VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

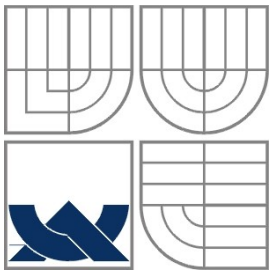
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

HUDEBNÍ IMPROVIZACE

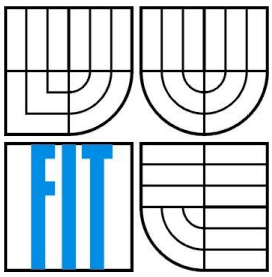
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MICHAEL ANGELOV
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HUDEBNÍ IMPROVIZACE

MUSICAL IMPROVISATION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — MICHAEL ANGELOV
AUTHOR

VEDOUCÍ PRÁCE — ING. MICHAL FAPŠO
SUPERVISOR

BRNO 2008

Abstrakt

Bakalářská práce sa zabývá problematikou algoritmické kompozice hudby, zejména oblastí počítačem řízené hudební improvizace. V úvodních kapitolách je uveden přehled existujících nástrojů a přístupů nejčastěji používaných v oblasti počítačové hudby. Následně je v práci uveden návrh nového systému pracujícího na principech markovských řetězců a prediktivních stromů spolu s popisem jeho implementace. Hlavní úlohou navrhované aplikace je analýza uživatelem předložené externí MIDI nahrávky a následná tvorba nového a inovativního hudebního materiálu ve formátu MIDI s tematikou blízkou originální nahrávce, čímž se vytváři dojem počítačové improvizace na dané hudební téma.

Abstract

The thesis deals with problems concerning algorithmic music compositon, especially the domain of musical improvisation. There is an opening presentation of some of existing tools and approaches that are commonly used in domain of computer music. Consenquently there is a proposal of a new system, using main principles of markov chains and prediction suffix trees (PST) with description of its implementation. The main task of developped application is to analyze an external MIDI recording that is proposed to the system by user and create a new and inovative musical material in MIDI format that would sound close to the original recording giving an impression of a computer improvised music to the listener.

Klíčová slova

MIDI, improvizace, algoritmická kompozice hudby, počítačová hudba, prediction suffix tree, PST, markovské řetězce

Keywords

MIDI, improvisation, algorithmic music composition, computer music, prediction suffix tree, PST, markov chains

Citace

Michael Angelov: Hudební improvizace, bakalářská práce, Brno, FIT VUT v Brně, 2009

Hudební improvizace

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Fapša. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Michael Angelov
20.5.2009

Poděkování

Na tomto mieste sa chcem v prvom rade poďakovať svojmu konzultantovi, Ing. Michalovi Fapšovi, za pomoc a podporu, ktorú mi poskytoval počas celého obdobia riešenia bakalárskej práce. Vysoko si cením hlavne jeho ľudský prístup, otvorenosť a nadšenie, ktorým mi dodával odhodlanie do ďalšej práce vo chvílach, keď som ho sám strácal. Ďalej sa chcem poďakovať mojim rodičom, príbuzným a priateľom za morálnu podporu a za všetko dobré z ich strany čo mi v živote do tohto času preukázali.

Obsah

1 Úvod

Hudba je fenomén, ktorý nás obklopuje takmer na každom kroku. Odjakživa je zdrojom oddychu, potešenia a inšpirácie, priestorom pre realizáciu ľudskej kreativity, v ktorej sa odrážajú snáď všetky zložky ľudskej osobnosti - cit, intelekt, kreativita, podvedomie. Avšak napriek každodennej skúsenosti s ňou, hudba ako umenie ostáva zahalená rúškom tajomstva, pretože dodnes sa nepodarilo vystihnúť jej pôvod a táto téma neustále podlieha výskumu rôznych vedných odborov. Ešte väčšmi neprebádanou oblasťou je mechanizmus hudobnej kompozície, s ktorou úzko súvisí slovné spojenie hudobná improvizácia.

Napriek tomu, že oblasť hudby sa tradične spája s ľudskou tvorivosťou a umom, s príchodom informačných technológií do každodenného života prišli aj prvé pokusy o hudobnú kompozíciu riadenú počítačom. S väčším či menším úspechom sa táto oblasť vyvíja až dodnes a prináša nové zaujímavé výsledky, ktoré posilňujú myšlienku využitia umelej inteligencie aj v predtým nedostupných a „výhradne ľudských“ oblastiach života.

Hudobná kompozícia je jedným z mnohých aspektov ľudského správania, a tak akákoľvek snaha namodelovať ju, môže viesť k novým poznatkom nielen v oblasti hudby, ale aj v oblasti umelej inteligencie a kognitívnych vied, čo môže znamenať prínos aj do iných vedných oblastí.

Vo svojej bakalárskej práci sa venujem téme hudobnej improvizácie riadenej počítačom na základe vstupnej MIDI nahrávky. Hlavným štýlom, ktorým by som sa chcel venovať je blues a jazz pretože v ich koreňoch je improvizácia ich nedielnou súčasťou. Výstupom systému bude originálna počítačom komponovaná MIDI nahrávka, s hudobnou tematikou blízkou pôvodnej nahrávke, čím chcem demonštrovať možnosti využitia umelej inteligencie v kompozícií hudby v praxi.

V 2. kapitole sa venujem zadefinovaniu dvoch príbuzných pojmov „kompozícia“ a „improvizácia“ a poukazujem na ich vzájomný vzťah, v 3. kapitole sa venujem analýze MIDI formátu a prehľadu existujúcich nástrojov na analýzu hudby vo formáte MIDI. Vo 4. kapitole vádzam prehľad niektorých používaných prístupov v oblasti počítačom tvorenej hudby. V 5. kapitole sa venuje bližšej špecifikácii zadania a analýze jednotlivých častí improvizačného systému. V 6. kapitole rozpracovávam do detailov návrh budúceho systému a v 7. kapitole popisujem jeho implementáciu. Vo 8. kapitole sa nachádzajú testy hudobného výstupu a v záverečnej 9. kapitole sa venujem celkovému vyhodnoteniu vytvoreného improvizačného systému a úvahám o ďalších možných rozšíreniach systému do budúcna.

2 Hudobná kompozícia a improvizácia

Schopnosť komponovať hudbu je nesporrne výnimočná zručnosť, ktorou disponuje iba malé percento populácie. Keďže mechanizmy sprevádzajúce komponovanie sú dodnes neprebádané, akékoľvek snahy o formalizáciu tohto procesu zatiaľ pracujú iba s nepresným modelmi.

Pod pojmom hudobná kompozícia sa zvyčajne rozumie cieľavedomá činnosť hudobného skladateľa, ktorej výstupom je ucelená skladba. Táto skladba má predom premyslenú štruktúru a motív, pretože je tvorená v predstihu pred jej interpretáciou. Miera náhodnosti v skladbe je nižšia, aj keď na druhej strane ani táto činnosť nie je celkom náhodná. Je potrebné si uvedomiť, že tak ako každá umelecká alebo vedecká činnosť, aj kompozícia hudby by nemohla existovať bez náhodných faktorov ako spontánny nápad, fantázia a inštinktívne zachytenie. Všetky tieto javy predpokladajú okrem cieľavedomej tvorby aj náhodný postup v myslení, teda sčasti ide o náhodné procesy. Bez týchto náhodných procesov, metódy pokusu a omylu, by bola umelecká aj vedecká činnosť nepredstaviteľná. [1]

Akousi podmnožinou hudobnej kompozície je improvizácia. Pod improvizáciou sa spravidla rozumie spontánna kompozícia hudby s vyššou mierou náhodnosti spojená s jej okamžitým prevedením na hudobnom nástroji. Improvizáciu spravidla delíme na dve skupiny – voľná improvizácia a improvizácia viazaná harmonickým základom. Ako názov napovedá, voľná improvizácia nemá vopred dohodnutý harmonický základ a jej štruktúra sa prenecháva improvizujúcemu umelcovi. Viazaná improvizácia naopak predpokladá vopred dohodnutý, spravidla obmieňajúci sa harmonický základ, nad ktorým improvizujúci umelec tvorí nové rytmické a melodické útvary. Mnohí ľudia si myslia, že jazzoví hráči prídu na pódium a hrajú neustále nové sóla, ktoré tvoria priamo na mieste. Väčšinou však ide o vyskúšané a preverené variácie a tak iba malé percento tvorí skutočná „čistá“ improvizácia. Aj najlepší svetoví hráči priznávajú, že táto čistá improvizácia tvorí iba minoritnú časť ich vystúpení a väčšinu ich tvorby tvoria rôzne melodické klišé a frázy, ktoré vhodne spájajú a obmieňajú. [2]

Hudobná improvizácia sa taktiež spája so slovným spojením „improvizovať na danú tému“, čím sa myslí, že novovzniknutý improvizovaný útvar by mal prebrať niektoré črty pôvodnej témy, ktorá je zvyčajne komponovaná vopred, a zároveň by tento útvar mal obsahovať prvky novosti, čím by sa zreteľne odlíšil od pôvodnej skladby. Z týchto úvah možno vytvoriť dva základné body, ktoré by mal akýkoľvek improvizujúci systém spĺňať:

1. Systém by mal byť schopný analyzovať pieseň a zachytiť štýl piesne.
2. Systém by mal byť schopný vo vhodnej miere simulovať náhodné procesy.

V nasledujúcich kapitolách uvediem prehľad existujúcich systémov vzhľadom na oba body.

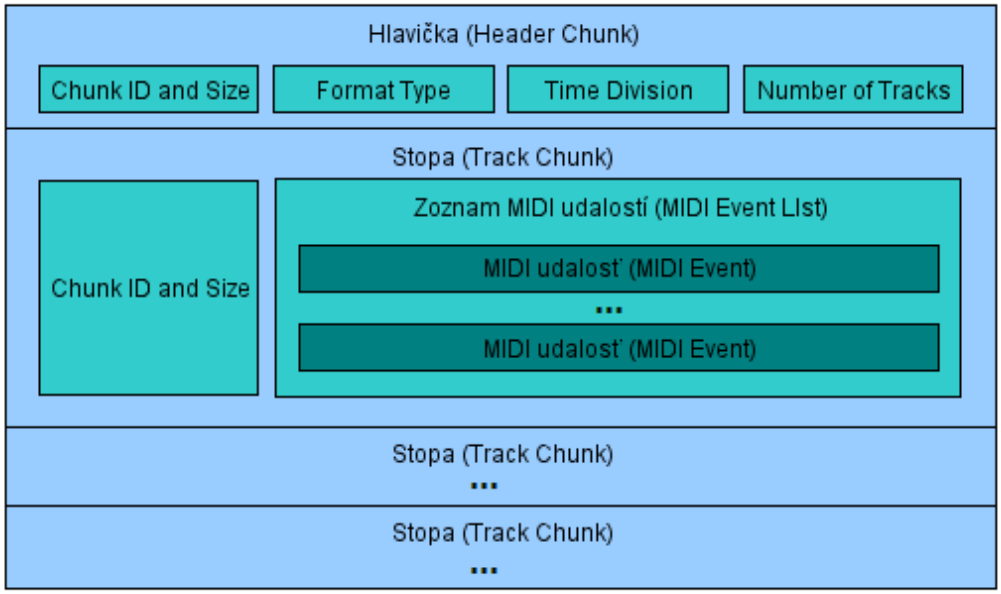
3 Prehľad existujúcich nástrojov na analýzu hudby

Ako som naznačil v predchádzajúcej kapitole, veľmi dôležitým aspektom predchádzajúcim samotný proces improvizácie je proces analýzy hudby. Keďže táto bakalárska práca je zameraná na analýzu a improvizáciu hudby vo formáte MIDI, zamerám sa iba na nástroje pracujúce s týmto formátom. Pokúsim sa ich v dostatočnej miere popísať a poukázať na ich pozitíva a negatíva a posúdiť vhodnosť ich použitia pre moju samostatnú prácu. V rámci tejto kapitoly najprv popíšem formát MIDI a potom sa zamerám na tri nástroje používané na analýzu a spracovanie hudby v MIDI formáte.

3.1 Protokol MIDI a formát MIDI

MIDI (Musical Instrument Digital Interface) je protokol na nahrávanie, zápis, prenos a následnú syntézu hudby v počítačovom prostredí. Jeho počiatky siahajú do začiatku 80. rokov. Umožňuje prenos hudby zo syntetizátora do počítača, a jej následné spracovanie a interpretáciu pomocou počítača. Pre potreby ukladania hudby v počítači slúži formát MIDI, presnejšie SMF (Standard MIDI File). SMF formát narozdiel do formátov WAV alebo MP3, SMF formát neukladá digitalizovanú (prípadne komprimovanú) podobu zvuku. Namiesto toho ukladá parametre použitých hudobných nástrojov, informácie o tempe, časovom predznamenaní a pod. do binárneho súboru s koncovkou .mid. Pri prehrávaní sa tieto parametre jednoducho interpretujú vstavaným syntetizátorom počítača do zvukového výstupu. Vstavaný MIDI syntetizátor obykle povoľuje 16 nezávislých kanálov, v ktorých môže hrať až 256 rôznych hudobných nástrojov. Jediným nástrojom, ktorý sa nepodarilo nasyntetizovať je pochopiteľne ľudský hlas. Sortiment hudobných nástrojov a ich zvukové vlastnosti sú špecifický pre ten ktorý systém, preto sa zvukový výstup pre rovnaký SMF súbor môže systém od systému mierne líšiť.

Hudobná informácia v SMF súbore je rozdelená do tzv. dátových zväzkov (data chunks). Každý zväzok je uvedený hlavičkou identifikujúcou typ zväzku a počet bytov zväzku nasledujúcich za hlavičkou. Prvým dátovým zväzkom každého SMF súboru je hlavičkový zväzok obsahujúci informáciu o type formátu MIDI, základnej dĺžke jednej doby a počte stôp v midi súbore. Tento počet určuje počet nasledujúcich dátovych zväzkov reprezentujúcich jednotlivé stopy. Každá stopa následne obsahuje zoznam MIDI udalostí, o ktorých sa zmienim ďalej. Základnú štruktúru SMF súboru najlepšie vystihuje Obrázok 1.



Obrázok 1: Štruktúra SMF súboru

Dôležitým prvkom každého midi súboru je zoznam MIDI udalostí, ktoré ovplyvňujú všetky detaily týkajúce sa MIDI nahrávky. Každá MIDI udalosť má 3 základné parametre – informácia o čase, v ktorej udalosť nastane, ďalej číslo kanálu, v ktorom nastane a nakoniec informácia o type MIDI udalosti. Každý typ MIDI udalosti má okrem základných parametrov aj niekoľko vlastných. Keďže typov MIDI udalostí je veľké množstvo, v nasledujúcom prehľade sa zamerám iba na niekoľko podstatných.

- *Zapnutie noty* (Note On): Slúži na signalizáciu stlačenia MIDI klávesy. Tento typ udalosti má dva parametre – číslo klávesy reprezentujúce výšku tónu (od 0 do 127) a silu úderu, s akou sa nota zahrá (0 - 127).
- *Vypnutie noty* (Note Off): Slúži na signalizáciu uvoľnenia MIDI klávesy. Rovnako ako udalosť „Zapnutie noty“ má dva paramete – číslo klávesy, ktorá sa má uvoľniť a razanciu, s akou sa nota uvoľní.
- *Zmena programu* (Program Change Event): Slúži na zmenu hudobného nástroja, ktorý hrá v danom kanáli. Má jediný parameter a to index nástroja v tabuľke MIDI nástrojov, ktorý bude hrať.
- *Koniec stopy* (End of Track): Jedná sa o metaudalosť nachádzajúcu sa vždy na konci zväzku stopy a ako názov napovedá, signalizuje koniec stopy.
- *Nastavenie tempa* (Set Tempo): Táto metaudalosť sa zvyčajne nachádza v prvej stope a slúži na nastavania tempa nahrávky. Jediným parametrom je číslo zakódované na 3 bajoch, ktorého hodnota určuje dĺžku štvrťovej noty v mikrosekundách. Ak tempo nie je nastavené,

automaticky sa uvažuje tempo 120 úderov za minútu. Na prepočet medzi klasickým značením tempa (BPM – beats per minute, úderov za minútu) a značením v mikrosekundách za štvrťovú notu (MPQN, microseconds per quarter note) možno použiť nasledujúci vzorec:

```
MICROSECONDS_PER_MINUTE = 60000000
BPM = MICROSECONDS_PER_MINUTE / MPQN
MPQN = MICROSECONDS_PER_MINUTE / BPM
```

- *Časové značenie* (Time Signature): Táto metaudalosť nastavuje časové značenie MIDI nahrávky a predpokladá štyri parametre. Prvý a druhým parametrov je čitateľ a menovateľ časového predznamenania (napr. 4/4), pričom menovateľ je v MIDI súbore zapísaný ako číslo, na ktoré je treba umocniť základ 2, aby sme dostali požadované značenie. Tretím parametrom je značenie rytmu metronómu. Zvyčajne má tento parameter hodnotu 24, čo znamená úder na každú štvrťovú notu, ale je možné sa stretnúť aj so značením 32 (úder na každú štvrťovú notu s bodkou) alebo 48 (úder na každú polovú notu). Štvrtým parametrom je počet 32-tinových nôt za štvrťovú notu, zvyčajne nastavený na hodnotu 8. Ak sa tento parameter neuvedie, implicitne sa používajú hodnoty 4 4 24 8.
- *Značenie tóniny* (Key Signature): Táto metaudalosť sa používa na značenie tóniny MIDI nahrávky. Má dva parametre – prvým je číslo označujúce počet krížikov alebo béčok (v rozsahu od -7 do 7), pričom ak je toto číslo kladné označuje počet krížikov a ak je záporné, označuje počet béčok. Druhým parametrom je označenie módu – buď 0 (dur, major) alebo 1 (mol, minor). Keďže toto značenie implicitne neurčuje základný tón tóniny, na jeho získanie môžme použiť nasledujúci algoritmus (pri predpoklade, že chromatická stupnica C – H je reprezentovaná číslami 0 – 11):

```
IF POCET_#_b >= 0:
      ZAKLADNY_TON = (POCET_#_b * 7) % 12
ELSE:
      ZAKLADNY_TON = (POCET_#_b * -5) % 12

IF MOD == 1: // mol
      ZAKLADNY_TON = (ZAKLADNY_TON + 9) % 12
```

Ako z vyššie uvedených informácií vyplýva, MIDI zápis nepoužíva objektové značenie hudobných útvarov, tak ako je tomu napr. v notovom zápise (nota je objekt). Skôr má podobu zoznamu udalostí (stlačenie noty, pustenie noty) na časovej ose. Tento formát je síce výhodný pre účely prehrávania hudby, avšak pre účely extrakcie zrozumiteľných informácií z MIDI súboru a ich následnej analýzy bude potrebné použitie špecializovaných nástrojov (či už existujúcich, alebo vlastných) na konverziu do objektového formátu, ktorý bude následne použiteľný pre analýzu a improvizáciu. Zároveň ak má

byť výstupom aplikácie MIDI súbor, je potrebné uvažovať aj o spätnej konverzii do MIDI formátu. Prehľadu takýchto nástrojov sa venujem v nasledujúcich kapitolách.

3.2 Mf2t, T2fm

Ako prvý nástroj na analýzu MIDI nahrávky uvádzam dvojicu jednoduchých programov Mf2t (midi file to text) a T2fm (text file to midi). Jedná sa o nízkoúrovňový analyzačný nástroj, pretože jeho hlavný význam spočíva v konverzii z binárneho formátu MIDI do textu a naopak. Asi jedinou funkciou je schopnosť zarovnať MIDI udalosti do úderov a následne do taktov. Program pracuje na platforme Windows a je voľne dostupný.

3.3 Melisma Music Analyzer

Ako z nadpisu vyplýva, Melisma je program pre analýzu hudby v MIDI formáte a extrakciu informácií z nej. [4] Vstupom analyzátoru je „event list“ teda zoznam nôt s ich výškou, časom zapnutia a časom vypnutia (ako vstup možno použiť výstup z programu mf2t). Program extrahuje informácie o usporiadaní do taktov, štruktúru hudobných fráz, ich usporiadanie do melodických línií, harmóniu a tóninu skladby. Jedná sa o voľne dostupný nástroj, skladajúci sa z niekoľkých na seba nadväzujúcich programov. V nasledujúcom prehľade popíšem ich činnosť a pokúsim sa okrem pozitív zamerať aj na ich negatíva.

3.3.1 The Meter Program

Jedná sa o program usporiadavajúci noty do taktov. Vstupom modulu je textový súbor obsahujúci zoznam nôt (ďalej len „notelist“) s časom zapnutia, časom vypnutia a číslom reprezentujúcim výšku noty, ktoré program preusporiada do dôb a taktov. Príkladom vstupu a výstupu je nasledujúca Tabuľka 1.

Vstup	Výstup
Note 0 250 67 Note 250 500 67 Note 500 750 69 Note 750 1000 71 Note 1000 1250 67 Note 1250 1500 71 Note 1500 1750 69 Note 1750 2000 62 Note 2000 2250 67	Beat 0 4 Beat 105 0 Beat 245 1 Beat 350 0 Beat 490 2 Beat 595 0 Beat 735 1 Beat 875 0 Beat 1015 3 Beat 1120 0 Beat 1260 1

Tabuľka 1: Vstup a výstup The Meter Program-u

Výstupom programu je zoznam dôb (ďalej len „beatlist“) s príslušných časovým značením, ktorý sa následne použije v ďalších moduloch programu. Hlavnou nevýhodou tohto programu je, že predpokladá, že sa noty začínajú presne na hranici taktov. Avšak prax ukazuje, že iba malá časť reálnych MIDI nahrávok spĺňa tento požiadavok na presnosť tónov, pretože častokrát ide o skladby nahrávané zo syntetizátora, na ktorom hrá človek, a tak sú noty často rôzne posunuté voči taktom, rôzne sa prekrývajú atď. Aj keď sa jedná o malé odchýlky, pre bežného poslucháča prakticky nepozorovateľné, pre *The Meter Program,* ako som sa presvedčil, predstavujú ťažko prekonateľnú prekážku.

3.3.2 The Grouper Program

Hlavnou činnosťou *The Grouper Program*-u, je organizácia melódie do fráz. Vstupom programu je notelist a beatlist z predchádzajúceho modulu. Výstupom programu je pozmenený notelist s pridaným značením o začiatku a konci hudobých fráz. Program vie pracovať iba s monofonickým vstupom, takže prekrývajúce sa noty „oreže“, avšak v prípade, že sa noty začínajú v rovnakom čase (napr. v dvojhlase alebo akorde) program vyhlási chybu a ukončí sa. Táto vlastnosť je z pohľadu reálnej MIDI nahrávky, bohužiaľ, nežiadúca.

3.3.3 The Streamer Program

The Streamer Program zoskupuje noty do melodických liniek. Vstupom programu je takisto ako v predchádzajúcom module notelist a beatlist, pričom melodické linky sa nesmú prekrývať ani križovať. Výstup programu je vskutku pôsobivý (viď Tabuľka 2), avšak jeho využitie vo väčšine MIDI nahrávok je otázne, pretože zvyčajne hrá každú stopu v MIDI nahrávke iný nástroj v inom kanáli, teda jednotlivé linky melodické linky sú od seba implicitne oddelené.

```
Seg   98 (3):           .  3        .      1    4          .          .
Seg   99 (0):           .           3      1    4          .          .
Seg  100 (1):           .          3.      1    .      4   .          .
Seg  101 (0):           .        3  .      1    .     4    .          .
Seg  102 (2):           .      3    .      |    .      4   .          .
Seg  103 (0):           .    3      .      |    .      4   .          .
Seg  104 (1):           .  3        .          1. 4         .          .
Seg  105 (0):           . 3         .          1. 4         .          .
Seg  106 (4):           3           .           1  4        .          .
Seg  107 (0):           . 3         .           1  4        .          .
Seg  108 (1):           .  3        .           1  4        .          .
Seg  109 (0):           . 3         .          1.  4        .          .
Seg  110 (2):           3           .           1  |        .          .
Seg  111 (0):         3 .           .           1  |        .          .
Seg  112 (1):       3   .           .      1    .   4       .          .
Seg  113 (0):      3    .           .      1    .   4       .          .
```

Tabuľka 2: Výstup programu The Streamer

3.3.4 The Harmony Program

The Harmony Program je program na harmonizáciu melódie. Mechanizmus programu spočíva v oddelení najnižších tónov (spravidla predstavujúcich koreňovú notu) od zbytku melódie a hľadaní najlepšej kombinácie tónov vo vyšších tónoch k danej koreňovej note. Ak je pre daný úsek možných viacero kombinácií, vyberie sa tá „najlepšia“, teda tá, ku ktorej prislúcha najviac tónov.

3.3.5 The Key Program

The Key Program je program na analýzu tóniny nahrávky a rozdelenie skladby na sekcie označené tóninou danej sekcie. Zároveň určuje jednotlivé akordy v skladbe a ich harmonickú funkciu voči tónine celej skladby. Mechanizmus jeho činnosti je podobný ako je tomu v prípade *The Harmony Program*-u – pre každú sekciu sa vytvorí vektor reprezentujúci danú sekciu a následne sa porovná s vektormi charakteristickými pre jednotlivé akordické funkcie.

3.3.6 Celkové hodnotenie

Program *Melisma Music Analyzer* predstavuje nepochybne zaujímavý nástroj na analýzu hudby formátu MIDI. Ako som však už spomínal v predchádzajúcich kapitolách, Melisma disponuje niekoľkými vážnymi nedostatkami, ktoré rapídne znižujú možnosti jej nasadania v reálnom prostredí rôznorodých MIDI nahrávok, a určite nepoteší ani fakt, že program priamo nepracuje s formátom SMF. Avšak Melisma sa môže stať nepochybne bohatým zdrojom inšpirácie pre prípadný vlastný nástroj, aj keď s možno nie až tak prepracovanou funkcionalitou, ktorá na druhej strane nie je vždy úplne nevyhnutná.

3.4 Midi Toolbox

Midi Toolbox (ďalej len „Toolbox“) je zbierka funkcií pre analýzu a vizualizáciu MIDI súborov vo výpočetnom prostredí Matlab. [5][6] Toolbox je voľný dostupný nástroj šírený pod licenciou GNU GPL, ktorý však predpokladá inštaláciu Matlabu-u, ktorý zdarma nie je. Na druhej strane však Toolbox ponúka veľké množstvo funkcií, ktoré možno rozdeliť do niekoľkých kategórií:

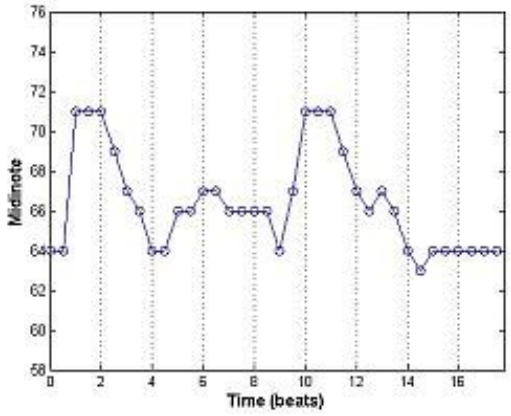
- Konverzné funkcie – umožňujú import a export SMF súboru do Matlabu a naopak
- Filtračné funkcie – umožňujú editáciu hudobného materiálu, napríklad transpozíciu, extrakciu hudobných fráz a pod.
- Kresliace a štatistické funkcie – vyhodnocujú a vizualizujú vlastnosti a štruktúru MIDI súboru

Veľkou výhodou programu je schopnosť pracovať priamo so SMF súborom, ktorý sa načíta pomocou jedného príkazu do dátovej štruktúry zvanej *notová matica* (notematrix). Nad touto maticou následne program vykonáva jednotlivé funkcie.

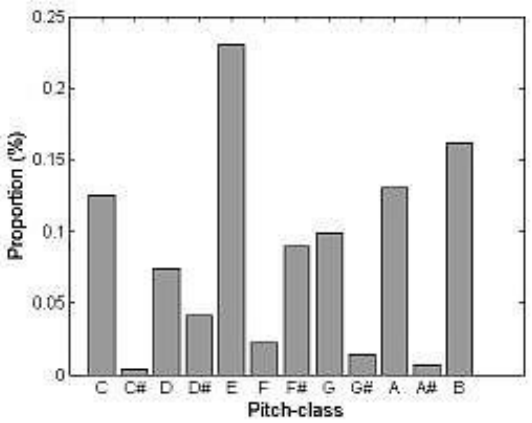
Hlavnou silou Toolboxu sú jeho okrem jeho analytických funkcií aj štatistické a vizualizačné funkcie, ktorých výstup je však určený predovšetkým na spracovanie užívateľom, preto sa domnievam, že vhodnejšie ako pri vstupnej analýze, by bolo použiť tento nástroj pri analýze výstupov improvizačného systému. Preto sa aj v nasledujúcich riadkov pri popise funkcií tohto nástroja, zamerám na jeho štatistické funkcie.

Zaujímavou funkciou je napríklad funkcia na určenie melodickej podobnosti dvoch súborov. Táto funkcia umožňuje okrem porovnávania melódie ako celku aj porovnávanie menších častí a podľa môjho úsudku by bola veľmi dobre využiteľná pri porovnávaní originálnej a improvizovanej nahrávky na určovanie miery vzájomnej podobnosti, resp. odlišnosti.

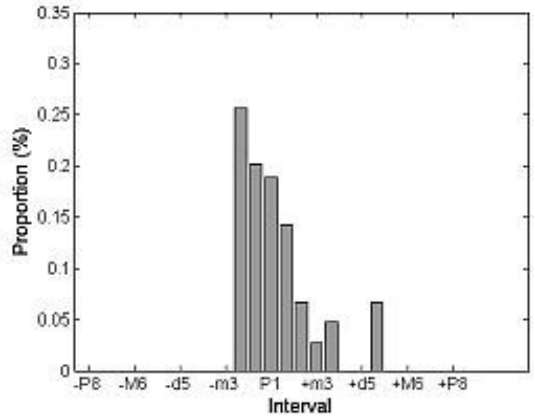
Medzi štatistické funkcie ďalej patrí znázornenie tvaru melódie, distribučné rozloženie jednotlivých tónov, intervalov a pravdepodobnostné rozloženie prechodov medzi dvojicami tónov (tzv. prechodová matica) a ich následné grafické zobrazenie, ako možno vidieť na obrázkoch 2, 3, 4, a 5. Posledné dve funkcie fungujú, bohužiaľ, iba na monofonických skladbách.



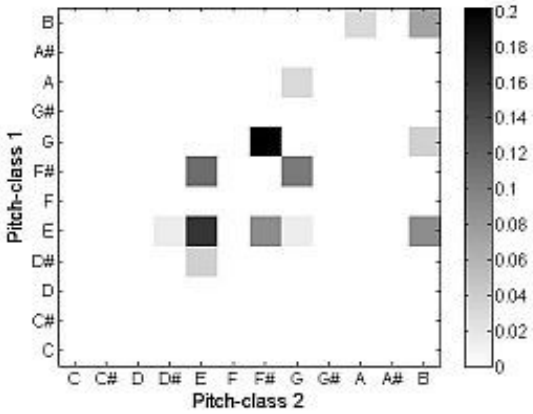
Obrázok 2: Tvar melódie



Obrázok 3: Štatistické rozloženie tónov



Obrázok 4: Štatistické rozloženie intervalov



Obrázok 5: Prechodová matica

4 Prehľad existujúcich systémov na hudobnú kompozíciu a improvizáciu

V tejto kapitole sa chcem venovať prehľadu existujúcich systémov na hudobnú kompozíciu a improvizáciu. Tieto dva pojmy som odlíšil v kapitole 2, avšak z pohľadu algoritmických systémov bude dobré rozlíšiť systémy na kompozíciu a systémy na improvizáciu a to aspoň v rámci tejto práce. Hlavný rozdiel medzi obidvomi systémami chápem v ich spojenií s existujúcim hudobným vstupom. Kým systémy na kompozíciu nepotrebujú pri svojej činnosti na začiatku hudobný vstup, sú akoby „samočinné“ - teda štruktúra a štýl skladby sú plne v moci algoritmu, systémy na improvizáciu ho nevyhnutne na vstupe potrebujú, pretože ich činnosť spočíva v „improvizácií na danú tému“. Aj keď by sa dalo namietať, že improvizácia nemusí byť nevyhnutne „na danú tému“, pre potreby tejto práce to tak budem uvažovať.

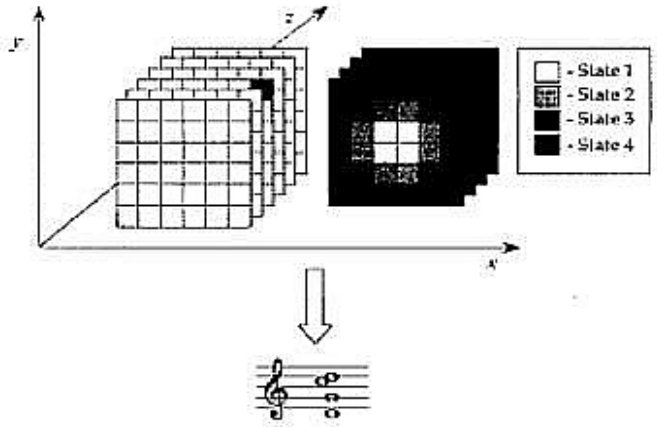
Ďalším delením je rozdelenie systémov na deterministické a stochastické. Kým deterministické systémy sa správajú podľa vopred daného algoritmického mechanizmu, ktorého budúci výsledok je v každom čase dobre predikovateľný, stochastické systémy závisia na zákonoch pravdepodobnosti, rozhodnutia sa robia na základe náhodných čísel, teda v akomkoľvek čase je nemožné presne predpovedať výsledok systému. Vo všeobecnosti možno povedať, že stochastické systémy sú prístupnejšie, pretože nevyžadujú komplikovanú algoritmizáciu toho ktorého hudobného štýlu, tak ako je tomu pri deterministických systémoch. [7]

4.1 Celulárne automaty

Celulárne automaty (CA) sú spojené systémy, zložené z množstva menších systémov (buniek), spravidla stavových automatov. Stavový automat je systém, ktorého výstup závisí od jeho vnútorného stavu a vstupu. Pri použití v hudbe môžu stavové automaty modelovať správanie rôznych hudobných parametrov (výška, farba, amplitúda atď). Ak sú stavové automaty pospájané, vytvárajú systém, ktorý prenáša výstup z jedného automatu na vstup druhého. Takýto systém po počiatočnom vstupe beží autonómne. Jeho evolúcia je podmienená vnútornou analýzou meniacou vstupy a váhu jednotlivých prechodov medzi automatmi.

V prípade CA sú takéto systémy medzi sebou vzájomne symetricky pospájané a paralelne vykonávajú evolučný algoritmus, ktorého výsledok závisí od lokálneho susedstva. Takýto CA sa buď ustáli v chaose, periodickom deji, alebo zanikne.

Príkladom takéhoto systému používajúceho CA je systém CAMUS 3D. [8] V priebehu celej kompozície dochádza k transformácií hudobnej tematiky. Program využíva dve CA štruktúry - Game of Life a Demon Cyclic Space. V každom kroku sú analyzované súradnice živých buniek, a pomocou nich sa určí 4-notový akord. Ako možno vidieť na Obrázku 6, v štruktúre Game of Life je bunka (5, 5, 2) aktívna, teda sa vyberie akord, ktorého tóny sú 0, 5, 5+5 a 5+5+2 poltónov nad zvoleým základom. V štruktúre Demon Cyclic Space je cyklická štruktúra v stave 4, to znamená že daný akord sa zahrá na nástroji číslo 4. Základný tón sa volí buď podľa určitého algoritmu, alebo stochasticky.[7]



Obrázok 6: Algoritmická kompozícia pomocou CA (prevzaté z [7])

4.2 Gramatiky

Vo väčšine hudby prirodzene existuje istý stupeň hierarchizácie. Jednotlivé noty sa navzájom spájajú a vytvárajú frázy, sekcie a nakoniec celé skladby. Problém týkajúci sa mnohých kompozičných algoritmov spočíva v nedostatku makroštruktúry. Hudba, ktorá má iba jednoduchú hierarchiu môže iba ťažko zaujať, preto je nutné vniesť do nej istú hierarchiu, čo sa dá dosiahnuť práve formálnymi gramatikami. Pod pojmom gramatika rozumieme množinu pravidiel na rozšírenie symbolov vyššej úrovne na detailnejší popis nižšej úrovne. Takéto gramatiky sa používajú aj v prirodzených jazykoch.

Mechanizmus kompozície pomocou gramatiky spočíva v niekoľkých krokoch. Najprv musí hudobný skladateľ zostrojiť hudobnú gramatiku, ktorá sa skladá z rôznych symbolov zvaných *tokeny*. Neterminálne tokeny reprezentujú makroštruktúry a terminálne tokeny reprezentujú najjednoduchšie štruktúry – noty. Hierarchia rôznych tokenov je určená tzv. *prepisovacími pravidlami*, ktoré jednoznačne určujú akými tokenmi nižšej úrovne môže byť nahradený token vyššej úrovne. Takéto gramatiky sú buď kontextové (tj. berú do úvahy okolie tokenu vyššieho tokenu) alebo bezkontextové. [7]

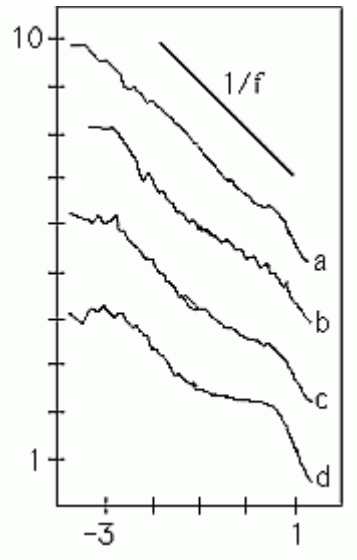
4.3 Expertné systémy

Expertné systémy spočívajú v izolácií estetických pravidiel istého žánru a v následnej algoritmickej implementácií týchto pravidiel. Avšak okrem pravidiel tieto systémy pozostávajú z rôznych heuristických funkcií simulujúcich štruktúru špecifického žánru alebo činnosť niektorého skladateľa. Expertné systémy sú teda vhodné na generovanie hudby, ktorej štýl sa dá popísať faktami, pravidlami a heuristikou, ako je napríklad štýl Bachových klavírnych skladieb, alebo niektorých iných skldateľov tonálnej éry. Najväčšou prekážkou v aplikácií expertných systémov na moderné štyly je, že pre ich variabilitu týmto štýlom zväčša nevyhovuje akýkoľvek popis pomocou pravidiel. V takomto prípade by sa systém musel učiť z predloženého materiálu a vytvoriť si vlastnú znalostnú základňu, čo však predstavuje neľahký problém. [7]

4.4 Fraktály

Ďalšou možnosťou hudobnej kompozície sú fraktály. Ich hlavnou využívanou vlastnosťou je *sebapodobnosť,* teda že menšia časť fraktálu sa podová na väčšiu časť, iba v inej mierke. Táto sebapodobnosť však fascinovala hudobných skladateľov už dlhé obdobie predtým ako boli objavené fraktály. J. S. Bach a iní skladatelia v niektorých svojich dielach použili tento princíp sebapodobnosti a to tak, že postupnosť nôt v malých melodických úsekoch určovala harmonickú postupnosť vyšších hudobných celkov – fráz a partitúr. Podľa Mandelbrota hudba preukazuje fraktálové správanie práve preto, že je vysoko hierarchizovaná.[7]

Ďalšie využitie fraktálov v hudbe súvisí so spektrálnou hustotou šumu 1/*f*. Pomocou experimentu, ktorý prebiehal v 70. rokoch sa zistilo, že mnohé prakticky všetky hudobné štýly vykazujú spektrálnu hustotu veľmi podobnú spektrálnej hustote 1/*f* (Obrázok 7). [9] Preto sa autori tohto objavu rozhodli použiť generátor 1/*f* šumu na výber výšky tónov v hudobnej kompozícií. Náhodné čísla boli zaokrúhlené a pretransformované do tónov v rozsahu dvoch oktáv. Takisto použili generátor bieleho a hnedého šumu. Kým melódie generované bielym šumom boli príliš náhodné, hnedý šum generoval melódie veľmi jednotvárne. Melódie generované šumom 1/*f* zneli hudobne najprirodzenejšie.



Obrázok 7: Spektrálna hustota rôznych hudobných štýlov:

(a) klasická hudba

(b) jazz a blues

(c) rock

(d) správy a hovorené slovo

Všetko spriemerované v priebehu 12 hodín vysielania.

(Prevzaté z [9])

4.5 Stochastické generátory

Ďalšiu skupinu zo skupiny systémov na hudobnú kompozíciu sú stochastické generátory. Ako názov napovedá, tieto systémy využívajú na generovanie hudby pravdepodobnostný model. Takéto systémy sú väčšinou zamerané na experimentovanie s melódiami na základe rôznych užívateľom zadaných parametrov, ktoré riadia proces generovania hudby. Medzi takéto parametre patrí napríklad miera tonality, tónina, tempo, rytmus, rozsah použitých nôt, koeficient opakovania nôt, maximálna dĺžka frázy a iné. Príkladom takýchto systémov sú systémy *Melisma Stochastic Melody Generator* a systém českého pôvodu *CCOMP*.

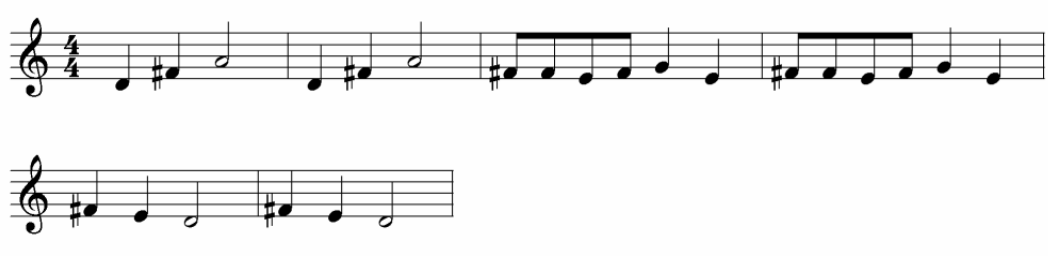
4.6 Markovské procesy

Úplne odlišný prístup k popisu hudby ponúkajú markovské procesy. Pod pojmom markovský proces rozumieme náhodný proces bezpamäťového systému, pre ktorý platí, že pravdepodobnosť prechodu do budúceho stavu je závislá len na aktuálnom stave systému. Ide teda o stochastický systém. Ak sa takýto systém nachádza v diskrétnom čase, nazýva sa tiež markovský reťazec. Vytvorenie takéhoto systému je závislé na vstupe systému, preto akýkoľvek systém pracujúci na princípe markovských procesov potrebuje na vstupe hudobnú nahrávku, podľa ktorej sa nastavia parametre systému. Preto pri popise systému dám prednosť pojmu hudobná improvizácia pred kompozíciou.

4.6.1 Jednoduché markovské reťazce

Pri použití jednoduchých markovských reťazcov v hudbe sa hudobná skladba popíše ako sled rôznych stavov, pričom jednotlivé stavy môžu byť reprezentované rôznymi veličinami (výška noty, dĺžka noty, sila úderu). Následne sa pre každý z jedinečných stavov systému vytvorí vektor prechodových funkcií $P(A, B)$s pravdepodobnosťou prechodu zo stavu A do stavu B. Spojením všetkých vektorov do jednej tabuľky vznikne prechodová matica. Prechodová matica následne slúži ako stochastický model pre generovanie novej postupnosti tónov, s danými pravdepodobnosťami prechodu z aktuálneho stavu do iného. Ekvivalentným zápisom prechodovej matice je aj orientovaný graf s ohodnotenými hranami (Obrázok 8).

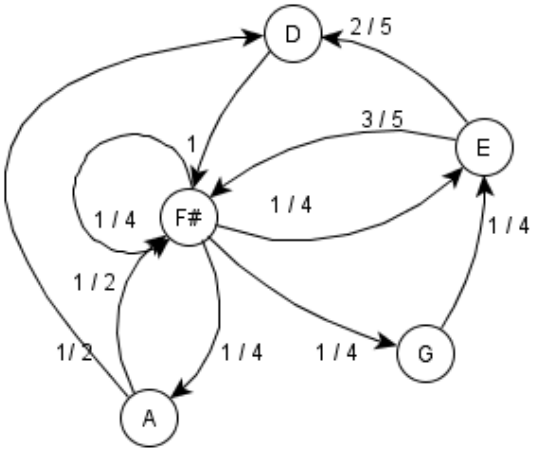
Príklad : (pieseň Ovčáci čtveráci)



Stavy systému (podľa výšky nôt): *D*, *E*, *F#*, *G*, *A*

$P(A, B)$	**D**	**E**	**F#**	**G**	**A**
D	0	0	1	0	0
E	2 / 5	0	3 / 5	0	0
F#	0	1 / 4	1 / 4	1 / 4	1 / 4
G	0	1	0	0	0
A	1 / 2	0	1 / 2	0	0

Tabuľka 3: Prechodová matica prvého rádu



Obrázok 8: Ekvivalentný zápis prechodovej matice pomocou grafu

Keďže markovský systém je bezpamäťový systém, neberie do úvahy minulé stavy, čo pre potreby hudobnej improvizácie nie je príliš vyhovujúce. Avšak existuje možnosť ako takýto bezpamäťový systém pretransformovať na systém berúci do úvahy aj minulé stavy systému a to zvyšovaním rádu prechodovej matice. Kým prechodová matica prvého rádu berie do úvahy iba aktuálny stav tvorený

jedinou notou, prechodová matica druhého a vyššieho rádu rozširuje svoje stavy o predchádzajúce noty (Tabuľka 4).

$P(A, B)$	**D**	**E**	**F#**	**G**	**A**
D F#	0	1 / 3	0	0	2 / 3
F# A	1 / 2	0	1 / 2	0	0
A D	1	0	0	0	0
A F#	1	0	0	0	0
F# F#	0	1	0	0	0
F# E	1 / 2	0	1 / 2	0	0
E F#	0	0	1 / 2	1 / 2	0
F# G	0	1	0	0	0
G E	0	0	1	0	0
E D	0	0	1	0	0

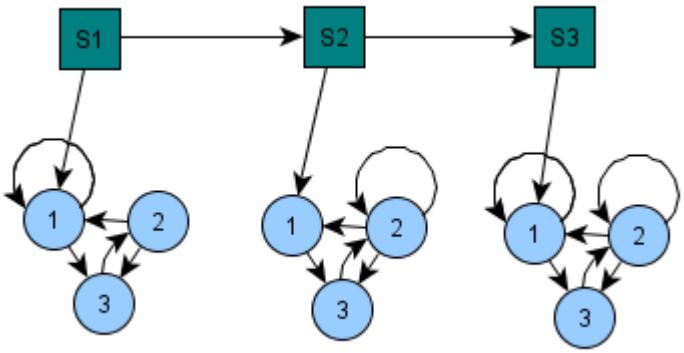
Tabuľka 4: Prechodová matica druhého rádu

Kým markovské reťazce prvého rádu sa takmer nepoužívajú, reťazce druhého a vyššieho rádu sa osvedčili ako hudobne zaujímavé. [7]

4.6.2 Kontrolované markovské reťazce

Hlavnou nevýhodou jednoduchých markovských reťazcov, je malá miera interakcie medzi vstupnou skladbou a systémom na nej vybudovaným. V jednoduchom systéme neexistujú prostriedky na simuláciu „vyšších“ atribútov skladby ako prechody medzi harmonickými funkciami, zmeny rytmu, tempa atď.

Princíp kontrolovaných markovských reťazcov spočíva práve v nájdení a poprepájaní týchto „skrytých“ stavov v rámci danej skladby, pričom každý zo skrytých stavov disponuje vlastným jednoduchým markovským reťazcom simulujúcim postupnosť tónov v danom skrytom stave (Obrázok 9). Tieto skryté stavy môžu byť určené rôznymi faktormi ako napríklad meniaca sa tónina skladby, harmonická funkcia, zmena rytmu, alebo tempa. Takto vytvorená hiearchia lepšie simuluje reálnu skladbu. [3]



Obrázok 9: Grafické znázornenie kontrolovaných markovských reťazcov

4.7 Prediction Suffix Tree

Napriek tomu, že markovské reťazce vykazujú pri improvizácií zaujímavé výsledky, ukázalo sa, že kým reťazce nižšieho rádu generujú často veľmi náhodný výstup, reťazce vyšších rádov generujú výstup veľmi podobný pôvodnej nahrávke. [12] Aby sa predišlo týmto nežiadúcim javom, ako kompromis medzi nimi sa ukázali tzv. stromové metódy. Ich hlavný princíp spočíva v rozdelení hudobnej skladby na slovník fráz a melodických vzorov (motívov) a vytvorení pravidiel pre výber ďalšieho hudobného objektu, ktorý najlepšie zapadne do aktuálneho hudobného motívu.

Veľkou výhodou stromových metód oproti klasickým markovským reťazcom (ktoré pre vyššie rády budujú spravidla veľmi veľké prechodové matice) je schopnosť „kompresie“ veľkého množstva dát nachádzajúcich sa v hudobnom vstupe do relatívne malého stromu. Ďalej je možné automatické odfiltrovaniu tých motívov, ktoré sa v skladbe vyskytujú len okrajovo.

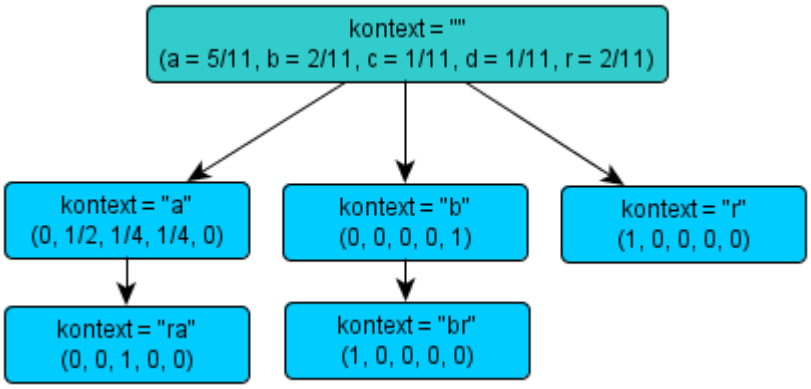
Špecifickým príkladom stromovej metódy je metóda Prediction Suffix Tree (PST). [13] Jej mechanizmus spočíva v prehľadávní do šírky (breadth-first search) všetkých motívov, ktoré spĺňajú nasledujúce parametre:

- Dĺžka motívu nie je dlhšia ako maximálna dĺžka *L*.
- Pravdepodobnosť výskytu motívu v sekvencii prekračuje prah *Pmin.*
- Pravdepodobnosť symbolu *S* v danom uzle *M* je *R*-krát vyššia ako v uzle o jeden symbol kratšom.

Pravdepodobnosť výskytu motívu v sekvencii sa rovná podielu početnosti jeho výskytu a maximálnej teoretickej početnosti jeho výskytu. Napríklad pravdepodobnosť výskytu motívu „*aa*“ v sekvencii „*aabaaab*“ je 3/6 = 0.5. Pravdepodobnosť symbolu *S* v danom uzle s motívom *M* sa rovná podielu

početnosti výskytu tohto symbolu za týmto motívom a početnosi výskytu tohto motívu v sekvencii. Pravdepodobnosť symbolu „*b*“ za motívom „*aa*“ je v sekvencii „*aabaaab*“ rovná 2/3. Pravdepodobnosť symbolu „*b*“ za motívom „*a*“ v rovnakej sekvencii je rovná 2/5.

PST analýzou sekvencie „*abracadabra*“ s parametrami $L = 2$, $Pmin = 0.15$ a $R = 2$ sa dopracujeme k nasledujúcemu stromu (Obrázok 10):



Obrázok 10: Prediction Suffix Tree sekvencie "abracadabra"

Tento strom sa následne použije v procese generovania. Napríklad pravdepodobnosť vygenerovania sekvencie „abrac“ sa rovná:

P(„a“ | „”) * P(„b” | „a”) * P(„r” | „ab”) * P(„a” | „abr”) * P(„c” | „abra”) = 5/11 * 1/2 * 1 * 1 * 1

5 Analýza

Ako bolo ukázané v predchádzajúcich kapitolách, v doméne hudobnej kompozície a improvizácie riadenej počítačom existuje mnoho veľmi odlišných prístupov. Preto je dôležité problém dôkladne rozanalyzovať a následne navrhnúť vhodnú kombináciu analytických, improvizačných a synetizačných nástrojov tak, aby ich vzájomná kombinácia prinášala čím lepšie výsledky zodpovedajúce zadaniu.

5.1 Analýza a špecifikácia zadania

Ako zo zadania bakalárskej práce vyplýva, mnou vytvorená aplikácia má vykonávať dve hlavné činnosti – spracovanie a analýza predloženej MIDI nahrávky a následná improvizácia podľa nahrávky "na danú tému“. Aj keď výstupný formát aplikácie nie je priamo určený, budem predpokladať, že výstupom by mala byť taktiež nahrávka vo formáte MIDI.

Ako som v teoretickom úvode naznačil, pojem improvizácia nie je celkom jasne definovaný a taktiež existujú rôzne druhy improvizácie, ktoré sa svojou povahou od seba do značnej miery líšia. Preto je z pohľadu analýzy a následného návrhu potrebné vyjasnenie, o aký typ improvizácie sa má v programe jednať.

Z pohľadu „voľnosti“ improvizácie sa ako prvá alternatíva ponúka freejazzová improvizácia, tj. výstupom programu by bola nahrávka na báze pôvodnej nahrávky, avšak už bez jasného harmonického základu. Osobne nie som zástancom tohto štýlu, pretože mám pocit, že takýto výstup programu, nech aj by bol akokoľvek kvalitný, oslovil by iba malú časť poslucháčov. Ako oveľa zaujímavejší prístup sa mi javí improvizácia na základe harmónie pôvodnej nahrávky, teda improvizácia nie len tak „zbrucha“, ale na základe harmonických pravidiel pôvodnej skladby. Tento prístup je síce návrhársky a implementačne náročnejší, avšak ak sa ho podarí zrealizovať, mohol by priniesť zaujímavé výsledky.

Ďalším pohľadom na problém je otázka polyfónie. Ľahko možno súhlasíť s názorom, že jednodhlasná hudba je z pohľadu poslucháča menej zaujímavá ako hudba viachlasná. Zároveň však treba podotknúť, že polyfonická improvizácia je z pohľadu hudobníka aj z pohľadu algoritmickej kompozície oveľa náročnejšia ako monofonická, pretože vyžaduje vhodnú aplikáciu veľkého množstva znalostí tak, aby sa predišlo rušivej disharmónií. Prakticky všetky mne známe publikácie venujúce sa téme hudobnej improvizácie (nie kompozície) uvažujú monofonickú improvizáciu. Tento prístup je logický – monofonická improvizácia je ľahšie realizovateľná a prípadnou následnou harmonizáciou melódie možno docieliť požadovaný polyfonický výstup. Keďže mojim zámerom je improvizácia na danom harmonickom základe, vo svojej práci sa zamerám na improvizáciu

monofonickej melódie, ktorá však vo výsledku bude znieť nad harmonickým doprovodom pôvodnej skladby, takže v konečnom dôsledu nedôjde k narušeniu polyfonického rázu skladby.

Ďalším pohľadom na zadanie je otázka kvantitatívnej a kvalitatívnej náročnosti programu na vstupe. Z pohľadu kvantitatívnej náročnosti prichádzajú do úvahy dve hlavné alternatívy. Prvou alternatívou je budovanie databázy hudobných motívov z veľkého počtu vstupných nahrávok, teda ide o akési „zachytenie žánru“. Druhou alternatívou je improvizácia iba na základe jedinej vstupnej nahrávky, teda ide skorej o „zachytenie motívu“. Dôležitým činiteľom je taktiež kvalitatívna náročnosť aplikácie na vstup. Mojím zámerom je, aby bol program schopný spracovať čím širšiu škálu reálne dostupných MIDI nahrávok reprezentujúcich konkrétne piesne. V prípade realizácie improvizácie na štýl „zachytenie žánru“ by boli takéto nahrávky podľa môjho názoru ťažko spracovateľné a výstup ktorý by vznikol spojením niekoľkých rôznych piesní by pravdepodobne pôsobil nekonzistentne, preto sa skorej prikláňam k improvizácií na štýl „zachytenie motívu“, teda vstupom programu by bola jediná skladba. Takýto prístup je taktiež oveľa priateľskejší k užívateľovi, pretože od neho nepožaduje veľké množstvo vstupov.

Z uvedených úvah a s istou dávkou abstrakcie teda možno vyvodiť o funkcionalite aplikácie nasledujúci „jednoduchý“ záver – aplikácia by sa mala správať ako hudobník (prípadne skupina hudobníkov - sólistov), ktorý sa započúva do predloženej skladby a po jej vypočutí jednotlivé party na príslušných nástrojoch zaimprovizuje na „danú tému“.

5.2 Výber MIDI analyzačného nástroja

Výberu vhodného analyzačného nástroja je potreba venovať veľkú pozornosť, pretože nevhodný výber by mohol v ďalších etapách projektu skomplikovať prácu a následne ohroziť kvalitu výstupu improvizačného systému. Pred samotným výberom je však v prvom rade potrebné stanoviť si požiadavky na funkcionalitu nástroja. Medzi hlavné mnou preferované funkcie patrí:

- priame spracovanie MIDI nahrávky vo formáte SMF
- segmentácia nahrávky na logické celky (takty alebo melodické úseky)
- určenie rytmu a tempa
- extrakcia melódie a oddelenie od doprovodu
- určenie tóniny skladby vrátane módu (dur/mol)
- určenie harmonickej funkcie daného úseku (relatívne voči tónine skladby alebo voči referenčnému tónu)

V predošlých kapitolách som načrtol tri funkcionálne odlišné nástroje na analýzu hudby v MIDI formáte – nástroj Mf2t T2fm, Melisma Music Analyzer a Midi Toolbox. Z týchto troch aplikácií vie s MIDI formátom priamo pracovať iba jeden – Midi Toolbox. Ako som však už uviedol, funkcie Toolboxu sú zamerané predovšetkým na analýzu MIDI nahrávok v zmysle štatistického vyhodnocovania a grafického znázornenia rôznych hudobných javov, preto tento nástroj nepovažujem za vhodný na vstupnú analýzu MIDI nahrávky. Okrem toho má Toolbox problémy so spracovávaním polyfonických nahrávok, tj, nahrávok s viacerými stopami.. Avšak jeho funkcie sa budú hodiť pri vyhodnocovaní výstupov a štatistickom porovnávaní s originálnymi nahrávkami.

Ďalšou spomínanou aplikáciou je Melisma Music Analyzer. V porovnaní s Toolboxom poskytuje Melisma omnoho vyššiu flexibilitu pri práci s MIDI nahrávkou, aj keď program nie je schopný pracovať priamo so SMF formátom na vstupe a ani nepodporuje polyfóniu v zmysle viacerých stôp v MIDI. Čo sa týka funkcionality Melismy, jedná sa o prepracovaný nástroj, avšak jeho tvorcovia pravdepodobne nerátali s nasadením nástroja v rôznorodom prostredí reálnych nahrávok. Ako som sa totiž sám mohol presvedčiť, sofistikované funkcie toho programu zlyhávajú už na pomerne jednoduchých „reálnych“ vstupoch.

Posledným spomínaným nástrojom je dvojica Mf2t – T2fm. Ako je už v tejto práci spomenuté, jedná sa o nástroje na konverziu binárneho MIDI formátu do textového prepisu a naopak. Program síce neposkytuje prakticky žiadnu funkcionalitu, avšak javí sa ako vhodný nástroj na prvotnú analýzu neprehľadnej binárnej informácie uloženej MIDI súbore a v neposlednom rade aj ako nástroj na finálnu syntézu výstupnej MIDI nahrávky.

Z uvedených informácií vyvodzujem nasledujúci záver – keďže sa mi pri prieskume existujúcich nástrojov nepodarilo naraziť na úplne vyhovujúci analyzačný nástroj, prikláňam sa k možnosti vytvorenia vlastného nástroja s vyššie uvedenou požadovanou funkcionalitou na báze nástroja Mf2t a T2fm, pričom sa pri návrhu inšpirujem hlavne nástrojom Melisma Music Analyzer.

5.3 Výber improvizačného mechanizmu

Kľúčovou časťou celého improvizačného systému je improvizačný mechanizmus, ktorý po fáze analýzy vstupnej nahrávky generuje nový hudobný materiál. Ako z analýzy a špecifikácie zadania vyplýva, improvizačný mechanizmus má improvizovať „na danú tému“, pričom má rešpektovať harmonické zákonitosti pôvodnej nahrávky.

Z uvedeného východiska sa ako veľmi dobré riešenie javí použitie stochastických modelov. Ich mechanizmus spočíva v „natrénovaní“ modelu podľa vstupných sekvencií a následnom generovaní nových sekvencií s rovnakým pravdedpodobnostným rozložením ako v pôvodnej sekvencii. Do úvahy prichádzajú najmä kontrolované markovské reťazce – ich hlavnou výhodou oproti

jednoduchým markovským reťazcom je existencia „skrytých“ stavov, pomocou ktorých možno simulovať akúsi „kontextovosť“, teda závislosť generovanej sekvencie od kontextu, v ktorom sa systém práve nachádza. Tu sa ponúka možnosť spojiť pojem „kontext systému“ s pojmom „harmonická funkcia“, čím sa defacto vytvorí improvizačný systém generujúci tóny na základe aktuálnej harmonickej funkcie reprezentujúcej skrytý stav (teda kontext) systému.

Hlavnou nevýhodou markovských reťazcov je ich ťažkopádnosť – zvyšovaním rádu prechodovej matice sa priamo úmerne zvyšuje jej pamäťová náročnosť, čo sa pri veľkých maticiach vysokých rádov ukazuje ako nevýhoda. Okrem toho, ako som už v rámci tejto práce spomenul, kým prechodové matice vyššieho rádu generujú výstup veľmi podobný pôvodnej nahrávke, matice nižšieho rádu generujú vstup veľmi náhodný. Ako kompromis medzi oboma prístupmi sa ukazujú stromové metódy.

Hlavnou výhodou stromových metód je ich pamäťová nenáročnosť. Ich nevýhodou je zase vyššia algoritmická náročnosť implementície. Aj keď je mechanizmus stromových metód diametrálne odlišný od markovských reťazcov, oba mechanizmy fungujú na podobnom princípe – generujú nové sekvencie na stochastickom princípe podľa aktuálneho stavu systému. Jedná sa teda o dva takmer ekvivalentné markovské systémy s tým rozdielom, že stromové metódy dokážu modelovať markovské reťazce s premenným rádom (ekvivalent rádu prechodových matíc).

Ako vhodná stromová metóda sa javí metóda prediktívneho stromu (Prediction Suffix Tree). Zaujímavou je taktiež jej schopnosť filtrácie okrajových motívov, ktoré sa v skladbe vyskytujú iba v minimálnej miere, teda z pohľadu improvizácie predstavujú zanedbateľnú informáciu.

Z uvedeného sa mi ako najvhodnejšie riešenie improvizačného systému javí použitie princípu kontrolovaných markovských reťazcov so substitúciou prechodových matíc za prediktívne stromy.

5.4 Výber MIDI syntetizačného nástroja

V neposlednom rade je potreba venovať pozornosť aj výberu vhodného syntetizačného nástroja, keďže výstupom improvizačného systému má byť štandardná MIDI nahrávka. Mnohé programovacie jazyky a prostredia ponúkajú knižnice na generovanie priameho MIDI výstupu cez vstavaný syntetizátor na reproduktory, alebo aj priamo do formátu SMF. Alternatívou je použitie programu T2mf na generovanie MIDI súboru v SMF formáte podľa textového súboru podľa zadanej syntaxe. Jeho hlavnou výhodou oproti knižniciam je jednoduchá upravovateľnosť vstupu a nezávislosť na použitom programovacom jazyku.

6 Návrh

V tejto kapitole chcem plynule nadviazať na výsledky analýzy a následne sa venovať podrobnému návrhu jednotlivých častí improvizačného systému. Podľa záverov analýzy by ho bolo možné v tejto chvíli rozdeliť do troch hlavných časti – analyzátor, improvizátor a syntetizátor. Analyzátor extrahuje potrebné informácie z MIDI skladby, improvizátor vytvára nové hudobné útvary a syntetizátor generuje novú MIDI nahrávku. Tieto časti tvoria navzájom poprepájané logické celky (viď Obrázok 11). V nasledujúcich kapitolách uvediem podrobný návrh funkcionality každého z celkov.

Obrázok 11: Štruktúra improvizačného systému

6.1 Analyzátor

Prvou logickou časťou improvizačného systému je analyzátor. Keďže zoznam požadovaných funkcií, ktoré musí analyzátor poskytovať som vymenoval už v kapitole 5.2 (Výber MIDI analyzačného nástroja), vyhnem sa ich opätovnému vymenovaniu a zameriam sa rovno na návrh mechanizmu jednotlivých funkcií.

- **Priame spracovanie MIDI nahrávky vo formáte SMF**

  Napriek tomu, že pre rozličné programovacie jazyky a prostredia existujú rôzne knižnice na extrakciu binárnej MIDI informácie, ako najvhodnejšia alternatíva sa javí použitie nástroja Mf2t s následným presmerovaním textového výstupu program na vstup vlastnej aplikácie. Následne bude potrebné spracovanie textovej informácie, čo sa ľahko dosiahnuť napríklad pomocou regulárnych výrazov.

- **Segmentácia nahrávky na logické celky**

  Ako dve základné alternatívy segmentácie nahrávky sa javí delenie na frázy a melodické úseky alebo delenie na takty. V prípade živého vstupu z externého MIDI syntetizátora by bolo delenie nahrávky na takty obtiažne, avšak podľa špecifikácie SMF formátu je informácia o dĺžke taktu uložená v hlavičkovom zväzku. Teda kým pri živom vstupe by bolo vhodnejšie deliť skladbu na melodické úseky, v prípade SMF súboru sa javí ľahšie a presnejšie delenie

na takty. Túto funkciu ponúka aj nástroj Mf2t, avšak prevod do taktov podľa exaktnej informácie je záležitosť pomerne jednoduchá a implementačne nenáročná.

- **Určenie rytmu a tempa skladby**

  Kým určenie rytmu a tempa skladby v prípade živého MIDI vstupu by bolo nesporne návrhovo a implementačne náročné, táto záležitosť je v rámci SMF formátu jednoducho riešiteľná tak ako v predchádzajúcom bode načítaním príslušnej informácie nachádzajúcej sa v hlavičkovom zväzku. Ak sa táto informácia nenachádza v súbore, použijú sa predvolené hodnoty, ktoré sú oficiálne určené v rámci formátu SMF.

- **Extrakcia melódie a oddelenie od harmonického a rytmického doprovodu**

  Podľa špecifikácie MIDI nemôžu hrať dva nástroje zároveň v jednom MIDI kanáli, teda pre každý nástroj musí byť vyhradený vlastný kanál. Preto vo veľkej väčšine prípadov netreba uvažovať polyfóniu v rámci jednej stopy (okrem niektorých čisto klavírnych skladieb). Ako ostačujúce sa teda javí zamerať sa na odlíšenie melodickej stopy od harmonických a rytmických stôp.

  V rámci špecifikácie zadania som sa rozhodol pracovať s MIDI nahrávkami reálnych piesní, teda budem uvažovať, že skladba môže obsahovať 4 základné druhy stôp – melodická, harmonická, basová stopa a stopa s perkusiami.

  Podľa MIDI špecifikácie je pre stopu s perkusiami vyhradený kanál číslo 10, preto ak stopa obsahuje noty s týmto kanálom, vyplýva z toho, že sa jedná o stopu s perkusiami.

  Basovú linku možno odlíšiť jednoducho tak, že basové tóny sú o bežne o oktávu nižšie ako melodické tóny, preto ak stopa obsahuje vysoký pomer veľmi nízkych tónov voči všetkým tónom v stope, jedná sa s veľkou pravdepodobnosťou o basovú stopu.

  Pri extrakcii harmonických a melodických stôp sa javí vhodné určovanie tzv. zhlukov nôt. Pod pojmom zhluk nôt rozumiem skupinu prekrývajúcich sa nôt, pričom takéto prekrývanie je indikátorom akordickej hry, teda poukazuje na harmonickú stopu. Ak sa takéto zhluky v stope nenachádzajú, alebo nachádzajú sa len do hodnoty explicitne určeného prahu (threshold), jedná sa s najväčšou pravdepodobnosťou o stopu melodickú.

- **Určenie tóniny skladby**

  Ako som už spomínal v kapitole 3.1 (Protokol MIDI a formát MIDI), jednou z metaudalostí v SMF formáte je aj nastavenie značenie tóniny. Avšak tento parameter je voliteľný, pretože pri prehrávaní MIDI táto informácia nehrá žiadnu úlohu, keďže tóny sú v rámci MIDI značené absolútne ako číslo od referenčného tónu C0 (zastúpený číslom 0). Preto sa niekedy stáva, že táto informácia sa v súbore nenachádza. V tomto prípade prichádza

do úvahy použitie postupu, pri ktorom sa určí tónina prvého alebo posledného taktu. Keďže určovanie tóniny nie je hlavnou činnosťou improvizačného systému a prípadné nesprávne určenie neohrozuje správnosť jeho činnosti, keďže sa improvizácia deje v rámci jedinej tóniny, načrtnutý postup považujem za dostačujúci.

- **Určenie harmonickej funkcie daného úseku**

Ako bolo vyššie spomenuté, navrhovaný improvizačný systém delí skladbu na takty, preto sa budem zaoberať mechanizmom určovania harmonickej funkcie pre takty.

Pri určovaní harmónie v takte sa vo všeobecnosti možno vybrať dvoma cestami. Prvou cestou je harmonizácia melódie, avšak jedná sa o veľmi komplikovaný postup vyžadujúci náročné návrhové a implementačné riešenie. [14] Druhou alternatívou je určovanie harmonickej funkcie z doprovodných stôp sprevádzajúcich daný úsek melódie. Táto alternatíva vyzerá na prvý pohľad schodnejšie, preto sa jej budem v nasledujúciách riadkoch.

Pri návrhu tejto funkcie sa chcem inšpirovať vlastným myšlienkovým postupom, ktorý by som používal pri „ručnej“ analýze daného taktu nahrávky:

1. Vyberiem všetky tóny basovej stopy v danom takte a uložím do špeciálnej stopy. Basová linka hrá často základný tón tóniny v danom takte, prípadne jeho terciu alebo kvintu.
2. Vyberiem všetky tóny nachádzajúce sa v zhlukoch harmonických stôp v danom takte a taktiež ich uložím do tej istej špeciálnej stopy ako basové noty.
3. Pre každú basovú notu určím interval (v rozmedzí prima – oktáva) od danej basovej noty po každú z nôt v zhlukoch a početnosť každého intervalu uložím do špeciálneho vektoru pre danú basovú notu.
4. Podľa početnosti výskytu intervalov pre danú basovú notu určím váhu vektoru pre ten ktorý akord, pričom beriem do úvahy aj prípadne obraty akordu a mód akordu (dur / mol).
5. V danom takte spočítam váhu každého možného akordu a vyberiem ten s najväčšou váhou, teda ten, ktorý tvorí najviac nôt.
6. Ak je už určená tónina skladby, vypočítam harmonickú funkciu daného akordu voči tónine.

Tento postup sa použije pre všetky takty v rámci skladby. Ak sa nepodarí určiť harmonickú funkciu, takt sa špeciálne označí, čím improvizátor pri prechádzaní taktov dostane signál, že daný takt nemôže improvizovať podľa harmonických funkcií.

6.2 Improvizátor

Kľúčovou časťou celého improvizačného systému je časť zvaná improvizátor. Úlohou improvizátora je vytvárať nový hudobný materiál na základe informácií získaných pri vstupnej analýze nahrávky. Úlohu improvizátora možno zhrnúť do dvoch funkcií:

1. Výšková improvizácia – tj. improvizácia výšky nôt.
2. Dĺžková improvizácia – tj. improvizácia dĺžky nôt.

Ako som spomenul v kapitole o analýze, najvhodnejšou metódou na improvizáciu sa javí spojenie kontrolovaných markovských reťazcov so stromovými metódami.

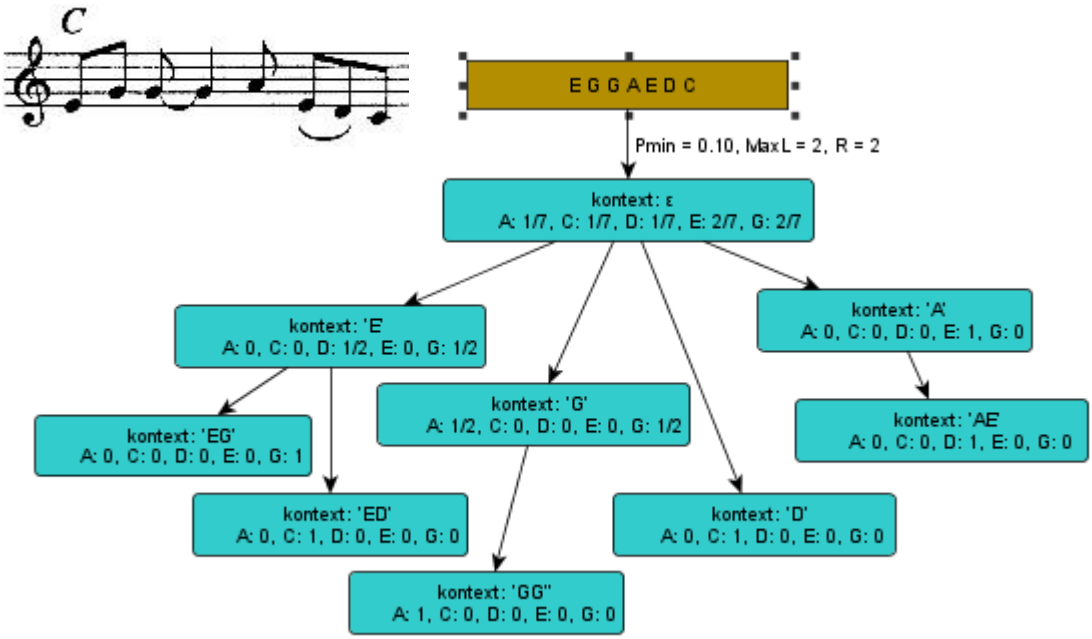
Princíp mnou navrhovaného systému spočíva vo využití informácií o harmónií v jednotlivých taktoch skladby na budovanie prediktívnych stromov špecifických pre tú ktorú harmonickú funkciu, čím chcem dosiahnuť už spomínanú improvizáciu na harmonickom základe pre každý takt zvlášť. Ako prediktívny strom budem používať dátovú štruktúru Prediction Suffix Tree (PST), ktorej princíp fungovania som podrobne opísal v kapitole 4.7.

Prvou fázou je „natrénovanie“ stromu podľa vstupnej sekvencie symbolov. Tu vyvstáva otázka, čo sa bude považovať za symbol v rámci skladby. Keďže mojim cieľom je improvizácia melódie, vstupná sekvencia bude určite vytvorená práve z melodickej stopy. Vyvstávajú dve hlavné alternatívy – buď sa za symbol bude považovať nejaká krátka hudobná fráza v rámci taktu melódie, alebo symbol bude reprezentovaný priamo parametrami noty (výška, dĺžka). Ako lepšia sa javí druhá varianta, pretože v rámci taktu sa logicky nachádza viac nôt ako fráz, čím sa dosiahne vyššia rozmaniťosť hudobného vstupu, od ktorej následne možno očakávať rozmanitejšie výstupy.

Ďalšou otázkou je otázka či výšku a dĺžku nôt improvizovať spolu alebo zvlášť. V prípade súčasnej improvizácie by vo vstupnej sekvencii figurovali dvojrozmerné symboly, teda každý symbol by mal podobu binárnej relácie medzi výškou a dĺžkou noty. Tento prístup taktiež podľa môjho názoru znižuje možnú rozmanitosť a variabilitu hudobného výstupu, preto sa prikláňam k variante dvojnásobnej improvizácie s jednoduchými symbolmi.

Ďalej je potrebné určenie parametrov PST. Prvým parametrom je minimálna pravdepodobnosť, s akou sa musí daný motív vyskytovať v danej sekvencii, aby bol pridaný do stromu. Ďalším parametrom je maximálna dĺžka motívu v strome a posledným parametrom je minimálny činiteľ *R*, tj. podiel dvoch pravdepodobností *P1* a *P2*, pričom *P1* je pravdepodobnosť výskytu symbolu *S* za daným kontextom *K1* a *P2* je pravdepodobnosť výskytu rovnakého symbolu *S* za kontextom *K2*, pričom dĺžka kontextu *K1* je o jeden znak menšia ako kontext *K2*.

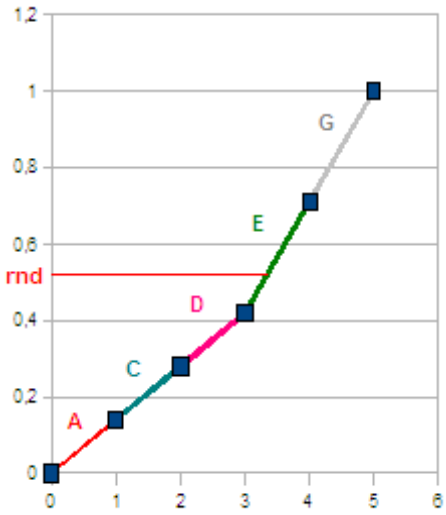
Následne môže začať fáza trénovania PST (Obrázok 12).



Obrázok 12: PST generovaný z danej postupnosti tónov

Po fáze trénovania je pre každú harmonickú funkciu v skladbe k dispozícii príslušný PST strom a môže dojsť k druhej fáze – generovanie nového hudobného materiálu. Vo fáze generovania sa použije mechanizmus opačný ako v prípade trénovania stromu, teda strom generuje novú sekvenciu symbolov, ktorá sa následne pretransformuje na sekvenciu nôt v takte.

Mechanizmus generovania nových sekvencií spočíva v prekonvertovaní stochastických pravdepodobností v jednotlivých uzloch PST na príslušné distribučné funkcie a následnom generovaní pseudonáhodných čísel v intervale 0 – 1 (viď Graf 1)



Graf 1: Príklad distribučnej funkcie s naznačeným princípom generovania nového symbolu podľa generátora pseudonáhodných čísel

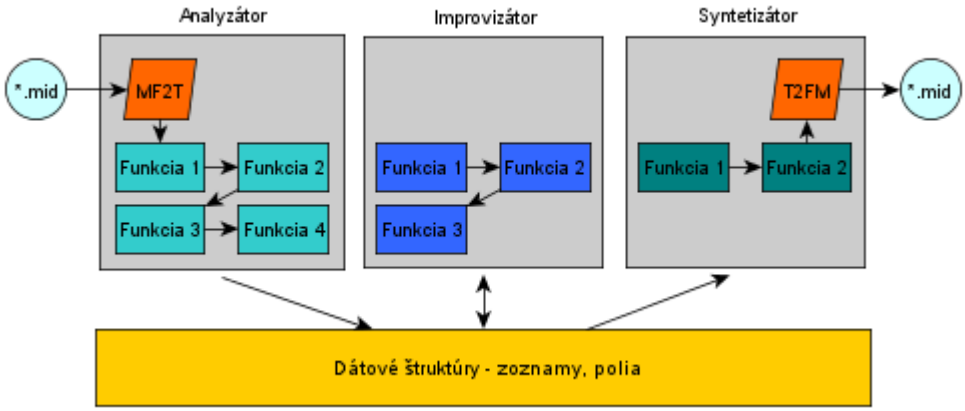
6.3 Syntetizátor

Poslednou súčasťou improvizačného systému je syntetizátor, ktorého úlohou je z dátových štruktúr vo vnútri aplikácie vysyntetizovať novú MIDI nahrávku vo formáte SMF. Jediný mne známy nástroj na tento účel je už spomínaný nástroj T2mf. Keďže vstupom T2mf je textový súbor s MIDI informáciami, pred samotnou konverziou je potrebné vytvoriť výpis z dátových štruktúr podľa požadovanej syntaxe.

6.4 Procedurálny vs. objektovo orientovaný prístup

V tejto podkapitole sa chcem zamerať na porovnanie možností oboch prístupov – procedurálneho a objektového pre potreby následnej implementácie improvizačného systému.

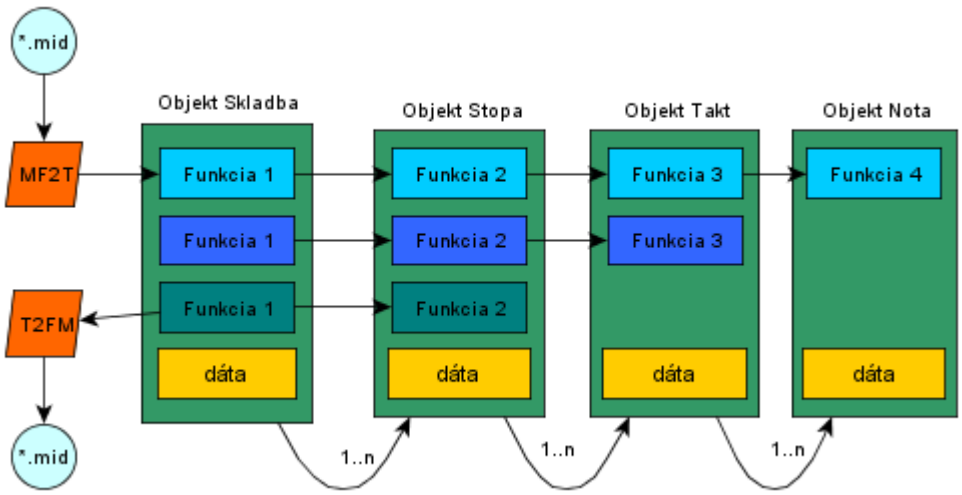
Procedurálny prístup spočíva v postupnom volaní procedúr (alebo funkcií) pracujúcich s dátovými štruktúrami, pričom tieto dátové štruktúry sú oddelené od tiel procedúr. V prostredí improvozačného systému by schéma procedurálneho prístupu mohla približne zodpovedať Obrázku 13. Výhodou tohto návrhu je logická štruktúra volania jednotlivých funkcií vykonávajúcich jednotlivé operácie v rámci daného logického celku. Avšak zároveň je potrebné zamyslieť sa nad vhodným výberom dátových štruktúr s ohľadom na predpokladanú činnosť improvizačného systému.



Obrázok 13: Schéma procedurálneho prístupu

Ako som naznačil v kapitole 3.1 (Protokol MIDI a formát MIDI), štruktúra SMF súboru pripomína vnorené polia – skladba je rozdelená do jednotlivých zväzkov reprezentujúcich stopy, pričom v každej stope je následne časovo zoradený zoznam jednotlivých MIDI udalostí tak ako nasledujú za sebou. Takýto zápis hudby pomocou udalostí je síce výhodný pre potreby nahrávania

a následného prehrávania, avšak nevyhovuje pre potreby analýzy obsahu nahrávky. V reálnom svete sa s takýmto prístupom môžeme stretnúť napr. v hudobných automatoch (napr. zvonkohry). Treba si však uvedomiť že, skladatelia ani hudobníci nepracujú so zápisom hudby v podobe zozname udalostí typu zapni notu, vypni notu, tak ako je tomu v prípade MIDI. V ľudskom svete sa hudba zapisuje pomocou nôt tvoriacich takty, pomocou taktov tvoríacich party (v rámci MIDI formátu budem používať pojem stopy) a partov tvoríacich celú skladbu. Preto vyvstáva myšlienka konverzie zoznamu udalostí na skupinu hiearchizovaných objektov. Takýto prístup je následne už iba krok od objektovo orientovaného návrhu. Po vytvorení tried reprezentujúcich šablóny pre jednotlivé objekty sa funkcie z procedurálneho návrhu vhodne rozdelia do príslušných metód realizujúcich operácie nad objektami, čím dôjde ku sprehľadneniu dátových typov, čo by malo priniesť zjednodušenie implementačnej náročnosti systému. Tento prístup sa preto javí vhodnejší ako procedurálny (Obrázok 14).



Obrázok 14: Schéma objektovo orientovaného prístupu

6.5 Návrh tried

Ako som v predchádzajúcej kapitole naznačil, objektový prístup sa z implementačného hľadiska javí vhodnejší ako klasický procedurálny, preto sa chcem v krátkosti venovať návrhu tried improvizačného systému.

Zo štruktúry MIDI súboru a taktiež aj z klasického chápania hudobného zápisu možno povedať, že štruktúra hudobnej skladby je vždy do istej miery hierarchizovaná. Noty tvoria takty, takty tvoria party, party (alebo stopy) tvoria pieseň. Preto aj triedy reprezentujúce hudobnú skladbu by mali byť hierchizované podobným štýlom. Následne bude potrebné rozdeliť funkcie z procedurálneho návrhu do príslušných metód pre každú z tried.

Analogicky teda možno usúdiť, že hlavnou triedou by mala byť trieda *Skladba* s príslušnými metódami. Analyzačnými metódami by mali metódy na načítanie textového vstupu reprezentujúceho skladbu, ďalej nastavenie atribútov skladby - rytmus, tempo a zistenie tóniny. Ďalej by mala obsahovať metódu na pridanie nových stôp. Samozrejme by mala obsahovať aj metódy na improvizáciu výšky a dĺžky a na generovanie textového výstupu, ktorý sa predloží na konverziu programu T2fm.

Hierarchicky nižšou triedou by mala byť trieda *Stopa.* Hlavnými metódami tejto triedy by malo byť nastavenie hudobného nástroja, a ostatných parametrov stopy, pridávanie nových taktov a určovanie typu stopy, teda či ide o stopu melodickú, harmonickú, basovú, alebo stopu s perkusiami.

Ďalšou triedou by mala byť trieda *Takt*. Metódami tejto triedy by malo byť pridávanie nôt do taktu a určovanie harmónie v takte, ktorá podľa návrhu predpokladá metódu na určovanie zhlukov nôt.

Hierarchicky najnižšou triedou bude trieda *Nota.* V objektoch tejto triedy sa budú zhromažďovať prakticky všetky informácie zo skladby ako výška nôt, dĺžka nôt a dynamika nôt. Metódou tejto triedy by malo byť určenie či ide o notu basovú alebo výškovú a ďalej by mala obsahovať metódu na určenie intervalov, v ktorých nota figuruje.

6.6 Návrh testov

Keďže výstupom improvizačného systému má byť vysyntetizovaná hudobná nahrávka, je potrebné sa zamyslieť nad návrhom vhodných testov, ktoré by preukázali funkčnosť systému a zároveň by poukázali na závislosti medzi vstupnými parametrami systému (parametre PST) a finálnou nahrávkou. Okrem týchto exaktných testov je však veľmi dôležitá aj celková „počúvateľnosť“ a inovatívnosť novej nahrávky voči originálu, avšak na takéto testy prirodzene chýba metrika, preto

bude potrebné (hlavne prvotných fázach) sa spoľahnúť predovšetkým na vlastný sluch a hudobný vkus.

Ako som už spomínal v kapitole o analýze, veľmi dobré prostriedky na štatistickú analýzu MIDI nahrávky poskytuje nástroj Midi Toolbox. Preto ako hlavné testy navrhujem využitie niektorých jeho štatistických funkcií – pravdepodobnostné rozloženie tónov a intervalov a vykreslenie prechodovej matice. Zaujímavou funkciou je taktiež vykreslenie tvaru melódie a štatistické rozloženie dĺžok tónov v skladbe.

Ďalšou možnosťou je vykreslenie notovej osnovy MIDI nahrávky pomocou aplikácie LilyPond.

7 Implementácia

7.1 Výber programovacieho jazyka

Hlavnými kritériami, ktorými som sa riadil pri výbere programovacieho jazyka boli:

- jednoduchosť syntaxe
- prepracovaná funkcionalita, rozvinuté dátové štruktúry
- dobrá práca s texotvými súbormi, podpora regulárnych výrazov
- podpora objektovo orientovaného programovania

Po dôkladnom zvážení som sa priklonil k skriptovacím jazykom, a ako najvhodnejší kandidát sa mi javil jazyk Python. Vyniká jednoduchosťou syntaxe, zároveň poskytuje dobrú prácu s dátovými štuktúrami (zoznamy, slovníky), ľahko pracuje s textovými súbormi, podporuje regulárne výrazy a umožňuje objektovo orientované programovanie.

7.2 Výber platformy

Napriek tomu, že interpret jazyka Python je dostupný prakticky pre všetky platformy, mnou navrhovaný improvizačný systém používa dvojicu externých programov Mf2t a T2fm, ktoré pracujú na platforme Windows. Preto som sa pri implementácií zameral iba na toto prostredie.

7.3 Implementácia tried, metód a funkcií

Keďže návrhu jednotlivých mechanizmov tvoriacich improvizačný systém som sa obšírnejšie venoval v kapitole o návrhu, v tejto kapitole sa pokúsim len stručne popísať jednotlivé triedy s ich metódami a popísať implementáciu niektorých externých funkcií na celkové dokreslenie funkčnosti systému. Okrem navrhovaných tried sa počas implementácia ukázalo vhodné vytvoriť aj niektoré nové, o ktorých bude zmienka v nasledujúcich podkapitolách.

7.3.1 Trieda TSong

Hlavnou triedou improvizačného systému je trieda `TSong` reprezentujúca celú skladbu. Trieda je napojená na externú funkciu `ParseMidiInput`, ktorá pomocou regulárnych výrazov spracováva textový výstup programu Mf2t. Pomocou skupiny metód `SetTicksPerBeat`, `SetKey`, `SetTimeSig`, `SetTempo`, `SetSmpte`, `SetSysex` sa nastavia príslušné atribúty kopírujúce atribúty MIDI súboru.

Následne sa pri priebežnom načítavaní súboru jednotlivé stopy pridajú do zoznamu stôp pomocou metódy `AddTrack`. Keďže sa často stáva, že niektoré stopy neobsahujú žiadnu informáciu o hudbe, tieto stopy sa odstránia metódou `RemoveRedundatTracks`. Po načítaní súboru sa pomocou skupíny metód `GetMelodyTracks`, `GetBassTracks`, `GetPercussionTracks` určia stopy s melódiou a harmóniou, basové stopy a stopy s perkusiami. Keďže sa niekedy stáva, že niektoré melodické stopy sú zdvojené, je dobré tento jav detekovať. Na detekciu slúži funkcia `GetSimilarTracks`. Nakoniec sa zavolá metóda `ImprovisePitch` na improvizáciu výšky a metóda `ImproviseLength` na improvizáciu dĺžky. Obe tieto metódy volajú externú funkciu `PST` na generovanie prediktívneho stromu a následne funkciu `Generate` na generovanie novej sekvencie z neho. Obe metódy majú niekoľko parametrov nastaviteľných v konfiguračnom súbore `config.txt` a ich mechanizmus som podrobne popísal v kapitole o návrhu. Po úspešnom priebehu improvizácie sa dátové štruktúry predajú externej funkcii `OutputMidi` na vytvorenie textového výstupu.

7.3.2 Trieda TTrack

Trieda `TTrack` reprezentuje stopu v MIDI skladbe. Na začiatku sa nastavia niektoré parametre stopy a informácia o nástroji, ktorý bude hrať v danej stope. Následne sa pomocou metód `AddNote` pridávajú do stopy nové noty. Zároveň s pridávaním nôt sa pridávajú aj príslušné takty pomocou metódy `AddBar`.

7.3.3 Trieda TBar

Trieda `TBar` reprezentuje takt v stope MIDI skladby. Po naplenení taktu notami sa zavolá metóda `Clusterize` na určenie zhlukov nôt. Následne sa tieto zhluky zo všetkých taktov všetkých stôp nakopírujú do špeciálnej stopy na určovanie harmónie a pomocou metódy `GetChords` sa určí harmonická funkcia (akord) v danom takte. Mechanizmus určovania harmónie som podrobne popísal v kapitole o návrhu.

7.3.4 Trieda TNote

Trieda `TNote` reprezentuje notu v rámci taktu MIDI skladby. Keďže je každá nota v MIDI formáte zapísaná v tvare dvoch časovo oddelených udalostí (zapnúť notu, vypnúť notu), vytvorenie noty súvisí s dvoma metódami. Pri udalosti „zapni notu“ sa zavolá konštruktor `__init__` (s príslušnymi argumentami ako výška noty, kanál noty, hlasitosť noty) a pri vypnutí noty sa zavolá metóda `endNote`, ktorou sa nota „ukončí“ a zároveň sa vypočíta sa jej dĺžka. Následne sa vypočítajú všetky intervaly (reprezentované triedou `TInterval`) , v ktorých nota v danom takte figuruje a tieto intervaly sa pripoja do špeciálneho zoznamu v rámci noty.

7.3.5 Trieda TInterval

Významom triedy `TInterval` je zjednodušenie práce pri hľadaní harmónie v taktoch. Trieda má jedinú metódu a tou je konštruktor `__init__` na vytvorenie nového objektu. Ako už bolo spomenuté, informácia o intervaloch sa pripája do objektov triedy `TNote`.

7.3.6 Trieda TChord

Podobne ako pri triede `TInterval`, aj trieda `TChord` má pomocný význam pri určovaní harmónie v rámci taktu. Pre jeden takt sa určí viacero potenciálnych akordov a podľa počtu nôt tvoriacich akord sa vyberie akord s najväčšou váhou a prehlási sa za harmonický akord daného taktu.

7.3.7 Trieda TCluster

Trieda `TCluster` je taktiež pomocná trieda, na určovanie zhlukov nôt v takte. Ak sa noty v takte prekrývajú, zavolaním metódy `Clusterize` sa vytvorí nový objekt `TCluster` a pomocou metódy `AddNote` sa noty do zhluku pripoja.

7.3.8 Trieda TNode

Keďže jazyk Python neobsahuje predvolené metódy a štruktúry na prácu so stromami, vytvoril som si vlastnú triedu reprezentujúcu uzol v n-árnom strome. Trieda okrem konštruktora disponuje dvoma metódami, prvou je metóda `AddValue` na pridávanie hodnoty do uzlu stromu (hodnoty sa ukladajú do slovníka) a druhou je metóda `AddChild`, pomocou ktorej sa do slovníka potomkov ukladajú odkazy na potomkov daného uzlu.

7.3.9 Funkcie ParseMidiInput a OutputMidi

Ako názov prvej funkcie napovedá, funkcia má za úlohu spracovanie textového vstupu z programu Mf2t. Prvým parametrom funkcie je reťazec reprezentujúci textový súbor a druhým je odkaz na hlavný objekt triedy `TSong`, do ktorého sa následne pri priebežnom spracovávaní súboru pomocou regulárnych výrazov cez metódy objektu pridávajú nové podobjekty (stopy, takty, noty) a príslušné dáta. Sémanticky opačnou funkciou je funkcia `OutputMidi` realizujúca textový výstup z už vytvorených objektov.

7.3.10 Funkcie PST a Generate

Opäť sa jedná o dve sémanticky opačne pracujúce funkcie. Kým funkcia `PST` má na starosti generovanie prediktívneho stromu pre vstupnú sekvenciu symbolov, funkcia Generate sa stará o generovanie novej sekvencie symbolov. Funkcia `PST` pri vytváraní stromu využíva rekurzívny priechod stromom, čo značne zjednodušuje implementačnú náročnosť algoritmu (viď Príloha 2). Vstupom funkcie sú jednotlivé parametre *Pmin, MaxLength, R* (podrobne sú popísané v kapitole o návrhu) a výstupom funkcie je strom reprezentovaný koreňovým uzlom.

Pred samotným zavolaním funkcie `Generate` sa hodnoty v uzloch stromu reprezentujúce stochastické pravdepodobnosti symbolu pre daný kontext prekonvertujú pomocou funckie `TreeToDict` na slovník, ktorý sa následne predloží ako parameter generujúcej funkcii `Generate`.

8 Testy

Ako som už naznačil v kapitole o návrhu testov, hudba je špecifické médium, pre ktoré síce existujú metriky na zhodnotenie kvantity, avšak chýbajú exaktné metriky na zhodnotenie jej kvality. Hudba vyvoláva v človeku emocionálnu odozvu, a preto sa jej kvalita hodnotí subjektívne hlavne podľa tohto faktora. Keďže je však subjektívny, je zároveň aj ťažko merateľný, a preto ním budem zaoberať až pri záverečných hodnotiacich úvahách.

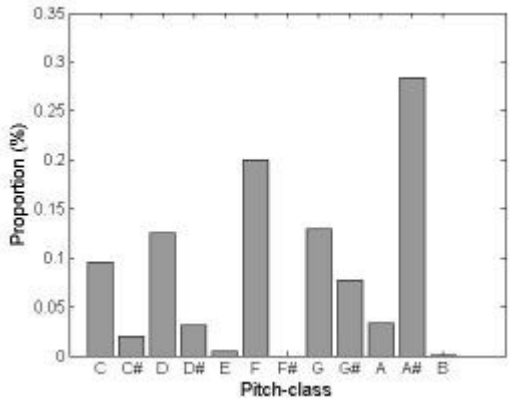
Z pohľadu improvizácie podľa vstupnej nahrávky sa ako dobrý test črtá určovanie miery podobnosti medzi melódiu originálnej a improvizovanej nahrávky. Ďalším vhodným testom je určovanie závislosti vzájomnej podobnosti nahrávok od parametrov prediktívneho stromu. Testovacím prostredím je spomínaná knižnica Midi Toolbox. Nevýhodou Toolboxu je, že nedokáže pracovať s polyfonickými skladbami v zmysle viacerých stôp v súbore a taktiež ani s polyfonickými stopami v zmysle prekrývajúcich sa nôt, preto je potrebné v prvej fáze rozdeliť originálnu aj improvizovanú nahrávku na niekoľko nahrávok s jedinou stopou a zároveň eliminovať prekrývanie nôt. Takto upravené nahrávky možno následne pomocou Toolboxu načítať a realizovať nad nimi potrebné testy.

Testovacou skladbou bola jazzová skladba „Mercy, Mercy, Mercy“ (dostupná na priloženom CD, notový zápis je v Prílohe 4) od skladateľa menom Joe Zawinul, ktorá bola následne predložená improvizačnému sytému. Keďže sa jedná o polyfonickú skladbu, pre účely testovania som z nej vybral iba stopu hrajúcu hlavnú melódiu na vibrafóne.

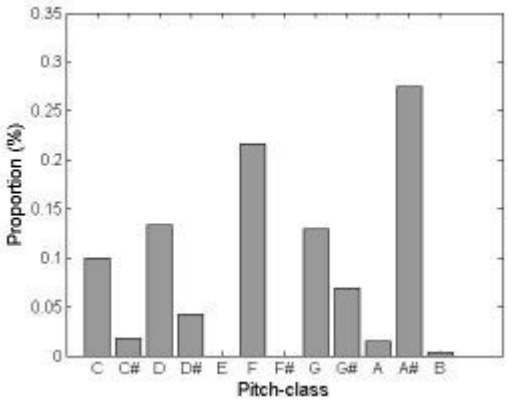
Prvá séria testov prebieha pri nasledujúcich parametroch:

PITCH_LOCAL=0 // výška nôt sa improvizuje z celej skladby podľa harmonických funkcií
PITCH_SAME=0 // určuje koľko počiatočných tónov má mať rovnakú výšku ako v origináli
PST_PITCH_L=2 // maximálna dĺžka frázy v PST strome pre výšku nôt
PST_PITCH_PMIN=0.2 // minimalna pravdepodobnosť frázy pre výšku nôt
PST_PITCH_R=2 // násobiaci koeficient R pre výšku nôt
LENGTH_LOCAL=1 // dĺžka nôt sa bude improvizovať len z daného taktu
LENGTH_SAME=2 // koľko počiatočných tónov má mať rovnakú dĺžku ako v origináli
PST_LENGTH_L=1 // maximálna dĺžka frázy v PST strome pre dĺžku nôt
PST_LENGTH_PMIN=0.0 // minimalna pravdepodobnosť frázy pre dĺžku nôt
PST_LENGTH_R=0 // násobiaci koeficient R pre dĺžku nôt

Prvým testom je test na štatistické rozloženie tónov v oboch skladbách:



Graf 2: Originálna nahrávka

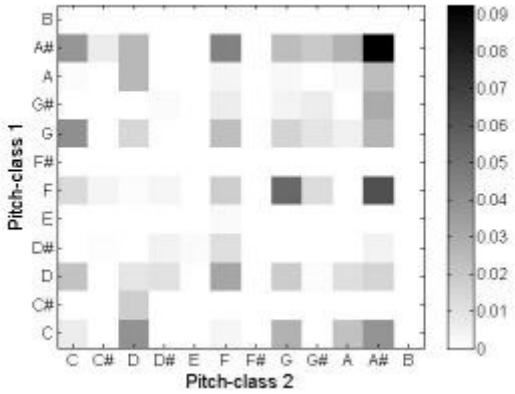


Graf 3: Improvizácia

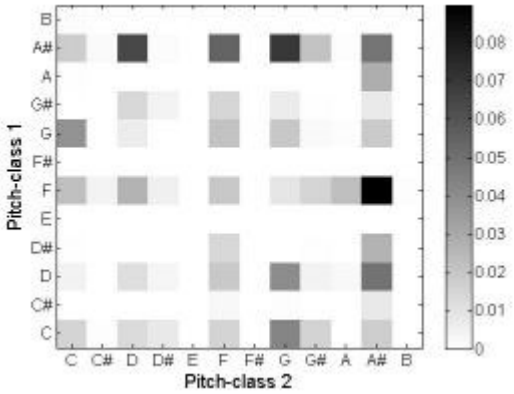
Ako možno vidieť z grafov, rozloženie tónov sa takmer nezmenilo, defacto sa teda použili tie isté tóny. Tento záver sa dal očakávať, keďže prediktívny strom funguje na stochastických pravdepodobnostiach natrénovaných z pôvodnej skladby.

Jednou z funkcií Toolboxu je taktiež určenie miery podobnosti medzi dvoma skladbami vzľadom na daný ukazateľ v rozsahu (0, 1), pričom číslo 1 reprezentuje zhodnosť skladieb. V tomto prípade vyšiel koeficient rovný 0.9574.

Ďalším testom je testovanie prechodovej matice tónov, teda matice pravdepodobností, že za aktuálnou notou bude nasledovať iná nota (Graf 4, Graf 5). Ako možno z grafov vidieť, v tomto prípade je už miera podobnosti medzi oboma nahrávkami nižšia, aj keď stále nie zanedbateľná. Miera podobnosti bola Toolboxom vyčíslená na 0.6177.



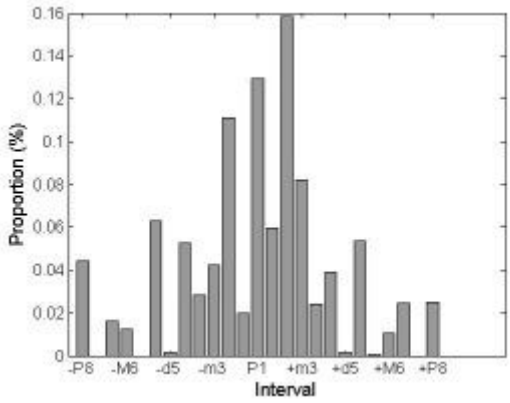
Graf 4: Originálna nahrávka



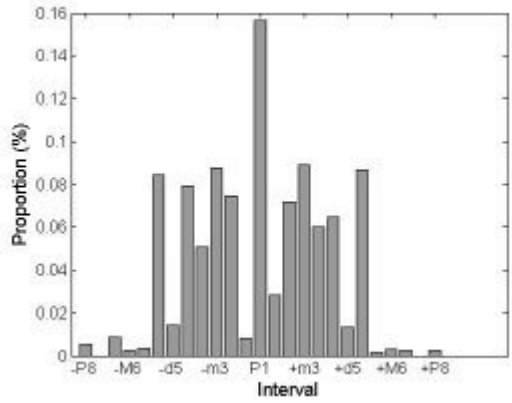
Graf 5: Improvizácia

Ďalším testom je test na štatistické rozloženie intervalov v skladbe (Graf 5, Graf 7). Ako možno z grafov vidieť, kým v originálnej nahrávke je najčastejším intervalom sekunda, v improvizácií je to

prima. Ďalej možno vidieť, že aj keď sú intervaly v oboch skladbách prakticky rovnaké, graf originálu pripomína normálne rozloženie intervalov, kým graf improvizácie viac pripomína rovnomerné rozloženie, čo je logické, pretože podľa vstupných parametrov bola výšková improvizácia globálna (pre danú harmonickú funkciu boli dáta zozbierané zo všetkých taktov s rovnakou harmonickou funkciou), takže pri trénovaní PST stromu došlo k akémusi „spriemerovaniu“ tónov v skladbe. Miera podobnosti je podľa Toolboxu 0.7261.

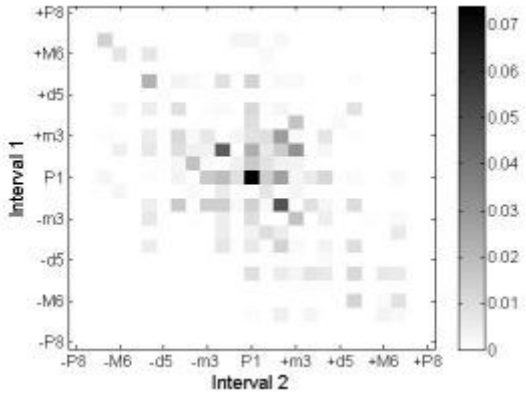


Graf 6: Originálna nahrávka

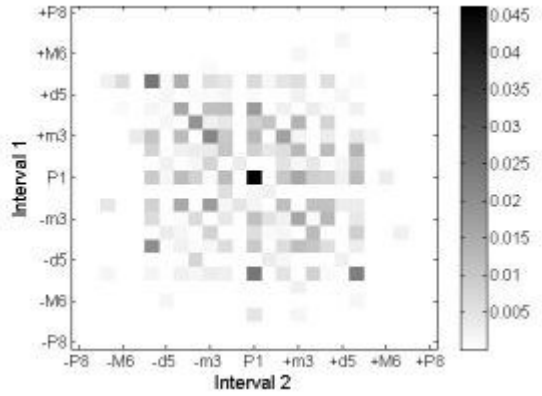


Graf 7: Improvizácia

Ďalším testom je podobne ako v prípade tónov, prechodová matica intervalov (Graf 8, Graf 9). Analogicky ako v predchádzajúcom prípade, prechodová matica označuje pravdopodobnosti prechodu z aktuálneho intervalu na iný a ako možno vidieť obe matice sa značne líšia. Kým prechodová matica originálu pripomína diagonálu, prechodová matica improvizácie skôr pripomína štvorec. Táto vlastnosť podľa môjho úsudku súvisí hlavne s vyššie uvedeným štatistickým rozložením intervalov. Miera podobnosti je 0.4598.



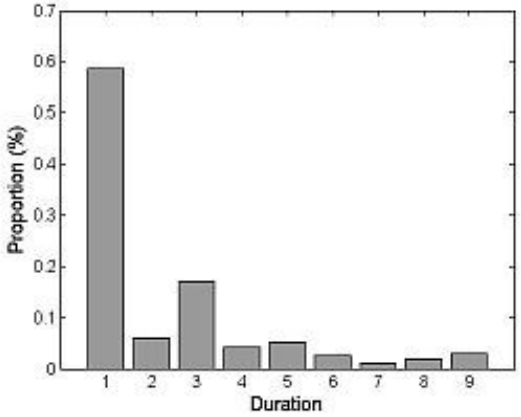
Graf 8 Originálna nahrávka



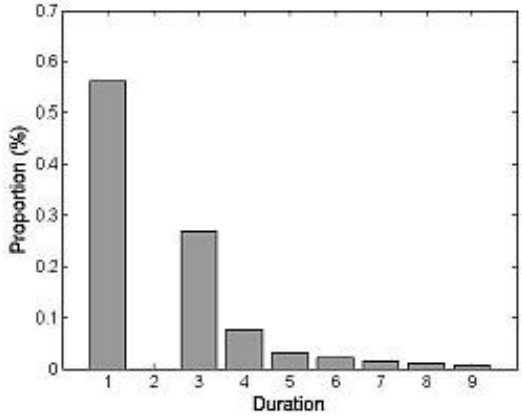
Graf 9: Improvizácia

Ďalším testom je test na tvar a podobnosť melódie (Graf 10, Graf 11). Ako možno vidieť, graf improvizácie je podobný originálnej nahrávke, aj keď je o niečo „kostrbatejší“. V praxi to znamená, že melódia je o niečo náhodnejšia (ale zároveň rozmanitejšia a ozdobnejšia), čo je prirodzeným dôsledkom použitého stochastického prístupu. Miera podobnosti je podľa Toolboxu 0.8001.

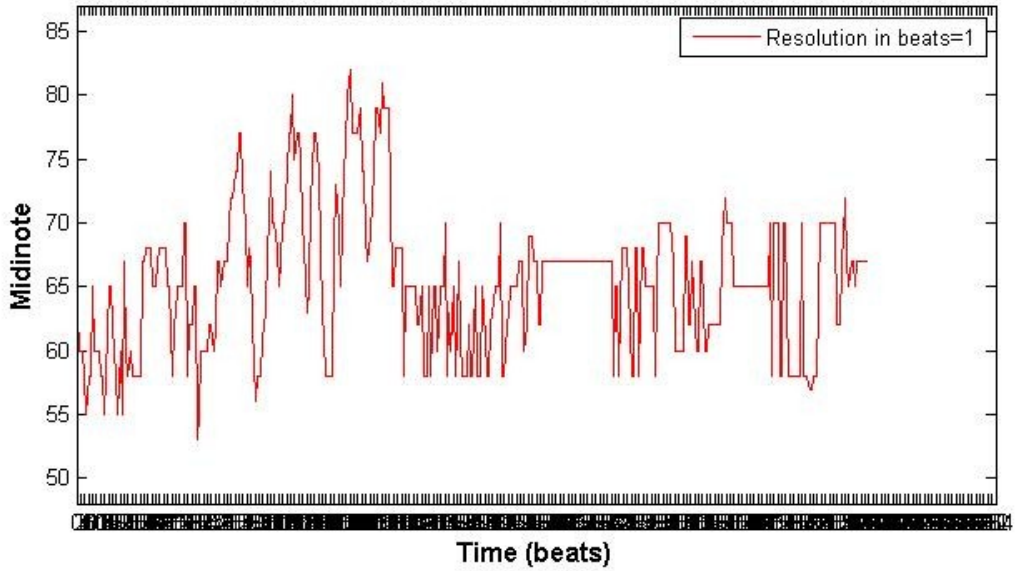
Posledný test sa týka dĺžky nôt (Graf 12, Graf 13). Ako možno vidieť, štatistické rozloženie jednotlivých dĺžok tónov je približne rovnaké. Miera podobnosti je 0.9654.



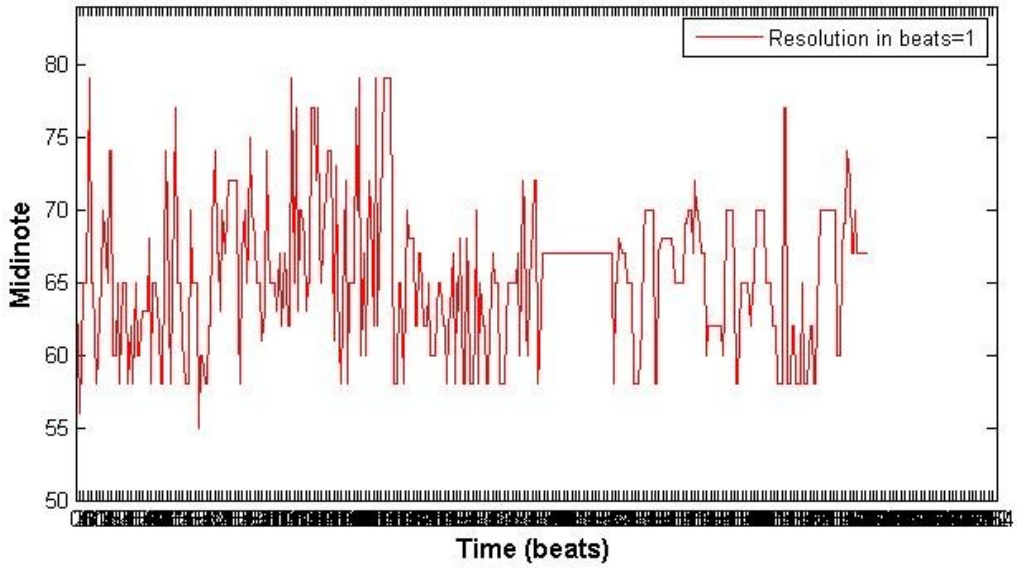
Graf 12: Originálna nahrávka



Graf 13: Improvizácia



Graf 10: Tvar melódie originálnej nahrávky



Graf 11: Tvar melódie improvizovanej nahrávky

Výsledky druhej a tretej série série testov, ktoré prebehli na rovnakej vstupnej nahrávke iba so zmenenými parametrami predkladám v Prílohe 3.

Celkovo možno zhrnúť, že testy v Toolboxe poukázali na vzájomné podobnosti i odlišnosti improvizovanej nahrávky vzhľadom k originálnej nahrávke, a zároveň preukázali vplyv parametrov *Pmin* a *MaxLength* na celkový ráz výstupnej improvizácie.

9 Záver

9.1 Vyhodnotenie systému

Systém bol od počiatočnej fázy koncipovaný tak, aby pracoval s čím väčšou škálou dostupných nahrávok, preto je potešujúci fakt, že v konečnej fáze systém pracuje s prakticky všetkými voľne dostupnými MIDI nahrávkami reprezentujúcich konkrétne piesne. Tieto nahrávky nepotrebujú žiadnu „predúpravu“ predtým než sa predložia na vstup systému, a tak jediné obmedzenie v rámci vstupu sa týka verzie MIDI formátu, pre ktorý bol systém koncipovaný. V tomto formáte je väčšina klasických MIDI nahrávok. Pozitívny je tiež fakt, že systém na vstupe nepožaduje žiadne zložité parametre — tieto sú užívateľovi dostupné v oddelenom konfiguračnom súbore.

Čo sa týka výstupov systému, zo subjektívneho hľadiska môžem povedať, že improvizované MIDI nahrávky vo finálnej fáze systému boli pre mňa príjemným prekvapením. Keďže výstup systému je veľmi závislý od vstupnej nahrávky, je veľmi ťažko súhrnne charakterizovať rôznorodé výstupy systému, avšak som presvedčený, že výstupy systému možno rozhodne považovať za „improvizáciu na danú tému“. Určite pomáha aj fakt, že nová nahrávka používa pôvodné harmónické stopy a stopy s perkusiami. Samozrejme, keďže sa jedná v istom zmysle o experimentálnu hudbu (avšak aj improvizácia je často do veľkej miery experimentom), hudobné výstupy občas obsahujú viac či menej disharmonické party, ktoré môžu odradiť neskúseného poslucháča očakávajúceho ladné melódie, na ktoré je zvyknutý z rádia pri počúvaní populárnej hudby. Pre skúsenejšieho poslucháča, ktorý má aké-také skúsenosti s jazzom, bluesom a celkovo s improvizovanou hudbou môžu byť však aj tieto „ťažšie“ party príjemným oživením skladby, a tak môžu dotvárať celkovú atmosféru špecifickú pre improvizovanú hudbu.

Na isté problémy som narazil v prípade skladieb, kde viacero nástrojov hrá zároveň rôzne melódie. Keďže jednotlivé stopy sú improvizované zvlášť a teda o sebe vzájomne „nevedia“, v niektorých častiach niektorých piesní dochádza k rušivej disharmónií spôsobenej nezávislou improvizáciou na sebe závislých stôp. S týmto nedostatkom však bolo treba počítať, keďže systém nepracuje so znalosťami potrebnými na odstránenie alebo harmonizáciu takýchto odchýliek.

Ako som už niekoľkokrát spomenul, keďže hudba je v prvom rade záležitosťou estetiky, jej kvalita sa hodnotí na základe subjektívnej odozvy v človeku. Takéto hodnotenie je úzko spojené s hudobným vkusom daného jedinca, ktorý je do veľkej miery ovplyvnený jeho hudobnými skúsenosťami a taktiež mnohými inými faktormi počnúc aktuálnou náladou. Preto zo svojej pozície nemôžem výstupy systému prehlásiť ani za „dobré“, ani za „zlé“, avšak dovolím si ich nazvať ako „zaujímavé“ a „hodné pozornosti“ a to nielen z pohľadu hudobníka, ale aj z pohľadu poslucháča.

9.2 Možné budúce rozšírenia

V budúcnosti by systém mohol byť vybaveným grafickým prostredím, v ktorom by bola možnosť vizualizácie vstupu a výstupu pomocou nôt, čo by taktiež prinieslo možnosť vyššej interaktivity pri práci s programom. Ako zaujímavá sa javí možnosť improvizácie zvolených časti skladby spojená s okamžitým prehratím improvizovanej časti, ktorá by sa dala viacnásobne meniť podľa preferencií užívateľa.

Oveľa väčšou výzvou by bola nadstavba programu pracujúca s MIDI vstupom v reálnom čase napríklad cez klávesový syntetizátor. Avšak takáto vymoženosť by vyžadovala oveľa väčšie úsilie pri spracovávaní a analýze nahrávky, keďže by neboli k dispozícií žiadne priame údaje týkajúce sa hudobného vstupu (ako je tomu pri MIDI nahrávkach). Takto navrhnutý program by sa natrénoval na živom vstupe a následne by po istej chvíli začal sám tvoriť hudbu v danom štýle.

9.3 Prínos systému

Hlavný prínos mnou vytvoreného systému vidím v dvoch jeho hlavných atribútoch. Prvým je schopnosti spracovať a analyzovať reálne, do veľkej miery „neideálne“ nahrávky, s ktorými si zväčša nevedeli poradiť ani renomonované analyzačné nástroje. Ďalej si na systéme cením jeho schopnosť tvoriť nový hudobný materiál nad existujúcim hudobným základom pri rešpektovaní pôvodnej harmónie skladby, pretože s touto myšlienkou som sa v rámci literatúry týkajúcej sa hudobnej kompozície a improvizácie ešte nestretol. Vytvorený improvizačný systém je dobre použiteľný v praxi, aj keď na jeho širšie nasadenie by boli potrebné niektoré úpravy, hlavne grafické užívateľské rozhranie, o ktorom som sa zmienil v predchádzajúcej kapitole.

Aj keď som pri vývoji systému nepoužil žiadne prevratné metódy a techniky, myslím si, že mnou vybrané existujúce techniky som použil novým spôsobom, a tak sa mi podarilo vytvoriť nový systém, pracujúci s hudbou originálnym a nevídaným spôsobom.

Literatúra

[1] Růžička, R.: *Využití samočinných počítačů při vzniku uměleckých děl se zvláštním zaměřením na hudbu a soudobou hudební kompozici* [online]. [cit. 24. 4. 2009]. Ediční středisko JAMU, Brno, 1980
Dostupné z URL: <http://www.fi.muni.cz/~qruzicka/Vyuziti/FI-vyuziti.HTM>

[2] Šindler, J.: *Kytarová praktika*. Muzikus, 2002

[3] Marom, Y.: *Improvising Jazz With Markov chains*, Disertation thesis, University of Western Australia, 1997

[4] Sleator, D., Temperley D.: *The Melisma Music Analyzer* [online]. [cit. 27. 4. 2009].
Dostupné z URL: <http://www.link.cs.cmu.edu/music-analysis/>

[5] Eerola, T., Toiviainen, P.: *MIDI Toolbox: MATLAB Tools for Music Research*. University of Jyväskylä: Kopijyvä, Jyväskylä, 2004

[6] Eerola, T., Toiviainen, P.: *Mir In Matlab: The Midi Toolbox* [online], In: ISMIR 2004
Dostupné z URL: <http://ismir2004.ismir.net/proceedings/p004-page-22-paper193.pdf>

[7] Järveläinen, H.: *Algorithmic Musical Composition*, Helsinki University of Technology, 2000
Dostupné z URL: <http://www.tml.tkk.fi/Studies/Tik-111.080/2000/papers/hanna/alco.pdf>

[8] McAlpine, K., Miranda E., Hoggar S.: *Making music with algorithms: A case-study system*. Computer Music Journal, 23(2):19–30, 1999.

[9] Voss, R. F., Clarke J.: *1/f noise in music and speech*,Nature 258 (1975), 317- 318.

[10] Temperley, D., Sleator D.: *Melisma Stochastic Melody Generator* [online]. [cit. 28.4.2009].
Dostupné z URL: <http://www.link.cs.cmu.edu/melody-generator/>

[11] Růžička, R.: *CCOMP - počítačový program pro kompozici instrumentálních, vokálních a elektroakustických skladeb z oblasti soudobé vážné hudby, jejich automatickou notaci a zvukovou realizaci* [online]. [cit. 28.4.2009].
Dostupné z URL: <http://www.fi.muni.cz/~qruzicka/Ccomp-ce.htm>

[12] Brooks F.: *An Experiment in Musical Composition Machine Models of Music*, MIT Press, 1993 pp. 23-40

[13] Dubnov, S., Assayag G., Lartillot O., Bejerano G..: *Using Machine-Learning Methods for Musical Style Modeling,* IEEE Society, August, 2003

[14] Maňák, O.: *Harmonizace melodie*. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Zoznam príloh

Príloha 1. Návod na použitie improvizačného systému
Príloha 2. Vybrané zdrojové texty
Príloha 3. Ďalšie testy
Príloha 4. Notové záznamy
Príloha 5. CD nosič

Obsah CD nosiča:

- inštalačný súbor interpretu jazyka Python 2.6
- externé programy Mf2t a T2fm
- zdrojové texty improvizačného systému
- originálne a improvizované MIDI nahrávky
- technická správá vo formáte odt a pdf

Príloha 1

Návod na použite improvizačného systému

Aplikácia je vytvorená v jazyku Python 2.6 pre platformu Windows, preto je ako prvý krok potrebné mať nainštalovaný interpret tohto jazyka. Aplikácia bola odladená pre interpret vo verzii 2.4.2.

Po inštalácií interpreta je potrebné pridať si príslušný adresár kam bol interpret nainštalovaný do zoznamu ciest, napríklad v príkazovom riadku príkazom „`set path=%path%;C:\python26`“. Následne je možné pristupovať k súborom s koncovkou .py ako k štandardným spustiteľným súborom. Aplikácia sa spúšťa príkazom impro.py s parametrami v nasledujúcom tvare:

```
impro.py [-p] [-t] infile.mid [outfile.mid]
```

Prepínač `-p` značí improvizáciu v rámci výšky nôt a prepínač `-t` značí improvizáciu v rámci dĺžok nôt. Oba prepínače sú voliteľné. Ak sa ani jeden z nich neuvedie, aplikácia len načíta vstupnú nahrávku a prekopíruje jej obsah do výstupného súboru. Ak sa neuvedie názov výstupného súboru, názov výstupného súboru sa automaticky odvodí z názvu vstupného súboru pripojením predpony „n_“. Z dôvodu nekompatibility programov Mf2t a T2fm so súbormi s dlhými názvami, dĺžka názvu vstupných aj výstupnych súborov je obmedzená na 8 znakov.

Neoddeliteľnou súčasťou improvizačného systému je konfiguračný súbor `config.txt` s parametrami pre tvorbu prediktívnych stromov ovplyvňujúcich výslednú improvizáciu. Jednotlivé parametre aj s popisom ich funkcionality sú uvedené v tomto súbore.

Príloha 2

Vybrané zdrojové texty

Algoritmus tvorby PST

```
def PST(parent_node, in_sequence, p_min, max_symbol_len, r):

  for char in set(in_sequence):
    a = in_sequence.count(parent_node.key + char)
    b = in_sequence.count(parent_node.key, 0, - len(char))
    if b > 0:
      cond_prob = float(a) / b # vypocet „conditional probability“
    else:
      cond_prob = 0

    parent_node.addValue(char, cond_prob)

  for char in set(in_sequence):
    symbol = parent_node.key + char
    if len(symbol) <= max_symbol_len:
      if symbol in in_sequence:
        emp_prob = float(in_sequence.count(symbol)) / (len(in_sequence) -
                                                       len(symbol) + 1)
                                    # vypocet „empirical probability“
        if emp_prob > p_min:
          new_node = TNode(parent_node, symbol) # novy uzol

          bigger_cond_prob = False
          for char in set(in_sequence):
            a = in_sequence.count(symbol + char)
            b = in_sequence.count(symbol, 0, - len(char))

            if b > 0:
              cond_prob = float(a) / b
            else:
              cond_prob = 0

            if cond_prob >= r * parent_node.value_dict[char] and \
              cond_prob > 0:
              bigger_cond_prob = True

            # novy uzol ak splna vsetky parametre, rekurzivne sa pren
            # zavola funkcia PST

            if bigger_cond_prob == True:
              parent_node.addChild(symbol, new_node)
              PST(new_node, in_sequence, p_min, max_symbol_len, r)
```

Algoritmus generovania novej sekvencie podľa PST

```
def Generate(dict, length, in_sequence):
  out_sequence = in_sequence
  tmp_key = ''

  while len(out_sequence) < length:
    tmp_key = ''
    i = 0
    # hlada sa najdlhsi kontext v slovniku reprezentujucom PST strom
    while not dict.has_key(out_sequence[i:]):
      i += 1
    context = out_sequence[i:]
    rnd = random.random()
    p = 0

    # podla nahodneho cisla sa vyberie nasledujuci symbol
    for key in dict[context].keys():
      p += dict[context][key]
      if p >= rnd:
        tmp_key = key
        break

    # symbol sa pripoji k vystupnej sekvencii
    out_sequence += tmp_key

  return out_sequence
```

Príloha 3

Ďalšie testy

Druhá séria testov prebehla na rovnakej nahrávke, pričom sa menili iba parametre týkajúce sa výškovej modulácie nôt.

PITCH_LOCAL=1 // výška nôt sa improvizuje pre daný takt iba z daného taktu

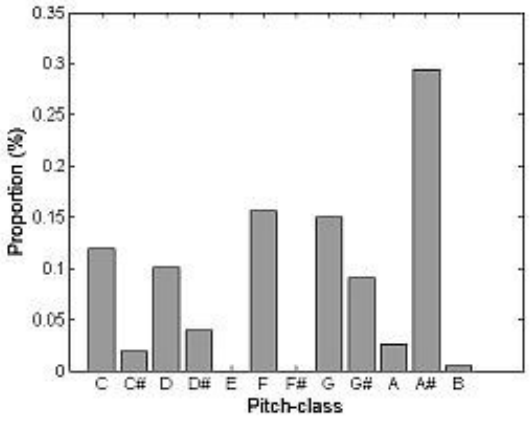
PITCH_SAME=0 // určuje koľko počiatočných tónov má mať rovnakú výšku ako v originály

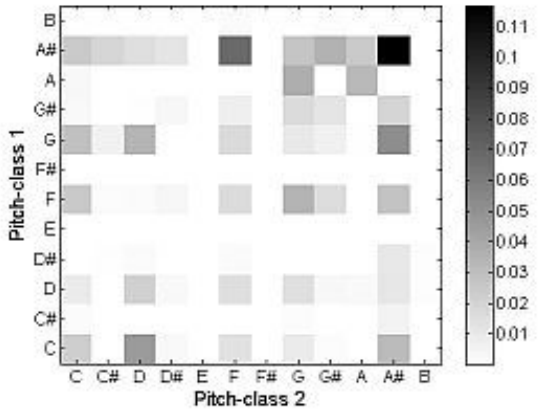
PST_PITCH_L=4 // maximálna dĺžka frázy v PST strome pre výšku nôt

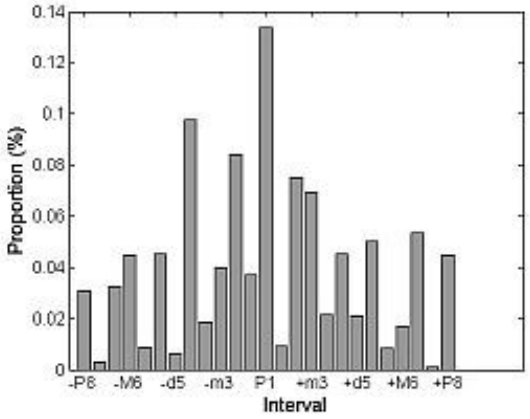
PST_PITCH_PMIN=0.2 // minimalna pravdepodobnosť frázy pre výšku nôt

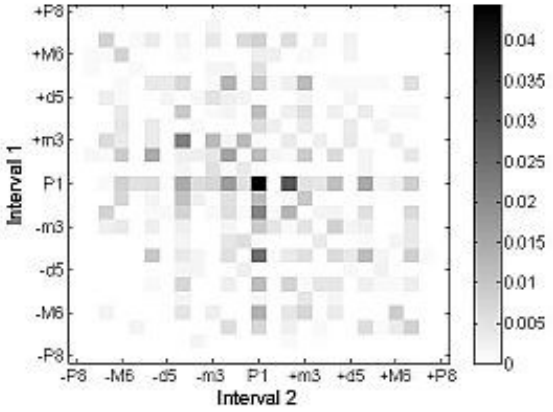
PST_PITCH_R=2 // násobiaci koeficient R pre výšku nôt

Ako možno vidieť z obrázkov, zvýšená maximálna dĺžka fráza a improvizácia len z nôt v rámci daného taktu spôsobila, že grafy štatistického rozdelenia sa viac podobajú na originálnu nahrávku. Možno teda usúdiť, že zvyšovanie maximálnej dĺžky frázy v PST strome spôsobuje väčšiu podobnosť s originálnou nahrávkou.









Tretia séria testov prebehla na rovnakej nahrávke, pričom sa opäť menili iba parametre týkajúce sa výškovej modulácie nôt.

PITCH_LOCAL=1 // výška nôt sa improvizuje pre daný takt iba z daného taktu

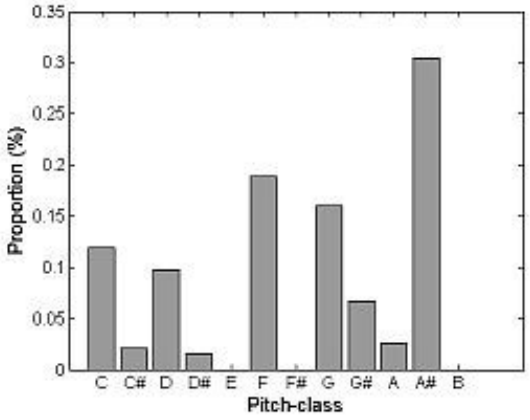
PITCH_SAME=0 // určuje koľko počiatočných tónov má mať rovnakú výšku ako v origináli

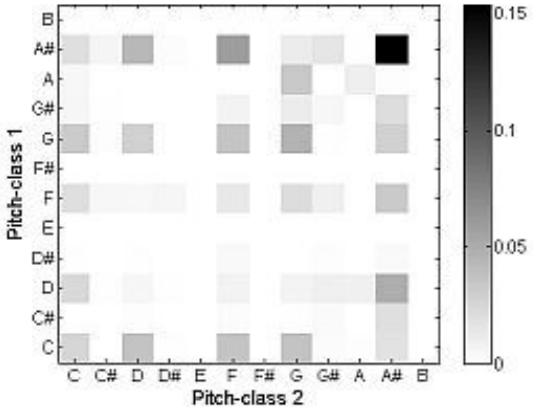
PST_PITCH_L=2 // maximálna dĺžka frázy v PST strome pre výšku nôt

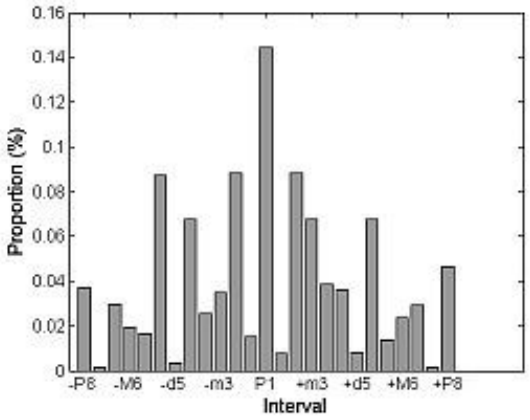
PST_PITCH_PMIN=0.1 // minimalna pravdepodobnosť frázy pre výšku nôt

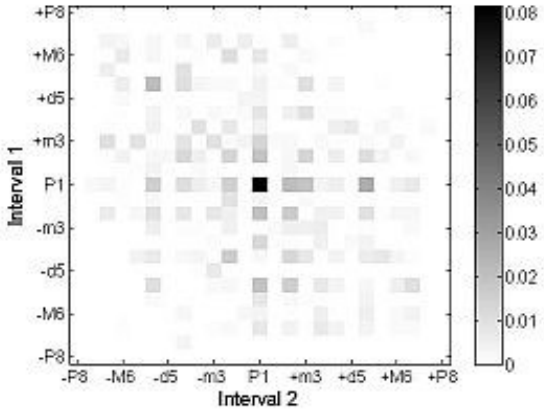
PST_PITCH_R=2 // násobiaci koeficient R pre výšku nôt

Ako možno vidieť z obrázkov, znížený prah pravdepodobnosti pre výskyt frázy opäť spôsobil, že improvizovaná skladba sa viac podobá originálu.



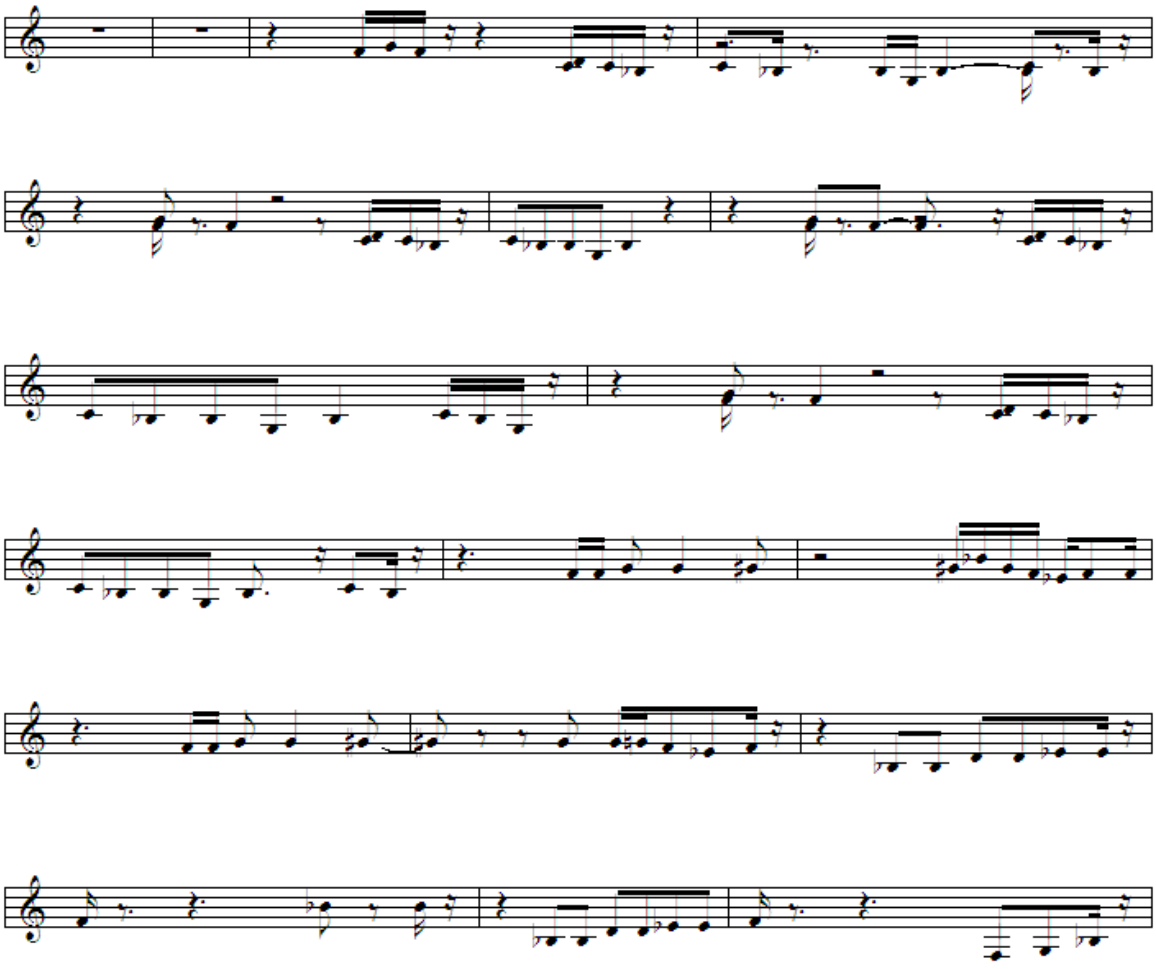




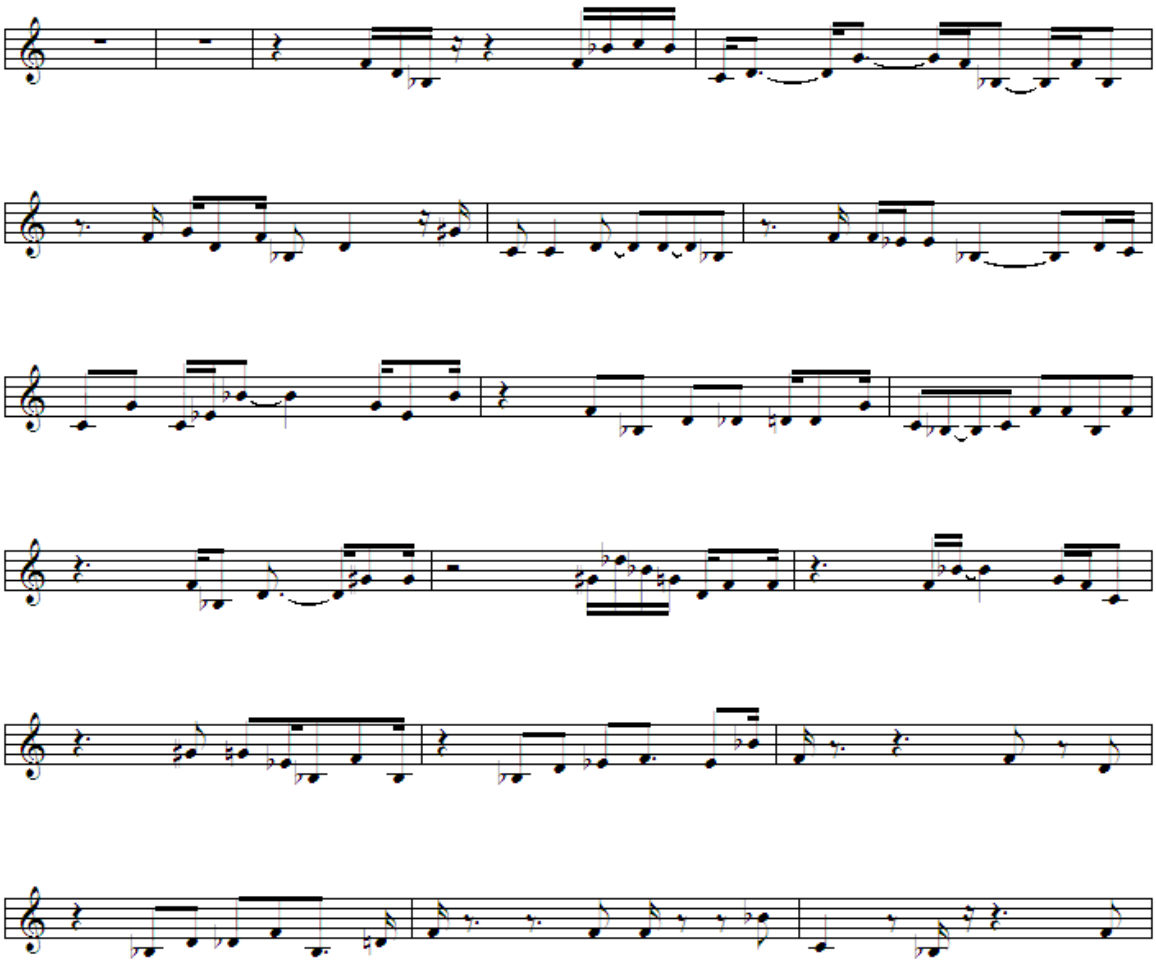


Príloha 4

Notové záznamy



Obrázok 15: Niekoľko začiatočných taktov vibrafónu z originálu skladby „Mercy, Mercy, Mercy“ od autora Joe Zawinul



Obrázok 16: Niekoľko začiatočných taktov vibrafónu z improvizácie podľa skladby „Mercy, Mercy, Mercy“ od autora Joe Zawinul