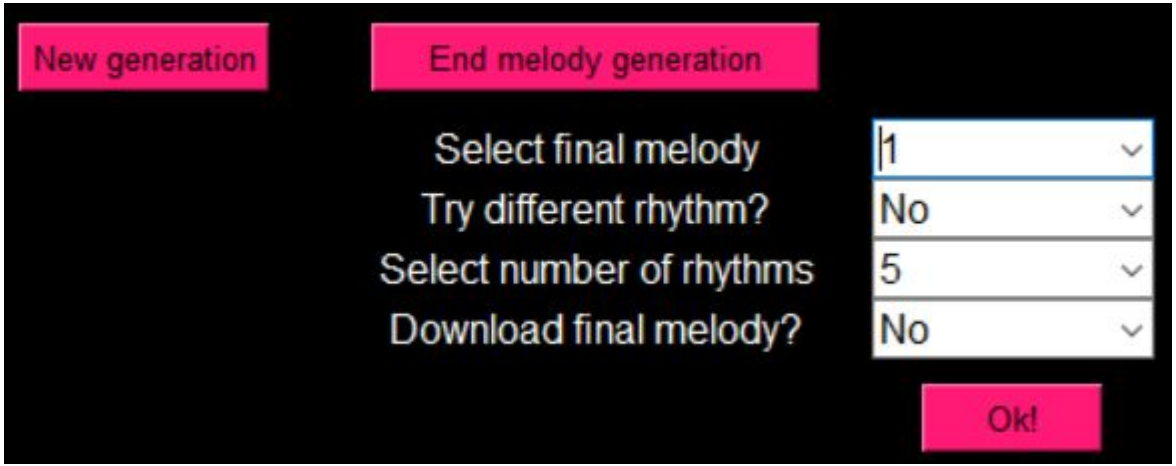
The user **listen** and **rates** the tracks

Introduction

Melody Evolution: Options

Once **satisfied** with the melody, the user can choose:

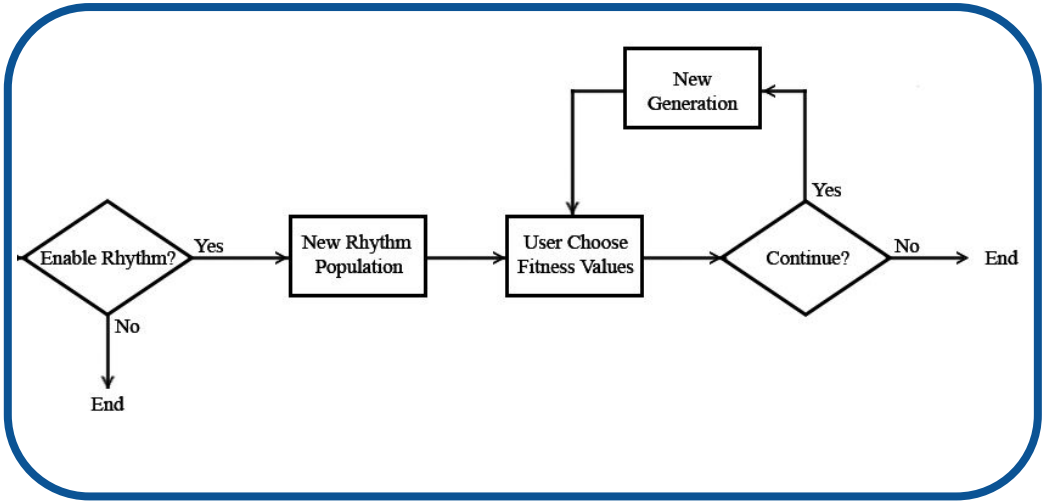
- final melody
- to evolve rhythm
- the number of rhythm to evolve at each generation
- to download the final melody



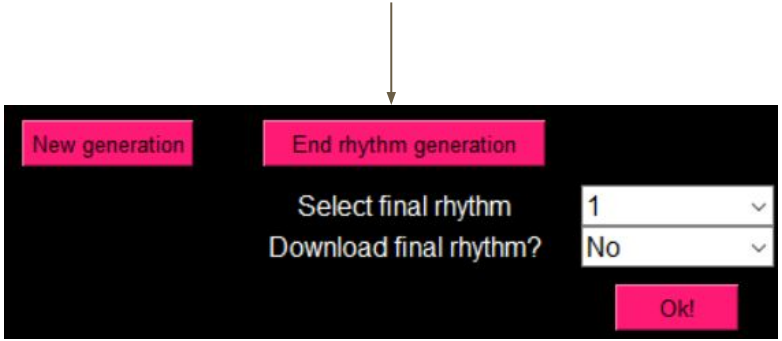
Introduction

Rhythm Evolution

The **same steps** are applied here for the **rhythm evolution!**



The only difference is the **options** that pop up when the user is satisfied with the track



ENCODING

A main **issue** in GA is how to **encode** problem specific **informations into genes**

- Individual encoding
- Note encoding - Pitch
- Note encoding - Rhythm
- Obtain Phenotype

Individual encoding

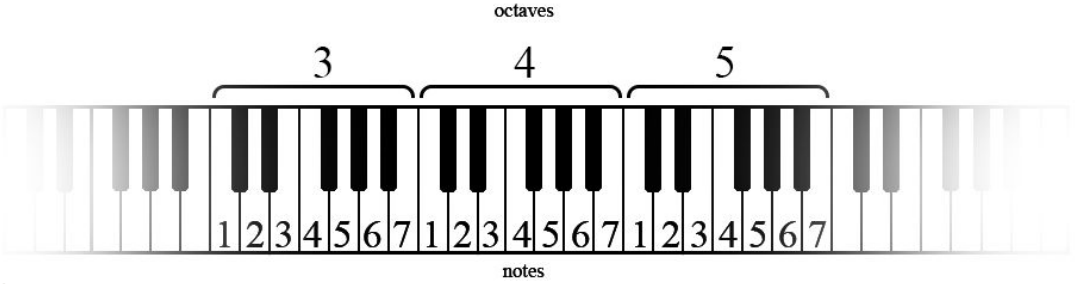
- Individual is a **list of notes**
- Length: **8**
- Implemented in the class **Notelist**

Individual 1							
Measure 1				Measure 2			
V = 6 O = 4 R = 2	V = 1 O = 4 R = 2	V = 4 O = 4 R = 4	V = 3 O = 4 R = 3	V = 2 O = 4 R = 2	V = 4 O = 4 R = 2	V = 1 O = 4 R = 3	V = 6 O = 4 R = 2

Example of individual

Note encoding - Pitch

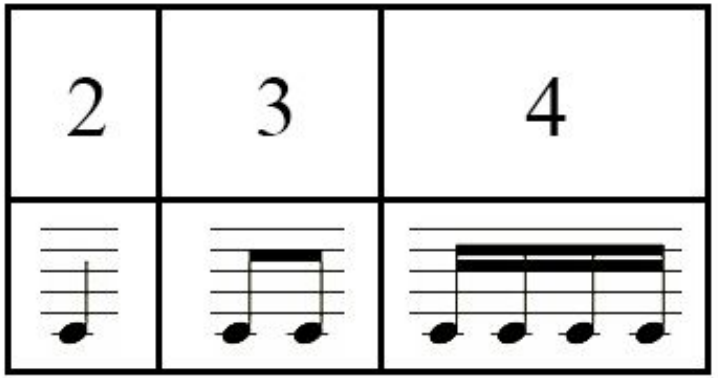
- Note described from **value**, **octave** and **rhythm**
- Implemented in the class **Note**



- Work with max. **3 octaves**
- **No alterations** to the note value
- Octave value from a **Gaussian centered in 4**
- Note value belongs to {1, ..., 7} generated with **Zipf's law**

Note encoding - Rhythm

- Rhythm is **simplified** as a **repetition** of the **same note** with **different durations**
- Available note durations: 1/4, 1/8, 1/16
- Encoded as the **power of ½**

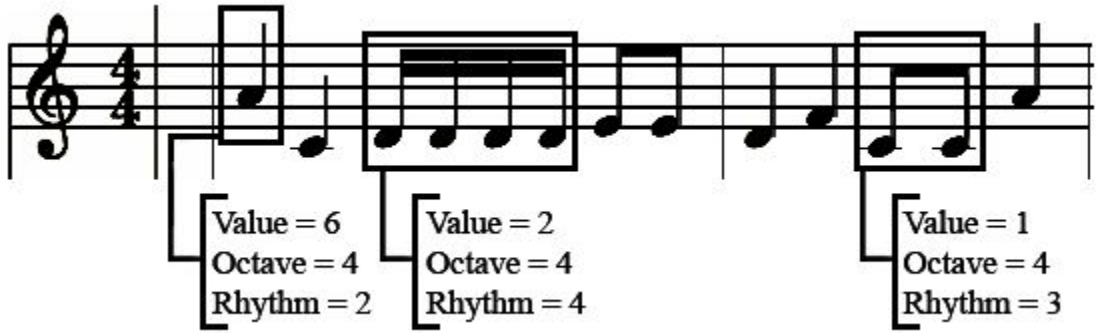


- The value is always 2 in the melody evolution
- In the rhythm evolution generated from right side of a gaussian centered in 2

Individual and Note encoding

An individual has this form:

Individual 1							
Measure 1				Measure 2			
V = 6 O = 4 R = 2	V = 1 O = 4 R = 2	V = 4 O = 4 R = 4	V = 3 O = 4 R = 3	V = 2 O = 4 R = 2	V = 4 O = 4 R = 2	V = 1 O = 4 R = 3	V = 6 O = 4 R = 2



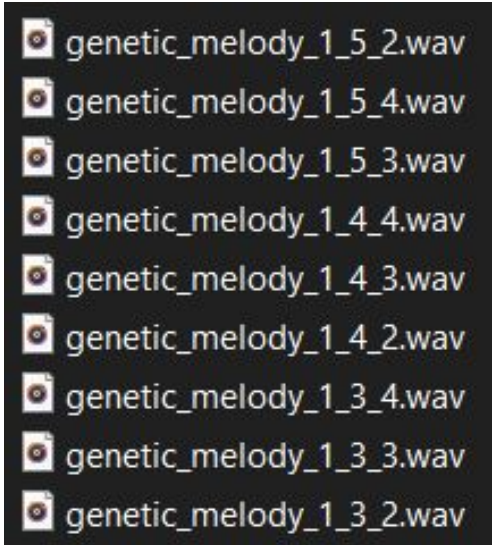
From Genotype to Phenotype

Phenotype:

- User have to **listen** to the generated individual
- Use Wave to **append** Note **sounds**
- Saved as indiv_n

Dataset:

- **63 audio** files (7x3x3)
- Named as genetic_melody_V_O_R
- Sampled from an electric piano
- Beat 104 bpm



All different files for C note

GENETIC ALGORITHM

1. Subjectivity of the fitness

2. GA: algorithmic choices
- Population and generations
- Parent selection
- Crossover
- Mutation

Subjective fitness

Centrality of the **user interaction**
in the evolutionary computation

Subjective fitness

subject-dependent

Fitness of a melody

=

rate (**from 1 to 10**) of the user

Centrality of the **user interaction** in the evolutionary computation

Subjective fitness

subject-dependent

Fitness of a melody

=

rate (**from 1 to 10**) of the user

Centrality of the **user interaction** in the evolutionary computation

Influence of **human fatigue** on the parameters of the GA

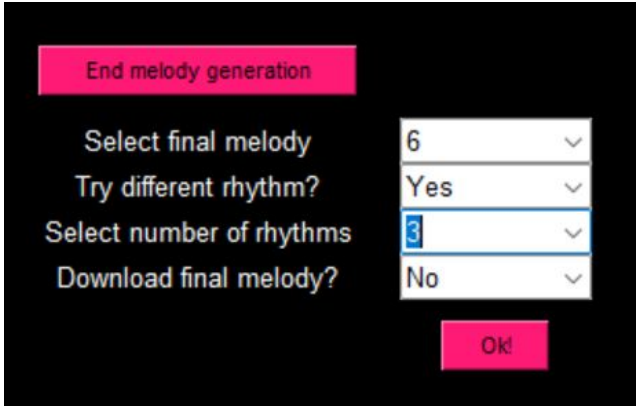
GA: algorithmic choices

Population and generations

- The **number of generations** and the **population size** are lower than classic GA settings (**human fatigue**)

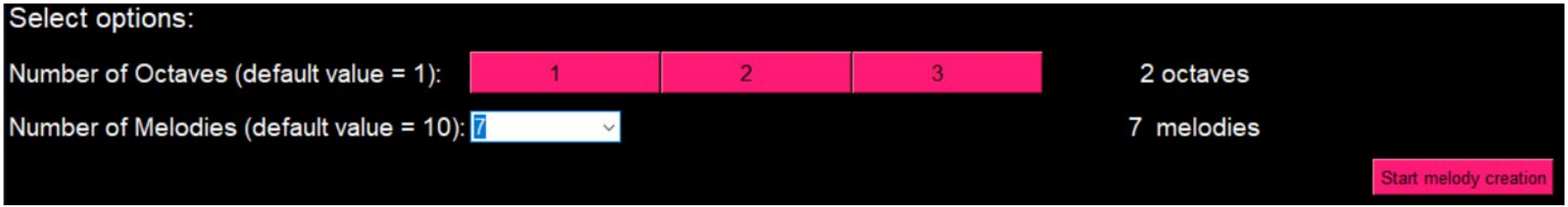
GA: algorithmic choices

Population and generations



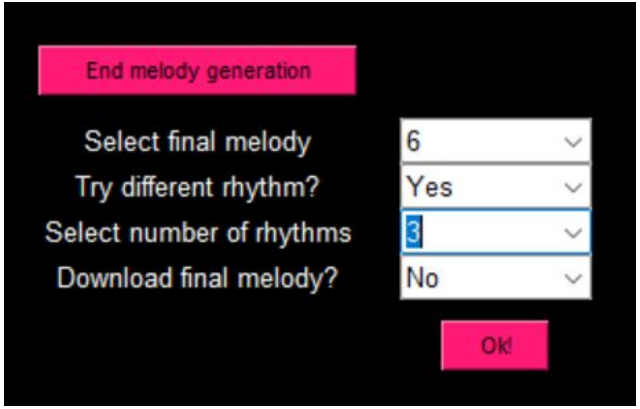
- The **number of generations** and the **population size** are lower than classic GA settings (**human fatigue**)

- The **melody** population size can be set to a value from 1 to 10
- The **rhythm** population size can be set to a value from 1 to 5



GA: algorithmic choices

Population and generations

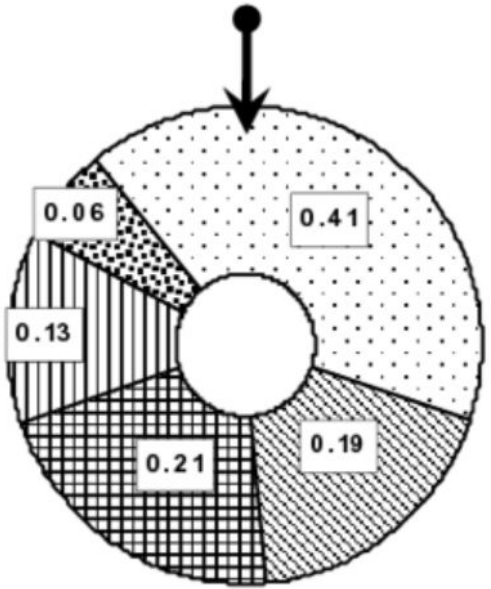


- The **number of generations** and the **population size** are lower than classic GA settings (**human fatigue**)

- The number of generations depends on when the **user** ends the algorithm

GA: algorithmic choices

Parent selection



Roulette wheel prediction formula

Fitness-proportionate parent selection

$$p_i = \frac{f_i}{\Sigma_{j=1}^{N} f_j}$$

Promote **exploitation**

GA: algorithmic choices

Survivor selection

- ★ **Replace** the old population by offspring
- ★ Maintain some **elite** individuals

GA: algorithmic choices

Survivor selection

★ **Replace** the old population by offspring

★ Maintain some **elite** individuals

Their quantity depends on the population size

GA: algorithmic choices

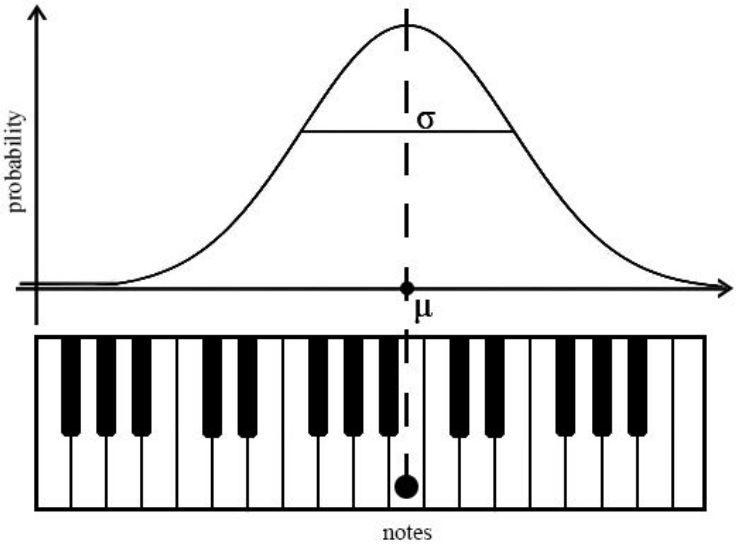
Mutation

Melody:

- **Gaussian** centered in the actual **note position**
- Generate a new different position
- Recover new note value and octave

Rhythm:

- New random value from the **half** Gaussian distribution centered in **2**



Representation of melody's mutation

GA: algorithmic choices

Crossover

Melody:

- One point crossover
- Between 2 Notelist

Rhythm:

- One point crossover
- Only rhythm values

Individual 1							
Measure 1				Measure 2			
V = 6 O = 4 R = 2	V = 1 O = 4 R = 2	V = 4 O = 4 R = 2	V = 3 O = 4 R = 2	V = 2 O = 4 R = 2	V = 4 O = 4 R = 2	V = 1 O = 4 R = 2	V = 6 O = 4 R = 2

Individual 2							
Measure 1				Measure 2			
V = 1 O = 4 R = 2	V = 2 O = 4 R = 2	V = 3 O = 4 R = 2	V = 1 O = 4 R = 2	V = 1 O = 4 R = 2	V = 2 O = 4 R = 2	V = 3 O = 4 R = 2	V = 1 O = 4 R = 2

Generated Individual							
Measure 1				Measure 2			
V = 1 O = 4 R = 2	V = 2 O = 4 R = 2	V = 3 O = 4 R = 2	V = 1 O = 4 R = 2	V = 1 O = 4 R = 2	V = 4 O = 4 R = 2	V = 1 O = 4 R = 2	V = 6 O = 4 R = 2

Melody Crossover

Generated Individual							
Measure 1				Measure 2			
V = 6 O = 4 R = 2	V = 1 O = 4 R = 2	V = 4 O = 4 R = 4	V = 3 O = 4 R = 2	V = 2 O = 4 R = 3	V = 4 O = 4 R = 2	V = 1 O = 4 R = 4	V = 6 O = 4 R = 2

Rhythm Crossover

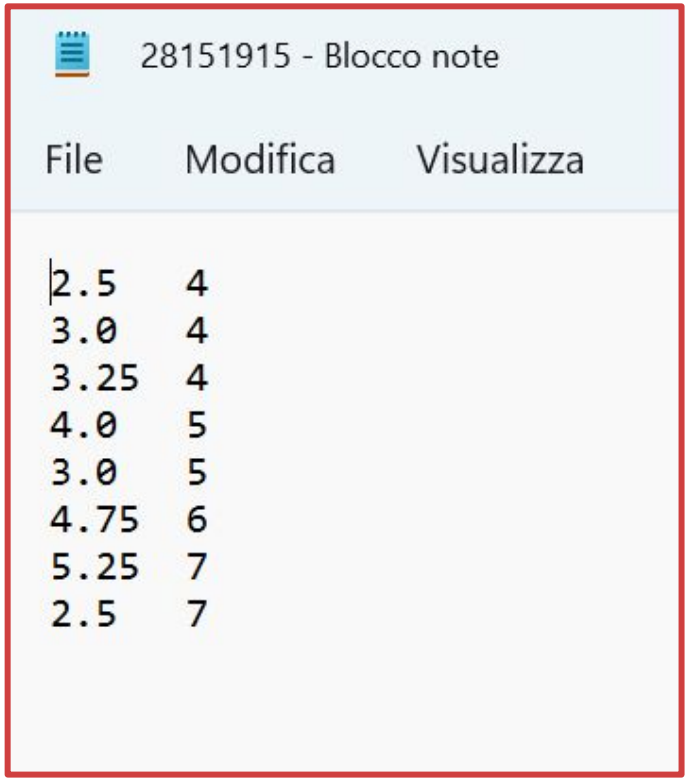
STATISTICS

We carried out an analysis to **evaluate** the effectiveness of our system

Statistics

Methodology

- We stored the **average** score and **maximum** score in each generation, for **40** different executions

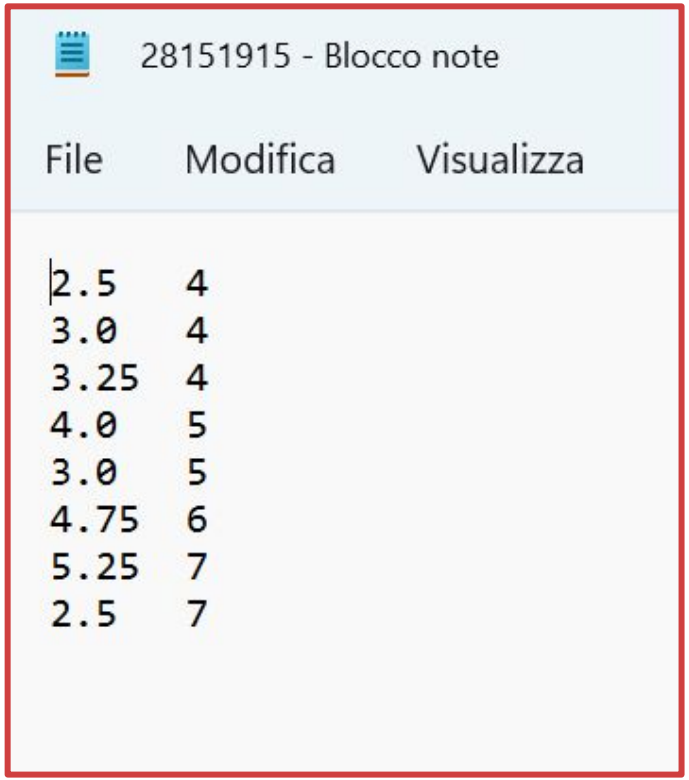


Statistics

Methodology

- We stored the **average** score and **maximum** score in each generation, for **40** different executions

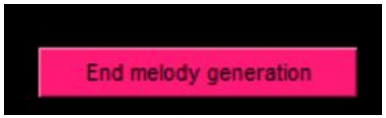
- We performed:
  1. Statistics on **number of generations**
  2. Analysis of the **fitness trend** by averaging results over all executions



Statistics

Number of generations

The average number of iterations in evolution resulted to be



4.96
for the melody

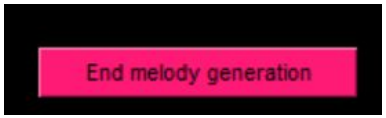
1.67
for the rhythm

total 10-15 mins of runtime

Statistics

Number of generations

The average number of iterations in evolution resulted to be



4.96
for the melody

1.67
for the rhythm

total 10-15 mins of runtime

not subject to proper evolution

Statistics

Fitness trend for melody evolution

	absolute fitness value		fitness gain
	first gen	last gen	last - first
average score	3.75	5.10	1.46
max score	5.67	7.30	1.76

Statistics

Fitness trend for melody evolution

	absolute fitness value		fitness gain
	first gen	last gen	last - first
average score	3.75	5.10	1.46
max score	5.67	7.30	1.76

Tendency to a substantial **positive gain**

Statistics

Fitness trend for melody evolution

	absolute fitness value		fitness gain
	first gen	last gen	last - first
average score	3.75	5.10	1.46
max score	5.67	7.30	1.76

Tendency to a substantial **positive gain**

The algorithm well adapts to the **subjective taste** of the human user in the melody evolution!

DIFFICULTIES & IMPROVEMENTS

Difficulties

Issues we encountered and how we resolved them

Improvements

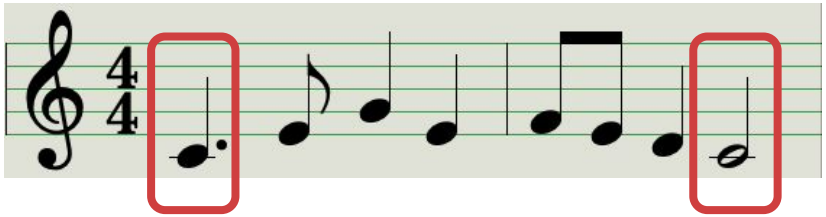
Possible future works

Difficulties

- **Relative** fitness
- **Fitness bottleneck**
- Implementation of the **rhythm**
- **Hyperparameters** difficult to set
- High influence of the **first generation**

Improvements

- Assumptions **relaxation** for note and rhythm encoding
- Adding more **constraints**
- Defining an **objective fitness function**

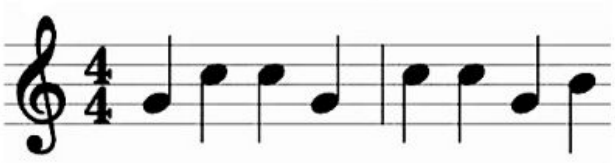


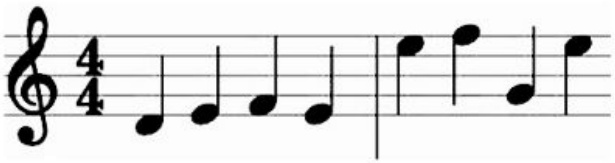
Melody that cannot be generated using the actual encoding

CONCLUSIONS

- ★ Nice melody can be generated:
  - with few constraints
  - in a few number of generation
- ★ We collected the most interesting results

Interesting creations





Example of two melodies produced by the GA.

They have been generated in the same evolutionary process and they are **objectively musically coherent**, in addition to complementing each other well!

THANKS FOR YOUR ATTENTION!

Difficulties

subjectivity and relative fitness: the fitness is different from person to person, and is biased by the overall score of the other melody in the context.

fitness bottleneck: due to the active participation of the user in the listening of each individual and the setting of a score to each melody, it is likely to get tired after some iterations.

influence of the first generation: if the first generation is good than the number of generation to obtain a satisfactory score will be low. We tried to solve this issue by using zipf's law.

Difficulties

implementation of the rhythm: implementing the rhythm structure maintaining a fixed length and duration of the melodies.

Hyperparameters difficult to sets: finding the best setting for the hyperparameters of the GA without burdening the user.

High influence of the first generation: the convergence of the GA is faster with at least a good individual in the initial populations while it stagnate without

IMPROVEMENTS

Possible improvements could be:

- Assumption relaxation for note and rhythm encoding
- Adding more constraints
- Defining an objective fitness function

CONCLUSION

we can say that the interactive evolution computation system that we implemented is capable of producing nice melodies in a restricted number of generations

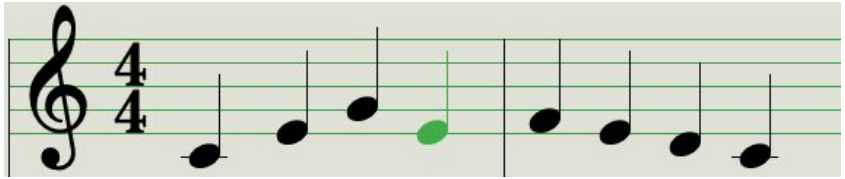
Statistics

Improvements

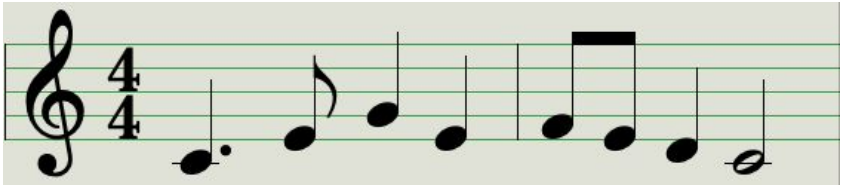
Assumption and relaxation for rhythm encoding:

the actual encoding of the rhythm consist of splitting the same note by maintaining its total duration.

maybe a better encoding could be changing just the duration of the note and doing a control on the entire melody length

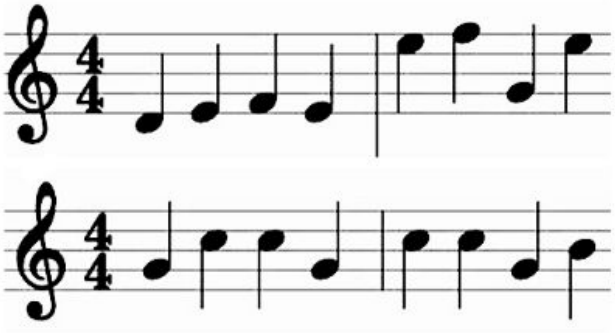


melody that can generate
using actual encoding



melody that can not generate
using actual encoding

Interesting creations



Example of two melodies produced by the GA.

They have been generated in the same evolutionary process and they are **objectively musically coherent**, in addition to complementing each other well!

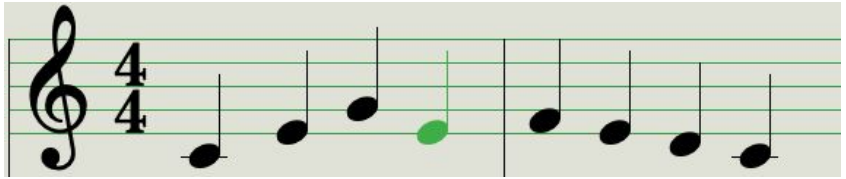
Improvements

Rhythm encoding

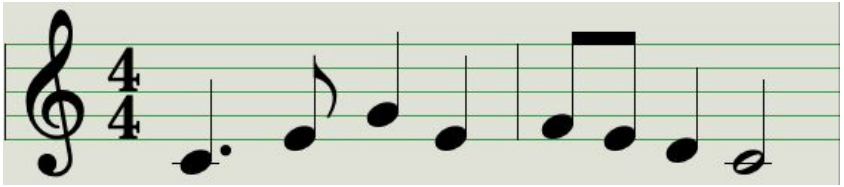
repetition of the same note maintaining its total duration

↓

changing just the duration of the note and doing a control on the entire melody length



melody that can generate using actual encoding



melody that can not generate using actual encoding

Subjective fitness

subject-dependent

Fitness of a melody

=

rate (**from 1 to 10**) of the user

Centrality of the **user interaction** in the evolutionary computation

Influence of **human fatigue** on the parameters of the GA

Subjective fitness

Fitness of a melody

=

rate (**from 1 to 10**) of the user

Centrality of the **user interaction** in the evolutionary computation

Statistics

Results : fitness trend

	absolute fitness value		fitness gain	
	first gen	last gen	raw = last - first	mean = $\frac{\text{last - first}}{\text{\# generations}}$
average score	3.75	5.10	1.46	0.35
max score	5.67	7.30	1.76	0.42

GA: algorithmic choices

Population and generations

- The **number of generations** and the **population size** are lower than classic GA settings (**human fatigue**)
- The **melody** population size can be set to a value from 1 to 10, **a priori**
- The **rhythm** population size can be set to a value from 1 to 5, **a priori**
- The number of generations depends on when the **user** ends the algorithm, **a posteriori**