UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matevž Jekovec

Računalniška analiza tem v skladbah

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Janez Demšar
Somentor: dr. Andrej Brodnik

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01720/2010
Datum: 01.12.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MATEVŽ JEKOVEC**

Naslov: **RAČUNALNIŠKA ANALIZA TEM V SKLADBAH**
COMPUTER-AIDED MUSICAL THEME ANALYSIS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Muzikološka analiza glasbe obsega tudi nekaj postopkov, ki so sicer rutinski, vendar dolgotrajni in duhamorni. Eden od njih je iskanje pogostih tem, ki se pojavljajo v skladbi.

V okviru diplomske naloge opišite ozadje problema in predlagajte možno rešitev v obliki algoritma, ki bo čim natančneje našel - ali pomagal muzikologu najti - teme v skladbah. Implementirajte predlagane algoritme in razvijte primeren uporabniški vmesnik, ki naj omogoča branje datotek MIDI (vključno s potrebno transformacijo v notni zapis in linearizacijo glasov) in izpis v primerni obliki, predvsem pa naj omogoča preprosto preiskovanje seznama oziroma drevesa najdenih tem. Razvite metode preskusite na ustrezno izbranem naboru skladb z izrazito glavno temo.

Mentor:

doc. dr. Janez Demšar

Somentor:

doc. dr. Andrej Brodnik

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a Matevž Jekovec,

z vpisno številko 63040056,

sem avtor/-ica diplomskega dela z naslovom:

Računalniška analiza tem v skladbah

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom
  doc. dr. Janeza Demšarja
  in somentorstvom
  dr. Andreja Brodnika
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki „Dela FRI“.

V Ljubljani, dne 20. 6. 2011 Podpis avtorja/-ice:

Zahvala

Iskreno rad bi se zahvalil muzikologoma dr. Matjažu Barbu in dr. Alešu Nagodetu, ki sta me uvedla v sistematično analizo glasbe, mi predstavila težave muzikologov in me spodbujala pri razvoju aplikacije za analizo glasbe.

Zahvaljujem se moji ženi, staršem, bratu, tašči in tastu ter sošolcem, ki so me podpirali v napornih trenutkih študija in pri pisanju diplomskega dela.

Mojima mentorjema se zahvaljujem za motivacijo in vztrajnost ter vso tehnično in strokovno pomoč pri pisanju diplomskega dela.

Moji družini

Kazalo

Seznam uporabljenih kratic in simbolov

CAAC — *Computer Aided Algorithmic Composition*,

CAD — *Computer Aided Design*,

GPL — *GNU General Public License*,

LGPL — *GNU Lesser General Public License*,

MIDI — *Musical Instrument Digital Interface*,

GM — *General MIDI*,

SMF — *Standard MIDI Format*,

OCR — *Optical Character Recognition*,

PPQN — *Pulses Per Quarter Note*,

WTK — *Das Wohltemperierte Klavier*.

Povzetek

V diplomskem delu analiziramo postopek muzikološke sistematične analize kompozicijskih tehnik posamezne skladbe. Ta postopek je za muzikologa glede na obseg skladbe dolgotrajen in zahteven. Z namenom izboljšanja postopka zgradimo ustrezen teoretični model za analizo melodičnih prvin skladbe, ki ga je mogoče računalniško podpreti. Jedro modela predstavlja postopek linearizacije not in gradnja priponskega drevesa notnih parov, ki ga pretvorimo v drevo motivov in tem. Model nato uporabimo za izvedbo celostne uporabniške aplikacije *Harmonia*, ki omogoča uvoz datotek iz več glasbenih zapisov, analizo motivov in tem ter prikaz rezultatov posameznih korakov analize. Aplikacijo uporabimo za analizo fug v obeh zvezkih Dobro uglašenega klavirja J. S. Bacha in ugotovimo, da muzikološko interpretacijo dopolnjuje s pomembnimi statističnimi informacijami o skladbi ter z zgradbo, mesti pojavitev in oceno posameznih tem in motivov. Postopek analize tem testiramo glede na oceno posamezne teme v vlogi samodejnega iskanja glavnih tem v fugah in ugotovimo, da so fuge preveč raznolike, da bi bilo iskanje z enakim pristopom in izkustveno določenimi parametri uspešno. Dosežemo 16,67 % uspešnost.

Ključne besede:

računalniška analiza glasbe, muzikologija, MIDI, iskanje vzorcev, priponsko drevo, tema, Bach, fuga

Abstract

Computer-aided musical theme analysis

In this thesis we analyze the process of musicological systematic analysis of composition techniques in music scores. This process is difficult and can take several hours depending on the score length. In order to improve this process we propose a computer-supported theoretical model for analyzing melodic parts of the score. The core of this model comprises of notes linearization algorithm and a suffix tree of note pairs, which is transformed into a suffix tree of themes and motifs. We use this model to build the end-user application *Harmonia* which allows the import of music stored in different music file formats, motifs and themes analysis and visualization of the results of each analysis step. Later on we use the application to analyze J. S. Bach's fugues in Well-Tempered Clavier and improve musicological interpretation with important statistical information on melody and the structure, positions and evaluation of themes and motifs in the score. We also test the analysis algorithm to find the main themes in the given fugues using the theme evaluation function. We conclude that fugues in WTK are too heterogeneous for the test using the same approach and experimentally set parameters to be successful. We achieve 16.67 % accuracy.

Key words:

computer aided music analysis, musicology, MIDI, pattern searching, suffix tree, theme, Bach, fugue

Poglavje 1

Uvod

Muzikologija je veda, ki se ukvarja z glasbo kot s širšim fiziološkim, psihološkim in kulturnim fenomenom. Tradicionalno se deli na dve področji: zgodovinsko in sistematično. Slednje zajema raziskave razumljene teorije glasbe. Pod to med drugim spada analiza kompozicijskih tehnik (harmonija, kontrapunkt, oblikoslovje, nauk o melodiji, inštrumentacija in orkestracija), estetika in filozofija glasbe, sociologija in glasbena antropologija [56]. Analiza kompozicijskih tehnik zajema analizo ritmičnih, melodičnih, harmonskih in oblikovnih vzorcev, s katerimi muzikolog poskuša razumeti širši pomen glasbe [3]. Tovrstna analiza je glede na obseg skladbe dolgotrajna in zahtevna.

Trenutna vsakdanja programska oprema muzikologov omogoča delo z obsežnim, digitaliziranim arhivom slovenske in svetovne glasbe ter s širšo glasbeno refleksijo [40, 47, 38]. Uveljavljenih programov, ki bi omogočali analizo posamezne skladbe z vidika kompozicijskih tehnik, ni. V diplomskem delu smo se osredotočili na razvoj programske opreme, ki muzikologom pomaga pri glasbeni analizi osnovnih motivov, figur in tem. V ta namen je bilo najprej potrebno definirati teoretični model za reševanje tega problema, nato pa razviti konkretno računalniško aplikacijo.

Diplomsko delo je sestavljeno iz treh delov. Prvi zajema poglavji 2 in 3, v katerih spoznamo, kako ljudje zapisujemo, ustvarjamo in analiziramo glasbo. Najprej predstavimo glasbeni stavek in definiramo potrebne glasbene izraze, vede, ki se ukvarjajo z glasbo, in posebno tehniko kompozicije, imenovano imitacija. V tretjem poglavju analiziramo strokovna dela muzikologov, dosežena ročno ali s pomočjo računalnika, in nekaj enostavnejših obstoječih uporabniških aplikacij za tovrstno delo.

Drugi del diplomskega dela zajema teoretične osnove našega pristopa k analizi glasbe. V poglavju 4 opišemo metode za pripravo glasbe na analizo.

Pri tem spoznamo vrsto vhodnih podatkov, kot so zapis MIDI in notni zapis, ter pregledamo tudi druge zapise glasbe in možne pretvorbe v ta dva zapisa. Nato glasbo v notnem zapisu lineariziramo in dobljene segmente uporabimo v poglavju 5. Tu spoznamo podatkovne strukture, kot so številsko drevo, priponsko drevo in PAT drevo. Slednji dve nato uporabimo za zapis notnih parov in tem. Na koncu poglavja predstavimo uporabljeno funkcijo za ocenjevanje in rangiranje najdenih tem.

Zadnji del opisuje ustvarjeno uporabniško aplikacijo za računalniško analizo tem. V poglavju 6 podrobneje opišemo potek načrtovanja in izvedbe same aplikacije. V poglavju 7 predstavimo uporabo aplikacije za analizo del J. S. Bacha.

Sklepne ugotovitve diplomskega dela predstavlja poglavje 8, v katerem povzamemo dosežke in težave ter omenimo smernice za nadaljnje delo.

Poglavje 2

Teoretične osnove glasbe

V tem poglavju bomo spoznali definicije osnovnih gradnikov glasbe skozi oči glasbenikov. Nato bomo pregledali tudi nekaj temeljnih analitičnih in kompozicijskih elementov.

2.1 Diatonični sistem

Glasba je zavestno oblikovano, časovno urejeno zaporedje tonov. Tone danes zapisujemo v notah. Vsaka nota ima določeno višino in dolžino.

Za zapis višine se je uveljavil diatonični sistem, ki ima za najmanjšo enoto polton. Dva poltona tvorita cel ton. Enačba (2.1) opisuje višino poltona glede na frekvenco zvoka. Ton c (opisan v naslednjem odstavku) je na vseh poltonih p, ki so deljivi z 12.

$$p = 69 + 12 \times \log_2 \left(\frac{f}{440 \text{ Hz}} \right) \tag{2.1}$$

Diatonični sistem je postal osnova za lestvici dur in mol. Lestvici imata 7 stopenj. Vsaka lestvica ima lahko naravno, harmonično ali melodično obliko, vendar bomo v diplomskem delu za dur vedno privzeli naravni dur in za mol harmonični mol, če ne določimo drugače. Razdalje med stopnjami pri naravnem duru so celi toni, razen med 3. in 4. ter 7. in 8. stopnjo, kjer se nahaja polton. Pri harmoničnem molu se polton nahaja med 2. in 3., 5. in 6. ter 7. in 8. stopnjo, med 6. in 7. stopnjo pa cel ton in pol. Durova lestvica je postala osnova za definiranje imen tonov ali stopenj: c, d, e, f, g, a in h. Zaporedje teh tonov se periodično ponavlja. Razporeditev poltonov in celih tonov med omenjenimi toni je enaka razporeditvi v durovi lestvici, kjer je ton c prva stopnja in ton h sedma stopnja. Poleg teh sedmih tonov poznamo tudi za pol tona

Tabela 2.1: Možne kvalitete intervalov glede na kvantiteto.

razdalja	**kvantiteta**	**kvaliteta (število poltonov)**
0	prima	čista (0), zvečana (1)
1	sekunda	velika (2), mala (1), zvečana (3), zmanjšana (0)
2	terca	velika (4), zala (3), zvečana (5), zmanjšana (2)
3	kvarta	čista (5), zvečana (6), zmanjšana (4)
4	kvinta	čista (7), zvečana (8), zmanjšana (6)
5	seksta	velika (9), mala (8), zvečana (10), zmanjšana (7)
6	septima	velika (11), mala (10), zvečana (12), zmanjšana (9)
7	oktava	čista (12), zvečana (13), zmanjšana (11)

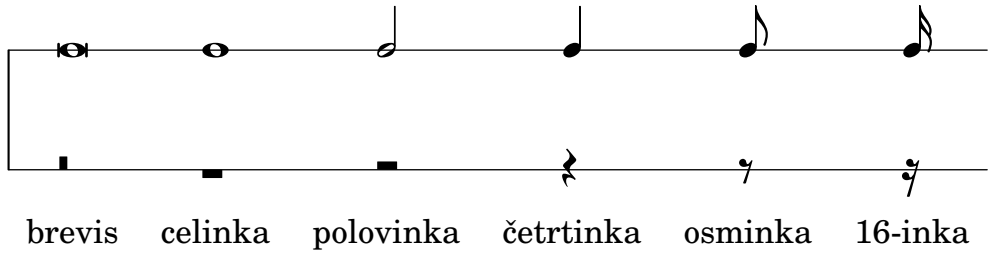
zvišane ali znižane tone. Pripona „is“ pomeni zvišanje za pol tona glede na prvotni ton, pripona „es“ pa znižanje. To nam omogoči opis tonov, ki niso v durovi lestvici. Še več, s tem lahko zapišemo katero koli lestvico z izhodiščem na poljubnem tonu, ki ima za najmanjšo enoto polton.

Interval v glasbi opisuje razdaljo višin med dvema tonoma. Interval sestavljata kvantiteta in kvaliteta. Kvantiteta predstavlja razdaljo med stopnjama tonov (od prime do oktave za razdalje od 0 do 7 stopenj). Lahko je absolutna, lahko pa podamo tudi smer (npr. terca navzgor, kvarta navzdol). Kvaliteto intervala opisuje število poltonov v intervalu. Pravilo, ki definira kvaliteto je, da so po durovi lestvici navzgor vsi intervali čisti (prima, kvarta, kvinta, oktava) ali veliki (sekunda, terca, seksta, septima), navzdol pa vsi čisti ali mali. Če je interval dodatno zvišan ali znižan za polton, postane zvečan ali zmanjšan. Tabela 2.1 prikazuje prvih 8 kvantitet intervalov glede na razdaljo med stopnjama tonov in z možnimi kvalitetami. Lastnosti intervalov z daljšo kvantiteto se periodično ponavljajo čez oktavo.

2.2 Notne dolžine in metrum

Za zapis notnih dolžin se uporablja relativna mera glede na dolžino ostalih not in izbrano mero za dobo. Slika 2.1 prikazuje grafični prikaz not in pavz ter imena notnih trajanj. Z leve proti desni je vsako notno trajanje na sliki dolgo polovico prejšnjega. Note z zastavicami imajo za vsako razpolovitev trajanja eno zastavico več.

Doba je konstanten pulz, ki ga čutimo pri poteku glasbe. Več dob sestavlja



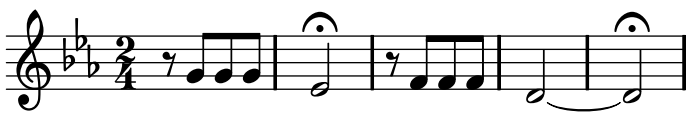
Slika 2.1: Grafični prikaz notnih dolžin.

takt. Metrum nam pove, katera notna dolžina je izbrana za dobo, in koliko dob meri en takt. Tipični metrumi so 3- ali 4-četrtinski. Metrum je pomemben z vidika poudarjenih dob znotraj takta. Zgradba takta ni nikoli simetrična. Prva doba v taktu je vedno najmočnejša. V 3-dobnem taktu sledita dve nepoudarjeni, v 4-dobnem pa sta druga in četrta nepoudarjeni, tretja doba pa je poudarjena, vendar manj kot prva.

2.3 Vloga glasu v glasbi

Glas je linearno zaporedje tonov, ki se časovno ne prekrivajo. Večglasje ali polifonija se pojavi ob hkratnem izvajanju vsaj dveh samostojnih glasov. Homofonija pa je način kompozicije, kjer glasovi ritmično niso neodvisni, ampak imajo postope skupne. V homofoniji se zato uporablja izraz akord, ki pomeni sočasni zven vsaj treh imensko različnih tonov [17]. Vsako večglasno skladbo lahko analiziramo na dva načina — z vidika analize glasov in z vidika analize akordov.

V diplomskem delu se bomo osredotočili na analizo glasov. Glas sestavljajo motivi. Motiv je najmanjša, melodično-ritmična enota, značilna tvorba, ki je sposobna samostojnosti [15]. Lahko se ponavlja, nastopi na drugi višini, v drugem glasu ali pa se spremeni. Če je sprememba večja, govorimo o novem motivu, ki je soroden prejšnjemu.



Slika 2.2: Začetna motiva Simfonije št. 5 Ludwiga van Beethovna.

Na sliki 2.2 sta prikazana začetna motiva Simfonije št. 5 Ludwiga van Beethovna. Simfonija se začne s slavnim „trkanjem usode na vrata“. Motiv je sestavljen iz zaporedja treh kratkih, nepoudarjenih in ene dolge, poudarjene note. Ta vzorec se ponavlja skozi celo skladbo na različnih višinah, v različnih inštrumentih in glasovih ter v avgmentiranih (dvakrat daljših) in diminuiranih (za pol krajših) notnih dolžinah.

Poleg motivov je bistvena tudi tema. To je daljša, vsaj nekaj taktov dolga miselna celota. Ponavadi vsebuje simetrično členjenje motivov in je zaključena s kadenco [15]. Z izjemo počasnih stavkov, se tema v skladbi skoraj vedno večkrat ponovi.

2.4 Algoritmična kompozicija

Algoritmična kompozicija združuje algoritme, ki nam pomagajo ustvarjati glasbo [31]. Ti algoritmi vsebujejo različna formalna pravila, ki skladateljem vnaprej dajo informacijo o tem, kaj se sliši dobro in kaj ne. Tovrstna pravila so se razvijala izkustveno skozi stoletja in pojavile so se različne vede. Ena izmed njih, ki se ukvarja s pravili za gradnjo in vezavo akordov, je harmonija [46]. S pravili za priredbo glasbe specifični zasedbi se ukvarja orkestracija [60]. Z zgradbo glasbenih del se ukvarja glasbena teorija z oblikoslovjem [55].

V diplomskem delu pomembno vlogo igra kontrapunkt [37]. Ta veda nas uči, kako istočasno voditi več melodij, ki so si v medsebojnem sozvočnem sožitju, pri tem pa oblikovati posamezne melodije v samostojne, enakovredne celote. Posebna tehnika skladanja, ki jo uči kontrapunkt, je imitacija — posnemanje ritmičnih in melodičnih značilnosti neke teme v drugem glasu [19]. Primer izvrstne rabe imitacije so renesančni moteti. Skladatelj zasnuje temo tako, da jo je mogoče kasneje brez sprememb ponoviti na drugih stopnjah. Tu so med sosednjimi notami in med poudarjenimi notami v taktu dovoljeni le veliki, mali ali čisti intervali. Pomemben je tudi obseg (*ambitus*) posameznega istosmernega gibanja. Razlog za težave se skriva v naravni prisotnosti tritonusa (t. i. *diabolus in musica*) v tonskem sistemu. To je relacija med tonoma f-h v obliki intervalov zvečane kvarte ali zmanjšane kvinte, ki ni uporabna. Kvaliteta skladatelja se kaže tudi v tem, koliko vzporedno izvedenih tem lahko zloži v smiselno celoto. V tem primeru mora pri gradnji začetka teme upoštevati sočasni zven ene ali več naslednjih not v temi na različnih stopnjah, kar je miselno izredno zahtevno. Z muzikološkega vidika nas zanima, kje in na kakšen način so skladatelji uporabljali imitacijo, kakšne teme so sestavljali, kolikokrat so temo ali posamezne motive imitirali, kakšne variacije so se pri tem pojavile.

Poglavje 3

Dosedanje delo

V tem poglavju bomo spoznali pomembnejše dosežke na področju sistematične muzikološke analize kompozicijskih tehnik, računalniške analize in kompozicije posameznih prvin glasbe ter uporabniških aplikacij. Algoritmična pravila za kompozicijo, ki smo jih spoznali v prejšnjem poglavju, nam povedo, da zgradba kompozicije ni naključna, ampak so vedno prisotne določene strukture in vzorci. Če so skladbe zapisane z notami, so strukture jasne (note in pavze, akord, motiv, takt ipd.). Naloga vseh vrst analiz glasbe je torej najti vzorce izbranih struktur in jih interpretirati na uporabniku razumljiv način.

3.1 Sistematična muzikološka analiza kompozicijskih tehnik skladb

Riemann je v priročniku za harmonijo [21] utemeljil harmonski dualizem oziroma funkcionalno harmonijo, ki je postala podlaga za harmonsko analizo skladb. Tone durove in molove lestvice je razdelil na 7 funkcij, na katerih je zgradil akord. Med funkcijami so bile tri glavne — tonika (prva), subdominanta (četrta) in dominanta (peta). Pokazal je, da lahko vsako durovo in molovo funkcijo interpretiramo kot svojo tonaliteto in tako s „stranskimi vejami“ razširimo prvotno. Dualizem se kaže v tem, da se tuj akord lahko interpretira kot akord, ki je del stranske veje prvotne tonalitete ali pa kot modulacija v novo tonaliteto. S funkcionalno harmonijo je uspešno analiziral veliko del, med njimi tudi vseh 48 preludijev in fug iz zbirke Dobro uglašenega klavirja (WTK) J. S. Bacha [20]. Iskal je predvsem harmonska razmerja znotraj tonalitete.

Schenker [22] je z analizo petih znanih del Bacha, Haydna in Chopina pokazal, da lahko vse skladbe interpretiramo tako, da ima harmonska zgradba

obliko T-D-T. Melodija se začne na terci tonike, nadaljuje s kvinto dominante in konča na osnovni stopnji tonike. Schenker je uvedel inovativen potek analize po plasteh. Najprej je poiskal in analiziral makro strukture, nato pa te drobil, dokler ni prišel do prvotne skladbe. Pristop k analizi skladbe po plasteh od splošnega k specifičnemu je v uporabi še danes. Veliko muzikologov je skeptičnih do t. i. „šenkerjanske analize“, saj je definiranje le treh makro struktur omejujoče, zvočna slika pa se lahko močno razlikuje od struktur na papirju.

Forte [8] se je ukvarjal z analizo atonalne glasbe Schönberga, Weberna, Stravinskega, Berga, Busonija, Skrjabina in drugih. Namesto funkcionalne harmonije je za osnovo vzel poltonski prostor. Definiral je popolnoma nove strukture, kot so razred višin, intervali v poltonski razdalji, dogodek modulacije v novo bazo, razmerje med vektorji intervalov ali razredov višin itd. Forte se je ukvarjal predvsem z melodijo v določenem prostoru. Kot rezultat analize je za posamezne stavke skladb zgradil različne simetrične matrike medsebojnih razmerij vektorjev intervalov in razredov višin.

3.2 Analiza in kompozicija s pomočjo računalnika

Temperley [25] je v knjigi Music and Probability v splošnem raziskoval uporabnost verjetnosti v glasbi. Najprej je analiziral melodične in ritmične vzorce, ki se pojavljajo v klasični glasbi, in nato razložil, kako ta model lahko zazna morebitna odstopanja (napake) v zapisu. V nadaljevanju je utemeljil uporabo Bayesovega modela verjetnosti za različne namene: izboljšanje algoritma Baum-Welch [4] za transkripcijo glasbe iz zvočnih datotek, analizo in napovedovanje melodij pri nemški ljudski glasbi, generiranje improviziranih grških pravoslavnih koralov s pomočjo končnih avtomatov in podobno. Temperley se ni osredotočal le na analizo, ampak tudi na kompozicijo glasbe s pomočjo statističnih metod.

Z računalniško harmonsko analizo skladb se je ukvarjala Ferkova [6]. V ta namen je s kolegi razvila več algoritmov, ki jih je uporabila v programu Analysis [32]. Ta za vhod vzame poljuben zapis MIDI, nato poskuša glede na prvih sedem tonov zaznati tonaliteto skladbe in glede na metrum in ritem določiti vrste akordov (trozvok, četverozvok, dur, mol) ter funkcije (tonika, subdominanta, dominanta, paralele). Program Analysis je uporabila za analizo del Mozarta, Schuberta in Brahmsa ter v članku predstavila statistično najbolj uporabljene trozvoke in četverozvoke ter verjetnosti njihovih sosednjih

akordov.

Pomemben strukturni pojav, ki je prisoten v vseh umetnih skladbah, je našel Tse [26]. Analiziral je dela Bacha, Chopina, Mozarta, rusko ljudsko glasbo in šansone. Za vhod je vzel zapis MIDI in zgradil graf, kjer je vozlišče predstavljalo noto s točno določeno višino in dolžino, povezave med vozlišči pa so mu predstavljali sosednji pari not v skladbi. Uteži povezav so predstavljale frekvenco ponovitev posameznega para — večkrat, ko se je notni par pojavil v skladbi, bolj je utežil povezavo. Nato je pokazal, da je dobljeni graf daleč od naključnega in da ima vsa analizirana glasba podobno strukturo tega grafa — podobno topologijo, eksponentno verjetnostno porazdelitev sosednjih vozlišč, povprečno stopnjo vozlišč, koeficient uvrščanja in težišče. Lastnosti in topologija grafa so podobni kompleksnim omrežjem, ki so se razvili instinktivno (biološki sistemi, splet, socialna omrežja, letalske povezave) [2]. Tse je graf, pridobljen iz Mozartovih del, uporabil tudi za kompozicijo.

S kompozicijo novih motivov sta se ukvarjala Yang [30] in Anders [1]. Prvi je analiziral ritem in melodijo japonske ljudske glasbe in ustvaril zbirko ritmičnih in melodičnih motivov. Nato je pokazal, da lahko že z naključnim izborom množice motivov in z upoštevanjem frekvence pojavitev not v njih sestavimo zanimive melodije. Anders pa je po vzoru sistemov za harmonizacijo glasbe (pisanje akordov na podano melodijo) predstavil sistem za kompozicijo motivov in njihovih variacij s pomočjo logičnega programiranja CSP (*Constraint Satisfaction Problem*). Ta deluje tako, da mu v začetku podamo dejstva (možni skoki in ritmična razmerja), kot rezultat pa na podlagi vgrajenih predikatov dobimo možne motive in njihove variacije. Sistem ne pozna verjetnosti posameznih dejstev, ampak je to potrebno doseči z ustreznim zaporedjem in strukturo teh.

Z iskanjem ter analizo motivov in tem sta se ukvarjala Weyde [29] in Takasu [24]. Weyde je utemeljil segmentacijo posameznih motivov. Pokazal je, da motiv lahko razdelimo na več delov tako, da meje med njimi določajo podobna ritmična in melodična zaporedja znotraj motiva. To pa je uporabno pri ocenjevanju podobnosti motivov med seboj, saj ne primerjamo več celotnih nizov, ampak njihove posamezne segmente.

Takasu se je ukvarjal z obsežnim problemom iskanja glavne teme skladbe. Za vhod je vzel enoglasne melodije v zapisu MIDI. Nato je s pomočjo segmentacije (privzel je, da motive obdajajo pavze) dobil vrsto „podmelodij“. Z mero LCS (*Longest Common Sequence*) je izračunal podobnosti med njimi in statistično določil prag. Nato je s pomočjo klasifikacijskega drevesa ocenil, katere podmelodije so lahko začetki glavne teme skladbe. Klasifikacijsko drevo je vsebovalo več stopenj kriterijev za relativno povprečno višino in število po-

javitev. Konstante in verjetnosti v drevesu so bile vnaprej določene. Oceno podmelodij, izračunano iz verjetnosti dobljene ciljne množice v klasifikacijskem drevesu, je izboljšal s pomočjo verjetnosti iz končnega avtomata. Tega je sestavil iz frekvenc ponovitev vseh trojk zaporednih podmelodij. Pri analizi japonske pop-glasbe je s tem načinom dosegel 88 % natančnost pri iskanju glavne teme.

3.3 Uporabniške aplikacije

V prejšnjem podpoglavju smo spoznali teoretične pristope za računalniško analizo in deloma tudi kompozicijo glasbe. Uveljavljenih produktov, ki bi ta znanja vključevala, danes ni. Obstajajo le manjše aplikacije, ki so namenjene demonstraciji določenih teoretičnih pristopov.

Na trgu se sicer za področje algoritmične kompozicije in analize omenja izraz *Computer-Aided Algorithmic Composition* [36]. Žal je izraz „*Computer-Aided*“ pogosto napačno interpretiran kot programska oprema za pisanje not, podobno, kot so orodja CAD namenjena tehničnemu risanju 2D/3D modelov. Izkaže se, da ima vsa današnja programska oprema za pisanje glasbe zelo skromno podporo za pravo algoritmično kompozicijo ali analizo.

Primer manj znane aplikacije, namenjene glasbeni analizi, je *JRing* [11]. Ta omogoča uvoz glasbe iz zapisa MIDI, označevanje ritmičnih, harmonskih in melodičnih posebnosti ter izvoz v tekstovni zapis za nadaljnjo obdelavo. Program ne vsebuje nobenih pravil za samodejno zaznavo teh posebnosti ali njihovo analizo. Žal ga uporablja le ozek krog ljudi in nikoli ni doživel večjega razmaha. Drugi primer je v prejšnjem poglavju omenjena aplikacija Analysis [32], ki je bila razvita za potrebe članka in nima več aktivnega razvoja.

Poleg samostojnih aplikacij pa obstajajo tako brezplačne kot komercialne razširitve splošnih aplikacij za delo z notami. Na področju analize glasbe pa je teh razširitev malo [62]. Predvsem gre za statistični prikaz intervalov in akordov v melodiji, ambitus inštrumentov, primerjavo vsebine dveh črtovij ipd. brez interaktivnega uporabniškega vmesnika. Zahtevnejše analize skladbe, kot je analiza in prikaz motivov in harmonije, ne omogočajo.

Poglavje 4

Priprava glasbe za analizo

V tem poglavju bomo najprej spoznali glasbene zapise, ki jih podpiramo kot vhod v našo aplikacijo. Poleg teh bomo pogledali še druge digitalne zapise glasbe in možne načine za pretvorbo na podprte. V nadaljevanju sledi njihova obdelava — najprej na ustrezen notni zapis, nato pa še s postopkom linearizacije na segmente, ki so primerni za nadaljnje korake analize, opisane v naslednjem poglavju.

4.1 Zapis MIDI

Standard MIDI se je oblikoval v 80. letih prejšnjega stoletja v prvi vrsti kot odgovor na nezdružljive vmesnike za komunikacijo med digitalnimi inštrumenti [52, 50]. MIDI je zasnovan kot komunikacijski protokol, poleg tega pa standard definira tudi zapisovanje vsebine v datoteko (*Standard MIDI format* ali *SMF*) [51]. Leta 1991 je bila sprejeta dopolnitev prvotnega zapisa, imenovana *General MIDI* ali *GM* [44]. Med njenimi največjimi novostmi je bila uvedba obveznih inštrumentov, ki jih morajo odjemalci podpirati, kar je v praksi močno poenostavilo izmenjavo glasbenih vsebin. Večjih sprememb standard kasneje ni več doživel.

V diplomskem delu uporabljamo med drugimi zapisi tudi uvoz glasbe iz datotek v zapisu MIDI. Za potrebe diplomskega dela bomo v grobem spoznali zgradbo tega zapisa, ne pa tudi dejanske implementacije.

Datoteka MIDI vsebuje več sledi, ki naj se izvajajo sočasno. Glava vsake sledi poleg unikatnega ID vsebuje tudi informacijo o časovnem deljenju. Ta podatek nam pove trajanje abstraktnih dob (*Parts or Pulses per Quarter Note* ali PPQN) v milisekundah.

Jedro sledi vsebuje niz dogodkov MIDI. Poznamo tri vrste dogodkov:

- dogodke, ki so del kanala,
- meta dogodke in
- dogodke *SysEx*.

Kanal v sledi predstavlja izvajalca. Po standardu MIDI je kanalov 16. *General Midi* določa, da je kanal 10 rezerviran za tolkala. Dogodki, ki so del kanala, so sestavljeni iz petih delov: relativnega časa dogodka, tipa, ID kanala in dveh parametrov.

Relativni čas je čas, izražen v abstraktnih dobah, ki je minil od zadnjega dogodka v kanalu.

Tipi dogodkov so med drugimi: začetek ali konec tona, nastavitve izvajanja (jakost, pedal, drugi efekti) in izbira inštrumenta. Pri slednjem opazimo, da v zapisu MIDI inštrument ni določen za celotno sled, ampak se lahko z ustreznim dogodkom kadar koli spremeni.

Za naše delo je najpomembnejši dogodek začetek (`NOTE ON`) ali konec tona (`NOTE OFF`). Prvi parameter vsebuje višino, drugi pa jakost tona (*velocity*). Višina tona je določena v poltonski skali, ki jo opisuje enačba (2.1). Zapis MIDI torej ne vsebuje imen tonskih stopenj (od c do h). Prednost tega načina je enostavnost zajema, saj je preslikava med klavirskimi tipkami in poltonsko skalo trivialna. Po drugi strani pa v aplikacijah za delo z glasbo, zapisano v notah, tak način predstavlja potrebo po dodatni logiki za pretvorbo v notni zapis, kar ni enostavno. Poglavje 4.4 podrobneje opisuje omenjeni problem.

Poleg dogodkov, ki so del kanala, poznamo še meta dogodke in dogodki *System Exclusive* (SysEx). Prvi so med drugim namenjeni opisu tonalitete, taktovskega načina in tempa, komentarjem ter besedilu skladbe. Dogodki SysEx so namenjeni poljubnim podatkom, ki niso del standarda MIDI, in jih lahko proizvajalci strojne in programske opreme uporabijo za svoje namene.

4.2 Notni zapisi

Notni zapisi za razliko od MIDI vsebujejo glasbene dogodke, zapisane z notami. Notni zapis obvezno vključuje:

- zapis višine tonov s tonskimi stopnjami (od c do h) in z ustreznimi predznaki (višaji, nižaji, razvezniki),
- zapis trajanja tonov v notnih dolžinah (glej sliko 2.1),
- zapis tonov v ustrezen glas, ki je del notnega črtovja,

- notna črtovja z določenim ključem, stalnimi predznaki in taktovskim načinom.

Danes obstaja množica dvojiških in besedilnih tovrstnih zapisov, saj skoraj vsaka odprtokodna ali licenčna aplikacija za pisanje not uporablja svojega. Še bolj zaskrbljujoče je, da nobena od licenčnih aplikacij, ki uporablja dvojiški zapis, nima ustrezne specifikacije zanj, tako da je neposreden uvoz zapisa v drugo okolje praktično nemogoč.

Z namenom izmenjave glasbe med aplikacijami za pisanje not je podjetje Recordare leta 2004 objavilo odprt, besedilni zapis glasbe MusicXML (www.musicxml.org). Ta zapis lahko, poleg vseh zgoraj omenjenih elementov, vsebuje tudi oblikovne lastnosti skladbe, namenjene za tisk. Zaradi narave XML pa lahko zapis tudi razširimo s poljubnimi značkami za potrebe aplikacij za pisanje not. Podjetje Recordare je sodelovalo tudi z najbolj razširjenimi tovrstnimi aplikacijami in poskrbelo za ustrezne razširitve za izvoz in uvoz v in iz tega zapisa. Danes ga podpira več kot 140 programov — od licenčnih in odprtokodnih aplikacij za pisanje not, rešitev za razpoznavo not s slike in sekvenčnikov. V diplomskem delu smo za vhodne podatke poleg zapisa MIDI omogočili tudi uvoz not iz zapisa MusicXML.

4.3 Drugi zapisi

Med drugimi vrstami zapisov digitalne glasbe največkrat najdemo zvočne in slikovne zapise.

Strojna in programska oprema za zajem, predvajanje in kodiranje digitalnega zvoka je v zadnjih desetletjih močno napredovala [39]. Če gre za glasbo, je s pomočjo spektralne analize zvoka mogoče določiti frekvence tonov in zvočni zapis pretvoriti v zapis MIDI. Ta postopek je izredno zahteven, saj zven fizičnih inštrumentov vsebuje poleg frekvence osnovnega tona tudi vrsto drugih frekvenc, ki dajejo inštrumentu barvo. V živo zajetem zvoku pa delo še dodatno oteži šum. S pretvorbo iz zvočnega v zapis MIDI se intenzivno ukvarjajo že od njegovega nastanka. Današnji pristopi poskušajo razčleniti zvok s pomočjo obsežnih knjižnic zvenov inštrumentov na vseh višinah in njihovega optimalnega prileganja vhodnemu signalu [5]. Medtem ko za enoglasne skladbe pretvorba deluje dobro, natančnost pri polifonih delih redko presega 50 %. Na trgu je kljub temu na voljo vrsta komercialnih rešitev za pretvorbo enoglasnega ali večglasnega zvočnega zapisa v zapis MIDI [57, 74].

Slikovni zapisi not lahko vsebujejo optično zajet fizični notni material ali pa dokument, generiran s pomočjo aplikacije za pisanje not. Področje, ki se

Tabela 4.1: Primerjava zapisa MIDI z notnim zapisom.

funkcionalnost	**zapis MIDI**	**notni zapis**
tonske višine	v poltonih	v notnih višinah
dolžine tonov	v abstraktnih dobah	v notnih dolžinah
pavze	ne	da
tonaliteta	opcijsko	da
taktovski način	opcijsko	da
glasovi	s pomočjo kanalov	da
črtovja	s pomočjo sledi	da
inštrumenti	lahko določeni kadar koli	vedno določeni na začetku

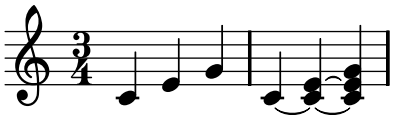
ukvarja z digitalizacijo znakov, se imenuje *Optical Character Recognition* ali *OCR* [59]. Posebno podpodročje, ki pokriva pretvorbo glasbe iz slikovnega v notni zapis, se imenuje *Music OCR* [54]. V primerjavi z današnjimi rešitvami za pretvorbo zvočnega zapisa v MIDI je pri aplikacijah za pretvorbo slikovnega zapisa v notnega natančnost neprimerno boljša (med 90 in 100 %). Na trgu obstajajo tako licenčne [67] kot odprtokodne rešitve [33]. Izhodni zapis teh aplikacij je običajno MusicXML.

4.4 Pretvorba zapisa MIDI v notni zapis

Tabela 4.1 prikazuje pomembnejše lastnosti zapisa MIDI v primerjavi z notnim zapisom.

Kakovostna pretvorba iz zapisa MIDI v notni zapis je v diplomskem delu izredno pomembna za analizo tem, saj ta temelji na treh prvinah: intervalih (notne višine), ritmu (notne dolžine) in linijah (glasovi). Zapis MIDI teh prvin neposredno ne opisuje. V nadaljevanju bomo zato za vsako lastnost v zgornji tabeli spoznali uporabljen postopek pretvorbe.

Preslikava tonske višine iz poltonske skale v notno višino ni enolično definirana. Razlog za to so enharmonični toni, kar je posledica možnosti zviševanja ali zniževanja osnovnih stopenj (glej poglavje 2.1). Enočrtani ton c' ima npr. v zapisu MIDI določeno višino 60, to višino pa lahko zasedajo tudi toni his', deses", aisisis', eseseses" ipd. Težave z enharmoničnimi toni smo rešili s pomočjo podane tonalitete in pravil za zapis labilnih alteracij. Če je tonaliteta podana in je ton harmonski (del tonalitete), je rešitev trivialna. Za notno višino privzamemo višino, ki jo vsebuje tonaliteta. Če pa je ton harmonsko tuj, pa se držimo pravila, da so prva, druga, četrta in peta stopnja tonalitete zvišane,



Slika 4.1: Zapis razloženega akorda v enem glasu.

sedma pa znižana. Razlog za to je skrit v dominantah stranskih stopenj, saj so zvišane stopnje prvotne tonalitete vedno lahko le terce stranskih dominant, edina znižana stopnja pa septima. Drugih kombinacij v duru in molu ni [18]. Če tonaliteta ni podana, privzamemo C-dur.

Poleg pravil, vezanih na tonaliteto, notni zapis dodatno izboljšamo še s pravilom za zapis prehajalnih in menjalnih tonov. Ta za vsako trojko sosednjih tonov pravi, da so prehajalni toni navzgor zvišani, navzdol pa znižani, kjer je interval med začetnim in končnim tonom v trojki sekunda. Menjalni toni gor so, obratno, vedno znižani, dol pa vedno zvišani, če je interval med začetnim in končnim tonom v trojki prima.

Za pretvorbo dolžine MIDI v notno dolžino privzamemo razmerje določeno v standardu MIDI (96 udarcev PPQN na četrtinko, 48 na osminko, 192 na polovinko itd.) [71]. Vmesne dolžine MIDI so dovoljene, vendar ne več kot do treh notnih pik. Sicer je vrednost MIDI zaokrožena na najbližjo notno vrednost do treh pik. Triol in drugih poddelitev ne podpiramo. Pavze zapišemo, če je čas med koncem prejšnje in začetkom naslednje note v istem kanalu daljši od 0. Različni tempi pri zapisu notnih dolžin ne igrajo vloge. Ravno tako smo za potrebe diplomskega dela ignorirali taktovski način (in posledično metrum).

Za razporeditev not po glasovih smo uporabili pravilo, da je vsak kanal svoj glas. Če so dogodki, ki predstavljajo ton, v istem kanalu razporejeni tako, da se prekrivajo, sočasni zven s pomočjo vezajev zapišemo v obliki akorda. Na sliki 4.1 je primer takšnega zapisa.

Za zapis črtovij smo privzeli, da vsaka sled predstavlja svoje črtovje. Glasove razporedimo po črtovjih v istem zaporedju, kot se pojavijo dogodki po kanalih v sledi. Črtovja poimenujemo po imenih prvih pojavitev inštrumentov v vsakem kanalu.

4.5 Linearizacija not

Linearizacija not je postopek, pri katerem iz zaporedja not v vsakem glasu dobimo krajše, enoglasne, disjunktne segmente, ki so primerni za nadaljnjo

obdelavo. Notni zapis kljub definiciji glasu v poglavju 2.3 dovoljuje naslednji značilnosti, ki nista primerni za analizo tem:

- en glas lahko hkrati izvaja več not (akord), če so te enako dolge in
- glas lahko traja le manjši del skladbe, v notnem zapisu pa je prisoten skozi celotno skladbo.

V ta namen smo zasnovali algoritma, ki s pomočjo najbližjih sosedov uvrstita note v ustrezno dolge, enoglasne segmente.

4.5.1 Iskanje najbližjih sosedov v globino

Za sosednost not smo uporabili naslednjo definicijo: *druga nota je prvi sosednja, če je evklidska razdalja med višinami in časi njunih začetkov v primerjavi z ostalimi notami v okolici prve najkrajša.* V enačbi (4.1) lastnost sosednosti note opisuje funkcija $ngb()$, kjer sta n_1 in n_2 sosednji noti, funkciji $p()$ in $t()$ pa opisujeta notno višino v poltonski skali in absolutni čas začetka note v milisekundah. Konstanti c_p in c_t sta uteži za notno višino in čas. Z njima določimo občutljivost funkcije na skoke. Uporabne vrednosti konstant se gibljejo od razmerja $100 : 1$ za hitre skladbe do $240 : 1$ za počasnejše.

$$\begin{aligned} ngb(n_i) &= \arg\min_{n_j} d(n_i, n_j); \\ d(n_i, n_j) &= c_p \cdot (p(n_j) - p(n_i))^2 + c_t \cdot (t(n_j) - t(n_i))^2 \end{aligned} \tag{4.1}$$

Algoritem 1 prikazuje implementacijo iskanja najbližjih sosedov v globino in razporejanja po segmentih. Za ime „iskanje najbližjih sosedov v globino“ smo se odločili, ker je iskanje podobno iskanju elementa v drevesu v globino (*Depth-first search*). Globino predstavlja čas v skladbi, stopnjo posameznih vozlišč pa sočasni zven tonov v istem glasu. Algoritem deluje tako, da za vsak glas izbere prvo noto, nato pa v istem glasu sledi njenim najbližjim kasnejšim sosedom. Postopek se ponavlja, dokler ne obišče vseh not v skladbi in jih ustrezno razporedi po segmentih. Note, povezane z vezaji, obravnava kot eno, daljšo noto.

Algoritem uporablja naslednje funkcije:

- $mark(n)$: Označi podano noto n.
- $getFirstUnmarkedNote()$: Poišče prvo neoznačeno noto v katerem koli glasu.

Algoritem 1: Algoritem za iskanje najbližjih sosedov v globino.

Input: V: seznam glasov
Output: S: seznam segmentov not

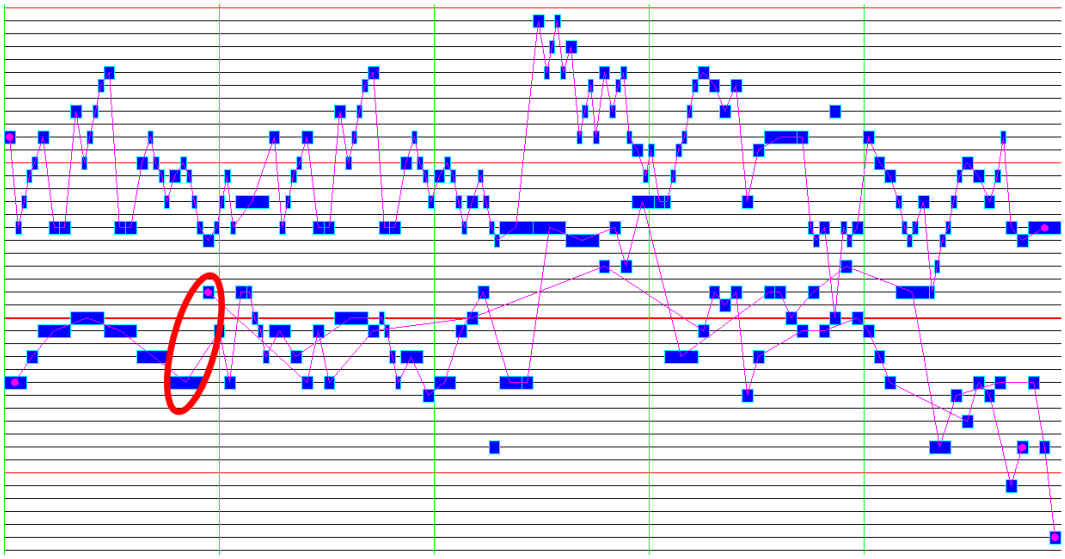
```
1  n = getFirstUnmarkedNote();
2  while n ≠ ∅ do
3  |   while n ≠ ∅ do
4  |   |   mark(n);
5  |   |   Dodaj n v trenutni segment s;
6  |   |   n ⟵ getClosestNote(n);
7  |   end
8  |   if s ≠ ∅ ∧ Len(s) ≥ MIN_LENGTH then
9  |   |   Dodaj s na seznam segmentov S;
10 |   end
11 |   Clear(s);
12 |   n ⟵ getFirstUnmarkedNote();
13 end
```

- *getClosestNote*(n): Poišče najbližjo noto, ki v istem glasu sledi noti n. Za izračun razdalje med notama smo uporabili enačbo (4.1). Funkcija lahko upošteva tudi največjo horizontalno (časovno) ali vertikalno (višinsko) razdaljo med notama.

Poleg zgornjih funkcij algoritem uporablja tudi konstanto MIN_LENGTH. Ta zagotavlja, da algoritem krajših segmentov od te konstante ne bo generiral.

Časovna zahtevnost algoritma je odvisna od funkcije *getClosestNote*(n). Če je ta linearna, pregleda vse naslednje note v istem glasu od trenutne dalje. Število primerjav celotnega algoritma je $O(n^2)$, kjer je n število not. Suboptimalna rešitev je, da funkcija pregleda le naslednjih k not za trenutno v istem glasu. Časovna zahtevnost v tem primeru je $O(kn)$. Uporabne vrednosti za k so zaradi strukture figur pri hitrih skladbah (štiri ali osem šestnajstink v obeh rokah in pedalu, kjer so prve v ločenem segmentu) ponavadi od 4 do 24. Če je število not v glasovih približno enako, funkcija *getClosestNote*(n) pregleda namesto $\frac{n}{2}$ v povprečju le $\frac{n}{2v}$ not, kjer je v število glasov. Posledično se časovna zahtevnost celotnega algoritma zmanjša za faktor števila glasov.

Da bi preizkusili delovanje algoritma, smo uvozili Menuet št. 1 v G-duru J. S. Bacha v zapisu MIDI, ki ima vse tone zapisane v enem kanalu. Ta oblika datoteke je tipična za zajete posnetke igranja klaviature v živo. Pretvorba v notni zapis je posledično generirala le en glas, ki je imel visoko stopnjo sočasno



Slika 4.2: Note, razvrščene po segmentih s pomočjo algoritma za iskanje najbližjih sosedov v globino. Med 7. in 8. noto v levi roki je označen skok, ki bi moral biti del istega segmenta.

zvenečih not. Optimalen rezultat našega algoritma bi bila razporeditev not desne roke v en zgornji segment, note leve roke pa v drugega, spodnjega. Slika 4.2 prikazuje dejanski rezultat algoritma z najboljšo kombinacijo parametrov konstant c_p, c_t in MIN_LENGTH. Omejitve najdaljših horizontalnih in vertikalnih razdalj med notami nismo upoštevali.

Linearizirana je bila najprej leva roka. V postopku so se pojavili naslednji tipi napak:

1. Večji skoki (na sliki je označen skok s 7. na 8. noto v levi roki) s podano kombinacijo c_p in c_t niso bili zaznani kot del istega segmenta, ampak je bil začet nov. Z zmanjšanjem vrednosti c_p bi število zaznanih skokov sicer povečali, a bi se povečalo število napačno razvrščenih not po segmentih;

2. Na več mestih se zgodi, da imata dva ali več segmentov isto noto najbližjo, tako da segment, ki je analiziran prvi, dobi najboljše kombinacije not, drugi segmenti pa le „ostanke“;

3. Segmenti se križajo, kar v naši skladbi ni pravilno.

Algoritem deluje dobro pri skladbah, kjer so glasovi bolj oddaljeni, v opisanem primeru pa je neuporaben. Vsi trije omenjeni tipi napak so posle-

dica zaporedne gradnje segmentov. Namesto iskanja najbližje sosednje note bi moral biti naš cilj najti kombinacijo najdaljših disjunktnih segmentov not (maksimizacija) s skupno najmanjšo vsoto razdalij med notami $d(n_i, n_j)$ (minimizacija). Problem predstavlja formula (4.2), kjer so s segmenti in funkcija $c(n)$ utežnostna funkcija števila segmentov. Od te funkcije je odvisno, kakšno bo razmerje med dolžino segmentov v primerjavi z vsoto razdalij med notami, ki jih vsebujejo.

$$\arg\max_{s_1 \ldots s_n} \left(\frac{|s_1| + |s_2| + \ldots + |s_n|}{c(n) \cdot \sum_{i=1}^{n} \sum_{\substack{j=0 \\ k=j+1}}^{|s_i|-1} d(n_j, n_k)} \right) \tag{4.2}$$

4.5.2 Vzporedno iskanje skupno najbližjih sosedov

Da bi rešili problem, opisan s formulo (4.2), smo prilagodili algoritem 1:

1. Trenutna nota n se spremeni v seznam trenutno aktivnih not N_c;
2. Posledično se referenca na trenutni element n spremeni v trenutni absolutni čas t;
3. Za vsak glas je vedno potreben le en prehod.

V algoritmu 2 lahko vidimo ustrezne prilagoditve. Za razliko od prvega ta deluje „vzporedno“ znotraj enega glasu. S_a vsebuje seznam trenutno aktivnih segmentov, N_c pa seznam naslednjih kandidatk not za segmente. Konstanta MAX_REST opisuje najdaljši dovoljen čas med dvema notama v istem segmentu.

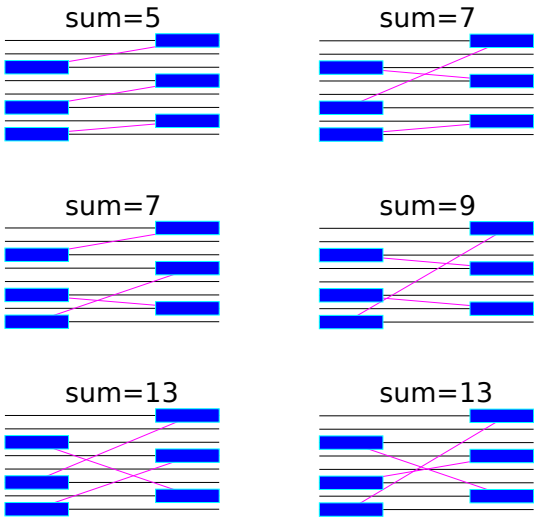
V algoritmu smo uporabili naslednje funkcije:

- $nextTime(t, v)$: Vrne čas začetka naslednje note v podani časovni rezini t in glasu v.
- $getNotesAt(t, v)$: Vrne seznam vseh not v podani časovni rezini t in glasu v.
- $time(n)$: Vrne čas začetka podane note n.
- $isValid(s)$: Preveri, če je segment veljaven (dovolj not v segmentu ipd.).

Algoritem 2: Algoritem za vzporedno iskanje skupno najbližjih sosedov.

Input: V: seznam glasov
Output: S: seznam segmentov not

```
1  foreach v ∈ V do
2      t = nextTime(−1, v);
3      foreach n ∈ getNotesAt(t, v) do
4          Dodaj nov segment v S_a z noto n
5      end
6      while S_a ≠ ∅ do
7          t = nextTime(t, v);
8          N_c = getNotesAt(t, v);
9          foreach s ∈ S_a do
10             if time(Last(s))−t > MAX_REST ∨ N_c = ∅ then
11                 if isValid(s) then
12                     Dodaj s v S;
13                 end
14                 Odstrani s iz S_a;
15             end
16         end
17         Dodaj kandidatke N_c najbližjim segmentom S_a;
18         if Len(N_c) > Len(S_a) then
19             ∀n ∈ N_c ∧ n ∉ S_a : Dodaj nov segment v S_a z noto n;
20         end
21     end
22 end
```

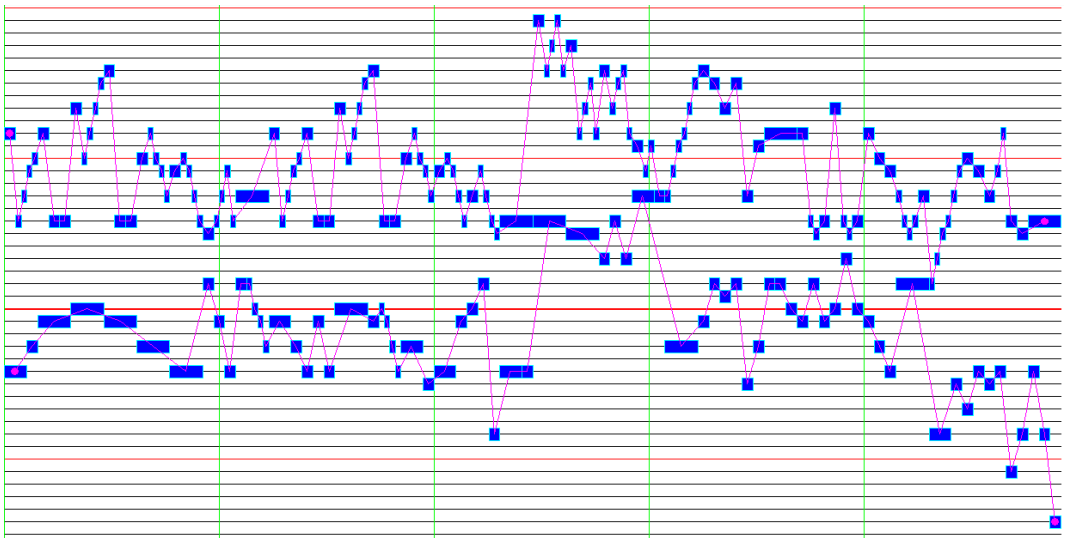


Slika 4.3: Primer permutacij povezav z njihovimi vsotami v poltonih za 3 segmente na levi s tremi kandidatkami not na desni.

Časovna rezina v funkciji $getNotesAt(t, v)$ se obnaša kot pas okoli podanega časa t. To pomeni, da se vse note v tem pasu obravnavajo, kot da bi bile zaigrane sočasno. Ta pristop je uporaben pri segmentaciji posnetkov v živo, saj izvajalci nikoli ne igrajo vseh not akorda sočasno. Za širino pasu se je izkustveno najbolje obnesla vrednost 50 ms. Ta širina je bila še dovolj ozka, da so bile kratke note zaznane.

Časovno zahtevnost algoritma sestavljata sprehod po notah vseh glasov in funkcija, ki razvrsti kandidatke not N_c v trenutno aktualne segmente S_a (algoritem 2, vrstica 17). Ta za vse možne permutacije naslednjih not izračuna vsote razdalj med zadnjimi notami v aktualnih segmentih in kandidatkami ter izbere permutacijo not z najmanjšo vsoto. Na sliki 4.3 so prikazane možne permutacije povezav in njihove vsote razdalj v poltonih za 3 segmente s tremi kandidatkami not. Časovna zahtevnost funkcije, ki razvrsti note po segmentih, je $m!$, kjer je m število vzporednih segmentov v skladbi. Časovna zahtevnost celotnega algoritma je $O(m!\frac{n}{m}) = O((m-1)!n)$. Podobno kot pri algoritmu za iskanje najbližjih sosedov v globino se tudi tu časovna zahtevnost zmanjša za faktor števila glasov, če je število not po glasovih približno enako.

Na sliki 4.4 so predstavljeni rezultati linearizacije skladbe iz prejšnjega poglavja z algoritmom za vzporedno iskanje najbližjih sosedov. Linearizacija je optimalna. Prisotna sta dva segmenta — desna roka igra zgornjega, leva pa spodnjega.



Slika 4.4: Note, razvrščene po segmentih s pomočjo algoritma za vzporedno iskanje najbližjih sosedov.

Poglavje 5

Računalniška analiza tem v skladbi

V poglavju 2.3 smo definirali temo in motiv. Povedali smo, da je tema daljša misel, sestavljena iz več krajših figur ali motivov. Kasneje, v poglavju 3, je bilo predstavljenih nekaj pristopov za zaznavo glavne teme v skladbi. Ugotovili smo, da je težava omenjenih pristopov ta, da ne predstavljajo celovite rešitve za zaznavo in analizo tem v večglasni skladbi na vhodu in seznama, zgradbe, pojavitev in podobnosti tem in motivov na izhodu. Rešitve delujejo le na izbranih melodijah skladb in kot izhod ponudijo najbolj primerno temo, ne pa tudi razčlembe posameznih tem na motive in njihovo analizo.

V ta namen smo zasnovali nov, enostavnejši pristop za analizo tem in motivov. Temelj našega pristopa sloni na naslednjih predpostavkah o definiciji teme:

- iščemo dovolj dolge vzorce not in pavz,
- vzorci se večkrat ponovijo,
- vzorci so dovolj zanimivi.

Enostavnost se kaže v tem, da za razliko od dosedanjih rešitev naš pristop ne upošteva taktovskega načina in metruma, jakosti, glasu, harmonije in drugih elementov v skladbi. Večglasno skladbo na vhodu najprej linearizi-ramo s postopkom, opisanem v poglavju 4. V teh segmentih nato poiščemo dovolj dolge vzorce not, ki se večkrat ponovijo in jih ocenimo glede na njihovo zanimivost.

V nadaljevanju bomo spoznali podatkovne strukture številsko drevo, priponsko drevo in PAT drevo. Nato bomo predstavili obliko priponskega drevesa

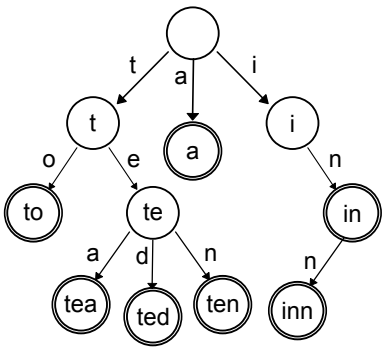
z ustrezno abecedo, primernega za zapis notnih parov in tem z motivi ter uporabljeno funkcijo za ocenjevanje in rangiranje tem.

5.1 Podatkovna struktura

V naslednjih podpoglavjih bomo spoznali formalne podatkovne strukture številsko drevo, priponsko drevo in drevo PAT. Predstavljeni priponsko in PAT drevesi bota vsebovali nekaj razširitev prvotne strukture za lažjo kasnejšo uporabo za zapis glasbe.

5.1.1 Številsko drevo

Številsko drevo (*trie*) je podatkovna struktura, prvotno namenjena zapisu množice besed [7]. Omogoča učinkovito iskanje, urejanje in izpis. Na sliki 5.1 je primer številskega drevesa za besede „to“, „tea“, „ted“, „ten“, „a“, „in“ ter „inn“. Drevo je zgrajeno iz vozlišč, ki so lahko končna (na sliki so dvakrat obkrožena) ali pa notranja. Povezave med vozlišči predstavljajo črke abecede Σ.



Slika 5.1: Številsko drevo za besede „to“, „tea“, „ted“, „ten“, „a“, „in“ ter „inn“.

5.1.2 Operacije

Gradnja številskega drevesa se začne pri korenu. Od leve proti desni za vsako črko besede izberemo obstoječo povezavo s to vrednostjo ali dodamo novo, če take povezave še ni. Postopek rekurzivno ponavljamo od naslednje črke

Tabela 5.1: Časovne zahtevnosti operacij posameznih implementacij vozlišč številskih dreves.

	iskanje	**vstavljanje**	**sprehod**
kazalčni seznam ali neurejeno polje	$O(\|\Sigma\|)$	$O(1)$	$O(1)$
zgoščevalne tabele	$O(1)$	$O(1)$	$O(\|\Sigma\|)$
uravnoteženo drevo	$O(log\|\Sigma\|)$	$O(log\|\Sigma\|)$	$O(1)$
urejeno polje	$O(log\|\Sigma\|)$	$O(\|\Sigma\|)$	$O(1)$
zgoščevalne tabele + kazalčni seznam naslednikov	$O(1)$	$O(1)$	$O(1)$

in vozlišča dalje. Ko pridemo do konca besede, vozlišče označimo za končno. Če celoten postopek ponovimo za vsako besedo iz besedila, v številsko drevo shranimo celotno množico besed.

Iskanje vsebovanosti določene besede poteka od korena proti listom. Za vsako črko iskane besede sledimo ustreznim povezavam. Če nas konec iskane besede privede do končnega vozlišča, je ta beseda vsebovana v množici. V kolikor pa med iskanjem ni ustreznega vozlišča ali pa končamo na vozlišču, ki ni končno, besede v množici ni.

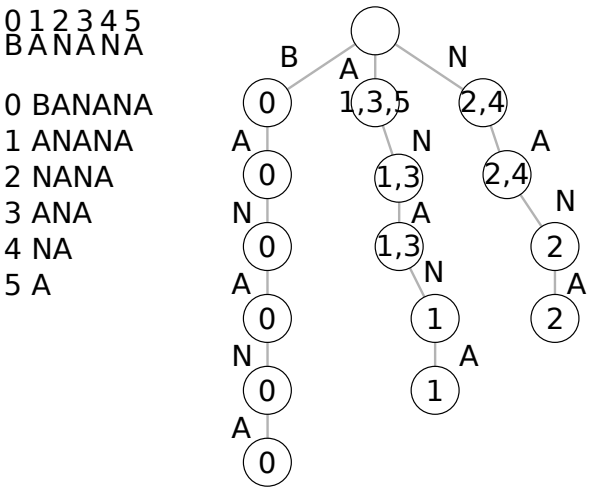
Zapis naslednikov v vozlišču številskega drevesa je običajno implementiran v obliki slovarja poddreves, če je $|\Sigma|$ velika, sicer pa v obliki polja fiksne dolžine $|\Sigma|$. Tabela 5.1 prikazuje časovne zahtevnosti operacij posameznih implementacij vozlišč številskega drevesa.

Kazalčni seznam je implementacija, pri kateri vsako vozlišče vsebuje referenci na svojega prvega naslednika na naslednjem nivoju in na naslednika na istem nivoju. Pri zgoščevalnih tabelah preslikamo črko na naslov, kjer se nahaja naslednik na naslednjem nivoju. Tu je upoštevana popolna preslikava brez podvajanj. Uravnoteženo iskalno drevo vsebuje informacijo o naslednikih na naslednjem nivoju. Vsako vozlišče ima svoje iskalno drevo. Urejeno polje s pomočjo bisekcije dostopa do želene črke in naslova naslednika na naslednjem nivoju. Kombinacija zgoščevalnih tabel in seznama naslednikov pa poleg zgoščevalnih tabel vsebuje še seznam vseh črk, ki so prisotne na določenem nivoju, in s tem pohitri sprehod po drevesu.

Prostorska zahtevnost številskega drevesa za besedilo, ki vsebuje n znakov, je $O(n)$. Teoretično najslabši primer je opis Fibonaccijeve besede [42], kar zahteva $2n$ vozlišč. V praksi zapis vozlišč in povezav potrebuje med 10 in 20-kratno količino pomnilnika glede na dolžino izvornega besedila. Obstaja več optimizacij, ki zmanjšajo to konstanto približno na tretjino prvotne porabe pomnilnika [10, 14].

5.1.3 Priponsko drevo

Priponsko drevo je podatkovna struktura, ki se uporablja za obdelavo in iskanje po nizih [28, 9]. Strukturo lahko izpeljemo iz številskega drevesa tako, da za vhod vzamemo vse priponske besede podanega besedila.



Slika 5.2: Priponsko drevo, zgrajeno za besedo BANANA.

Na sliki 5.2 je prikazano priponsko drevo, zgrajeno za besedo BANANA. Gradnja priponskega drevesa se začne tako, da prvotno besedilo razdelimo na priponske besede. V našem primeru jih je 6: BANANA, ANANA, NANA, ANA, NA in A. Vsako besedo vstavimo v drevo, podobno kot vstavljamo besede v številsko drevo. V diplomskem delu smo priponsko drevo dodatno razširili s tem, da vsakemu obiskanemu vozlišču dodamo referenco na pojavitev priponske besede v prvotnem besedilu. Prvotno priponsko drevo vsebuje reference priponskih besed le v listih drevesa.

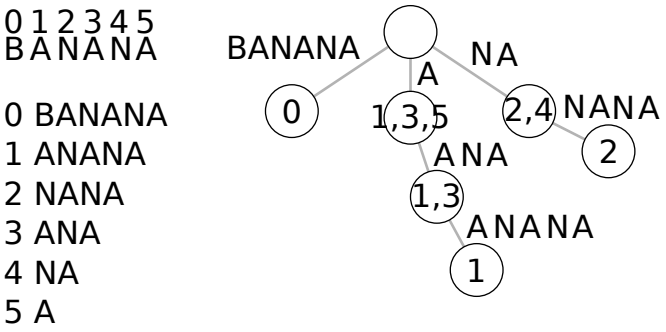
Iskanje niza v priponskem drevesu poteka s sprehodom po drevesu od korena proti listom, kjer za vsako i-to črko v iskanem nizu izberemo na i-tem nivoju ustrezno vozlišče. Če takega vozlišča ni, tudi podanega niza ni v besedilu. Če je končno vozlišče list, je iskani niz končnica vsaj ene besede. Če pa končno vozlišče ni list drevesa, iskani niz ni končnica nobene besede, je pa prisoten znotraj vsaj ene.

Pogosta operacija je iskanje mest pojavitev zahtevanega niza. Zaradi razširitve drevesa z dodajanjem referenc vsakemu vozlišču na prvotno besedilo je ta postopek trivialen. Tovrstna operacija je še posebej uporabna, če želimo prila-

gajati iskani niz glede na želeno število zadetkov.

5.1.4 Stiskanje poti (PAT)

Na sliki 5.2 je prikazano priponsko drevo, ki vsebuje eno črko na povezavo. Poleg takega pa se je v istem obdobju razvil še krajši in bolj berljiv zapis vozlišč, imenovan PATRICIA (*Practical Algorithm To Retrieve Information Coded In Alphanumeric*) ali krajše, PAT drevo [16]. Osnovno priponsko drevo spremenimo v PAT tako, da vozlišče z le enim naslednikom združimo v eno vozlišče. Namesto ene črke na povezavo tako lahko dobimo od 1 do n znakov dolg niz na enem vozlišču, kjer je n najdaljša dolžina besede izvornega besedila. Namen PAT dreves je z zmanjšanjem števila vozlišč za konstanto zmanjšati časovno in prostorsko zahtevnost prvotnih priponskih dreves. Slika 5.3 prikazuje PAT drevo za besedo BANANA.



Slika 5.3: PAT drevo, zgrajeno za besedo BANANA.

Operacijo združevanja vozlišč storimo tako, da za vsa vozlišča z enim naslednikom črko na povezavi do naslednika pridružimo povezavi v trenutnem vozlišču. Ko na tak način predelamo drevo, uvedemo v diplomskem delu zaradi lažje kasnejše uporabe strukture še dodatno razširitev: v smeri od korena do listov prvotnim črkam vseh povezav vrinemo še črke starševskih povezav. Na sliki se ta razširitev lahko opazi v sredinski poti A-ANA-ANANA in desni poti NA-NANA. Poti prvotnega PAT drevesa bi se glasili A-NA-NA in NA-NA.

5.2 Zapis glasbe v priponskem drevesu

Zapisovanje notnega zapisa glasbe v priponskem drevesu prinaša naslednje prednosti:

- učinkovito iskanje po nizih not,
- vozlišča priponskega drevesa nam predstavljajo nize not s številom ponovitev,
- zgradba priponskega drevesa nam prikaže zgradbo nizov z njihovimi podnizi,
- gradnjo priponskega drevesa lahko optimiziramo glede na želene dolžine nizov ali število ponovitev.

V diplomskem delu smo s pomočjo priponskih dreves glasbo zapisali v dveh korakih. Najprej smo zgradili v prejšnjem poglavju opisano razširjeno priponsko drevo. Za besedilo smo vzeli segmente not, opisane v poglavju 4.5. Vsak segment smo pred gradnjo drevesa pretvorili iz niza not v niz notnih parov. Dobljeno drevo smo poimenovali priponsko drevo notnih parov, saj za črke vsebuje notne pare. To drevo smo nato pretvorili v razširjeno PAT drevo. Poimenovali smo ga priponsko drevo tem, saj vozlišča predstavljajo nize notnih parov oz. not, ki predstavljajo teme, motive ali druge figure.

Razlog, da smo se odločili za priponsko drevo in ne za predponsko, je v tem, da so začetki (glave) tem praviloma enaki, medtem ko se razlike začnejo pojavljati v končnicah. Značilnost priponskega drevesa je, da ima bliže korena začetke besed, kasneje pa se drevo glede na različnost nizov v nadaljevanju besed razveji. Ker vemo, da so glave istih tem vedno enake, se zato ti začetki združijo v ista vozlišča, zaključki pa razvejijo v več vozlišč.

5.2.1 Priponsko drevo notnih parov

Priponsko drevo notnih parov za črke (povezave) uporablja notne pare. Vozlišče drevesa definiramo z naslednjimi atributi:

- reference na konkretne pojavitve notnih parov v skladbi (v besedilu priponskega drevesa),
- interval med notama,
- ritmično razmerje med notama.

Gradnja priponskega drevesa notnih parov je odvisna od primerjalne funkcije notnih parov. Ta deluje po principu ujemanja notnih parov po ritmu in/ali višinah not.

Primerjalna funkcija notnih parov

Ritmičnega ujemanja lahko ni ali pa je relativno oz. absolutno. Če je ritmično ujemanje izključeno, bo primerjalna funkcija notnih parov delovala le na podlagi melodičnega ujemanja not. Ta možnost je uporabna, če dolžine uvoženih not vsebujejo veliko šuma (npr. snemanje v živo).

Relativno ujemanje opisuje enačba (5.1) in pomeni, da bosta notna para enaka, če bosta imela enako ritmično razmerje med prvo in drugo noto. Ta možnost daje običajno najboljše rezultate, saj skladatelji v isti skladbi temo velikokrat zapišejo v enkrat daljših ali krajših notah.

$$f(< n_{11}, n_{12} >, < n_{21}, n_{22} >) = \begin{cases} 1 & \frac{n_{11}}{n_{12}} = \frac{n_{21}}{n_{22}} \\ 0 & \text{sicer} \end{cases} \tag{5.1}$$

Absolutno ritmično ujemanje notnih parov opisuje enačba (5.2). Notna para bosta enaka natanko takrat, ko bodo ritmične vrednosti prvih in drugih not identične.

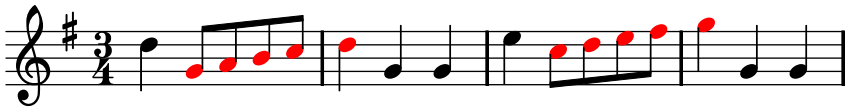
$$f(< n_{11}, n_{12} >, < n_{21}, n_{22} >) = \begin{cases} 1 & n_{11} = n_{21} \wedge n_{12} = n_{22} \\ 0 & \text{sicer} \end{cases} \tag{5.2}$$

Melodičnega ujemanja notnih parov lahko ni (upoštevano bo le ritmično ujemanje) ali pa je relativno oz. absolutno. Melodično ujemanje izključimo, ko želimo analizirati izključno ritmične vzorce skladbe.

Poznamo tri vrste relativnega in eno absolutno melodično ujemanje:

- ujemanje po kvantiteti intervala,
- ujemanje po kvantiteti in kvaliteti intervala,
- ujemanje po razdalji v poltonih,
- absolutno melodično ujemanje.

Melodično ujemanje notnih parov po kvantiteti intervala opisuje enačba (5.3). Notna para bosta enaka, če bo enaka kvantiteta njunih intervalov. Ta je lahko pozitivna ali negativna, odvisno od smeri gibanja. Možnost ujemanja notnih parov po kvantiteti je najpogostejša izbira, če so predznaki not pravilno



Slika 5.4: Primer sekvence sestavljene iz štirih notnih parov v začetku skladbe Menuet za klavir št. 1 v G-duru J. S. Bacha.

zaznani pri uvozu iz zapisa MIDI. Omogoča zaznavanje iste teme, tudi če gre za sekvenco in so kvalitete intervalov različne. Primer sekvence, ki je sestavljena iz dveh istih motivov (štirje notni pari) in jo primerjalna funkcija zazna, je prikazan na sliki 5.4.

$$f(< n_{11}, n_{12} >, < n_{21}, n_{22} >) = \begin{cases} 1 & qnt(n_{11}, n_{12}) = qnt(n_{21}, n_{22}) \\ 0 & \text{sicer} \end{cases} \tag{5.3}$$

Melodično ujemanje notnih parov po kvantiteti in kvaliteti intervala opisuje enačba (5.4). Notna para bosta enaka, če bosta enaki obe, kvantiteta in kvaliteta njunih intervalov. Ta možnost je uporabna, če iščemo ponovitve tem le v določenem spolu tonalitete (npr. izključno dur ali mol), ob pogoju, da so predznaki not pri uvozu iz zapisa MIDI pravilno zaznani.

$$f(< n_{11}, n_{12} >, < n_{21}, n_{22} >) = \begin{cases} 1 & qnt(n_{11}, n_{12}) = qnt(n_{21}, n_{22}) \wedge \\ & qlt(n_{11}, n_{12}) = qlt(n_{21}, n_{22}) \\ 0 & \text{sicer} \end{cases} \tag{5.4}$$

Melodično ujemanje notnih parov po poltonih opisuje enačba (5.5). Notna para bosta enaka, če bosta enaki števili njunih poltonov. Ta možnost je najbolj robustna od omenjenih melodičnih ujemanj notnih parov in deluje tudi ob napačno zaznanih predznakih not ali takrat, ko tonalitete sploh nimamo (atonalna glasba).

$$f(< n_{11}, n_{12} >, < n_{21}, n_{22} >) = \begin{cases} 1 & midipitch(n_{11}) - midipitch(n_{12}) = \\ & midipitch(n_{21}) - midipitch(n_{22}) \\ 0 & \text{sicer} \end{cases} \tag{5.5}$$

Absolutno melodično ujemanje notnih parov opisuje enačba (5.6). Notna para bosta enaka, če sta enaka po kvantiteti in kvaliteti intervala ter se začneta na imensko isti noti ne glede na oktavo. Ta možnost je uporabna, ko želimo

najti identično ponovitev teme. Primer je analiza sonatnega stavka, pri katerem nas zanima, kolikokrat se ponovi tema v glavnem stavku in ne želimo najti moduliranih tem v prehodnem delu.

$$f(< n_{11}, n_{12} >, < n_{21}, n_{22} >) = \begin{cases} 1 & qnt(n_{11}, n_{12}) = qnt(n_{21}, n_{22}) \wedge \\ & qlt(n_{11}, n_{12}) = qlt(n_{21}, n_{22}) \wedge \\ & notename(n_{11}) = notename(n_{21}) \\ 0 & \text{sicer} \end{cases} \tag{5.6}$$

Gradnja

Gradnja priponskega drevesa notnih parov poteka drugače od običajne gradnje priponskega drevesa, opisane v poglavju 5.1.3. Namesto, da bi drevo gradili po posameznih besedah, ga gradimo po posameznih črkah vzporedno po vseh besedah. Razlog za to je v sprotni optimizaciji na podlagi najmanjšega števila ponovitev. Algoritem deluje tako, da notne pare uredi s pomočjo urejanja z vedri [35]. Nato za vsako vedro algoritem ustvari novo vozlišče z referencami na pojavitve konkretnih notnih parov v besedilu. Postopek se rekurzivno ponovi nad vsakim novim vozliščem tako, da za nove pare algoritem vzame desne sosede notnih parov v besedilu.

Algoritem 3: Algoritem za gradnjo priponskega drevesa notnih parov.

Input: S: segmenti not
Output: T: priponsko drevo notnih parov

```
1  initialNotePairs = nextNotePairs(∅, S);
2  return buildSubtree(initialNotePairs);
3  buildSubtree(notePairs)
4  begin
5  |  T = new NotePairSuffixTree(notePairs);
6  |  buckets = bucketSort(notePairs);
7  |  foreach bucket, bNotePairs ∈ buckets do
8  |  |  AppendChild(T,buildSubtree(nextNotePairs(bNotePairs, S)));
9  |  end
10 |  return T;
11 end
```

Algoritem 3 formalno opisuje zgoraj opisan postopek gradnje priponskega drevesa notnih parov. Bistvo algoritma se skriva v rekurzivni funkciji *buildSubtree*(*notePairs*). Ta za podane notne pare vrne priponsko drevo notnih parov. Za prvi klic te funkcije je potrebno zgraditi seznam vseh notnih parov po vseh segmentih not S. To nalogo opravi funkcija $nextNotePairs(notePairs, S)$. Za vhod vzame trenutne notne pare in celotne segmente not besedila, vrne pa desne sosede podanih notnih parov. Če podanih notnih parov ni, funkcija vrne vse notne pare vseh segmentov. Funkcija $bucketSort(notePairs)$ upošteva primerjalno funkcijo notnih parov in uredi podane notne pare po vedrih.

Poleg osnovnega algoritma smo uvedli še tri optimizacije:

1. Določimo najkrajšo dolžino teme. Funkcijo $nextNotePairs()$ prilagodimo tako, da pri gradnji začetnih notnih parov vrača le tiste, ki so oddaljeni do konca segmenta še vsaj MIN_THEME_LENGTH $-$ 1 notnih parov.

2. Določimo najdaljšo dolžino teme. Funkcijo *buildSubtree* spremenimo tako, da ji kot dodatni parameter podamo trenutno globino. Če je ta daljša od MAX_THEME_LENGTH, se rekurzivni klic ne izvede.

3. Določimo najmanjše število ponovitev. V vrstici 8 ne izvedemo rekurzivnega klica, če število elementov v $bNotePairs$ ne presega konstante MIN_THEME_OCCURANCES.

Z zgoraj naštetimi optimizacijami omejimo velikost drevesa v primeru analize daljše skladbe z veliko segmenti.

5.2.2 Priponsko drevo tem

Priponsko drevo tem je priponsko drevo, katerega povezave predstavljajo teme, motivi in druge figure. Vsako vozlišče ima naslednja atributa:

- reference na konkretne pojavitve tem v skladbi,
- ocena teme.

Priponsko drevo tem zgradimo iz priponskega drevesa notnih parov po principu pretvorbe priponskega drevesa v PAT drevo, opisanem v poglavju 5.1.4. Algoritem 4 se sprehodi od korenskega vozlišča priponskega drevesa notnih parov proti listom in doda ustrezno vozlišče v priponsko drevo tem, če:

1. je trenutno vozlišče priponskega drevesa tem še korensko vozlišče,
2. trenutno vozlišče notnih parov nima le enega naslednika,
3. je število referenc na besedilo med notnim parom in njegovim naslednikom različno.

Zadnji pogoj se nanaša optimizacijo MIN_THEME_OCCURANCES pri gradnji priponskega drevesa notnih parov, opisano v prejšnjem poglavju. Zaradi te lahko pride do rezanja poddreves (*prunning*), zato je potrebno takšna vozlišča v priponskem drevesu tem ločiti. Če ni izpolnjen nobeden od zgornjih pogojev, se trenutnemu vozlišču priponskega drevesa tem podaljša temo za en notni par.

Rekurziven del algoritma predstavlja procedura $buildThemesTree()$. Ta potrebuje dva parametra: trenutno priponsko poddrevo notnih parov T in trenutno priponsko poddrevo tem TT. Funkcija $isNotRootNode()$ preveri, če je podano vozlišče korensko vozlišče celotnega drevesa. Lastnost *children* predstavlja vse naslednike podanega vozlišča, $references$ pa vse reference na konkretne note v besedilu. Funkcija $appendCharacter(TT, newChar)$ podaljša temo, ki jo predstavlja povezava v TT, za en znak $newChar$.

Posebnih optimizacij za pretvorbo priponskega drevesa notnih parov na priponsko drevo tem nismo uporabili.

Algoritem 4: Algoritem za gradnjo drevesa tem.

Input: T: priponsko drevo notnih parov
Output: TT: priponsko drevo tem

```
1  TT = new ThemeSuffixTree();
2  return buildThemesTree(T, TT);

3  buildThemesTree(T, TT)
4  begin
5      if isNotRootNode(T) ∧
       |T.children| = 1 ∧
       |T.references| = |T.children[0].references| then
6          appendCharacter(TT, T.children[0]);
7          buildThemesTree(TT, T.children[0]);
8      end
9      else
10         foreach subT ∈ T.children do
11             subTT = new ThemeSuffixTree(subT);
12             buildThemesTree(subT, subTT);
13             AppendChild(TT, subTT);
14         end
15     end
16 end
```

5.3 Ocenjevanje tem

V prejšnjih poglavjih smo zgradili drevo tem glede na njihovo zgradbo. Teme blizu korena so krajše in se velikokrat ponavljajo, medtem ko se ob premikanju proti listom iste teme daljšajo, število njihovih pojavitev pa pada. Pri ocenjevanju posamezne teme uporabimo naslednje parametre:

- dolžino,
- število pojavitev,
- zanimivost.

Dolžina teme je definirana kot število notnih parov v temi +1. Število pojavitev teme predstavlja število referenc na sezname notnih parov v skladbi.

Zanimivost teme smo določili glede na njeno melodično in ritmično razgibanost. To smo definirali z normalizirano entropijo intervalov in ritmičnih razmerij. Enačba (5.7) opisuje normalizirano entropijo [23]. Funkcija p predstavlja verjetnost istega intervala ali ritmičnega razmerja v vseh notnih parih teme, n pa število notnih parov v temi.

$$H_O(p_1...p_n) = \frac{H}{H_{max}} = -\sum_{i=1}^{n} \frac{p_i \cdot log\ p_i}{log\ n} \tag{5.7}$$

Enačba (5.8) prikazuje celotno ocenitveno funkcijo za temo. Sestavljena je iz ocen za dolžino teme, števila ponovitev in normaliziranih entropij melodije in ritma. Spremenljivke funkcije so dolžina teme n, število pojavitev teme occ ter verjetnosti posameznih melodičnih (p_{mel}) in ritmičnih (p_{rhy}) dogodkov v temi. V enačbi opazimo, da se oceni za dolžino teme in število pojavitev množita, medtem ko se oceni za zanimivost teme seštevata. Konstanti B_{len} in B_{occ} predstavljata logaritemsko osnovo za računanje ocene teme glede na dolžino in število pojavitev. S parametroma nastavimo, katere dolžine tem in števila pojavitev nas najbolj zanimajo. Za logaritemsko funkcijo bi lahko izbrali tudi katero iz družine parabol, vendar se je za oceno logaritemska obnesla bolje, saj velikokrat ne vemo točno, kje naj ima parabola ekstrem. Posledica tega je, da so predolge ali prevečkrat pojavljene teme preslabo ocenjene. V praksi se logaritemski osnovi gibljeta okoli 20 za dolžino teme in 3 za število pojavitev. Konstante C predstavljajo razmerja pomembnosti posameznih ocen znotraj celotne ocene. C_{len} in C_{occ} vplivata na pomembnost dolžine teme in števila pojavitev. Lahko bi ju združili v eno konstanto, vendar ju zaradi boljše berljivosti v enačbi ločimo. Konstanti C_{mel} in C_{rhy} predstavljata pomembnost

melodične ali ritmične raznolikosti teme. Za uporabno razmerje se izkaže 60 proti 20+20 v prid dolžine teme oz. števila pojavitev.

$$\begin{aligned} f =& C_{len} \cdot log_{B_{len}} n \cdot C_{occ} \cdot log_{B_{occ}} occ + \\ & C_{mel} \cdot H_O(p_{mel1}...p_{meln}) + C_{rhy} \cdot H_O(p_{rhy1}...p_{rhyn}) \end{aligned} \tag{5.8}$$

S celotno ocenitveno funkcijo lahko za vsako temo izračunamo njeno oceno in jo rangiramo glede na ostale. Priponsko drevo lahko sploščimo in zgradimo seznam tem, urejenih po oceni. Glede na najbolje ocenjeno temo lahko izluščimo glavno temo v skladbi.

Poglavje 6

Izvedba orodja za analizo tem

Opisane algoritme za glasbeno analizo v prejšnjem poglavju smo realizirali v uporabniški aplikaciji. V ta namen smo ustanovili odprtokodni, eksperimentalni projekt *Harmonia* (harmonia.berlios.de). Ta vključuje brezplačno namizno aplikacijo za operacijske sisteme Linux, Windows in MacOS X in integracijo v program za pisanje not *Canorus*. Zaenkrat je uporabniški vmesnik le v angleškem jeziku, vendar je enostavno prevedljiv tudi v ostale jezike.

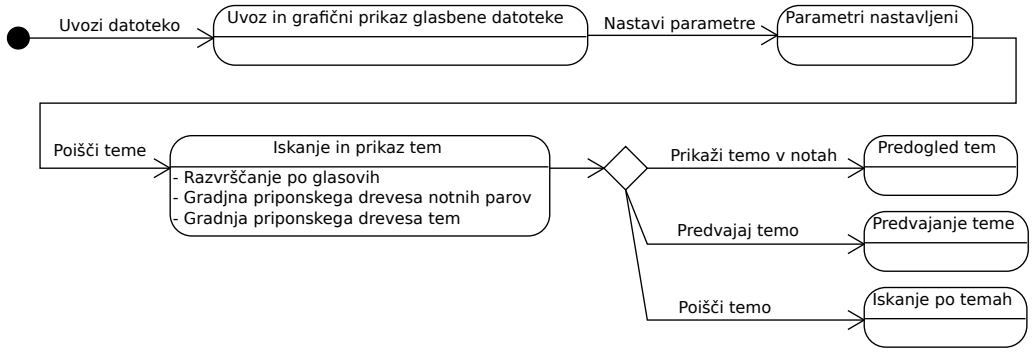
Harmonia je v prvi vrsti namenjena muzikologom in drugim, ki proučujejo melodične prvine skladbe. Sprva naj bi s pomočjo prijaznega uporabniškega vmesnika omogočila analizo (iskanje tem — kontrapunktska analiza, iskanje akordov in funkcij — harmonska analiza) in grafični prikaz glasbenih datotek. Kasneje pa bi Harmonia s pridobljenim znanjem analize glasbe nudila tudi pomoč pri kompoziciji novih skladb v programu za pisanje not. V sklopu diplomskega dela smo se osredotočili na modul za iskanje tem.

6.1 Načrtovanje

6.1.1 Funkcionalnosti in diagram poteka

Orodju za analizo tem smo najprej določili želene funkcionalnosti:

- uvoz glasbenih datotek v zapisu MIDI in MusicXML,
- grafični prikaz uvoženih datotek,
- linearizacija not,
- gradnja priponskega drevesa notnih parov in tem,



Slika 6.1: Diagram poteka uporabe aplikacije Harmonia.

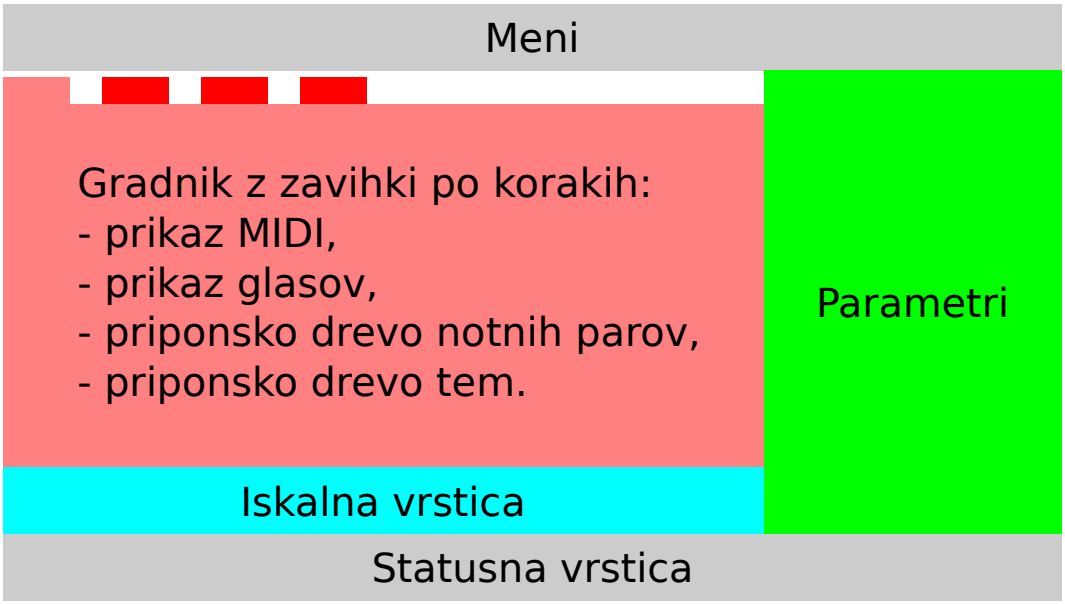
- nastavljanje vseh parametrov, potrebnih za uvoz, linearizacijo not in gradnjo priponskih dreves,
- grafični prikaz posameznih tem v notah,
- predvajanje skladbe kot celote ali njenih posameznih tem,
- iskanje po temah z vnosom imen not.

Za opisane funkcionalnosti smo zasnovali ustrezen diagram poteka (diagram 6.1). Uporabnik preko menija uvozi ustrezno glasbeno datoteko in prikaže se mu grafični prikaz uvožene skladbe. Še pred zagonom algoritma za linearizacijo not in analizo motivov se uporabniku prikažejo parametri, ki jih je mogoče nastaviti. Sledi zagon algoritmov in pregled rezultatov. Algoritme je mogoče ponovno zagnati z drugačnimi parametri. Pregled rezultatov omogoča predogled tem v notah, predvajanje in iskanje tem na način, kot smo ga uporabniki navajeni pri urejevalnikih besedil.

6.1.2 Uporabniški vmesnik

Pri snovanju uporabniškega vmesnika smo imeli v mislih predvsem dve načeli:

- nov uporabnik razume aplikacijo že ob prvi uporabi in
- zadoščeno je tudi naprednejšim uporabnikom, ki jih zanimajo tudi posamezni koraki analize.

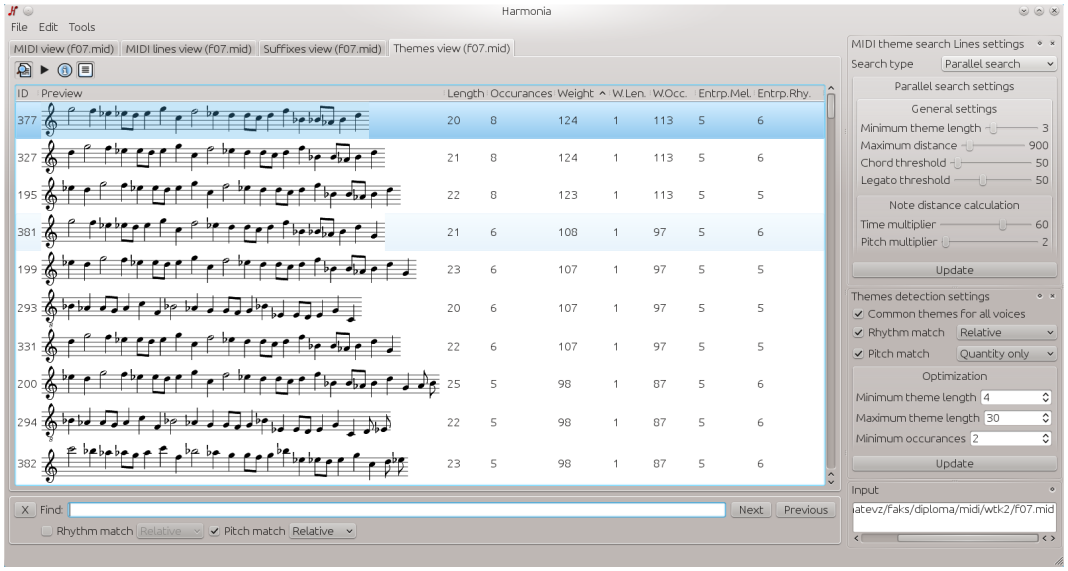


Slika 6.2: Predvidena zaslonska maska aplikacije.

Na sliki 6.2 lahko vidimo koncept uporabniškega vmesnika. V centralnem gradniku se nahaja vsebina analizirane skladbe. Da bi zadostili zgoraj omenjenim načelom, smo se odločili za uporabo gradnika z zavihki. Prvi zavihek vsebuje grafični prikaz uvožene glasbene datoteke. Drugi ta prikaz dopolni še s črtami, ki povezujejo note, razvrščene po posameznih glasovih. V tretjem zavihku se nahaja prikaz priponskega drevesa notnih parov, v četrtem pa priponsko drevo tem. Zavihki predstavljajo korake našega algoritma za iskanje tem. Četrti zavihek je privzet pogled na rešitev, ki nam jo ponudi aplikacija. Naprednejši uporabniki si lahko ogledajo tudi prejšnje zavihke analize in spremenijo določene parametre.

Na desni strani se nahajajo parametri algoritma za iskanje tem. Zaradi današnjih razmerij zaslonov 16:10 ali celo 16:9 smo postavili te parametre na desno, saj je horizontalnega prostora bistveno več kot vertikalnega. Aplikacija samodejno izpolni vrednosti parametrov z najprimernejšimi.

Na vrhu se nahaja meni, ki vsebuje postavke: nov dokument, shranjevanje, odpiranje, uvoz, izhod, iskanje, urejanje nastavitev aplikacije in zagon algoritma za iskanje. Spodaj se nahaja statusna vrstica, ki opisuje trenutno stanje, v primeru daljših operacij pa vsebuje tudi vrstico z napredkom.



Slika 6.3: Končna zaslonska maska aplikacije.

Po zgledu sodobnih brskalnikov se na spodnji strani centralnega gradnika za iskanje prikaže vrstica. Ta ponudi vnos iskanega niza, gumba Naslednji in Prejšnji zadetek in parametre za iskanje.

Dokončano zaslonsko masko aplikacije lahko vidimo na sliki 6.3.

6.1.3 Podatkovni model

Na sliki 6.4 je prikazan razredni diagram aplikacije. Zaradi odprtosti aplikacije smo podatkovni model zasnovali kar se da razširljivo.

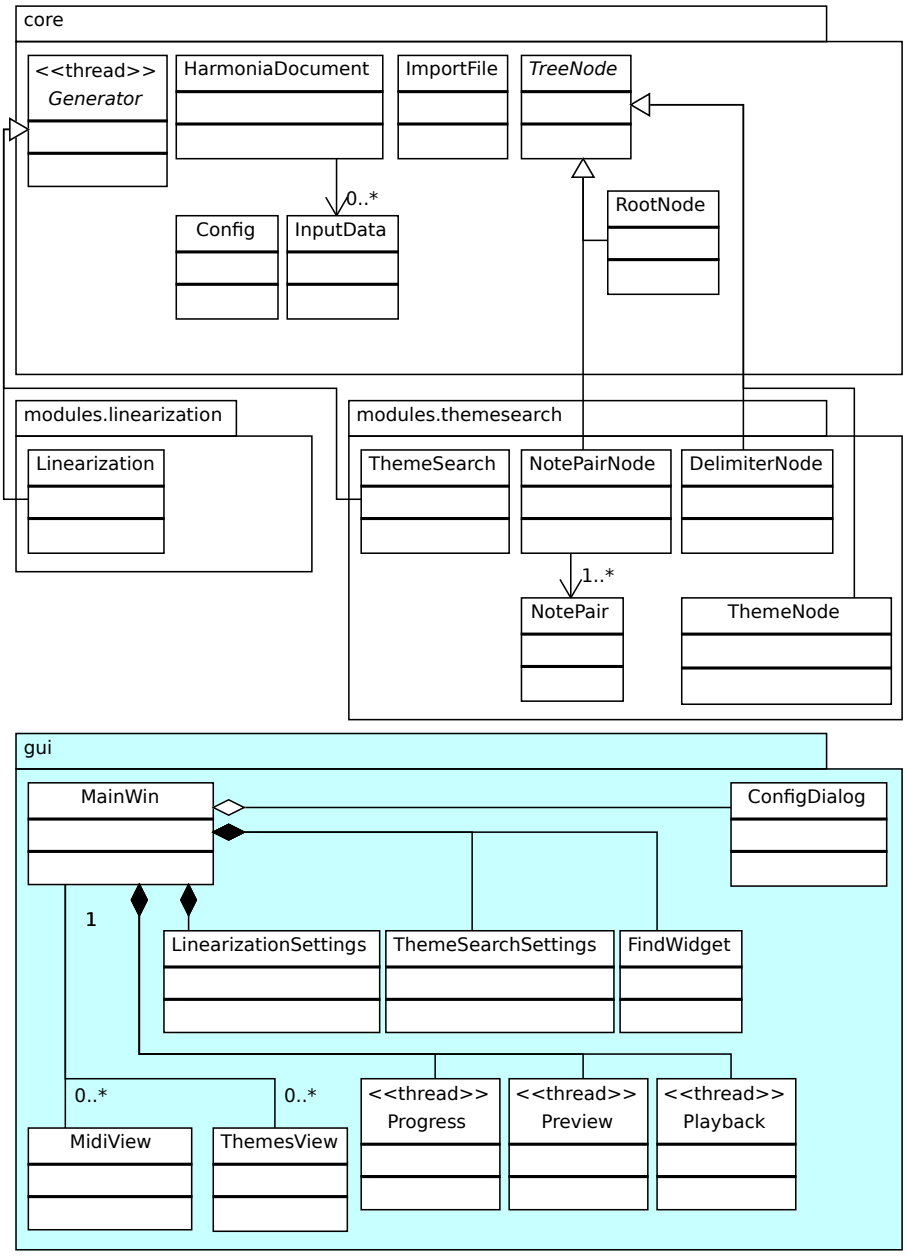
Jedrni del aplikacije se nahaja v paketu *core*. Sem spadajo abstraktni razred za generiranje vsebine, dokument aplikacije, abstraktno vozlišče drevesa s korenskim vozliščem, razred za delo z nastavitvami, razred za uvoz podatkov in abstraktni vhodni podatek.

Poleg jedrnega paketa aplikacije je prisoten paket *modules*, ki vsebuje podpaketa za linearizacijo not *linearization* in iskanje tem *themesearch*[1]. V prvem se nahaja le en razred, ki implementira algoritma za iskanje not po globini in vzporedno iskanje. V drugem pa se nahaja razred za gradnjo priponskega drevesa z razredom vozlišče, ki predstavlja notni par, temo in končno vozlišče

[1]Poleg modulov, ki zadevajo diplomsko delo, se v aplikaciji nahaja tudi modul za harmonsko analizo glasbe, vendar ga tu ne bomo omenjali.

(list drevesa). Oba glavna razreda podpaketov tečeta v svojih nitih in dedujeta razred za generiranje vsebine. Ta se nahaja v jedrnem paketu.

Uporabniški vmesnik aplikacije se nahaja v paketu *gui*. Tu je razred, ki predstavlja glavno okno aplikacije in se pokaže ob zagonu. Ta vsebuje gradnike s parametri algoritmov za linearizacijo not in analizo tem ter gradnik, namenjen iskanju po zadetkih. Glavno okno vsebuje več grafičnih prikazov uvožene datoteke in drevesna prikaza priponskih dreves notnih parov in tem. Poleg tega so v ozadju še razredi za prikaz trenutnega napredka, predogleda in za predvajanje. Ti tečejo v svojih nitih. Poleg glavnega okna obstaja še razred za prikaz konfiguracije aplikacije, kot so poti do različnih programov.



Slika 6.4: Razredni diagram UML za aplikacijo Harmonia. Razredi na modri podlagi so del grafičnega vmesnika.

6.1.4 Uporabljena tehnologija

Odločili smo se, da bo orodje za iskanje tem namizna aplikacija. Razlogi za razvoj nove aplikacije, ki ni spletna, so bili predvsem v hitrosti razvoja namiznih v primerjavi s spletnimi aplikacijami in zrelosti orodij.

Canorus (www.canorus.org) je odprtokodna namizna aplikacija za pisanje not. Njegov razvoj sega v leto 2006 in poleg drugih združuje tudi razvijalce starejšega programa za pisanje not *NoteEdit* (noteedit.berlios.de). *Canorus* je napisan v programskem jeziku C++ in teče pod operacijskimi sistemi Linux, Windows in MacOS X. Napisan je modularno in poleg grafičnega vmesnika vsebuje tudi ločen, zaledni del. Slednji med drugim vsebuje tudi podporo za programski jezik *Python* [41]. To pomeni, da lahko zunanje aplikacije, napisane v tem jeziku, uporabljajo notranje dele *Canorusa* in obratno — *Canorus* lahko vsebuje kodo, napisano v *Pythonu*. Prvo možnost smo v *Harmonii* izkoristili za pomoč pri uvozu in izvozu glasbenih zapisov, interpretaciji notnega zapisa ter računanju intervalov in ritmičnih vrednosti. Razredi, ki so del aplikacije Canorus, se začnejo s predpono `CA`.

Python (www.python.org) je zrel, visoko-nivojski, skriptni programski jezik. Prvič se je javno pojavil leta 1991, njegovi začetki pa segajo že v osemdeseta. Njegova filozofija temelji predvsem na čistosti in berljivosti kode, bogatem naboru že vgrajenih funkcionalnosti in podpori več paradigmam razvoja (poleg objektnega programiranja tudi funkcijsko in metaprogramiranje). Najpogosteje uporabljena implementacija jezika je odprtokodni *CPython*, napisan v programskem jeziku C.

Qt (qt.nokia.com) je obsežna, odprtokodna knjižnica za razvoj aplikacij v programskem jeziku C++. Pomembna je predvsem zaradi hitrosti izvajanja in majhni porabi pomnilnika, saj je že od začetka prilagojena tako namiznim računalnikom kot tudi mobilnim napravam. Z Nokiinim prevzemom knjižnice leta 2008 je ta podpora postala še izdatnejša [58]. Knjižnico poleg jedrnih razredov (npr. za delo z nizi, datotekami, nitmi, dinamičnimi lastnostmi objektov, XML) sestavlja še grafični vmesnik (z vgrajenim izgledom za Windows, KDE, Gtk, MacOS X in s podporo za OpenGL (www.opengl.org)), pogon za brskanje po spletu WebKit [73], modul za delo z mrežo [65], skriptni jezik ECMA-262 (bolj znan pod imenom JavaScript) [66, 69], orodje za gradnjo SQL stavkov ter povezava s podatkovno bazo [68] in orodjem za testiranje (več vrst testiranj enot, testiranje zmogljivosti, testiranje uporabniškega vmesnika, testiranje niti in podatkovnih tipov) [64]. Razredi, ki so del knjižnice Qt, se vedno začnejo s črko `Q`.

Posebni okolji za razvoj aplikacij v programskem jeziku *Python* z upo-

rabo knjižnice *Qt* sta *PyQt* (www.riverbankcomputing.co.uk/software/pyqt) in *PySide* (www.pyside.org). Obe spadata med prosto programje. Prvo je starejše, licencirano pod pogoji GPL [70], drugo pa mlajše (ustanovljeno je bilo z Nokiinim prevzemom knjižnice *Qt*) in je licencirano pod pogoji LGPL [45]. V diplomskem delu smo uporabili okolje *PyQt*.

6.2 Izvedba

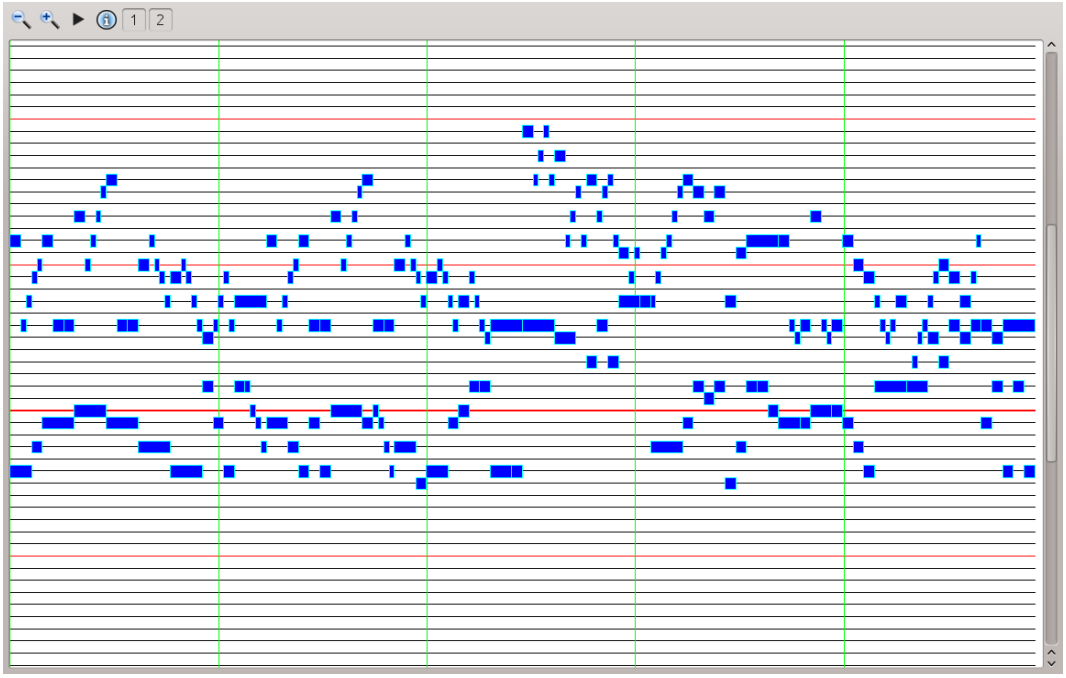
Na sliki 6.3 vidimo izgled naše aplikacije. Za centralni gradnik z zavihki smo uporabili gradnik *QTabWidget*, v katerega se dodajajo zavihki, ki vsebujejo bolj specifične gradnike za posamezni korak algoritma. Na desni strani smo za parametre izbrali niz gradnikov *QDockWidget*. Prednost teh je, da lebdijo in se lahko priključijo poljubnemu robu glavnega okna (privzeto so na desni). Ravno tako jih lahko skrijemo, če nam primanjkuje prostora na zaslonu, in ponovno prikažemo, ko jih potrebujemo.

6.2.1 Uvoz podatkov

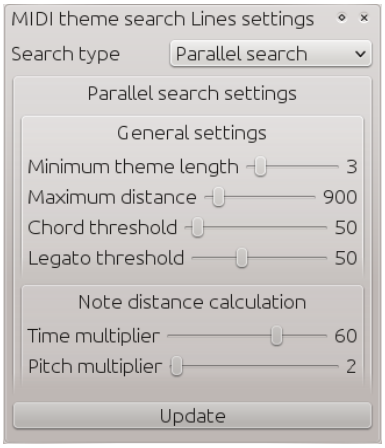
Način uvoza podatkov je odvisen od vrste zapisa datoteke. Če uvažamo podatke v zapisu MIDI, je najprej potrebna pretvorba zapisa MIDI v notni zapis. Če pa so podatki že v notnem zapisu (zapisa *MusicXML* ali *Canorus*), sledi le še grafični prikaz uvoženega dokumenta.

Uvoz datoteke MIDI se izvaja s pomočjo *Canorusovega* razreda `CAMidiImport`. Podrobnosti algoritma za pretvorbo zapisa so opisani v poglavju 4.4.

Drugi korak izriše note kot pravokotnike v 2D prostoru, kjer os X predstavlja čas, os Y pa višino v poltonski skali. Za pomoč pri pogledu poleg not izrišemo tudi vodoravne črte, ki predstavljajo poltone, podebelimo vse črte, ki predstavljajo višine C in vsakih 10 sekund dodamo navpične črte. Za izris uporabljamo gradnik `QGraphicsView`, ki je del knjižnice *Qt*. Ta gradnik za vhod vzame podatke o sceni iz razreda `QGraphicsScene` in jih glede na zmogljivosti grafičnega okolja in strojne opreme izriše pospešeno. Uporabili smo tudi že vgrajene matrične transformacije za premik po sceni in povečavo. Poleg tega smo dodali še možnost filtriranja izbranih kanalov MIDI, prikaz statistike (število not in inštrumenti po posameznih kanalih) in možnost predvajanja celotne skladbe. Na sliki 6.5 je grafični prikaz uvožene skladbe. Zgoraj opazimo gumbe za povečavo, predvajanje, prikaz podrobne statistike in za vklop in izklop kanalov MIDI.



Slika 6.5: Grafični prikaz skladbe Menuet za klavir št. 1 v G-duru J. S. Bacha.



Slika 6.6: Parametri algoritma za vzporedno iskanje najbližjih sosedov pri razvrščanju not po glasovih.

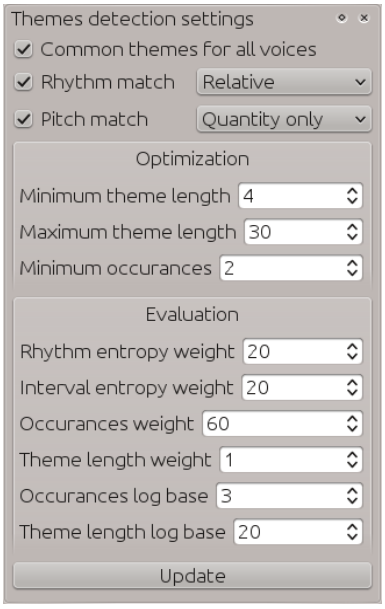
6.2.2 Linearizacija not

Po uvozu glasbene datoteke ima uporabnik možnost zagona algoritma za iskanje tem. Algoritem deluje nad seznamom not razvrščenih po glasovih, zato je potrebno z algoritmom, opisanem v poglavju 4.5, najprej najti segmente not. Na desni strani okna imamo vrsto parametrov. Prvi je izbira algoritma (iskanje najbližjih sosedov v globino ali vzporedno iskanje najbližjih sosedov), sledijo parametri specifični za tip iskanja, parameter za občutljivost na dolžino in višino not ter gumb za uveljavitev novih parametrov. Slika 6.6 prikazuje parametre za algoritem vzporednega iskanja najbližjih sosedov.

Po razvrstitvi not po glasovih se v centralni gradnik doda nov zavihek, ki prikazuje grafični prikaz skladbe z razvrščenimi notami po glasovih. To prikažemo tako, da med seboj povežemo pravokotnike (note), ki jih izvaja isti glas. Pogled prikazuje slika 4.4. Poleg prikaza se k statistiki skladbe doda podatek, koliko segmentov je bilo najdenih za vsak kanal MIDI.

6.2.3 Gradnja priponskega drevesa notnih parov

Po razvrstitvi not po glasovih imamo pripravljen vhod za gradnjo priponskega drevesa notnih parov. Ta z nekaj dopolnitvami deluje po algoritmu, opisanem v poglavju 5.2.1. Slika 6.7 prikazuje parametre za gradnjo priponskega drevesa na desni strani okna. Ti so združitev dreves vseh glasov, nastavitve primerjalne funkcije notnih parov, optimizacijski parametri in parametri ocenitvene



Slika 6.7: Parametri za gradnjo priponskega drevesa.

funkcije.

Če je združitev dreves vseh glasov vključena, bo aplikacija pred gradnjo priponskih dreves najprej združila vse glasove v en dolg glas. Posledica je, da se ponovitve tem v več glasovih upoštevajo kot iste teme. Tako delovanje je največkrat smiselno. Če je ta parameter izključen, se za vsak glas zgradi ločeno priponsko drevo, drevesa tem pa se nato združijo. Posledica je, da na koncu ne dobimo več priponskega drevesa, ampak lahko korensko vozlišče vsebuje več povezav z istimi črkami. Takšno delovanje je zaželeno, če nas zanimajo teme ali drugi vzorci v specifičnem glasu in bi nam drugi glasovi vnesli le več šuma.

Nastavitvi ritmičnega in melodičnega ujemanja vplivata na delovanje primerjalne funkcije, opisane v poglavju 5.2.1. Ritmično ujemanje je lahko izključeno, relativno ali absolutno. Pri melodičnem ujemanju pa izključeno, po kvantiteti, kvantiteti in kvaliteti intervala, po poltonih ali z absolutnim ujemanjem.

S parametri ocenitvene funkcije nastavimo konstante C_{rhy}, C_{mel}, C_{occ}, C_{len}, B_{len} in B_{occ}, opisane v enačbi (5.8). To so uteži entropije ritma in melodije ter uteži in logaritemska osnova števila pojavitev in dolžine tem.

6.2.4 Gradnja drevesa tem

Gradnja drevesa tem iz priponskega drevesa notnih parov poteka po algoritmu, opisanem v poglavju 5.2.2. Posebnih parametrov ni. Zaradi lažje analize rešitev smo za ID teme prevzeli ID prvega najplitvejšega unikatnega notnega para za to temo.

6.2.5 Prikaz priponskih dreves

Za prikaz drevesnega prikaza tem in notnih parov smo uporabili gradnik `QTreeWidget`. Ta omogoča možnost prikaza drevesno strukturiranih podatkov z več stolpci. Za vsak notni par prikažemo naslednje podatke: ID, predogled (opcijsko), globina, število pojavitev, kvantiteto intervala, kvaliteto intervala, število poltonov in ritmično razmerje not. Za vsako temo pa: ID, predogled (opcijsko), dolžino teme, število pojavitev, oceno in podrobnosti ocene (ocena iz dolžine teme, ocena iz števila pojavitev, entropija melodije, entropija ritma). Ob premiku miške na predogled se v obliki namiga izriše celoten predogled teme, če je stolpec v tabeli zanjo preozek, zaporedje intervalov v temi in reference na mesta v skladbi, kjer se tema pojavi. Ob kliku miške na naslov stolpca v tabeli, se podatki ustrezno uredijo po tem stolpcu (naraščajoče oz. padajoče ob ponovnem kliku).

Drevesni pogled poleg vklopa in izklopa predogleda ponuja še naslednje funkcionalnosti: predvajanje ali ustavitev predvajanja izbrane teme ali notnega para, statistične informacije o drevesu (število vozlišč in povezav) in preklop iz drevesnega pogleda v seznam. Slednje nam omogoča, da vse podatke uredimo po želenem stolpcu, saj v primeru drevesnega pogleda urejanje deluje le za vozlišča na posameznem nivoju. Slika 6.3 prikazuje drevesni pogled tem.

6.2.6 Predogled tem

Predogled notnih parov in tem se izvaja s pomočjo zunanjega programa LilyPond (www.lilypond.org). Nastavitve LilyPond se lahko urejajo v oknu za konfiguracijo aplikacije.

Postopek se izvaja v ločeni niti in je sledeč:

1. Iz vsebine gradnikov `QTreeWidget`, ki prikazujejo notne pare ali teme, se ustvari urejen seznam notnih parov ali tem. Notni pari so urejeni naraščajoče po času pojavitve (ID), teme pa padajoče po oceni.

2. Za vsak notni par ali temo se ustvari dokument (`CADocument`), črtovje (`CAStaff`) in glas (`CAVoice`) s prvo pojavitvijo notnega para ali teme v

skladbi. LilyPond samodejno doda tudi ključ tako, da se note izrišejo s čim manj pomožnimi črtami.

3. Ustvarjeni dokument se izvozi v sintakso LilyPond.

4. Izračuna se zgoščena vrednost (*hash*) dokumenta, izvoženega v LilyPond, in preveri, če predogled za ta dokument že obstaja.

5. Če predogleda še ni, se z ustreznimi parametri (generiranje slike PNG, rezanje robov) izvede aplikacija LilyPond in ustvari sliko PNG z imenom zgoščene vrednosti.

6. Nit signalizira, da je izvoz za notni par ali temo končan.

7. Glavna nit[2] uvozi sliko s predogledom in jo nastavi kot vsebino celice za predogled določenega notnega para ali teme.

6.2.7 Predvajanje

Predvajanje skladbe, notnih parov in tem se izvaja s pomočjo zunanjega programa. Privzet predvajalnik MIDI je programski predvajalnik Timidity (timidity.sf.net), vendar obstaja še mnogo drugih programskih in strojnih predvajalnikov za različne sisteme [43, 53, 61]. Nastavitve predvajalnika se lahko urejajo v oknu za konfiguracijo aplikacije.

Ob kliku na gumb „predvajaj“ se v ločeni niti zgodi naslednje:

1. Če je izbran pogled celotne skladbe, se zažene predvajalnik MIDI nad izvorno datoteko MIDI.

2. Če je izbran pogled notnih parov ali tem, se izbrani elementi s pomočjo razreda `CAMidiExport` izvozijo v zapis MIDI.

3. Sledi zagon predvajalnika MIDI nad izvoženo datoteko.

4. Ob zaključku predvajanja nit signalizira, da je predvajanje končano.

5. Glavna nit izklopi gumb za predvajanje.

Če med predvajanjem pride do izklopa gumba za predvajanje, se izvajanje niti zaključi (klic funkcije `terminate()`).

[2]Qt omogoča izris uporabniškega vmesnika le v glavni niti, ki se ustvari v konstruktorju `QApplication` [63].

Tabela 6.1: Podprta sintaksa LilyPond za vnos not.

`c d e f g a b r`	Gibanje not po najbližji poti od c do h in pavza. Pozor: Nota H je v jeziku LilyPond označena kot B.
`c g' c,`	Gibanje not v večjih skokih v obeh smereh. Primer vsebuje skok za kvinto navzgor in navzdol.
`0 1 2 4 8 16 32 64 128`	Notne dolžine od note brevis do 128-inke.
`cis ces cisis ceses as es bis bes`	Predznaki za višaje in nižaje. Pozor: Zvišana nota H je v jeziku LilyPond zapisana kot bis, znižana pa kot bes.
`2. 2.. 2...`	Punktirane notne dolžine.
`d4 g,8 a b c d4 g, g`	Primer prvih dveh taktov zgornjega glasu Menueta za klavir v G-duru J. S. Bacha.

6.2.8 Iskanje po zadetkih

Aplikacija omogoča iskanje po analiziranih motivih s pritiskom na Ctrl+F ali preko menija. Na spodnjem robu centralnega gradnika se prikaže iskalna vrstica, v katero vpišemo zaporedje not ali pavz v priročni sintaksi LilyPond.

Oblika vnosa vedno vsebuje višino note in nato njeno dolžino. Če dolžine ni, se predpostavi prejšnja. Tabela 6.1 prikazuje podprto sintakso za vnos not. Melodijo božične pesmi „Sveta noč“ F. X. Gruberja bi na primer lahko zapisali kot `g4. a8 g4 e2. g4. a8 g4 e2. d'2 d4 b2. c2 c4 g2..`

Po vnosu iskanega niza se ta s pomočjo razreda `CALilyPondImport` pretvori v glasbeni dokument z notami. Vzorec not se nato išče od trenutno izbranega notnega para ali teme naprej ali nazaj.

Aplikacija podpira iskanje po ritmu, melodiji ali obeh kriterijih. Primerjalna funkcija je podobna tisti pri gradnji priponskih dreves, opisani v poglavju 5.2.1. Za iskanje vzorcev po ritmu poznamo relativni ali absolutni ritem, za iskanje vzorcev po melodiji pa relativno iskanje po kvantiteti in kvaliteti intervala ali po absolutnih višinah.

Za lažjo predstavo sintakse LilyPond se ob dvojnem kliku na izbrani notni par ali temo v iskalnem polju pojavi vsebina not, izvožena v ta jezik. Na ta način uporabniku ni potrebno vpisati celotnega iskanega niza, ampak ta samo prilagodi obstoječega, ki mu je podoben.

6.3 Harmonia kot razširitev aplikacije za pisanje not

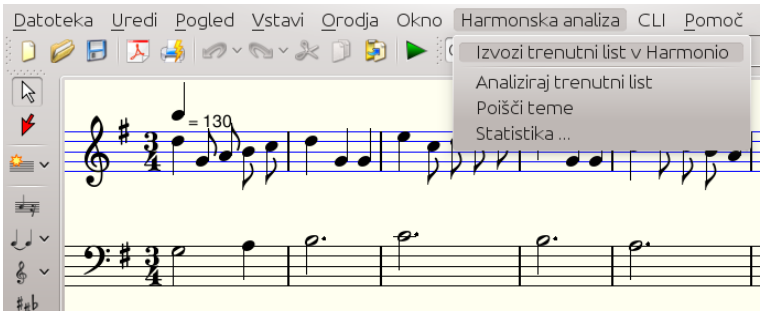
Da bi približali delo z aplikacijo za analizo tem končnim uporabnikom, ki so navajeni že obstoječih rešitev za pisanje glasbe, smo razširili aplikacijo Canorus, da omogoča analizo trenutnega dokumenta s pomočjo Harmonie. Razširitev podpira naslednje funkcionalnosti:

- zagon aplikacije Harmonia iz Canorusa,
- izvoz trenutno odprtega dokumenta v Harmonio,
- neposreden zagon analize trenutno odprtega dokumenta iz Canorusa,
- interaktivni prikaz vseh pojavitev izbranega notnega para ali teme v Canorusu,
- izvoz funkcijskih oznak pri harmonski analizi iz Harmonie v Canorus.

Canorusove razširitve imajo obliko mape, shranjene v Canorusovi mapi z razširitvami *plugins/*. Poleg vsebine razširitve, mapa vsebuje opisno datoteko canorusplugin.xml z informacijami o razširitvi (ime in opis, različico, URL, prevode) in s točkami integracij v aplikaciji (preko menijev, orodjarn ali Canorusovih notranjih signalov npr. ob izbiri glasbenih elementov, predvajanju itd.).

Za prve tri funkcionalnosti smo v mapo z razširitvijo dodali funkcije, napisane v Pythonu za prikaz glavnega okna Harmonie brez inicializacije Qt, saj je bila knjižnica že inicializirana ob zagonu Canorusa in funkcije za uvoz že zgrajenih dokumentov in njihovo analizo. V opisno datoteko canorusplugin.xml smo dodali meni s postavkami za prikaz okna Harmonie in izvoz trenutnega dokumenta vanj, neposreden zagon analize tem, iskanje glavne teme in prikaz statistike. Ta meni je prikazan na sliki 6.8.

Za interaktivni prikaz vseh pojavitev izbranega notnega para ali teme v Canorusu je bila potrebna sprememba njegovega vmesnika za razširitve. Težava je bila v tem, da so bili razširitvam na voljo le jedrni razredi (dokument, črtovje, glas, nota, interval ipd.) in razredi za uvoz in izvoz dokumentov, medtem ko interakcije z grafičnim vmesnikom iz razširitev ni bilo. V ta namen smo v vmesnik dodali funkciji `setSelection()` in `repaintUi()`. Prva v trenutno aktivnem oknu označi podane glasbene elemente, medtem ko druga osveži notni prikaz v trenutnem oknu. Ob kliku na notni par ali temo v Harmonii, ta preveri, če deluje kot razširitev in pokliče ti dve funkciji.



Slika 6.8: Meni, ki omogoči dostop do Harmonie neposredno iz programa za pisanje not Canorus.

Izvoz funkcijskih oznak pri harmonski analizi doda najdene funkcijske oznake v novo črtovje analiziranega dokumenta. V sklopu tega diplomskega dela harmonske analize ne obravnavamo.

Poglavje 7

Primer rabe orodja

V tem poglavju bomo predstavili uporabo aplikacije za analizo tem v fugah J. S. Bacha. Najprej bomo podrobneje spoznali zgradbo fuge. Nato bomo konkretno fugo uvozili v našo aplikacijo in zagnali postopek analize tem. Preverili bomo, če je bila glavna tema najdena, pregledali zgradbo motivov in tem ter omenili ostale informacije o skladbi, ki nam jih nudi aplikacija. V nadaljevanju bomo zagnali samodejni postopek iskanja glavne teme nad vsemi 48 fugami v zvezkih WTK, interpretirali dobljene rezultate in pogledali možne rešitve za izboljšanje iskanja.

7.1 Zgradba fuge

Fuga je večglasna glasbena oblika, za katero je značilna sistematična imitacija ene teme [27]. S pomočjo aplikacije smo analizirali teme v obeh zvezkih WTK J. S. Bacha, saj je ravno v teh delih fuga v zgodovini doživela svoj vrhunec. Vsak zvezek iz zbirke WTK vsebuje 24 preludijev in fug za inštrumente s tipkami (običajno klavir, čembalo ali orgle).

Fugo vedno uvede en glas, ki predstavi temo. Prvi nastop teme je na osnovni stopnji in se imenuje *dux* (lat. vodja). Nato se temi pridruži drugi glas, ki v razdalji kvinte navzgor ali kvarte navzdol imitira temo. Tak nastop je na peti stopnji in se imenuje *comes* (lat. spremljevalec). Tema se, odvisno od števila glasov — v WTK so fuge 3- ali 4-glasne, — ponovi še v tretjem (*dux*) in četrtem (*comes*) glasu. Na sliki 7.1 je prvih 9 taktov štiriglasne Fuge št. 20, WTK 1, v a-molu. Obarvane note prikazujejo temo. Skladbo uvede *dux*, nato v četrtem taktu nastopi *comes*, v osmem spet *dux* in v enajstem *comes*. V nadaljevanju fuge se teme prosto pojavijo še na drugih stopnjah tonalitete. Prisotne so tudi medigre, kjer je poudarek predvsem na harmoniji in sekvencah,

ter mostički, ki so namenjeni prehodom med stopnjami. V zadnjem, tretjem delu, se tema ponovi pogosteje, prisotna pa je tudi *stretta* in pedalni ton [12]. Prvo pomeni pojavitev nove teme, še preden se prejšnja zaključi. Ponavadi se zaradi tega sama tema tudi skrajša. Na ta način skladatelj doseže stopnjevanje napetosti proti koncu skladbe. Pedalni ton omenimo le kot zanimivost in pomeni, da spodnji glas več taktov obleži na enem tonu (ali v oktavi), medtem ko preostali glasovi pletejo svoje melodije. S tem skladatelj umiri napetost in poslušalca pripravi na zaključek. Na sliki 7.2 vidimo konec iste fuge. *Stretta* je prisotna v taktih 81 in 85. V začetku skladbe teme vstopajo vsake 3 takte, tu pa na vsak takt. Ravno tako so teme krajše — namesto treh polnih taktov, se v taktih 82, 86 in 87 teme zaključijo predčasno in so dolge od enega do dveh taktov. V fugi je od druge polovice takta 83 dalje prisoten tudi pedalni ton.

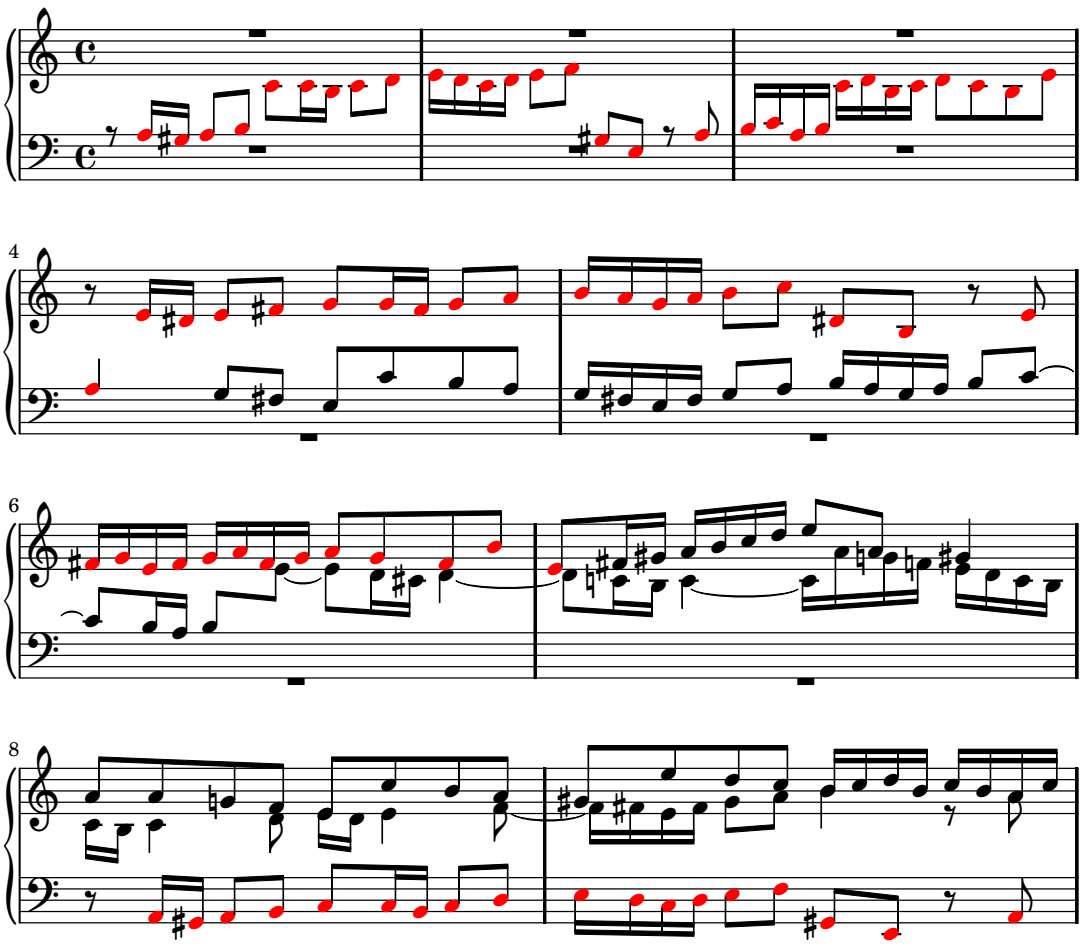
7.2 Raba orodja pri analizi fuge

Fugo, predstavljeno v prejšnjem podpoglavju, smo analizirali ročno. Takšna melodična analiza lahko traja tudi več ur. S pomočjo naše aplikacije za analizo tem pa lahko ta čas močno skrajšamo, saj aplikacija sama že predstavi možne teme, njihove skrajšane oblike in celo posamezne motive. Naloga muzikologa je, da skladbo uvozi v aplikacijo, zažene postopek analize tem in s pomočjo rezultatov sestavi interpretacijo glasbeno forme. Če ima zametke te že sestavljeno, pa z aplikacijo lahko preveri ali dopolni svoje ugotovitve.

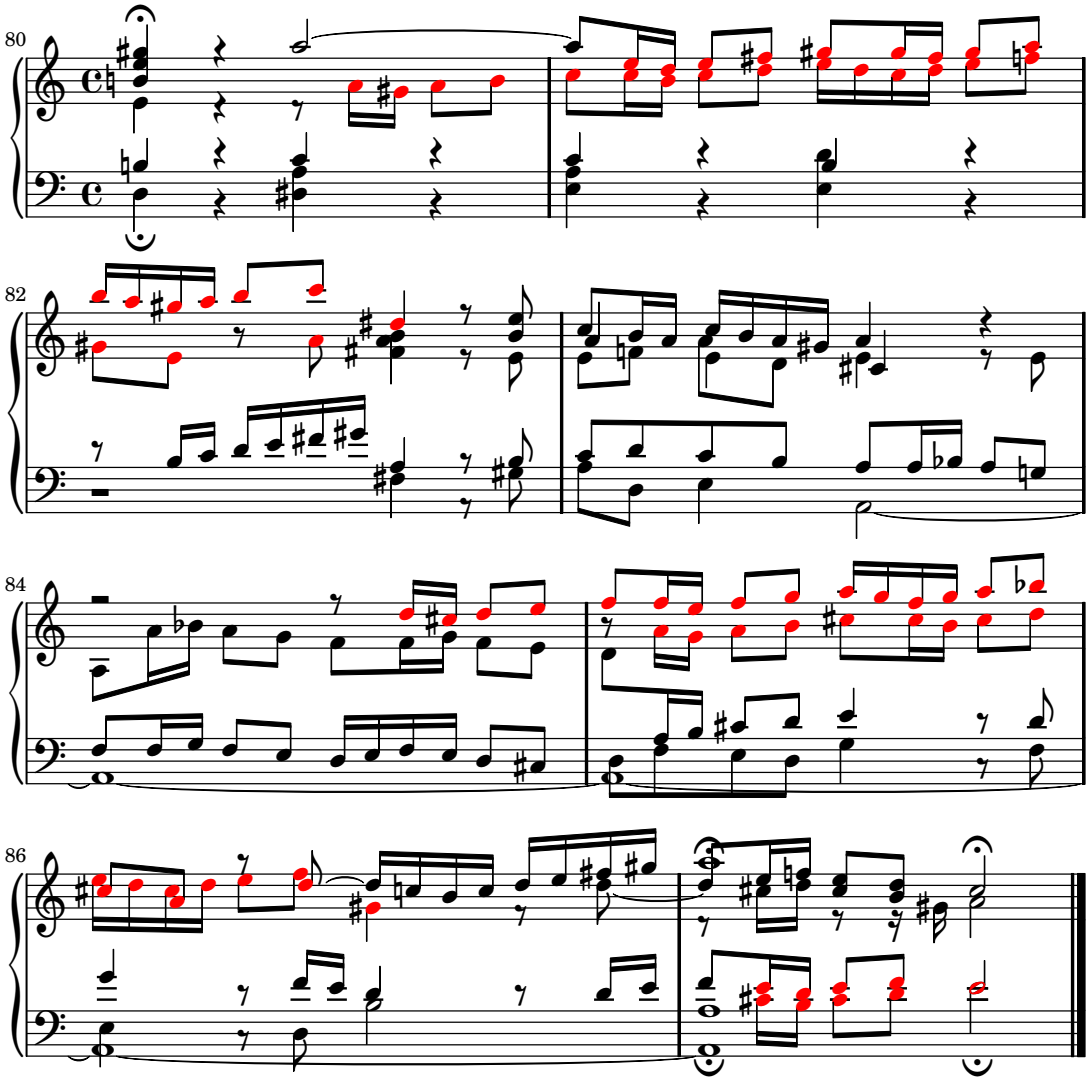
V aplikacijo smo uvozili datoteko MIDI Fuge št. 20, WTK 1, v a-molu [34] in zagnali postopek analize tem. Za postopek linearizacije smo izbrali vzporedno iskanje najbližjih sosedov, za primerjavo notnih parov pa ujemanje po ritmičnem razmerju in po kvantiteti intervala med notami. Najdaljšo dolžino teme smo nastavili na 30 not, najkrajšo na 4, najmanjše število ponovitev teme pa na 2. Ostale parametre smo pustili privzete.

V drugem koraku — pri pripravi skladbe na analizo — ni bilo posebnosti. Zaradi podane že večglasne datoteke MIDI je bila linearizacija relativno enostavna. Noben glas, razen nekaj akordov, ni sočasno izvajal več not. Postopek linearizacije je tako le še glede na postavitev pavz v njih razdelil glasove na skupno 37 segmentov.

V tretjem koraku se je zgradilo priponsko drevo notnih parov, ki je vsebovalo 2480 vozlišč. Statistično je bilo v fugi več kot polovica postopov sekundnih, v enakem ritmičnem razmerju (606 velikih sekund gor in 655 malih sekund dol). Največja dosežena globina je bila zgornja omejitev dolžine teme, to je 29 parov, oziroma 30 not.

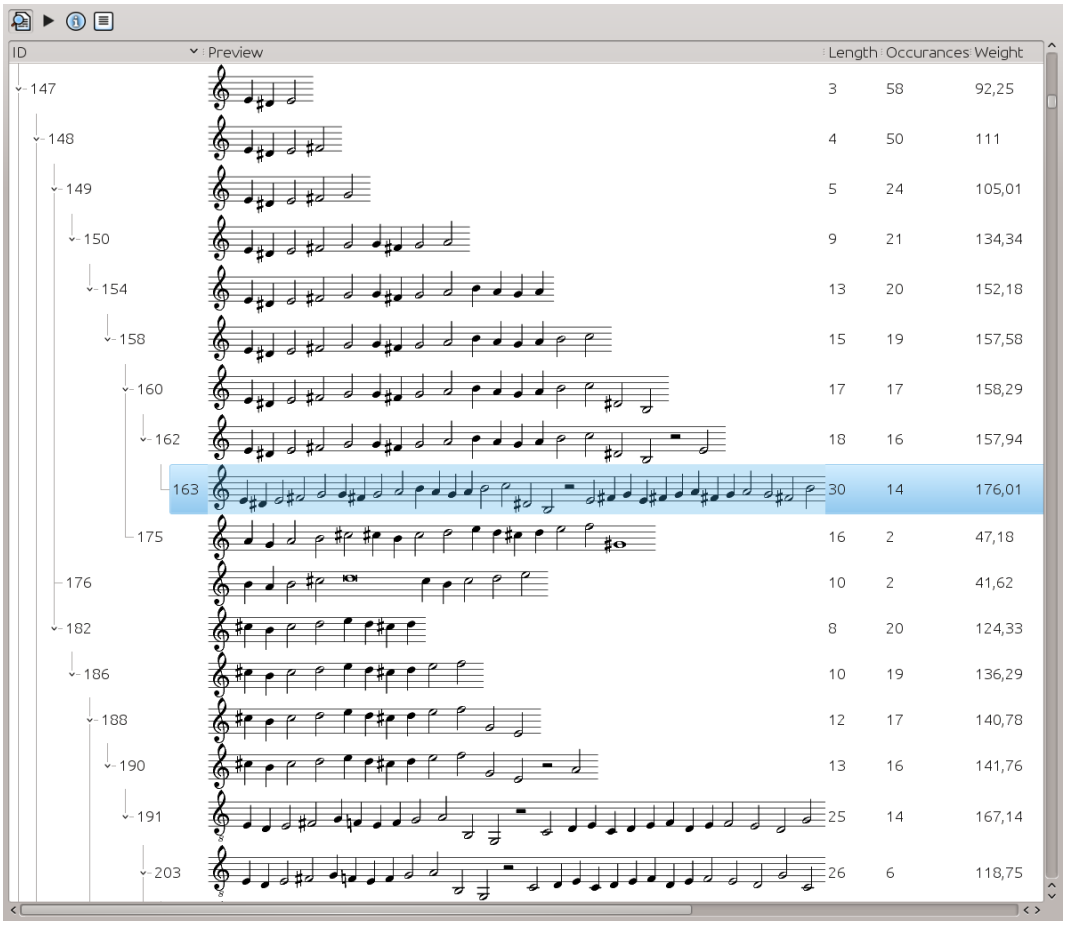


Slika 7.1: Začetek Fuge št. 20, WTK 1, v a-molu, J. S. Bacha



Slika 7.2: Konec Fuge št. 20, WTK 1, v a-molu, J. S. Bacha

V končnem, četrtem koraku, se je iz priponskega drevesa notnih parov zgradilo priponsko drevo tem, sestavljeno iz 1080 vozlišč. Na sliki 7.3 je prikazano poddrevo, v katerem je vsebovana glavna tema (ID teme 163). Ta je dolga 30 not in se je v identični obliki *duxa* in *comesa* v celotni skladbi ponovila 14-krat. Tema je najbolje ocenjena v skladbi. Za primer lahko omenimo prednika teme, dolgega 15 not (ID 158). Ta se ponovi 19-krat in ima dve nadaljevanji — s skokom dol za septimo na polovinko (kjer se nadaljuje tudi glavna tema) ali na celinko. Prvo nadaljevanje se ponovi 17-krat, drugo pa 2-krat. Zaradi optimizacije pa se lahko zgodi, da vsota števila pojavitev naslednikov ni vedno enaka številu pojavitev korenskega vozlišča. Tak primer je tema z ID 160 (17 ponovitev) in njen edini naslednik z ID 162 (16 ponovitev). V primeru, da bi zmanjšali konstanto MIN_THEME_OCCURANCES na 1, bi se poleg obstoječega naslednika pojavil še eden z eno samo pojavitvijo v skladbi. Zanimiv je tudi najbolj značilen motiv v skladbi, to je začetni postop glavne teme s kratkim menjalnim tonom navzdol in s tremi dolgimi postopi navzgor (ID 149), ki se ponovi kar 24-krat.



Slika 7.3: Priponsko poddrevo, v katerem je vsebovana glavna tema Fuge št. 20, WTK 1, v a-molu, J. S. Bacha.

7.3 Iskanje glavnih tem v fugah

Postopek za analizo tem, predstavljen na konkretni fugi v prejšnjem podpoglavju, nam poda veliko informacij o zgradbi skladbe, med njimi tudi oceno pomembnosti vsake teme. Če predpostavimo, da je med najbolje ocenjenimi temami tudi glavna tema, lahko uporabimo naš postopek za iskanje glavne teme skladbi. Če postopek avtomatiziramo, lahko iskanje glavne teme poteka samodejno nad več podanimi skladbami. Za primer lahko navedemo nekaj aplikacij, ki bi to uporabljale:

- samodejni izris glavnih tem v podanih skladbah,
- gradnja in iskanje po obsežni glasbeni literaturi z vnosom melodije,
- učenje računalniškega skladanja z analizo zbirke tem iz skladb v določenem slogu,
- iskanje podobnosti oziroma plagiatov med skladbami,
- statistične informacije o temah več skladb,
- kalibracija parametrov našega postopka na učnih primerih (npr. fugah).

Da bi preizkusili naš postopek v vlogi iskanja glavne teme v skladbi, smo izvedli test, ki samodejno analizira vseh 48 fug v obeh zvezkih WTK J. S. Bacha, zapisanih v datotekah MIDI [34, 48]. Test za vsako skladbo izvede vse 4 korake analize, nato pa preveri, na katerem mestu se nahaja najbolje ocenjena tema, ki je na začetku skladbe. Fuga je namreč zgrajena tako, da se glavna tema (*dux*) pojavi vedno na začetku. Če je ta tema na prvem mestu po oceni, potem je test uspešno opravljen.

Za postopek analize smo uporabili iste parametre pri vseh fugah. Izkustveno so se najbolje izkazali naslednji parametri:

- postopek linearizacije:

  uporabljen algoritem: vzporedno iskanje skupno najbližjih sosedov,

  največja razdalja med segmenti MAX_REST: 900,

  najkrajša dolžina segmenta MIN_LENGTH: 3,

- postopek gradnje priponskega drevesa notnih parov:

  najkrajša dolžina teme MIN_THEME_LENGTH: 4,

  najdaljša dolžina teme MAX_THEME_LENGTH: 30,

najmanjše število ponovitev teme MIN_THEME_OCCURANCES: 2,

ujemanje parov: kvantiteta intervala in ritmično razmerje,

- postopek gradnje priponskega drevesa tem in ocenjevanje:

  utež entropije po ritmu C_{rhy}: 50,

  utež entropije po melodiji C_{mel}: 50,

  utež števila pojavitev teme C_{occ}: 60,

  logaritemska osnova pojavitev teme B_{occ}: 3,

  utež dolžine teme C_{len}: 1,

  logaritemska osnova dolžine teme B_{len}: 25.

Tabela 7.1 prikazuje rezultate testa.

Tabela 7.1: Rezultati testa samodejnega iskanja tem.

fuga	**test opravljen**	**rang glavne teme**	**število vseh pripon**
WTK 1 Fuga št. 01	DA	1	290
WTK 1 Fuga št. 02	NE	7	383
WTK 1 Fuga št. 03	NE	86	804
WTK 1 Fuga št. 04	NE	149	524
WTK 1 Fuga št. 05	DA	1	333
WTK 1 Fuga št. 06	DA	1	427
WTK 1 Fuga št. 07	NE	72	519
WTK 1 Fuga št. 08	NE	91	649
WTK 1 Fuga št. 09	NE	35	424
WTK 1 Fuga št. 10	NE	31	630
WTK 1 Fuga št. 11	NE	5	329
WTK 1 Fuga št. 12	NE	64	681
WTK 1 Fuga št. 13	NE	5	541
WTK 1 Fuga št. 14	NE	3	398
WTK 1 Fuga št. 15	NE	120	955
WTK 1 Fuga št. 16	NE	3	326
WTK 1 Fuga št. 17	NE	83	404
WTK 1 Fuga št. 18	NE	15	390
WTK 1 Fuga št. 19	NE	34	587
WTK 1 Fuga št. 20	DA	1	1079

WTK 1 Fuga št. 21	NE	108	462
WTK 1 Fuga št. 22	NE	40	298
WTK 1 Fuga št. 23	NE	6	448
WTK 1 Fuga št. 24	NE	107	959
WTK 2 Fuga št. 01	NE	32	1162
WTK 2 Fuga št. 02	NE	16	166
WTK 2 Fuga št. 03	NE	11	302
WTK 2 Fuga št. 04	NE	62	677
WTK 2 Fuga št. 05	DA	1	379
WTK 2 Fuga št. 06	NE	24	468
WTK 2 Fuga št. 07	NE	92	395
WTK 2 Fuga št. 08	NE	31	247
WTK 2 Fuga št. 09	NE	9	109
WTK 2 Fuga št. 10	NE	15	413
WTK 2 Fuga št. 11	NE	4	283
WTK 2 Fuga št. 12	NE	34	406
WTK 2 Fuga št. 13	NE	329	388
WTK 2 Fuga št. 14	NE	111	514
WTK 2 Fuga št. 15	DA	1	308
WTK 2 Fuga št. 16	NE	79	496
WTK 2 Fuga št. 17	NE	18	656
WTK 2 Fuga št. 18	DA	1	674
WTK 2 Fuga št. 19	NE	2	398
WTK 2 Fuga št. 20	NE	67	513
WTK 2 Fuga št. 21	DA	1	446
WTK 2 Fuga št. 22	NE	2	769
WTK 2 Fuga št. 23	NE	31	586
WTK 2 Fuga št. 24	NE	112	476

Test je v vseh fugah zaznal glavno temo, vendar jo je najbolje ocenil le v 8 od 48 fug (16,67 % uspešnost). Povprečen rang glavne teme je bil $44,84$ od povprečno $501,48$ zgrajenih priponskih besed na skladbo. Iz teh rezultatov lahko razberemo, da ocena teme sicer je v korelaciji z glavno temo, vendar naš postopek analize tem še zdaleč ni primeren za samodejno iskanje glavne teme v podanih skladbah. Z analizo napak smo med drugim ugotovili, da se v fugah, ki so bile najslabše analizirane, pojavljajo naslednje kategorije napak:

1. variacije na temo so zaznane kot ločene teme,
2. neprimerni parametri za posamezne fuge,
3. slaba izvorna datoteka MIDI.

Primer prve kategorije napak je analiza fug WTK 1 št. 2, 7 in 8. Tovrstne napake so posledica dejstva, da sta *dux* in *comes* v večini fug različna zaradi tehnike imitacije. Te razlike so majhne (drugačna višina ene ali dveh not), tako da muzikolog z vidika sistematične analize obravnava nastop in odgovor teme kot isto temo. Ena izmed možnih rešitev za to kategorijo napak bi bilo računanje podobnosti tem in združevanje v primeru zadostne podobnosti. Za mero, ki ocenjuje podobnost tem, bi bila primerna Levenshteinova razdalja [13]. Ta pozna tri vrste operacij: vstavljanje, zamenjavo ali brisanje elementa, s katerimi lahko pretvorimo poljuben prvi niz na drugega. Če bi za nize uporabili višine in dolžine not, bi tako lahko izračunali razdalje med vsemi temami. Nato bi teme združili v gruče s pomočjo prilagojene metode iskanja najbližjih sosedov [49], kjer bi namesto konstante števila najbližjih sosedov k vzeli največjo dovoljeno razdaljo med njimi. Pri analizi fug bi bila ta razdalja 1 ali 2 za eno ali dve dovoljeni različni noti v temah. Dobljene gruče bi predstavljale združene teme in zaradi večjega števila ponovitev bi te dobile boljšo oceno.

Rešitev problema z variacijami na temo, omenjena v prejšnjem odstavku, bi delovala nad tistimi vrstami variacij, ki imajo količinsko čim manj različnih not. Še posebej v klasicističnih sonatah pa so variacije tudi take, da razdelijo dolge note teme na več krajših [72]. Poslušalec temo vseeno zazna, ker notne višine na poudarjenih dobah v variaciji ostanejo enake tistim v prvotni temi. Da bi upoštevali to dejstvo, bi morali poleg notnih višin in dolžin iz datoteke MIDI zajeti tudi metrum (glej poglavje 4). Nato bi pri gradnji priponskega drevesa notnih parov upoštevali le notne pare s poudarjenimi levimi notami.

Druga kategorija napak se najbolj kaže v analizah fug WTK 1 št. 4, 9, 10 in WTK 2 št. 3. Izstopata dva parametra ocenitvene funkcije tem (glej enačbo (5.8)): logaritemska osnova za dolžino teme B_{len} in utež za entropijo po ritmu C_{rhy}. V fugi št. 4 je tema dolga le 5 not (ta je znana po podpisu avtorja v višinah not b-a-c-h). Če zmanjšamo konstanto B_{len} na 5, je glavna tema ocenjena najbolje. V omenjenih fugah WTK 1 št. 9 in WTK 2 št. 3 je najbolje ocenjena tema spremljevalni motiv. Tega sestavljajo enako dolge note, medtem ko je glavna tema ritmično bolj razgibana. Če povečamo konstanto C_{rhy} na 200, glavna tema postane najbolje ocenjena. Nasprotno pa je zgrajena fuga št. 10, ki ima glavno temo sestavljeno iz ritmično identičnih not, kar 61 % postopov v vsej skladbi pa je ritmično enako dolgih v intervalu sekunde. Najbolje

ocenjena tema v fugi je eden izmed takih postopov, medtem ko glavna tema zaradi entropije po ritmu nima dobre ocene. Ta postane najbolje ocenjena, če zmanjšamo konstanto C_{rhy} na 0. Vidimo, da s spreminjanjem parametrov lahko močno popravimo ocene glavnih tem. Žal pa enega nabora parametrov, ki bi bil optimalen za vse fuge, nismo našli, kot tudi ne enostavnega načina za samodejno prilagajanje parametrov posamezni skladbi.

V tretji kategoriji napak najbolj izstopajo napačno razporejeni toni po kanalih MIDI (WTK 2 št. 2, 14 in 24) in drugače zapisan trilček ob ponovitvah (WTK 2 št. 13). Če bi kanale sploščili, bi lahko rešili prvo vrsto težave. To pa ne bi bilo primerno, če bi analizirali skladbo, v kateri nastopa več inštrumentov. Tam bi pri postopku linearizacije s tem zgradili segmente, ki vsebujejo tone več inštrumentov, kar ni pravilno. Optimalne rešitve tega problema ni. Potrebno bi bilo dobiti boljše vhodne datoteke MIDI, kar pa ni bilo mogoče. Napačen zapis trilčkov ali triol nastane zaradi zaokrožitvene napake aplikacije, ki je generirala datoteke MIDI. Rešitev, ki rešuje problem variacij na temo, bi bila primerna tudi za reševanje tovrstnih težav.

Poglavje 8

Zaključek

Naloga muzikološke sistematične analize glasbe je analiziranje razumljene glasbene teorije in znotraj te, kompozicijskih tehnik. Analiza teh je dolgotrajna in naporna, saj zahteva veliko koncentracije in predznanja. V ta namen smo raziskali možnosti umestitve te analize v teoretični model, ki bi ga bilo mogoče računalniško podpreti. Osredotočili smo se na analizo melodičnih prvin glasbe. Zasnovali smo algoritme za pripravo glasbenega zapisa za analizo in za izvedbo same analize v več zaporednih korakih. Teoretični model smo nato uporabili, da smo razvili celostno uporabniško aplikacijo za uvoz glasbe iz najpogostejših zapisov, analizo te glasbe in prikaz rezultatov posameznih korakov analize, imenovano *Harmonia*. S tem smo muzikologom omogočili, da svoje interpretacije, ki se tičejo melodičnih prvin skladbe, preverijo ali dopolnijo s pomočjo računalniške analize.

Harmonio smo preizkusili na fugah J. S. Bacha, saj je ta polifona oblika ravno v njegovih zvezkih WTK dosegla svoj vrhunec. Z muzikološko pomočjo smo ugotovili, da aplikacija uporabniku predstavi veliko pomembnih informacij, kot so informacije o številu fraz, statistične informacije o intervalih v skladbi, število in zgradba vseh motivov in tem, njihova mesta pojavitev in informativna ocena zanimivosti. Algoritme za analizo v *Harmonii* smo uporabili tudi za samodejno iskanje glavne teme v fugah in ugotovili, da so te preveč raznolike, da bi bilo z enakim pristopom in parametri vedno mogoče najti pravilno.

Na trgu uveljavljenih aplikacij, ki bi bile specializirane za muzikološko sistematično analizo posamezne skladbe, ni. Izpostavimo lahko manjši projekt *Analysis* [32], ki je bil razvit v sklopu raziskave s področja statistične analize harmonije v klasični glasbi. Program je namenjen le demonstraciji posameznih idej in nikoli ni bil mišljen kot končni produkt.

V diplomskem delu smo sodelovali z več muzikologi, ki so bili mnenja, da tržna niša na področju uporabniških aplikacij za tovrstno analizo glasbe nedvomno obstaja. *Harmonia* vsebuje več inovativnih pristopov za melodično analizo glasbe in ustrezen uporabniški vmesnik za krmiljenje le-teh. Nadaljnji razvoj aplikacije bi lahko potekal v dveh smereh: v smeri razvoja aplikacije v končni produkt za celostno sistematično analizo kompozicijskih tehnik v skladbi ali razvoj posameznih specifičnih rešitev, ki za osnovo uporabljajo *Harmoniine* algoritme.

Če bi želeli aplikacijo razviti v končni produkt, bi bile potrebne naslednje večje dopolnitve:

- integracija interaktivnega gradnika za izris celotne skladbe in označevanje posameznih struktur,
- harmonska in oblikoslovna analiza skladbe,
- dopolnitev obstoječega pristopa za analizo tem s podporo za variacije,
- poenostavitev ali samodejna zaznava večine parametrov.

Uporaba posameznih algoritmov, ki sestavljajo *Harmonio*, bi omogočila enostaven razvoj aplikacij, kot so iskalniki po temah skladb v obsežnih arhivih glasbe ali iskanje plagiatov med skladbami. Posebej zanimivo področje pa bi bila uporaba samodejno pridobljenega znanja iz analize glasbe v namene izboljšanja procesa kompozicije s pomočjo računalnika. Rešitev bi se lahko integrirala v obstoječe aplikacije za pisanje not, sproti analizirala semantiko že napisanega in predlagala nove rešitve. Če pa bi podprli že obstoječe sloge kompozicije, bi lahko uporabniki brez glasbenega predznanja s „pametno aplikacijo za skladanje“ v nekaj korakih na enostaven način zložili novo skladbo. Take aplikacije uvedejo popolnoma novo tržno nišo.

Slike

Tabele

Algoritmi

Literatura

[1] T. Anders „A Model of Musical Motifs“, *University of Plymouth*, 2007.

[2] A. L. Barabási, E. Bonabeau „Scale-Free Networks“, *Scientific American*, št. 288, str. 60-69, 2003.

[3] M. Barbo, *Obča muzikologija*, Ljubljana: Filozofska fakulteta Univerze v Ljubljani, Oddelek za muzikologijo, 2010

[4] L. E. Baum, T. Petrie, G. Soules, and N. Weiss „A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains“, *Ann. Math. Statist.*, zv. 41, št. 1, str. 164-171, 1970.

[5] J. J. Carabias-Orti, P. Vera-Candeas, F. J. Cañadas-Quesada, N. Ruiz-Reyes „Music scene-adaptive harmonic dictionary for unsupervised note-event detection,“ *IEEE Transactions on Audio, Speech, and Language Processing*, zv. 18, št. 3, 2010.

[6] E. Ferkova, M. Ždimal, P. Šidlik „Chord Evaluation in MIDI-Based Harmonic Analysis: Mozart, Schubert, Brahms“, *Tonal Theory for the Digital Age (Computing in Musicology)*, št. 15, str. 172-186, 2007.

[7] E. Fredkin „Trie memory“, *Comm. ACM, New York, NY, USA*, zv. 3, št. 9, str. 490-499, 1960.

[8] A. Forte, *The Structure of Atonal Music*, New Haven and London: Yale University Press, 1973

[9] J. W. Hunt, T. G. Szymanski „A fast algorithm for computing longest common subsequences,“ *Commun. ACM, New York, NY, USA*, zv. 20, št. 5, str. 350-353, 1977.

[10] E. Joanis, S. Larkin „Tightly Packaged Tries: How to Fit Large Models into Memory, and Make them Load Fast, Too“, *Proceedings of the NAACL*

HLT Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing, Boulder, Colorado, str. 31-39, 2009.

[11] A. Kornstädt „The JRing System for Computer-Assisted Musicological Analysis“, *Arbeitsbereich Softwaretechnik, Fachbereich Informatik, Universität Hamburg*, 2001.

[12] D. Ledbetter *Bach's Well-tempered Clavier, The 48 Preludes and Fugues*, New Haven and London: Yale University Press, 2002, str. 218-220

[13] V. Levenshtein „Binary codes capable of correcting deletions, insertions, and reversals,“ *Soviet Physics Doklady*, zv. 10, No. 8, pp. 707-710, 1966.

[14] U. Manber, G. Myers „Suffix arrays: a new method for on-line string searches,“ *SIAM Journal on Computing*, zv. 22, št. 5, str. 935-948, 1993.

[15] U. Michels *Glasbeni atlas*, Ljubljana: DZS, 2002

[16] D. R. Morrison „PATRICIA — Practical Algorithm To Retrieve Information Coded in Alphanumeric,“ *Journal of the ACM*, zv. 15, št. 4, 1968.

[17] J. Osredkar *Glasbeni stavek, Harmonija I*, Ljubljana: Zavod Republike Slovenije za šolstvo, 2009

[18] J. Osredkar *Glasbeni stavek, Harmonija II*, Ljubljana: Zavod Republike Slovenije za šolstvo, 2004

[19] J. Osredkar *Glasbeni stavek, Kontrapunkt*, Ljubljana: Zavod Republike Slovenije za šolstvo, 2006

[20] H. Riemann, *Analysis of J. S. Bach's Wohltemperirtes clavier*, London: Augener Ltd., 1893

[21] H. Riemann, *Handbuch der Harmonielehre*, Leipzig: Breitkopf & Härtel, 1929

[22] H. Schenker, *Fünf Urlinie-Tafeln*, New York: David Mannes Music School, 1933

[23] C. E. Shannon „A Mathematical Theory of Communication“, *Bell System Technical Journal*, zv. 27, str. 379-423, 623-656, 1948

[24] A. Takasu, T. Yanase, T. Kanazawa, J. Adachi „Music Structure Analysis and Its Application to Theme Phrase Extraction“, *S. Abiteboul, A.-M. Vercoustre (Eds.): ECDL '99, LNCS 1696*, str. 92-105, 1999.

[25] D. Temperley *Music and probability*, Cambridge, Massachusetts, London: The MIT Press, 2007.

[26] C. K. Tse, X. Liu, M. Small „Analyzing and Composing Music with Complex Networks: Finding Structures in Bach's, Chopin's and Mozart's“, *Polytechnic University, Hong Kong*, 2008.

[27] L. Vrhunc, *Glasbeni stavek: oblikoslovje*, Ljubljana: Znanstvena založba Filozofske fakultete, 2010, Uvod

[28] P. Weiner „Linear pattern matching algorithms,“ *14th Annual IEEE Symposium on Switching and Automata Theory*, str. 1-11, 1973.

[29] T. Weyde „Integrating segmentation and similarity in melodic analysis“, *University of Osnabrück*, 2004.

[30] C. Yang, C. K. Tse, X. Liu „Analyzing and composing music from network motifs“, *Polytechnic University, Hong Kong*, 2009.

Spletni viri

[31] (7. 4. 2011) Algorithmic composition. Dostopno na:
http://en.wikipedia.org/wiki/Algorithmic_composition

[32] (19. 5. 2011) Analysis — Milan Ždimal, Eva Ferkova. Dostopno na:
ftp://ftp.sac.sk/pub/sac/sk_made/dpan.zip

[33] (26. 5. 2011) Audiveris. Dostopno na:
http://audiveris.kenai.com

[34] (10. 6. 2011) Bach WTK MIDI files. Dostopno na:
http://www.topology.org/midi/wtk/#download

[35] (19. 6. 2011) Bucket sort Dostopno na:
http://en.wikipedia.org/wiki/Bucket_sort

[36] (7. 4. 2011) Computer-Aided Algorithmic Composition. Dostopno na:
http://en.wikipedia.org/wiki/Computer_music#Computer-Aided_Algorithmic_Composition

[37] (7. 4. 2011) Counterpoint. Dostopno na:
http://en.wikipedia.org/wiki/Counterpoint

[38] (18. 5. 2011) cpdl.org — Choral Public Domain Library. Dostopno na:
http://cpdl.org

[39] (23. 5. 2011) Digital audio. Dostopno na:
http://en.wikipedia.org/wiki/Digital_audio

[40] (15. 5. 2011) dLib.si — Digitalna knjižnica Slovenije. Dostopno na:
http://dlib.si

[41] (15. 5. 2011) Features — Canorus. Dostopno na:
http://canorus.berlios.de/wiki/index.php/Features

[42] (2. 6. 2011) Fibonacci word. Dostopno na:
http://en.wikipedia.org/wiki/Fibonacci_word

[43] (18. 5. 2011) FluidSynth. Dostopno na:
http://sourceforge.net/apps/trac/fluidsynth/

[44] (22. 5. 2011) General MIDI. Dostopno na:
http://en.wikipedia.org/wiki/General_MIDI

[45] (15. 5. 2011) GNU Lesser General Public License v3.0. Dostopno na:
http://www.gnu.org/copyleft/lesser.html

[46] (7. 4. 2011) Harmony. Dostopno na:
http://en.wikipedia.org/wiki/Harmony

[47] (18. 5. 2011) imslp.org — International Music Score Library. Dostopno na:
http://imslp.org

[48] (11. 6. 2011) J. S. Bach's Well-Tempered Clavier II (MIDI) by Yo Tomita. Dostopno na:
http://www.music.qub.ac.uk/tomita/midi.html

[49] (11. 6. 2011) k-nearest neighbor algorithm. Dostopno na:
http://en.wikipedia.org/wiki/KNN

[50] (22. 5. 2011) Midi. Dostopno na:
http://en.wikipedia.org/wiki/Midi

[51] (22. 5. 2011) MIDI File Format. Dostopno na:
http://www.sonicspot.com/guide/midifiles.html

[52] (17. 3. 2011) MIDI Manufacturers Association. Dostopno na:
http://www.midi.org

[53] (18. 5. 2011) Midi player for ALSA. Dostopno na:
http://www.parabola.me.uk/alsa/pmidi.html

[54] (23. 5. 2011) Music OCR. Dostopno na:
http://en.wikipedia.org/wiki/Music_OCR

[55] (18. 5. 2011) Musicology — Music theory, analysis and composition. Dostopno na:
http://en.wikipedia.org/wiki/Musicology#Music_theory.2C_analysis_and_composition

[56] (7. 4. 2011) Musicology. Dostopno na:
http://en.wikipedia.org/wiki/Musicology

[57] (23. 5. 2011) Neuratron AudioScore. Dostopno na:
http://www.neuratron.com/audioscore.htm

[58] (16. 5. 2011) Nokia acquires Trolltech. Dostopno na:
http://www.theregister.co.uk/2008/01/28/nokia_acquires_trolltech/

[59] (23. 5. 2011) Optical Character Recognition. Dostopno na:
http://en.wikipedia.org/wiki/Optical_character_recognition

[60] (7. 4. 2011) Orchestration. Dostopno na:
http://en.wikipedia.org/wiki/Orchestration

[61] (18. 5. 2011) PlayMidi. Dostopno na:
http://playmidi.sourceforge.net/

[62] (18. 5. 2011) Plugins for Sibelius — Analysis. Dostopno na:
http://www.sibelius.com/download/plugins/index.html?versionname=&category=3

[63] (18. 5. 2011) Thread-Support in Qt Modules. Dostopno na:
http://doc.qt.nokia.com/latest/threads-modules.html

[64] (16. 5. 2011) QTestLib Manual. Dostopno na:
http://doc.trolltech.com/latest/qtestlib-manual.html

[65] (16. 5. 2011) QtNetwork Module. Dostopno na:
http://doc.qt.nokia.com/latest/qtnetwork.html

[66] (15. 5. 2011) QtScript Module. Dostopno na:
http://doc.qt.nokia.com/latest/qtscript.html

[67] (23. 5. 2011) SharpEye. Dostopno na:
http://www.visiv.co.uk/quote.htm

[68] (16. 5. 2011) SQL Programming. Dostopno na:
http://doc.trolltech.com/latest/sql-programming.html

[69] (15. 5. 2011) Standard ECMA-262. Dostopno na:
http://www.ecma-international.org/publications/standards/Ecma-262.htm

[70] (15. 5. 2011) The GNU General Public License v3.0. Dostopno na:
http://www.gnu.org/licenses/gpl.html

[71] (24. 5. 2011) The MIDI File Format. Dostopno na:
http://midi.mathewvp.com/aboutMidi.htm

[72] (11. 6. 2011) W. A. Mozart, Sonata št. 11 v A-duru, KV 331. Dostopno na:
http://dme.mozarteum.at/DME/nma/nma_cont.php?vsep=197&gen=edition&l=1&p1=14

[73] (15. 5. 2011) WebKit in Qt. Dostopno na:
http://doc.qt.nokia.com/latest/qtwebkit.html

[74] (23. 5. 2011) WIDI Recognition System. Dostopno na:
http://www.widisoft.com/english/products.html