# Introduction to AI, Assignment 2

**RANGULOV DENIS BS2-4**

# PSO №1

## Used parameters

By default generated chords are minor but if you want major chords then you should set false this variable in PSO1.Swarm class:

```
boolean isMinor = true;
```

Inertia coefficient is calculated given that number of generations(iterations)

```
double inertia = (i * 0.05 < 0.4) ? 0.9 - i * 0.05 : 0.4;
```

'c1' - coefficient of an influence of the local best position to velocity changing:

```
double c1 = 1;
```

'c2' - coefficient of an influence of the global best position to velocity changing:

```
double c2 = 2;
```

'minValues' and 'maxValues' – limit's values of generated notes

```
int minValues = 0;
int maxValues = 127;
```

## Particle structure

One particle contains 3 objects of Coordinates class:

currentPos – current position of this particle

velocity – velocity of this particle

localBestPos - local best position of this particle

One coordinate object consist of 5 chords (objects of MyChord class). The MyChord object contains 3 notes (int values).

Summarize, every particle is 15 dimension (5 chords * 3 notes). Every note is randomly generated in range [minValues . . maxValues] in my case [0 . . 127]. After getting 5 normal chords (fitness function value is equal to 0) it's transforming in 16 chords (5,5,5,1). It will works for case if particle is 48 dimension (16 chords * 3 notes) but it will takes a much more times so I choose generating for 5 chords.

## Fitness function

I use 2 additional arrays 'values' and 'chords'. Every of that is in range [0 . . 11]. Indexes of them is associated with one of 12 notes. After fitness function working, 'values' array will contains fitness values of chord's starting note and 'chords' array will contains fitness values of note if it

would be starting note of tonic chord. Particle is considered normal if his fitness function value is equal to 0.

Algorithm of working of fitness function:

1) Move across all chords in particle and fill a cell associated with starting note of chord in 'values' array from output of an additional fitness function for one chord. The additional fitness function for chord takes into account differences between 0 and 1, 1 and 2 notes in a chord. In accordance with lad (major or minor).
2) Fill 'chords' array given that 'values' array (filling every cell given that associated note is tonic for whole particle) . If some chord's starting notes isn't tonic(this note), subdominant or dominant in some octave then instead of fitness function values for this chord add a constant value.
3) Return minimum value in 'chords' array.

## Used input values

Particle amount: 100000

Spent time: 13 seconds

Generations used to get the output: 24

# PSO №2

## Used parameters

```
double inertia = (i * 0.05 < 0.4) ? 0.9 - i * 0.05 : 0.4;
```

'c1' - coefficient of an influence of the local best position to velocity changing:

```
double c1 = 1;
```

'c2' - coefficient of an influence of the global best position to velocity changing:

```
double c2 = 2;
```

'minValues' and 'maxValues' – limit's values of generated notes

```
int minValues = 0;
int maxValues = 127;
```

## Particle structure

One particle contains 3 objects of Coordinates class:

currentPos – current position of this particle

velocity – velocity of this particle

localBestPos - local best position of this particle

One coordinate object consist of 32 notes (int).

Summarize, every particle is 32 dimension. Every note is randomly generated in range [minValues . . maxValues] in my case [0 . . 127].

## Fitness function

I use 2 additional arrays 'possibleValues' and 'perfectNotes'. Indexes of them is associated with one of 32 notes.

The 'possibleValues' array is filled for PSO2.Swarm's creating process. It contains boolean variables signing possible values of notes imposed on chord (even notes in 32).

The 'perfectNotes' array will contains coordinates of the nearest suitable position (32 suitable notes) for some random generated particle.

Algorithm of working of fitness function:

1) Fill even notes in 'perfectNotes' array in accordance to 'possibleValues' array.
2) Fill odd notes in 'perfectNotes' array in accordance to the previous and next even notes in 'perfectNotes' array.
3) Summarize difference between notes in 'perfectNotes' array and real generated in particle for whole 32 notes. It will be value of fitness function, when it will be equal to 0 then we get right particle(32 notes).

## Used input values

Particle amount: 100000

Spent time: 7 seconds

Generations used to get the output: 22