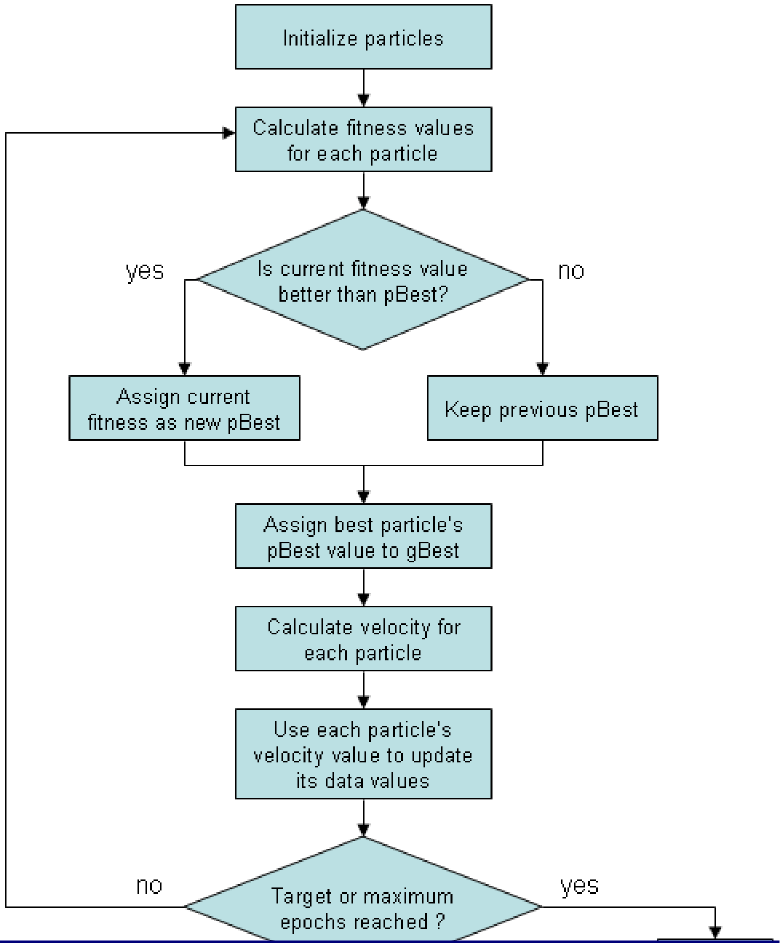
Introduction to Artificial Intellegent
Assignment #2
Report from Borgalinov Timur BS2-2

Task : Generate MIDI file which consists of
1) Any sequence of chords (tonic, dominant and subdominant triads) of the same duration (quarters) in fixed tonality (any of 24 possible major or minor tonalities) without modulations with fixed tempo (120 BPM) and fixed time signature (4/4) using PSO №1
2) Melody for this chord sequence in the same tonality with the same tempo and time signature using PSO №2. There should be no chords inside melody, only sequence of single notes with fixed duration (eights) Number of bars should be equal to 4. So, output file should contain sequence of 16 chords (4 chords per each bar) and 32 single notes (8 notes for each bar) which should be played simultaneously. Modulation is change of tonality. Mention that space between 7 scale degrees for major and minor tonalities differ (check lecture). Your fitness function should find aesthetically pleasing combination of chords and melody. For correct melody generation, each single note played simultaneously with the chord should have a value which is by modulo 12 equal to one of chord's notes. So, for tonic triad in C major (60, 64, 67) single note's values should be 60+12*N or 64+12*N or 67+12*N. N is an octave difference between chords and melody. Chords represent accompaniment for melody. In this assignment consider that accompaniment should be lower than melody on 1 or more octaves. Consider that dissonant intervals are not aesthetically pleasing. Each interval has a different measure of aesthetic enjoyment. There should be no continuous repetitions of the same chords (maximum 4). Possible midi note values are in the range [48,96]. Better to use lower values for chords. Difference between neighbor notes should not exceed 12. Difference

between lowest notes of neighbor chords also should not exceed 12. You should also provide a report with explanation of used parameters, representation of particle structures and fitness function explanation for both PSOs. Report should contain used input values for PSOs: particle amount, spent time, generations used to get the output.

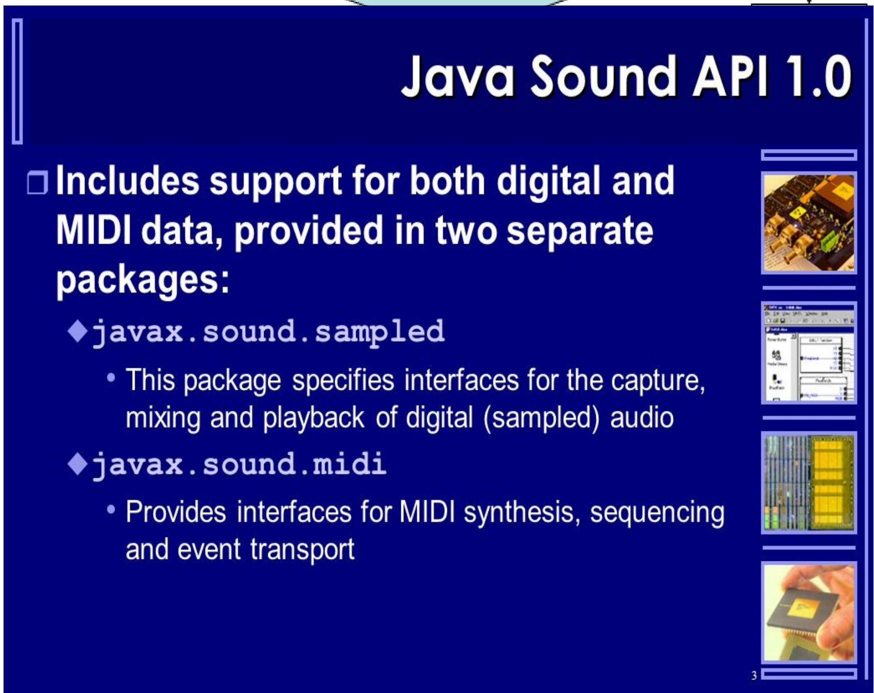
Solution:

To make the code more readable and structured i created model of music (Octave,Tone,Chord,Note classes),
then we needed two implementations of PSO algorithm.
First one for generating chords and another one for generating notes.


Initialize particles
Calculate fitness values for each particle
Is current fitness value better than pBest?
yes
no
Assign current fitness as new pBest
Keep previous pBest
Assign best particle's pBest value to gBest
Calculate velocity for each particle
Use each particle's velocity value to update its data values
Target or maximum epochs reached ?
no
yes

Music Library
In my code i used standard java library to work with MIDI.

```
import javax.sound.midi.*;
```



Java Sound API 1.0

- Includes support for both digital and MIDI data, provided in two separate packages:
  - `javax.sound.sampled`
    - This package specifies interfaces for the capture, mixing and playback of digital (sampled) audio
  - `javax.sound.midi`
    - Provides interfaces for MIDI synthesis, sequencing and event transport

Exactly it was Synthesizer to generate MIDI sounds, and then i save it in sequence by adding notes to Track to finally save it in a MIDI file;

PSO #1 (chords)

To make chords sound at least not bad we needed them to be tonic, subdominant and dominant and not to repeat more than 4 times in a row.

So for my first PSO algorithm the fitness function was to calculate that chords which didn't fit the restrictions and tried to make swarm improve itself.It has a variable target, which is responsible for level of reliability of an algorithm

Chords are in 48 dimensional space. Because totally we have 16 chords, 3 notes each

----

Number of particles : 1000

Number of max iterations : 10000

Maximum velocity of particle : 4

Average time : 3435 miliseconds

PSO #2 (notes)

In second PSO we needed to generate notes for chords in first PSO, so here we need to have smooth fitness function, i like when the music goes smoothly without big jumps in interval, so i decided to play my notes nearly of each other.

Logically my fitness function was to have average distance between notes as close as possible to one and also it increases when the note which is first for each accord in not that note on the higher octave

Notes are in 32 dimensional space.

Average time : 14462 milisecond

Maximum velocity of particle : 4

Number of particles   : 750
Number of max iterations : 12200

Ready track

Ready track generates into song.mid file in root directory of a project

Tonality

At the beginning of the program tonality generates randomly. It can be one of all 24 tonalities present in music.

Octaves

Notes are being played octave higher than chords

Classes include:

Chord – representing chord
ChordParticle – represents PSO for chords algorithms particle
MIDI – class representing features responsible for converting to MIDI and from MIDI
MidiWriterToFile – writes MIDI sequence to .mid file
MyChords – need to start creating my chords
Note – represents note
NotesParticle – represents PSO for notes algorithms particle
Octave – represents octave
Player – responsible for playing midi sounds (chords and notes)
PsoChords – PSO algorithm for chords
PsoNotes – PSO algorithm for notes
Tone – responsible for tonality