Datamusikalisaatio

Aurora Tulilaulu

Pro gradu -tutkielma
Helsingin yliopisto
Tietojenkäsittelytieteen laitos

Helsinki, 11. huhtikuuta 2017

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Aurora Tulilaulu

Työn nimi — Arbetets titel — Title

Datamusikalisaatio

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu -tutkielma	11. huhtikuuta 2017	68

Tiivistelmä — Referat — Abstract

Pro gradu -tutkielmassani esittelen datan perusteella ohjattavaa automaattista säveltämistä eli datamusikalisaatiota. Datamusikalisaatiossa on kyse datasta löytyvien muuttujien kuulumisesta automaattisesti sävelletyssä musiikissa. Tarkoitus olisi, että musiikki toimisi korville tarkoitetun visualisaation tavoin havainnollistamaan valittuja attribuutteja datasta.

Erittelen tutkielmassa erilaisia tapoja, miten sonifikaatiota ja automaattista tai koneavustettua säveltämistä on tehty aikaisemmin sekä millaisia sovelluksia niillä on. Käyn läpi yleisimmät käytetyt tavat generoida musiikkia, kuten tyypillisimmät stokastiset menetelmät, kieliopit ja koneoppimiseen perustuvat menetelmät. Kerron myös lyhyesti sonifikaatiosta eli datan suorasta kuvaamisesta äänisignaalina ilman musiikillista elementtiä. Kommentoin erilaisten menetelmien vahvuuksia ja heikkouksia. Käsittelen lyhyesti myös sitä, mihin asti automatisoidussa säveltämisessä ja sen uskottavuudessa ihmisarvioijien silmissä on pisimmillään päästy. Käytän esimerkkinä muutamia erilaisia tunnustusta saaneita säveltäviä ohjelmia.

Käsittelen kahta erilaista tekemääni musikalisaatio-ohjelmaa. Ensimmäinen generoi kappaleita tiivistäen käyttäjän yhdestä nukutusta yöstä kerätyn datan neljästä kahdeksaan minuuttia kestävään kappaleeseen. Toinen tekee musiikkia reaaliaikaisesti ja muutettavien parametrien pohjalta, jolloin sen pystyy kytkemään toiseen ohjelmaan, joka analysoi dataa ja muuttaa parametreja. Käsitellyssä esimerkissä musiikki tuotetaan keskustelulokin pohjalta ja esimerkiksi keskustelun sävy ja nopeus vaikuttavat musiikkiin.

Käyn läpi tekemieni ohjelmien periaatteet musiikin generoimiselle. Käsittelen myös tehtyjen päätösten syitä käyttäen musiikin teorian ja säveltämisen perusteita. Selitän, millaisilla periaatteilla käytetty data kuuluu tai voidaan saada kuulumaan musiikissa, eli miten musikalisaatio eroaa tavallisesta konesäveltämisestä ja sonifikaatiosta, sekä miten se asettuu näiden kahden jo olemassa olevan tutkimuskentän rajoille.

Lopuksi esittelen myös käyttäjäkokeiden tulokset, joissa käyttäjiä on pyydetty arvioimaan keskustelulokeista tehdyn musikalisaation toimivuutta, ja pohdin saatujen tulosten ja alan nykytilan pohjalta musikalisaation mahdollisia sovelluskohteita ja mahdollista tulevaa tutkimusta, jota aiheesta voisi tehdä.

Avainsanat — Nyckelord — Keywords
datamusikalisaatio, musikalisaatio, algoritminen säveltäminen, konesäveltäminen, laskennallinen luovuus

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto

Datamusikalisaatio tarkoittaa datan kuvaamista musiikilla. Tässä tutkielmassa keskitytään erityisesti musiikkimateriaaliin, joka on sävelletty kokonaan ilman ihmisen puuttumista prosessiin. Koska termillä ei kuitenkaan ole vakiintunutta merkitystä, samaa termiä voidaan käyttää myös prosessista, jossa ihminen säveltää jotain kuvaamaan dataa tai kone säveltää yhdessä ihmisen kanssa jotain kuvaamaan dataa.

On paljon erilaisia tapoja, joilla algoritmista säveltämistä voi tehdä, mutta kaikki niistä eivät sovellu datamusikalisaatiossa käytettäviksi yhtä hyvin. Jotta datassa olevat ominaisuudet saataisiin kuulumaan musiikissa, musiikin sävellysprosessiin pitää pystyä vaikuttamaan. Joissain prosesseissa vaikuttaminen tarkoittaa yksinkertaisia toimenpiteitä, kuten jonkin ohjaavan arvon muokkaamista datassa esiintyvien ilmiöiden mukaan, ja joissain se vaatii paljon monimutkaisempia ratkaisuja, kuten sävellyksen generoinnin pilkkomista lyhyisiin paloihin, joiden generointiin käytetään eri varianttia valitusta ratkaisusta, riippuen jonkin ohjaavan attribuutin arvosta.

Tutkielmassa esitellään ensin datamusikalisaatio käsitteenä, sen yhteydet laskennalliseen luovuuteen ja algoritmiseen säveltämiseen, ja verrataan sitä sonifikaatioon, joka on samankaltainen ala. Tämän jälkeen esitellään musiikkitermejä, jotta myöhempien lukujen ymmärtäminen olisi mahdollista myös musiikkitermistöä tuntemattomalle lukijalle. Sitten käydään läpi algoritmisen säveltämisen ja sonifikaation historiaa ja annetaan muutamia esimerkkejä viimeaikaisista menestyneistä säveltävistä ohjelmista. Historian jälkeen käsitellään erilaisia menetelmiä, joita on käytetty algoritmiseen säveltämiseen, kuten esimerkiksi neuroverkot, Markovin ketjut sekä geneettiset algoritmit. Neljännessä luvussa tutustutaan kahteen erilaiseen sovellukseen, jotka käyttävät datamusikalisaatiota tuottaakseen lähdedataa kuvaavaa musiikkia. Viimeisessä luvussa on yhteenveto ja pohdintaa rajoituksista, käyttötarkoituksista ja mahdollisista jatkotutkimuksista liittyen datamusikalisaatioon.

Tavoitteena on antaa lukijalle yleiskatsaus siihen, mitä datamusikalisaatio on, muutama esimerkki siitä, miten sitä on toteutettu ja miten sitä voitaisiin ehkä tulevaisuudessa toteuttaa ja millaisiin tarkoituksiin se on suunniteltu. Esittelen myös pintapuolisesti monia esimerkkejä erilaisten tietojenkäsittelyyn liittyvien menetelmien soveltamisesta musiikin generointiin ja analysoin niiden haittoja, hyötyjä ja soveltuvuutta datamusikalisaatioon. Lukija saa musiikin generointimenetelmistä yleiskuvan, josta on hyvä lähteä syventämään.

Musikalisaation yhteydet muihin aloihin

Merriam–Webster-sanakirja määrittelee säveltämisen (engl. *compose*) tarkoittamaan muun muassa musiikillisen teoksen suunnittelua ja kirjoittamista [53]. Algoritminen säveltäminen on tämän prosessin automatisoitu muoto, jossa säveltämisen tekee ihmisen sijaan algoritmi. Termiä konesäveltäminen käytetään algoritmisen säveltämisen kanssa rinnakkain. Algoritmista säveltämistä ei ole määritelty kovin tarkasti, ja on esimerkiksi epäselvää, kattaako se terminä tietokoneavusteiseen säveltämiseen tehdyt järjestelmät, joissa sävellys tapahtuu tietokoneen ja ihmisen yhteistyönä.

Sonifikaation voidaan määritellä tarkoittavan ”informaation siirtämistä sanoja sisältämättömällä audiomateriaalilla” [38]. Tämä määritelmä ei ole kovin tarkka, eikä se esimerkiksi ota kantaa siihen, millaista ”informaation siirtämistä” tarkoitetaan. Määritelmää voidaankin laajentaa seuraavasti: ”Sonifikaatio on datan relaatioiden muuntamista havainnoitaviksi relaatioiksi akustiseen signaaliin, tarkoituksena fasilitoida kommunikaatiota tai tulkintaa” [38]. Erityisen oleellista on erottaa sonifikaatio musiikista: musiikissa paino on enemmän taiteellisessa ilmaisussa ja estetiikassa, kun taas sonifikaatiossa oleellista ei ole, kuulostaako ääni hyvälle, vaan tärkeää ovat yksittäiset liikkeet datassa ja niiden kuuleminen.

Hermann [38] laajentaa sonifikaation määritelmää vielä tarkemmaksi. Hänen määritelmässään äänen tulee täyttää neljä eri kriteeriä ollakseen sonifikaatiota: sen pitää kuvata datassa olevia objektiivisia attribuutteja tai relaatioita, muunnoksen tulee olla systemaattinen eli täytyy olla tiedossa täsmälleen, miten mikäkin muutos datassa kuvautuu äänenä, tuloksen pitää olla toistettavissa eli samalla datalla ja muilla määritteillä pitäisi tulla aina identtinen tulos, ja järjestelmää pitää pystyä käyttämään monella eri datalla.

Määrittelen datamusikalisaation, lyhyemmin musikalisaation, tarkoittamaan prosessia, jossa algoritmi säveltää musiikkia, jossa lähteenä olevan datan ominaisuudet ovat kuultavissa. Musikalisaatio on sonifikaation ja algoritmisen säveltämisen yhdistävä ala.

Perinteisessä data-analyysissä tavoitteena on yleensä löytää jotain poikkeuksia tai säännönmukaisuuksia datasta ja prosessin tarkoitus on tuottaa objektiivista informaatiota käsiteltävästä asiasta [77]. Datamusikalisaatio poikkeaa tässä suhteessa huomattavasti sekä perinteisestä datan havainnollistamisesta (esim. visualisaatiot) että sonifikaatiosta. Datamusikalisaation arvo on juuri siinä, että musiikilla pystyy kuvaamaan dataa tavalla, joka herättää käyttäjässä, joka on usein myös datan lähde, subjektiivisia kokemuksia ja

ehkä jopa tunteita.

Musikalisaatio on konesäveltämisen tavoin laskennallisen luovuuden osa-alue. Laskennallinen luovuus on sellaisten järjestelmien tutkimusta, joiden käyttäytyminen voidaan tulkita luovaksi [15]. Yleensä tutkitaan järjestelmiä, joiden tuottamat teokset ovat perinteisten luovien alojen ammattien tuottamia, kuten kirjallisuus, runous, säveltäminen ja graafinen suunnittelu. Wiggins ja Colton [15] käyttävät laskennalliselle luovuudelle tarkempaa määritelmää, johon sisältyy, että järjestelmällä pitää olla vastuu tuloksesta, eli sen pitää olla oleellisilta osiltaan autonominen. Lisäksi heidän määritelmässään järjestelmän toiminnan pitää olla puolueettomien sivullisten mielestä luovaa. Hieman epäselväksi jää, miten tätä voidaan käytännössä arvioida, sillä Wiggins ja Colton itsekin myöntävät, että ihmiset ovat usein liian puolueellisia arvioimaan luovien konejärjestelmien tuottamia asioita.

Tarkemmin sanottuna musikalisaatio kuuluu laskennallisen luovuuden aloista jo aiemmin mainitun algoritmisen säveltämisen alle, sillä musikalisaatio on yksi tapa soveltaa ja käyttää algoritmisen säveltämisen oppeja.

Sonifikaation ja musikalisaation erot

Jos tutkitaan edellisessä osiossa esiteltyjä määritelmiä sonifikaatiolle ja säveltämiselle, voidaan puhua siitä, miten musikalisaatio ilmiönä suhtautuu näihin kahteen jo olemassa olevaan alaan. Jos käytetään epätarkempia esiteltyjä määritelmiä sonifikaatiolle, musikalisaatio voitaisiin hahmottaa sen alakäsitteeksi. Tarkempaa määritelmää käytettäessä musikalisaation sonifikaatiosta erottava tekijä on, että tuloksena olevan äänen pitää olla esteettistä ja tunnistettavissa musiikiksi, kun taas sonifikaatioksi voidaan laskea mikä tahansa ääni, joka ei ole puhetta, ja sama data voi tuottaa useita erilaisia lopputuloksia, sillä data vain ohjaa sävellysprosessia ja siinä voi olla lisäksi esimerkiksi stokastisia elementtejä.

Useimmista sonifikaatioista musikalisaation erottaa jo sekin, että suurin osa sonifikaatiotavoista on melko suoria kuvauksia ilman mitään erikoisempaa logiikkaa äänen ja datan välissä tai takuuta siitä, että lopputulos olisi musikaalinen [77], vaikka tämä ei päde kaikkiin sonifikaatioksi luettaviin tapoihin tuottaa ääntä. Sonifikaation tavoite on puhtaasti informaation siirtäminen, musikalisaation tavoite on luoda esteettinen kokemus datan pohjalta.

Musikalisaatiossa äänen symbolinen taso, eli kuvauksen suoruus, on usein eri kuin sonifikaatiossa. Sonifikaatiossa ilmiöitä kuvataan joko suoraan tai metaforisesti [62], kun taas musikalisaatiossa kuvaus on aina metaforinen,

eli dataa ei kuvata suoraan ääneksi. Luvussa 4 esitetään esimerkkejä metaforisesta datan kuvauksesta. Suora kuvaus voisi olla esimerkiksi DNA:n sonifikaatio, jossa jokaiselle neljälle symbolille on asetettu oma äänenkorkeus ja DNA on näin soitettu ääneksi.

Sonifikaatiossa on oleellista, että eri ominaisuudet datasta voidaan kuvata eri kuuloisilla tavoilla, jolloin kokonaisinformaation määrää saadaan nostettua niin, että kuuntelija vielä ymmärtää kuulemansa [81]. Musikalisaatiossa painopiste on musiikissa, jolloin äänien ja kuvaamistapojen valinnassa täytyy ottaa huomioon myös niiden esteettisyys. Musiikki on myös rajoittuneempi informaationkantokyvyltään [77].

Sonifikaatiosta poiketen musikalisaatiossa tarvitaan huomattavasti enemmän tietoa musiikista. Joissain tapauksissa musiikillinen tieto on ohjelmoijalla, jolloin se ohjelmoidaan käsin, joissain tapauksissa se opitaan koneoppimisella esimerkeistä. Lähteestä riippumatta musiikillinen tieto on olennaista [77]. Sonifikaatioon ei tarvitse minkäänlaista musiikillista tietoutta, vaikka on myös sonifikaatiometodeja, joissa sitä voi käyttää.

Musiikkitermejä myöhempiä lukuja varten

Jotta musiikillisia periaatteita olisi helpompaa selittää myöhemmissä luvuissa, lukijan on hyvä tuntea muutama yleisimmin tekstissä käytetty musiikkitermi.

Intervalli on kahden joko peräkkäisen tai samaan aikaan soivan sävelen välinen etäisyys. Intervalleista voidaan puhua joko intervallien nimillä tai numeroina, jolloin käytetään yksikkönä kokosävelaskelia.

Sointu on kolmen tai useamman yhtä aikaa soivan sävelen muodostama joukko, siinä missä kahta samaan aikaan soivaa säveltä sanotaan intervalliksi. Soinnun tyypin määrittää sen sisäisten sävelten intervallit toisiinsa nähden.

Puolisävelaskel on kahden vierekkäisen sävelen etäisyys eli pienin tyypillisessä länsimaisessa musiikissa käytetty etäisyys, esimerkiksi etäisyys c:n ja ylennetyn c:n eli cis:n välillä.

Kokosävelaskel on kahden puolisävelaskelen kokoinen etäisyys.

Sävellaji on säveljärjestelmä, joka määrittää, mitä säveliä, ja täten myös mitä sointuja, kappaleessa enimmäkseen käytetään. Tyypilliseen länsimaiseen duuri- tai mollisävellajiin kuuluu 7 säveltä (kaikki samalla kirjaimella nimetyt sävelet, korkeudesta riippumatta, esimerkiksi kaikkien oktaavien c:t). Myös sävellajin säveliä, jotka eivät kuulu sävellajiin, voidaan käyttää ja käytetäänkin musiikissa, mutta ne täytyy nuottikirjoituksessa merkitä

tilapäisillä ylennys-, alennus- tai palautusmerkeillä. Yksinkertaistettuna, yleisistä sävellajityypeistä duurit ”kuulostavat iloisilta” ja mollit ”kuulostavat surullisilta”.

Asteikko viittaa peräkkäisten sävelkorkeuksien sarjaan. Asteikko koostuu sävelten välisistä etäisyyksistä, eli se ei ole sidoksissa sävelkorkeuteen. Esimerkiksi C-duuri-asteikko on sarja säveliä, joka alkaa c:stä ja jossa kolmannen ja neljännen sekä seitsemännen ja kahdeksannen sävelen välissä on puolisävelaskeleet ja muiden välissä kokosävelaskeleet.

Sointuaste tarkoittaa joltain asteikon säveleltä lähtevää sointua, jossa on kyseisen asteikon säveliä. Esimerkiksi C-duurin toisen asteen sointu on d-molli, johon kuuluvat sävelet d, f ja a. Sointuasteista voidaan puhua riippumatta siitä, missä sävellajissa ollaan, sillä sointujen funktio kappaleessa riippuu lähinnä niiden sointuasteesta, eli esimerkiksi toisen asteen soinnun funktio on sama riippumatta siitä onko se C-duurin d-molli vai D-duurin e-molli.

Tempo on musiikin esitysnopeus. Sitä voidaan ilmaista joko käyttämällä adjektiivia tai ilmaisemalla, montako kappaletta jotain tiettyä nuotin pituutta mahtuu minuuttiin.

Konsonanssi viittaa sellaisten sävelien yhdistelmään, joiden sanotaan soivan hyvin yhteen. Vaikka eri kulttuuriympäristöissä onkin eri käsityksiä siitä, mitkä intervallit ovat konsonoivia, yleensä intervallin sanotaan olevan sitä konsonoivampi, mitä pienemmällä nimittäjällä ja osoittajalla varustetulla murtoluvulla äänien taajuuksien suhteen pystyy esittämään.

Dissonanssi on konsonanssin vastakohta, eli sävelyhdistelmä, joka kuulostaa korvaan riitaisalle ja useimpien mielestä epämiellyttävälle.

Tahtilaji kertoo, montako jotain nuottien pituusyksikköä mahtuu yhteen tahtiin, eli moneenko kappaleessa lasketaan. Esimerkiksi 4/4-tahtilajissa lasketaan neljään laskien neljäsosanuotteja.

Nuottien pituudet määräytyvät normaalisti sen mukaan, mikä on kappaleen tahtilaji, mutta tässä tutkielmassa tarkemmin esitellyt esimerkit toimivat 3/4 tai 4/4 -tahtilajeissa. Tällöin yksi neljäsosanuotti on musiikissa yksi rytmin peruselementti, joita mahtuu yhteen tahtiin kolme tai neljä. Puolinuotti on kahden neljäsosanuotin pituinen, kokonuotti neljän. Piste nuotin perässä pidentää keston puolitoistakertaiseksi. Kahdeksasosa on puolikas neljäsosanuotti, kuudestoistaosa puolikas kahdeksasosa ja 32-osa puolikas kuudestoistaosanuotti. Taukoja kuvataan samoilla nimityksillä.

Kontrapunkti on tapa kirjoittaa ja sovittaa yhteen melodioita tiettyjen sääntöjen mukaan [61]. Erilaisia kontrapunktisäännöstöjä on useita, mutta ne kaikki jakavat samat peruselementit.

Koraali on hyvin konsonoiva kirkkomusiikkisävellystyyppi. Monet koraalit kirjoitetaan neljälle lauluäänelle. Monia koraaleja voi myös sanoa hymneiksi.

2 Historia

Koska musikalisaatio on kahden olemassa olevan alan, algoritmisen säveltämisen ja sonifikaation leikkauksessa, esittelen tässä luvussa kummankin alan historiaa. Koska sonifikaatio on nuori ala ja koska musikalisaatio nojaa enemmän tekniikoiltaan algoritmiseen sävellykseen, esittelen sen historiaa tarkemmin, jotta seuraavassa luvussa esitellyt menetelmät olisi mahdollista laittaa historialliseen kontekstiin.

Sonifikaation historia

Tunnetuimpia vanhoja esimerkkejä sonifikaation periaatteesta on geigermittari, joka keksittiin vuonna 1908 [69]. Geigermittari pitää tikittävää ääntä ja mitä vahvempi säteily ympäristössä on, sitä tiheäpi tikitys. Toinen hyvin vanha esimerkki on vuonna 1913 keksitty Optophone, jonka on tarkoitus sonifikoida tekstiä [21]. Sen oli tarkoitus toimia lukukoneena sokeille henkilöille, mutta se ei koskaan yleistynyt koska sen käyttäminen ei ollut tarpeeksi nopeaa. Optophone soitti soinnun ja jätti soinnusta pois kirjaimia sen mukaan, missä kohti lukupään alla olevaa kirjainta ei ollut painomustetta.

Sonifikaatiolla on myös juurensa taiteissa. Alun perin abstraktien asioiden muuttamista ääneksi pohtivat muun muassa saksalainen runoilija Raine Maria Rilke vuonna 1919 kirjoitetussa esseessään kallon rakenteen muuttamisesta ääneksi ja László Moholy-Nagy vuonna 1923 kirjoittaessaan ”uudesta musiikista”, jota voisi tuottaa kaivertamalla kuvioita suoraan äänilevyyn, joka on käytännössä sonifikaatio graafisista elementeistä reilusti ennen tietokoneen keksimistä [24].

Sonifikaation tutkiminen oli pitkään mahdollista vain sellaisille tutkijoille, joilla oli käytössään supertietokoneita, sillä sonifikaation tai jopa moniulotteisten visualisaatioiden tekeminen oli liian raskasta yleisimmille tietokoneille [29]. Sonifikaation potentiaalista puhuttiin 1990-luvulla psykologien mielestä hyvinkin kyseenalaisesti ja sen väitettiin esimerkiksi lisäävän ihmisen datankäsittelypotentiaalia huomattavasti. Sonifikaatioon myös sovellettiin,

ja jossain määrin sovelletaan edelleen, virheellisesti samoja periaatteita, joita käytetään hyvien visualisaatioiden tekemisessä, vaikka kyse on hyvin eri tavalla toimivasta aistista [29].

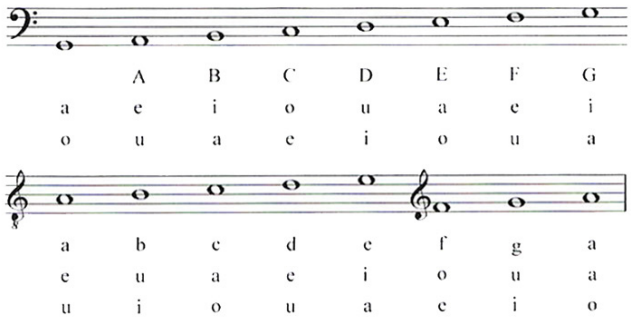
Sonifikaatio ei ole vanha ala eivätkä sen termistö tai rajat ole kovin vakiintuneita [38]. Hermannin artikkelin "Taxonomy and Definitions for Sonification and Auditory Display" mukaan sonifikatio omana alanaan syntyi vasta vuoden 1992 ICAD-konferenssissa, jonka artikkeleista koottu "Auditory Display" on vieläkin yksi alan tärkeimpiä teoksia [38]. Hermann on kuitenkin sitä mieltä, että lyhyestä olemassaoloajastaan huolimatta sonifikaatio on jo vakiinnuttanut itsensä oikeana alana ja sen potentiaali on tunnustettu.

Konesäveltämisen historia

Ennen tietokoneita

Jos määritellään algoritminen säveltäminen laajemmin niin, että määritelmään ei sisälly, että sen pitäisi olla tietokoneella tehtävä prosessi, idea on huomattavan vanha. Käsikäyttöisiä prosesseja musiikin generointiin on ollut jollain tasolla olemassa jo ainakin vuoden 1026 tienoilta, kun Guido d'Arezzo kehitti menetelmän, jolla voitiin generoida melodia tekstin pohjalta eli muuttaa annettu teksti lauluksi [45]. Guido d'Arezzon menetelmä esiteltiin hänen kirjassaan "Micrologus", ja se perustuu siihen, että säkeen osat, tavut ja yksittäiset kirjaimet kuvautuvat sävelkorkeuksiksi ja melodisiksi fraaseiksi [56]. Arezzon menetelmä on kuvattu kuvassa 1. Musiikkia on jo pitkään käsitelty proseduraalisesti, mutta tälle käsittelytavalle ei ole kirjoitettu toistaiseksi mitään koostettua historiaa [68].

Lukujen suhteisiin perustuvat menetelmät, jotka ovat nykyään yleisesti



Kuva 1: Guido d'Arezzon menetelmä melodian generointiin tekstin pohjalta [56].

käytössä modernissa taidemusiikissa, ovat myös algoritminen tapa ajatella musiikkia. Niitä onkin myöhemmin käytetty myös konesävellyksessä, mutta ensimmäiset esimerkit ovat reilusti tietokoneita vanhempia. Jo 1400-luvulla elänyt Guillaume Dufay, ja muutamat muutkin hänen aikalaisensa, käyttivät sävellyksissään transformaatioita, kuten inversioita (positiivisten intervallien kääntäminen negatiivisiksi ja toisinpäin) ja järjestyksien kääntämisiä. Dufay myös esimerkiksi johti yhden motettinsa tempot erään katedraalin mittasuhteista [68]. 1650-luvulla Kircher, kuten myös monet muut, esitteli kirjassaan "Musurgia Universalis" tavan generoida kontrapunktia ja pienillä muutoksilla myös muita tyylejä [56].

Luultavasti kuuluisin vanha sävellysalgoritmi on Wolfgang Amadeus Mozartin kehittämä Musikalisches Würfelspiel eli musikaalinen noppapeli [68]. Kyseinen noppapeli julkaistiin vuonna 1792 ja vaikka sen julkaisi Mozartin julkaisija, niin ei ole varmuutta siitä, että Mozart todella olisi suunnitellut ja säveltänyt sen [16]. Noppapelissä pelaaja päättää nopanheitolla, minkä taulukossa olevista kahden tahdin pituisista pätkistä hän soittaa seuraavaksi. Joihinkin kohtiin on 12 erilaista vaihtoehtoa, joihinkin vain yksi. Lopputuloksena on 16 tahdin valssi ja erilaisia lopputuloksia on miljardeja. Erilaiset musikaaliset noppapelit olivat suosittuja ja niitä oli samoihin aikoihin muitakin. Ensimmäinen esimerkki konseptista oli Johann Philipp Kirnbergerin menuettien ja poloneesien säveltämispeli, joka julkaistiin jo vuonna 1751 [56].

Vuonna 1822 Bostonissa myytiin tuotetta nimeltä "Kaleidacousticon system", jota voisi ehkä sanoa ensimmäiseksi musiikkiohjelmistoksi [68]. Se oli pelikorttipakka, jonka mukana tuli ohjeet, miten sillä pystyy säveltämään valsseja. Kaleidacousticonilla pystyi generoimaan yhteensä noin 214 miljoonaa erilaista valssia. Vastaavia järjestelmiä oli myöhemmin muitakin, kuten "Quadrille Melodist", jota myytiin Lontoossa vuonna 1865 professori J. Clintonin toimesta "ammattimaiseksi sävellysavuksi" [70].

1900-luvulla monet matemaattiset ja todennäköisyyspohjaiset menetelmät otettiin käyttöön. Säveltämisessä käytettiin esimerkiksi joukko-oppiin kuuluvia menetelmiä musiikillisen materiaalin generoimiseen [68]. Lähes mikä tahansa matemaattinen kaava voidaan muuttaa jollain tavalla musiikiksi [68].

Jonkinlaista koneellista automaatiota säveltämiseen on myös ollut jo ennen tietokoneiden keksimistä. Kaukaisimmat juuret voidaan nähdä jo tuulivoimalla toimivissa Aeolisissa harpuissa, jotka tuottivat ambienttia ja satunnaista musiikkia antiikin aikoina [68]. Vuonna 1660 Athanasius Kirchner suunnitteli käsikäyttöisen sävellyskoneen nimeltä Arca Musirithmica ja myöhemmin vuonna 1821 Dietrich Winkel suunnitteli samantapaisen,

mutta rattailla ja käsipumpulla toimivan Componium-nimisen sävellyskoneen, joka tuotti variaatioita siihen etukäteen ohjelmoiduista teemoista [68]. Näissä esimerkeissä ei vielä käytetty sähköä.

Alalogisekvensoinnin pioneerit H. Olson ja H. Belar rakensivat jo vuonna 1951 ensimmäisen elektronisen sävellyskoneen [68]. Kyseisen koneen suurin innovaatio oli todennäköisyyksiin perustuvan sävellysmenetelmän automaatio. Kone käytti neliöaaltogeneraattoria lähteenä satunnaislukugeneraattorille.

1950-luvulla yleistyivät myös ensimmäiset kaupalliset laitteet, joita sanottiin ”elektronisiksi aivoiksi”. Jotkut niistä, kuten esimerkiksi GENIAC, mainostivat itseään sillä, että ne osaavat ”säveltää” muiden toimintojensa kuten ”leikkimisen” ja ”ajattelun” lisäksi [68]. Käytännössä GENIACin sävellykset olivat ilmeisesti hyvin yksinkertaisia.

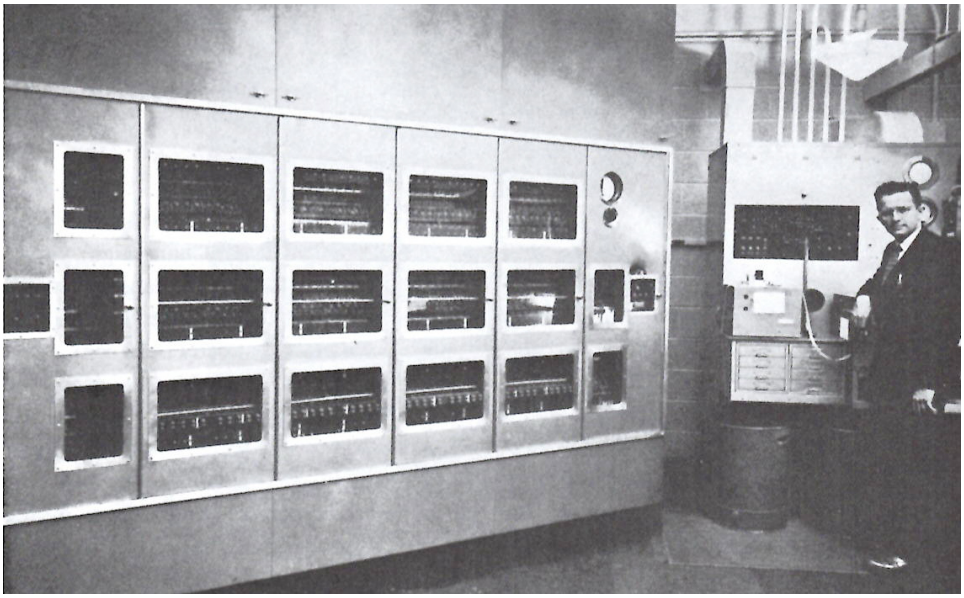
Alkeellisten syntetisaattorien yleistyessä algoritmeilla sävellettyä musiikkia pystyi toistamaan kätevästi, ja tämä osaltaan lisäsi kiinnostusta konesäveltämiseen [68]. Sekvenssikontrollikielten avulla ohjattavia syntetisaattoreita alkoi tulla markkinoille vuoden 1929 jälkeen [68]. Vaikka monet näistä rakennettiin tietokoneiden keksimisen jälkeen, tietokoneet olivat vielä niin suurikokoisia ja kalliita, että oli edullisempaa tehdä yksinkertaisempia koneita, jotka pystyivät sekä soittamaan että generoimaan musiikkia.

Barrin ja Stroudin säveltävä tietokone ”Solidac” asennettiin Glasgown yliopistolle vuonna 1959. Paperinauhoilla toimiva Solidac sekä sävelsi että toisti itse säveltämänsä musiikin. Sen ohjelmisto nimeltä ”Dice-Music Master Programme” osasi tekijöidensä mukaan generoida melkein miljardi erilaista Haydn-tyylistä trioa [68].

Myös tietokoneavusteinen sävellys keksittiin ennen tietokoneiden kultakautta. Electronium oli isokokoinen säveltävä kone, jota käytettiin vetämällä napeista ja kytkimistä [68]. Electronium ehdotti käyttäjälle melodiaa kaiutinten kautta, ja kun se ehdotti käyttäjälle mieluista melodiaa, käyttäjä veti vivusta ja kone siirtyi seuraavaan sävellysvaiheeseen. Koneen tuottamaa musiikkia pystyi muutenkin manipuloimaan kääntämällä erilaisia koneessa olevia vipuja.

Tietokoneiden alkuajat

Tietokoneita alettiin käyttää musiikin tuottamiseen 1950-luvun puolivälissä [28], vaikka idealla spekuloitiin jo ainakin Ada Lovelacen aikoihin vuonna 1842 [68]. Yleisesti ensimmäisenä tunnettuna tietokoneen säveltämänä kappaleena pidetään Hillerin ja Isaacsonin Illiac Suitea [40]. Se sävellettiin vuonna 1956 ja soitettiin yleisölle kyseisen vuoden kesäkuussa konsertissa, joka sai



Kuva 2: Tietokone, jolla Illiac Suite sävellettiin, sekä Illiac Suiten säveltäneen ohjelman luoja Lejaren Hiller [68].

kohtuullisen paljon julkisuutta. Säveltämiseen ohjelma käytti sääntöjärjestelmiä ja Markovin ketjuja. Illiac Suite koostui neljästä erillisestä osasta, joista ensimmäiset kaksi käyttivät kontrapunktia, kolmas satunnaislukuja ja 12-säveljärjestelmää ja neljäs Markovin ketjuja [2]. Kuvassa 2 on Hiller ja säveltämiseen käytetty tietokone. Hiller oli myös tiettävästi ensimmäinen ihminen, joka käytti tietokoneita musiikin tulostamiseen [68].

Hillerin kollegat samassa yliopistossa jatkoivat konesävellyskokeiluja käyttäen apunaan Bakerin ohjelmoimaa MUSICOMP-kirjastoa [28]. Se oli tiettävästi ensimmäinen kirjasto, joka tarjosi valmiita implementaatioita erityisesti asioihin, joita tarvitaan konesävellyksessä. MUSICOMP tulee sanoista MUsic Simulator Interpreter for COMpusitional Procedures [2]. Hiller ja Baker tekivät MUSICOMP-kirjastoa käyttäen vuonna 1963 seuraavan merkkiteoksen nimeltään ”Computer Cantata”, jossa erikoista oli esimerkiksi sen käyttämät poikkeukselliset viritysjärjestelmät, joissa oktaaviin kuuluu yhdeksästä viiteentoista säveltä [17].

Lähes samaan aikaan Illiac Suiten kanssa Martin Klein ja Douglas Bolitho suunnittelivat Burroughs Inc. yritykselle säveltävän koneen [2]. Tämä mainostempuksi tarkoitettu säveltävä järjestelmä toimi DATATRON-nimisellä tietokoneella. Ensimmäinen televisioon päätynyt DATATRONin säveltämä melodia oli nimeltään ”Push Button Bertha” ja se esitettiin heinäkuussa

1956.

1960-luvulta eteenpäin

1960- ja 1970-luvuilla kirjoitettiin kolme tunnettua sävellysohjelmaa, joita käyttivät muutkin kuin ne kirjoittaneet henkilöt [68]. Iannis Xenakis kirjoitti ohjelman nimeltä Stochastic Music Program, G. M. Koenig ohjelman nimeltä Project 1 ja Barry Truax kirjoitti monta versiota ohjelmasta nimeltä POD. Stochastic Music Program ja Project 1 tulostivat alun perin vain listoja nuoteista, jotka oli tarkoitus kääntää käsin perinteisiksi nuoteiksi ja sitten soittaa akustisilla instrumenteilla ihmisvoimin, mutta POD-ohjelmat suunniteltiin kiinnittymään digitaalista ääntä tuottaviin laitteisiin.

Stochastic Music Program mallinsi sävellystä käyttäen kaavoja, joita oli alun perin käytetty kuvaamaan partikkelien käytöstä kaasussa. Ohjelma mallinsi sävellystä osina, joista jokaisella oli kesto ja sävelten tiheys osan sisällä. Käyttäjä antoi sävellyksen attribuutit eli määritti nuottien maksimi- ja minimitiheydet osien sisällä, instrumenttien luokitukset, keskimääräisen osan pituuden, instrumentin pisimmän mahdollisen sävelen keston, instrumenttien luokitusten jakauman sävellyksessä ja yksittäisten instrumenttien todennäköisyydet soittaa. Xenakis käytti apuna Stochastic Music Programia moniin merkkiteoksiinsa, kuten vuonna 1964 ilmestyneeseen Eonta-nimiseen teokseen. Vuonna 1979 ohjelmaa paranteli John Myhill, jonka jälkeen se julkaistiin myös kotitietokoneille.

Project 1 -ohjelma sävelsi valikoimalla sävellyksen elementtien parametreja tietokannasta (tietokantapohjaiset ratkaisut yleistyivät myöhemmin). Ohjelmalla oli tietokanta, missä oli seuraaviin kategorioihin kuuluvia parametreja: harmonia, rytmi, instrumentti, rekisteri ja dynamiikka. Tietokannasta se poimi parametrit jokaiselle musiikilliselle elementille käyttäen vaihtelevasti erilaisia valintakriteerejä, joista osa oli täysin deterministisiä ja tuottivat hyvin itseään toistavaa musiikkia, kun taas jotkut olivat hyvinkin satunnaisia. Ohjelma pyysi käyttäjältä aluksi lopullisen musiikkielementtien määrän, tempot, satunnaislukugeneraattorin alustukseen käytettävän luvun ja eri kokoisten sointujen painoarvot. Koenig ei koskaan itse ajatellut että Project 1 olisi kokonainen sävellysohjelma, vaan enemmänkin raakamateriaalin tuottaja, josta säveltäjä laajentaa täyden sävellyksen.

Kolmas kuuluisa sävellysohjelma oli Truaxin POD, jonka nimi tulee sanoista ”Poisson Distribution” eli Poissonin jakauma. POD-ohjelmasta oli monta versiota, mutta yksinkertaistamiseksi puhun niistä kollektiivisesti yhtenä ohjelmana. POD poikkesi edellä esitellyistä kahdesta ohjelmasta sillä,

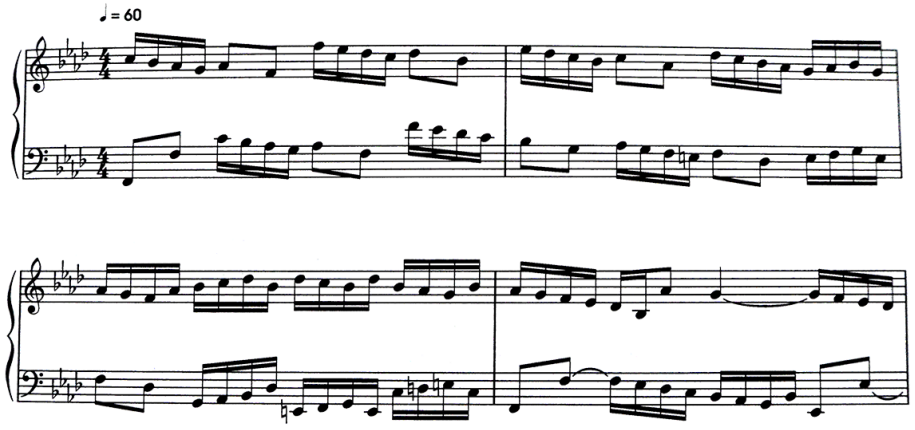
että sitä ei alun perinkään suunniteltu tuottamaan perinteisiä nuotteja, vaan soittamaan suoraan sävelmänsä syntetisoidulla äänellä. POD käytti nimensä mukaisesti Poissonin jakaumaa määrittämään elementtien jakaumaa ajassa ja nuottikorkeudessa ja täytti käyttäjän asettamien rajoitteiden ja attribuuttien mukaan sävellyksen.

1970-luvulla tehtiin yksi ensimmäisiä järjestelmiä, joka pohjautui materiaalin tarkkaan analyysiin. F. Brooks, A. Hopkins, P. Neumann ja W. Wright tekivät hymnejä säveltävän ohjelman, joka loi sävellyksensä pohjautuen 37 hymnin tilastolliseen analyysiin [17].

Vuonna 1986 ohjelmoidulla CHORAL-nimisellä järjestelmällä pystyi generoimaan neliäänisiä koraaleja eli kirkollistyylisiä kuorokappaleita, J. S. Bachin tyyliin [26]. CHORALin rakentamista varten sen tekijä Kemal Ebcioglu teki poikkeuksellisesti oman ohjelmointikielen Algolin kaltaisesti suorittuvan, mutta syntaktisesti Lispin tapaisen kielen nimeltä BSL. Järjestelmä toteuttaa yli 270 erilaista sääntöä, heusristiikkaa ja rajoitetta, joita käytetään uusien kappaleiden generoinnissa. CHORALin toimintaperiaatteet saatiin tutkimalla manuaalisesti Bachin koraaleja ja musiikkianalyysin perusteella. Omassa arviossaan Ebcioglu sanoi, että ohjelman tuottamat kappaleet ovat vain kohtalaista laatua ja että sen tyyli ei ole Bachin tyyli, mutta se oli kuitenkin edustava muihin aikalaisiinsa verrattuna ja osasi ottaa huomioon useita kappaleen attribuutteja päätöksenteossaan.

David Cope on säveltäjä ja ohjelmoija, joka on tehnyt pitkän uran algoritmisen sävellyksen parissa. Hänen työnsä konesävellyksen parissa on tuottanut kaksi merkittävää säveltävää ohjelmaa: Emmyn/EMIn ja Emily Howellin. Emmyn, toisissa lähteissä nimeltään EMI, nimi tulee sanoista "Experiments in Musical Intelligence" eli vapaasti suomennettuna kokeita musiikillisessa älyssä. Cope rakensi Emmyn 1980-luvun alkupuolella, mutta vasta 1980-luvun puolivälin jälkeen se alkoi tuottaa onnistuneesti kappaleita määrätyissä tyyleissä [18]. Hän jatkoi ohjelman kehittämistä ja ylläpitoa vuoteen 2003 asti.

Lopullisessa muodossaan Emmy on hyvin aineistopohjainen sävellysohjelma. Cope itse kuvasi asiaa näin: Emmy on laaja tietokanta ihmisten säveltämää musiikkia, jonka päällä on melko kookas analyysiohjelmisto, joka syöttää tietoa pienelle sävellysohjelmalle [18]. Emmyn toimintaperiaate on, että se analysoi jonkin klassisen säveltäjän kappaleita ja kerää niistä tietoa siirtymistä eri elementtien välillä, kuten esimerkiksi siitä, mihin sointuasteeseen siirrytään jostain tietystä sointuasteesta. Siirtymiä käsitellään kuitenkin kokonaisuuksina pohjaten siihen, että ohjelma ymmärtää nuottien funktion



Kuva 3: Ote Emmyn Bachin tyyliin säveltämästä pianokappaleesta [16].

sen kautta, missä asemassa ne ovat sillä hetkellä aktiivisena olevassa soinnussa ja sävellajissa. Emmy käyttää analyysiin menetelmää nimeltä SPEAC. Lyhenne tulee sanoista 'statement' (lause), 'preparation' (valmistelu), 'extension' (jatke), 'antecedent' (edeltäjä) ja 'consequent' (seuraus). Ohjelma luokittelee musiikin palasia näihin luokkiin sen mukaan, mikä niiden funktio musiikissa on, ja rekombinaatiovaiheessa, jossa ohjelma yhdistelee musiikin palasia sävellykseksi, se käyttää siirtoverkkoja järjestelemään palasia takaisin johonkin koherenttiin järjestykseen. SPEAC-luokitelluille palasille on säännöt, joiden mukaan niitä voi laittaa jonoon, ja ne löytyvät kokonaisuudessaan esimerkiksi Nierhaussin kirjasta [56]. Copen analyysikoneisto on monimutkainen ja sen selittämiseen tarvitsisi enemmän musiikin teoriaa kuin tässä tutkielmassa on mahdollista käydä läpi, mutta asiasta voi lukea lisää kirjasta "Experiments in Musical Intelligence" [16], jossa mekanismit on selitetty tarkemmin. Kuvassa 3 on esimerkki Emmyn tuottamasta musiikista. Copen mielestä Emmyn tapa säveltää rekombinaation kautta on luonnollinen, sillä kaikki ihmisenkin säveltämä musiikki on pohjimmiltaan olemassa olevien elementtien uudelleenjärjestelyä uusiksi kokonaisuuksiksi [19].

Emmyn erikoispiirre, joka tekee siitä yhden onnistuneimmista matkivista järjestelmistä, on se, että sen monimutkainen analyysikoneisto tunnistaa ja käyttää hyödyksi sävellyksien erikoispiirteitä. Se etsii elementtejä, joita Cope kutsuu nimellä säveltäjän "signature" eli kädenjälki. Ne ovat harmoniaa, melodiaa ja rytmiä yhdisteleviä hahmoja, jotka toistuvat monissa saman säveltäjän teoksissa ja ovat usein noin yhdestä kolmeen tahtia. Nämä elementit auttavat kuuntelijaa tunnistamaan säveltäjän tyylin ja löytäessään niitä

analyysissään Emmy tallentaa elementin kokonaisena ja käyttää sitä omissa kopioteoksissaan saaden näin aikaiseksi uskottavampia jäljitelmiä. Toinen elementti, jota Emmy etsii lähdeteoksista auttamaan jäljittelyssä, on elementti, jota Cope kutsuu sanalla "character" eli ominaispiirre. Se koostuu rytmin ja yhtäaikaisesti soivien eri nimisten nuottien lukumäärän yhdistelmästä. Emmy tallentaa tietoja siitä, millainen ominaispiirre esiintyy usein jossain tietyssä kontekstissa ja pyrkii säveltäessään noudattamaan näitä tallennettuja suuntaviivoja [18].

Viimeaikaiset saavutukset

Nykyään laskennallista luovuutta tutkivia tahoja on maailmalla paljon ja moni niistä on tehnyt tutkimusta konesäveltämisen parissa, joten olisi lähes mahdotonta esitellä tässä kaikkia säveltäviä algoritmeja, jotka ovat saavuttaneet jonkinlaisia onnistumisia. Esittelen kuitenkin muutamia 2000-luvun puolelta olevia, suhteellisen tunnettuja säveltäviä algoritmeja. Valitsin esiteltäväksi sellaisia algoritmeja, jotka ovat saaneet tunnustusta, enkä esittele useampaa samanlaista menetelmää käyttävää algoritmia.

Yksi hyvin tunnettu esimerkki säveltävästä ohjelmasta on ihmisenkaltaisesti nimetty Emily Howell. David Copen ohjelmoima Emily on suoraa jatkumoa aikaisemmin mainitulle Emmylle. Kummankin ohjelman toimintaperiaatteet ovat samankaltaisia ja kumpikin käyttää laajaa musiikkitietokantaa apunaan. Emilyn tarkoitus ei kuitenkaan ole matkia vaan säveltää musiikkia omalla uniikilla tyylillään [18]. Emilyn tietokannassa on Emmyn eri tyylejä matkien säveltämiä kappaleita, eikä kaikilla kappaleilla tietokannassa välttämättä ole samankaltaisia tyylipiirteitä.

Isompi erottava piirre Emmyn ja Emilyn välillä on se, että Emilyn toimintaa ohjaa assosiaatioverkko (engl. *association network*). Assosiaatioverkko koostuu sisääntulosta, ulostuloista ja toisiinsa kytketyistä solmuista, joihin on säilötty tietoa ja analyysin tuloksia, mutta sitä ei pidä sotkea neuroverkkoon [18]. Emilyn assosiaatioverkko ottaa vastaan missä tahansa muodossa olevaa tietoa, jonka se sitten varastoi. Varastoituaan tiedon verkko arpoo uusien solmujen ja kaikkien aikaisempien solmujen välille aluksi satunnaiset siirtymätodennäköisyydet. Käyttäjä sitten kouluttaa verkkoa niin, että verkon tuottamiin asioihin vastataan joko myöntävästi tai kieltävästi. Kieltävä vastaus pienentää verkossa olevien relevanttien kaarien painoja ja myöntävä vastaus nostaa niitä. Käyttäjä kouluttaa pikkuhiljaa ohjelman tuottamaan järkeviä tuloksia. Emilyn säveltämä levy julkaistiin toukokuussa 2010 Centaur Records -yhtiön toimesta.

Vuonna 2012 Francisco Vico ja hänen kollegansa Malagan yliopistosta kehittivät säveltävän algoritmin nimeltä Iamus [4]. Iamuksen sävellyksiä julkaistiin levyllä vuoden 2012 syyskuussa. Levyllä on useita algoritmin säveltämiä kappaleita erilaisille kokoonpanoille. Isoimmat kappaleet on sävelletty kokonaiselle sinfoniaorkesterille ja niiden esittäjänä on levyllä London Symphony Orchestra. Nimekkään orkesterin saaminen algoritmista sävellystä esittelevälle levylle onkin poikkeuksellista ja se erottaa osaltaan Iamusta edeltäjistään, sillä yleensä musiikin ammattilaiset eivät ole kiinnostuneita soittamaan konesävellettyä musiikkia [4]. Tyyliltään Iamuksen sävellykset ovat modernia klassista musiikkia eli täynnä riitasointuja ja vanhempaan klassiseen musiikkiin verrattuna sekavia ja omituisia ratkaisuja.

Toinen poikkeuksellinen ominaisuus Iamuksessa on, että se ei käytä edes välillisesti ihmisten tuottamaa musiikkia sävellystensä pohjana. Iamus generoi lyhyitä teemoja, joita se sitten mutatoi geneettisillä algoritmeilla. Vaikka geneettiset algoritmit eivät olekaan uusi keksintö konesäveltämisen kentällä, Iamuksen sävellykset ovat poikkeuksellisen monimutkaisia [66].

Myös Google on lähtenyt mukaan algoritmisen sävellyksen tutkimiseen. Google Magenta [34] on Google Brain -tiimin avoimen lähdekoodin sävellysprojekti, jolla on kaksi tavoitetta: edistää laskennallisen luovuuden tutkimusta, ja toisaalta myös kerätä yhteen artisteja, koneoppimistutkijoita ja ohjelmoijia. Google Magenta ei ole vielä saanut suuria läpimurtoja, mutta siihen kohdistuu odotuksia taustaorganisaation muiden projektien menestyksen takia.

3 Konesävellysmenetelmät

Erilaisia tapoja generoida musiikkia koneella on monia. Käytännössä monet järjestelmät ovat hybridejä, joissa käytetään useampaa erilaista menetelmää. Monia menetelmiä voi yhdistellä sujuvasti toisiinsa, mutta ne toimivat myös yksittäin. Eri lähteissä käytetään erilaisia taksonomioita sävellysalgoritmien jaottelulle, ja monissa eri menetelmissä on myös jonkin verran päällekkäisyyttä. Tässä luvussa on käytetty Gerhard Nierhausin kirjan "Algorithmic Composition" [56] tapaa jaotella algoritmejä. Muita tapoja löytyy esimerkiksi Vicon, Fernin ja Rodriguezin artikkelista "AI Methods in Algorithmic Composition: A Comprehensive Survey" [28] tai Papadopoulosin ja Wigginsin artikkelista "AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects" [59]. Jos lähdettä ei ole mainittu erikseen, se on Nierhausin kirja. Arvioin myös jokaisen aliluvun lopuksi sitä,

vaikuttaako kyseinen musiikin generointitapa siltä, että se soveltuisi hyvin musikalisaatioon.

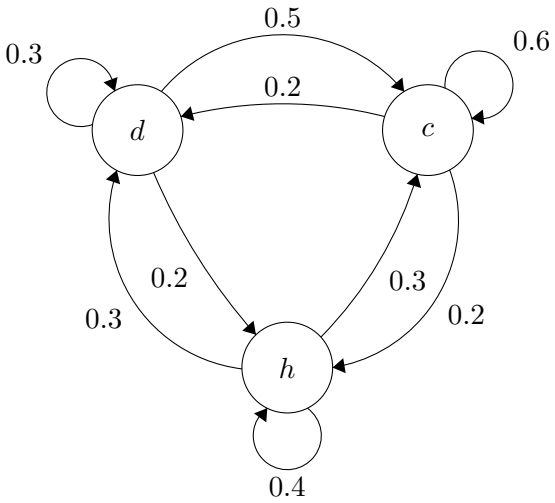
3.1 Markovin ketjut

Markovin ketjut saivat nimensä keksijältään Andrey Andreyevich Markovilta, joka oli venäläinen matemaatikko. Markov julkaisi jo vuonna 1906 ensimmäisen Markovin ketjuihin liittyvän artikkelinsa, joka käsitteli yleisemmin ajasta riippuvaisia satunnaismuuttujia. Markovin ketjuja kutsutaan myös Markov-malleiksi. Termiä käytti ensimmäistä kertaa venäläinen matemaatikko Sergey Natanovich Bernstein vuonna 1926. Ensimmäinen kuuluisa sovellus oli Markovin. Hän analysoi A. S. Pushkinin Eugene Onegin -runon 20 000 kirjainta ja sai muun muassa todennäköisyydet sille, että vokaalia seuraa konsonantti.

Markovin ketjuissa seuraava tila riippuu aina sitä edeltävästä tilasta tai tiloista. Jos malli ottaa huomioon useamman edeltävän tilan, sitä sanotaan korkeamman asteen malliksi. Esimerkiksi kaksi edellistä tilaa yhden sijaan huomioon ottava malli olisi toisen asteen Markovin ketju. Markovin ketjujen rajoitus on, että ne ottavat huomioon vain edellisen tilan tai edelliset tilat, eivätkä samanaikaisesti tapahtuvia muita asioita, kuten toisia ääniä.

Esitetään yksinkertainen esimerkki, jossa seuraavan sävelen korkeus riippuu edellisen sävelen korkeudesta. Markovin ketjuja voi havainnollistaa siirtymämatriiseilla tai tilakaavioilla. Kuvassa 4 on hyvin yksinkertaisen Markovin ketjun havainnollistus. Tässä mallissa on kolme säveltä: d, c ja h. Malli kuvaa siirtymistodennäköisyyksiä niiden välillä. Tilakaaviossa jokainen sävel on oma tilansa ja todennäköisyydet siirtyä kustakin sävelestä toiseen (mukaan lukien sävel itse) on merkitty verkon kaarille. Mistä tahansa yhdestä tilasta lähtevien todennäköisyyksien tulee Markovin ketjussa olla aina summaltaan yksi. Taulukossa on sama tieto esitettynä eri tavalla niin, että pystyakselilla on aktiivinen sävel ja vaaka-akselilla todennäköisyydet seuraavalle sävelelle. Esimerkiksi jos nykyinen sävel on c, todennäköisyys sille, että seuraava sävel on myös c, on 0.6 eli 60 prosenttia.

Markovin ketjut muodostetaan usein aineiston pohjalta niin, että siirtymät saavat todennäköisyytensä sen mukaan, kuinka yleisiä ne ovat käytettävässä aineistossa. Esimerkiksi monet säveltävät ohjelmat perustuvat siihen, että nuottien, sointujen tai jonkin muun musiikillisen attribuutin tai näiden yhdistelmien väliset siirtymätodennäköisyydet lasketaan aineistosta, joka on yleensä kokoelma tietyn säveltäjän tai tyylilajin musiikkia. Markovin ketju-



	d	c	h
D	0.3	0.5	0.2
C	0.2	0.6	0.2
H	0.3	0.3	0.4

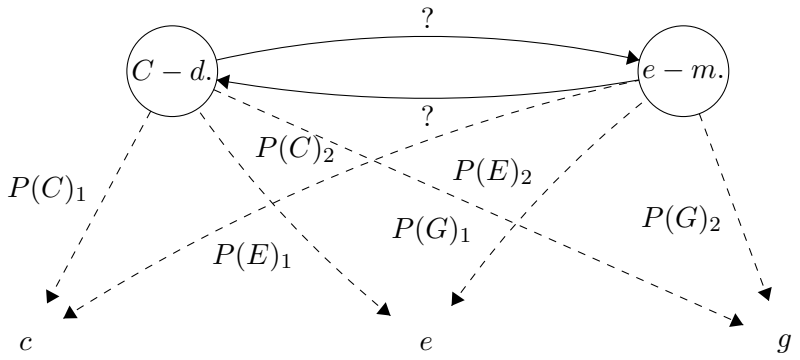
Kuva 4: Esimerkki yksinkertaisen Markovin ketjun tilakaaviosta siirtymämatriisista, jossa pystyakselilla on nykyinen sävel ja vaaka-akselilla todennäköisyydet seuraaville sävelille.

jen siirtymätodennäköisyydet voidaan saada myös kirjoittamalla ne käsin esimerkiksi musiikin teorian sääntöjen perusteella tai kokeilemalla erilaisia vaihtoehtoja ja iteroimalla [28].

Mitä isompi Markovin ketjun aste on, sitä ennustettavammaksi lopputulos yleensä muuttuu ja sitä läheisemmin se alkaa muistuttaa käytettyä aineistoa, jos todennäköisyydet on opittu aineistosta. Korkeissa asteluvuissa on toisaalta se ongelma, että jos jotain asteluvun pituista sekvenssiä ei ole aineistossa, sitä ei myöskään voi syntyä, koska sille ei ole asetettu todennäköisyyttä. Tätä ongelmaa voidaan kiertää interpoloimalla puuttuvalle sekvenssille todennäköisyys käyttäen matalamman asteen ketjuja. On myös olemassa muuttuvan kertaluvun Markovin ketjuja (engl. *variable-order Markov chain*), jotka eivät ole sidottuja tiettyyn astelukuun, vaan käyttävät montaa eri astetta [28].

Markovin piilomalli (engl. *Hidden Markov Model*, HMM) on muunnelma Markovin ketjusta. Markovin piilomalli on järjestelmä, jonka tulokset voi nähdä, mutta niitä tuottava Markov-malli on tuntematon. Markovin piilomallista ei voi varmasti tietää, missä tilassa se milloinkin on, mutta tuloksen ja niiden todennäköisyyksien perusteella asiasta voi esittää approksimaation. Piilomallia käytetään usein esimerkiksi kuviotunnistukseen liittyvissä tehtävissä, kuten eleiden, käsialan tai puheentunnistukseen.

Otetaan esimerkiksi järjestelmä, joka voi olla joko tilassa, jossa valittu sointu on C-duuri, tai tilassa, jossa se on e-molli, mutta se on Markovin



Kuva 5: Esimerkki Markovin piilomallista. Tässä kysymysmerkit kuvaavat tilojen välisiä siirtymätodennäköisyyksiä, joita ei tiedetä ja P-kirjaimet todennäköisyyksiä sille, että jokin tila antaa tietyn tuloksen.

piilomalli, joten emme voi tietää kummassa tilassa järjestelmä milloinkin on, emmekä yleensä sisäisten mallien siirtymätodennäköisyyksiä. Sanotaan, että tiedämme, että soivia säveliä on neljä: c, e, g ja h. Joten jos tiedämme, että sävel on e, tiedämme vain todennäköisyydet sille kummassa tilassa järjestelmä on. Voimme esimerkiksi päätellä, että koska sävel c kuuluu vain C-duuriin mutta ei h-molliin, c:n soidessa on todennäköisempää, että sointu on c. Tästä esimerkistä on graafinen esitys kuvassa 5. Riippuen siitä, mitä järjestelmästä tiedetään, voidaan soveltaa erilaisia algoritmeja, kuten Viterbi [79] ja Baum-Welch [82], selvittämään sisäinen tila tai sen siirtymätodennäköisyydet.

Markovin ketjuja on käytetty algoritmisessa säveltämisessä paljon, ja ensimmäiset esimerkit niiden käytöstä ovat vanhoja. Ensimmäinen, joka käytti Markovin ketjuja musiikin generointiin, oli Harry F. Olson noin vuonna 1950 [56]. Markovin ketjuja käytettiin myös esimerkiksi luvussa 2 mainitussa Illiac Suitessa. Vuonna 1957 Brooks ja hänen ryhmänsä tekivät koraalimelodioita generoivan ohjelman, jonka todennäköisyydet opittiin aineistosta, johon kuului 37 keskenään samankaltaista koraalia [12]. Edellä mainitussa ohjelmassa sävelkorkeudet oli edustettu numeroilla ja aineistosta johdettiin kahdeksannen asteen Markovin ketjut. Ohjelma pystyi generoimaan yhteensä 1701 erilaista melodiaa. Vuonna 1992 Ames ja Domino käyttivät Markovin ketjuja rytmin generoimiseen jazzia ja rockia säveltävässä projektissaan Cybernetic Composer [3]. Viimeaikainen esimerkki on Toivasen ja hänen kollegoidensa M.U. Sicus-Apparatus [76], joka säveltää itse generoimiinsa lyriikoihin lauluja, joihin se generoi sointukulkuja toisen asteen Markovin ketjuilla. Eräs esimerkki on myös luvussa 4 esitellyt järjestelmät, jotka käyttävät

Markovin ketjuja.

Markovin piilomallia on käytetty säveltämiseen vasta viime aikoina, kun taas perinteisten Markovin ketjujen suosio ainoana sävellysmetodina on laskenut huomattavasti viime vuosituhannen puolelta [28]. Ensimmäinen esimerkki on Farboodin ja Schönerin Palestrina-tyylin kontrapunktia säveltävä ohjelma vuodelta 2001 [27]. Markovin piilomalleilla on myös tehty kaupallisesti menestyneitä projekteja, kuten Microsoftin ohjelma Songsmith [5], joka generoi automaattisesti piilomalleilla taustat sillä nauhoitettuun tai nuotinnettuun melodiaan.

Musikalisaatiossa käytettäväksi Markovin ketjut käyvät hyvin. Niiden yksinkertaisuus tekee niistä helposti ohjattavia. Jos esimerkiksi käytetään kertymätodennäköisyyttä seuraavan rytmin tai sävelen valintaan ja vaihtoehdot ovat järjestyksessä (rytmit lyhimmästä pisimpään, nuotit lähimmästä kauimpaan), voidaan valintaa painottaa jompaankumpaan suuntaan yksinkertaisesti lisäämällä vakio vaihtoehdoista arpovaan funktioon. Markovin ketjut voidaan myös säilöä ohjelmassa taulukkorakenteeseen, jonka kautta niitä voidaan muokata suorituksenkin aikana. Markovin ketjut soveltuvat myös hyvin reaaliaikaiseen musikalisaatioon, koska niillä voi generoida nopeasti seuraavan sävelen edellisen perusteella ja silti muokata rakenteita edellä mainituilla tavoilla.

3.2 Generatiiviset kieliopit

Generatiivisia kielioppeja käytetään sekä algoritmiseen säveltämiseen että musiikin analyysiin. Generatiivisten kielioppien alkuperä on Noam Chomskyn vuonna 1957 kehittämissä lingvistisissä malleissa [14]. Tiivistettynä generatiivinen kielioppi on tapa muuttaa ketju korkean tason symboleita matalamman tason symboleiksi, kuten yksittäisiksi sanoiksi, käyttäen joukkoa sääntöjä. Joissain luokitteluissa, kuten Vicon ja kumppanien luokittelussa, L-systeemit kuuluvat myös generatiivisiin kielioppeihin [28]. Tässä luokittelussa ne käsitellään osana kaaosteoriaa luvussa 3.4.

Generatiivisissä kieliopeissa käsitellään merkkijonoja, joissa on yleensä kahta erityyppistä symbolia: välikkeitä (engl. *non-terminal*) ja päätesymboleita (engl. *terminal*). Päätesymboli on merkkijonossa esiintyvä symboli, jota ei enää laajenneta useammaksi eri symboliksi eikä muuteta. Välike on symboli, joka ei jää lopulliseen merkkijonoon. Se on välivaihe, joka muutetaan yhdeksi tai useammaksi välikkeeksi, päätesymboliksi tai näiden yhdistelmäksi generaatioprosessissa. Alkusymboli on välike, josta generointi aloitetaan.

Kaikkien kieliopin tuloksena olevien merkkijonojen joukkoa sanotaan formaaliksi kieleksi.
Määritellään esimerkiksi seuraava kielioppi:

$$S \to aB$$
$$B \to Bb$$
$$B \to c$$

Esimerkissä isot kirjaimet ovat välikkeitä ja pienet päätesymboleja. Aloitetaan merkkijonosta, joka on vain aloitusmerkki S. Sille on vain yksi sääntö, joten siitä tulee aB. Seuraavassa vaiheessa B voidaan korvata joko merkkijonolla Bb tai c. Jos valitaan ensimmäinen vaihtoehto, saadaan uusi välike, jolloin generointi jatkuu. Jos valitaan c, merkkijonossa on jäljellä vain päätesymboleja ja generointi loppuu ja saatu merkkijono on ac. Määritellyllä kieliopilla voidaan myös generoida merkkijono abc ja kaikki muutkin merkkijonot, jotka alkavat a, jonka jälkeen on mikä tahansa määrä kirjainta b ja päättyvät c.

Chomsky erottelee generatiiviset kieliopit neljään eri luokkaan sen mukaan, kuinka paljon ilmaisuvoimaa ja rajoitteita niillä on. Tätä luokittelua sanotaan Chomskyn hierarkiaksi. Yleisenä sääntönä voidaan sanoa, että mitä pienempi luokan numero on, sitä laajemmin sillä pystyy tekemään asioita ja sitä vähemmän rajoitteita siinä on, mutta sitä vaikeampi se on implementoida. Jokaiselle kielioppityypille on olemassa myös vastaavaan tulokseen pääsevä automaattityyppi.

Tyypin 0 kielioppi, toiselta nimeltään rajoittamaton kielioppi, ei sisällä mitään rajoituksia. Sekä oikealla että vasemmalla puolella sääntöjä saa olla rajoittamaton määrä sekä välikkeitä että päätesymboleja. Tyypin 0 kielioppia vastaa Turing-kone ja sen generatiivinen kapasiteetti on erittäin korkea. On mahdollista, että kieliopin monimutkaisuus on joissain tapauksissa rajaton ja lopputilaa ei koskaan saavuteta.

Tyypin 1 kielioppi eli yhteysherkkä kielioppi vastaa lineaarirajoitettua Turing-konetta eli Turing-konetta, jolla ei ole loputonta muistia. Vaikka sen generatiivinen kapasiteetti on tyyppiä 0 matalampi, se on edelleen hyvin korkea. Tyypin 1 kieliopin rajoitus edelliseen tyyppiin verrattuna on, että siinä ei sallita tilanteita, missä säännön oikealla puolella on vähemmän symboleita kuin vasemmalla.

Tyypin 2 kielioppi eli yhteydetön kielioppi on keskitasoa generatiiviselta kapasiteetiltaan. Sen säännöt ovat jo huomattavasti rajoittuneempia: niissä

saa vasemmalla puolella olla vain yksi välike eikä yhtään loppusymbolia. Oikea puoli ei ole rajoitettu. Vastaava automaatti on pinoautomaatti.

Tyypin 3 kielioppi on säännöllinen kielioppi, jota vastaava automaatti on äärellinen automaatti. Sen monimutkaisuus on lineaarinen. Vasemmalla puolella sääntöä ei saa olla kuin yksi välike ja oikealla puolella saa olla maksimissaan yksi loppusymboli ja yksi välike, tässä järjestyksessä.

Koska generatiivisten kielioppien kaltaisia hierarkkisia rakennelmia on nähtävissä useimmissa musiikkityyleissä, niitä on käytetty musiikin analyysissä ja generoinnissa jo kauan [28]. Sääntöjen laatiminen on tärkein vaihe, koska säännöt ohjaavat koko prosessia, mutta on myös tärkeää päättää, miten päätesymbolit muunnetaan musiikillisiksi elementeiksi. Säännöt voidaan kirjoittaa käsin käyttämällä musiikin teoriasta johdettuja tietoja, mutta nykyään ne voidaan vaihtoehtoisesti myös johtaa aineistosta. Generatiivisilla kieliopeilla tehdyissä tutkimuksissa on käytetty sekä aineistosta opittuja että ohjelmoijien käsin tekemiä sääntöjä, mutta aineistopohjainen menetelmä näyttäisi yleistyvän viimeaikaisissa tutkimuksissa [28]. Generatiivisia kielioppeja käytetäänkin usein olemassa olevien tyylien matkimiseen, koska säännöt voidaan johtaa materiaalista ja kielioppien ilmaisuvoima on riittävä.

Generatiivisella kieliopilla voidaan generoida koko sävellys rakenteesta yksittäisiin nuotteihin tai vain osa siitä. Generatiivisten kielioppien iso heikkous on, että ne ottavat huomioon vain edeltävät tapahtumat, mutta eivät samanaikaisesti tapahtuvia asioita, kuten samanaikaisesti soivia muita ääniä, mikä haittaa moniäänisen musiikin generointia. Rakenteen generointi on ehkä sopivimpia sovelluskohteita generatiivisille kieliopeille, koska edellä mainittu ongelma ei haittaa rakenteen generoinnissa.

Ensimmäisen kerran generatiivisia kielioppeja käytettiin musiikin melodian generointiin vuonna 1973 [49]. Niitä on käytetty myös esimerkiksi rumpuimprovisaatioihin [46], jazz-sooloihin [33] ja yleisemmin sekä rakenteen, rytmin että melodian generointiin [41]. Esimerkiksi Jon Gillickin, Kevin Tangin ja Robert M. Kellerin jazz-improvisaatiogeneraattorin kieliopissa on määritelty erilliset päätesymbolit esimerkiksi seuraaville asioille: nykyisen soinnun sävel, tauko, sointusävelestä puolisävelaskeleen päässä oleva sävel ja satunnainen sävel. Lisäksi on tapa esittää erilaisia ylös tai alas meneviä nuottikulkuja. Säännöt soolojen generointiin johdetaan lähdemateriaalista [33].

Generatiivisten kielioppien käyttämisessä musikalisaatioon on potentiaalisena ongelmana se, miten saada sävellysprosessia ohjattua niin, että datan ominaisuudet voidaan kuulla lopputuloksesta. Normaalissa tapauksessa kielioppi generoi lopullisen merkkijononsa ilman mitään ohjausta, jolloin data

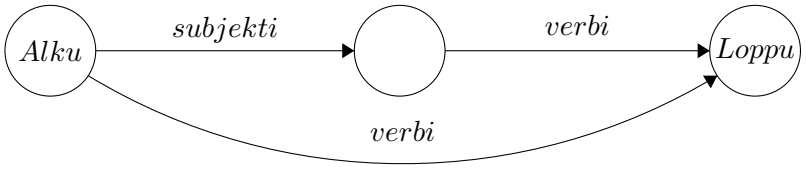
voi olla vaikea saada kuulumaan musiikissa. Tätä ongelmaa voitaisiin ehkä kiertää niin, että säveltäminen tehtäisiin kieliopilla, jossa monille välikkeille, joista voidaan johtaa useita erilaisia loppusymboleja, olisi joko datan mukaan erilaisiin vaihtoehtoihin painottuva tai jopa datan mukaan vaihtoehdon valitseva versio. Vaihtoehtoisesti kielioppeja voitaisiin käyttää johonkin sellaiseen osaan sävellysprosessista, johon ei haluta datan vaikuttavan, kuten esimerkiksi sointukulkujen valitsemiseen. Generatiivisista kieliopeista saadaan musiikillista materiaalia vasta, kun koko korvausprosessi on käyty loppuun ja kaikki välikkeet on korvattu, minkä vuoksi reaaliaikaiseen musikalisaatioon tämä menetelmä ei käy.

3.3 Siirtymäverkot

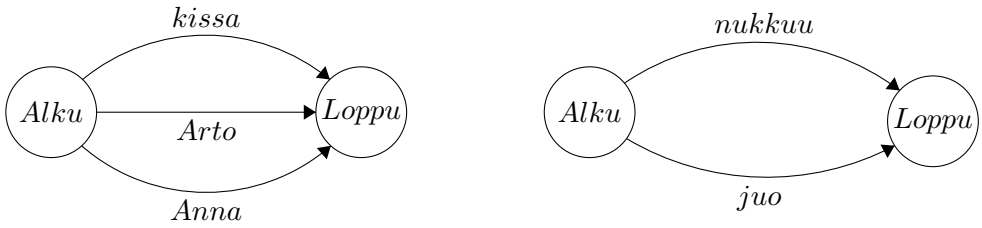
Siirtymäverkko on joukko tilakoneita, joiden viittaukset toisiinsa muodostavat verkkorakenteen. Sitä voidaan kuvata verkkona, jossa solmut edustavat tiloja ja kaaret edustavat siirtymiä. Käytännössä siirtymäverkko on vaihtoehtoinen formalismi jollekin generatiivisten kielioppien tyypeistä, riippuen verkon tyypistä. Siirtymäverkosta saadaan lopputulos vasta, kun se on käyty kokonaan läpi. Tämä ominaisuus erottaa sen esimerkiksi Markovin ketjuista, jotka generoivat uusia tiloja edellisten pohjalta jatkuvasti, mutta eivät välttämättä päädy mihinkään lopputilaan.

Verkon läpikäynti aloitetaan ensimmäisestä solmusta ja pyritään pääsemään lopetussolmuun, josta ei ole enää lisää kaaria eteenpäin. Siirtymät solmusta toiseen voivat olla joko päätesymboleita tai äärellisiä automaatteja, jotka toimivat kuin ohjelma, joka kutsuu aliohjelmia. Jokainen tilakone vastaa jotain vastaavan kieliopin välikettä. Jos kahden solmun välillä on useampi mahdollinen siirtymä, niistä valitaan yksi.

Otetaan esimerkiksi lauseita generoiva siirtymäverkko, joka on esitetty kuvassa 6. Tämä sama esimerkki olisi mahdollista muodostaa myös edellisessä



Kuva 6: Yksinkertaisia lauseita generoiva siirtymäverkko. Siirtymät vastaavat kuvan 7 verkkoja.



Kuva 7: Vasemmalla puolella verkko, johon siirrytään lause-verkon kaaresta subjekti ja oikealla verkko, johon siirrytään kaaresta verbi.

luvussa esitellyillä kieliopeilla. Tässä hyvin pelkistetyssä esimerkissä kuvan 6 verkon läpikäynti aloitetaan solmusta, jossa lukee *Alku*. Tästä siirrytään eteenpäin joko alla olevaa suoraa kaarta tai ylempää, välissä olevaan solmuun vievää kaarta pitkin. Jos mennään välissä olevaan solmuun, kaaren päällä lukee *subjekti*, jolloin siirrytään kuvan 7 vasemmanpuoleiseen verkkoon. Kuvan 7 vasemmanpuolisesta verkosta valitaan jokin poluista alkusolmusta loppusolmuun, vaikka "kissa". Tämän jälkeen aliverkon loppu on löydetty ja olemme kuvan 6 keskimmäisessä solmussa. Siitä eteenpäin vie kaari, jossa lukee *verbi*, jolloin siirrytään kuvan 7 oikeanpuoleiseen verkkoon. Tässä verkossa alkusolmusta loppusolmuun vie kaksi kaarta, valitaan "nukkuu". Saavuimme alkuperäisen verkon loppusolmuun, tai lopputilaan, ja generoitu lause on syntaktisesti järkevä lause: "kissa nukkuu". Samalla verkolla voidaan myös ohittaa välisolmu ja tehdä pelkästä verbistä koostuva lause, kuten "juo".

Jos verkossa on mahdollista olla ylemmän tason verkkoja alemman tason verkkojen kaarina (esimerkiksi kuvan 6 verkko kuvan 7 verkon kaarena), verkkoa sanotaan rekursiiviseksi siirtoverkoksi. Normaalissa siirtymäverkossa verkon pystyisi esittämään yhdellä verkolla niin, että kaarien verkot purettaisiin ylempään verkkoon, mutta rekursiiviselle siirtymäverkolle tätä ei voi tehdä. Toinen erikoistapaus on augmentoitu siirtymäverkko, jossa siirtymäverkkoon lisätään ominaisuudet lisätä tarkempia siirtymäohjeita ja ehdollisia hyppyjä. Augmentoidussa siirtymäverkossa verkolla on muistia, mihin se pystyy säilömään asioita, ja verkon siirtymissä on mahdollista olla ehtoja, jotka voivat riippua muistissa olevista arvoista.

Luvussa 2 käsitelty Emmy (tai EMI) on esimerkki merkittävästä säveltävästä ohjelmasta, joka käyttää siirtymäverkkoja. Emmy käyttää tietokannasta haettujen musiikin palojen rekombinointiin ja SPEAC-järjestelmän määrittämän sävellyksen rakenteen toteuttamiseen augmentoituja siirtymä-

verkkoja.

Eräs variantti siirtymäverkosta on petriverkko (engl. *Petri net*). Se on siirtymäverkon erikoistapaus, jota käytetään tapahtumapohjaisten prosessien simulointiin. Solmut ovat dataa, ehtoja ja tiloja tai toimintoja. Petriverkkoja ovat käyttäneet konesävellyksessä ainakin Goffredo Haus ja Alberto Sametti vuonna 1991 valmistuneessa sovelluksessaan nimeltä ScoreSynth [37]. Heidän sovelluksessaan petriverkon solmut ovat ”musiikkiobjekteja” ja toimintosolmut jonkinlaisia transformaatioita, joita musiikille voidaan tehdä, kuten esimerkiksi voimakkuudenvaihteluita. ScoreSynthin tarkoitus on toimia työkaluna, jossa sekä musiikin parametreja (korkeus, kesto, intensiteetti jne.) että niiden järjestystä voidaan muokata algoritmisesti sitomalla erilaisia muunnoksia toimintosolmuihin.

Siirtymäverkkojen hyödyt ja haitat niin algoritmiseen sävellykseen kuin musikalisaatioon sovellettuna ovat samat kuin mitä edellisessä luvussa esitellyillä kieliopeilla. Verrattuna joihinkin myöhemmin esiteltyihin konsepteihin kuten soluautomaatteihin, siirtymäverkkoja pystyy soveltamaan musiikin rakenteen ohjaamiseen hyvinkin erilaisissa tyylilajeissa. Vaikka siirtymäverkot ovat ilmaisuvoimaltaan identtisiä edellisessä luvussa esiteltyjen kielioppien kanssa, niiden erona voidaan pitää sitä, että siirtymäverkkojen esitystapa on lähtökohtaisesti graafinen ja helpompi hahmottaa joillekin ihmisille. Graafinen ja verkkoihin pohjautuva esitystapa musiikillisille konsepteille on havaittu hyväksi yleisesti käytetyissä ohjelmissa, kuten esimerkiksi ohjelmissa MAX [20] ja PureData [64].

3.4 Kaaosteoria

Kaaosteoria yleistyi 1980-luvulla, kun Edward N. Lorenzin ja Benoit Mandelbrotin [51] tutkimuksien tulokset tulivat yleiseen tietoon ja niihin kuuluvista konsepteista, perhosvaikutuksesta sekä fraktaaleista tuli pian suosittuja. Kaaosteoriaan kuuluu monimutkaisten systeemien toiminnan kuvaaminen, niiden attraktorit, eli joukot, joita systeemi lähestyy kun aikaa kuluu, ja monet itsesimilaariset (eli itsensä kanssa samanlaiset) rakennelmat, kuten fraktaalit. Eräs tutkimus viittaa siihen, että itsesimilaarisuus olisi klassisessa musiikissa usein esiintyvä ominaisuus [43].

Lindenmayer-systeemit eli L-systeemit käsitellään myös fraktaalien yhteydessä, sillä niitä käytetään usein kuvaamaan itsesimilaarisia järjestelmiä, vaikka botanisti Astrid Lindermayer kehitti ne alun perin vuonna 1968 kuvaamaan kasvien kasvuprosessia. Vaikka L-systeemit ovat sukua generatiivisille

kieliopeille, niissä on yksi ratkaiseva ero: generatiivisten kielioppien generoimat asiat ovat valmiita, kun kaikki korvausoperaatiot on tehty, jolloin tulos ei enää muutu, kun taas L-systeemeissä elementti yleensä korvataan muun muassa useilla uusilla, ketjua jatkavilla elementeillä, jolloin generointi ei koskaan lopu, mutta generoitujen asioiden määrä kasvaa nopeasti jokaisella kierroksella. L-systeemejä on erilaisia ja niitä voidaan luokitella seuraavien kriteerien mukaan: kontekstivapaat ja kontekstisensitiiviset, deterministiset ja stokastiset sekä parametriset ja epäparametriset.

Kaaosmenetelmiin kuuluvista kenties käytetyin on L-systeemit. L-systeemeillä pystyy generoimaan loputtomasti materiaalia. Kuvauksen L-systeemin tuottamasta datasta musiikkiin voi tehdä monella eri tavalla. Joskus käytetään kuvausta suoraan L-systeemin tuottamasta asiasta, joka on yleensä merkkijono, musiikillisiin elementteihin, joskus taas kuvausta jostain toisesta kuvauksesta musiikkiin. L-systeemit voidaan kuvata esimerkiksi kilpikonnagrafiikaksi sanotulla esitystavalla, joka on viivoilla piirretty graafinen representaatio, joka voidaan jatkokuvata musiikiksi. DuBois kirjoitti siitä, miten Lindenmayer-järjestelmistä saadaan erilaisilla formalismeilla monia mahdollisuuksia musiikin generointiin [25]. L-systeemit voivat tuottaa luonnostaan rakenteita, joissa on eri kohdissa samoja toistuvia elementtejä, kuten ihmisten säveltämässä musiikissakin on. L-systeemit sopivat myös hyvin musiikin rakenteen generointiin.

L-systeemeissä on musikalisaation suhteen sama ongelma kuin muissa tämän luokan menetelmissä: ne eivät ole kovin hyvin ohjattavissa kuvattavan datan perusteella. Ne ovat kuitenkin monikäyttöisiä, jolloin esimerkiksi yhdistettynä johonkin toiseen järjestelmään niitä voisi soveltaa musikalisaatioon käyttämällä niitä johonkin sellaiseen osaan, johon kuvattavan datan ei tarvitse vaikuttaa.

Yksi musiikin generoinnissa usein käytetty kaaosteorian alainen osa on ”fractional noise”. Termin alle kuuluu erilaisia kohinaksi luokiteltavia asioita, joita voidaan jaotella niiden spektrin tiheyden mukaan. Yleisin on valkoinen kohina, joka koostuu satunnaisista arvoista, joilla ei ole mitään relaatiota toisiinsa. Toinen ääripää on ruskea kohina, jossa jokaisen arvon on oltava edellisen arvon vieressä. Näiden ääripäiden välissä on pinkki kohina eli toiselta nimeltään $1/f$-kohina. Pinkin kohinan on havaittu olevan hyvää musiikin generointiin, sillä se ei ole liian ennalta-arvattavaa, mutta ei kuitenkaan täysin satunnaistakaan. Kohinaa on käytetty esimerkiksi suoraan säveltämiseen niin, että kohina on kuvattu äänenkorkeuksiksi ja rytmeiksi. Yleisin käyttötapaus on ollut melodian generointi [28]. Tätä lähestymistapaa käyttivät

tutkimuksessaan Voss ja Clarke jo vuonna 1978 [80]. Pinkillä kohinalla on saatu joitain melko onnistuneita tuloksia, vaikka puhtaasti näin generoidulta musiikilta puuttuu tavoitteellisuus ja rakenne.

Tämä tapa soveltunee huonosti musikalisaation käyttötarkoitukseen muuten kuin ehkä raakamateriaalin tuottajana, koska kohinafunktiota on vaikea ohjata muuttamaan toimintaansa datan attribuuttien mukaan, sillä se on satunnainen funktio eikä siis sovi minkään asian kuvaajaksi.

Kaoottisiksi laskettavia yhtälöitä tai yhtälöjärjestelmiä käytetään musiikin generoinnissa kuvaamalla niiden tulokset musiikin parametreihin. Riippuen tapauksesta yhtälön tuloksia saatetaan käyttää esimerkiksi dynamiikkojen, kestojen, nuottien välisten aikojen tai nuottien korkeuksien kontrollointiin. Arvaamattomasti käyttäytyvistä yhtälöistä esimerkiksi Pierre-François Verhulstin vuonna 1837 kehittämää populaatioiden kasvua kuvaavaa yhtälöä voidaan soveltaa tähän tarkoitukseen. Tällä menetelmällä musiikkia ovat tehneet ainakin Jeff Pressing [63], Rick Bidlack [9] sekä Jeremy Leach ja John Fitch [48]. Tässä lähestymistavassa näkisin musikalisaation kannalta samat ongelmat kuin esimerkiksi pinkin kohinan käyttämisessä, mutta yhtälöistä saa musikalisaatioon raakamateriaalia tai yhtälöitä voi tarvittaessa käyttää jonkin musiikillisen elementin tuottamiseen ja kuvata data jollain toisella.

3.5 Geneettiset algoritmit

Geneettiset algoritmit, joita sanotaan joskus myös evolutiivisiksi algoritmeiksi, perustuvat biologisen evoluution ideaan. Geneettisissä algoritmeissa ratkaisua valittuun ongelmaan etsitään generoimalla sukupolvia mahdollisia ratkaisuja, joita testataan jotain tehtävästä riippuvaa laatukriteeriä vasten.

Ennen geneettisen algoritmin soveltamista ongelmaan täytyy ratkaisuehdotuksille löytää jokin tapa kuvata ne sellaisessa muodossa, että niitä voidaan generoida ja tarkistaa automaattisesti ja ratkaisuissa mahdollisten osien on oltava tiedossa. Esimerkiksi jos haetaan ratkaisua siihen, mitkä jonkin järjestelmän kytkimistä pitää olla päällä ja mitkä pois päältä, että saataisiin haluttu tulos, voidaan ratkaisuehdotukset kuvata jonoina ykkösiä ja nollia ja ratkaisun merkistö olisi myös vain ykkösiä ja nollia. Jos haettava asia on esimerkiksi algoritmi jonkin asian ratkaisemiseen, voi merkistö olla jokin määritelty joukko ohjelmointikielen operaatioita. Merkistön pitää olla määritelty, jotta sen pohjalta saadaan generoitua alkupopulaatio ja jotta mutaatio-operaatiot muuntavat yksilöitä tapaukseen sopivilla tavoilla. Vaikka merkkijonoiksi kuvaaminen oli ensimmäinen käytetty esitystapa yksilöille,

nykyään käytössä on myös muita menetelmiä, kuten yksilöiden kuvaaminen puurakenteina.

Geneettisten algoritmien toiminta noudattaa yleensä seuraavia askelia:

1. Generoidaan satunnainen alkupopulaatio elementeistä, joita vastaus voisi sisältää.

2. Arvostellaan jokainen populaation yksilö valitun laatukriteerin mukaan. Jos jokin yksilö saa tarpeeksi korkeat pisteet (etukäteen valitun rajan suhteen), valitaan ja tulostetaan se ja lopetetaan ohjelman suoritus.

3. Parhaan arviointituloksen saaneet yksilöt valitaan seuraavan sukupolven tuottajiksi. Valintamenetelmiä on erilaisia ja niiden satunnaisuus vaihtelee, mutta kaikki vähintään painottavat valinnassa paremmin laatukriteerin täyttäviä yksilöitä.

4. Generoidaan uusi sukupolvi yksilöitä käyttäen rekombinaatiota ja mutaatioita. Rekombinaatioksi sanotaan sitä, kun uudet yksilöt generoidaan valitsemalla kaksi uuden sukupolven tuottajiksi valittua ja yhdistelemällä satunnaisesti tai jollain logiikalla niiden ominaisuuksia niin, että syntyy yksi tai useampi uusi yksilö. Prosessissa voi myös tapahtua mutaatioita, jolloin jälkeläinen saa ominaisuuden, jota ei ollut kummallakaan vanhemmista, eli jokin yksilön osa muutetaan satunnaiseksi toiseksi merkistöön kuuluvaksi osaksi. Siirrytään takaisin vaiheeseen 2.

Tätä menetelmää käyttävällä ohjelmalla voi olla myös muita lopetusehtoja kuin ratkaisun löytyminen. Suoritus voidaan lopettaa, jos etukäteen päätetty generaatioiden maksimimäärä tulee täyteen, jos ratkaisut alkavat muistuttaa liikaa toisiaan, tai jos ohjelmalta loppuvat resurssit, esimerkiksi muisti. Se, onko vaihtoehtoisia lopetusehtoja implementoitu, riippuu toteutuksesta. Yllä olevien askelten toteutuksessakin saattaa olla eroavaisuuksia eri koulukuntien ja toteutusten välillä, esimerkiksi joissain tapauksissa osa hyvin menestyneistä yksilöistä otetaan suoraan seuraavalle kierrokselle ilman muuntelua ja niin edelleen.

Geneettisiä algoritmeja sovelletaan usein haku- tai optimointiongelmiin erityisesti tapauksissa, joissa ongelman matemaattinen tarkastelu on jostain syystä vaikeaa. Ratkaistavan ongelman alakohtaista tietoa ei myöskään välttämättä tarvita enempää kuin mitä tarvitaan merkistön muodostamiseen, jotta sitä voidaan yrittää ratkaista geneettisellä ohjelmoinnilla.

Yksi ensimmäisiä esimerkkejä geneettisten algoritmien soveltamisesta musiikkiin oli Hornerin ja Goldbergin tutkimus vuonna 1991 [42]. Heidän ohjelmansa otti lähtökohdaksi jonkin sekvenssin nuotteja, tavoitteeksi toisen nuottisekvenssin ja lisäksi nuottimäärän, jonka aikana siirtymä tuli saada tehtyä. Näiden välille pyrittiin luomaan jatkumo geneettisellä algoritmilla. Laatukriteerissä arvioitiin tämänhetkisen nuottisekvenssin samankaltaisuutta tavoitesekvenssiin yksinkertaisella kaavalla. Tässä esimerkissä nuottien pituutta ei otettu ollenkaan huomioon.

Marquesin ja hänen ryhmänsä vuonna 2000 rakentama ohjelma tuotti geneettisellä algoritmilla kolmiäänisiä kappaleita [52]. Ohjelmassa sävelkorkeudet, nuottien soimaan jääminen ja tauot on kuvattu kokonaislukuina ja jokainen luku edustaa yhtä pitkää aikayksikköä. Arviointifunktio käyttää kolmea kriteeriä sävellyksen arviointiin: harmoniaa mittaamaan samanaikaisesti soivien äänien intervallien konsonanssia, sävyä mittaamaan sävelen sopivuutta valittuun sävellajiin ja melodiaa mittaamaan peräkkäisten intervallien sopivuutta. Tämänkaltainen monta eri ominaisuutta arvosteleva arviointifunktio on suhteellisen yleinen ratkaisu [28]. Muita esimerkkejä geneettisten algoritmien käytöstä sävellykseen ovat ohjelmoineet esimerkiksi Gartland [32], De Prisco [23] ja Birchfield [10].

Melomics on uudehko, geneettisiin algoritmeihin pohjaava idea sävellyksen automatisoimiseksi. Melomicsilla pyritään automatisoimaan koko ammattimainen sävellysprosessi [66]. Luvussa 2 esitelty Iamus käyttää melomics-lähestymistapaa. Perinteisen evoluution sijaan melomics käyttää niin sanottua evolutiivisen kehitysbiologian lähestymistapaa. Erona perinteisiin geneettisiin algoritmeihin on, että sen sijaan, että käytettäisiin suoraa kuvausta merkkijonoksi, käytetäänkin epäsuoria kuvauksia. Kuvaukset ovat abstraktioita kasvuprosesseista, jotka määrittelevät ratkaisun merkistön ja lopputuloksen välisen suhteen. Biologiassa vastaava ilmiö on se, miten samalla geneettisellä informaatiolla varustetut solut pystyvät kasvamaan ja eriytymään hyvin erilaisiin tehtäviin. Jos epäsuorasta kuvauksesta saadaan tehokas, voidaan kuvata huomattavasti perinteisiä geneettisiä ongelmia monimutkaisempia kokonaisuuksia. Iamuksen kanssa samalla tekniikalla on luotu myös Melomics 109, joka säveltää populaarimusiikkia [66].

Uskoisin geneettisten algoritmien soveltuvan hyvin musikalisaatioon, joskin ehkä paremmin osana jonkinlaista hybridijärjestelmää. Jos musiikki esimerkiksi generoitaisiin lyhyehköissä yksiköissä ja datan ominaisuudet muokkaisivat laatukriteeriä, voitaisiin datan ominaisuudet saada hyvinkin konkreettisesti vaikuttamaan valintaprosessiin ja kuulumaan musiikissa. Re-

aaliaikaiseen musikalisaatioon geneettiset algoritmit eivät sovi kovin hyvin, sillä laatukriteeriin tehdyt muutokset näkyvät kunnolla vasta joidenkin sukupolvien jälkeen. Vaikka data saataisiin kuuluviin jollain toisella tavalla esimerkiksi hybridijärjestelmässä, ongelmia aiheuttaisi silti se, että geneettisen algoritmin täytyy usein saada generoida useita sukupolvia, ennen kuin tulokset alkavat olla oikeasti musikaalisia.

3.6 Soluautomaatit

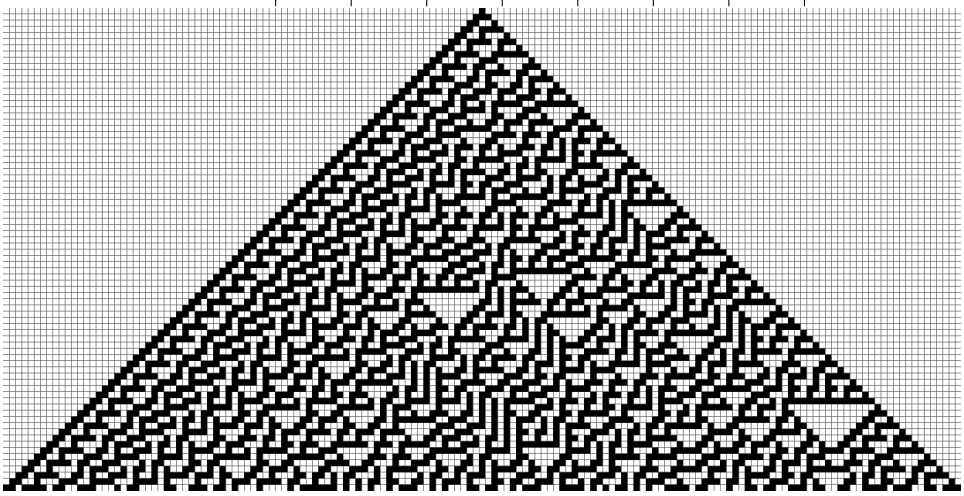
Soluautomaatit ovat itsenäisiä ja diskreettejä, tiettyjen sääntöjen mukaan operoivia yksiköitä, jotka on yleensä järjestetty jonkinlaiseen n-ulotteiseen taulukkoon tai vastaavaan rakenteeseen, joka voi joissain tapauksissa olla ääretön. Yksiulotteinen soluautomaatti on riippuvainen vain kahdesta naapuristaan ja sillä on yksinkertaisimmillaan vain kaksi eri tilaa, mutta yleisimmin käytetään kaksi- tai kolmiulotteisia automaatteja. Solun tila voi myös joissain tapauksissa olla reaaliluku, esimerkiksi 0 ja 1 väliltä, jolloin automaattia sanotaan sumeaksi automaatiksi.

Soluilla on äärellinen määrä ennalta määrättyjä tiloja, joissa se voi olla, ja jokainen solu on jossain tilassa. Solujen tila tarkastetaan jokaisella diskreetillä aika-askeleella, ja ne vaihtavat tilaansa siirtymäsääntöjen mukaan, jotka ottavat huomioon solun oman tilan sekä naapurisolujen tilat. Naapurisolut voivat joissain tapauksissa tarkoittaa vain vierekkäisiä soluja, mutta monimutkaisemmissa automaateissa saatetaan ottaa huomioon naapurit isommaltakin säteeltä. Yleensä jokainen taulukossa oleva solu noudattaa samoja sääntöjä.

Kuvassa 8 on esimerkki yksinkertaisesta yksiulotteisesta soluautomaatista siirtymäsääntöineen. Kuvassa on esitetty soluautomaatin tila jokaisen iteraation jälkeen niin, että seuraava rivi on se, mitä edellisestä rivistä tulee kun siihen on sovellettu taulukossa annettuja sääntöjä. Aika etenee kuvassa siis alaspäin, koska uusimmat kerrokset ovat alimpana. Ensimmäisellä rivillä on vain yksi solu tilassa ”1”. Seuraavalla kierroksella jokaisen taulukon solun kohdalla tarkistetaan sen vieressä olevat kaksi solua ja solu itse. Kuvassa ne ovat suoraan yläpuolella oleva solu ja vinottain olevat solut. Näiden tilat määräävät, tuleeko käsiteltävästä solusta seuraavalla iteraatiolla tyhjä vai ei. Kuvan 8 yläpuolella on kuvattu säännöt, joiden perusteella tila päätetään. Esimerkiksi jos kolme lähiruutua ovat kaikki tyhjiä eli niiden tilat ovat ”000”, ruutu jää tyhjäksi, mutta jos vain keskimmäinen ei ole tyhjä, eli tilat ovat ”010”, niiden alle tulee täytetty solu eli ”1”. Jokainen ruudukon kohta

käydään läpi samoja sääntöjä käyttämällä, yksi rivi kerrallaan.

naapurusto	111	110	101	100	011	010	001	000
seuraava tila	0	0	0	1	1	1	1	0



Kuva 8: Esimerkki kaksiulotteisesta soluautomaatista ja sen tilasta 75 kierroksen jälkeen. Kuvan yllä on esitetty säännöt, joiden mukaan se on rakennettu [56].

Ensimmäiset soluautomaatit kuvattiin Konrad Zusen, Stanislav Marcin Ulamin ja John von Neumannin 1950-luvun tutkimuksessa, jossa esiteltiin vaihtoehtoinen fysiikan mallinnus nimeltä ”calculating space” [84]. Von Neumannin kuoltua 1957 Arthur W. Burks jatkoi hänen työtään ja julkaisi vuonna 1966 von Neumannin paperin nimeltä ”Theory of self-reproducing automata” [13].

Kuuluisin yksittäinen soluautomaattisovellus on John Horton Conwayn 1970-luvulla Scientific American -lehden matemaattisten pelien osastolla julkaistu ”Game of Life” [31], josta tuli julkaisunsa jälkeen suosittu. Soluautomaattejä käytetään monilla eri tieteenaloilla mallintamaan ilmiöitä, joissa on monimutkaisia vaikutussuhteita ja toimijoita, joiden käytös riippuu toisistaan [28].

Soluautomaatteja on usein käytetty raakamateriaalin luomiseen, esimerkiksi ihmissäveltäjien työstettäväksi, eikä valmiiden sävellysten generointiin, koska niiden tuottamat asiat ovat mielenkiintoisia mutta usein epämusikaalisia [28]. Algoritmisessä sävellyksessä soluautomaatteja on käytetty jo 1980-luvun loppupuolelta asti. Ensimmäinen esimerkki, jossa soluautomaatteja käytettiin säveltämiseen ilman että säveltäminen tehdään yhteistyössä

ihmisen kanssa, oli Peter Beylssin artikkeli vuodelta 1989 [7].

Beyls on tehnyt useita sävellyskokeiluja soluautomaateilla, joista musikalisaation kannalta kiinnostavin on vuoden 1991 tutkimus, jossa Beyls antoi käyttäjälle välineet muokata kaksiulotteisia soluautomaatteja reaaliajassa [8]. Beyls kuvasi tutkimustaan niin, että hän halusi antaa käyttäjälle joustavan ja asteittaisen kontrollin systeemiin ilman suurta määrää kontrolliparametrejä. Jos soluautomaatteihin ilmestyi kiinnostavia kuvioita, haluttiin, että käyttäjä pystyy edesauttamaan niiden säilymistä niin, että seuraavat kuviot olisivat muunnelmia niistä. Käyttäjä pystyi esimerkiksi muokkaamaan parametria, joka määrää kuinka moni solu siirtyy seuraavalla kierroksella aktiiviseksi.

Eduardo Reck Miranda on tehnyt useita tutkimuksia soluautomaateilla säveltämisestä. Vuonna 1993 Miranda ryhtyi tekemään sävellysohjelmaa nimeltä CAMUS, jonka nimi tulee sanoista Cellular Automata Music, eli soluautomaattimusiikki [54]. CAMUS käyttää kahta eri soluautomaattityyppiä generoidakseen musiikkia: Conwayn Game of Lifeä ja David Griffeathin Demon Cyclic Spacea, joista ensimmäinen on vastuussa nuottien pituuksista ja kestoista ja jälkimmäinen instrumentaatiosta. CAMUS-ohjelmassa parametrit kuvataan soluautomaatista musiikkiin niin, että soluautomaatin solujen paikoille määrätään musiikilliset arvot ja musiikin tila määräytyy täten siitä, mitkä solut ovat milloinkin aktiivisia.

Soluautomaattien käyttö säveltämiseen poikkeaa monista muista esitellyistä tekniikoista siinä, että soluautomaattien luonne tekniikkana tekee tyylien matkimisesta tai lähdemateriaalin käytöstä ja siitä oppimisesta mahdotonta tai liian monimutkaista.

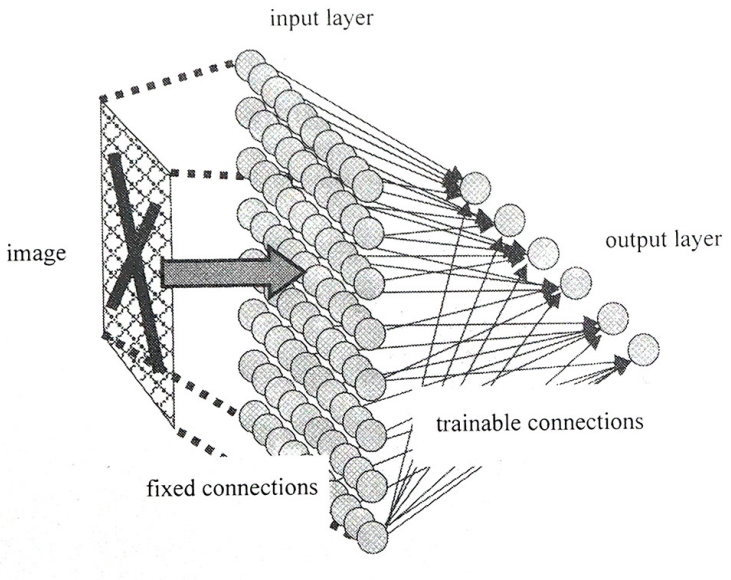
Soluautomaatit eivät sovellu datamusikalisaatiossa käytettäväksi niiden tyypillisessä käyttötapauksessa, jossa soluautomaatin annetaan generoida asioita ilman ohjausta. Kuitenkin Beylsin käyttämä järjestelmä, jossa soluautomaattien parametreja pystyy säätämään, saattaisi toimia. Jos datalähde laitettaisiin ohjaamaan soluautomaattien toimintaa ohjaavia parametreja ja soluautomaatti saataisiin toimimaan niin, että siitä ulos tuleva materiaali olisi musikaalista, tai vaihtoehtoisesti jokin toinen menetelmä muokkaisi sitä ennen kuin musiikki olisi lopullisessa muodossaan, soluautomaatit saattaisivat olla hyvä menetelmä musikalisaatiojärjestelmän muodostamiseen.

3.7 Neuroverkot

Neuroverkot ovat verkkoja, jotka koostuvat neuroneita mallintavista rakenteista. Niiden rakenne muistuttaa biologisia neuroverkkoja eli esimerkiksi

ihmisaivoja. Neuroverkot koostuvat neuroneista, joiden nimi tulee myös biologiasta. Biologiset neuronit ovat soluja, jotka koostuvat dendriiteistä, aksoneista (joita kutsutaan myös viejähaarakkeiksi), synapseista ja tumasta. Dendriitit ovat haarautumia solun keskeltä ja ne tuovat implusseja viereisistä soluista. Niiden päissä on synapseja, jotka kiinnittyvät viereisiin soluihin. Solun keskellä on tuma ja tumasta lähtee aksoni, jolla solu lähettää viestejä eteenpäin.

Keinotekoisissa neuroneissa on samankaltaiset elementit kuin niiden biologisissa vastineissakin: sisääntulo eli synapsi, jolla saa signaaleja joko ulkomaailmasta tai läheisiltä neuroneilta, summaaja, joka yhdistää kaikki sisään tulevat tiedot yhteen, ja aktivaatiofunktio, joka summaajan tuloksen perusteella määrittää, tuleeko neuronin aktivoitua. Neuroverkot koostuvat yleensä selkeistä kerroksista, joihin tulee sisään tietoa yhdestä suunnasta ja ulos toisesta. Nämä kerrokset voivat kytkeytyä joko suoraan syötteeseen tai toisiin kerroksiin. Lisäksi yksi kerros neuroneita toimii ulostuloina, joista luetaan neuroverkon laskennan tulos. Usein yksi kokonainen kerros neuroneita ottaa signaalia sisään, jonka jälkeen verkossa on nollasta moneen piilotettua kerrosta, joiden jälkeen on ulostulokerros. Perinteisissä verkoissa kaikki sisääntulon neuronit on kytketty kaikkiin seuraavan kerroksen neuroneihin ja



Kuva 9: Havainnollistava kuva yksinkertaisesta neuroverkosta, jota käytetään hahmontunnistukseen [56].

niin edelleen, mutta verkkotyyppejä on useita muitakin riippuen niiden soveltuvuudesta ongelmaan. Neuronien väliset kytkökset ja niiden painotukset ovat se osa verkosta, jota päivitetään lähdeaineistosta oppiessa. Kuvassa 9 on yksinkertainen neuroverkko, jota käytetään tunnistamaan asioita kuvista. Tässä esimerkkiverkossa on vain yksi kerros koulutettavia yhteyksiä.

Normaalisti neuroverkko opetetaan syöttämällä sen sisääntuloon paljon esimerkkejä asiasta, johon sitä halutaan myöhemmin käyttää. Koulutusvaiheessa neuronien väliset painokertoimet johdetaan aineistosta ja niitä käytetään myöhemmin siihen, että verkko joko tunnistaa tai pystyy generoimaan samantapaisia asioita kuin millä se koulutettiin. Neuroverkon käyttämiseen tarvitsee esimerkkejä ja yleensä esimerkkien tulee olla saman tyylisiä, jotta neuroverkko oppisi matkimaan kyseistä tyyliä, esimerkiksi musiikkityyliä [28].

Yksinkertaisimmat neuroverkot ovat yleensä eteenpäin syöttäviä verkoja, joissa signaali kulkee vain yhteen suuntaan ilman silmukoita. Muita tyyppejä ovat muun muassa takaisinkytkeytyvät verkot ja itseorganisoituva kartta. Takaisinkytkeytyvissä verkoissa neuronin syöte voi vaikuttaa silmukan kautta siihen itseensä. Takaisinkytkeytyvät verkot on suunniteltu mallintamaan ajallista dataa, jossa takaisinkytkentä vie edellisen aikayksikön informaatiota eteenpäin samalle päättäjäneuronille. Tämän avulla neuroni muistaa, mihin laskennan tilaan se edellisillä aika-askeleilla jäi. Itseorganisoituva kartta on neuroverkko, jossa neuronit muokkaavat tilaansa muistuttamaan syötettä ja kilpailevat siitä, kuka aktivoituu. Kun jokin neuroni voittaa kilpailun, myös sen naapureita muokataan, jolloin syntyy verkko, joka alkaa reagoimaan tietyllä tavalla tietylle syötteellä ja tunnistamaan erilaiset syötteet toisistaan.

Ensimmäisen kerran neuroverkoista kirjoitettiin jo vuonna 1943, mutta musiikkiin liittyvissä tehtävissä niitä käytettiin ensimmäisen kerran vasta 1970- ja 1980-luvuilla, kun niillä analysoitiin musiikkia ja mallinnettiin musiikkikognitioon liittyviä teorioita [75]. Ensimmäinen musiikkia säveltävä neuroverkko oli Peter Toddin vuonna 1989 ohjelmoima verkko, joka sävelsi yksiäänisiä melodioita tuottamalla listoja nuottien korkeuksista [74]. Sen jälkeen neuroverkkoja on käytetty moniin eri osa-alueisiin musiikin generoinnissa, kuten jazz-improvisaatioon, melodioiden generointiin, kontrapunktion generointiin sekä harmonisointiin eli sointujen tai säestyksien sovittamiseen valmiiseen melodiaan [28].

Vuonna 1991 Hild kollegoineen rakensi HARMONET-nimisen ohjelman [39]. HARMONET otti syötteenä melodialinjan ja sävelsi sen ympärille J. S. Bachin koraalien tyylisen neliäänisen sävellyksen. HARMONET oli takaisinkytkeytyvä neuroverkko, jolle syötettiin opetusvaiheessa 20 koraalia

duurissa ja 20 mollissa. HARMONET sisälsi myös sääntöpohjaisen osan, joka piti kirjaa musiikillisesta kontekstista, kuten seuraavan sävelen asemasta fraasissa ja siitä, mitä melodiassa ja harmoniassa on viimeksi tapahtunut. Nämä tiedot annettiin syötteenä neuroverkolle, joka antoi seuraavan sävelen.

Neuroverkkoja käytetään usein hybridijärjestelmissä toisen tyyppisten algoritmien kanssa yhdessä, esimerkiksi geneettisen algoritmin laatufunktiona. Mozer toteutti vuonna 1994 järjestelmän nimeltä CONCERT [55]. CONCERT ottaa huomioon yksittäisten intervallien sijaan korkeuden, harmonian ja rytmin ja antaa tulokseksi nuottien sijaan todennäköisyyksiä tiettyjen nuottien generoinnille. Tämän kaltaisen järjestelmän kanssa voisi olla hyvä käyttää jotain toista algoritmityyppiä suorittamaan nuottien lopullinen valinta.

Sävellyksessä neuroverkot pystyvät tuottamaan onnistunutta materiaalia lyhyillä aikaväleillä, mutta yksi niiden heikkouksista on, että isompien, koherenttien teosten tuottaminen on haastavaa. Jos neuroverkko haluttaisiin opettaa tuottamaan lyhyiden kappaleiden tai katkelmien sijaan isompia teoksia, myös opetukseen tarvittavan lähdemateriaalin määrä kasvaisi nopeasti. Neuroverkkojen etu, erityisesti verrattuna esimerkiksi kielioppeihin tai Markovin ketjuihin, on että neuroverkko pystyy tuottamaan sävelkulkuja, joita lähdemateriaalissa ei ollut, kunhan ne sopivat lähdemateriaalin tyyliin.

Tyypillisellä neuroverkolla voisi olla haastavaa tehdä musikalisaatiota, sillä sen jälkeen kun neuroverkko on koulutettu, sen toimintaan on useimmissa malleissa vaikea vaikuttaa. Musikalisaatiossa olisi kuitenkin tärkeää pystyä säätelemään jotenkin sävellysprosessia. Neuroverkkoja voitaisiin käyttää esimerkiksi yllä esitetyn CONCERT-järjestelmän tapaisesti, jossa neuroverkko antaisi vain ehdotuksia, joihin voitaisiin sitten soveltaa jotain toista logiikkaa. Toinen vaihtoehto olisi kouluttaa useita neuroverkkoja erilaisilla lähdemateriaaleilla ja päättää kuvattavan datan perusteella, mitä verkoista käytetään minkäkin osuuden generoimiseen. Tällainen ratkaisu saattaisi tosin olla haastava toteuttaa niin, että musiikin osat toimisivat yhteen.

3.8 Muut tekoälyt

Tekoäly on yläkäsite monille erilaisille menetelmille, joiden yhteinen tavoite on älykkään toiminnan automatisointi. On monia syitä miksi tämän kategorian rajat ovat vaikeita määrittää, muun muassa termin ”älykäs” tarkan määritelmän puuttuminen. Tässä luvussa käsitellään joitain musiikin generoinnin tapoja, jotka eivät kuuluneet mihinkään aiempaan lukuun, mutta

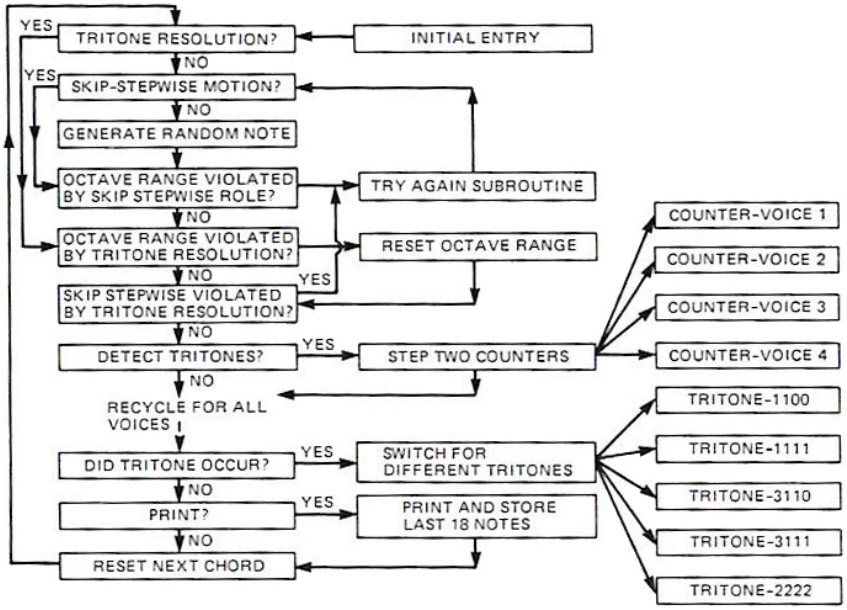
kuuluvat tekoälyn yläkäsitteen alle. Jotkin jo aiemmin käsitellyt termit, kuten generatiiviset kieliopit ja neuroverkot ovat myös tekoälymenetelmiä, vaikka ne on käsitelty omissa luvuissaan.

Monet menetelmät käyttävät hyödykseen heuristisia hakuja. Jos ongelman kuvauksessa on monia mahdollisia tiloja, niitä voidaan käsitellä tila- tai hakuavaruutena. Tätä avaruutta käydään lävitse, kun etsitään mahdollisia ratkaisuja. Yksinkertaisimmillaan tila-avaruutta voidaan käydä läpi käymällä kaikki mahdolliset polut tai kaikki polut, jotka vievät alusta loppuun (engl. *data-driven search*) tai lopusta alkuun (engl. *goal-driven search*), mutta useimmissa tapauksissa se on laskennallisesti liian raskasta ja aikaa vievää. Tällaisissa tapauksissa käytetään yleensä erilaisia heuristiikkoja. Heuristiikka viittaa tapaan valita, mihin tilaan seuraavaksi pitäisi siirtyä. Esimerkiksi kauppamatkustajan ongelmassa, jossa jokaisessa kartan kaupungissa täytyy käydä kerran, yksi mahdollinen heuristiikka on valita aina seuraavaksi tilaksi eli kaupungiksi lyhimmän etäisyyden päässä oleva kaupunki. Esimerkiksi Dannenbergin [22] säestyksiä soittavassa ohjelmassa heuristiikkoja käytetään siihen, että tietokoneen soittama säestys saadaan pysymään soittavan ihmisen perässä arvaamalla, missä kohdassa ihminen on nyt menossa. Reaaliaikaisessa järjestelmässä ei ole aikaa etsiä koko kirjoitetun sävellyksen sisällöstä.

Tuotantojärjestelmät (engl. *production systems*) ja sääntöpohjaiset järjestelmät (engl. *rule-based systems*) ovat järjestelmiä, joiden tarkoitus on tehdä jonkinlaisia päätöksiä. Tyypillisessä tuotantojärjestelmässä on kolme osaa: joukko sääntöjä, työmuisti tai tietokanta ja pitkäaikainen muisti. Tuotantojärjestelmän toiminnalla on kolme vaihetta: tunnista, ratkaise ja toimi.

Säännöissä on ehtoja ja toimintoja, joissa ehdot määräävät, mitkä toiminnot suoritetaan. Työmuistissa on järjestelmän ja ympäristön tämänhetkinen tila eli ne faktat, joiden perusteella sopivia toimintoehtopareja löydetään. Järjestelmä suorittaa tunnista-toimi-sykliä, jossa se tekee ne toiminnot, joiden ehto löytyy työmuistista. Asiantuntijajärjestelmä on tuotantojärjestelmän päälle rakennettu järjestelmä, joka käyttää aihekohtaista tietoa annetun ongelman ratkaisemiseen tai annettujen faktojen arviointiin.

Sääntöpohjaiset järjestelmät koostuvat säännöistä ja rajoitteista (engl. *constraints*). Säännöt ovat yleensä samanlaisia kuin tuotantojärjestelmissä. Rajoitteet ovat sääntöjä, jotka rajaavat hakuavaruutta ja voivat esimerkiksi antaa arvoille välejä, joiden sisällä niiden tulee pysyä. Useimmiten sääntöjä ja rajoitteita käytetään yhdessä niin, että sääntöjen pohjalta generoidaan jotain, jota sitten testataan rajoitteita vasten ja jos se ei toteuta rajoitteita,



Kuva 10: Illiac Suiten kolmannen osan kromaattisen, eli puolisävelaskelittain etenevän, etenevän musiikin generoineen koodin graafinen esitys [56].

se hylätään.

Ensimmäinen edellä mainitun kaltaisia periaatteita käyttänyt järjestelmä oli luvussa 2 mainittu ”Illiac Suite”. Sen säveltämiseen käytettiin yhtenä menetelmistä sitä, että generoitiin arvoja, joita testattiin erilaisilla rajoitteilla, ja ne, jotka eivät päässeet rajoitteista läpi, arvottiin uudelleen. Kuvassa 10 on esimerkki siitä, millaisia sääntöjä Illiac-järjestelmässä käytettiin. Myös Ebcioglun CHORAL-järjestelmä [26] ja Phon-Amnuaisukin ja hänen kollegoidensa Bach-harmonisaattori [60] käyttää sekä sääntöpohjaista lähestymistapaa että heuristiikkoja.

Ohjattu oppiminen on koneoppimisen muoto, jossa järjestelmälle opetetaan syöte-tulos-pareilla millainen tulos minkäkin tyyppisestä syötteestä pitäisi tulla. Tarkoituksena on saada aikaiseksi järjestelmä, joka pystyy opettamisen jälkeen päättelemään uusistakin syötteistä, mitä niistä pitäisi tulla tulokseksi. Vahvistusoppiminen on koneoppimisen laji, missä järjestelmä saa ympäristöltä positiivista tai negatiivista palautetta sen mukaan, miten se toimi, ja järjestelmä pyrkii maksimoimaan saadun positiivisen palautteen määrän. Ohjaamaton oppiminen toimii niin, että järjestelmä etsii aineistosta säännönmukaisuuksia ilman että opetusmateriaalia on valmiiksi merkitty.

Myös esimerkiksi aiemmin esitellyt neuroverkot ovat eräs koneoppimisen alalaji.

Erilaisten koneoppimismenetelmien käytöstä musiikin generointiin löytyy monia esimerkkejä, kuten Widmerin järjestelmä [83], joka rakentaa muut äänet annetun melodian ympärille. Järjestelmä oppii opetusaineistosta käyttäjän valmiiksi merkitsemien harmonisointien avulla yleisiä periaatteita, joita se sitten käyttää generoidessaan uusia harmonisointeja. Se arvostelee joitain musiikillisia elementtejä, kuten esimerkiksi jännitystä tai kontrastia, asteikolla matalasta hyvin korkeaan. Järjestelmä myös rakentaa hypoteeseja siitä, miksi jokin tietty kohta musiikista koetaan positiiviseksi. Käyttäjä joko vahvistaa tai kumoaa sen. Vahvistetut hypoteesit muutetaan säännöiksi, joita järjestelmä käyttää tulevien teosten luontiin.

Tapauskohtainen päättely (engl. *case-based reasoning*) on menetelmä, jossa ongelmia ratkaistaan neljän vaihteen kautta käyttäen jo tiedettyjä tai opittuja ratkaisuja:

1. Haetaan muistista tapaukset, joiden ratkaisua voisi soveltaa myös kysyttyyn tapaukseen.

2. Käytetään alustavaan ratkaisuun sen tapauksen ratkaisua, joka vastaa kaikista eniten kysyttyä tapausta.

3. Sovitetaan ja päivitetään valittua ratkaisua sopimaan paremmin juuri tähän tapaukseen.

4. Ratkaistu tapaus ja sitä varten päivitetty ratkaisu päivitetään tietokantaan tulevaa käyttöä varten, jolloin jokainen ratkaistu tapaus tekee järjestelmästä vähän monipuolisemman.

Tätä menetelmää on sovellettu algoritmiseen sävellykseen esimerkiksi Pereiran ja hänen kollegoidensa ohjelmassa nimeltä MusaCazUza [67]. MuzaCazUza tuottaa annettujen sointukulkujen pohjalta niihin soveltuvat melodiat. Ohjelmalla on tietokanta, jossa on melodiafraaseja yhdistettynä niiden pohjalla oleviin sointuasteisiin. Ohjelma tuottaa melodialinjan valitsemalla tietokannan pätkistä kuhunkin kohtaan mielestään sopivimman, jonka jälkeen se tarjoaa käyttäjälle mahdollisuuden tehdä melodiaan erilaisia transformaatioita. Tämän jälkeen uusi generoitu melodia voidaan tallentaa myös tietokantaan tulevaa käyttöä varten.

Heuristisia menetelmiä ja sääntöpohjaisia osia on useimmissa sävellysjärjestelmissä jonkinlaisena osana, myös niissä, jotka on esitelty luvussa 4.

Heuristiikat, rajoitukset ja säännöt käyvätkin osaksi musikalisaatiojärjestelmää hyvin, sillä heuristiikat voi alusta lähtien laatia niin, että ne ottavat yhtenä elementtinä huomioon jonkun tai joitain elementtejä datasta, jota halutaan musikalisoida.

Koneoppimisen käyttäminen musikalisaatioon on myös suhteellisen käytännöllistä, jos esimerkit tai tietokannat, joita ohjelma käyttää, tehdään niin, että ne ottavat huomioon datan ominaisuudet. Esimerkiksi yllä esitetyssä MusaCazUsa-melodiageneraattorissa tietokannassa olisi voinut olla melodian ja soinnun yhdistelmien lisäksi jokaisessa yhdistelmässä on tieto siitä, millaisten datan ominaisuuksien ollessa voimassa sitä tulee käyttää. Jos seurattava ominaisuus datassa olisi vaikka syke, melodiat voisivat olla luokiteltu niihin, joita käytetään, kun datassa on matala sykearvo ja niihin, joita käytetään, kun se on korkea.

4 Datamusikalisaatio

Datamusikalisaatio on vähän tutkittu konesäveltämisen sovelluskohde. Perinteiset tutkimukset algoritmisen musiikin alalla keskittyvät pääsääntöisesti siihen, kuinka esteettistä musiikkia saadaan tuotettua, tai siihen, kuinka hyvin jotain tyylilajia tai säveltäjää saadaan matkittua. Musiikkia on kyllä käytetty kuvaamaan asioita tai konsepteja, kuten tunteita, mutta suoraan dataa ja sen vaihteluita kuvaava musiikki on harvinaista. Automaattisesti sävelletystä dataa kuvaavasta musiikista löytyy joitain esimerkkejä internetistä, mutta niiden etsiminen on haastavaa, sillä proseduurille, jota tässä kirjoituksessa kutsun datamusikalisaatioksi, ei ole vakiintunutta nimeä.

Musikalisaatioesimerkkejä muualta

Joitain esimerkkejä musikalisaatiosta löytyy internetistä. Brian Foo on tehnyt datapohjaista musiikkia projektinimellä Data-Driven DJ [30]. Data-Driven DJ:n kappaleita varten Brian Foo tekee suhteellisen paljon manuaalista työtä, mutta kappaleet ovat pohjimmiltaan kuitenkin datamusikalisaation määritelmään käyviä. Hän etsii osittain manuaalisesti käytettävät datat ja jälkikäsittelee ja hioo kappaleita, mutta rungon on generoinut algoritmi sille annetun datan pohjalta. Hän on tehnyt tähän mennessä kymmenen kappaletta, joiden aiheisiin kuuluu esimerkiksi eri etnisyyksien esiintyminen Hollywood-elokuvissa tai Pekingin ilmansaastetilanne.

Paalasmaa ja hänen kollegansa tekivät vuonna 2012 musiikkiesityksen

esiintyjän biologisista signaaleista [57]. He käyttivät kehoon kiinnitettäviä mittalaitteita, ja esimerkiksi esiintyjän sykkeen tiheys vaikutti musiikin tempoon ja asento syntetisaattorin parametreihin, kuten nuottien siirtymään hiukan ylös tai alas.

Florian Alt ja hänen ryhmänsä tekivät vuonna 2010 musikalisaatiotutkimuksen, vaikka he käyttivätkin sanaa sonifikaatio, jossa käyttäjien saamat Skype-viestit muunnettiin lyhyiksi melodioiksi [1]. Tekijät mainitsevat motivaatiokseen sen, että viestinotifikaatioäänet voisivat antaa enemmän informaatiota kuin mitä ne nykyään antavat. Heidän mielestään niiden täysi potentiaali ei ole käytössä.

Skype-viestit muutettiin yksinkertaisiksi melodioiksi kuvaamalla niiden vokaalit eri säveliksi ja kuvaamalla yleisimmät hymiöt ja välimerkit soinnuiksi. Tutkimus tehtiin Saksassa, joten viestien kielenä toimi saksa. Viisi yleisintä vokaalia kuvattiin määrätyiksi säveliksi, jotka olivat aina samat. Konsonantit sen sijaan kuvattiin satunnaisiksi säveliksi, lukuun ottamatta loppukonsonantteja, joista yleisimmille oli määrätyt sävelet. Viesteissä käytetään pentatonista asteikkoa, jossa asteikkoon kuuluu viisi säveltä, jotka ovat duuri- tai molliasteikon 1., 2., 3., 5. ja 6. sävel. Jos järjestelmä päättelee, että tekstin viesti on positiivinen, se käyttää pentatonista duuriasteikkoa, ja negatiivisen tapauksessa pentatonista molliasteikkoa.

Skype-viestijärjestelmää arvioitiin käyttäjäkokeilla [1]. Noin seitsemälläkymmenellä koehenkilöllä tehty tutkimus sisälsi testejä siitä, ymmärtävätkö koehenkilöt viestien sävyn tai sen, ovatko ne kysymyksiä. Tilastollisesti merkittävä osa koehenkilöistä pystyi tulkitsemaan mitattuja asioita. Kokeissa myös havaittiin, että musiikkiharrastuneisuudella ei ollut merkitystä viestien musiikkiesityksen ymmärtämiselle. Noin neljäntoista henkilön kenttätutkimuksessa ei enää musikalisoitu koko viestiä vaan vain lauseet, joissa oli ryhmän määrittelemiä avainsanoja, jotka kertovat sävystä. Havaittiin, että käyttäjät lukivat nopeammin musikalisoidut viestit ja eri sävyisten viestien avaamisnopeudessa oli eroja. Käyttäjät lukivat positiiviset viestit nopeammin kuin negatiiviset ja kysymykset negatiivisiakin viestejä hitaammin.

4.1 Esimerkki 1: Unimusiikki

Unimusikalisaatio-projekti tehtiin yhteistyössä Beddit-nimisen yrityksen kanssa, joka valmistaa unianalyysilaitteistoa, ja siitä julkaistiin artikkeli ”Intelligent Data Analysis” -konferenssissa [78]. Projektin tarkoitus oli tehdä automaattisesti sävelletty kappale käyttäjän yhden yön liikettä, hengitystä,

sykettä ja univaiheita kuvaavasta datasta. Sävellyksen tuli kuvata pohjana olevaa dataa mutta olla myös esteettisesti miellyttävä. Kyseisestä datasta oli käyttäjille saatavilla jo havainnolliset visualisaatiot, joten musikalisaatio suunniteltiin vain täydentämään visualisaatioita.

Tässä projektissa käytetyt algoritmit ovat yksinkertaisia, todennäköisyyspohjaisia ratkaisuja. Ne on valittu siksi, että projektin päämäärä ei ole tehdä taiteellisesti perustelluinta ja monimutkaisinta algoritmeilla sävellettyä musiikkia, vaan onnistua kuvaamaan dataa. Yksinkertaiset menetelmät ovat helpommin hallittavissa datan muutoksien mukaan, joten niiden käyttäminen tässä on perusteltua.

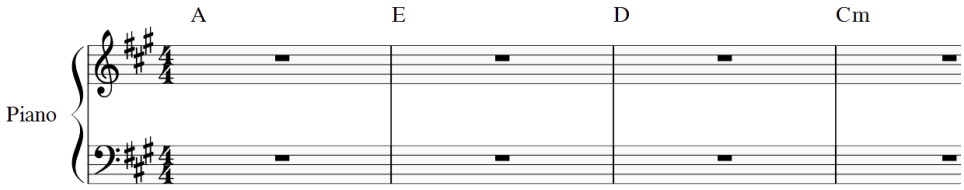
Musikalisaatioon käytettävä data kerättiin käyttäjän nukkuessa elektronisella paineanturilla, joka asetetaan käyttäjän patjan alle. Käyttäjän ollessa sensorin päällä anturi mittaa raakasignaalia, jonka se lähettää sensoriin kiinnitetylle pienelle Linux-pohjaiselle tietokoneelle. Se lähettää datan eteenpäin Bedditin palvelimelle. Tietokone lähettää sensorin signaalin lisäksi mittauksia nukkujan ympäristöstä kuten valoisuudesta ja lämpötilasta [58].

Esikäsittely

Palvelin jatkokäsittelee laitteen lähettämän datan ja etsii siitä sydämenlyönnit, liikkeen ja hengityksen. Näiden perusteella se laskee, missä univaiheessa käyttäjä on kulloinkin ollut. Datassa esiintyy neljää erilaista vaihetta: valveillaoloa, REM-unta, kevyttä unta ja syvää unta. Näissä mitatut arvot ovat matalimmillaan syvässä unessa, ja REM-unessa hengityksen ja sykkeen arvot ja vaihteluväli ovat korkeammalla kuin kevyessä unessa.

Palvelinpuolen esikäsittelyn jälkeen data siirretään varsinaiselle sävellysohjelmalle, joka on se osa järjestelmästä, jonka olen suunnitellut ja toteuttanut. Sävellysohjelma tuottaa yön datasta noin yhden minuutin musiikkia kahta nukuttua tuntia kohden. Ennen varsinaisen säveltämisen aloittamista ohjelma käy univaihedatan läpi ja poistaa sieltä hyvin lyhyet univaiheet, jotta musiikista tulisi selkeämpää eikä siinä olisi liian nopeita muutoksia.

Ennen säveltämisen aloittamista ohjelma valitsee asteikon tyypin sekä sävellajin. Vaihtoehtoina ovat kaikki duuriasteikot ja luonnolliset mollit. Normaalisti ohjelma valitsee duuriasteikon, mutta konfiguraatiota muuttamalla se generoi myös mollivoittoista musiikkia. Ohjelma myös arpoo, onko generoitava kappale tahtilajiltaan kolmi- vai nelijakoinen. Tahtilaji vaikuttaa musiikissa esimerkiksi siihen, kuinka pitkään pysytään samalla soinnulla, sillä sointuja arvotaan yksi jokaiselle tahdille.



Kuva 11: Katkelma sävellyksestä sointuvaiheen jälkeen.

Soinnut

Sävellyksen ensimmäinen vaihe on sointukulkujen generointi. Ohjelma valitsee Markov-ketjujen avulla jokaiselle tulevan sävellyksen tahdille soinnun. Vaikka erilaisia sointuja on olemassa paljon muitakin kuin ohjelman käyttämät, yksinkertaistamisen vuoksi ohjelma käyttää vain valitun asteikon sävelistä muodostettuja sointuja, joissa on täsmälleen kolme säveltä. Käsittelystä jäävät siis pois esimerkiksi nelisoinnut tai sävellajiin kuulumattomat soinnut, jotka ovat yleisiä useimmissa musiikkityyleissä.

Markovin ketjut, joilla käytettävät soinnut päätetään, ovat käsin kirjoitettuja, jotta on saatu järkevät sointukulut ilman aineistoa, sillä sopivaa aineistoa ei löytynyt. Sointujen esiintymistiheydet on myös valittu niin, että ne kuvastavat sointujen yleisyyttä kyseiseen sävellajiin pohjaavassa musiikissa. Esimerkiksi ensimmäisen asteen sointu, joka on sävellajinsa pohjasointu, on huomattavasti yleisempi kuin harvinainen ja riitasointuinen seitsemäs aste. Markovin ketjuja on myös yritetty saada suosimaan hyvin tyypillisiä sointukulkuja.

Markovin ketjut ottavat huomioon maksimissaan kaksi edellistä sointua, mutta osassa tapauksista kahden yhdistelmiä ei ole kirjoitettu, jolloin ketju ottaa huomioon vain edellisen soinnun. On myös tapauksia, joissa seuraava sointu on pakotettu olemaan jokin tietty sointu. Esimerkiksi riitasointuisen seitsemännen sointuasteen soinnun jälkeen tulee aina ensimmäisen asteen sointu, jotta riitasointu saadaan purettua. Kuvassa 11 nähdään esimerkkisävellyksen tila sointujen generoinnin jälkeen (kuva on havainnekuva, sillä ohjelma ei missään vaiheessa suoritustaan tulosta ihmisluettavia nuotteja).

Melodian sävelkorkeudet

Kun jokaiselle tahdille on määrätty sointu, ohjelma siirtyy generoimaan melodiaa. Ohjelman sisällä intervallit on luokiteltu suuriin ja pieniin intervalleihin. Priimi, pieni sekunti ja suuri sekunti eli nollasta yhteen sävelaskeleen kokoi-

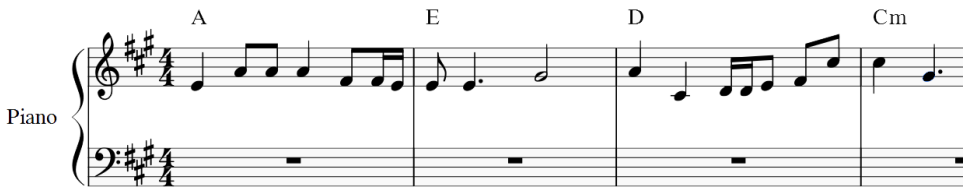
set intervallit ovat pieniä intervalleja, ja kaikki muut oktaaviin eli kuuteen sävelaskeleeseen asti ovat suuria intervalleja. Oktaavia isompia intervalleja on olemassa, mutta ne ovat harvinaisia, joten ne on jätetty huomiotta aiheen yksinkertaistamiseksi.

Melodian nuotit generoidaan myös Markovin ketjuilla. Ketjujen todennäköisyydet on asetettu niin, että kaikkien suurien intervallien todennäköisyys yhteensä on noin pienten intervallien todennäköisyys yhteensä. Koska suuria intervalleja on 20, kun lasketaan sekä ylös että alaspäin menevät, ja pieniä on yhteensä viisi, tämä tarkoittaa, että pienten intervallien esiintymistiheys on isompi. Tämä on tavoiteltu tilanne, sillä yleisesti musiikissa asteikkokulku tai samassa sävelessä pysyminen on varsinkin melodioissa isoja hyppyjä yleisempää. Esimerkiksi David Temperleyn kirjassaan käyttämässä kansanmusiikkitietokannassa nollan, yhden tai kahden puolisävelaskeleen intervallien esiintymistiheys on jopa yli 60 prosenttia kaikista intervalleista [72].

Jotkut tietyt siirtymät ovat melodian generoimisessa pakotettuja, sillä haluttiin estää tilanne, missä kaksi isoa hyppyä tehdään peräkkäin samaan suuntaan. Kaksinkertaiset hypyt samaan suuntaan ovat epätyypillisiä musiikissa, sillä ne tekevät musiikista epälaulettavaa – kontrapunktisäännöissä ne onkin kielletty. Lisäksi jos arvottu nuotti ei ole osa aikaisemmin päätettyä asteikkoa tai se on pahasti riitasoinnussa kyseisen tahdin soinnun kanssa, arvotaan melodiaan uusi nuotti sen tilalle.

Jos esimerkiksi edellinen intervalli on ollut yksi sävelaskel alas, Markovin ketju saattaa antaa seuraavaksi intervalliksi esimerkiksi puolikkaan sävelaskeleen eli pienen sekuntin ylös. Jos sävellaji on C-duuri ja edellinen sävel on ollut c, se tarkoittaisi että seuraavaksi säveleksi valikoituu cis. Cis ei kuulu C-duuriin, joten valinta hylätään ja suoritetaan uusi arvonta. Tällä kertaa ketjusta arvotaan intervalliksi kaksi sävelaskelta eli suuri terssi ylös. Seuraava sävel olisi tällöin e. E kuuluu asteikkoon, joten se hyväksytään. Lisäksi tarkistetaan, että uuden sävelen ja tällä hetkellä voimassa olevan soinnun pohjasävelen välinen etäisyys ei ole puolikas sävelaskel, kolme sävelaskelta tai viisi ja puoli sävelaskelta, sillä edellä mainitut ovat hyvin vahvasti dissonoivia yhdistelmiä.

Kappaleen sisäisen koherenssin vuoksi jokaiselle univaiheelle generoidaan oma teema. Teemaksi valitaan kunkin unenvaiheen ensimmäisen esiintymiskerran ensimmäiset kaksi tahtia, jotka tallennetaan myöhempää käyttöä varten. Poikkeus on valveillaolo, jonka kohdalle generoidaan hiljaisuutta. Univaiheiden melodiat generoidaan yksi vaihe kerrallaan ja myöhemmissä univaiheissa univaiheen melodia aloitetaan aikaisemmin tallennetulla, juuri



Kuva 12: Katkelma sävellystä melodian generoinnin jälkeen.

sen kyseisen vaiheen teemalla.

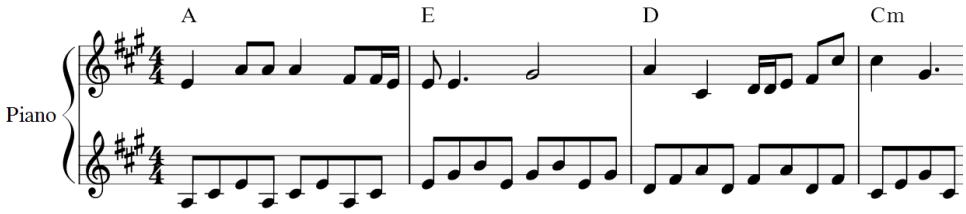
Melodian rytmi

Seuraava vaihe on rytmin generointi. Rytmiä generoidaan samanaikaisesti melodian kanssa niin, että kun seuraava melodian nuotti on päätetty, sille päätetään saman tien kesto. Myös rytmi generoidaan Markovin ketjuilla, joissa otetaan huomioon vain edellisen nuotin pituus. Ketjut on suunniteltu niin, että tietyn pituisen nuotin jälkeen on todennäköistä, että myös seuraava nuotti on saman pituinen. Todennäköisyydet suosivat myös sitä, että tällä hetkellä valittavasta nuotista seuraava nuotti osuisi iskulle, eli jos edellinen on 1,5 iskua pitkä, niin seuraavalla on suuri todennäköisyys olla 0,5 iskua. Jos tällä hetkellä käsittelyssä olevaa univaihetta on jäljellä alle 2,5 iskua, seuraavasta iskusta tulee koko univaiheen lopun pituinen. Tällä sekä vältetään tilanteita, joissa nuotti menisi univaiheen rajan yli, että annetaan univaiheen viimeiselle nuotille pituutta niin, että se tuntuu enemmän lopetukselta.

Rytmin generoinnissa otetaan myös huomioon nukkujan hengityksen tiheys. Jos siinä vaiheessa yötä, johon tällä hetkellä generoidaan musiikkia, käyttäjän hengitystiheys on ollut korkea, algoritmi lievästi suosii lyhyempiä nuotteja. Kuvassa 12 on esimerkki siitä, missä tilassa sävellys saattaisi tämän vaiheen jälkeen olla.

Säestys

Rytmin jälkeen generoidaan säestys melodialle. Myös säestyksen generointi seuraa univaiheita. Säestykselle on olemassa neljä erilaista funktiota, jotka generoivat sitä: yksi valveillaololle ja kolme eri univaiheille. Käyttäjän ollessa valveilla säestys koostuu soinnun pohjasävelellä olevista neljäsosista, joita tulee jokaiselle iskulle. REM-unen säestys on nopeahkoa liikettä, joka käyttää soinnun kaikkia säveliä, ja kevyen unen säestys liikkuu sointua ylös ja alas hiukan hitaammin kuin REM-unen säestys. Syvässä unessa säestys on hitaita



Kuva 13: Katkelma sävellyksestä missä kaikki soivat elementit on valittu.

nuotteja sen hetkisestä soinnusta. Kaikki säestystä generoivat funktiot käyttävät ainoastaan tämänhetkisen soinnun nuotteja, jotta vältyttäisiin siltä, että melodian ja säestyksen nuotit olisivat kovin pitkiä aikoja riitasoinnussa keskenään. Osassa univaiheita laitetaan pieni ylimääräinen korostus tahdin ensimmäiselle nuotille. Kuvassa 13 on ote sävellyksestä, joka kuvaa kevyttä unta. Esimerkissä on jo kaikki lopulliseen sävellykseen tulevat nuotit, vaikka kappale ei olekaan loppuun asti käsitelty.

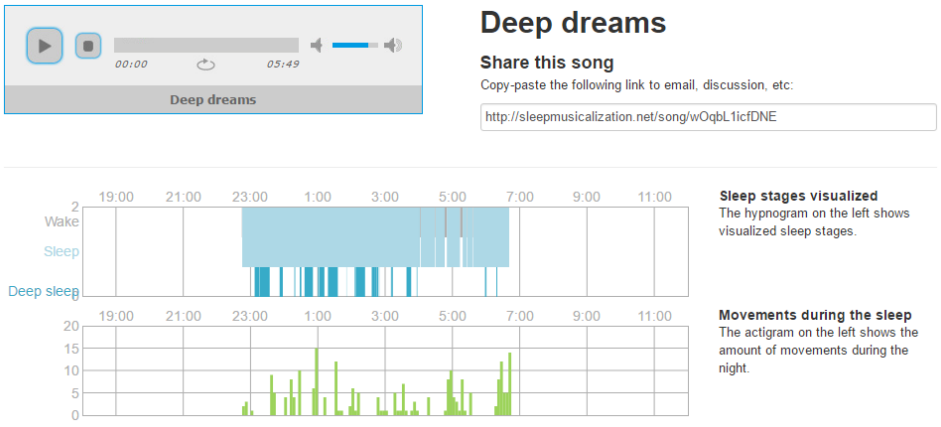
Aktigrammidata eli käyttäjän liikkeitä kuvaava data tiivistetään kuvaamaan liikkeiden määrää musiikin iskua kohden. Sitten musiikki käydään läpi ja nuottien äänenvoimakkuus asetetaan yksitellen vastaamaan liikettä. Jos musiikin kohdan kuvaamassa osassa yötä on ollut paljon liikettä, nuotin äänenvoimakkuutta nostetaan, ja vastaavasti jos liikettä on ollut vähän, sitä lasketaan.

Tempo

Kun itse nuotit ovat valmiita, asetetaan lopuksi musiikin tempo seuraamaan käyttäjän sykkeen tiheyttä. Ensin sykedataa kerrotaan vakiolla, jotta erot saadaan helpommin käsittelyssä näkyviin. Sen jälkeen datasta otetaan liukuva keskiarvo, jotta yksittäiset piikit datassa eivät tee isoja heittoja tempoon. Keskiarvon perusteella tempoa joko lasketaan tai nostetaan sen mukaan, nouseeko vai laskeeko sykkeen keskiarvo. Tämän jälkeen kappaletta muokataan niin, että hitaiden osien suhteellista pituutta vähennetään ja nopeiden pidennetään, jotta eri univaiheiden pituus musiikissa kuvaisi edelleen niiden kestoa käyttäjän unessa.

Äänisynteesi

Viimeinen vaihe kappaleen generoinnissa on äänitiedoston tuottaminen. Tähän asti sävellys on ollut vain ohjelman muistissa, mutta lopuksi se muutetaan muotoon, missä *Kunquat*-trakkeriohjelma [44] pystyy sen prosessoimaan.



Kuva 14: Loppukäyttäjän näkymä sivustolta missä sävellykset ovat kuultavissa.

Kunquat valittiin äänisynteesin tekoon, koska sen formaatti on kätevä generoida pala kerrallaan ja koska sille oli sopiva lisenssi ja tukea tarjolla. Kunquat käyttää tapahtumapohjaista lähestymistapaa musiikin generointiin, ja tässä projektissa käytetään eventtejä nuotin soittamiseen, nuotin sammuttamiseen, tempon muutokseen ja nuotin äänenvoimakkuuden säätöön. Yhdellä raidalla voi soida samanaikaisesti vain yksi nuotti, joten kappale generoidaan kahdessa erillisessä raidassa, joista toinen sisältää säestyksen ja toinen melodian nuotit. Kunquat ei käytä aikayksikkönä sekunteja vaan iskuja, joten kappale luodaan alun perinkin iskumetriikalla. Kunquat generoi sille annetuista tiedoista valmiin äänitiedoston.

Jotta Beddit-unimittarin omistaville loppukäyttäjille olisi mahdollista kuulla heidän unistaan generoidut kappaleet, koko systeemi toteutettiin verkkopalveluna. Käyttäjä pystyy kirjautumaan sivustolle ja antamaan sille luvan käyttää Beddit-tunnuksen dataa, ja hän saa valita yön, josta musiikki generoidaan. Käyttäjä saa kuunnellakseen web-käyttöliittymässä omasta unestaan generoidun kappaleen, jonka hän voi halutessaan jakaa esimerkiksi sosiaalisen median kautta. Kuvassa 14 on käyttäjän näkymä sivustolle. Käyttäjä saa nimetä kappaleen itse ja voi kuuntelun lisäksi katsoa visualisaation univaiheista ja liikkeen määrästä.

Unimusikalisaatio-ohjelmaa ei koskaan testattu millään mitattavalla tavalla. Se julkaistiin Bedditin oheispalveluna ja konferenssiartikkeli kirjoitettiin esittelemään idea. Artikkeli palkittiin konferenssissa visionäärisimmän artikkelin Frontier Prize -nimisellä palkinnolla. Subjektiivisesti arvioituna

musiikki voisi olla paljon parempaakin, mutta datan ominaisuudet, ja niistä erityisesti univaiheet, kuuluvat musiikissa hyvin. Musiikki on geneeristä, mutta suurimman osa ajasta melko miellyttävää.

4.2 Esimerkki 2: Reaaliaikainen geneerinen musikalisaatio

Toinen käsiteltävä esimerkki datamusikalisaatiosta on reaaliaikainen geneerinen musikalisaatio-ohjelma [77]. Toisin kuin edellisessä luvussa esitelty unimusikalisaatio, tämän ohjelman tavoite on toimia monilla erilaisilla datatyypeillä. Musikalisaatio voidaan toteuttaa tällä ohjelmalla niin, että lähteenä toimii mikä tahansa reaaliaikainen datavirta. Datan sisällöstä tulee tietää etukäteen sen verran, että sille voidaan kirjoittaa sopiva parametrisointi, eli päättää, mikä datan ominaisuus ohjaa mitäkin musiikin ominaisuutta. Varsinaisen datan ja sävellysalgoritmin väliin pitää kirjoittaa pieni ohjelma, joka saa muutokset datassa kuulumaan musiikissa.

Sävellysalgoritmin puoli on geneerinen, ja sitä voidaan soveltaa useisiin erilaisiin käyttötarkoituksiin. Tämän tutkielman esimerkissä musikalisoitavaksi on valittu keskustelu. Vaikka kuvatussa esimerkissä onkin käytetty valmiita lokitiedostoja keskustelusta, niin mikään ei estäisi käyttämästä musikalisaatio-ohjelmaa reaaliaikaisesti käynnissä olevaankin keskusteluun. Eräässä Helsingin yliopiston tapahtumassa ohjelmaa käytettiin myös musiikin generointiin riippukeinusta niin, että käyttäjät saivat mennä istumaan tai makaamaan riippukeinuun ja riippukeinun liike vaikutti musiikkiin, joka soi taustalla. Tämä oli toteutettu laittamalla riippukeinuun kiinni kiihtyvyysanturi, jonka datan perusteella muutettiin musiikin attribuutteja, kuten esimerkiksi tempoa.

Sävellysalgoritmia pystyy ohjaamaan parametrista riippuen joko täysin tai melkein reaaliaikaisesti muokkaamalla sen konfiguraatiotiedostoa. Konfiguraatiotiedostossa on määritelty erilaisia muuttujia, joita ohjelma käyttää generoidessaan musiikkia. Sävellysalgoritmia ohjaavissa muuttujissa on sekä korkean että matalan tason muuttujia. Korkeammalla tasolla toimittaessa voidaan käyttää konfiguraatiomuuttujia, kuten musiikin riitasointuisuus tai tempo, ja matalammalla tasolla voidaan tarvittaessa asettaa ulkopuolelta jopa se, millä soinnulla tällä hetkellä pitäisi olla, vaikka ohjelma osaa päättää soinnun itsekin.

Koska järjestelmä on suunniteltu reaaliaikaiseksi, käytettävien mekanismien pitää olla sellaisia, että ne pystyvät reagoimaan muutoksiin hyvin lyhyellä varoitusajalla. Tässäkin on käytetty samankaltaisia Markovin ketjuja

kuin unimusiikkisovelluksessa, koska ne täyttävät kyseisen kriteerin hyvin tuottaen kuitenkin riittävän esteettisen tuloksen. Ohjelma ei myöskään voi reaaliaikaisuuden takia suunnitella toimintaansa kovin paljoa etukäteen. Tämän vuoksi korkeamman tason rakenne on paljolti jätetty implementaatiossa huomiotta, eikä sävellykseen synny teemoja, säkeistöjä tai muita ihmisen tekemässä musiikissa yleisiä rakenteita. Osittain tästä johtuen tuotettu musiikki on jossain määrin päämäärättömän kuuloista, mutta koska sen on tarkoitus toimia taustamusiikkina, päämäärättömyys ei ole pelkästään negatiivinen ominaisuus. Silloin kun musiikkia ei ole erityisesti säädetty olemaan dissonoivaa, tuotettu musiikki on geneeristä eikä siinä ole paljoa elementtejä, mitkä erityisesti kiinnittäisivät huomiota.

Ohjelman toimintaperiaate

Musikalisaatio-ohjelma on kirjoitettu Pythonilla ja se käyttää rtmidi-pythonnimistä kirjastoa. Rtmidin avulla voi kirjoittaa MIDI-määritelmän mukaisia komentoja reaaliajassa [65].

Kun ohjelman käynnistää, se luo kaksi erillistä säiettä, joista toinen lukee käyttäjän tai jonkin toisen ohjelman kirjoittamia komentoja komentoriviltä ja toinen säveltää musiikin. Komentojen lukusäie muokkaa ohjelman konfiguraatiota sen mukaan, mitä komentoja ohjelmalle kulloinkin annetaan. Täydellinen listaus ohjelman tunnistamista komennoista ja niiden hyväksymistä arvoista löytyy liitteestä A. Konfiguraatiolle on oma tiedostonsa, josta luetaan ohjelman käynnistyessä konfiguraation aloitustila, mutta syötteen tai ohjelman itsensä tilaan tekemiä muutoksia ei tallenneta, vaikka ne vaikuttavatkin ohjelman suorituksen aikana.

Varsinaisessa sävellyssäikeessä ohjelma luo ensin yhteyden rtmidin kautta midi-portteihin ja asettaa ne oikeaan tilaan, jotta niihin voi kirjoittaa pythonilla. Sen jälkeen ohjelma asettaa joitain sävellykseen tarvittavia muuttujia, kuten asteikon, joka luodaan sen perusteella, mikä asteikkotyyppi on luettu konfiguraatiotiedostosta. Asteikkotyyppejä ovat duurin, harmonisen mollin ja melodisen mollin lisäksi kirkkosävellajit: doorinen, fryyginen, lyydinen, miksolyydinen ja lokrinen. Alussa myös alustetaan luokat, jotka käsittelevät ääniä, niin monelle äänelle kuin konfiguraatiossa on määritelty.

Kun alkuvalmistelut on tehty, ohjelma siirtyy silmukkaan, josta poistutaan vain, jos käyttäjä antaa tyhjän rivin, joka toimii lopetuskomentona ohjelmalle. Silmukka on kuvattu pseudokoodina algoritmissa 1. Koko pääsilmukka suoritetaan kerran jokaista 32-osanuottia kohden, mikä tarkoittaa myös, että lyhin nuottikesto, joita ohjelma pystyy käsittelemään, on

```
Hae alkuperäiset muuttujien arvot konfiguraatiotiedostosta.
Alusta komentoja lukeva säie.
Alusta tietorakenteet eri osille sävellystä.
while ei lopetussignaalia do
    Laske missä kohti fraasia ja tahtia ollaan.
    Päivitä sävellaji, tempo ja äänien lukumäärä jos muuttuneet.
    Haetaan ja lopetetaan loppuvat nuotit.
    Soita nuottiklusteri tai yksittäinen nuotti.
    Päivitä rummut.
    for jokaiselle äänelle do
        if äänellä ei ole nuottia soimassa then
            Nuotin korkeuden valinta (Algoritmi 2)
            Nuotin pituuden valinta (Algoritmi 3)
            Lisätään nuotti käynnissä oleviin nuotteihin.
    if sointu on ollut sama määritellyn ajan then
        Haetaan uusi sointu Markovin ketjusta.
        if satunnaisluku(0 − 1) < 0, 4 then
            lisäsävelet = [tyhjä lista]
            if harmonia = 8 then
                lisäsäveliin lisätään 6. aste ja vähennetty 7. aste
            if harmonia = 7 then lisäsäveliin lisätään 4. aste
            if harmonia = 6 then lisäsäveliin lisätään 7. aste
            if harmonia = 5 then lisäsäveliin lisätään 2. aste
            Sointuun lisätään satunnainen sävel lisäsävelistä.
```

Algoritmi 1: Pseudokoodiesitys ohjelman toiminnasta.

32-osanuotti.

Pääsilmukan alussa ohjelma tarkistaa joidenkin sellaisten muuttujien tilan, joiden muutokset otetaan huomioon vain silmukan alussa, koska ne vaikuttavat kaikkiin eri ääniin. Esimerkiksi asteikon tyypin muutos täytyy käsitellä silmukan alussa, koska silloin täytyy generoida uusi asteikko, jota kaikki eri äänet käyttävät. Jos tempo on muuttunut, se käsitellään myös silmukan alussa, sillä tempon muutos aiheuttaa sen, että kaikki sillä hetkellä soivien nuottien kestot pitää lyhentää tai pidentää uuteen tempoon sopiviksi. Suurin osa muuttujista, joita toinen säie muuttaa, katsotaan juuri ennen kuin niitä käytetään, jolloin muutokset kuuluvat heti. Jotkin katsotaan vain silmukan alussa ja jotkin, kuten esimerkiksi säestyskuvion vaihdot, vaihdetaan vain, kun edellinen kuvio on suoritettu loppuun.

Ohjelma hahmottaa musiikkia yksittäisten nuottien lisäksi pidempinä

yksikköinä: tahteina ja fraaseina. Tahtilaji on ohjelman nykyisessä versiossa sidottu neljään neljäsosaan. Tahtilaji kuuluu nykyisessä versiossa lähinnä siitä, että soitinten, joille on määritelty seurattavaksi jokin tietty kuvio, kuviot ovat neljä neljäsosanuottia pitkiä. Fraasi on normaalisti musiikissa yksi musiikillinen ajatus, kuin musiikillinen lause. Fraasin pituus on määritelty konfiguraatiossa ja se on myös muokattavissa suoritusaikana. Fraasi annetaan komentoriviltä tahdeissa, ei neljäsosissa. Nykyisessä versiossa fraasin ainoa merkitys on, että fraasin loppuun tulee yleensä pidempi nuotti tai tauko, koska pidempien rakenteiden, kuten fraasien, tekeminen on haastavaa hyvin responsiiviseen reaaliaikaiseen järjestelmään kuten tämä. Mikäli musiikin laatua haluttaisiin tulevissa versioissa parantaa, fraasien parempi huomioon ottaminen olisi hyvä tuleva kehityskohde.

Ohjelmassa voi antaa muutaman yksittäisen signaalin komentoriviltä, jos esimerkiksi halutaan antaa merkkiääni kuuntelijalle. Toinen näistä on komento, joka soittaa yksittäisen nuotin, jonka ei tarvitse sopia mihinkään muuhun soivaan materiaaliin ja jolle voidaan vapaasti määritellä kesto ja korkeus. Toinen vastaava merkkiääni on nuottiklusteri eli komento, jolla ohjelma soittaa parametrien mukaan tietyn pituisen ja tietyltä korkeudelta alkavan riitasoinnun, jossa on kymmenen säveltä pienen sekunnin päässä toisistaan. Näiden komentojen tullessa ne käsitellään ennen muita ääniä, jotta niiden vasteaika olisi mahdollisimman lyhyt.

Merkkiäänikomentojen käsittelyn jälkeen ohjelma lopettaa kaikki nuotit, joiden kesto on nyt tullut täyteen. Ohjelmalla on tietorakenne tällä hetkellä soiville äänille soittimissa, edellä mainituissa merkkiäänityyppisissä eventeissä ja rumpuraidoissa, jotka ovat monessa suhteessa erikoistapaus. Jokaiselta näistä haetaan lista äänistä, joiden pitäisi loppua nyt. MIDI-formaatissa jokainen äänen aloitus ja äänen lopetus ovat erillisiä komentoja, joten myös lopetuksista on huolehdittava. Jokaiselle päättyvälle sävelelle lähetetään lopetuskomento ennen kuin uusia säveliä aloitetaan. Jos äänenvoimakkuutta on muutettu, myös siitä lähetetään MIDI-komento ennen uusien sävelten valintaa.

Rumpuraita käsitellään erillään muista äänistä ja sille on oma luokkansa. Rummut ovat poikkeuksellinen soitin, sillä toisin kuin muita musiikillisia rooleja, eli basso-, melodia- ja säestysääniä, rumpuraitoja voi olla vain yksi ja se on aina sidottu toimimaan jonkin tietyn valmiin kaavan mukaan. Midi-speksissä rummuille tarkoitettujen soittokäskyjen äänenkorkeuden parametri säätää korkeuden sijaan sitä, millä rummulla kyseinen ääni soitetaan. Jotta rumpuraita olisi koherentti, erilaisia yleisiä rumpukuvioita on kirjattu val-

miiksi ja käyttäjä voi halutessaan vaihtaa kuviota. Jotta puolikkaita kuvioita ei syntyisi, etukäteen ohjelmoidut kuviot, joita on rummuilla ja säestysäänillä, suorittavat aina kesken olevan tahdin pituisen kuvion loppuun.

Sävellyksen jokainen erillinen ääniraita on oma itse ohjautuva yksikkönsä, joka ei välitä muista raidoista muulloin kuin tarkistaessaan säveltä dissonanssien varalta. Jokainen ääni on luokka, joka säilöö äänen tilan ja osaa hakea äänelle määrätyn soittimen konfiguraatiosta. Lukuun ottamatta rumpuraitaa, joka käsitellään täysin erillään, äänillä voi olla kolme erilaista roolia: melodia, taustaääni ja basso. Äänen rooli vaikuttaa siihen, miten sille valitaan rytmit ja sävelkorkeudet. Jokaiselle äänelle voi määritellä soittimen, äänen ylä- ja alarajan ja roolin ja niitä voi vaihtaa suoritusaikana.

Kun äänen edellinen sävel on loppunut, pääsilmukka kutsuu sen päivitysfunktiota. Uuden sävelen päättämisessä on monta vaihetta, jotka on kuvattu algoritmissa 2. Ensin tarkistetaan, onko asetus, joka määrää otetaanko fraasien lopetukset huomioon, päällä. Jos on ja fraasin loppu on alle kolmen neljäsosan päässä, laitetaan viimeiseksi ääneksi nykyisestä sävelestä seuraava tämän hetkiseen sointuun kuuluva sävel alaspäin tai tauko ja nuotin pituudeksi fraasin loppu.

Jos fraasi ei ole loppumassa, päätetään seuraavaksi seuraavan nuotin korkeus. Jos äänen tyyppi on basso, se soittaa aina äänelle määritellystä alarajasta seuraavaa soinnun pohjasäveltä. Rytmi on bassolla aina neljäsosia. Jos kyseessä on taustaääni, käytetään rumpujen tapaan valmiiksi määriteltyä kuviota, jossa on vain soinnun säveliä. Niitä on muutama erilainen, ja niiden välillä pystyy halutessaan vaihtamaan.

Ohjelmassa on olemassa asetus nimeltä harmonia, joka vaikuttaa paljon äänien valintalogiikkaan. Harmonia säätelee sitä, kuinka riitasointuista musiikki on. Harmonian vaikutus on esitetty lyhyempänä versiona algoritmissa 2. Harmonian asteikko on 1–10. Jos harmonia on alle 4, säestävät äänet lakkaavat noudattamasta niille määriteltyjä kuvioita ja tekevät samaa kuin melodiaäänet. Harmonian ollessa 1 kaikki äänet soittavat satunnaisia ääniä niin, että äänet, jotka ovat valitsemassa nuotin, joka ei ole riitasoinnussa muiden nuottien kanssa, arvotaan uusiksi. Harmonian ollessa 2 äänet soittavat täysin satunnaisia ääniä. Vaikka korkeudet arvottaisiin, äänet pysyvät aina määritellyissä rajoissaan. Harmonialla 3 sävelet ovat edelleen satunnaisia, mutta ne arvotaan uusiksi, jos ne eivät osu muodostettuun asteikkoon.

Kun harmonia-asetus on yli 3, taustaäänet siirtyvät takaisin soittamaan määriteltyjä kuvioita, mutta melodiaäänet jatkavat satunnaisten, mutta asteikkoon kuuluvien sävelien soittamista. Harmonian ollessa 5–7 melodia-

```
if fraasin loppu on lähellä then
    pituus = loput fraasista tai tauko
    while korkeus ei kuulu sointuun do
        korkeus = korkeus - 1
if harmonia = 1 then
    while korkeus ei ole riitasoinnussa do
        korkeus = satunnainen korkeus
if harmonia = 2 then  korkeus = satunnainen korkeus
if harmonia = 3 then
    korkeus = satunnainen korkeus oikeassa sävellajissa
if harmonia > 3 then
    if äänen tyyppi = melodia then
        if harmonia = 4 then
            korkeus = satunnainen korkeus oikeassa sävellajissa
        if harmonia > 4 and harmonia < 8 then
            while korkeus on yhden päässä soinnun pohjasävelestä or
            korkeus ei kuulu sävellajiin or korkeus on yhden päässä
            jonkun toisen äänen sävelestä do
                haetaan seuraavan intervalli Markovin ketjusta
                korkeus = korkeus + haettu intervalli
        if harmonia = 8 or harmonia = 9 then
            korkeus = jokin soinnun sävelistä (läheltä edellistä)
        if harmonia = 10 then
            korkeus = soinnun alin tai ylin sävel (läheltä edellistä)
    if äänen tyyppi = säestys then
        korkeus = seuraava sävel säestyskuviosta
    if äänen tyyppi = basso then
        korkeus = soinnun pohjasävel niin matalalta kuin intsrumentille
        annetut rajat sallivat
```

Algoritmi 2: Pseudokoodiesitys nuottien korkeuksien valintalogiikan toiminnasta. Selkeyden vuoksi jokaiseen haaraan ei ole kirjoitettu ”break”, mutta jos korkeus on valittu ja on poistuttu silmukasta, hypätään koodissa loppuun.

äänien seuraavan nuotin valinta toimii samoilla Markovin ketjuilla, joita käytettiin unimusikalisaatiossa. Melodiaan valittavista nuoteista tarkistetaan, että ne ovat osa asteikkoa, ne eivät ole puolisävelaskeleen päässä soinnun pohjasävelestä eivätkä ne ole puolisävelaskeleen päässä mistään jo soivasta tai päätetystä äänestä.

Korkeimmat harmonia-asetukset tekevät musiikista jo monotonista. Tasol-

```
if äänen tyyppi = melodia then
    if valittu korkeus kuuluu sointuun then
        pituus = seuraava pituus Markovin ketjusta
    else
        while pituus ≥ 1 do
            pituus = seuraava pituus Markovin ketjusta
if äänen tyyppi = säestys then
    pituus = seuraava pituus valitusta säestyskuviosta
if äänen tyyppi = basso then
    pituus = neljäsosanuotti
```

Algoritmi 3: Pseudokoodiesitys nuotin pituuden valinnasta.

la 8 harmonia pakottaa kaikki melodiaäänet pysymään vain soinnun sävelillä ja tasolla yhdeksän tai kymmenen ainoastaan soinnun alin ja ylin sävel ovat sallittuja. Tulevassa kehityksessä harmonian vaikutusta voisi tuoda vieläkin selvemmäksi ja tehdä tasojen eroja suuremmiksi. Nyt esimerkiksi tasot 9 ja 10 ovat vielä identtiset.

Kun melodiaäänelle on päätetty sävelkorkeus, päätetään sille vielä rytmi. Rytmin päättämisestä on pseudokoodi algoritmissa 3. Melodian rytmi toimii samanlaisilla Markovin ketjuilla kuin unimusikalisaatiossa. Tämä järjestelmä tukisi pienimmillään 32-osanuotteja, mutta rytmiä säätelevässä Markovin ketjuissa on käytetty minimissään 16-osanuotteja. Rytmille on myös olemassa konfiguraatiomuuttuja, joka painottaa valintaa lyhyempiin tai pidempiin nuotteihin. Basson rytmi on aina pelkkiä neljäsosanuotteja ja säestysäänet katsovat seuraavan rytmin niistä säestyskuvioista, jotka niille on määritelty.

Pääsilmukan lopuksi ohjelma tarkistaa, että jos nykyinen sointu on ollut voimassa sitä ohjaavan muuttujan osoittaman ajan, vaihdetaan sointua. Sointujen toiminta toimii samalla logiikalla ja samoilla todennäköisyyksillä kuin unimusikalisaatio-projektissa. Soinnun kolme säveltä tallennetaan, ja jos harmonia on tarpeeksi alhaalla, tallennetaan lisäksi neljäs sävel 40 prosentin todennäköisyydellä. Alle 8 harmonia mahdollistaa asteikon 6. sävelen tai vähennetyn 7. lisäyksen, alle 7 harmonia asteikon 4. sävelen, alle 6 harmonia 7. ja alle 5 2. sävelen lisäyksen. Tiivistettynä tämä on kuvattuna algoritmissa 1.

Mahdollisen sointumuutoksen jälkeen ohjelma odottaa ajan, joka on 32-osanuotin kesto nykyisessä tempossa miinus kaikkien eri silmukan vaiheiden käsittelemiseen mennyt aika. Koska midikomennot jäävät soimaan, siirrytään siis käytännössä seuraavalle 32-osaiskulle.

Chat-musikalisaatio

Järjestelmää testattiin käyttötapauksilla, joissa järjestelmää käytettiin kuvaamaan keskustelua. Kuten jokaisessa mahdollisessa musikalisoitavassa datalähteessä, tässäkin on tärkeää valita hyvin se, mikä lähdedatan attribuutti liikuttaa mitäkin musiikkiin liittyvää muuttujaa.

Osallistujien määrä on keskustelun oleellinen ominaisuus, joten sen haluttiin kuuluvan myös musikalisaatiossa. Keskustelijoiden määrä muuttaa musiikissa kuuluvien samanaikaisten äänien eli erillisten instrumenttien määrää niin, että yksi keskustelija on kolme ääntä, kaksi keskustelijaa on neljä ääntä ja kolme keskustelijaa on viisi ääntä. Ohjelma sallii äänien lisäämisen yhdeksään ääneen asti, mutta chat-musikalisaation demossa käytettiin vain maksimissaan viittä ääntä. Yhtä tai kahta ääntä ei haluttu käyttää, koska teos jää musiikillisesti ohueksi, jos siinä on vähän samanaikaisia ääniä, ja yksikään raidoista ei tuota samanaikaisesti useampaa kuin yhtä ääntä. Jokaiselle äänelle on oma instrumenttinsa, jotta ne erottuisivat paremmin toisistaan, vaikka teknisesti mikään ei estäkään käyttämästä samaa instrumenttia useammalla raidalla.

Tempo on hyvä kuvaaja keskustelun aktiivisuudelle [11], joten chat-musikalisaatiossakin käytettiin tätä hyväksi todettua kuvausta. Kun keskustelun viestien välissä on pitkä viive, musiikin tempo laskee, ja viestien tullessa tiheästi tempo nousee. Tempo vaihtelee välillä 20–117 iskua minuutissa.

Chat-musikalisaatiossa musiikki kuvaa myös keskustelun sävyä. Tämä toteutettiin niin, että käytetään SentiStrength-nimistä [73] algoritmia analysoimaan viestien tunnelatausta. Jotta musiikki ei muuttuisi liian yhtäkkiä, viimeisimmän kolmen viestin sävystä otetaan keskiarvo, jota käytetään ohjaamaan musiikkia. Kun viestien sävy muuttuu negatiivisemmaksi, kuten aggressiiviseksi tai surulliseksi, musiikista tulee dissonoivampaa ja mollivoittoisempaa, kun taas positiivisia tunteita sisältävät viestit tekevät musiikista duurivoittoista ja konsonoivaa.

Chat-musikalisaation arviointi

Chat-musikalisaatiota pyrittiin arvioimaan myös kokeellisesti. Arvioinnissa kysymykset keskittyivät kahteen aiheeseen: kuinka miellyttävää musiikki on ja kuinka hyvin musiikki kuvaa käyttäjälle näytettyä keskustelua. Arviointi tehtiin käyttäjäkokeilla, joihin osallistui 24 16–18-vuotiasta lukiolaista ei-musiikkipainotteisesta lukiosta. Kokeissa koehenkilö luki keskustelua ja kuuli samanaikaisesti musiikkia. Viestien loputtua koehenkilöltä kysyttiin

kysymyksiä liittyen musiikkiin ja musiikin ja keskustelun väliseen yhteyteen.

Vertailuun otettiin mukaan myös Skype-viestien musikalisointiin tehty järjestelmä [1]. Se on suunniteltu lyhyiden tekstien muuttamiseen melodioiksi, joten se käy vertailukohdaksi hyvin. Jokaiselle koehenkilölle näytettiin kolme erilaista keskustelua: keskustelu, jossa musiikki oli generoitu tällä ohjelmalla kyseisen keskustelun pohjalta, keskustelu, jossa musiikki oli generoitu tällä ohjelmalla, mutta eri keskustelun pohjalta, ja keskustelu, jossa musiikki oli generoitu Skype-musiikki-ohjelmalla. Kokeeseen valitut näytteet olivat kaikki saman pituisia ja ne näytettiin eri järjestyksessä eri koehenkilöille, jotta esimerkiksi koehenkilön väsyminen ei vaikuttaisi tuloksiin.

Kokeen aluksi koehenkilöille selitettiin kysymykset, joita heiltä tultaisiin kysymään eri keskustelujen lopuksi. Koehenkilöille ei kuitenkaan kerrottu mitään siitä, mitkä asiat musiikissa vastaisivat mitäkin keskustelun elementtiä. Jokaisen keskustelun jälkeen koehenkilöltä kysyttiin viisiportaisella Likert-asteikolla, kuinka miellyttävää musiikki oli ja kuinka hyvin se vastasi henkilön lukemaan keskustelua.

Tulokset viittaisivat siihen, että kumpikin musikalisaatioalgoritmi pärjäsi lähes yhtä hyvin. Tulokset ovat nähtävillä taulukossa 1. Vastausten perusteella tässä tutkielmassa esitelty musikalisaatioalgoritmi sai marginaalisesti paremmat arviot musiikin kiinnostavuudessa ja sen sopivuudessa keskusteluun, mutta vertailukohtana ollut järjestelmä sai hiukan paremmat arviot miellyttävyydessä, luonnollisuudessa ja keskustelun kuvaamisessa. Nämä erot eivät kuitenkaan olleet tilastollisesti merkittäviä. Huomionarvoista on, että monien kysymysten keskiarvot olivat lähellä kolmea, joka tarkoittaa neutraalia suhtautumista, eli musiikkia voisi parantaa vielä paljon.

Käyttäjäkokeen tuloksista nähdään, että vaikka koehenkilöille ei ollut selitetty periaatteita musiikin taustalla, tekstin pohjalta tehdyt musikalisaatiot saivat parempia tuloksia kuin eri tekstistä generoitu musikalisaatio, eli vaikuttaisi siltä, että valitut kuvaukset tekstin attribuuteille ovat ymmärrettäviä ilman esittelyä. Lisähuomiona tuloksista voidaan päätellä, että musiikki, joka tuntuu sopivan tekstiin hyvin, on myös koehenkilöiden mielestä miellyttävämpää.

Tässä arvioinnissa koehenkilöiden määrä oli kohtuullisen pieni ja monet mittaukset tuottivat niin pieniä eroja, että ne eivät ole tilastollisesti merkitseviä, joten lisää arviointia aiheesta tarvitaan, jotta saadaan varmempia tuloksia. Subjektiivisesti katsottuna ohjelma tuottaa tietyillä asetuksilla sellaista musiikkia, johon ei välttämättä jaksa keskittyä, mutta jota kestää kuunnella taustamusiikkina. Ohjelmalla on paljon erilaisia asetuksia, joten sillä voisi

Kysymys	Tämän paperin musikalisaatio kesk. (var.)	Skypeviestien musikalisaatio [1] kesk. (var.)	signif.	Eri keskustelun musikalisaatio kesk. (var.)	signif.
K1. Mielestäni musiikki oli miellyttävää	3.32 (1.56)	3.71 (1.21)	e.m.	2.31 (0.90)	$p < 0.05$
K2. Musiikki oli kiinnostavaa	3.86 (0.98)	3.43 (1.16)	e.m.	2.29 (1.26)	$p < 0.05$
K3. Musiikki kuulosti luonnolliselle	2.95 (1.57)	3.05 (1.15)	e.m.	1.94 (0.86)	$p < 0.01$
K4. Musiikin kuuleminen teki keskustelun seuraamisesta vaikeampaa	2.68 (2.42)	2.67 (1.93)	e.m.	3.06 (1.80)	e.m.
K5. Musiikki kuvasi keskustelua	3.09 (1.13)	3.38 (1.55)	e.m.	2.44 (1.60)	$p < 0.1$
K6. Musiikki sopi yhteen keskustelun sisällön kanssa	3.14 (1.46)	2.86 (1.13)	e.m.	2.26 (1.85)	e.m.

Taulukko 1: Keskiarvot ja varianssit kokeessa olleelle kolmelle eri musikalisaatiolle ja tilastollinen merkittävyys verrattaessa tämän tutkielman esittelemään musikalisaatiotapaan (e.m. = ei merkityksellinen, signif. = signifikanssi).

tehdä monenlaisia eri datalähteiden musikalisaatioita ja käyttäjäkokeita.

5 Yhteenveto ja pohdinta

Tässä tutkielmassa käsiteltiin datamusikalisaation syntyyn liittyvien alojen historiaa. Aikaisemmissa luvuissa käytiin läpi historiallisia esimerkkejä musiikin generoinnista ja kuinka pitkälle algoritminen säveltäminen on edennyt. Datamusikalisaation kehittyminen ei olisi ollut mahdollista ilman sävellysmetodien aiempaa kehitystä. Tämän jälkeen esiteltiin erilaisia menetelmiä musiikin generointiin, kuten generatiivisia kielioppeja ja soluautomaatteja ja niiden ominaisuuksia ja soveltuvuutta datamusikalisaatioon. Neljännessä luvussa käsiteltiin sovelluksia, jotka tuottavat musiikkia datan pohjalta käyttäen aikaisemmin esiteltyjä menetelmiä, ja niillä saavutettuja tuloksia.

Ennakkoasenteet algortimista säveltämistä vastaan

Jotkut pitävät luovuutta viimeisenä alueena, jossa ihminen on parempi ja tietokoneiden ei pitäisi pärjätä. Jopa shakkimestari Garry Kasparov, joka hävisi vuonna 1997 supertietokoneelle, sanoi: ”Tietokoneilla on valtava rooli yhteiskunnassa. Ne ovat kaikkialla. Mutta on olemassa rajoja, joita tietokoneet eivät saa ylittää. Ne eivät saa laajentua ihmisen luovuuden alueelle. Se uhkaisi ihmisen kontrollia sellaisilla aloilla kuin taide, kirjallisuus ja musiikki.” [56]. Tämä sitaatti ei ole ainoa laatuaan, mutta osoittaa, minkä tyyppistä vastustusta laskennallista luovuutta vastaan on.

David Cope viittaa ihmisten laskennallista luovuutta vastaan olevaan ennakkoasenteeseen jo 1996 kirjoitetussa kirjassaan [16]. Hän kuitenkin muotoilee asian eri tavalla ja toteaa, että ”tietokoneet eivät sävellä”. Hänen mielestään tietokoneet eivät ole meidän kilpailijoitamme luovuudessa vaan orjia, ja ohjelmoija, joka rakentaa säveltävän algoritmin, on tässä tapauksessa säveltäjä. Hän argumentoi, että tässä, kuten muissakin taiteellisissa asioissa, arviointikriteeri pitäisi olla taiteellinen arvo eikä ideologia. Niin säveltäviä ohjelmia tekevät ohjelmoijat kuin perinteiset säveltäjätkin pitäisi nähdä säveltäjinä.

Osittain syy tähän saattaa kuitenkin olla ihmisten tavassa tulkita musiikkia. Tutkimuksissa on havaittu, että FMRI-koneella mitattaessa ihminen reagoi eri tavalla koneen ja ihmisen säveltämiin sävellyksiin. Kun ihminen kuuntelee ihmisen säveltämää musiikkia, aivoissa aktivoituu alue, jonka oletetaan olevan kytköksissä toisen ihmisen aikeiden ennustamiseen esimerkiksi sosiaalisissa tilanteissa, mutta tämä sama alue ei aktivoidu kuunnellessa koneen säveltämää musiikkia [71]. Kyseisessä kokeessa koehenkilöille kerrottiin ennen musiikin kuuntelua, onko seuraava esimerkki koneen vai ihmisen säveltämä. On mahdollista, että esimerkiksi tämä ero vaikuttaa siihen, miksi ihmiset pitävät ihmisten säveltämää musiikkia parempana kuin koneiden säveltämää.

On myös selvää, ettei musiikki ja sen vaikutus ihmisten tunteisiin ole mitenkään tutkimuksen yläpuolella oleva maaginen voima. Musiikkikognitiota tutkimalla on saatu selville, että erilaisten odotuksien luominen musiikissa ja niiden täyttäminen tai täyttämättä jättäminen saa aikaiseksi luotettavasti tiettyjä tunteita ihmisissä [4]. Tätä käytetään taitavasti hyväksi esimerkiksi elokuvien tai pelien taustaraidoissa, ja koska se on luultavasti mahdollista kirjoittaa jonkinlaisiksi säännöiksi, algoritmeilla sävelletyt kappaleet voisivat käyttää samoja periaatteita.

Mahdollisia tulevia sovelluskohteita

Musikalisaatiolle on paljon potentiaalisia käyttökohteita, joita ei ole tutkittu vielä ollenkaan. Ehdotan tässä muutamia, mutta tämä ei ole mitenkään kattava lista. Tutkimuksissa on havaittu, että iloinen ja surullinen musiikki herättävät ihmisissä tunteiden lisäksi erilaisia muutoksia esimerkiksi ihon lämpötilaan ja muihin ihmisen tietoisessa hallinnassa olemattomiin prosesseihin [50] ja että positiivinen musiikki voi esimerkiksi ehkäistä aggressiota koehenkilöissä [47]. Musikalisaatiota voisi soveltaa internetkeskusteluihin myös esimerkiksi niin, että kirjoitettu mutta vielä lähettämätön teksti vaikuttaisi musiikkiin. Jos esimerkiksi negatiivinen tai aggressiivinen viestittely tekisi taustamusiikista dissonoivaa ja kaoottista, voisiko tällainen palaute auttaa ihmisiä käyttäytymään paremmin verkossa?

On myös havaittu, että musiikilla voi olla selvä rauhoittava vaikutus. Harlingin ja hänen kollegoidensa tutkimuksessa pediatrisessa sairaalassa hoidetut lapset kokivat vähemmän stressiä ja kipua, kun heille soitettiin rauhoittavaa musiikkia [36]. Musiikin hyödyt on havaittu myös tutkimuksissa koskien unta ja sen laatua. Tutkimuksessa huomattiin, että rauhoittavan musiikin kuuntelu vähensi univaikeuksia tutkittujen opiskelijoiden keskuudessa [35]. Tässä voisi olla mahdollinen sovelluskohde myös datamusikalisaatiolle. Stressin lievitykseen käytetty musiikki voisi esimerkiksi reagoida potilaiden biologisiin stressisignaaleihin, ja potilaan kuulema musiikki voisi muuttua rauhallisemmaksi sen mukaan, kuinka stressaantunut potilas on. Tai univaikeuksista kärsivän ihmisen radio voisi soittaa nukahtaessa rauhallista musiikkia, joka muuttuisi pirteämmäksi ja äänekkäämmäksi, kun nukkuja on unenvaiheissa, mistä on hyvä herätä, kuvaten samankaltaisia muuttujia datasta kuin unimusikalisaatio-projektissa.

Musikalisaatiolla voi tehdä myös taide-installaatioita. Esimerkiksi Paalasmaan ja hänen kollegoidensa paperissa esitellään biosignaalien käyttöä musiikkiesityksessä [57]. Muitakin mahdollisia datalähteitä, joita voisi käyttää taiteeseen, on varmasti paljon. Nosteessa oleva virtuaalitodellisuus- ja peliala voisi myös hyödyntää musikalisaatiota esimerkiksi linkittämällä jotkin pelaajan teot muutoksiin musiikissa.

Tutkimuksissa on havaittu, että sopivasta musiikista esimerkiksi kaupan taustamusiikkina voi olla paljon apua brändin luomisessa ja erityisesti tunnesiteen luomisessa brändiin [6]. Kun tiedetään, että musiikilla voi olla esimerkiksi brändiä vahvistava vaikutus, voitaisiinko musikalisaatiota käyttää siihen, että pyritään luomaan tunneside musiikkia tuottavaan dataan tai

objektiin, esimerkiksi luvussa 4 mainittuun Beddit-laitteeseen? Tämä voisi olla yksi tuleva tutkimuskohde.

Kiinnostus automatisoituun taiteeseen on noussut viime aikoina, oli kyse viihteellisistä peleistä ja elokuvista tai alan tutkimuksesta. Ihmiset ovat alkaneet kiinnittämään huomiota ja kiinnostumaan enemmän esimerkiksi tilojen ambienssista, tietokoneiden käytöstä taiteessa tai bioresponsiivisista laitteista. Vaikka datamusikalisaation tutkimus on vasta aluillaan, aloja, joilla sitä voidaan käyttää, on noussut kasvavassa määrin.

Lähteet

[1] Alt, F., Shirazi, A. S., Legien, S., Schmidt, A. ja Mennenöh, J.: *Creating Meaningful Melodies from Text Messages.* Teoksessa *Proceedings of the 2010 Conference on New Interfaces for Musical Expression*, NIME 2010, ss. 63–68, Sydney, Australia, 2010. nime.org.

[2] Ames, C.: *Automated Composition in Retrospect: 1956-1986.* Leonardo, 20(2):169–185, 1987.

[3] Ames, C. ja Domino, M.: *Cybernetic Composer: an overview.* Teoksessa *Understanding music with AI*, ss. 186–205. MIT Press, 1992.

[4] Ball, P.: *Computer science: Algorithmic rapture.* Nature, 488(7412):458–458, 2012.

[5] Basu, S., Morris, D. ja Simon, I.: *The songsmith story, or how a small-town hidden Markov model made it to the big time.* The Journal of the Acoustical Society of America, 135(4):2378–2378, 2014.

[6] Beverland, M., Lim, E. A. C., Morrison, M. ja Terziovski, M.: *In-store music and consumer–brand relationships: Relational transformation following experiences of (mis) fit.* Journal of Business Research, 59(9):982–989, 2006.

[7] Beyls, P.: *The Musical Universe of Cellular Automata.* Teoksessa *Proceedings of the International Computer Music Conference*, ICMC 1989, Columbus, Ohio, 1989.

[8] Beyls, P.: *Self-organizing control structures using multiple cellular automata.* Teoksessa *Proceedings of the International Computer Music Conference*, ICMC 1991, ss. 254–254, Montreal, Quebec, Canada, 1991. Michigan Publishing.

[9] Bidlack, R.: *Chaotic systems as simple (but complex) compositional algorithms.* Computer Music Journal, 16(3):33–47, 1992.

[10] Birchfield, D.: *Generative model for the creation of musical emotion, meaning, and form.* Teoksessa *Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence*, ETP 2003, ss. 99–104, Berkeley, California, 2003. ACM.

[11] Bresin, R. ja Friberg, A.: *Emotion rendering in music: range and characteristic values of seven musical variables.* Cortex, 47(9):1068–1081, 2011.

[12] Brooks, F. P., Hopkins, A., Neumann, P. G. ja Wright, W.: *An experiment in musical composition.* IRE Transactions on Electronic Computers, 6(3):175–182, 1957.

[13] Burks, A. W. ja Von Neumann, J.: *Theory of self-reproducing automata.* University of Illinois Press, 1966.

[14] Chomsky, N.: *Syntactic structures.* Walter de Gruyter, 2002.

[15] Colton, S. ja Wiggins, G. A.: *Computational Creativity: The Final Frontier?* Teoksessa *Proceedings of the 20th European Conference on Artificial Intelligence*, ECAI 2012, ss. 21–26, Montpellier, France, 2012. IOS Press.

[16] Cope, D.: *Experiments in Musical Intelligence.* A-R Editions, Inc., Madison, Wisconsin. 1. painos, 1996.

[17] Cope, D.: *New directions in music.* Waveland Press, Prospect Heights, Illinois. 7. painos, 2001.

[18] Cope, D.: *The Well-programmed Clavier: Style in Computer Music Composition.* XRDS: Crossroads, The ACM Magazine for Students, 19(4):16–20, 2013.

[19] Cope, D.: *Experiments in Musical Intelligence*, 2017. `http://artsites.ucsc.edu/faculty/cope/experiments.htm`, vierailtu 2017-03-13 .

[20] Cycling '74: *MAX.* `https://cycling74.com/products/max/`, vierailtu 2017-03-14 .

[21] d'Albe, E. E. F.: *On a Type-Reading Optophone.* Proceedings of the Royal Sociery of London, 90(619):373–375, 1914.

[22] Dannenberg, R. B.: *An On-Line Algorithm for Real-Time Accompaniment.* Teoksessa *Proceedings of the 1984 International Computer Music Conference*, ICMC 1984, ss. 193–198, Paris, France, 1984.

[23] De Prisco, R., Zaccagnino, G. ja Zaccagnino, R.: *EvoBassComposer: a multi-objective genetic algorithm for 4-voice compositions.* Teoksessa

Proceedings of the 12th annual conference on Genetic and evolutionary computation, GECCO, 2010, ss. 817–818, Portland, Oregon, 2010. ACM.

[24] Dombois, F. ja Eckel, G.: *Audification.* Teoksessa Hermann, T., Hunt, A. ja Neuhoff, J. G. (toim.): *The Sonification Handbook*, ss. 301–325. Logos Verlag Berlin, 2011.

[25] DuBois, R. L.: *Applications of generative string-substitution systems in computer music.* väitöskirja, Columbia University, 2003.

[26] Ebcioglu, K.: *An Expert System for Chorale Harmonization.* Teoksessa *Proceedings of the 5th National Conference on Artificial Intelligence*, AAAI 1986, ss. 784–788, Philadelphia, Pennsylvania, 1986.

[27] Farbood, M. ja Schöner, B.: *Analysis and Synthesis of Palestrina-Style Counterpoint Using Markov Chains.* Teoksessa *Proceedings of the 2001 International Computer Music Conference*, ICMC 2001, Havana, Cuba, 2001.

[28] Fernández, J. D. ja Vico, F. J.: *AI Methods in Algorithmic Composition: A Comprehensive Survey.* Journal of Artificial Intelligence Research, 48:513–582, 2013.

[29] Flowers, J. H.: *Thirteen years of reflection on auditory graphing: Promises, pitfalls, and potential new directions.* Teoksessa *Proceedings of the International Conference on Audiotory Display*, ICAD 2005, ss. 406–409, Limerick, Ireland, 2005.

[30] Foo, B.: *Data-Driven DJ.* `https://datadrivendj.com`, vierailtu 2017-02-13 .

[31] Gardner, M.: *Mathematical games: The fantastic combinations of John Conway's new solitaire game "life".* Scientific American, 223(4):120–123, 1970.

[32] Gartland-Jones, A.: *Can a genetic algorithm think like a composer.* Teoksessa *Proceedings of the Generative Art and Design Conference*, GA 2002, ss. 14.1–14.12, Milan, Italy, 2002.

[33] Gillick, J., Tang, K. ja Keller, R. M.: *Learning jazz grammars.* ss. 125–130, 2009.

[34] *Google Magenta*. `https://magenta.tensorflow.org/`, vierailtu 2017-03-27 .

[35] Harmat, L., Takács, J. ja Bodizs, R.: *Music improves sleep quality in students*. Journal of advanced nursing, 62(3):327–335, 2008.

[36] Hartling, L., Newton, A. S., Liang, Y., Jou, H., Hewson, K., Klassen, T. P. ja Curtis, S.: *Music to reduce pain and distress in the pediatric emergency department: a randomized clinical trial*. JAMA pediatrics, 167(9):826–835, 2013.

[37] Haus, G. ja Sametti, A.: *Scoresynth: A system for the synthesis of music scores based on Petri nets and a music algebra*. Computer, 24(7):56–60, 1991.

[38] Hermann, T.: *Taxonomy and Definitions for Sonification and Auditory Display*. Teoksessa *Proceedings of the 14th International Conference on Auditory Display*, ICAD 2008, Paris, France, 2008. IRCAM.

[39] Hild, H., Feulner, J. ja Menzel, W.: *HARMONET: A neural net for harmonizing chorales in the style of JS Bach*. Teoksessa *Advances in Neural Information Processing Systems 4*, NIPS 1991, ss. 267–274, Denver, Colorado, 1991.

[40] Hiller, Jr., L. A. ja Isaacson, L. M.: *Musical Composition with a High-Speed Digital Computer*. Journal of the Audio Engineering Society, 6(3):154–160, 1958.

[41] Holtzman, S.: *Using generative grammars for music composition*. Computer Music Journal, 5(1):51–64, 1981.

[42] Horner, A. ja Goldberg, D. E.: *Genetic algorithms and computer-assisted music composition*. Urbana, 51(61801):437–441, 1991.

[43] Hsü, K. J. ja Hsü, A. J.: *Fractal geometry of music*. Proceedings of the National Academy of Sciences, 87(3):938–941, 1990.

[44] Jylhä-Ollila, T.: *Kunquat tracker*. `http://kunquat.org/`, vierailtu 2017-02-01 .

[45] Kirchmeyer, H.: *On the historical constitution of a rationalistic music*. Die Reihe, 8:11–24, 1968.

[46] Kitani, K. M. ja Koike, H.: *ImprovGenerator: Online Grammatical Induction for On-the-Fly Improvisation Accompaniment.* Teoksessa *Proceedings of the 10th International Conference on New Interfaces for Musical Expression*, NIME 2010, ss. 469–472, Sydney, Australia, 2010.

[47] Krahé, B. ja Bieneck, S.: *The Effect of Music-Induced Mood on Aggressive Affect, Cognition, and Behavior.* Journal of Applied Social Psychology, 42(2):271–290, 2012.

[48] Leach, J. ja Fitch, J.: *Nature, music, and algorithmic composition.* Computer Music Journal, 19(2):23–33, 1995.

[49] Lidov, D. ja Gabura, J.: *A melody writing algorithm using a formal language model.* Computers in the Humanities, 3:138–48, 1973.

[50] Lundqvist, L. O., Carlsson, F., Hilmersson, P. ja Juslin, P. N.: *Emotional responses to music: experience, expression, and physiology.* Psychology of Music, 37(1):61–90, 2009.

[51] Mandelbrot, B. B.: *Fractals.* John Wiley & Sons, Inc., 1977.

[52] Marques, M., Oliveira, V., Vieira, S. ja Rosa, A.: *Music composition using genetic evolutionary algorithms.* Teoksessa *Proceedings of the 2000 Congress on Evolutionary Computation*, CEC 2000, ss. 714–719, La Jolla, California, 2000. IEEE.

[53] Merriam-Webster, Incorporated: *Merriam-Webster Dictionary.* `https://www.merriam-webster.com/dictionary/compose`, vierailtu 2017-02-17 .

[54] Miranda, E. R.: *On the Music of Emergent Behavior: What Can Evolutionary Computation Bring to the Musician?* Leonardo, 36(1):55–59, 2003.

[55] Mozer, M. C.: *Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing.* Connection Science, 6(2-3):247–280, 1994.

[56] Nierhaus, G.: *Algorithmic Composition: Paradigms of Automated Music Generation.* Springer-Verlag Wien. 1. painos, 2009.

[57] Paalasmaa, J., Murphy, D. J. ja Holmqvist, O.: *Analysis of Noisy Biosignals for musical performance.* Teoksessa *Proceedings of the 11th*

International Symposium on Advances in Intelligent Data Analysis, IDA 2012, ss. 241–252, Helsinki, Finland, 2012. Springer.

[58] Paalasmaa, J., Waris, M., Toivonen, H., Leppäkorpi, L. ja Partinen, M.: *Unobtrusive online monitoring of sleep at home.* Teoksessa *Proceedings of the 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, EMBC 2012, ss. 3784–3788, San Diego, California, 2012. IEEE.

[59] Papadopoulos, G. ja Wiggins, G.: *AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects.* Teoksessa *AISB Symposium on Musical Creativity*, ss. 110–117, Edinburgh, Scotland, 1999. AISB.

[60] Phon-Amnuaisuk, S., Smaill, A. ja Wiggins, G.: *A computational model for chorale harmonisation in the style of J.S. Bach.* Teoksessa *Proceedings of the 7th International Conference on Music Perception and Cognition*, ICMPC 2002, ss. 219–222, Sydney, Australia, 2002.

[61] Piston, W.: *Counterpoint.* Norton Company Inc., Lontoo. 4. painos, 1964.

[62] Pousman, Z. ja Stasko, J. T.: *A taxonomy of ambient information systems: four patterns of design.* Teoksessa *Proceedings of the working conference on Advanced visual interfaces*, AVI 2006, ss. 67–74, Venezia, Italy, 2006. ACM Press.

[63] Pressing, J.: *Nonlinear maps as generators of musical design.* Computer Music Journal, 12(2):35–46, 1988.

[64] *Pure Data.* `https://puredata.info/`, vierailtu 2017-03-14 .

[65] *Python Rtmidi.* `https://pypi.python.org/pypi/python-rtmidi`, vierailtu 2017-03-13 .

[66] Quintana, C. S., Arcas, F. M., Molina, D. A., Rodriguez, J. D. F. ja Vico, F. J.: *Melomics: A Case-Study of AI in Spain.* AI Magazine, 34(3):99–103, 2013.

[67] Ribeiro, P., Pereira, F. C., Ferrand, M., Cardoso, A. ja Marrocos, P. de: *Case-based melody generation with MuzaCazUza.* Teoksessa *Proceedings of the Symposium on Artificial Intelligence and Creativity in Arts and Science*, AISB 2001, ss. 67–74, York, United Kingdom, 2001.

[68] Roads, C.: *The Computer Music Tutorial.* MIT Press, Cambridge, MA, USA, 1996.

[69] Rutherford, E. ja Geigerg, H.: *An Electrical Method of Counting the Number of α-Particles from Radio-Active Substances.* Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character, 81(546):141–161, 1908.

[70] Scholes, P.: *The Oxford Companion to Music.* Oxford University Press, London, 1975.

[71] Steinbeis, N. ja Koelsch, S.: *Understanding the Intentions Behind Man-Made Products Elicits Neural Activity in Areas Dedicated to Mental State Attribution.* Cerebral Cortex, 19(3):619, 2008.

[72] Temperley, D.: *Music and Probability.* MIT Press, Cambridge, MA, USA, 2007.

[73] Thelwall, M., Buckley, K., Paltoglou, G., Cai, D. ja Kappas, A.: *Sentiment strength detection in short informal text.* Journal of the American Society for Information Science and Technology, 61(12):2544–2558, 2010.

[74] Todd, P. M.: *A connectionist approach to algorithmic composition.* Computer Music Journal, 13(4):27–43, 1989.

[75] Todd, P. M. ja Loy, D. G.: *Music and connectionism.* MIT Press, 1991.

[76] Toivanen, J. M., Toivonen, H. ja Valitutti, A.: *Automatical Composition of Lyrical Songs.* Teoksessa *Proceedings of the Fourth International Conference on Computational Creativity*, ss. 87–91, Sydney, Australia, 2013.

[77] Tulilaulu, A., Nelimarkka, M., Paalasmaa, J., Johnson, D., Ventura, D., Mylly, P. ja Toivonen, H.: *Data Musicalization.* Lähetetty arvioitavaksi.

[78] Tulilaulu, A., Paalasmaa, J., Waris, M. ja Toivonen, H.: *Sleep musicalization: Automatic music composition from sleep measurements.* Teoksessa *Proceedings of the 11th International Symposium on Advances in Intelligent Data Analysis*, IDA 2012, ss. 392–403, Helsinki, Finland, 2012. Springer.

[79] Viterbi, A.: *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.* IEEE transactions on Information Theory, 13(2):260–269, 1967.

[80] Voss, R. F. ja Clarke, J.: *"1/f noise" in music: Music from 1/f noise.* The Journal of the Acoustical Society of America, 63(1):258–263, 1978.

[81] Walker, B. N. ja Nees, M. A.: *Theory of Sonification.* Teoksessa Hermann, T., Hunt, A. ja Neuhoff, J. G. (toim.): *The Sonification Handbook*, ss. 9–39. Logos Verlag Berlin, 2011.

[82] Welch, L. R.: *Hidden Markov models and the Baum-Welch algorithm.* IEEE Information Theory Society Newsletter, 53(4):10–13, 2003.

[83] Widmer, G.: *Qualitative perception modeling and intelligent musical learning.* Computer Music Journal, 16(2):51–68, 1992.

[84] Zuse, K.: *Rechnender Raum (Calculating Space).* MIT Technical Translation, 1970.

A Geneerisen musikalisaatio-ohjelman komentolistaus

Komento	Arvot	Selitys
sound	on, off	aloittaa tai lopettaa musiikin tuottamisen
tempo	11-450	Säätää tempoa
voices	1-9	Ääniraitojen määrä
suddennote	1-127 (korkeus) ja 0.01-4 (kesto)	pakottaa komennon annosta seuraavalle syklille annetun nuotin
badsound	1-120 (korkeus) ja 0.01-4 (kesto)	soittaa seuraavalla syklillä nuottiklusterin annetulta korkeudelta ylöspäin
drums	on, off	kytkee rummut päälle tai pois
drumpattern	all/1-16	mitä kuviota rummut käyttävät, vai vaihteeko se
accompaniment-pattern	all/1-13	mitä kuviota säestävät äänet käyttävät, vai vaiheteleeko se
accompaniment-sameforall	on, off	käyttävätkö kaikki säestävät äänet samaa kuviota, jos ohjelmalla on lupa vaihdella sitä
globalvolume	1-127	koko ohjelman globaali äänenvoimakkuus
voicevolume	1-10 (äänennumero) ja 1-127 (voimakkuus)	yksittäisen äänen äänenvoimakkuus
drumvolume	1-127	rumpuraidan äänenvoimakkuus
harmony	1-10	musiikin dissonanssipitoisuus niin että 1 on eniten dissonoiva
rythmdensity	1-10	rytmien painotus pidempiin/lyhyempiin rytmeihin
phraselenght	2-24	kuinka monta tahtia per fraasi
voicerole	1-10 (äänen numero) ja melody, accompaniment, bass	määrittää äänen roolin
noticephraseends	on, off	laitetaanko fraasien loppuun pidempi nuotti tai tauko
scaletype	major, naturalminor, harmonicminor, dorian, phrygian, lydian, mixolydian, locrian	määrittää käytettävän asteikon tyypin
scalebase	C, Cs, D, Ds, E, F, Fs, G, Gs, A, As, B	määrittää asteikon pohjasävelen

Komento	Arvot	Selitys
instrument	1-10 (äänen numero) ja 1-127 (instrumentin numero)	muuttaa äänen intrumenttia*
instrument	1-10 (äänen numero) ja instrumentin nimi	muuttaa äänen instrumenttia*
upperbound	1-10 (äänen numero) ja 2-127 (korkeus)	äänen yläraja, ei voi olla matalampi kuin alaraja
lowerbound	1-10 (äänen numero) ja 1-126 (korkeus)	äänen alaraja, ei voi olla korkeampi kuin yläraja

*lista soittimista nimineen ja koodeineen löytyy esimerkiksi osoitteesta:
`http://www.midi.org/techspecs/gm1sound.php`