TARTU ÜLIKOOL
Arvutiteaduse instituut
Andmeteaduse õppekava

Ludvig Leis

Diginoodistuste sarnaste järgnevuste tuvastamine ja visualiseerimine

Magistritöö (15 EAP)

Juhendaja: Sven Aller, MSc

Tartu 2022

Diginoodistuse sarnaste järgnevuste tuvastamine ja visualiseerimine

Lühikokkuvõte:

Magistritöö eesmärk on analüüsida diginoodistusi ning visualiseerida sarnaseid järgnevusi. Uurimise põhjal valminud rakenduse abil on kasutajal võimalik leida diginoodistuste seast sarnaseid järgnevusi erinevatel meetoditel. Töös antakse ülevaade muusika andmeanalüütika erinevatest uurimissuundadest ning tutvustatakse andmetüüpidest tingitud erinevusi. Samuti kirjeldatakse noodistuste järgnevuste sarnasuse leidmise mõõtmisviisidest ning tutvustatakse varasemate tööde meetodeid. Töö lahenduse arendusprotsessist ja -valikutest antakse detailne ülevaade ning kirjeldatakse lahenduse testimise tulemusi. Lõpuks pakutakse välja lahenduse edasiarendamise võimalused.

Võtmesõnad:

Muusika andmeanalüütika, noodijärgnevus, diginoodistus, MusicXML

CERCS: P170 Arvutiteadus, arvanalüüs, süsteemid, kontroll

Identification and Visualization of Similar Sequences in Digital Sheet Music

Abstract:

The purpose of this Master's thesis is to analyze digital sheet music and visualize similar sequences. The paper describes a solution which allows the user to find similar sequences among digital sheet music using different methods. The paper gives an overview of the concept of music information retrieval and what are the different data types and their research methods. The thesis gives an overview of the ways of measuring sequence similarity of digital sheet music and describes the methods of previous work. A detailed overview of the development process of the solution is described and the results of testing the solution are discussed. Finally, further development opportunities of the solution are proposed.

Keywords:

Music information retrieval, note sequence, digital sheet music, MusicXML

CERCS: P170 Computer science, numerical analysis, systems, control

Sisukord

Sissejuhatus

Muusika andmeanalüütika (ingl *Music Information Retrieval, MIR*) on uurimissuund, mille eesmärk on arendada muusikalise informatsiooni töötlemise meetodeid. Muusikalistel sümbolitel põhinev infootsing on üks muusika andmeanalüütika alamharusid, kus on aluseks võetud muusika tunnuseid ja väljendusvahendeid kirjeldavad noodikirja sümbolid [1]. Need sümbolid ei defineeri heli, vaid on juhised, kuidas tekitada heli. Noodistustest leitavad andmed erinevad teistest muusika andmetüüpidest selle poolest, et need kirjeldavad detailselt muusika struktuuri elemente kui ka väljendusvahendeid (nagu näiteks meloodia, harmoonia ja rütm). Seetõttu on noodistuste põhjal võimalik tuvastada muusikat iseloomustavat informatsiooni, mis on aluseks sarnaste teoste tuvastamiseks.

Muusikaliste teoste sarnasuse määramise viise on erinevaid sõltuvalt andmetüübist, tuvastusmeetodist ja eesmärgist. Noodistuste võrdlemise aluseks on digitaalne noodistus, mille analüüsimisel kasutatakse peamiselt tekstitöötluse uurimismeetodeid ja rakendusi. Nende meetoditega on võimalik võrrelda noodistuste partiide vahelisi sarnaseid noodijärgnevusi. Selle töö raames käsitletakse noodijärgnevustena ka neid järgnevusi, mis sisaldavad kooskõlasid ja pause.

Magistritöö eesmärk on luua digitaalsete noodistuste andmete töötlemise lahendus, mis võimaldab kasutajal erinevate meetodite kaudu tuvastada noodistustevahelisi ühiseid sarnaseid järgnevusi. Kasutaja saab valida etteantud tunnuste järgi, milliste tingimuste alusel noodistuste partiidest pikimaid ühisjadasid leitakse. Leitud järgnevused on võimalik kuvada nii visuaalset kui ka auditiivselt, et anda kasutajatele võimalus hinnata leitud järgnevuste sarnasust ja meetodite mõju järgnevuste tuvastamisele.

Magistritöö koosneb kolmest osas. Esimeses osas tutvustatakse muusika andmeanalüütikas uuritavate andmete tüüpe, tuuakse näiteid erinevatest alamvaldkondadest ning kirjeldatakse muusika andmeanalüütika tuleviku väljakutseid. Teises osas antakse ülevaade noodijärgnevuste sarnasuse mõõtmise viisidest ning milliseid lahendusi on varem loodud. Kolmandas osas tutvustatakse noodistuste andmete töötlemise ja noodijärgnevuste leidmise lahendust. Seal kirjeldatakse, milliseid tehnoloogilisi vahendeid arendamisel kasutati, kuidas on võimalik lahendust kasutada ja milline on lahenduse algoritm. Lõpuks tutvustatakse lahenduses läbiviidud

teste ja nende tulemusi ning kirjeldatakse, millised võiksid olla lahenduse võimalikud edasiarendused.

1. Muusika andmeanalüütika

Muusika andmeanalüütika on valdkond, mille eesmärk on uurida muusikaga seotud andmeid ning arendada tarkvara, mis võimaldaksid muusika andmeid efektiivsemalt otsida, organiseerida, töödelda ja tõlgendada [2]. Muusika andmeanalüütika on kombinatsioon erinevatest teadusharudest, nagu näiteks muusikateadus, psühholoogia, psühhoakustika, informaatika, masinõpe ja signaalitöötlus [3]. Seos iga eelnimetatud teadusharuga kirjeldab, milliseid muusikaga seotud andmeid uuritakse. Näiteks signaalitöötlus on muusika andmeanalüütikas seotud helilainete uurimisega. Samuti saab jaotada muusika andmeanalüütika harudeks vastavalt uuritavatele andmetele. Neli põhilist andmeliiki on:

1) Muusika metaandmed - muusikat kirjeldavad tekstilised andmed (žanr, teose pealkiri, esitaja, avaldamise aasta, õigused, emotsioon jt.) [4].
2) Helilainetel põhinevad muusikaandmed - muusikaliste andmete akustiline kirjeldus (helisalvestis, esitus) [3].
3) Sümbolitel põhinevad muusikaandmed - digitaalsel kujul kirjeldatavad muusikalised andmed (diginoodistus, MIDI) [3].
4) Noodistus - heliteose üleskirjutis noodikirjas (pildi või dokumendi kujul noodistus) [3].

Muusika andmeanalüütika on kiiresti arenev valdkond. Infotehnoloogia ja informaatika areng on võimaldanud kasutada teiste valdkondade (nagu näiteks keele- ja signaalitöötlus) uurimismeetodeid muusika analüüsimiseks. 2008. aastal loodi ametlik selle valdkonna teadlaste mittetulundusühing ISMIR (ingl *International Society for Music Information Retrieval*), mille missiooniks on muusikaandmeid uurida, analüüsivõimalusi edasi arendada ja selle rakendamist õpetada. ISMIR koondab muusika andmeanalüütika märkimisväärsemaid uurimustöid ning korraldab iga-aastaseid maailma juhtivaid teaduskonverentse. Arvestades valdkonna erinevaid alamharusid ning võrdlemisi lühikest tegutsemisaega on võimalusi erinevate rakenduste väljatöötamiseks mitmeid. [2] [5]

1.1 Muusika andmeanalüütika andmetüüpide uurimisvaldkonnad

Muusika andmeanalüütika uurimisvaldkondi saab jagada mitmeti. Järgnevalt on kirjeldatud andmetüüpide põhjal erinevaid uurimise eesmärke ning viise, kuidas muusika andmeid töödelda ja analüüsida.

1.1.1 Muusika metaandmed

Metaandmed muusika kontekstis on teave helifailide kohta, mis kirjeldavad nendega seotud informatsiooni. Metaandmed koostatakse käsitsi ja nende alla kuuluvad näiteks helilooja nimi, esitaja, teose pealkiri, žanr, avaldamise aasta ja info muusika õiguste kohta. Metaandmed võimaldavad helifaile grupeerida, organiseerida, määrata autoriõigust ning neid digitaalselt üles otsida. [4]

Andmeanalüütikas on metaandmetel oluline roll muusika andmebaaside haldamisel, muusika kategoriseerimisel, sarnasuse leidmisel ning soovitussüsteemide rakendamisel. Helifailide metaandmetes sisalduvaid annotatsioone ja subjektiivseid märgendeid (nagu näiteks muusika meeleolu kirjeldused, sarnasuse ja populaarsuse skoorid, tekstilised tunnused) kasutatakse masinõppe ja tehisintellekti lahendustes, et soovitada kuulajatele uut muusikat. Populaarsed muusika voogedastuse platvormid (nagu näiteks Spotify, Apple Music, Deezer) kasutavad oma soovitusmudeli sisendina teoste metaandmeid, et pakkuda kuulajatele sarnaseid teoseid juba meeldinud või kuulatud muusika põhjal. Iga platvormi meetodid ja algoritmid metaandmete kasutamiseks on erinevad, et leida oma andmebaasidest sobivad lood kasutajale kuulamiseks [6]. Muusika metaandmete märgendamine on üks andmeanalüütika osa, millest on välja arenenud teenus. MusiMap[1] on üks neist, mis pakub ettevõtetele muusika märgendamise ja soovitussüsteemide rakendamise teenust. Selleks kasutatakse rakendust, mille süvaõppe võrgustik ja algoritmid leiavad andmebaasidest muusikat ning pakuvad välja sarnaseid teoseid vastavalt kuulaja profiilile. [7]

Metaandmetega seotud uurimisvaldkondade eesmärkideks on edasi arendada masinõppe mudeleid ja suurendada nende täpsust. Küll aga on takistuseks muusika metaandmete puudulik kvaliteet. Esiteks ei ole globaalset standardiseerimise süsteemi muusika helifailide identifikaatoritele ja metaandmetele, mis tekitab probleeme nii autoriõiguste tuvastamisel,

[1] https://musimap.net/

andmebaaside haldamisel (puudulikud andmeväljad või samade teoste mitmekordne sisestus ja töötlused) kui ka soovitussüsteemide täpsuses [8]. Helifailide hoomamatu kogus on põhjustanud olukorra, kus manuaalne metaandmete sisestamine ei ole mõistlik. Automaatne muusika märgendamine metaandmetega helifailist on järgmine masinõppe või tehisintellekti lahendus, mille täpsuse tõstmine nõuab mudelite edasiarendust [9].

Metaandmed kirjeldavad muusikaga seotud informatsiooni, aga mitte muusikat ennast. Seda on võimalik analüüsida kas muusika helisalvestisest või sümbolitel põhinevatest muusikaandmetest.

1.1.2 Helilainetel põhinevad muusikaandmed

Helilained on helisalvestistel põhineva muusika andmeanalüütika aluseks. Helilained tekivad mingis keskkonnas füüsikalise keha võnkumise tulemusena, mida inimkõrv kuulates tajub [10]. Muusikalised helilained kirjeldavad heliallika ühtlast perioodilist võnkumist teatava kindla kõrgusega toonil [11]. Muusika andmeanalüütikas kasutatakse signaalitöötlusvahendeid helilainetest informatsiooni saamiseks ja analüüsimiseks [12]. Helilaine tunnused, mida muusika andmeanalüütikas uuritakse, on:

1) Võnkeamplituud, mis kirjeldab helitugevust;
2) Sagedus, mis määrab ära helikõrguse (põhitooni ja ülemhelid);
3) Helispekter, tänu millele saab kirjeldada heli tämbrit;
4) Võnkumise kestus ehk helikestus.

Helilainete alusel on võimalik analüüsida mitmeid muusikalisi omadusi. Helilaine analüüsiga seotud peamised alamvaldkonnad on järgmised:

- **Sarnasuse tuvastamine**. Helilainete omaduste alusel on võimalik tuvastada ja klassifitseerida sarnaseid helisalvestisi. Masin- ja süvaõppe tehnoloogiad suudavad eraldada helisignaalist vajalikke tunnuseid ja neid tõlgendada muusika meetrumiks, žanriks, stiiliks või meeleoluks. Leitud omaduste alusel saab tuvastada ka teisi sarnaste omadustega helisalvestisi. Sarnasuse tuvastamist kasutatakse muusika soovitussüsteemides kui ka rakendustes (Shazam, Soundhound), kus kasutajad saavad enda genereeritud viisijupi järgi pärida sarnaseid teoste nimesid. Samuti on sarnasuse tuvastamise üks väljundeid plagiaadituvastus [13]. [1]

- **Transkriptsioon**. See hõlmab helisalvestisest muusikalise sisu eraldamist. Muusikalised omadused, mida helisalvestisest eraldatakse, võivad olla näiteks helistik, meloodia, harmoonia (akordijärgnevused), meetrum (löögi alguse ja pikkuse tuvastamine), korduvad struktuurid ja mängitav instrument. Transkriptsiooni teeb keeruliseks helisalvestiste mitmehäälsus (ja polürütmia[2]), mille puhul on keeruline eraldada üle kõigi partiide noodijärgnevusi ja harmooniat (ning rütmi), sest partiide samaaegne kõlamine väljendub nende kombinatsioonina ühtses helilaines. Ühehäälse heliteose puhul on transkriptsiooni tulemused paremad, kuid mitte täielikult täpsed, sest näiteks vibraato[3] esinemine meloodia partiis võib olla valesti transkribeeritud. [12]
- **Akustiline sõrmejäljestamine**. Helisalvestisest eraldatakse väike osa, mis teisendatakse n-ö "digitaalseks helisalvestise kokkuvõtteks" ehk teose sõrmejäljeks, mis on identifikaator kõlanud helisalvestisele. Sõrmejälg võib olla loodud näiteks akustiliste omaduste põhjal (nt helitugevus ja helispekter) või muusikaliste omaduste põhjal (nt rütm ja harmoonia). Sõrmejälg lisatakse andmebaasi ning toimub sobitamine teiste sõrmejälgedega. Sõrmejälgede sobitamisel omistatakse kõlanud helisalvestise sõrmejälg andmebaasis oleva elemendi metaandmetele. Nii on võimalik sõrmejälgede ja sobitusalgoritmidega identifitseerida kõlanud muusikat, isegi moonutatud versioonide puhul. [14] Selline heli sõrmejäljestamise meetod on oluline muusika andmebaaside haldamiseks, muusika kasutamise (sh autoriõiguste) järgimiseks ja sarnasuse tuvastamiseks [15].
- **Muusika genereerimine.** Selle uurimisvaldkonna eesmärk on luua täielikult arvuti poolt komponeeritud muusikat, mis kõlaks ka inimesele meelepäraselt ja arvestaks muusikateoreetilisi reegleid. Muusika loomisel kasutatakse näiteks reeglipõhiseid süsteeme, algoritme või süvaõppe meetodeid, et treeningandmestikest genereerida muusikapalu. [16]

Helilainetel põhinevate muusikaandmete põhjal on võimalik tuvastada ja analüüsida erinevaid muusika väljendusvahendeid. Küll aga on muusikaliste elementide (nagu näiteks noodid ja

[2] Mitme erineva rütmi üheaegne koosesinemine [11].
[3] Lauldes hääle või pilli mängides heli väristamine [11].

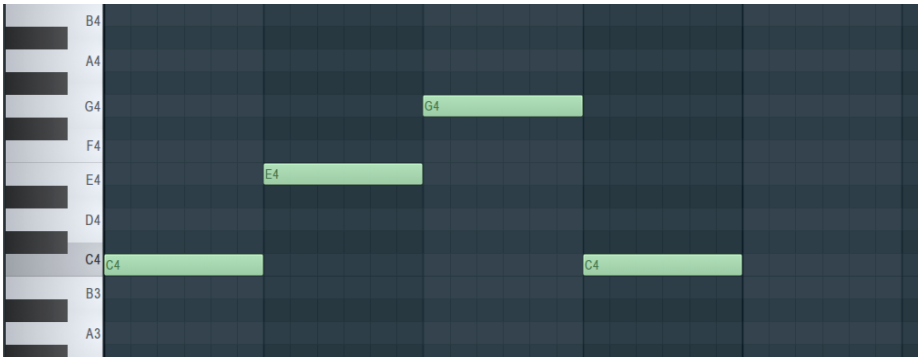
akordid) täpne tuvastamine keeruline. Andmetüübid, mis sisaldavad täpseid muusika elemente, on sümbolitel põhinevad.

1.1.3 Sümbolitel põhinevad muusikaandmed

Muusikalistel sümbolitel põhineva muusika andmeanalüütika aluseks on muusikat kirjeldavad diskreetsed sümbolid, mis kirjeldavad muusika sisu. Masinloetavad muusikalisi sümboleid sisaldavad failid on näiteks digitaalse noodistuse failid (MusicXML) ning MIDI-failid, millest on võimalik lugeda muusika väljendusvahendeid kirjeldavaid aspekte. Nendeks on näiteks noodid, intervallid, akordid, rütmijärgnevused ning nendega seotud elemendid ja altereeringud. Muusika väljendusvahendite sümbolite mõistmine nõuab muusikalisi taustateadmisi, mistõttu võib tavalisele andmeanalüütikule sümbolite lahti mõtestamine keeruline ja aeganõudev olla.

MIDI

MIDI (ingl *Musical Instrument Digital Interface*) fail on vahend elektrooniliste instrumentide ja arvutite omavaheliseks suhtluseks [17]. MIDI-faili noot sisaldab kolme elementi: noodi helikõrguse MIDI-koodi (näiteks ’60’ on C-noot 4. oktavis), kui tugevasti seda nooti vajutati ja ajatemplit [18]. Samuti võimaldab MIDI-fail salvestada infot näiteks teose tempo, helistiku, taktimõõdu ja metaandmete kohta [19].



Joonis 1. MIDI nootide esitus programmis FL Studio[4].

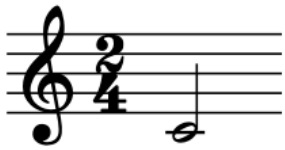
MusicXML

MusicXML (joonis 2) on XML failitüübil põhinev rangelt struktureeritud tekstidokument digitaalsete noodistuste esitamiseks [20]. See sisaldab informatsiooni noodistuse ülesehituslikest

[4] https://www.image-line.com/

elementidest (struktuur, faktuur, taktid) kuni nootide üksikute detailideni (altereerimismärgid, rõhud, rütmisümbolid) [21].

```
<note>
    <pitch>
     <step>C</step>
     <octave>4</octave>
     </pitch>
    <duration>2</duration>
    <voice>1</voice>
    <type>half</type>
    <stem>up</stem>
    </note>
```



Joonis 2. C-noot MusicXML failis.

Võrreldes helifailidega on digitaalse noodikirja puhul selgelt määratletud partiid, hääled ja nende sisu, mis võimaldab uurida muusikat, mis sisaldab mitmehäälsust ja polürütmiat. Põhilised uurimisvaldkonnad muusikalistel sümbolitel põhinevate andmetega on järgmised:

- **Sarnasuse tuvastamine.** Sümbolitel põhinevad muusikafailid võimaldavad uurida muusikalisi seoseid ja sarnasusi erinevate lugude vahel palju detailsemalt. Üheks valdkonnaks on sarnaste teoste tuvastamine kasutades nii MIDI-faile kui ka MusicXML-faile. Sarnasuse tuvastamiseks kasutatakse teisenduskauguse algoritme. Kahe noodistuse noote võrreldakse ja arvutatakse välja skoor, mis näitab, kui palju lisamisi, asendusi või kustutamisi peab ühes noodijärgnevuses tegema, et saada teise noodistuse noodijärgnevus. Mida madalam on skoor, seda sarnasemad on noodid. [22]
- **Sarnaste järgnevuste tuvastamine.** Ühe noodistuse meloodiajärgnevuse olemasolu teises noodis on uuritud kas sõnede baasil sobitusalgoritmidega või geomeetriliste sobitusalgoritmidega. Sõnede sobitamine töötab peamiselt ühehäälses muusikas, sest kooskõlasid ei ole võimalik lineaarse järgnevusena kirjeldada. Näiteks genereeritakse kindla pikkusega nootide jada (või “muster”, tihti n-grammi meetodil), mida hakatakse sobitama teiste noodijärgnevustega. Eesmärk on leida samasuguseid mustreid otsitavatest nootidest, mille õnnestumisel kuvatakse otsitud nootides mustri asukohad. Geomeetrilise

sobitusalgoritmide puhul tekitatakse muusikaandmetest geomeetriline esitus, mille kahedimensioonilised omadused võimaldavad kuvada ka kooskõlasid. Saadud mustrit võrreldakse seejärel teiste diginoodistuste partiide mustritega. [22]

Diginoodistuste kättesaadavus on võrreldes helisalvestistega halvem, sest muusika levib digitaalselt peamiselt helifailide kujul. Andmete puudumine on ajendanud inimesi looma enda versioone helisalvestiste noodistustest, mille versioonide varieeruvus on igale autorile omane. See ei ole ainuke viis, kuidas noodistusi digitaliseeritakse. Muusika andmeanalüütika uurimisvaldkond otsib lahendust, kuidas muuta skannitud dokumendi või pildi kujul noodistused masinloetavaks.

1.1.4 Pildi ja dokumendi kujul noodistused

Muusika andmeanalüütika üks alamharu on optiline muusika tuvastamine. Selle põhiline eesmärk on teisendada pildi või dokumendi kujul noodistused masinloetavaks formaadiks - näiteks MIDI- või MusicXML-failiks. Muusikaliste sümbolite teisendamine on sarnane optilisele märgituvastusele või käsitsi kirjutatud teksti tuvastamisele, kuid noodikirja puhul ei ole ainus ülesanne teisendada märgid tähestiku järgi ja panna õiges järjekorras ritta (et moodustuks sõna). Oluline on see, kuidas on sümbolid noodijoonestikule paigutatud ning milline on nende sümbolite muusikaline kontekst. Kuigi optilisi muusika tuvastamise rakendusi on loodud (nagu näiteks SmartScore[5], PhotoScore[6]), siis nende täpsus on madal - struktureeritud kodeeringu kättesaamine noodistuse pildilt on sama suur väljakutse nagu HTML-struktuuri teisendamisega veebilehe fotolt. [23]

1.2 Tuleviku väljakutsed muusika andmeanalüütikas

Teadlaste ja analüütikute huvi muusika andmeanalüütika valdkonna vastu on viimasel ajal kasvanud tänu treening- ja testandmete kättesaadavusele, tehnoloogiliste rakenduste ja meetodite edasiarendusele ning uutele avastustele [22]. Olenemata muusika andmeanalüütika kiirest arengust on lahendamata palju küsimusi, mis nõuavad pikaajalist uurimustööd. Muusika andmeanalüütiliste lahenduste võistlus MIREX (ingl *Music Information Retrieval Evaluation*

[5] https://www.musitek.com/
[6] https://www.neuratron.com/photoscore.htm

eXchange) on toimunud aastast 2005, kus muusika andmeanalüütikud ja teadlased saavad esitada oma lahendusi ja mudeleid välja pakutud võistluse kategooriatele. 2021. aasta MIREX-i teemad on välja toodud lisas 1 [24].

Muusika andmeanalüütika uurimisteemad muutuvad ajaga kitsamaks ning detailsete uurimisküsimuste puhul on takistuseks õigete andmete kättesaadavus. Esiteks puuduvad märgendatud andmed, mida saaks kasutada masinõppe või närvivõrkude mudelite loomisel. Teiseks on muusikafailide kasutamise puhul õiguslikud piirangud, mistõttu ei ole paljud andmestikud avalikud. Uute andmestike loomine on ajamahukas tegevus. Andmete kvaliteet ja standardiseerimata kuju peatükis 1.1 nimetatud andmetüüpide puhul nõuab ulatuslikku eeltöötlust ning kuna automaatsed ja 100% täpsed lahendused näiteks andmetüüpide teisendamiseks või metaandmete määramiseks puuduvad, on tulemused tarvis alati käsitsi üle kontrollida. Kuna uute andmete genereerimine on keeruline, siis kasutatakse juba olemasolevaid andmestikke, kuid samade andmestike kasutamine igal uurimusel ei arenda valdkonda edasi. [1] Erinevad andmestikud muusika andmeanalüütika tegemiseks on leitavad ISMIR-i veebilehelt [25].

Selle magistritöö raames on uurimise aluseks valitud digitaalne noodistus, kuna selle andmetüübi muusika väljendusvahendite andmete rohkus ja detailsus muudab analüüsi mitmekesisemaks ning struktuurne vorm lihtsustab andmete töötlust. Diginoodistuse analüüsimiseks kasutatakse erinevaid meetodeid ning algoritme, mille sisu kirjeldatakse järgmises peatükis.

2. Noodijärgnevuste sarnasuse mõõtmise viisid

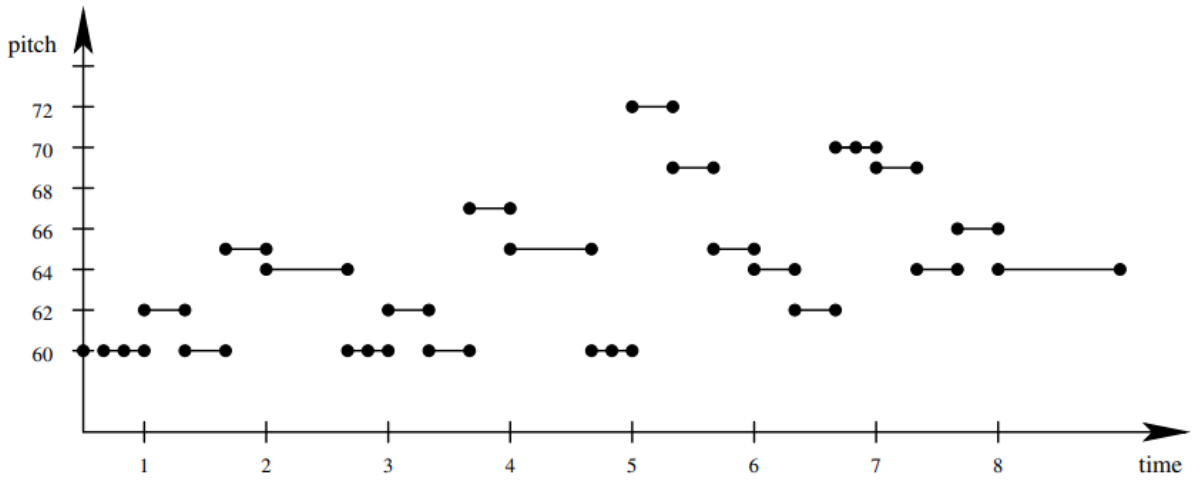
Muusika sarnaneb oma olemuselt tekstile. Nii nagu ka tekst, koosneb muusika kindlas järjekorras sümbolitest ja reeglitest, kuid need reeglid on tekstist vabamad ja keerukamad. Siiski on võimalik sümbolitel põhinevaid muusikaesitusviise analüüsida tekstitöötlusmeetodite ja algoritmidega.

2.1 Teisenduskauguse arvutamise meetodid

Meloodia sarnasuse määramiseks on kasutatud teisenduskauguse algoritme. Nende algoritmide puhul võrreldakse kahte sümbolite jada ning arvutatakse välja tulemus, mis näitab kahe jada omavahelist sarnasust. Kui teksti puhul on sümbolite jadaks tähed, siis noodistuste puhul saab sümbolite jadaks teisendada nootide informatsiooni näiteks noodikõrgused, intervallid või noodipikkused. Kõige levinum on Levensteini algoritm, kus arvutatakse teisenduskaugus selle põhjal, kui palju lisamisi, kustutamisi või asendusi peab tegema, et saada ühest sümbolite jadast teine. [26] M. Mongeu ja D. Sankoff panid aluse Levensteini algoritmi kasutamisele muusikanootidega, kus sümbolite jadaks teisendati nootide helikõrgus ja kestus [27]. Erinevate MIREX-i võistluste raames on analüüsitud ja kombineeritud Levensteini algoritmi kasutusviise [28].

Teisenduskauguse arvutamine Levensteini algoritmiga töötab vaid ühehäälse muusika puhul, sest teisendatud sümbolite jada on lineaarne. Teisenduskaugust kahe noodistuse vahel on arvutatud ka geomeetriliselt, mis võimaldab võrrelda ka mitmehäälset muusikat.

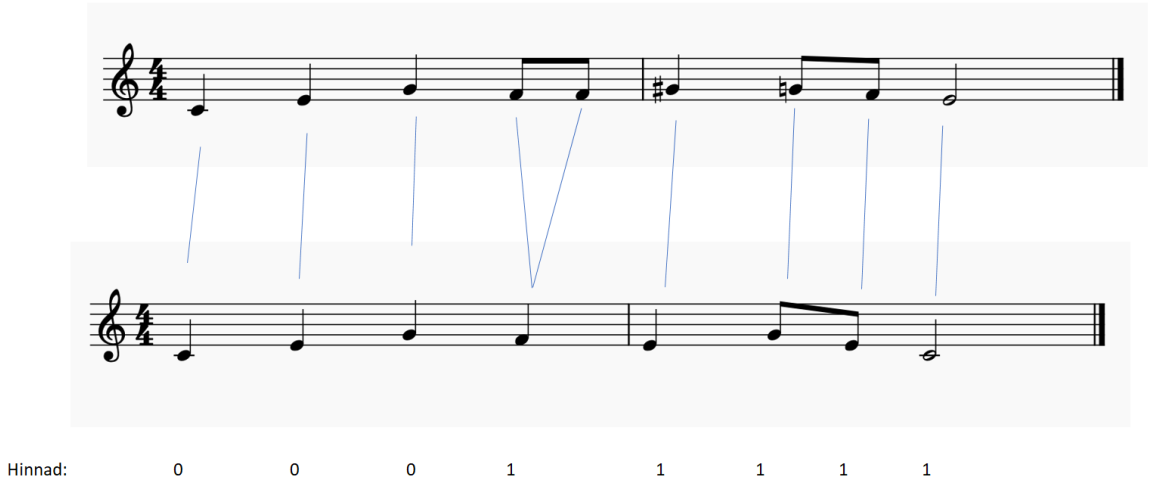
Geomeetrilise sarnasuse meetod põhineb meloodia kahedimensioonilisel esitusel, kus ühel teljel on helikõrgus ja teisel teljel on aeg (joonis 3). Geomeetriline esitusviis võimaldab võrrelda noodistuse harmoonilist sarnasust. Geomeetrilise sarnasuse puhul on võimalik rakendada tempo skaleerimist (piki horisontaaltelge) või helikõrguse transponeerimist (piki vertikaaltelge). Geomeetriline sarnasuse meetod on olnud MIREX-i võistlusel korduvaks teemaks, kus on välja pakutud erinevaid lahendusi - näiteks geomeetrilise sarnasuse arvutamine EMD kaudu (*Earth mover's distance)* või kasutades aegridade analüüsi algoritme. [28]



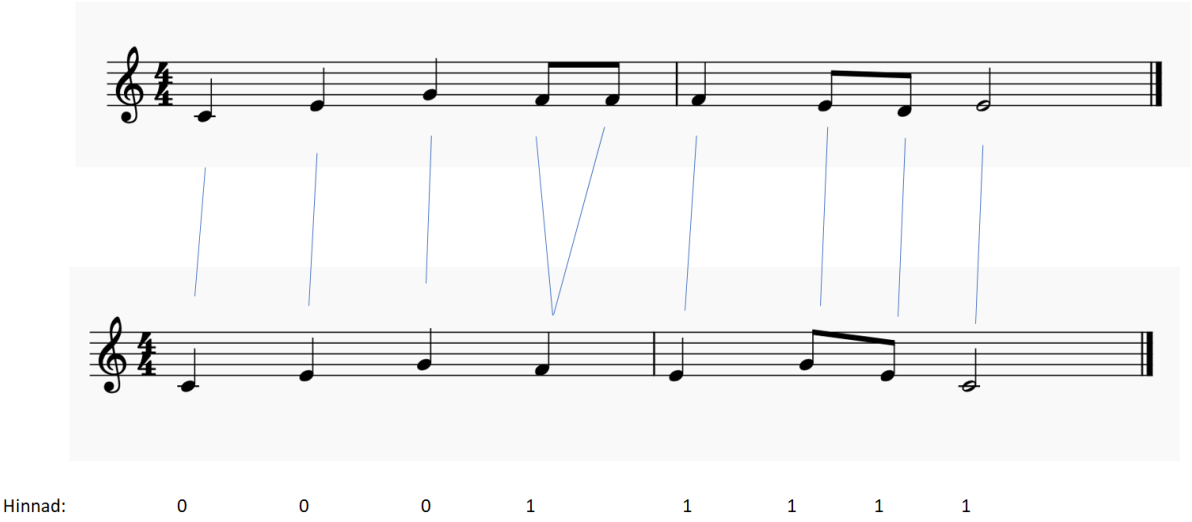
Joonis 3. Nootide geomeetriline esitus. [29]

Teisenduskauguse algoritme saab rakendada muusika andmeanalüütikas erinevatel alustel. N. Mitchell on oma töös [30] võrrelnud teisenduskauguse arvutamise meetodeid, lähtudes nii noodipikkustest, nootide kõrgusest, noodijärgnevuse regulaarsusest kui ka rõhulistest ja mitterõhulistest nootidest. Teisenduskauguse algoritmid on efektiivsed, kui soovitakse võrrelda, kas üks noodistus on teise täpne vaste või mitte. Erinevate noodistuste meloodiate omavahelise sarnasuse hindamiseks ei ole teisenduskauguste meetod parim, sest see arvestab ainult n-ö "muudatuste hinda", aga see hind ei pruugi olla tasakaalus muutustega meloodia nootides. N. Mitchell tõi oma töös esile, et teisenduskauguste arvutuste tulemusi erinevatel meetoditel võiks hinnata ka inimtaju põhjal. Eesmärgiks oleks mõista, kuidas hindavad inimesed noodijärgnevuste sarnasust - kui teisenduskaugus kahe noodijärgnevuse vahel on lähtudes nii noodipikkustest kui noodikõrgustest sama, siis milline neist tundub inimesele sarnasem.

Teisenduskauguste meetodite tulemuste võrdlus ei pea olema ainult meetoditevaheline. Joonisel 4 ja 5 on kujutatud olukorda, kus võrreldakse teisenduskaugusega ühte noodijärgnevust kahe erineva noodijärgnevusega, kus nootide lisamine, kustutamine kui ka asendamine on hinnaga 1. Mõlemal puhul üks kustutamine ja neli asendust ehk teisenduse hinnaks on 5. See tähendaks, et nii joonise 4 kui ja joonise 5 ülemised meloodiad on sama sarnased alumisele meloodiale. Ent kõlaliselt on need noodijärgnevused üsna erinevad.



Joonis 4. Noodijärgnevuste võrdluse teisenduskauguste abil



Joonis 5. Noodijärgnevuste võrdluse teisenduskauguste abil

Inimesed tajuvad muusikat erinevalt ning ühte kindlat vastust ei saa siin anda. Teine meetod, kuidas noodistuste meloodiate sarnasust hinnata, on leida noodijärgnevustega kattuvad mustrid. Samade mustrite leidmisel erinevatest noodijärgnevustest võib võrreldavate järgnevuse sarnasuse tuvastamine olla ka inimtajule üheselt mõistetavam.

2.2 Mustri sobitamise meetodid

Mustri sobitamise meetodite puhul võetakse aluseks üks kindla pikkusega muster (näiteks sümbolite jada) ning otsitakse, kas sama muster sisaldub mõnes teises jadas. Sümbolite mustri sobitamise meetodid on olnud kasutusel tekstitöötlusel ja bioinformaatikas, aga samuti leiavad kasutust ka muusika andmeanalüütikas. [22]

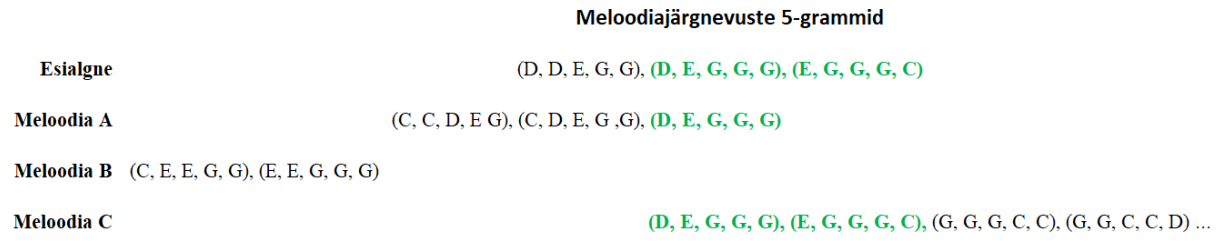
N-gramm on n-arvu pikkusega järjestikune sümbolite jada etteantud sümbolite järjestusest**.** Noodistuse puhul võib n-grammideks lugeda näiteks noodi- või intervallijärgnevusi. Meloodia sarnasuse määramiseks leitakse, kui palju on kahel võrreldaval meloodia jadal kattuvaid n-gramme. Mida rohkem on kattuvaid n-gramme, seda sarnasemad on meloodiad.

Olgu meloodia nootidega (D, D, E, G, G, G, C), millel on kolm 5-grammi, (D, D, E, G, G), (D, E, G, G, G) ja (E, G, G, G, C). Seda meloodiat võrreldakse kolme erineva meloodiajärgnevusega 5-grammide põhjal:

- Meloodia A, mille noodid on (C, C, D, E, G, G, G). Selle meloodia 5-grammid on:
    - (C, C, D, E G)
    - (C, D, E, G ,G)
    - (D, E, G, G, G)
- Meloodia B, mille noodid on (C, E, E, G, G, G). Selle meloodia 5-grammid on:
    - (C, E, E, G, G)
    - (E, E, G, G, G)
- Meloodia C, mille noodid on (D, E, G, G, G, C, C, D, E, G, G). Selle meloodia 5-grammid on:
    - (D, E, G, G, G),
    - (E, G, G, G, C)
    - (G, G, G, C, C)
    - (G, G, C, C, D)
    - (G, C, C, D, E)

- (C, C, D, E G)
- (C, D, E, G ,G)

Esialgse meloodiaga on kõige sarnasem meloodia C, sest meloodial C on kõige rohkem kattuvaid 5-gramme. Meloodia C esimene 5-gram kattub päringu meloodia teise 5-grammiga ja meloodia C teine 5-gram kattub päringu meloodia kolmanda 5-grammiga. Sarnasuselt järgmine on meloodia A, millel on ainult üks kattuv 5-gram ja meloodial B puuduvad kattuvad 5-grammid. [28] Kattumused on kuvatud joonisel 6.



Joonis 6. Meloodiajärgnevuste 5-grammide kattumus.

N-grammide puhul on üheks sarnasuse määrajaks kahe noodistuse puhul kattuvate n-grammide summa (ingl *Sum Common Measure).* Samuti saab ka sarnasuse mõõtmeks kokku lugeda unikaalsed kattuvad n-grammid (ingl *Count Distinct Measure)*. Muusika andmeanalüütikas on kasutatud ka veel Ukkoneni mõõtu, mis summeerib kokku hoopis kõik mittekattuvad n-grammid kahe noodistuse vahel. [31]

Noodijärgnevuste sarnasust hinnatakse ka pikima ühisjada põhjal (ingl *Longest Common Subsequence*). Sisuliselt leitakse suurim n-gramm, mis kattub kahe võrreldava noodistuse järgnevusega. Pikima järgnevuse leidmine tähendab vaikimisi seda, et kahe etteantud sümbolite jada puhul leitakse pikim ühine järgnevus, mis ei ole järjestikune. Näiteks sümbolite jadade ‘ABCDEA’ ja ‘AFCGEA’ pikim järgnevus on ‘ACEA’. [32] Noodijärgnevustel tuleks arvestada sümbolite järjestust, sest kui jätta sellisel meetodil mingid noodid vahele, võib see juba oluliselt muuta meloodiat. Veel uuritakse, millised neist sarnastest pikimatest järjestikkudest järgnevustest on korduvad ja omavahel mittekattuvad. Peamiselt kasutatakse sellist meetodit plagiaadituvastusel [13]**.**

2.3 Varasemad sarnaste noodijärgnevuste leidmise lahendused

MIREX-i võistlus on iga-aastaselt ajendanud teadlasi ja analüütikuid leidma muusika andmeanalüütikas uusi ja paremaid lahendusi. Sarnaste noodijärgnevuste leidmisel on aluseks võetud erinevaid failitüüpe (MIDI, diginoodistuse failid) ja rakendatud erinevaid meetodeid ning muusikaliste tunnuste kombinatsioone. MIREX-i üks võistlustest on olnud SMS (ingl *Symbolic Music Similarity*) ehk sümbolitel põhineva muusika sarnasuse leidmise probleem, kus osalejate ülesanne on olnud ühehäälse MIDI noodijärjestuse päringu alusel leida valitud teoste kogust kõige sarnasemad teosed [33]. Selles töös [34] on tehtud kokkuvõte erinevatest lahendustest, mida on SMS-i leidmisel rakendatud. Algoritmid on klassifitseeritud nende meetodi järgi - kas muusika tajumise, muusikateooria, matemaatika või eelnimetatud meetodite hübriidse lahenduse põhjal.

Sarnaste noodijärgnevuste leidmine on oluline plagiaadituvastusel. Mustri sobitamisel on kasutatud ka kahepoolseid (ingl *bipartite*) graafe. Noodijärgnevuste sarnasuste arvutamiseks rakendati optimeeritud hindadega teisenduskauguse meetodit. Hinnad optimeeriti järgmiselt [12]:

- Erinevatel helikõrgustel olevate nootide omavaheliseks hinnaks oli nende kõrguste toonide vahe absoluutväärtus.
- Eeldusel, et näiteks kaheksandiknoodi lisamine või kustutamine on meloodiliselt suurema tähtsusega kui kuueteistkümnendik noodi lisamine või kustutamine, kasutati hüperparameetrit noodipikkuste teisenduskauguse osatähtsuse määramiseks.
- Rõhuliste ja mitterõhuliste löökide osatähtsust määrati samuti hüperparameetriga, eeldades, et rõhulised löögid on tähtsamad.
- Konsonantside[7] absoluutse helikõrguse erinevuse puhul arvestati eraldi kustutamise hinda.

Meetodi hindamisfaasis saavutati hüperparameetrite erinevate väärtuste testimisel head tulemused võrreldes n-grammi põhjal sarnasuse arvutamise mõõtudega (*Sum Common Measure*, Ukkoneni mõõt). Tulemuste täpsust kinnitas ka leitud järgnevuste sarnasus kuulamisel. [13] Kirjeldatud lahendus oli inspiratsiooniks selle töö noodijärgnevuste sarnasuse leidmise lahenduse arendamisel.

[7] kahe või enama heli heakõlaline kooskõla, mis ei nõua lahendust [11] [35].

2.4 Sarnaste noodijärgnevuste leidmise rakendused

Sarnaselt MIREX-i SMS-i ülesande püstitusele on loodud ka rakendusi, kus samalaadseid noodijärgnevusi saab otsida samuti päringu põhjal.

2.4.1 Pythagoras - muusikaliste seoste avastamine ja visualiseerimine

Selle töö eesmärk oli analüüsida MusicXML diginoodistuse faile ja leida korduvaid mustreid erinevatest teostest, et luua seoseid muusika ja heliloojate vahel. Diginoodistuse failid teisendati CSV-formaati, kus iga rida tähistas ühte noodistuse nooti ning veergudes olid selle noodiga seotud tunnuste väärtused (nagu näiteks kõrgus ja kestus). Noodistuse sarnaseid mustreid otsiti regulaaravaldistega ning tulemused salvestati andmebaasi. Andmebaasi oli võimalik teha päring, mis sisaldas intervallide tooniliste suuruste järgnevust (näiteks 4, 3, -3, 1, 2) ning tulemuseks oli teoste arv ja teoste arv ja heliloojate nimekiri, kelle teostes selline intervallide järgnevus esines. [36]

2.4.2 Hollandi Laulu Andmebaas

Hollandi Laulu Andmebaas (ingl *Dutch Song Database*) on rakendus, kus on võimalik kasutaja sisestatud nootide järgi leida sarnaseid noodijärgnevusi üle 180000 hollandi- ja flaamikeelsete teoste digitaliseeritud noodistuste seast, mis võivad pärineda keskajast kuni 20. sajandini. Vastavalt päringule leitakse andmebaasist parim vaste, millises noodistuses sama noodijärgnevus esineb. Otsingufunktsioon töötab teisenduskauguse määramise põhimõttel, mis arvestab helikõrgust, noodipikkust ja fraaside kordumist asenduste arvutamisel [37]. [38]

3.4.3 Diginootide meloodiaanalüsaator

Autori varasemas töös loodi tarkvaralahendus digitaalsete nootide meloodia analüüsimiseks. Programm võimaldab võrrelda kahte noodistust ning kuvada noodistuste muusikaliste elementide kvantitatiivset informatsiooni nagu näiteks seda, millised on noodistuses enim esinevad noodid, intervallid ja akordid. Samuti arvutatakse kauguse tuvastamise algoritmi abil kahe noodistuse vaheline sarnasus intervallide tooniliste suuruste kaudu. Viimaks on võimalik leida võrreldavate noodistuste ühine pikim meloodiajärgnevus nootide liikumissuuna ja intervallide kaudu. [21]

2.4.4 Kokkuvõte varasematest lahendustest

Nii MIREX-i võistluste kui ka varasemate rakenduste puhul on levinud lahendused, kus antakse ette sümbolite muster, mis leiab üles päringu põhjal need noodistused, mis on sisendiga võrreldes kõige sarnasemad. Sellised rakendused on vähem levinud, mille sisendiks on terve noodistuse partiid, mitte ainult sümbolite jada. Eesmärk pole mitte leida, kas kasutaja sisestatud noodijärgnevus esineb mingis noodistuses, vaid otsida, kas sisendpartiil on kattuvaid järgnevusi mõne teise, andmestikus oleva, noodistuse partiiga. Autori varasemas töös [21] ainult ühte meetodit pikima sarnase noodijärgnevuse leidmiseks. Selle töö raames valmis lahendus, mille abil on võimalik tuvastada sarnaseid noodijärgnevusi erinevate meetodite kombinatsioonidega ja neid visualiseerida.

3. Diginoodistuste sarnasuste leidmine ning sarnaste palade järjestamine

Inspireerituna muusika sümbolitel põhinevate andmete analüüsi kitsaskohtadest ja varasematest töödest, kirjeldatakse selles peatükis magistritöö raames valminud diginoodistuste sarnaste noodijärgnevuste leidmise mudelit, arendusvalikuid ja tulemusi. Programmis[8] on võimalik üles laadida diginoodistuse andmestik (MusicXML-failid) ning määrata, millise noodistuse partiid soovitakse ülejäänud andmestiku partiidega võrrelda. Järgnevuste sarnasuse otsimiseks saab määrata soovile vastavad meetodid ning lõpuks reastatakse kõige pikema ühise noodijärgnevusega partiid. Sarnaste partiide noodijärgnevused on võimalik visualiseerida nii noodikirjas kui ka graafikul ja auditiivseks hindamiseks saab leitud noodijärgnevusi kuulata. Rakenduse eesmärk on võimaldada kasutajal uurida, kas noodistuste partiide vahel leidub sarnaseid järgnevusi ning kuidas erinevate muusikaliste elementide lisamine või eemaldamine mõjutavad leitud pikimaid ühiseid järgnevusi.

3.1 Kasutatud tehnoloogilised vahendid

Järgnevalt kirjeldatakse lahenduse arendamisel kasutatud tehnoloogilisi vahendeid. Lahendus on arendatud programmeerimiskeeles Python Jupyter Notebooki failina. Andmete töötlemiseks kasutati teeke Music21 ja Pandas. Noodistuste visualiseerimiseks rakendati programmi MuseScore 3. Lahendus on kirjutatud inglise keeles, et nii programmikood kui ka teisendatud andmed oleksid arusaadavad ka eesti keelt mittekõnelevatele inimestele. Lahendust on võimalik avada ka veebis, Google Colabis[9], kuid selle puhul ei tööta MIDI-failide genereerimine. Lahendust on testitud Windowsi operatsioonisüsteemi peal.

3.1.1 Jupyter Notebook

Jupyter Notebook on veebipõhine rakendus. See võimaldab luua ja jagada dokumente, mis koosnevad nii programmikoodist kui ka rikasteksti elementidest (ingl *rich text elements*). Samuti

[8] https://github.com/LudvigL123/digital_sheet_music_analysis

[9] https://colab.research.google.com/drive/1mFPlXeIAT6ybxUthAXqmwbgy10IWXvEz?usp=sharing

saab kuvada erinevaid interaktiivseid väljundeid nagu näiteks andmetabeleid, graafikuid, pilte. Jupyter Notebook toetab üle 40 erinevat programmeerimiskeelt, sealhulgas Pythonit. [39]

3.1.2 Python

Lahendus on arendatud programmeerimiskeeles Python (versioon 3.9). Pythoni keelel on olemas sobivad andmete töötlemise ning digitaalse noodikirja liigendamise teegid.

3.1.3 Music21

Music21 on Pythoni teek, mis võimaldab analüüsida muusika andmestikke, töödelda muusikafaile (sh noodikirja) ja luua muusikat [40]. Music21 on selles töös põhiline tööriist MusicXML-failide töötlemiseks, kuulamiseks ja visualiseerimiseks. Lahenduses kasutatakse Music21 versiooni 7.1.0.

3.1.4 Pandas

Pandas on Pythoni teek andmete töötlemiseks ja analüüsimiseks [41]. Selles töös kasutatakse Pandase teeki andmete loomiseks, hoidmiseks ning kuvamiseks.

3.1.5 MuseScore 3

Tulemuste visualiseerimiseks noodikirja kujul rakendati programmi MuseScore (3. versioon). See on rakendus noodistuste loomiseks, taasesitamiseks ja printimiseks [42]. Selle töö raames on MuseScore-i kasutatud ka jooniste tegemisel, mis sisaldavad noodikirja sümboleid.

3.2 Diginoodistuse andmete töötlus

Kasutajal on võimalik üles laadida lokaalsest kaustast diginoodistuste failid, mis teisendatakse analüüsiks sobivale kujule andmetabelis. Diginoodistused peavad olema MusicXML failiformaadis mxl- või musicxml-laiendiga. Andmetabelisse (joonis 7) sisestatakse failid kaustas olevas järjekorras järgneval kujul:

1) note_id - diginoodistuse identifikaator number;
2) note_part_id - diginoodistuse partii identifikaator number;
3) sheet_name - diginoodistuse faili nimi;
4) sheet_path - diginoodistuse failitee ehk inimloetav aadress failisüsteemis;

5) part_name - diginoodistuse partii instrument;
6) original_key - partii esialgne kõige tõenäolisem helistik, mis määratakse Music21 teegi Krumhansl-Schmuckleri algoritmiga [36] koos Krumhansl-Kessleri kaaludega [44]**;**
7) transposed_key - partii transponeerimise tulemusele määratakse kõige tõenäolisem helistik Music21 teegi Krumhansl-Schmuckleri algoritmiga [36] koos Krumhansl-Kessleri kaaludega [44];
8) measures - partii taktide andmestik.

	note_id	note_part_id	sheet_name	sheet_path	part_name	original_key	transposed_key	measures
0	1	1_1	Fr_Elise_WoO_59__Ludwig_va...	C:/Users/Lleis/Documents/S...	Piano	a minor	a minor	{0: (1, '76', 0.25, 0.0), ...
1	1	1_2	Fr_Elise_WoO_59__Ludwig_va...	C:/Users/Lleis/Documents/S...	Piano	a minor	a minor	{0: (1, 'rest_0.5'), 1: (2...
2	2	2_1	Fuer_Elise_WoO_59.mxl	C:/Users/Lleis/Documents/S...	Piano	g minor	a minor	{0: (0, '88', 0.25, 0.0), ...
3	2	2_2	Fuer_Elise_WoO_59.mxl	C:/Users/Lleis/Documents/S...	Piano	g minor	a minor	{0: (0, 'rest_0.5'), 1: (1...
4	3	3_1	Fur_Elise (1).mxl	C:/Users/Lleis/Documents/S...	Piano	a minor	a minor	{0: (1, '76', 0.5, 0.0), 1...
5	3	3_2	Fur_Elise (1).mxl	C:/Users/Lleis/Documents/S...	Piano	a minor	a minor	{0: (1, 'rest_3.0'), 1: (2...
6	4	4_1	FUR_ELISE.mxl	C:/Users/Lleis/Documents/S...	Piano	a minor	a minor	{0: (0, '76', 0.25, 0.0), ...
7	4	4_2	FUR_ELISE.mxl	C:/Users/Lleis/Documents/S...	Piano	a minor	a minor	{0: (0, 'rest_0.5'), 1: (1...
8	5	5_1	Fur_Elise__Reimagined_.mxl	C:/Users/Lleis/Documents/S...	Piano	f# minor	a minor	{0: (0, '81', 0.25, 0.0), ...
9	5	5_2	Fur_Elise__Reimagined_.mxl	C:/Users/Lleis/Documents/S...	Piano	f# minor	a minor	{0: (0, 'rest_0.5'), 1: (1...

Joonis 7. Andmetabel pärast noodistuse andmestiku töötlust.

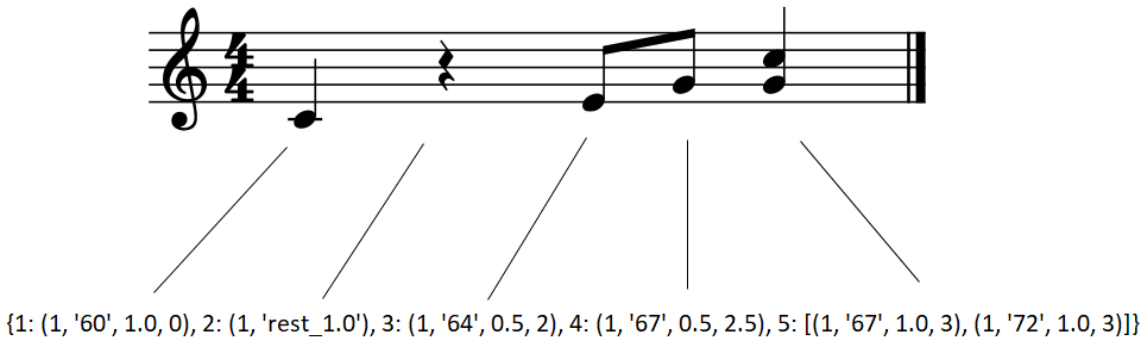
Taktide andmestik (veerg *measures*) on noodistuse analüüsimise kõige olulisem osa. Taktide andmestiku puhul pidi esmalt analüüsida, milliseid muusikalisi tunnuseid MusicXML-failid nootide, akordide ja pauside kohta sisaldavad. Seejuures tuli arvestada, et taktide andmestik oleks töödeldud sellisele kujule, et nii noodistuse elementide väärtusi kui ka nende muusikalisi tunnuseid oleks võimalik rakendada erinevate meetoditega pikimate ühiste järgnevuste otsimiseks.

Taktide andmestik on loodud sõnastikuna, kus igal noodil on oma identifikaator, mis võimaldab hiljem vastavusse viia leitud järgnevused noodistuse originaalse järgnevusega. Sõnastiku väärtused saavad olla kas noodid, kooskõlad või pausid. Nootide väärtused on kujul (a,b,c,d) (joonis 8), mis tähistavad järgmist:

- a - taktinumber;
- b - noodi helikõrguse MIDI-kood [45];
- c - noodi- või pausipikkus (lisa 2);

- d - väärtus, millisele löögile noot langeb - kas rõhulisel löögil (täisarv) või mitterõhulisel löögil (ujukomaarv).

Kui takti element on paus, siis see sisaldab kahte elementi, kus esimene neist on taktinumber ning teine sõne, kujul ‘rest_X’, kus X tähistab pausipikkust (näiteks ‘rest_1.0’ on veerandpaus). Kooskõlad koosnevad nootidest, kus esimesel kohal on kõige alumine kooskõla noot.



Joonis 8. Nootides teisendus taktide andmestikku.

Noodijärgnevuste leidmiseks transponeeritakse andmestiku noodistused intervallide kaudu kõik ühte helistikku, sealhulgas mažoorsed helistikud C-duuri ja minoorsed helistikud a-molli. Kui esialgselt tuvastatud helistik on C-duur või a-moll, siis transponeerimist ei tehta. See võimaldab leida noodijärgnevusi nootide, mitte intervallide põhjal. Intervallide põhjal järgnevuste leidmisel ei peaks noodistuse helistikke transponeerima, kuid sel juhul tekib teine takistus.

Olgu C-duuris noodistus, kus on noodijärgnevus C4-D4-E4-G4 (täht tähistab noodinime ja number oktavid), mille intervallid on S2-S2-V3, mis on võrdelised tooniliste suurustega 1-1-1.5. Samas noodistuses ja helistikus on noodid F4-G4-A4-C5 intervallidega S2-S2-V3, mis on samade tooniliste suurustega 1-1-1.5. Olenemata sellest, et intervallid on samad, on noodijärgnevuste noodid erinevad, mis tähendab, et nende kõla on erinev. Intervallide puhul ei arvestata helistikuga ning seetõttu võivad olla samade intervallide järgnevuste puhul nende absoluutsed noodikõrgused erinevad. Noodijärgnevuste leidmise eelis võrreldes intervallide järgnevuste leidmisega on see, et noodijärgnevuste puhul on võimalik kaasata noodiga seonduv info (noodipikkus, helikõrgus, rõhuline või rõhuta noot). Samuti saab järgnevuste leidmisel arvestada pausidega. Intervallide nimetused ja nende toonilised suurused on kirjeldatud lisas 3.

Noodistuste helistiku teisendamise tulemusi kontrolliti intervallide tooniliste suuruste kaudu, seda sellepärast, et veenduda, kas Music21 helistiku transponeerimise algoritm põhineb intervallidel. Originaalhelistikus nootidevahelisi intervallide toonilisi suuruseid võrreldi transponeeritud noodistuse nootide vaheliste intervallide tooniliste suurustega. Tulemused näitasid, et helistike transponeerimine ei muuda nootide omavahelisi intervalle.

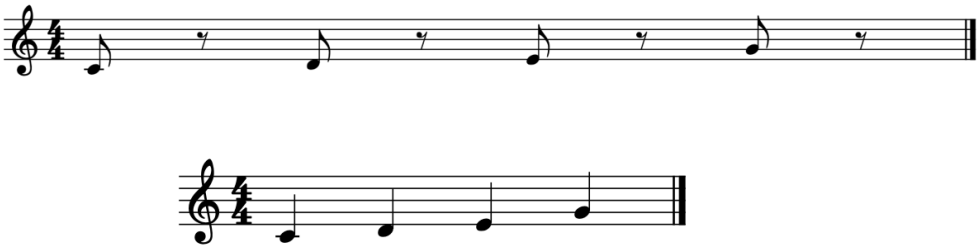
Noodistuste transponeerimine võib tekitada olukorra, kus noodistuste partiid ei ole kõik samas oktavis. Nootide oktavi muutumatus säiliks kui noodijärgnevusi võrrelda intervallidega, aga kuna selles töös leitakse sarnaseid järgnevusi nootide põhjal, siis on oluline, et noodijärgnevused oleksid samas oktavis, sest muidu mustrid ei sobitu. Selle takistuse eemaldamiseks tehti kitsendus, kus kontrollitakse noodijärgnevuste ühtimist transponeerimisel saadud oktavi nootidega ning ühe oktavi kõrgemal ja madalamal olevate nootidega.

3.3 Sarnaste noodijärgnevuste leidmine

Noodistuse andmestiku loomise järgselt saab kasutaja sisse lugeda noodistuse, mida soovitakse andmestikuga võrrelda. Seejärel on võimalik soovitud meetodiga otsida sarnaseid noodijärgnevusi. Selleks saab kasutaja otsida noodijärgnevusi kõikvõimalike kombinatsioonidega või teha valik soovitud meetodite hulgast.

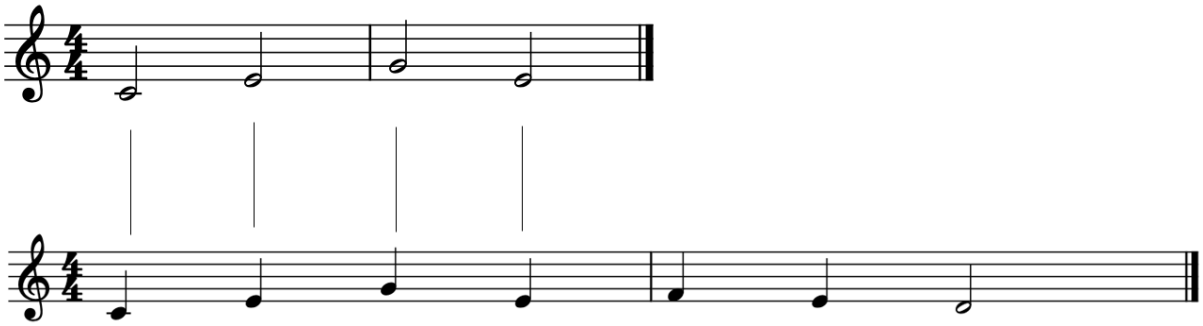
3.3.1 Meetodi valikud

- **Eemalda pausid** (ingl *Remove rests*) - Kui valik on aktiivne, siis noodijärgnevuste võrdlemisel eemaldatakse partiidest pausid. Paus on muusikas oluline sümbol, mis tähistab heli puudumist, kuid seoses noodikirja erinevate kirjutamisviisidega võib pauside eemaldamine tuvastada rohkem päringule sobivaid vasteid. Joonisel 9 on kuvatud kaks noodijärgnevust, mis on nootide poolest samad, kuid ülemisel järgnevusel on nootide vahel pausid. Pauside eemaldamisel muutub teoorias noodipikkus, et löökide arv taktis vastaks taktimõõdule, kuid andmestikus nootide pikkust ei muudeta.



Joonis 9. Näide pauside eemaldamisest.

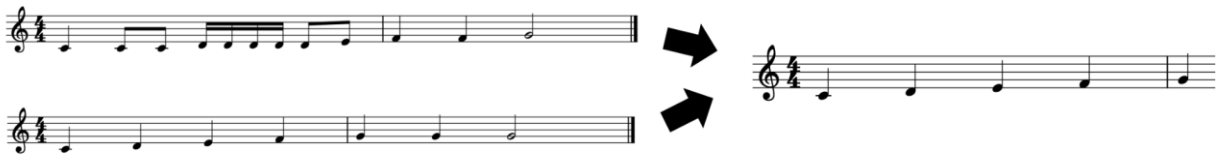
- **Eemalda akordid** (ingl *Remove chords*) - Kui valik on aktiivne, siis noodijärgnevuste leidmisel arvestatakse kooskõlade puhul ainult kooskõla alumist nooti. Kui ei ole aktiivne, siis mustri sobitamisel arvestatakse kooskõla kõiki noote.
- **Arvesta noodipikkust** (ingl *Consider note duration*) - Kui valik on aktiivne, siis noodijärgnevuste tuvastamisel arvestatakse originaalseid nootide pikkuseid. Kui valik ei ole aktiivne, siis sobitatakse mistahes pikkusega, kuid sama helikõrgusega noodijärgnevusi (joonis 10).



Joonis 10. Noodijärgnevuste sobitamine, kui noodipikkust ei arvestata.

- **Arvesta mitterõhulisi noote** (ingl *Consider offset notes*) - Muusikas langevad mõned noodid takti rõhulisele osale ja teised mitte. Kui see valik on aktiivne, siis arvestatakse noodijärgnevuste leidmisel kõiki originaalseid noote. Kui mitte, siis noodijärgnevustest eemaldatakse mitterõhulisele osale langevad noodid. Ainult rõhulisele osale langevaid noote on arvestatud muusika sarnasuse määramisel ka nendes töödes [28] [30]**.**

- **Arvesta korduvaid noote** (ingl *Consider repetitive notes***)** - Kui see valik on aktiivne, siis jäävad noodijärgnevused originaalkujule. Kui mitte, siis samade järjestikuste nootide puhul eemaldatakse kõik järjestikused korduvad noodid (joonis 11).



Joonis 11. Korduvate nootide eemaldamine.

Meetodite puhul, mis eemaldavad noodijärgnevustest elemente, ei ole oluline säilitada esialgse noodijärgnevuse rütmi struktuuri (näiteks asendada eemaldatud noodid sama pikkade pausidega, et löökide arv taktis oleks õige), sest takti elemente käsitletakse kui sümbolite jada. Olenevalt kasutaja valikust saab kas käivitada noodijärgnevuste otsingu üle kõikide andmestiku noodistuste partiide või võrrelda mõne konkreetse noodistuse partiiga.

3.3.2 Noodijärgnevuste leidmine

Noodijärgnevused teisendatakse sõne kujule, mis võimaldab luua nendest n-gramme. Sõne kujule teisendamisel teostatakse ka kõik kasutaja valitud meetodite operatsioonid. Esmalt luuakse kõik kahe noodistuse partiide järgnevuste 4-grammid. Valik alustada 4-grammidest tugineb autori isiklikult hinnangul. Väiksemaid kui neljanoodilisi järgnevusi on taasesitamisel keeruline tõlgendada sarnaste järgnevustena, sest nii lühikesed järgnevused ei oma kõlaliselt eristatavat ega unikaalset väärtust noodijärgnevuste leidmisel. N-grammid sobitatakse üksteisele ning n-grammide suurust tõstetakse ühekaupa nii kaua, kui enam ühtegi vastet ei anta. Sobitamine toimub Pythoni mooduli *difflib* SequenceMatcher algoritmi abil [46].

Kui pikim võimalik ühine noodijärgnevus on leitud, otsitakse kas sama pikki või lühemaid ühiseid noodijärgnevusi, mis teiste leitud noodijärgnevustega ei kattu. Tulemused reastatakse pikima sarnase noodijärgnevuse järgi ja suurima mittekattuvate sarnaste järgnevuste arvu põhjal. Tulemused salvestatakse tabelisse (joonis 12), mille veerud on järgmised:

1) note_part_ids - kombinatsioon noodistuste partiidest;
2) LCS (*Longest Common Sequence*) - pikim ühine järgnevus;

3) length_LCS - pikima ühise järgnevuse pikkus;
4) LCCNOS (*Longest Common Consequtive Non-Overlapping Sequence*) - pikimad omavahel mittekattuvad sarnased järgnevused;
5) size_LCCNOS - pikimate omavahel mittekattuvate sarnaste järgnevuste arv;
6) method_id - meetodi identifikaator;
7) remove_rests, remove_chords, consider_measure_idx, consider_note_duration, consider_offbeat, consider_repetition - meetodi valikuväärtused.

	note_part_ids	LCS	length_LCS	LCCNOS	size_LCCNOS	method_id	remove_rests	remove_chords	consider_note_duration	consider_offset_notes	c
31904	2_2 + 226_2	[(0, 0, (69...	14	[(16, 18, (...	1	TTFTT	True	True	False	True	
31905	2_2 + 226_2	[(0, 0, (69...	14	[(16, 18, (...	1	TTFTF	True	True	False	True	
31906	2_2 + 226_2	[(0, 0, (69...	14	[(16, 18, (...	1	TTFFT	True	True	False	False	
31907	2_2 + 226_2	[(0, 0, (69...	14	[(16, 18, (...	1	TTFFF	True	True	False	False	
31912	2_2 + 226_2	[(0, 0, (69...	14	[(16, 18, (...	1	TFFTT	True	False	False	True	
31913	2_2 + 226_2	[(0, 0, (69...	14	[(16, 18, (...	1	TFFTF	True	False	False	True	
31914	2_2 + 226_2	[(0, 0, (69...	14	[(16, 18, (...	1	TFFFT	True	False	False	False	
31915	2_2 + 226_2	[(0, 0, (69...	14	[(16, 18, (...	1	TFFFF	True	False	False	False	
31920	2_2 + 226_2	[(0, 0, (69...	14	[(16, 18, (...	1	FTFTT	False	True	False	True	
31921	2_2 + 226_2	[(0, 0, (69...	14	[(16, 18, (...	1	FTFTF	False	True	False	True	

Joonis 12. Järgnevuste leidmise tulemuste tabel.

Pikim ühine järgnevus (LCS) ja ka pikima järgnevusega mittekattuvad ühised järgnevused (LCCNOS) on andmetabelisse salvestatud kujul, kus mõlema noodistuse puhul on ära märgitud järgmised tunnused (joonis 13):

1) Noodi järjekorranumber töödeldud noodijärgnevuses. Selle abil on võimalik leida mittekattuvaid järgnevusi samast noodistuse partiist. Iga uue mittekattuva ühise järgnevuse puhul kontrollitakse, kas see moodustab osa leitud noodijärgnevustest.

2) Noodijärgnevus helikõrguste MIDI-koodidega. Näiteks järgnevus “60, 62, 64, 65, 67” tähistab nootides “C4, D4, E4, F4, G4”, kus täht näitab noodinime ning arv vastava noodi oktavi numbrit [45].

3) Noodijärgnevuse võtmeväärtused, mis viitavad originaalse noodistuse partii nootidele.

[(34,	Järgnevus algab sisendnoodistuse partii 34. elemendist.
311,	Järgnevus algab võrreldava noodistuse partii 311. elemendist.
('69', '67', '67', '76', '74', '72'),	Sisendnoodituse järgnevuse noodid.
('69', '67', '67', '76', '74', '72'),	Võrreldava noodistuse järgnevuse noodid
[41, 42, 43, 44, 45, 46],	Sisendnoodistuse nootide võtmeväärtused.
[660, 661, 662, 663, 664, 665])]	Võrreldava noodistuse nootide võtmeväärtused.

Joonis 13. Pikima ühise järgnevuse väärtuste kirjeldus.

3.3.3 Tulemuste visualiseerimine

Noodijärgnevuste visualiseerimiseks tuleb sarnased järgnevused tagasi teisendada originaalpartii noodijärgnevuseks. Selleks kasutatakse tulemustele liidetud noodistuste võtmeväärtusi. Esialgsest taktide andmestikust leitakse mõlema noodistuse partii puhul üles esialgsete nootide asukohad ning visualiseeritakse need taktid, kus järgnevus tuvastati. Noodistused jäetakse samasse (transponeeritud) helistikku, et kuulajal oleks lihtsam hinnata noodijärgnevuste sarnasust.

Noodijärgnevused esitatakse kolmel viisil:

1) Noodikirjana – mõlemast noodistusest kuvatakse tervikud taktid, kus sarnane järgnevus on tuvastatud. Kui järgnevuse algus on esimene noodistuse esimese takti viimane noot, siis kuvatakse kogu esimene takt. Noodikirja kuvamise jaoks on oluline, et kasutajal oleks installeeritud MuseScore 3 rakendus.

2) MIDI-failina – noodijärgnevusi on võimalik kuulata MIDI-failist, mille puhul on samuti genereeritud leitud noodijärgnevuste terviklikud taktid.

3) Graafikuna – graafiku horisontaalteljele on kujutatud noodistuse taktinumbrit ning vertikaalteljel noodi nime.

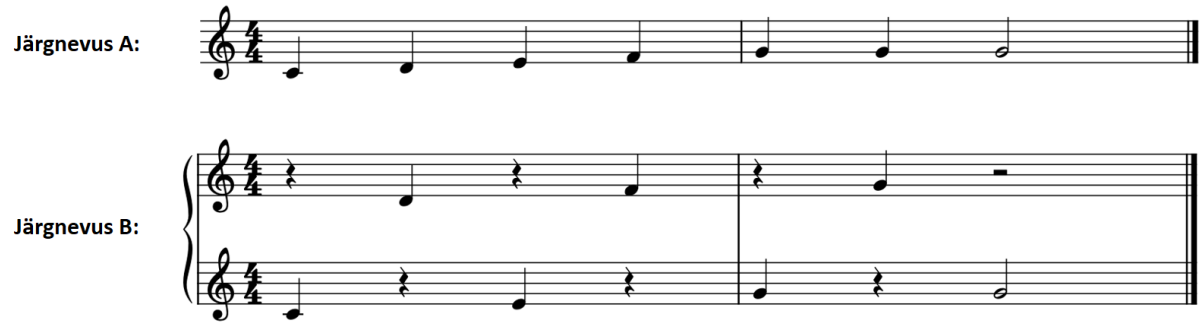
3.4 Tekkinud probleemid ja kitsendused

Lahenduse arendamise käigus ilmnesid mitmed takistused. Esiteks võib esineda probleeme diginoodistuste sisse lugemisel. Defektsete MusicXML-failide puhul võib tekkida veateateid, sest Music21 teek ei suuda noodistuse elemente liigendada (ingl *to parse*). Isegi kui sisse lugemine töötab, võivad inimeste loodud noodistused olla ebatavapäraselt struktureeritud (noodistus ilma taktijoonteta) või sisaldada Music21-le mitte teada noodikirja sümboleid.

Muusika meloodia võib noodistuses olla jagatud mitme partii peale. Sarnaste järgnevuste leidmise üks kitsaskoht on see, kui lineaarne noodijärgnevus on kirjutatud ühe partii asemel kahe või enama partii peale (joonis 14). Kuna järgnevusi võrreldakse partiide kaupa, ei tuvastata kahe noodistuse võrdlusel kõlaliselt pikimat ühehäälset järgnevust.

Teine kitsaskoht on, et noodistuste esialgsete helistike määramine ei ole täielikult täpne. See võib tekitada olukorra, kus noodistused ei transponeerita samale helistikule, mille tõttu väheneb võimalus leida pikimaid omavahelisi sarnaseid järgnevusi.

Sarnaste noodijärgnevuste MIDI-failide genereerimisel ei saa määrata esialgses noodistuses olevat esitamistempot. See on vaikimisi 120 lööki minutis. Samuti ei pruugi vahepeal MIDI-faili mängimisel olla heli või võib ka juhtuda, et MIDI-faili loomisel tekib viga. Täpne põhjus on teadmata, kuid võib eeldada, et see on seotud noodistuse struktuuriga ja Music21-l tekib diginoodistuse MIDI-failiks teisendamisel viga. Google Colabi versioonis pole võimalik järgnevusi kuulata, sest Google Colab ei toeta Music21-st MIDI-failide genereerimist.



Joonis 14. Sama noodijärgnevus ühe partii (järgnevus A) ja kahe partii peale (järgnevus B).

Lahenduse tulemuste kontroll on visuaalne või auditiivne. Hetkel puudub selline andmestik, mis sisaldaks sarnaste järgnevustega ekspertide poolt annoteeritud MusicXML-faile. Seetõttu pidi

tulemuste täpseks hindamiseks looma väikese kohandatud andmestiku, sest teisiti oleks kontrollimine olnud liiga ajakulukas.

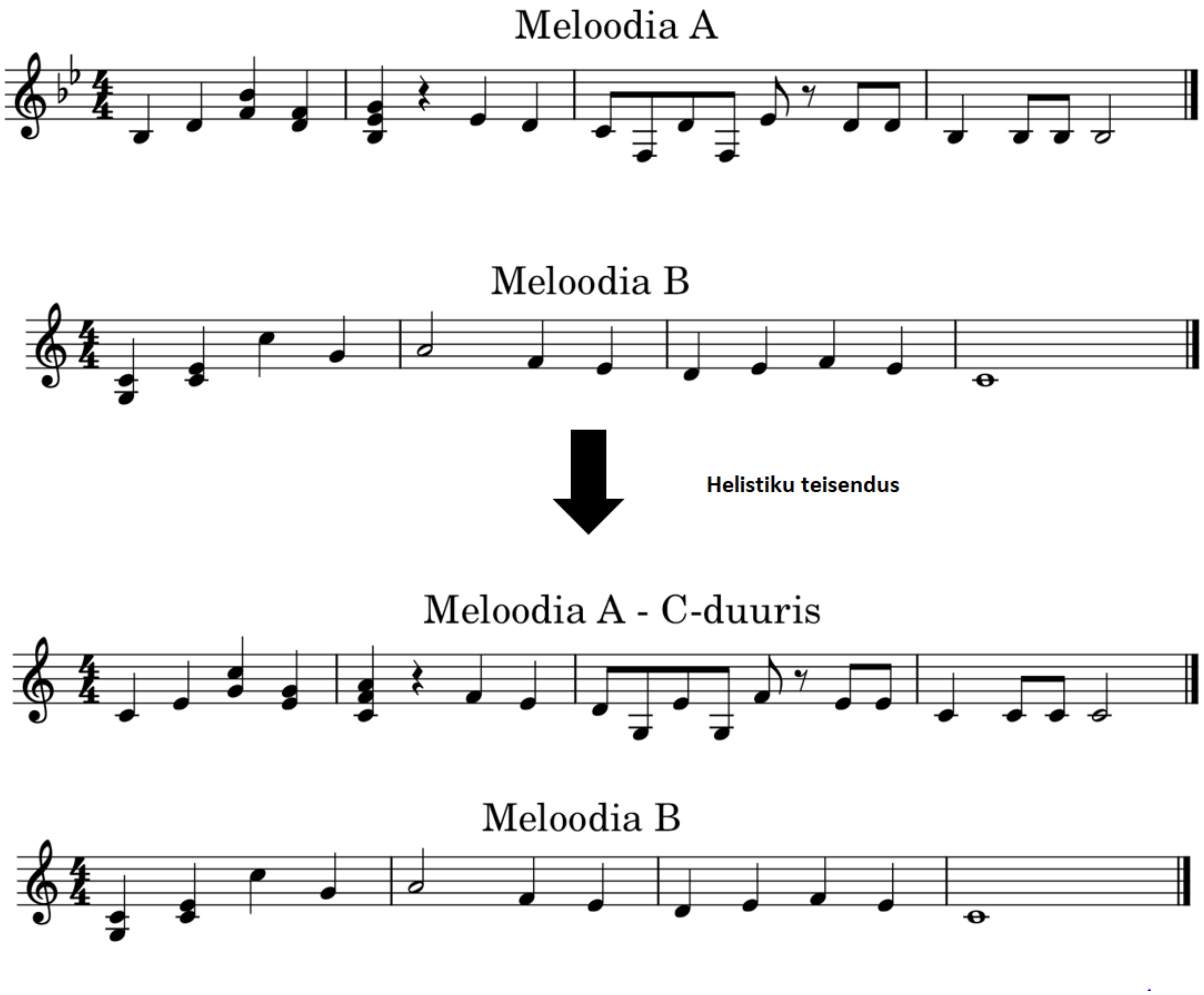
3.5 Testimine ja tulemused

Lahendust testiti kahel viisil. Esiteks kontrolliti sarnaste järgnevuste leidmise meetodite funktsionaalsust ning tulemusi. Teiseks testiti, kui hästi suudab rakendus leida töötlustevahelisi sarnaseid järgnevusi.

3.5.1 Järgnevuste leidmise meetodite funktsionaalne test

Lahenduse funktsionaalsusi testiti autori kahe loodud noodistusega. Noodistuste loomisel arvestati, et nende järgnevuste võrdlusel oleksid peatükis 3.3.1 kirjeldatud meetodi valikute funktsionaalsused kontrollitud. Võrreldavad noodistused transponeeriti samasse helistikku (joonis 15) ning töödeldi sõne genereerimiseks sobivale kujule (joonis 16). Sarnaste järgnevuste leidmine testiti läbi kõigi 32 kombinatsiooniga (viis tunnust, igal ühel kaks võimalikku väärtust ehk $2^5 = 32$). Kõige pikem järgnevus (joonis 17) leitakse kahel meetodil, kus on valitud järgmised tingimused:

- Eemalda pausid;
- Eemalda akordid;
- Ära arvesta noodipikkustega;
- Ära arvesta mitterõhuliste nootidega;
- Arvesta korduvate nootidega või ära arvesta korduvate nootidega.



Joonis 15. Meloodiajärgnevuste helistiku teisendus.

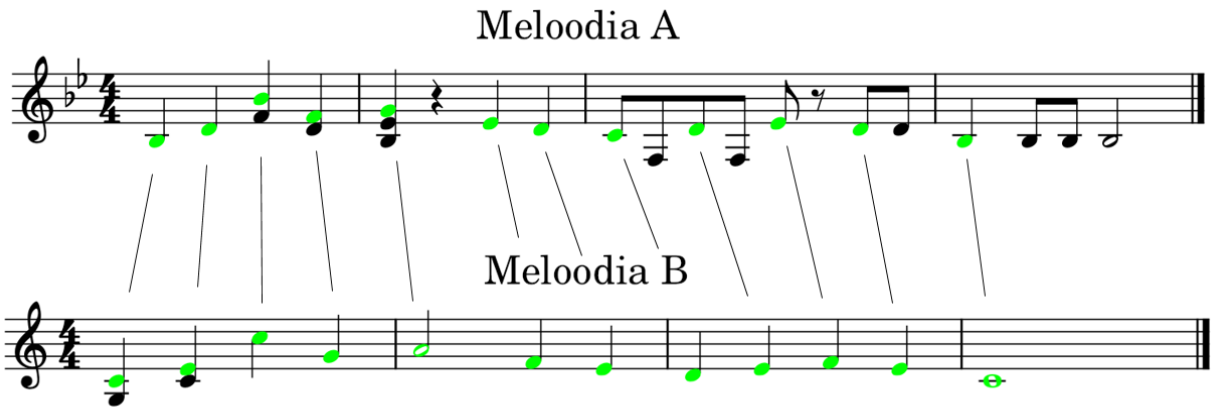
Meloodia A

```
{0: (1, '60', 1.0, 0.0),
 1: (1, '64', 1.0, 1.0),
 2: [(1, '67', 1.0, 2.0), (1, '72', 1.0, 2.0)],
 3: [(1, '64', 1.0, 3.0), (1, '67', 1.0, 3.0)],
 4: [(2, '60', 1.0, 0.0), (2, '65', 1.0, 0.0), (2, '69', 1.0, 0.0)],
 5: (2, 'rest_1.0'),
 6: (2, '65', 1.0, 2.0),
 7: (2, '64', 1.0, 3.0),
 8: (3, '62', 0.5, 0.0),
 9: (3, '55', 0.5, 0.5),
 10: (3, '64', 0.5, 1.0),
 11: (3, '55', 0.5, 1.5),
 12: (3, '65', 0.5, 2.0),
 13: (3, 'rest_0.5'),
 14: (3, '64', 0.5, 3.0),
 15: (3, '64', 0.5, 3.5),
 16: (4, '60', 1.0, 0.0),
 17: (4, '60', 0.5, 1.0),
 18: (4, '60', 0.5, 1.5),
 19: (4, '60', 2.0, 2.0)}
```

Meloodia B

```
{0: [(1, '55', 1.0, 0.0), (1, '60', 1.0, 0.0)],
 1: [(1, '60', 1.0, 1.0), (1, '64', 1.0, 1.0)],
 2: (1, '72', 1.0, 2.0),
 3: (1, '67', 1.0, 3.0),
 4: (2, '69', 2.0, 0.0),
 5: (2, '65', 1.0, 2.0),
 6: (2, '64', 1.0, 3.0),
 7: (3, '62', 1.0, 0.0),
 8: (3, '64', 1.0, 1.0),
 9: (3, '65', 1.0, 2.0),
 10: (3, '64', 1.0, 3.0),
 11: (4, '60', 4.0, 0.0)}
```

Joonis 16. Meloodiajärgnevuste taktid andmetabelis.



Joonis 17. Tuvastatud pikim sarnane järgnevus kahe esialgse noodistuse vahel oma vastavates helistikes.

Kõikide meetodite kombinatsioonid ning nende tulemused on kuvatud lisas 4.

3.5.2 Sarnaste järgnevuste leidmine teoste töötluste vahel

Veebilehelt Musescore.com laeti alla 30 noodistust. Nende hulgas oli kuue erineva teose viis erinevat töötlust. Iga teose puhul võeti suvaliselt üks töötlus, mida testiti kõigi teiste noodistustega ehk kokku tehti kuus testi ja igat noodistust võrreldi 24 noodistusega. Noodistuste puhul võrreldi omavahel kõiki partiisid 32 erineval võimalikul meetodil. Tulemused reastati (alates suurimast) pikima ühise noodijärgnevuse ja pikimate omavahel mittekattuvate

noodijärgnevuste järgi. Eesmärk oli tuvastada, kas lahendus määrab töötlused sarnasemateks noodistusteks või mitte.

Beethoven - Piano Sonata No. 14 (Moonlight sonata)

Esimene test näitas, et kõik sisendnoodistuse neli töötlust on märgitud kõige sarnasemateks (joonis 18). Sisendnoodistuse mõlemad partiid on saanud kõige sarnasemateks vasteteks oma töötluste partiid.

	note_part_ids	length_LCS	input_note	target_note
864	1_1 + 14_1	48	Beethoven_-_Mondscheinsonate_-_Moonlight_Sonat...	Opus_27_No_2_Moonlight_Sonata_1st_movement.mxl
736	1_1 + 12_1	48	Beethoven_-_Mondscheinsonate_-_Moonlight_Sonat...	Moonlight_Sonata_I.mxl
1272	1_1 + 20_1	45	Beethoven_-_Mondscheinsonate_-_Moonlight_Sonat...	Sonate_No._14_Moonlight_1st_Movement.mxl
672	1_1 + 11_1	45	Beethoven_-_Mondscheinsonate_-_Moonlight_Sonat...	Moonlight_Sonata_1st_movement.mxl
2704	1_2 + 14_2	32	Beethoven_-_Mondscheinsonate_-_Moonlight_Sonat...	Opus_27_No_2_Moonlight_Sonata_1st_movement.mxl
3075	1_2 + 20_2	26	Beethoven_-_Mondscheinsonate_-_Moonlight_Sonat...	Sonate_No._14_Moonlight_1st_Movement.mxl
2524	1_2 + 11_2	26	Beethoven_-_Mondscheinsonate_-_Moonlight_Sonat...	Moonlight_Sonata_1st_movement.mxl
2579	1_2 + 12_2	26	Beethoven_-_Mondscheinsonate_-_Moonlight_Sonat...	Moonlight_Sonata_I.mxl
2432	1_2 + 10_2	10	Beethoven_-_Mondscheinsonate_-_Moonlight_Sonat...	Lacrimosa_original_transcription_for_piano_-_M...
2817	1_2 + 16_2	9	Beethoven_-_Mondscheinsonate_-_Moonlight_Sonat...	PRELUDE_in_E_minor_-_Op._28_N._4.mxl

Joonis 18. Esimese testi tulemused.

Pachebel - Canon in D

Teise testi tulemused olid sarnased esimese testi tulemusele, kus kõigi töötluste partiid määrati kõige sarnasemateks sisendnoodistuse partiidele (joonis 19).

	note_part_ids	length_LCS	input_note	target_note
1696	1_2 + 1_2	159	Canon_in_D (1).mxl	Canon_in_D_-_Pachelbel.mxl
13	1_1 + 1_1	158	Canon_in_D (1).mxl	Canon_in_D_-_Pachelbel.mxl
144	1_1 + 3_1	124	Canon_in_D (1).mxl	Canon_in_D_Pachebel._Piano_Version_by_AC.mxl
1845	1_2 + 3_2	82	Canon_in_D (1).mxl	Canon_in_D_Pachebel._Piano_Version_by_AC.mxl
1789	1_2 + 2_2	67	Canon_in_D (1).mxl	Canon_in_D_lemontart.mxl
80	1_1 + 2_1	50	Canon_in_D (1).mxl	Canon_in_D_lemontart.mxl
893	1_1 + 14_1	48	Canon_in_D (1).mxl	Pachelbel_Canon_in_D_Fast_Easy_Version.mxl
2578	1_2 + 14_2	17	Canon_in_D (1).mxl	Pachelbel_Canon_in_D_Fast_Easy_Version.mxl
1536	1_1 + 23_1	7	Canon_in_D (1).mxl	We_Wish_You_A_Merry_Christmas- Arthur Warrell.mxl
1185	1_1 + 19_1	7	Canon_in_D (1).mxl	Von_Elise_-_Jazz_fr_Elise.mxl

Joonis 19. Teise testi tulemused.

Chopin - Prelude in E minor

Kolmanda testi tulemus näitas, et esimese kümne sarnaseima noodistuse seas esinevad kõik neli unikaalset sisendnoodistuse töötlust (joonis 20).

	note_part_ids	length_LCS	input_note	target_note
946	1_1 + 15_1	61	Prelude_in_E_Minor.mxl	PRELUDE_in_E_minor_-_Op._28_N._4.mxl
192	1_1 + 4_1	49	Prelude_in_E_Minor.mxl	Chopin_Prelude_in_E_Minor_Opus_28_No_4.mxl
284	1_1 + 5_1	46	Prelude_in_E_Minor.mxl	F._Chopin_Prelude_in_E_minor_Opus_28_no_4_-_Ar...
1020	1_1 + 16_1	46	Prelude_in_E_Minor.mxl	Prelude_No._4_in_E_Minor.mxl
2001	1_2 + 5_3	28	Prelude_in_E_Minor.mxl	F._Chopin_Prelude_in_E_minor_Opus_28_no_4_-_Ar...
1889	1_2 + 4_2	28	Prelude_in_E_Minor.mxl	Chopin_Prelude_in_E_Minor_Opus_28_No_4.mxl
2641	1_2 + 15_2	28	Prelude_in_E_Minor.mxl	PRELUDE_in_E_minor_-_Op._28_N._4.mxl
2705	1_2 + 16_2	28	Prelude_in_E_Minor.mxl	Prelude_No._4_in_E_Minor.mxl
1968	1_2 + 5_2	28	Prelude_in_E_Minor.mxl	F._Chopin_Prelude_in_E_minor_Opus_28_no_4_-_Ar...
2339	1_2 + 11_1	6	Prelude_in_E_Minor.mxl	Moonlight_Sonata_I.mxl

Joonis 20. Kolmanda testi tulemused.

Mozart - Lacrimosa

Neljanda testi tulemus näitas, et esimese kümne sarnaseima noodistuse seas esinevad kõik neli unikaalset sisendnoodistuse töötlust (joonis 21).

	note_part_ids	length_LCS	input_note	target_note
496	1_1 + 8_1	24	Lacrimosa.mxl	Lacrimosa_-_Requiem.mxl
760	1_1 + 12_1	24	Lacrimosa.mxl	Mozart_Lacrimosa.mxl
1373	1_1 + 20_1	24	Lacrimosa.mxl	W._A._Mozart_-_Lacrimosa_-_Requiem_Mass_in_d_m...
560	1_1 + 9_1	23	Lacrimosa.mxl	Lacrimosa_original_transcription_for_piano_-_M...
2401	1_2 + 12_1	17	Lacrimosa.mxl	Mozart_Lacrimosa.mxl
3009	1_2 + 20_1	17	Lacrimosa.mxl	W._A._Mozart_-_Lacrimosa_-_Requiem_Mass_in_d_m...
2144	1_2 + 8_1	17	Lacrimosa.mxl	Lacrimosa_-_Requiem.mxl
3860	1_3 + 8_2	11	Lacrimosa.mxl	Lacrimosa_-_Requiem.mxl
3924	1_3 + 9_2	11	Lacrimosa.mxl	Lacrimosa_original_transcription_for_piano_-_M...
4720	1_3 + 20_2	11	Lacrimosa.mxl	W._A._Mozart_-_Lacrimosa_-_Requiem_Mass_in_d_m...

Joonis 21. Neljanda testi tulemused.

Mozart - Für Elise

Viies test määras kõik neli erinevat töötlust kõige sarnasemateks, aga seekord mitte kõikide töötluste partiide raames (joonis 22). Näiteks leidus kümne sarnaseima teose hulgas ka üks Mozart - Lacrimosa töötlus (7. kohal), mis määrati sarnasemaks kui üks sisendnoodistuse töötlus Von_Elise_-_Jazz_fr_Elise.mxl (8. kohal) sellepärast, et Lacrimosa töötluse noodistuses leidus ka üks pikima järgnevusega mittekattuv unikaalne noodijärgnevus.

	note_part_ids	length_LCS	input_note	target_note
1057	1_1 + 17_1	62	FUR_ELISE.mxl	Ryans_version_of_Fur_Elise.mxl
1281	1_1 + 19_4	31	FUR_ELISE.mxl	Von_Elise_-_Jazz_fr_Elise.mxl
2752	1_2 + 17_2	30	FUR_ELISE.mxl	Ryans_version_of_Fur_Elise.mxl
354	1_1 + 6_1	19	FUR_ELISE.mxl	Fur_Elise (1).mxl
436	1_1 + 7_1	17	FUR_ELISE.mxl	Fur_Elise__Reimagined_.mxl
2114	1_2 + 7_2	12	FUR_ELISE.mxl	Fur_Elise__Reimagined_.mxl
1345	1_1 + 20_1	8	FUR_ELISE.mxl	W._A._Mozart_-_Lacrimosa_-_Requiem_Mass_in_d_m...
1195	1_1 + 19_1	8	FUR_ELISE.mxl	Von_Elise_-_Jazz_fr_Elise.mxl
145	1_1 + 3_1	7	FUR_ELISE.mxl	Canon_in_D_Pachebel._Piano_Version_by_AC.mxl
72	1_1 + 2_1	7	FUR_ELISE.mxl	Canon_in_D_lemontart.mxl

Joonis 22. Viienda testi tulemused.

We Wish You a Merry Christmas (traditsiooniline inglise jõululaul)

Kuuenda testi tulemused erinesid eelnevate testide tulemustest. Nimelt määrati kõige sarnasemateks kõigist neljast töötlusest ainult kaks tükki (joonis 23). Ülejäänud kaks töötlust mahtusid kümne sarnaseima vaste sisse, aga tuvastatud noodijärgnevuse pikkused olid lühikesed - 6 nooti. Töötluste noodikirja lähemal analüüsil avastati oluline nüanss.

Sisendnoodistus kõlab F-duuris, kuid see on noodigraafikaprogrammi abil sisestatud hoopis C-duuris. Noodistuse algusesse pole kirjutatud võtmemärke ning terve noodistuse raames on F-duurile omase ühe võtmemärgi asemel kasutatud läbivalt h-noodi madaldamist juhusliku märgi abil. Küll aga tunneb helistiku tuvastamise algoritm ära, et tegemist on F-duuris noodistusega

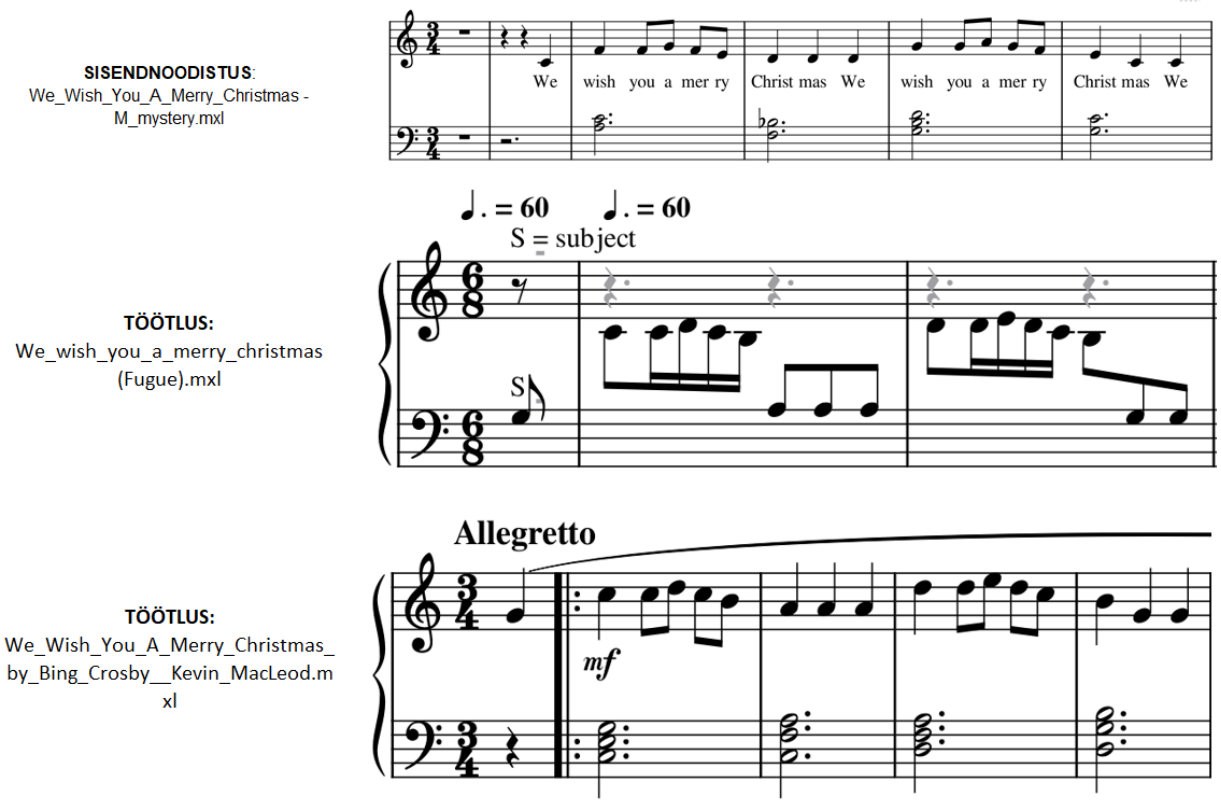
ning seetõttu transponeeritakse noodistus C-duuri, et kõik noodistused ja nende partiid oleksid samas helistikus (vt ptk 3.2).

Joonisel 23 viimasel real oleva sisendnoodistuse töötlus sisaldab modulatsiooni (üleminek ühest helistikust teise), kus esialgsest C-duurist minnakse D-duuri. Music21 helistiku määramise algoritm määrab tegelikuks helistikuks noodistuse üleselt hoopis G-duuri, mistõttu transponeeritakse ka noodi esimene osa neli tooni üles ehk töötluse esimene osa transponeeritakse F-duuri ja töötluse teine osa G-duuri. Helistiku tuvastuse algoritmi ebatäpsuse tõttu ei tuvastada sisendnoodistuse ja töötluse vahel pikemaid sarnaseid järgnevusi.

Põhjus, miks sisendnoodistuse ning 9. kohal oleva töötluse vahel on sarnane noodijärgnevus lühike, on autori hinnangul nende töötluste erinev noodikirja struktuur (joonis 24). Järgnevuse noodid on ära jaotatud kahe partii vahele nagu joonisel 14. Samuti sisaldab töötlus palju rohkem ornamente, mis vähendab võimalust leida pikema suurusega noodijärgnevusi.

	note_part_ids	length_LCS	input_note	target_note
1472	1_1 + 22_1	50	We_Wish_You_A_Merry_Christmas - M_mystery.mxl	We_Wish_You_a_Merry_Christmas mr faboulous pia...
3169	1_2 + 22_2	19	We_Wish_You_A_Merry_Christmas - M_mystery.mxl	We_Wish_You_a_Merry_Christmas mr faboulous pia...
1536	1_1 + 23_1	14	We_Wish_You_A_Merry_Christmas - M_mystery.mxl	We_Wish_You_A_Merry_Christmas- Arthur Warrell.mxl
737	1_1 + 12_1	7	We_Wish_You_A_Merry_Christmas - M_mystery.mxl	Mozart_Lacrimosa.mxl
481	1_1 + 8_1	7	We_Wish_You_A_Merry_Christmas - M_mystery.mxl	Lacrimosa_-_Requiem.mxl
882	1_1 + 14_1	7	We_Wish_You_A_Merry_Christmas - M_mystery.mxl	Pachelbel_Canon_in_D_Fast_Easy_Version.mxl
16	1_1 + 1_1	7	We_Wish_You_A_Merry_Christmas - M_mystery.mxl	Canon_in_D_-_Pachelbel.mxl
80	1_1 + 2_1	6	We_Wish_You_A_Merry_Christmas - M_mystery.mxl	Canon_in_D_lemontart.mxl
1432	1_1 + 21_1	6	We_Wish_You_A_Merry_Christmas - M_mystery.mxl	We_wish_you_a_merry_christmas (Fugue).mxl
1616	1_1 + 24_1	6	We_Wish_You_A_Merry_Christmas - M_mystery.mxl	We_Wish_You_A_Merry_Christmas_by_Bing_Crosby__...

Joonis 23. Kuuenda testi tulemused.



Joonis 24. We Wish You A Merry Christmas - erinevate töötluste noodistuse kirjapilt.

3.5.3 Sarnasuste leidmise meetodite analüüs J. S. Bachi noodistuste kogu põhjal

Music21 teek sisaldab J. S. Bach-i noodistuste kogu, milles on 413 MusicXML-faili. Noodistusi saab võrrelda failis *bach_analysis.ipynb* või Google Colabis[10]. Noodistused võib töödelda järgnevuste leidmiseks sobivale kujule otse Music21 kogust või sisse lugeda juba valmis töödeldud failist *bach_corpus.csv*. Andmestikus oli 1755 erinevat partiid, millest kõiki testiti omavahel andmestikus olevate partiidega (v.a iseendaga) läbi kõigi 32 erineva meetodiga (kokku 1755*1754*32 = 98,504,640 mõõtmist). Testimise eesmärk oli hinnata, milline mõju on tulemustele sarnase noodijärgnevuse leidmise meetoditel.

Noodistused võivad sisaldada palju erinevaid noote, pause, kooskõlasid ja ornamente. Autori püstitatud hüpotees oli, et kui noodistuse elementide detailsust vähendada (ehk elemente eemaldada ja noote kokku võtta), siis on võimalik leida rohkem sarnaseid noodijärgnevusi. Kõigi 32 meetodi puhul on enim elemente kas eemaldatud või kokku võetud järgmistel tingimustel:

[10] https://colab.research.google.com/drive/1XprlA764J-yNNRw0fVPGVq7LKpZB5uFp?usp=sharing

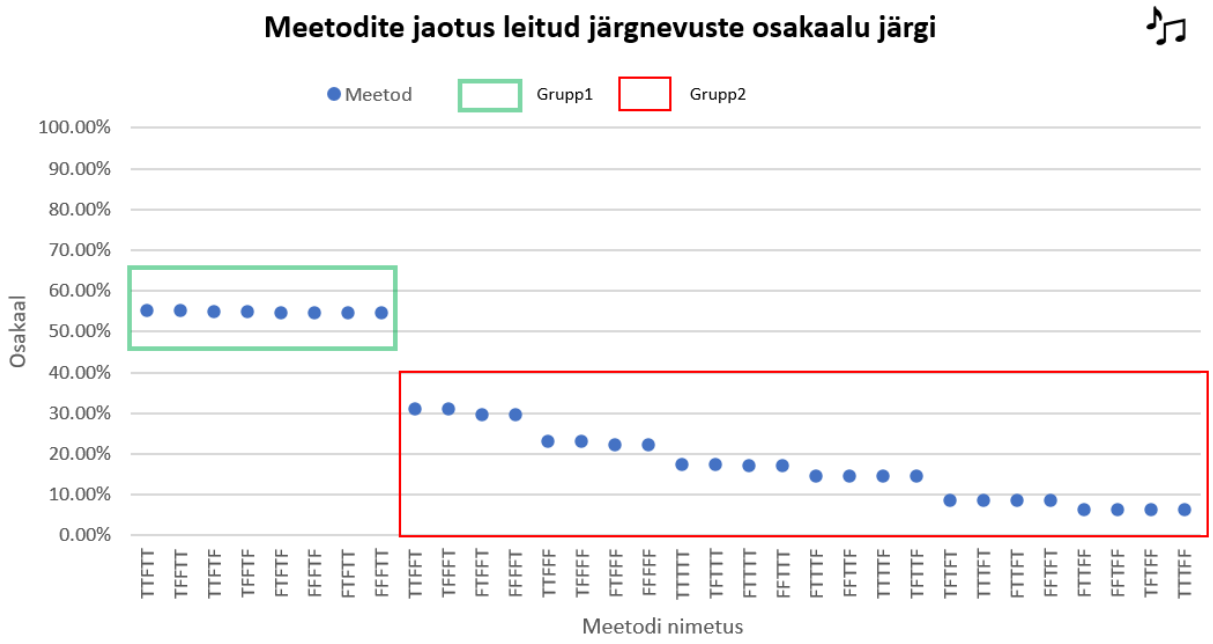
- Eemalda pausid;
- Eemalda akordid;
- Ära arvesta noodipikkustega;
- Ära arvesta mitterõhuliste nootidega;
- Ära arvesta korduvate nootidega.

Joonisel 24 on esitatud koondtabel kõigi mõõtmiste tulemustest iga meetodi kohta. Iga unikaalse meetodiga on tehtud 3,078,270 mõõtmist. Veerus *count_seq_exist* on näha tulemuste arvu, kus vähemalt nelja elemendi pikkune järgnevus leiti ning veerus *count_seq_none* arvu, mille puhul järgnevust ei tuvastatud. Samuti on tabelis märgitud ka leitud järgnevuste osakaalud meetodi põhiselt. Eelnevalt püstitatud hüpoteesi tingimustele vastab meetod TTFFF, mis on joonisel 24 real 12. Kuna kõikidest TTFFF meetodiga mõõtmistest leiti 23.1% kordadest sarnane järgnevus, siis lükkab see tulemus hüpoteesi ümber.

Tulemustest ei saa kindlalt väita, kas üks meetod on teistest selgelt tulemuslikum või mitte. Küll aga võib öelda, et mõned meetodid on halvemad sarnaste järgnevuste otsimiseks. Ühine tunnus on see, kui arvestatakse nootide pikkust ning samaaegselt ei võeta arvesse mitterõhulisi noote (joonis 25, grupp 2). Vastupidiselt on parimad tulemused saavutatud meetoditel, kus noodipikkust ei arvestata ja mitterõhulisi noote arvestatakse (joonis 25, grupp 1). Seega ei pea püstitatud väide paika, et noodistuse partiist elementide eemaldamine toetab järgnevuste leidmist, vaid mitterõhuliste nootide arvestamine on sarnaste noodijärgnevuste leidmisel oluline. Mitterõhuliste nootide kaasamine suurendab erinevate nootide hulka partiis ja seega hoopis soodustab erinevate noodijärgnevuste leidmist. Küll aga ei ole võimalik tuvastada valepositiivsete ja valenegatiivsete järgnevuste hulka, ilma et neid järgnevusi kuulata ja nende sarnasust hinnata.

	method_id	remove_rests	remove_chords	consider_note_duration	consider_offset_notes	consider_repetition	count_seq_exist	exist_%	count_seq_none	none_%
0	TTFTT	TRUE	TRUE	FALSE	TRUE	TRUE	1697973	55.20%	1380297	44.80%
1	TFFTT	TRUE	FALSE	FALSE	TRUE	TRUE	1697381	55.10%	1380889	44.90%
2	TTFTF	TRUE	TRUE	FALSE	TRUE	FALSE	1693861	55.00%	1384409	45.00%
3	TFFTF	TRUE	FALSE	FALSE	TRUE	FALSE	1693399	55.00%	1384871	45.00%
4	FTFTF	FALSE	TRUE	FALSE	TRUE	FALSE	1682920	54.70%	1395350	45.30%
5	FFFTF	FALSE	FALSE	FALSE	TRUE	FALSE	1682484	54.70%	1395786	45.30%
6	FTFTT	FALSE	TRUE	FALSE	TRUE	TRUE	1682227	54.60%	1396043	45.40%
7	FFFTT	FALSE	FALSE	FALSE	TRUE	TRUE	1681709	54.60%	1396561	45.40%
8	TTFFT	TRUE	TRUE	FALSE	FALSE	TRUE	953728	31.00%	2124542	69.00%
9	TFFFT	TRUE	FALSE	FALSE	FALSE	TRUE	953680	31.00%	2124590	69.00%
10	FTFFT	FALSE	TRUE	FALSE	FALSE	TRUE	914887	29.70%	2163383	70.30%
11	FFFFT	FALSE	FALSE	FALSE	FALSE	TRUE	914869	29.70%	2163401	70.30%
12	TTFFF	TRUE	TRUE	FALSE	FALSE	FALSE	712103	23.10%	2366167	76.90%
13	TFFFF	TRUE	FALSE	FALSE	FALSE	FALSE	712082	23.10%	2366188	76.90%
14	FTFFF	FALSE	TRUE	FALSE	FALSE	FALSE	685623	22.30%	2392647	77.70%
15	FFFFF	FALSE	FALSE	FALSE	FALSE	FALSE	685611	22.30%	2392659	77.70%
16	TTTTT	TRUE	TRUE	TRUE	TRUE	TRUE	531975	17.30%	2546295	82.70%
17	TFTTT	TRUE	FALSE	TRUE	TRUE	TRUE	531954	17.30%	2546316	82.70%
18	FTTTT	FALSE	TRUE	TRUE	TRUE	TRUE	530477	17.20%	2547793	82.80%
19	FFTTT	FALSE	FALSE	TRUE	TRUE	TRUE	530461	17.20%	2547809	82.80%
20	FTTTF	FALSE	TRUE	TRUE	TRUE	FALSE	446184	14.50%	2632086	85.50%
21	FFTTF	FALSE	FALSE	TRUE	TRUE	FALSE	446170	14.50%	2632100	85.50%
22	TTTTF	TRUE	TRUE	TRUE	TRUE	FALSE	445726	14.50%	2632544	85.50%
23	TFTTF	TRUE	FALSE	TRUE	TRUE	FALSE	445711	14.50%	2632559	85.50%
24	TFTFT	TRUE	FALSE	TRUE	FALSE	TRUE	263743	8.60%	2814527	91.40%
25	TTTFT	TRUE	TRUE	TRUE	FALSE	TRUE	263743	8.60%	2814527	91.40%
26	FTTFT	FALSE	TRUE	TRUE	FALSE	TRUE	262678	8.50%	2815592	91.50%
27	FFTFT	FALSE	FALSE	TRUE	FALSE	TRUE	262666	8.50%	2815604	91.50%
28	FTTFF	FALSE	TRUE	TRUE	FALSE	FALSE	190228	6.20%	2888042	93.80%
29	FFTFF	FALSE	FALSE	TRUE	FALSE	FALSE	190221	6.20%	2888049	93.80%
30	TFTFF	TRUE	FALSE	TRUE	FALSE	FALSE	189633	6.20%	2888637	93.80%
31	TTTFF	TRUE	TRUE	TRUE	FALSE	FALSE	189633	6.20%	2888637	93.80%

Joonis 24. Meetodite tulemused sarnaste järgnevuste leidmisel.



Joonis 25. Meetodite jaotus leitud järgnevuste osakaalu järgi.

3.5.4 Testimise kokkuvõte

Testimise tulemused näitasid, et magistritöö raames valminud lahendus täidab oma eesmärki. Erinevate meetodite kombinatsioonidega on võimalik tuvastada noodistuste vahel sarnaseid järgnevusi, neid visualiseerida ja taasesitada. Samuti leiti, et rakendus klassifitseerib noodistuse töötlused omavahel sarnasemaks ja leiab nendest ühiseid järgnevusi. Küll aga on tähtis, et noodistuste helistik oleks õigesti määratud. Noodistuste töötluste klassifitseerimise testimine oli oluline, et hinnata meetodite funktsionaalsust sarnaste järgnevuste tuvastamisel, sest noodistuste töötluste puhul on fikseeritud, et tegemist on samade teostega. See loob eelduse, et tõenäoliselt peavad need töötlused sisaldama omavahel sarnaseid noodijärgnevusi.

Noodistustest sarnaste järgnevuste leidmisel märgati tendentsi, et arvestades mitterõhulisi noote ja mitte arvestades noodipikkuseid võib olla suurem tõenäosus leida rohkem sarnaseid noodijärgnevusi. Selles ei saa täielikult kindel olla, sest kõik oleneb noodistustest endast, millised elemendid ja kuidas on need noodikirjas kujutatud. Seetõttu ei saa määrata, kas üks meetod töötab paremini või halvemini, aga kasutaja saab arvestada tulemusi auditiivsel järgnevuste hindamisel. Kasutaja saab lahendust kasutada muusikaandmete analüüsi abivahendina, et näiteks tuvastada plagiaati ning uurida, kuidas mõjutavad erinevad parameetrid järgnevuste kõla. Programm annab tulemuse sarnaste noodijärgnevuste kohta, kuid viimase kinnituse sarnasuse olemasolule määrab kuulaja. Testimise tulemus näitab, et rakendus võimaldab kasutajal leida sarnaseid järgnevusi ning hinnata rakendatud meetodite mõju järgnevuste leidmisele.

3.6 Edasiarendusvõimalused

Magistritöö raames arendatud lahendust on võimalik mitmel viisil edasi arendada. Järgnevalt on kirjeldatud ideid, mis muudaksid lahenduse terviklikumaks ja funktsionaalsemaks.

Lahenduse arendamisel pandi rõhku funktsionaalsusele, kuid veel võiks muuta disaini kasutajamugavaks ning pöörata tähelepanu IT arhitektuurile. Näiteks võiks lahenduse üle viia veebirakenduse kujule, mis on liidestatud (seotud) andmebaasiga noodistuste võrdlemiseks. Samuti võiks arendada kasutajaliidese, mille abil saab päringunoodistusi sisestada ning kuvada interaktiivselt tulemusi ja graafikuid ilma, et programmikood oleks kasutajale nähtav.

Praegusele lahendusele võiks arendada funktsiooni, mis enne noodistuste töötlust standardiseerib andmestikku sisestanud noodistused ühtsele kujule. Nii saaks näiteks noodistuste puhul, millel puuduvad taktijooned, need lisada või eemaldada nooditöötluse algoritmile tundmatud sümbolid. Samuti võiks kasutaja saada märgistada või grupeerida sisestatud noodistused kas näiteks helilooja, ajastu või noodistuse partii instrumendi järgi, et noodijärgnevuste leidmisel võrrelda sarnaste tunnustega teoseid.

Noodistuse elementide töötluse ja järgnevuste leidmisel võiks pöörata tähelepanu ka programmikoodi töö efektiivsusele. Koodi optimeerimine aitaks vähendada mälukasutust ning tõsta funktsioonide töökiirust.

Sarnaste noodijärgnevuste tuvastamisel võiks olla võimalik noodistuste partiid omavahel ühendada, et analüüsida mitmehäälseid noodistusi üle kõikide partiide. Sel juhul leitakse sarnased noodijärgnevused isegi siis, kui need on kirjutatud üle erinevate partiide (vt ptk 3.4, joonis 13).

Kasutajal võiks olla võimalus määrata ise noodistuse helistik. Praeguse helistiku määramise algoritmi vale tulemuse korral saaks kasutaja kahtluse korral selle manuaalselt ümber kirjutada.

Noodijärgnevuste visualiseerimisel võiks olla võrreldavate noodistuste noodikirjas kuvatud sarnase järgnevuse täpne asukoht. Selleks tuleks arendada lisafunktsioon mõnele olemasolevale noodikirja kuvavale Pythoni teegile või programmiliidesele.

Kokkuvõte

Käesoleva magistritöö eesmärk oli luua lahendus diginoodistuste sarnaste järgnevuste tuvastamiseks ja visualiseerimiseks. Lahendus võimaldab kasutajal leida sisestatud noodistuste vahel pikimaid sarnaseid mittekattuvaid noodijärgnevusi ja neid visualiseerida ning kuulata.

Muusika andmetüüpe ja nende uurimisvaldkondi on mitmeid, millest igaüks täidab kindlat eesmärki. Noodijärgnevuste sarnasuse mõõtmine on üks neist, mille eesmärk on tuvastada sarnaste omadustega noodistusi. Noodijärgnevuste sarnasuste leidmise meetodeid on erinevaid. Näiteks on üks võimalus hinnata noodistuste sarnasust teisenduskauguse meetodite abil. Mustri sobitamine on teine variant noodistuste vaheliste sarnaste järgnevuste tuvastamiseks. Sellekohaseid lahendusi ja rakendusi on loodud, kuid enamik neist võimaldavad kasutajal ette anda sümbolite mustri, mis leiab päringu põhjal noodistuste seast sarnaseid järgnevusi. Küll aga puudus lahendus, mis võrdleks omavahel noodistuste partiide järgnevusi.

Magistritöö raames valmis noodistuste andmete töötlemise ja sarnaste järgnevuste tuvastamise rakendus, mis võimaldab noodistuste partiide vahel otsida välja sarnaseid noodijärgnevusi. Töös kirjeldati, milliseid tehnoloogilisi vahendeid kasutati, kuidas töödeldakse noodistuse andmeid ning milliseid meetodeid on võimalik rakendada järgnevuste otsimisel. Kasutaja saab valida 32 erineva kombinatsiooni vahel, et uurida, kuidas mõjutavad need noodijärgnevuste tulemusi. Visuaalseks ja auditiivseks hindamiseks kuvatakse kasutajale leitud sarnased järgnevused, mille põhjal saab otsustada sarnasuse olemasolu üle. Lahenduse funktsionaalsust testiti nii töö autori loodud noodistustega kui ka erinevate noodistuste töötlustega. Testimise tulemused tõestasid, et lahenduse meetodid sarnasuste noodijärgnevuste leidmiseks säilitasid sama teose töötluste puhul klassifitseerimise paikapidavuse. Samuti näitasid tulemused, et meetodite mõju noodistuste järgnevuse leidmisele on kasutajal võimalik hinnata. Põhilisteks kitsaskohtadeks oli helistiku määramise algoritmi mõningane ebatäpsus ning MIDI-failide genereerimise vead. Lahendusele edasi arendamiseks pakuti välja erinevaid võimalusi, et täiendada selle kasutajamugavust, töökindlust ja funktsionaalsust.

Viidatud kirjandus

[1] Schedl M., Gómez E., Urbano J. Music Information Retrieval: Recent Developments and Applications.*Foundations and Trends® in Information Retrieval*. 2014. https://www.researchgate.net/publication/270818593_Music_Information_Retrieval_Recent_Developments_and_Applications (08.05.2022)

[2] ISMIR. https://www.ismir.net (29.03.2022).

[3] Müller M. Fundamentals of music processing. 2015. https://www.researchgate.net/publication/288774112_Fundamentals_of_Music_Processing (08.05.2022)

[4] Skinner J. Music Metadata How To: What Is Metadata?. *Music Gateway*. 2019. https://www.musicgateway.com/blog/how-to/music-metadata-for-sync-the-music-gateway-bible (30. 03.2022).

[5] Weninger F., Schuller B., Liem C. C. S., Kurth F., Hanjalic A. Music Information Retrieval: An Inspirational Guide to Transfer from Related Disciplines**.** 2012. https://www.researchgate.net/publication/224929609_Music_Information_Retrieval_An_Inspirational_Guide_to_Transfer_from_Related_Disciplines (08.05.2022)

[6] Pastukhov D. How Broken Music Metadata Affects the Music Industry, And What We Can Do About It?. *Soundcharts*. 2019. https://soundcharts.com/blog/music-metadata (30.03.2022)

[7] Musimap. https://www.musimap.com (vaadatud 30.03.2022).

[8] Hardjono T. Towards an Open and Scalable Music Metadata Layer. 2019. https://arxiv.org/pdf/1911.08278.pdf (08.05.2022)

[9] Ibrahim K. M., Royo-Letelier J., Epure E. V., Peeters G., Richard G. Audio-Based Auto-Tagging With Contextual Tags for Music. 2020. https://www.researchgate.net/publication/341084760_Audio-Based_Auto-Tagging_With_Contextual_Tags_for_Music (08.05.2022)

[10] Tempel E. Füüsika 8.klassile. Heli tekkimine ja levimine. https://opik.fyysika.ee/index.php/book/section/9295#/section/9295 (13.05.2022)

[11] Sõnaveeb. https://sonaveeb.ee/ (29.03.2022)

[12] Orio N. Music Retrieval: A Tutorial and Review. 2006. https://www.cs.cmu.edu/~pmuthuku/mlsp_page/lectures/Music_retr_tutorial.pdf (08.05.2022)

[13] He T., Liu W., Gong C. Yan J., Zhang N. Music Plagiarism Detection via Bipartite Graph Matching. 2021. http://arxiv.org/abs/2107.09889 (04.04.2022)

[14] Cano P., Batle E., Kalker T., Haitsma J. A review of algorithms for audio fingerprinting, 2003. https://www.researchgate.net/publication/2559014_A_Review_of_Algorithms_for_Audio_Fingerprinting (08.05.2022)

[15] Son H.-S, Byun S.-W., Lee S.-P. A Robust Audio Fingerprinting Using a New Hashing Method**.** 2020.https://www.researchgate.net/publication/346894474_A_Robust_Audio_Fingerprinting_Using_a_New_Hashing_Method (08.05.2022)

[16] Carnovalini F.,Rodà A. Computational Creativity and Music Generation Systems: An Introduction to the State of the Art**.** 2020. https://www.researchgate.net/publication/340411945_Computational_Creativity_and_Music_Generation_Systems_An_Introduction_to_the_State_of_the_Art (08.05.2022)

[17] MIDI Association. https://midi.org/specifications (05.04.2022)

[18] What is MIDI? Our Ultimate Guide To Music Software's Most Powerful Tool. *Music Repo*. 2014. https://www.musicrepo.com/what-is-midi/ (05.04.2022)

[19] Veltkamp R. C., Wiering F., Typke R. Content Based Music Retrieval. *Encyclopedia of Multimedia*. 2006. https://link.springer.com/referenceworkentry/10.1007/0-387-30038-4_32 (08.05.2022)

[20] Good M. MusicXML for Notation and Analysis | Songs and Schemas. 2019. http://michaelgood.info/publications/music/musicxml-for-notation-and-analysis/ (05.04.2022)

[21] Leis L. Diginootide meloodiaanalüsaator. TÜ arvutiteaduse instituudi bakalaureusetöö. 2020. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=69802&year=2020.

[22] Gurjar K., Moon Y.-S. A Comparative Analysis of Music Similarity Measures in Music Information Retrieval Systems**.** 2018.

https://www.researchgate.net/publication/323704849_A_comparative_analysis_of_music_similarity_measures_in_music_information_retrieval_systems (08.05.2022)

[23] Calvo-Zaragoza J., Hajič Jr J., Pacha A. Understanding Optical Music Recognition. 2020. https://www.researchgate.net/publication/342762210_Understanding_Optical_Music_Recognition (08.05.2022)

[24] MIREX 2021. https://www.music-ir.org/mirex/wiki/2021:Main_Page (31.03.2022)

[25] ISMIR - Datasets. https://www.ismir.net/resources/datasets/ (11.04.2022)

[26] Gilleland M. Levenshtein Distance, in Three Flavours. 2006. http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Fall2006/Assignments/editdistance/Levenshtein%20Distance.htm (08.05.2022)

[27] Mongeau M., Sankoff D. Comparison of musical sequences. 1990. https://link.springer.com/article/10.1007/BF00117340 (08.05.2022)

[28] Kelly M. B. Evaluation of Melody Similarity Measures. 2012. https://qspace.library.queensu.ca/bitstream/handle/1974/7442/Kelly_Matthew_B_201208_MSC.pdf;sequence=1 (08.05.2022)

[29] Lemström K., Mikkilä N., Mäkinen V., Ukkonen E. String Matching and Geometric Algorithm for Melodic Similarity. 2005. https://www.researchgate.net/publication/228918817_String_Matching_and_Geometric_Algorithm_for_Melodic_Similarity (08.05.2022)

[30] Mitchell N. Music similarity metrics: recognizing tempo, transposition, ornamentation, and accentuation properties. 2007.https://www.collectionscanada.gc.ca/obj/thesescanada/vol2/002/MR26511.PDF?oclc_number=458721495 (08.05.2022)

[31] Müllensiefen D., Frieler K. 8 Cognitive Adequacy in the Measurement of Melodic Similarity: Algorithmic vs. Human Judgments. 2003. https://www.researchgate.net/publication/237345998_8_Cognitive_Adequacy_in_the_Measurement_of_Melodic_Similarity_Algorithmic_vs_Human_Judgments (08.05.2022)

[32] Lin H.-J., Wu H.-H., Wang C.-W. Music Matching Based on Rough Longest Common Subsequence. 2011. https://www.researchgate.net/publication/220587662_Music_Matching_Based_on_Rough_Longest_Common_Subsequence (08.05.2022)

[33] 2019:Symbolic Melodic Similarity - MIREX Wiki. https://www.music-ir.org/mirex/wiki/2019:Symbolic_Melodic_Similarity (05.04.2022)

[34] Velardo V., Vallati M., Jan S. Symbolic Melodic Similarity: State of the Art and Future Challenges. 2016.https://www.researchgate.net/publication/303556143_Symbolic_Melodic_Similarity_State_of_the_Art_and_Future_Challenges (08.05.2022)

[35] Muusikateooria õpik - I.6 Intervall. http://mt.ema.edu.ee/muusika-elementaarteooria/i-6-intervall/ (22.04.2022)

[36] Bellanti B. Pythagoras: Discovering and Visualizing Musical Relationships Using Computer Analysis. 2021. https://journal.code4lib.org/articles/15949 (22.04.2022)

[37] Kranenburg P. V. A computational approach to content-based retrieval of folk song melodies. 2010.https://www.academia.edu/es/3583693/A_computational_approach_to_content_based_retrieval_of_folk_song_melodies (08.05.2022)

[38] Dutch Song Database. http://www.liederenbank.nl/searchmusic/piano.php?&lan=en (04.04.2022)

[39] Project Jupyter. https://jupyter.org (07.04.2022)

[40] What is music21? https://web.mit.edu/music21/doc/about/what.html (12.04.2022)

[41] Pandas - Python Data Analysis Library. https://pandas.pydata.org/ (07.04.2022)

[42] MuseScore. https://musescore.org/en (12.04.2022)

[43] Music21 Documentation - KrumhanslSchmuckler. https://web.mit.edu/music21/doc/moduleReference/moduleAnalysisDiscrete.html#music21.analysis.discrete.KrumhanslSchmuckler (07.04.2022)

[44] Humdrum Extras. http://extras.humdrum.org/man/keycor/ (07.04.2022)

[45] Inspired Acoustics -MIDI note numbers and center frequencies. https://www.inspiredacoustics.com/en/MIDI_note_numbers_and_center_frequencies (07.04.2022)

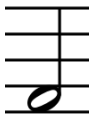
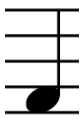
[46] difflib — Helpers for computing deltas. https://docs.python.org/3/library/difflib.html (12.04.2022)

Lisad

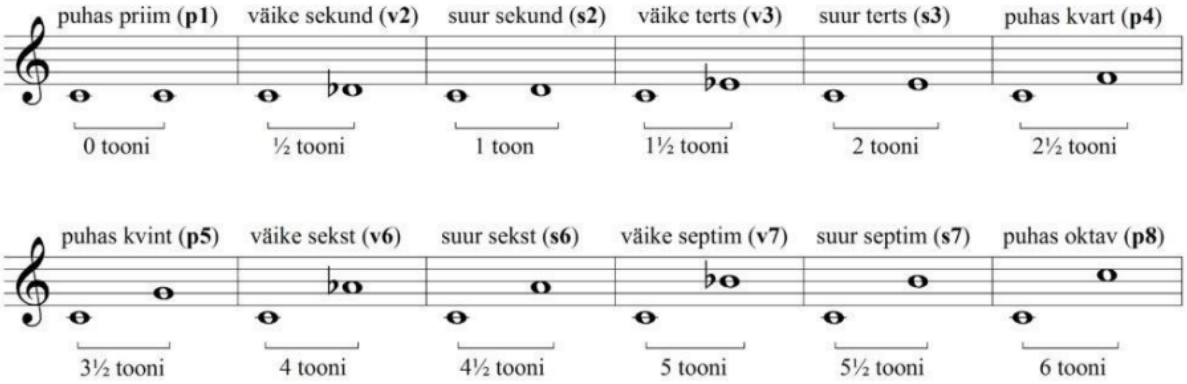
I MIREX 2021 võistluse teemad

Teema (inglise keeles)
Audio Beat Tracking
Audio Chord Estimation
Audio Cover Song Identification
Audio Downbeat Estimation
Audio Fingerprinting
Audio Key Detection
Audio Melody Extraction
Audio Onset Detection
Audio Tag Classification
Audio Tempo Estimation
Automatic Lyrics Transcription
Drum Transcription
Multiple Fundamental Frequency Estimation & Tracking
Music Detection
Patterns for Prediction
Query by Singing/Humming
Query by Tapping
Real-time Audio to Score Alignment (a.k.a Score Following)
Set List Identification
Structural Segmentation

II Noodi- ja pausipikkuste tabel

Noodipikkuse noodikirjamärk	Noodipikkuse nimetus	Pausipikkuse noodikirjamärk	Pausipikkuse nimetus	Pikkus löökides
	Tervenoot		Tervepaus	4.0
	Poolnoot		Poolpaus	2.0
	Veerandnoot		Veerandpaus	1.0
	Kaheksandiknoot		Kaheksandikpaus	0.5
	Kuueteistkümnendiknoot		Kuueteistkümnendikpaus	0.25
	Kolmekümnekahendiknoot		Kolmekümnekahendikpaus	0.125
	Kuuekümneneljandiknoot		Kuuekümneneljandikpaus	0.0625
	Ornament (eel- ja järellöök)	-	-	0

III Intervallide nimetused ja nende toonilised suurused



IV Lahenduse funktsionaalsuse testi tulemused

	note_part_ids	LCS	length_LCS	LCCNOS	size_LCCNOS	method_id	remove_rests	remove_chords	consider_note_duration	consider_offset_notes	consider_repetition
0	1_1 + 1_1	[(0, 0, ('60', '64', '72', '67', '69', '65', '64', '62', '64', '65', '64', '60'), ('60', '64', '72', '67', '69', '65', '64', '62', '64', '65', '64', '60'), [0, 1, 2, 3, 4, 6, 7, 8, 10, 12, 14, 16], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])]	12	0	0	TTFFT	TRUE	TRUE	FALSE	FALSE	TRUE
1	1_1 + 1_1	[(0, 0, ('60', '64', '72', '67', '69', '65', '64', '62', '64', '65', '64', '60'), ('60', '64', '72', '67', '69', '65', '64', '62', '64', '65', '64', '60'), [0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11])]	12	0	0	TTFFF	TRUE	TRUE	FALSE	FALSE	FALSE
2	1_1 + 1_1	[(0, 0, ('60', '64', '72', '67', '69', '65', '64', '62'), ('60', '64', '72', '67', '69', '65', '64', '62'), [0, 1, 2, 3, 4, 6, 7, 8], [0, 1, 2, 3, 4, 5, 6, 7])]	8	0	0	TTFTT	TRUE	TRUE	FALSE	TRUE	TRUE
3	1_1 + 1_1	[(0, 0, ('60', '64', '72', '67', '69', '65', '64', '62'), ('60', '64', '72', '67', '69', '65', '64', '62'), [0, 1, 2, 3, 4, 6, 7, 8], [0, 1, 2, 3, 4, 5, 6, 7])]	8	0	0	TTFTF	TRUE	TRUE	FALSE	TRUE	FALSE
4	1_1 + 1_1	[(5, 5, ('65', '64', '62', '64', '65', '64', '60'), ('65', '64', '62', '64', '65', '64', '60'), [6, 7, 8, 10, 12, 14, 16], [5, 6, 7, 8, 9, 10, 11])]	7	0	0	TFFFT	TRUE	FALSE	FALSE	FALSE	TRUE
5	1_1 + 1_1	[(5, 5, ('65', '64', '62', '64', '65', '64', '60'), ('65', '64', '62', '64', '65', '64', '60'), [6, 7, 8, 10, 12, 14, 16], [5, 6, 7, 8, 9, 10, 11])]	7	0	0	TFFFF	TRUE	FALSE	FALSE	FALSE	FALSE
6	1_1 + 1_1	[(0, 0, ('60', '64', '72', '67', '69'), ('60', '64', '72', '67', '69'), [0, 1, 2, 3, 4], [0, 1, 2, 3, 4])]	5	[(6, 5, ('65', '64', '62', '64', '65'), ('65', '64', '62', '64', '65'), [6, 7, 8, 10, 12], [5, 6, 7, 8, 9])]	1	FTFFT	FALSE	TRUE	FALSE	FALSE	TRUE
7	1_1 + 1_1	[(0, 0, ('60', '64', '72', '67', '69'), ('60', '64', '72', '67', '69'), [0, 1, 2, 3, 4], [0, 1, 2, 3, 4])]	5	[(6, 5, ('65', '64', '62', '64', '65'), ('65', '64', '62', '64', '65'), [6, 7, 8, 10, 12], [5, 6, 7, 8, 9])]	1	FTFFF	FALSE	TRUE	FALSE	FALSE	FALSE
8	1_1 + 1_1	[(0, 0, ('60', '64', '72', '67', '69'), ('60', '64', '72', '67', '69'), [0, 1, 2, 3, 4], [0, 1, 2, 3, 4])]	5	0	0	FTFTT	FALSE	TRUE	FALSE	TRUE	TRUE
9	1_1 + 1_1	[(0, 0, ('60', '64', '72', '67', '69'), ('60', '64', '72', '67', '69'), [0, 1, 2, 3, 4], [0, 1, 2, 3, 4])]	5	0	0	FTFTF	FALSE	TRUE	FALSE	TRUE	FALSE
10	1_1 + 1_1	[(6, 5, ('65', '64', '62', '64', '65'), ('65', '64', '62', '64', '65'), [6, 7, 8, 10, 12], [5, 6, 7, 8, 9])]	5	0	0	FFFFT	FALSE	FALSE	FALSE	FALSE	TRUE
11	1_1 + 1_1	[(6, 5, ('65', '64', '62', '64', '65'), ('65', '64', '62', '64', '65'), [6, 7, 8, 10, 12], [5, 6, 7, 8, 9])]	5	0	0	FFFFF	FALSE	FALSE	FALSE	FALSE	FALSE
12	1_1 + 1_1	[(0, 0, ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), [0, 1, 2, 3], [0, 1, 2, 3])]	4	0	0	TTTTT	TRUE	TRUE	TRUE	TRUE	TRUE
13	1_1 + 1_1	[(0, 0, ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), [0, 1, 2, 3], [0, 1, 2, 3])]	4	0	0	TTTTF	TRUE	TRUE	TRUE	TRUE	FALSE
14	1_1 + 1_1	[(0, 0, ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), [0, 1, 2, 3], [0, 1, 2, 3])]	4	0	0	TTTFT	TRUE	TRUE	TRUE	FALSE	TRUE
15	1_1 + 1_1	[(0, 0, ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), [0, 1, 2, 3], [0, 1, 2, 3])]	4	0	0	TTTFF	TRUE	TRUE	TRUE	FALSE	FALSE
16	1_1 + 1_1	[(0, 0, ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), [0, 1, 2, 3], [0, 1, 2, 3])]	4	0	0	TTTTT	FALSE	TRUE	TRUE	TRUE	TRUE
17	1_1 + 1_1	[(0, 0, ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), [0, 1, 2, 3], [0, 1, 2, 3])]	4	0	0	FTTTF	FALSE	TRUE	TRUE	TRUE	FALSE
18	1_1 + 1_1	[(0, 0, ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), [0, 1, 2, 3], [0, 1, 2, 3])]	4	0	0	FTTFT	FALSE	TRUE	TRUE	FALSE	TRUE
19	1_1 + 1_1	[(0, 0, ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), ('60 1.0', '64 1.0', '72 1.0', '67 1.0'), [0, 1, 2, 3], [0, 1, 2, 3])]	4	0	0	FTTFF	FALSE	TRUE	TRUE	FALSE	FALSE
20	1_1 + 1_1		0	0	0	TFFTT	TRUE	FALSE	FALSE	TRUE	TRUE
21	1_1 + 1_1		0	0	0	TFFTF	TRUE	FALSE	FALSE	TRUE	FALSE
22	1_1 + 1_1		0	0	0	TFTTT	TRUE	FALSE	TRUE	TRUE	TRUE
23	1_1 + 1_1		0	0	0	TFTTF	TRUE	FALSE	TRUE	TRUE	FALSE
24	1_1 + 1_1		0	0	0	TFTFT	TRUE	FALSE	TRUE	FALSE	TRUE
25	1_1 + 1_1		0	0	0	TFTFF	TRUE	FALSE	TRUE	FALSE	FALSE
26	1_1 + 1_1		0	0	0	FFFTT	FALSE	FALSE	FALSE	TRUE	TRUE
27	1_1 + 1_1		0	0	0	FFFTF	FALSE	FALSE	FALSE	TRUE	FALSE
28	1_1 + 1_1		0	0	0	FFTTT	FALSE	FALSE	TRUE	TRUE	TRUE
29	1_1 + 1_1		0	0	0	FFTTF	FALSE	FALSE	TRUE	TRUE	FALSE
30	1_1 + 1_1		0	0	0	FFTFT	FALSE	FALSE	TRUE	FALSE	TRUE
31	1_1 + 1_1		0	0	0	FFTFF	FALSE	FALSE	TRUE	FALSE	FALSE

V Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Ludvig Leis,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

Diginoodistuste sarnaste järgnevuste tuvastamine ja visualiseerimine,

mille juhendaja on Sven Aller,

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, alates **17.05.2022** kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Ludvig Leis
17.05.2022