Tampereen yliopisto

Mikko Haavisto

KANSANMUSIIKIN TIETOKONEAVUSTEINEN SÄVELTÄMINEN

Markovin ketjujen avulla

Kandidaatintyö
Informaatioteknologian ja viestinnän tiedekunta
Tarkastajat: Prof. Tapio Elomaa
Heinäkuu 2022

TIIVISTELMÄ

Mikko Haavisto: Kansanmusiikin tietokoneavusteinen säveltäminen
Kandidaatintyö
Tampereen yliopisto
Tietotekniikan tutkinto-ohjelma
Heinäkuu 2022

Tietokoneavusteinen säveltäminen on musiikin ja ohjelmoinnin yhdistävä aihe. Markovin ketjut ovat eräs tietokoneavusteisen säveltämisen menetelmä. Markovin ketjujen avulla säveltäessä on niille muodostettava malli joko käsin tai lähdeaineiston avulla. Työn tarkoituksena on esitellä Markovin ketjuja tietokoneavusteisen säveltämisen keinona ja selvittää, miten hyvin Markovin ketjut soveltuvat suomalaisen kansanmusiikin säveltämiseen.

Työssä esitellään Markovin ketjuja ja sitä, miten tietokone voidaan laittaa säveltämään niiden avulla. Markovin ketjuja ja niiden käyttömahdollisuuksia tietokoneavusteisessa säveltämisessä esitellään ensin teoriassa. Työn lopussa on myös yksinkertainen käytännön esimerkki esiteltyjen menetelmien soveltamisesta kansanmusiikin säveltämiseen ohjelmakoodin avulla.

Tietyt kansanmusiikin ominaispiirteet kompensoivat säveltämiseen liittyviä Markovin ketjujen puutteita. Siitä huolimatta työtä varten luotu esimerkkiohjelma vaatii vielä jatkokehitystä, jotta sillä luodut kappaleet olisivat mielekkäitä kokonaisuuksia. Esimerkiksi kappaleiden rytmiin tai sointuihin koodi ei nykyisellään ota kantaa. Lyhyiden musiikillisten ideoiden luomiseen koodi soveltuu kuitenkin jo sellaisenaan.

Avainsanat: tietokoneavusteinen säveltäminen, Markovin ketjut, musiikki, kansanmusiikki

ALKUSANAT

Kiitos professori Tapio Elomaalle kandidaatintyöni ohjaamisesta. Kiitos vaimolleni Viljalle, joka kärsivällisesti jaksoi lukea keskeneräistä tekstiäni. Kiitos isälleni ja Yläneen Pelimanneille kansanmusiikin tuomisesta lähelle sydäntä. Kiitos vielä Huittisten musiikkiopistolle soittoharrastuksen mahdollistamisesta.

Tämä työ oli mielenkiintoinen mahdollisuus päästä tutkimaan ohjelmoinnin ja musiikin rajapintaa. Toivon, että sen myötä on mahdollista laajentaa aiheeseen liittyvää tutkimusta ja ehkä joskus luoda tietokoneen avulla musiikkia, joka koskettaa kuulijaa. Tämä on pieni sävelaskel tietokoneavusteisen säveltämisen tutkimukselle, mutta suuri askel minulle.

Tampereella, 25. heinäkuuta 2022

Mikko Haavisto

SISÄLLYSLUETTELO

1. JOHDANTO

Tietokoneavusteinen säveltäminen sivuaa tieteen ja taiteen rajapintaa. Ihmisiä on pitkään mietityttänyt kysymys siitä, onko taide ihmiselle ainutlaatuinen ominaisuus, vai onko esimerkiksi tietokoneen mahdollista tuottaa taidetta.

Markovin ketjujen avulla voidaan imitoida ihmissäveltäjien sävellystyylejä [4, s. 71]. Vaikka Markovin ketjuilla luodut sävellykset on koottu puhtaasti lähdeaineiston osista, eroavat ne kuitenkin käytännössä aina alkuperäisistä sävellyksistä ja ovat siinä mielessä uusia sävellyksiä.

Tässä tutkielmassa esitellään Markovin ketjuja, tietokoneavusteista säveltämistä yleisesti sekä tietokoneavusteisen säveltämisen toteuttamista Markovin ketjujen avulla. Työhön sisältyy tietokoneohjelma, jolla sovelletaan Markovin ketjuja kansanmusiikin säveltämiseen.

Työn rakenne etenee luvussa 2 tehdyn tutkimusmenetelmän kuvailun jälkeen siten, että luvussa 3 kerrotaan Markovin ketjuista yleisesti, minkä jälkeen luvussa 4 käsitellään niiden käyttöä tietokoneavusteisessa säveltämisessä. Lopuksi esitellään kansanmusiikin ominaisuuksien yhteensopivuutta Markovin ketjujen kanssa ja demonstroidaan menetelmää käytännössä tietokoneohjelman avulla luvussa 5. Asiat esitellään lukijalle sellaisessa järjestyksessä, että kukin luku liittyy edelliseen lukuun, mutta tuo samalla jotakin lisää.

2. TUTKIMUSMENETELMÄ

Tutkielma on pääosin kirjallisuuskatsaus. Työn loppuvaiheilla on myös pieni käytännön koe, koska Markovin ketjujen soveltamisesta juuri suomalaisen kansanmusiikin säveltämiseen ei löytynyt akateemisia lähteitä. Kiinalaisen kansanmusiikin säveltämiseen liittyen löytyi vertaisarvioitu konferenssijulkaisu [10]. Suomalainen kansanmusiikki eroaa kuitenkin kiinalaisesta sen verran, ettei julkaisun pohjalta voida tehdä johtopäätöksiä Markovin ketjujen soveltuvuudesta suomalaisen kansanmusiikin säveltämiseen.

Työn aiheen rajaus tarkentui lähteitä hakiessa. Ensin aihe oli tietokoneavusteinen säveltäminen yleisesti, mutta se oli sellaisenaan liian laaja kandidaatintyölle, joten lisärajaukset olivat tarpeen. Aikaperusteisen rajauksen harkinnan jälkeen työn aihe päädyttiin rajaamaan käsittelemään tietokoneavusteista säveltämistä erityisesti Markovin ketjujen avulla. Lopuksi tarkentavaksi rajaukseksi lisättiin kansanmusiikin säveltäminen. Lyhyesti sanottuna tietokoneavusteisen säveltämisen tietokoneavusteisuus rajattiin Markovin ketjujen käytöksi ja säveltäminen kansanmusiikin säveltämiseksi.

Hakuja tehtiin lähinnä Andor-tietokantaan. Tietokoneavusteisesta säveltämisestä löytyi runsaasti erilaisia artikkeleita, joista noin puolet olivat vertaisarvioituja. Markovin ketjujen lisääminen hakulausekkeeseen rajasi lähteiden määrää moninkertaisesti ja vertaisarvioitujen julkaisujen osuus kaikista hakutuloksista kasvoi. Kansanmusiikin lisääminen hakusanoihin rajasi vielä tuloksia entisestään, mutta moni hakutulos ei ollut enää relevantti.

Hakulausekkeet muodostettiin siten, että OR-operaattorin ja sulkeiden avulla haettiin samanaikaisesti suomeksi ja englanniksi ja AND-operaattorilla yhdistettiin eri hakusanat. Esimerkki eräästä käytetystä hakulausekkeesta on ohessa:

```
("tietokoneavusteinen säveltäminen" OR "algorithmic composition") AND Markov*
```

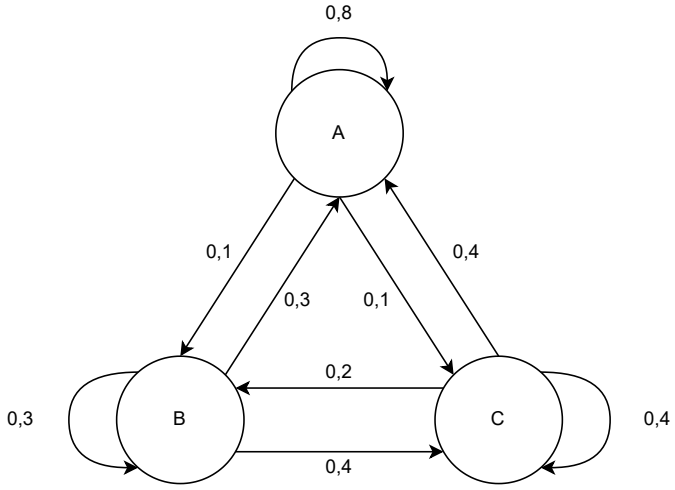
Luvussa 5 analysoidaan esimerkkiohjelman avulla Markovin ketjujen soveltuvuutta kansanmusiikin säveltämiseen. Tämä osa ei ole varsinaisesti perinteisen kirjallisuuskatsauksen mukainen, mutta se on valmiiden esimerkkien puutteen vuoksi tarpeellinen.

3. MARKOVIN KETJUT

Markovin ketju on stokastinen prosessi eli satunnaisesti etenevä prosessi, joka kuvaa siirtymien todennäköisyyksiä eri tilojen välillä. Seuraavaan tilaan siirtymisen todennäköisyys ei riipu aiemmista tiloista, vaan ainoastaan nykyisestä tilasta. Tätä ominaisuutta kutsutaan Markovin ominaisuudeksi (engl. Markov property). Kustakin tilasta lähtevien siirtymien todennäköisyyksien summa on 100%.

Tilan käsitteen käytännön merkitys riippuu käyttötarkoituksesta. Tämän työn yhteydessä tila voi tarkoittaa esimerkiksi yksittäisen nuotin sävelkorkeutta. Asiaa käsitellään tarkemmin seuraavissa luvuissa.

Tutkielmaa varten keksityssä esimerkissä A, B ja C ovat tiloja, joiden välisten siirtymien todennäköisyydet on esitetty kuvan 3.1 graafissa.



Kuva 3.1. *Esimerkkigraafi Markovin ketjun tiloista A, B ja C sekä niiden välisten siirtymien todennäköisyyksistä.*

Esimerkin tilojen välisten siirtymien välisistä todennäköisyyksistä voidaan muodostaa taulukko 3.1, jossa rivit kuvaavat eri lähtötiloja ja sarakkeet kohdetiloja. Esimerkiksi ensimmäisen rivin ja kolmannen sarakkeen leikkauspisteen arvo $0{,}1$ kuvaa todennäköisyyttä sille, että siirrytään tilasta A tilaan C.

Taulukko 3.1. *Esimerkin siirtymätodennäköisyydet taulukkomuodossa.*

Lähtötila	A	B	C
A	0,8	0,1	0,1
B	0,3	0,3	0,4
C	0,4	0,2	0,4

Taulukosta 3.1 voidaan edelleen muodostaa kaavan 3.1 mukainen siirtymätodennäköisyysmatriisi (engl. transition probability matrix) P ottamalla taulukon arvot ilman otsikkoriviä ja -saraketta. Matriisin rivit vastaavat luvussa 5 käytettäviä siirtymätodennäköisyysvektoreita.

$$P = \begin{bmatrix} 0{,}8 & 0{,}1 & 0{,}1 \\ 0{,}3 & 0{,}3 & 0{,}4 \\ 0{,}4 & 0{,}2 & 0{,}4 \end{bmatrix} \tag{3.1}$$

Markovin ketjun taso (engl. order) voi olla 1, kuten esimerkissä, mutta se voi olla myös korkeampi, jolloin seuraavan tilan valitseminen määrittyy useamman kuin yhden edellisen tilan perusteella. Jos ketju olisi esimerkiksi tasoa 3, valittaisiin seuraava tila aina kolmen edellisen peräkkäisen tilan perusteella. Korkeampi taso johtaa suurempaan samankaltaisuuteen lähdeaineiston kanssa [4, s. 4].

Eri tilojen välillä voidaan tehdä niin kutsuttu satunnaiskulku (engl. random walk), jossa valitaan ensin jokin lähtöpiste, josta lähtevien siirtymätodennäköisyyksien avulla valitaan painotetulla satunnaisvalinnalla seuraava tila. Valittu tila voi ketjun rakenteesta riippuen olla myös sama kuin lähtötila. Tämän jälkeen kulkua jatketaan siten, että uudesta tilasta lähtevien siirtymätodennäköisyyksien avulla valitaan taas seuraava tila samalla tavoin kuin lähtötilan kohdalla. Satunnaisvalinta suoritetaan mielivaltaisesti päätetty määrä, jonka jälkeen kulku loppuu.

Edellä esitetyn esimerkin tapauksessa voitaisiin esimerkiksi lähteä liikkeelle tilasta A, jolloin siirtymien todennäköisyydet olisivat $0{,}8$ tilaan A ja $0{,}1$ tiloihin B ja C. Näistä valitaan painotetulla satunnaisvalinnalla seuraava tila eli siirrytään 80% todennäköisyydellä tilaan A, 10% todennäköisyydellä tilaan B ja 10% todennäköisyydellä tilaan C. Uudesta tilasta lähtevien siirtymien todennäköisyyksien avulla valitaan jälleen seuraava tila.

4. TIETOKONEAVUSTEINEN SÄVELTÄMINEN

Tietokoneavusteinen säveltäminen on käsite joukolle tietokoneella suoritettavia ohjelmia ja menetelmiä, joilla pyritään helpottamaan tai automatisoimaan musiikin sävellysprosessia. Tässä työssä tietokoneavusteiseen säveltämiseen käytetään vain Markovin ketjuja, mutta tietokoneavusteiseen säveltämiseen voi käyttää myös muita koneoppimisen menetelmiä. Markovin ketjuja käytettiin ensimmäisen kerran musiikin yhteydessä vuoden 1950 vaiheilla [4, s. 71].

Tietokoneavusteinen säveltäminen sijaitsee musiikin ja tietotekniikan, rajapinnassa ja sen tutkiminen vaatii suhteellisen laajaa ymmärrystä molemmista aloista. Lähivuosilta löytyy algoritmiseen säveltämiseen liittyen kandidaatintöitä sekä musiikin [3] että tietotekniikan [7] opiskelijoilta.

Algoritminen säveltäminen on samankaltainen käsite kuin tietokoneavusteinen säveltäminen. Suurin ero on, ettei algoritmiseen säveltämiseen varsinaisesti vaadita tietokonetta, vaan esimerkiksi noppia käyttämällä on sävelletty algoritmisesti [4, s. 36] jo kauan ennen tietokoneen keksimistä. Tietokoneen käyttö nopeuttaa algoritmista säveltämistä huomattavasti, ja sen vuoksi algoritmista säveltämistä voidaan nykypäivän kontekstissa käyttää lähes synonyyminä tietokoneavusteiselle säveltämiselle.

Länsimaisessa musiikissa äänet jaetaan yleensä 12 sävelluokkaan (engl. pitch class), jotka toistuvat, kun siirrytään oktaavilla ylöspäin eli kun sävelen taajuus kaksinkertaistetaan. Tässä työssä kukin tila sisältää sävelluokan lisäksi tiedon siitä, mistä oktaavista on kyse. Markovin ketjun tiloina voidaan käyttää myös pelkkiä sävelluokkia, jolloin ei huomioida, mistä oktaavista on kyse. Tällöin oktaavi täytyy määrittää erikseen.

Tietokoneavusteisessa säveltämisessä Markovin ketjuja voidaan käyttää melodian lisäksi myös sointukiertojen [5] ja sanoitusten [1] luomiseen. Sanoitusten tapauksessa Markovin ketjut soveltuvat lähinnä parodiasanoitusten ja eri yhtyeiden ominaispiirteitä matkivien kappaleiden luomiseen, sillä vaikka ketjun luomat sanoitukset muistuttavat tyyliltään lähdeaineistoa, luotujen lauseiden sisältö on yleensä täyttä hölynpölyä.

Markovin ketjun todennäköisyydet voidaan luoda käsin tai lähdeaineiston pohjalta. Lähdeaineisto voi olla esimerkiksi lista jonkin yhtyeen kappaleiden melodioista tai sointukierroista. Sanoitusten tapauksessa lähdeaineistona toimisi esimerkiksi lista yhtyeen kappaleiden sanoituksista. Lähdeaineiston voi joutua muotoilemaan sellaiseen muotoon, että

käytettävät ohjelmat saavat määritettyä aineiston pohjalta halutun ketjun.

Tehdään lähempi katsaus melodioiden eli sävelkulkujen luomiseen Markovin ketjujen avulla. Jokaisesta lähdeaineiston sävelkorkeudesta tehdään oma tilansa. Lähdeaineiston pohjalta määritetään siirtymätodennäköisyydet tilojen välillä. Kukin lähdeaineiston siirtymä kahden tilan välillä eli sävelestä toiseen merkitään lähtösävelen siirtymätodennäköisyysvektoriin.

Kun siirtymätodennäköisyydet on määritetty, voidaan luodun Markovin ketjun avulla säveltää melodioita tekemällä satunnaiskulkuja ketjua pitkin. Lähtötila voidaan pyytää käyttäjältä, tietokone voi valita sen satunnaisesti tai se voidaan yrittää valita älykkäästi esimerkiksi sävellajia analysoimalla. Mikäli tiedetään, että ollaan jossakin tietyssä sävellajissa, on melodia järkevää aloittaa esimerkiksi sävellajin perussävelestä eli esimerkiksi G-duurissa lähdettäisiin G-sävelestä.

5. MARKOVIN KETJUJEN SOVELTAMINEN KANSANMUSIIKIN SÄVELTÄMISEEN

Tässä osiossa suoritetaan lyhyt kokeellinen tutkimus siitä, millä tavoin yksinkertainen Markovin ketjuilla toteutettu järjestelmä soveltuu kansanmusiikin säveltämiseen. Tutkimuksessa on käytetty lähdeaineistona yläneläistä kansanmusiikkia [6].

5.1 Kansanmusiikin ja Markovin ketjujen yhteensopivuus

Kansanmusiikilla on tiettyjä piirteitä, jotka vähentävät Markovin ketjujen heikkouksien vaikutusta lopputulokseen. Kansanmusiikki on luonteeltaan rentoa ja yksinkertaista ja sen merkityksellisyys tulee sen sosiaalisesta merkityksestä sekä soitinkokoonpanon ja esitystavan monipuolisesta vaihtelusta.

Markovin ketjut soveltuvat paremmin lyhyiden musiikillisten elementtien säveltämiseen [7]. Kansanmusiikkikappaleet voivat olla melko lyhyitä, jolloin ne soitetaan yleensä monta kertaa, mikä kompensoi myös Markovin ketjujen satunnaisuudesta johtuvaa pitemmän linjan musiikillisen rakenteen puutetta. Nämä ominaisuudet viittaavat siihen, että kansanmusiikin säveltäminen Markovin ketjuilla voisi onnistua suhteellisen hyvin tai ainakin paremmin kuin monen muun musiikkityylilajin.

5.2 Esimerkkiohjelma

Markovin ketjun avulla säveltämisen käytännön sovellus on Python 3 -ohjelmointikielellä ja `Music21`-kirjaston [2] avulla toteutettu ensimmäisen asteen Markovin ketju. Koodissa on käytetty myös Pythonin `random`-kirjastoa. Markovin ketjut ovat suhteellisen yksinkertaisia toteuttaa, mikä on yksi niiden eduista.

Ohjelmassa 5.1 luetaan MIDI-tiedostot Music21-kirjastoa käyttäen ja luodaan Markovin ketju niiden avulla. Ketjut luodaan käymällä lähdeaineiston tiedostojen nuotit läpi järjestyksessä. Edellisen nuotin sävelkorkeuden kohdalle ketjuun lisätään yksi siirtymä lisää nykyisen nuotin sävelkorkeutta vastaavaan kohtaan.

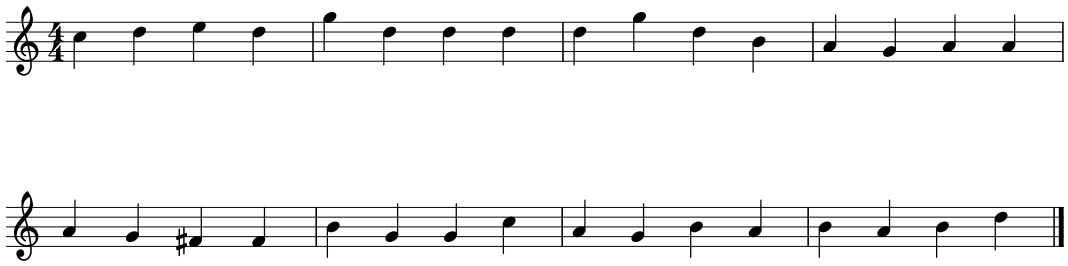
Kuvassa 5.1 on esitetty taulukkomuodossa ketju, joka on muodostettu edellä mainitun nuottikirjan kappaleista Mandoliinipolkka ja Tanhupolkka. Taulukko kuvaa siirtymätoden-

näköisyyksiä taulukon vasemmanpuoleisimman sarakkeen sävelistä ylärivin säveliin. Nuottien nimet on kirjoitettu käyttäen tieteellistä sävelkorkeuden kirjoitustapaa (Scientific Pitch Notation, SPN), jossa keski-C on C4. Kirjain kertoo nuotin sävelluokan ja numero oktaavin [9].

	G3	B3	C4	D4	E4	F#4	G4	G#4	A4	B4	C5	D5	E5	G5
G3	0	0,5	0	0	0	0	0,5	0	0	0	0	0	0	0
B3	0,33	0	0,17	0,5	0	0	0	0	0	0	0	0	0	0
C4	0	0,17	0,17	0	0,5	0,17	0	0	0	0	0	0	0	0
D4	0	0,12	0,25	0,25	0,06	0,31	0	0	0	0	0	0	0	0
E4	0	0	0	0,75	0,12	0,12	0	0	0	0	0	0	0	0
F#4	0	0	0	0	0,12	0,12	0,29	0	0,41	0,06	0	0	0	0
G4	0	0,06	0	0,03	0,03	0,09	0,42	0	0,09	0,15	0,06	0,06	0	0
G#4	0	0	0	0	0	0	0	0	1	0	0	0	0	0
A4	0	0	0	0,06	0	0,12	0,19	0,06	0,19	0,34	0,03	0	0	0
B4	0	0	0	0	0	0	0,26	0	0,29	0	0,39	0,06	0	0
C5	0	0	0	0	0	0,05	0	0	0,25	0,5	0	0,1	0,1	0
D5	0	0	0	0	0	0	0	0	0	0,15	0,11	0,48	0,19	0,07
E5	0	0	0	0	0	0	0	0	0	0	0,29	0,71	0	0
G5	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Kuva 5.1. *Taulukkoesitys esimerkkiohjelman Markovin ketjusta.*

Ketjun luonti on kuvattu ohjelmassa 5.2. Käytännössä ketju on toteutettu Python-ohjelmointikielen sanakirja-tietorakenteena (engl. dictionary), jossa avaimet ovat nykyisen äänen mahdolliset sävelkorkeudet ja kutakin avainta vastaavana arvona siirtymätodennäköisyysvektori äänestä muihin ääniin. Nämä siirtymätodennäköisyysvektorit yhdistämällä voidaan muodostaa siirtymätodennäköisyysmatriisi.



Kuva 5.2. *Esimerkki satunnaiskulusta.*

Valmiilla ketjuilla voidaan tehä halutun pituisia satunnaiskulkuja ohjelman 5.3 avulla. Ketju, satunnaiskulun pituus ja aloitusnuotti annetaan parametreina funktiolle `satunnaiskulku`. Kuvassa 5.2 on esimerkki satunnaiskulusta, kun ketju on muodostettu edellä mainitun nuottikirjan kappaleista Mandoliinipolkka ja Tanhupolkka. Kyseiset kappaleet on valittu sen vuoksi, että ne ovat molemmat samassa sävellajissa (G-duuri) ja samaa tanssilajia (polkkia).

Ohjelma kuvaa Markovin ketjujen käytön tietokoneavusteisessa säveltämisessä mahdollisimman yksinkertaisessa muodossa ja siihen voisi tehdä monenlaisia laajennuksia. Laajemman lähdeaineiston saamiseksi olisi hyvä toteuttaa ohjelma, joka transponoi kappaleet samaan sävellajiin, jolloin saataisiin suurempi määrä kunkin tanssilajin kappaleita analysoitavaksi niin, että ne on normalisoitu sävellajin suhteen. Nykyisellään ohjelma ei ota kantaa rytmiin lainkaan, mutta sekin on mahdollista toteuttaa Markovin ketjujen avulla, esimerkiksi luomalla Markovin ketju erilaisista kokonaisista tahdeista [4, s. 71], puolikkaista tahdeista tai neljäsosanuotin mittaisista rytmisistä rakenteista. Tauot voisi ottaa huomioon vaihtamalla tiedostomuodon MusicXML-muotoon tai muuhun vastaavaan, jossa taukojen käsittely on helpompaa. Melodiaa säestämään voisi luoda sointuja ja niistä koostuvia kadensseja. Lisäksi kappale yleensä aloitetaan ja lopetetaan ensimmäisen asteen sointuun, eli esimerkiksi G-duurissa G-duurisointuun.

Ohjelma 5.1. *MIDI-tiedostojen lukeminen.*

```
1  # Tuodaan ohjelmassa tarvittavat kirjastot.
2  import music21
3  import random
4
5  def lue_tiedostot(tiedostot):
6      """
7      tiedostot: lista tiedostojen sijainneista merkkijonoina
8      return: lista music21.stream.Stream-olioita
9      """
10     palautus = []
11     for tiedosto in tiedostot:
12         palautus.append(music21.converter.parse(tiedosto))
13     return palautus
```

Ohjelma 5.2. *Melodiaketjun luominen.*

```
1  def luo_melodiaketju(musiikkivirrat):
2    # Funktio luo melodioiden muodostamiseen tarvittavan
3    # Markovin ketjun music21.stream.Stream -musiikkivirroista.
4    ketju = {}
5    edellinen = None
6
7    for virta in musiikkivirrat:
8      # parametri noNone ei ota taukoja mukaan
9      for nuotti in virta.findConsecutiveNotes(noNone=true):
10       # Ensimmainen nuotti kasitellaan eri tavalla,
11       # koska silla ei ole edeltajaa.
12       if previous is None:
13         previous = nuotti
14         continue
15
16       # Otetaan indeksointia varten nuoteista nimi ja oktaavi
17       nuotti_merkkijono = nuotti.name + str(nuotti.octave)
18       edellinen_merkkijono = edellinen.name
19                     + str(edellinen.octave)
20
21       if edellinen_merkkijono not in ketju:
22         # Edellisesta nuotista ei ole viela tehty yhtaan
23         # siirtymaa, joten se taytyy lisata ketjuun.
24         ketju[edellinen_merkkijono] = {nuotti_merkkijono: 1}
25
26       elif nuotti_merkkijono not in ketju[edellinen_merkkijono]:
27         # Nykyiseen nuottiin ei ole viela ennen siirrytty
28         # edellisesta nuotista, joten sen arvoksi annetaan 1.
29         ketju[edellinen_merkkijono]
30             [nuotti_merkkijono] = 1
31
32       else:
33         # Nykyiseen nuottiin on siirrytty edellisesta
34         # nuotista ennenkin.
35         ketju[edellinen_merkkijono]
36             [nuotti_merkkijono] += 1
37
38       edellinen = nuotti # Laitetaan nuotti talteen muuttujaan
39                  # ennen siirtymista eteenpain
40
41   return ketju
```

Ohjelma 5.3. *Satunnaiskulku*

```
1  def satunnaiskulku(ketju, alku=None, pituus=10):
2    """
3    ketju: funktiolla luo_melodiaketju luotu Markovin ketju
4    alku: aloitusnuotti
5    pituus: satunnaiskulun pituus nuotteina
6    """
7    # Jos aloitusnuottia ei ole annettu,
8    # se arvotaan ketjun nuottien joukosta.
9    if alku is None:
10     alku = random.choice(list(ketju))
11   nyt = alku # Asetetaan alkutila nykyiseksi tilaksi (merkkijono).
12   # Luodaan 2 eri sailiota satunnaiskululle.
13   # music21-kirjaston Stream on helpompi kuunnella
14   # esimerkiksi MIDI-muodossa. Merkkijono on taas helpompi lukea
15   # suoraan ohjelman tulosteena.
16   kulku = music21.stream.Stream()
17   kulku_merkkijono = []
18
19   for _ in range(pituus):
20     nuotti = music21.note.Note(nyt)
21     kulku.append(nuotti)
22     kulku_merkkijono.append(nyt)
23     # Poimitaan ketjusta mahdolliset siirtymat ja niiden
24     # todennakoisyyksien painot ja arvotaan niiden avulla
25     # seuraava nuotti.
26     jatkovaihtoehdot = list(ketju[nyt])
27     jatkopainot = list(ketju[nyt].values())
28     nyt = random.choices(jatkovaihtoehdot, weights=jatkopainot)[0]
29   return kulku, kulku_merkkijono
```

6. YHTEENVETO

Markovin ketjut soveltuvat suhteellisen hyvin kansanmusiikin säveltämiseen. Moni kansanmusiikin piirre toimii vastapainona Markovin ketjujen puuttellisuuksille säveltämisen näkökulmasta. Markovin ketjujen yksinkertaisuus ei haittaa niin paljon, jos ketjuja käyttää lyhyiden musikaalisten ideoiden luomiseen ja jatkojalostaa parhaat ideat valmiiksi kappaleiksi. Tällöin Markovin ketjut eivät toimi itse säveltäjänä vaan säveltäjän työkaluna, joka auttaa tyhjän paperin kammon välttämisessä.

Kehitysmahdollisuuksia kuitenkin riittää. Tässä työssä tehty kokeellinen osuus on pyritty pitämään suhteellisen yksinkertaisena. Sen edistäminen esitettyjen kehitysehdotusten suhteen on mahdollinen jatkoselvityksen aihe. Eräs mahdollisuus on myös käyttää Markovin ketjuja muiden tietokoneavusteisen säveltämisen menetelmien, kuten neuroverkkojen, kanssa vielä parempien tulosten saamiseksi.

Siinä tapauksessa, että ohjelman saisi kehitettyä niin pitkälle, että sillä pystyisi luomaan kokonaisia kappaleita melodioineen, sointuineen ja mahdollisesti jopa sanoituksineen, voisi yrittää tehdä jotakin samantapaista kuin The Unreal Book [8], joka on jazz-nuottikirja, jonka kappaleet on tietokone muodostanut lähdeaineiston pohjalta.

LÄHTEET

[1] Gabriele Barbieri et al. ”Markov Constraints for Generating Lyrics with Style”. Teoksessa: vol. 242. Elokuu 2012. DOI: 10.3233/978-1-61499-098-7-115.

[2] Michael Cuthbert. *Music21*. URL: https://web.mit.edu/music21/doc/index.html.

[3] Sami Haukka. *Katsaus algoritmiseen säveltämiseen (David Copen johdolla)*. fin. 2020.

[4] Gerhard Nierhaus. *Algorithmic Composition: Paradigms of Automated Music Generation*. eng. 1. Aufl. Vienna: Springer Verlag, Wien, 2009. ISBN: 9783211755396.

[5] Shipra Shukla ja Haider Banka. ”An Automatic Chord Progression Generator Based On Reinforcement Learning”. Teoksessa: *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2018, s. 55–59. DOI: 10.1109/ICACCI.2018.8554901.

[6] Eino Sointula. *Pelimannin penkiltä*. Yläneen Pelimannit, 2017.

[7] Aleksi Tarvainen. ”Markovin ketjut ja Markovin piilomallit algoritmisessa säveltämisessä”. fin. 2020. URL: https://jyx.jyu.fi/handle/123456789/68953.

[8] Andrea Valle. ”THE UNREAL BOOK. ALGORITHMIC COMPOSITION for JAZZ LEAD SHEETS”. eng. Teoksessa: *Proceedings of the Sound and Music Computing Conferences*. Vol. 2021-. 2021, s. 308–315. ISBN: 9788894541540.

[9] Robert W. Young. ”Terminology for Logarithmic Frequency Units”. *The Journal of the Acoustical Society of America* 11.1 (1939), s. 134–139. DOI: 10.1121/1.1916017. eprint: https://doi.org/10.1121/1.1916017. URL: https://doi.org/10.1121/1.1916017.

[10] Xiaomei Zheng et al. ”An automatic composition model of Chinese folk music”. eng. Teoksessa: *AIP conference proceedings*. Vol. 1820. 1. 2017. ISBN: 9780735414884.