# Merkkijonoalgoritmeja moniäänisen musiikin hakuun sisällöin perusteella

Saara Nygren

**Tiivistelmä**

Merkkijonoalgoritmeja voidaan käyttää myös polyfonisen musiikin hakemiseen sisällön perusteella. Niiden avulla voidaan polyfonisesta musiikista hakea hahmon tai transpositioinvariantin tarkkaa, likimääräistä tai hajautettua esiintymää. Merkkijonoalgoritmit voidaankin jakaa tässä yhteydessä niihin, joissa musiikin esittämiseen käyteään absoluuttista sävelkorkeutta ja niihin, joissa musiikki esitetään transpositioinvariantteina.

## 1 Johdanto

Alunperin merkkijonomenetelmät on kehitetty tekstimuotoisien hahmojen paikallistamiseen laajoista tekstiaineistoista. Vaikka symbolimuotoinen musiikki koostuukin merkkijonoista monet musiikille tyypilliset piirteet eivät suinkaan ole relevantteja perinteisille teksimerkkijonoille. Siinä missä tekstimerkkijonot ovat aidosti vain merkkijonoja musiikissa merkkijonot tulkitaan soinnuiksi tai mahdollisesti sointujen välisiksi intervalleiksi. Länsimaalaisessa musiikissa on tyypillistä painottaa absoluuttisen sävelkorkeuden sijaan sävelkorkeuksien erotusta eli intervallia. Tätä länsimaalaiselle musiikille tyypillistä ominaisuutta kutsutaan transpositioinvarianssiksi. Monissa merkkijonomenetelmiin perustuvissa algoritmeissa musiikki esitetäänkin transpositioinvariantteina eikä sointuina.

Toinen musiikkille tyypillinen piirre on moniäänisyys eli polyfonisuus. Perinteisillä tekstimerkkijonoilla tämän tyyppistä piirrettä ei esiinny, sillä merkkijonon merkeillä on aina tietty yksikäsitteinen järjestys. Monofoninen musiikki voidaan palauttaa tekstimerkkijonoihin ja näin ollen perinteiset merkkijonoalgoritmit soveltuvat sellaisenaan monofonisen musiikin hakuongelmiin. Sen sijaan sovelettaessa merkkijonomenetelmiä polyfonisen musiikin hakuun on huomattava että musiikista saatetaan ehkä etsiäkin hahmon tai transpositioinvariantin "hajautettua esiintymää". Tällöin perinteiset merkkijonoalgoritmit ovat sellaisenaan käyttökelvottomia.

## 2 Merkkijonomenetelmistä

Olkoon $\Sigma$ äärellinen aakkosto ja $\Sigma^*$ aakkoston kaikkien merkkijonojen joukko. Aakkoston $\Sigma$ merkkijonot ovat muotoa $A = a_1, a_2, ..., a_n$ missä jokainen $a_i \in \Sigma$ kun $i = 1, 2, ..., n$. Lisäksi merkkijonon pituutta merkitään $|A| = n$. Olkoon merkkijono $A$ muotoa $A = \alpha\beta\gamma$ ja $\alpha, \beta, \gamma \in \Sigma^*$. Tällöin $\alpha$ on $A$:n alkuosa ja $\gamma$ sen loppuosa.

Perinteisessä merkkijonojen hahmontunnistusongelmassa etsitään hahmon $P$ esiintymää tai sen likimääräistä esiintymää tekstissä $S$. Likimääräistä esiintymää etsittäessä otetaan käytöön editointietäisyys, joka käytännössä tarkoittaa minimi määrää editointioperaatioita, joilla tekstin jokin alijono voidaan muuntaa hahmoksi. Yleisempiä editointioperaatioita ovat lisäys, poisto ja muutos. Koska musiikista puhuttaessa on hieman kummallista käyttää sanaa teksti, käytetään vastedes termiä lähde.

# 3 BD-algoritmi

Blochin ja Dannerbergin algoritmia [1] voidaan käyttää polyfonisen hahmon hakemiseen polyfonisesta lähteestä. Algoritmin toteutuksen lähtökohtana on tuottaa automaattisesti tahdistettu säestäjä polyfoniselle soololle. Systeemin syötteenä on polyfoninen soolosoitin ja tulosteena tahdistettu säestys kyseiselle soittimille. Perusideana on on verrata soolosoittimelta saatavaa informaatiota sen nuotinnoksesta tuotetuun symboliseen esitykseen ja tämän jälkeen pyrkiä päättämään missä kohtaa säestäjän olisi aloitettava säestys. Nuotinnos esitetään osittainjärjestettynä nuottimerkkijonona ja soolosoittimen ääniä etsitään yksitellen nuotinnoksen symbolisesta esityksestä dynaamisella ohjelmoinnilla.

Aluksi polyfonisesta hahmosta $P$, jossa on $m$ kappaletta sointuja muodostetaan monofoninen hahmo $P'$, jonka pituus on $m'$. Lisäksi muodostetaan matriisi $M_{i,j}$, jossa kullekin hahmon merkille $p'_i$ on sarake ja jokaiselle lähteen soinnulle $S_j = s_{j_1}, ..., s_{j_n}$ rivi. Kuhunkin matriisin elementtiin liitetään kokonaisluku $value$ ja joukko $Used$. Merkitään kutakin matriisin elementin $M_{i,j}$ $value$-arvoa $value_{i,j}$ ja $Used$-joukkoa $Used_{i,j}$.

Matriisi alustetaan sijoittamalla kaikille elementeille $M_{0,j}$ ja $M_{i,0}$ $value = 0$ ja $Used = \emptyset$. Tämän jälkeen jokaiselle matriisin elementille lasketaan $value$-arvo edeltävistä arvoista:

$$value_{i,j} = max \begin{cases} i) & value_{i,j-1} - |S_{j-1} - Used_{i,j-1}| \\ ii) & value_{i-1,j-1} - |S_{j-1} - Used_{i-1,j-1}| \\ iii) & value_{i-1,j} + \begin{cases} 1, & \text{jos } p_i \in S_j \text{ ja } p_i \notin Used_{i-1,j} \\ -\infty, & \text{muuten.} \end{cases} \end{cases}$$

sekä päivitetään kyseistä $Used$-joukkoa riippuen siitä, mistä elementistä $value$-arvo laskettiin:

$$Used_{i,j} = \begin{cases} i) & \{p_i\} \\ ii) & \emptyset \\ iii) & Used_{i-1,j} \bigcup \{p_i\} \end{cases}$$

Hahmon $P$ likimääräiset esiintymät löydetään tutkimalla matriisin rivin $m'$ $value$-arvoja, jotka edustavat hahmon ja lähteen yhtäläisyyksiä.

# 4 DR-algoritmi

Doraisamyn ja Rugerin algoritmi [2] perustuu musiikkidatan indeksointiin ja se käytää n-grammeja likimääräisten hajauttetujen transpositioinvarianttien löytämiseen.

Polyfoninen musiikki esitetään syttymishetken mukaan järjestettyinä pareina $\langle q, t \rangle$, jossa $q$ on nuotin korkeus ja $t$ sen syttymishetki. Ensin musiikkidata jaetaan liukuvan ikkunan -menetelmällä päällekäisiksi ikkunoiksi ikkunan koon ja syttymishetkien perusteella. Tämän jälkeen käydään kussakin ikkunassa läpi kaikki mahdolliset tapahtumajonot ja niistä muodostetaan kunkin ikkunan intervallisekvenssit laskemalla peräkkäisten sävelien erotukset. Rytmin mallintamiseksi lasketaan lisäksi peräkkäisten syttymishetkien erotuksen suhde.

N-grammi muodostetaan käyttämällä sekä intervalli- että syttymishetkien suhdeinformaatiota. Koska mahdollisia intervallien ja syttymishetkien suhteiden arvoja on erittäin paljon, koodataan n-grammit johonkin äärelliseen aakkostoon. Intervallit koodataan siten, että koodaus on herkempi oktaavin alueella, kuin sen ulkopuolella. Negatiiviset intervallit esitetään pienillä kirjaimilla a-z ja positiiviset isoilla kirjaimilla A-Z. Syttymishetkien suhteet esitetään siten että suhteen ollessa 1 sitä merkitään kirjaimella Z. Suhteiden ollessa välillä [1, 4.5] merkitään niitä isoilla kirjaimilla A-I ja jos suhde on yli 4.5 sitä merkitään isolla kirjaimella Y. Suhteiden ollessa alle yhden niitä merkitään vastaavasti, mutta pienin kirjaimin.

# 5 ShiftOr-algoritmi

ShiftOr-algoritmi [3] on Baeza-Yatesin ja Gonnetin esittämä bittirinnakkaisalgoritmi tarkan hahmon etsimiseen lähteestä. Algoritmi muodostaa bittitaulukon $T$, jossa on sarake kutakin lähteen $S$ aakkostoon $\Sigma$ kuuluvaa merkkiä $s_i$ kohti ja rivi kullekin hahmon $P$ merkille $p_j$. Jos hahmon merkki täsmää lähteen merkkiin merkitään taulukon kohtaan $T_{i,j}$ 0 muuten 1. Kun $T$ on muodostettu etsitään hahmon esiintymää lähteestä iteroimalla bittivektoria $E$ seuraavasti:

$$E \leftarrow shiftleft(E) \vee T_i$$

Hahmon esiintymä löydetään, kun $E$:n alimalle riville saadaan 0.

# 6 ShiftOr-algoritmin muunnokset

On selvää ettei edellä esitetyllä ShiftOr-algoritmilla löydetä hahmon hajautettua esiintymää tai hahmon transpositioinvarianttien esiintymää polyfonisesta lähteestä. Seuraavassa esitellään kaksi ShiftOr muunnosta ShiftOrAnd- ja MonoPoly- algoritmit, josta toinen löytää hahmon hajautetut esiintymät ja toinen hahmon transpositioinvarianttien esiintymät.

## 6.1 ShiftOrAnd-algoritmi

ShiftOr-algoritmi voidaan pienillä muutoksilla muokata löytämään hahmon hajautetut esiintymät lähteestä. ShiftOrAnd-algoritmissa [3] taulukko $T$ muodostetaan kuten ShiftOr-algoritmissa. Kullekin lähteen $S$ soinnulle $S_k$ saadaan esitys yhdistämällä soinnun säveliä $s_{k_i}$ vastaavia T:n sarakkeita loogisella ja -operaatiolla. Näin ollen ShiftOrAnd-algoritmi toimii kuten ShiftOr-algoritmi, mutta $E$:n muodostamiseen käytetään $T_i$ sijasta konjuktioa sointuun kuuluvien sävelien sarakkeista $T_i$. Kuten edellä esiintymä löydetaan kun $E$:n alimmalle riville saadaan 0.

### 6.2 MonoPoly-algoritmi

MonoPoly-algoritmi [3] ShifOr-algoritmin muunnelma hahmon transpositioinvarianttien esiintymien löytämiseksi lähteestä, kun lähde on staattinen. Algoritmi on jaettu kahteen vaiheeseen, jotka ovat esiprosessointi ja suodatusvaihe. Esiprosessoinnissa lähteestä $S = S_1...S_n$ muodostetaan bittivektorimerkkijono $\overline{S} = \overline{S}_1...\overline{S}_{n-1}$ missä kukin $\overline{S}_i$ koodaa sointujen $S_i$ ja $S_{i+1}$ väliset intervallit. Kunkin bittivektorin pituudeksi tuleva aakkoston koko $|\Sigma'|$ voidaan valita halutulla tavalla, mutta käytännössä 12 on sopiva.

Suodatusvaiheessa muodostetaan bittitaulukko $T$, jonka koko on $(m-1) \times 2^{|\Sigma'|}$. Taulukossa on sarake kullekin bittivektorin $\overline{S}_i$ arvolle ja rivi jokaista hahmosta $P$ laskettua intervallia $(p_{i+1} - p_i)$ kohti. Lisäksi laskennannan tehostamiseksi muodostetaan bittitaulukot $I$ ja $L$. Taulukon $I$ koko on $|\Sigma'| \times |\Sigma'|$ ja se sisältää sarakkeen jokaiselle aakkoston $\Sigma'$ mahdolliselle intervallille. Taulukon $L$ koko on puolestaan $|\Sigma'| \times (m-1)$ ja siihen tallennetaan tieto hahmon intervalleista. Suodatusvaiheen lopulla mahdollisia esiintymiä etsitään ShiftOr-algoritmilla ja sen antamat tulokset vielä tarkistetaan jollakin naiivilla menetelmällä.

## Viitteet

[1] Bloch,J., Dannenberg, R., Real-Time Accompaniment of Keyboard Performances, Proc. ICMC'85, 279-290.

[2] Doraisamy, S., Ruger, S., An Approach Towards A Polyphonic Music Retrieval System, Proc. ISIMIR'01, 187-193.

[3] Lemström, K., Tarhio, J., Transposition Invariant Pattern Matching for Multi-Track Strings, Nordic Journal of Computing , 2003.