BEAT GENERATION

Rhythmic Pattern generation using Genetic Algorithms

Advanced Coding Tools and Methodologies
Academic Year 2020/2021

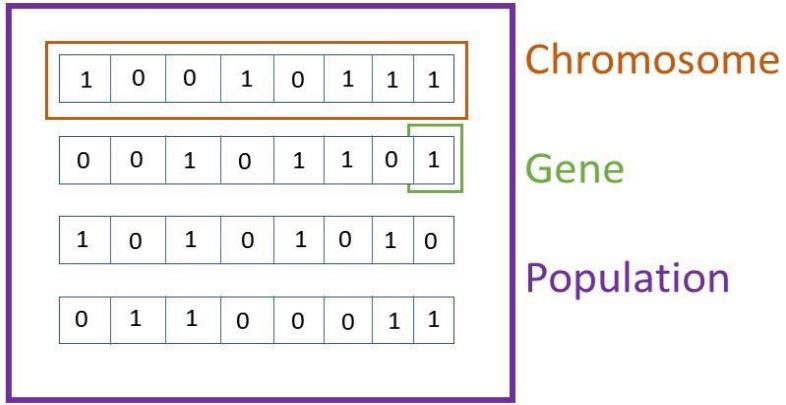
Francesco Boarino	10788381
Andrea Coppola	10749143
Alessandro Molteni	10497475

Our Goal

Analysis of Drum Patterns

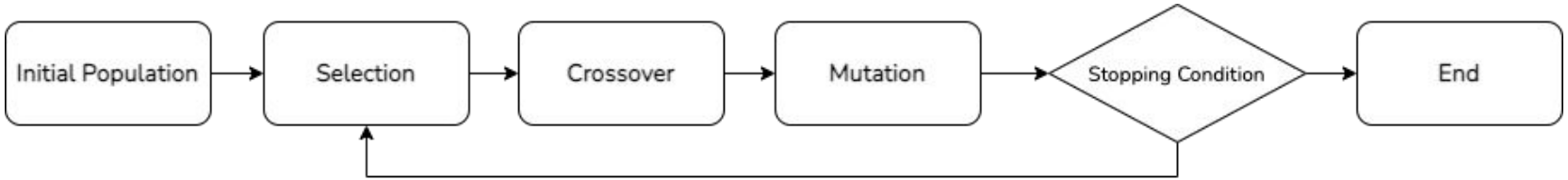
Bit String Representation

Drum Pattern generation using GAs



But what is a Genetic Algorithm?

Optimization technique inspired by Darwin's theory of evolution

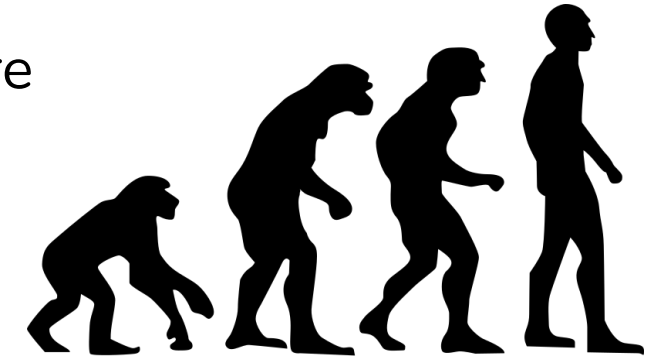


Fitness

Ability of an individual to compete with others

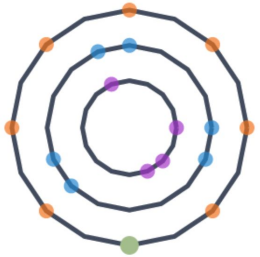
Fitness score assigned to each element

Reproduction chances based on the score

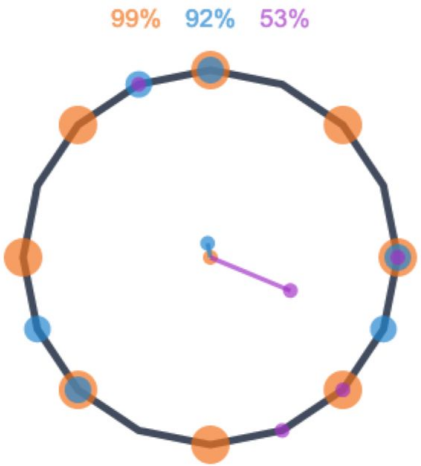


Fitness

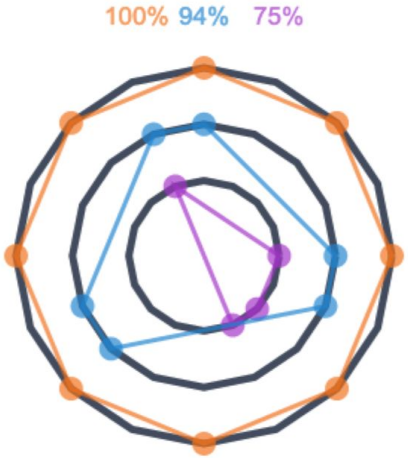
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	1	0	0	0	0	1	1	0	0	0	1	1	0	0	0
1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0



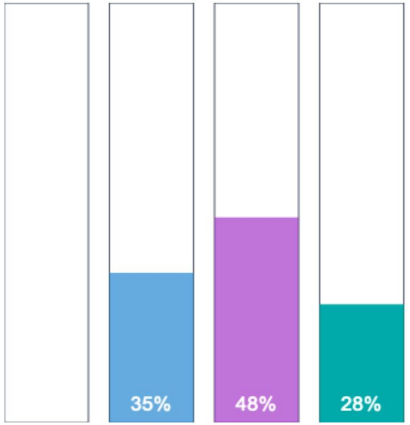
Balance



Evenness



Entropy



Selection

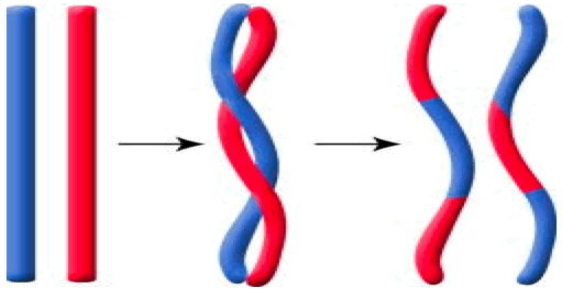
Fittest individuals selected to breed a new generation

Fittest Survive: Only the fittest individuals can survive

Roulette Wheel: Each individual has a probability to survive

Crossover

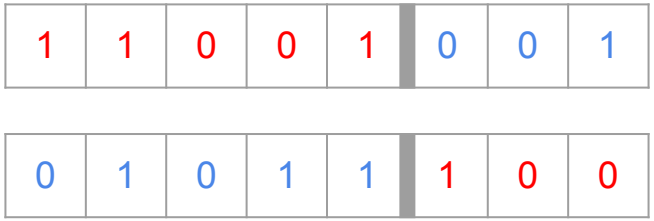
Couples of elements are combined to generate modified versions of themselves



Single Point Crossover

Parents

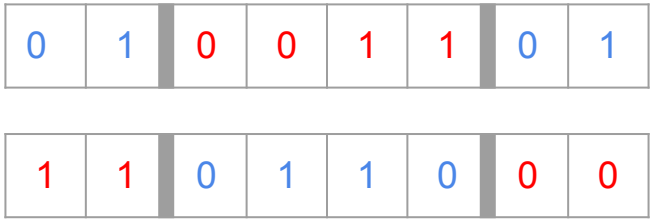
Offspring



Two Point Crossover

Parents

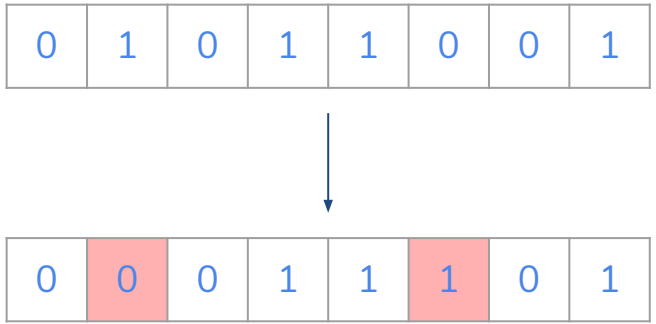
Offspring



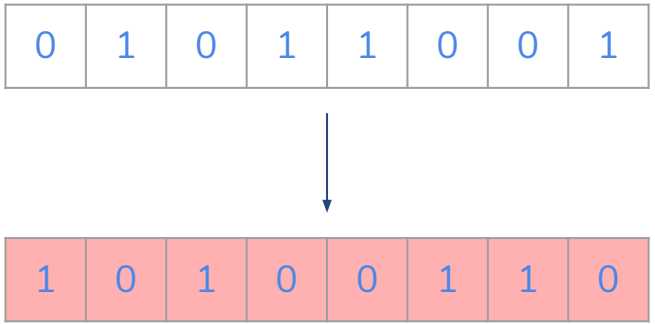
Mutation

Maintain diversity within the population

Bit String



Flip Bit

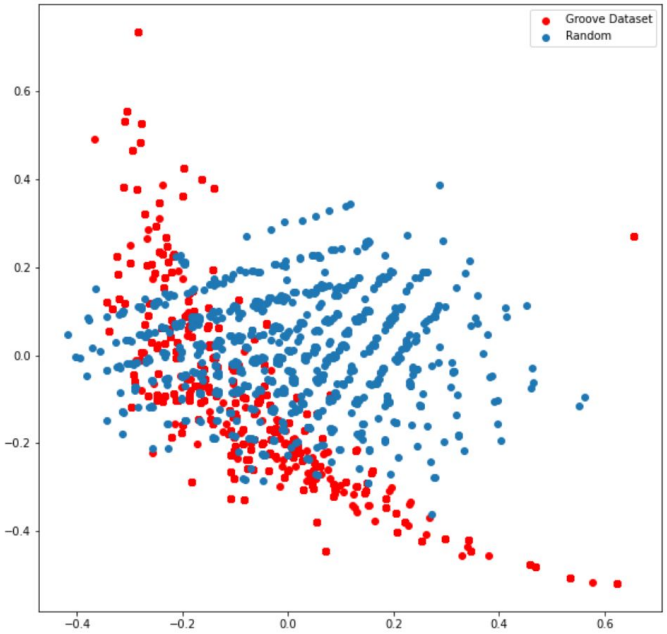


Analysis and Design

Research on the topic

Prototypes in Jupyter to test feasibility

Feature exploration



Built with extensibility in mind

Strategy Design Pattern

Dynamic *Genetic Algorithm* class

Pattern class

```
class TemplateStrategyManager {
    _strategies; // Strategies to be managed

    // Simply init the _strategies array
    constructor() {
        this._strategies = [
            new Strategy1(),
            new Strategy2()
        ];
    }

    getStrategy(name) {
        // Search and return the requested strategy
    }

    getStrategyNames() {
        // Returns list containing the name of all the strategies
    }
}
```

```
class TemplateStrategy {
    _name = "Template"

    compute(arguments) {
        // Apply the strategy over the arguments
    }
}
```

Tools

Semantic UI

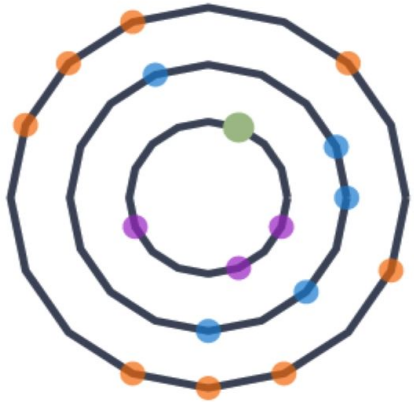
GUI

Hidden descriptions

Overlay Window containing the final generation

Pattern names taken random from 40.000+ names contained in a *json* file

4 Canvas for pattern representation



Player

Sequencer schedules BufferSource nodes with 3 different sequences running in parallel.

It is possible to mute, unmute each of the sequences, a metronome, and play stop the sequencer

Conclusions

Framework to generate and play with random rhythmic patterns

Take a deeper look in the features that describe each single sequence

Future Developments:
Adding more features and representations
Adding more strategies

Demo

[Github Page]