Erkki Mäkinen (toim.)

Tietojenkäsittelytieteellisiä tutkielmia
Kevät 2017

TAMPEREEN YLIOPISTO

INFORMAATIOTIETEIDEN YKSIKÖN RAPORTTEJA 54/2017

TAMPERE 2017

TAMPEREEN YLIOPISTO
INFORMAATIOTIETEIDEN YKSIKÖN RAPORTTEJA 54/2017
TOUKOKUU 2017

Erkki Mäkinen (toim.)

Tietojenkäsittelytieteellisiä tutkielmia Kevät 2017

ISBN 978-952-03-0468-3 (pdf)

ISSN-L 1799-8158
ISSN 1799-8158

Sisällysluettelo

Neural gas -algoritmi ja sen sovellukset

Marja Ahonen

Tiivistelmä

Tutkielmassani perehdyn neural gas -algoritmiin, joka perustuu vektorikvantisaatioon. Algoritmi mahdollistaa syötedatan tiivistämisen, ja suuri määrä syötevektoreita voidaan esittää pienellä määrällä painovektoreita. Neural gas -algoritmi vaatii vähän tietoa ennakkoon syötedatasta, toisin kuin esimerkiksi Kohosen itseorganisoituva kartta. Algoritmi pystyy mukautumaan itsenäisesti vastaamaan syötedataa saavuttaen samalla optimaalisen virhefunktion tuloksen.

Vielä vähemmän ennakkotietoja syötedatasta tarvitsee kasvava neural gas -algoritmi, joka aloittaa vain kahdella satunnaisesti valitulla neuronilla. Algoritmi pystyy sen jälkeen lisäämään uusia neuroneita neuroverkkoon. Tutkielmassani myös vertailen lyhyesti näiden kahden algoritmin eroja. Yksi merkittävä ero on se, että neural gas -algoritmissa parametrit aloittavat suurilla lähtöarvoilla ja pienenevät kohti nollaa algoritmin edetessä. Kasvavassa neural gas -algoritmissa näillä parametreilla on kiinteät arvot läpi algoritmin suorituksen. Molempia algoritmeja on käytetty kolmiulotteisten hahmojen analysoimisessa, ja varsinkin kasvava neural gas -algoritmi on tuottanut lupaavia tuloksia.

Avainsanat: vektorikvantisaatio, kasvava neural gas -algoritmi, Voronoin monikulmio

1 Johdanto

Koneoppiminen (machine learning) on mahdollistanut järjestelmien oppimisen suoraan tarjolla olevasta informaatiosta sen sijaan, että kaikki järjestelmän toiminnot pitäisi olla etukäteen ihmisen ohjelmoimia [2]. Koneoppiminen pohjautuu algoritmeihin, joiden avulla järjestelmä pystyy esimerkiksi parantamaan tai analysoimaan saamaansa dataa [9]. Erilaisia algoritmeja on tuhansia, mutta loppujen lopuksi ne ovat vain erilaisia yhdistelmiä kolmesta peruskomponentista: esitystavasta, arvioinnista ja optimoinnista [2].

Koneoppimisen ja ihmisen oppimisen välillä on suuria eroja, jotka ovat olleet suurimpia haasteita koneoppimisen kehityksessä [9]. Michalski ja Kodratoff [9] huomauttavatkin, että ihmiset oppivat väistämättä samalla, kun he suorittavat jotain tehtävää. Ihmiset pystyvät parantavat omia tuloksiaan välillä jopa huomaamattaan, vahingossa.

Nykypäivänä koneoppiminen on läsnä jokapäiväisessä toiminnassamme, sitä käytetään esimerkiksi Internet-hakujen tekoon, sähköpostien roskapos-

tisuodatuksessa, suositusjärjestelmissä, mainosten valitsemisessa ja asettelussa, petosten havaitsemisessa, osakekaupoissa ja lääkkeiden suunnittelussa [2]. Koneoppiminen tekeekin Internetin käytöstämme samalla sekä helpompaa että rasittavampaa. Saamme esimerkiksi vähemmän roskapostia sähköpostin saapuneiden postien kansioon, mutta selaimiin sijoitetut mainokset voi olla vaikeampi ohittaa, koska ne ovat kohdennettu jokaiselle käyttäjälle henkilökohtaisesti.

Neuroverkkojen (neural networks) avulla pystytään ratkomaan muun muassa tunnistukseen, ennustamiseen, optimointiin, assosiaatiomuistiin ja kontrollointiin liittyviä ongelmia [5]. Bishop [1] tiivistää neuroverkojen tärkeyden siihen, että ne pystyvät esittämään usean syötemuuttujan ei-lineaarisena kuvauksena useaan tulostesyötteeseen, ja tämän kuvauksen muotoa pystytään hallitsemaan useilla eri parametreilla. Eri algoritmeilla on erilaisia hyötyjä ja haittoja eri tilanteissa, eikä voikaan sanoa, että yksi ja ainoa algoritmi on aina oikea vaihtoehto joka tilanteeseen.

Haykin [4] mainitsee, että ihmisen aivojen neuronit eivät pysty työskentelemään yhtä nopeasti kuin nopeimmat keinotekoiset piisirut. Ihmisen aivot kuitenkin paikkaavat tämän suurella yhteyksien määrällä neuronista toiseen [4]. Osa neuroverkoista onkin biologisten neuroverkkojen innoittamia, ja niiden toivotaan saavuttavan joitain biologisen neuroverkkojen hyödyllisiä ominaisuuksia [5]. Nykypäivän tietokoneet ovat monessa suhteessa täysin ylivoimaisia verrattuna ihmisen aivoihin. On silti osa-alueita, joista ihmisen aivot selviävät nopeasti ja pienellä teholla, ja jotka ovat saavutettavissa tietokoneilla vain suurella vaivalla, jos silloinkaan.

Neuroverkot koostuvat keinotekoisista neuroneista, jotka muodostavat yhteyksiä keskenään. Neuroverkot hyödyntävät rinnakkaiskäsittelyä. [4] Tällöin monia pienempiä laskutoimituksia voidaan suorittaa samanaikaisesti, eikä aina tarvitse odottaa, että edellinen suoritusvaihe on saatu käsiteltyä loppuun.

Neuroneiden välisen yhteyden vahvuutta kutsutaan *synaptiseksi painoksi* (synaptic weight) tai painovektoreiksi, ja varsinainen tieto tallennetaan niiden avulla [4]. Synaptisia painoja kutsutaan usein myös *viitevektoreiksi* (reference vector), ja ne voi ymmärtää syöteavaruuden neuroneita vastaavina paikkoina [3]. Eri algoritmeissa painovektorit reagoivat eri tavoin havaittuihin syötteisiin. Joissain tapauksissa neuroverkosta vastaa vain yksi painovektori, kun taas joissakin algoritmeissa vastaavia painovektoreita voi olla useampia.

Havaitessaan *syöte-* tai *datavektorin* (input or data vector) neuroverkko muokkaa painovektoreitaan vastaamaan syötevektoria mahdollisimman tarkasti [4]. Tavoitteena on saada kuvattua iso joukko syötevektoreita pienellä määrällä painovektoreita.

Neuroverkolle annettevia syötevektoreita saatetaan myös esiprosessoida, jotta neuroverkko pystyy oppimaan datan sisältämän informaation paremmin. Esiprosessointimenetelmän valinta on merkittävä päätös, sillä esiprosessointi vaikuttaa voimakkaasti lopputulokseen. Esiprosessoinnin aikana saate-

taan esimerkiksi vähentää syötedatan ulottuvuuksia. [1]

Neuroverkko ei aina pysty reagoimaan syötevektoriin oikealla tavalla, esimerkiksi painovektoreiden tekemä mukautumisliike voi olla riittämätön. Tämän vuoksi käytetään *virhefunktioita* (error function), joiden avulla tarkastellaan tapahtunutta muutosta neuroverkossa ja muokataan painovektoreita pienentämään havaittua virhettä [5]. Virhefunktion tuottamien arvojen avulla saatetaan myös määritellä esimerkiksi lisättävien neuronien sijoituspaikka.

Neuroverkot oppivat eri tavoilla, riippuen käytetystä algoritmista. Jos kyseessä on *ohjattu oppiminen* (supervised learning), neuroverkolle tarjotaan opetusaineistoa, josta kustakin syötevektorista tiedetään haluttu vaste, jolloin voidaan ohjata neuroverkon toimintaa [4]. Jos kyseessä on *ohjaamaton oppiminen* (unsupervised learning), opetusaineistoa ei ole saatavilla, ja syötteiden vaste ei ole etukäteen tiedossa [1]. Useinkaan ei ole mahdollista saada opetusaineistoa neuroverkolle, joten tehokkaasti toimiva ohjaamatonta oppimista käyttävä algoritmi voi olla ainoa vaihtoehto.

Neuroverkkojen tärkein ominaisuus on se, että ne oppivat ympäristöstään ja parantavat suoritustaan oppimisen edetessä [4]. Neuroverkolla on tavoitteenaan jokin tietty tehtävä, jonka se voi saavuttaa iteroimalla ja päivittämällä neuroverkon painovektoreita ja yhteyksiä [5]. Bishop [1] määrittelee, että neuroverkkojen keskeisin tavoite on mallintaa taustalla olevan datan generointia, eikä opetella ulkoa harjoitteludataa.

Monet neuroverkkojen *oppimisalgoritmit* (learning algorithms) vaativat optimaalisten tulosten saavuttamiseen *aiempaa tietoa* (prior knowledge) esimerkiksi neuroverkon topologiasta [6]. Tämän tiedon saaminen etukäteen ei aina ole ongelmatonta, minkä takia on kehitetty monia algoritmeja, jotka eivät vaadi näitä tietoja etukäteen.

Neural gas -algoritmi on tiedon tiivistystekniikka. Algoritmi toimii ohjaamatta, ja se pystyy esittämään suuren määrän syötevektoreita pienellä määrällä painovektoreita. Algoritmi pystyy toimimaan pienellä määrällä ennakkotietoja syöteavaruudesta. Neural gas -algoritmilla saavutetaan pieniä virhevääristymiä.

Tutkielmani on kirjallisuuskatsaus neural gas -algoritmin toimintaan ja sen sovelluksiin. Työni lähteiksi olen valinnut sekä alan perusteoksia ja -artikkeleita että joitain uusimmista tutkimustuloksista kertovista artikkeleista. Ensiksi mainittujen avulla käsittelen neural gas -algoritmin perusteita ja taustoja, sekä esittelen kasvavan neural gas -algoritmin toiminnan. Jälkimmäisten avulla esittelen algoritmien käyttöä kolmiulotteisten kohteiden analysoinnissa.

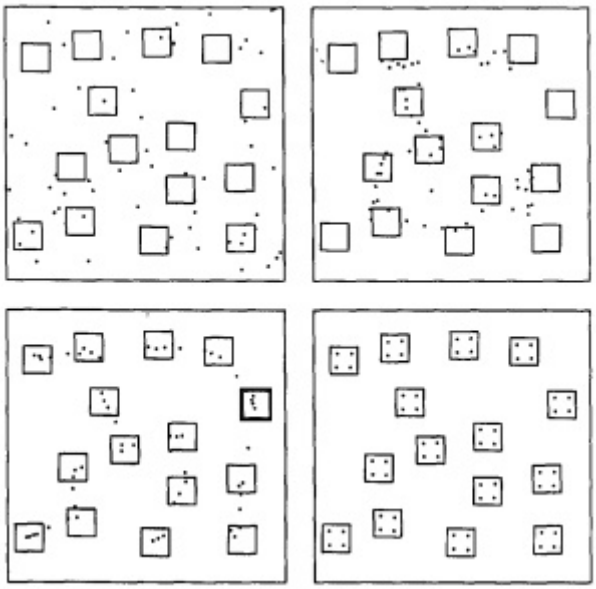
Luvussa 2 käsittelen neural gas -algoritmin periaatteita. Kolmannessa luvussa käyn läpi algoritmin toimintavaiheet. Neljännessä luvussa keskityn neural gas -algoritmin erityispiirteisiin. Viidennessä luvussa tarkastelen neural gas -algoritmin muunnelman kasvavan neural gas -algoritmin toimintavaiheita ja erityispiirteitä. Tämän jälkeen vertailen näiden kahden algoritmin eroja lyhyesti. Ennen yhteenvetoa käsittelen vielä luvussa 7 molempien algoritmien

käyttöä kolmiulotteisten objektien analysoinnissa.

2 Neural gas -algoritmin periaatteet

Neural gas -algoritmi pohjautuu vektorikvantisaatioon, joka on tekniikka muun muassa tiedon tiivistykselle. Tähän tekniikkaan on monia eri lähestymistapoja, joista esimerkiksi yksi on Kohosen itseorganisoituva kartta. Fritzke [3] tiivistää neural gas -algoritmin toimimisen seuraavanlaisesti: jokaisen syötesignaalin kohdalla mukautetaan sitä k:ta lähinnä olevaa pistettä, kun k pienenee suuresta lähtöarvosta pieneen lopulliseen arvoon.

Neural gas -algoritmin on todettu suppenevan nopeasti ja saavuttavan matalia virhevääristymiä, jotka ovat pienempiä kuin mitä on saavutettavissa k:n keskiarvon menetelmällä, suurimman entropian klusterointi -menetelmällä tai Kohosen itseorganisoituvalla kartalla [7]. Esimerkiksi Kohosen itseorganisoituva kartta perustuu "voittaja-vie-kaiken" -periaatteelle. Tämä tarkoittaa sitä, että neuroverkon neuronit kilpailevat keskenään siitä, mikä neuroni saa vastata syötteeseen, ja tulosteessa on vain tämä yksi neuroni [4]. Neural gas -algoritmin on kuitenkin kutsuttu toimivan niin sanotulla "voittaja-vie-suurimman-osan" -periaatteella [6].



Kuva 1: Neural gas -algoritmin vaiheet [7].

Kuvassa 1 on 60 painovektoria sekä 15 erillistä klusteria. Painovektorit kuvataan pisteinä ja niiden aloitusarvot on valittu satunnaisesti. Yläoikeassa kuvassa on neural gas -algoritmin rakenne 5000 oppimisaskeleen jälkeen. Alavasemmalla on tilanne 15000 oppismisaskeleen jälkeen. Alaoikealla on neural gas -algoritmin lopullinen rakenne 80000 oppimisaskeleen jälkeen, jolloin

algoritmi on pystynyt löytämään optimaalisen esityksen datan jakaumasta. Painovektorit ovat levittäytyneet optimaalisesti, jokaisessa klusterissa on neljä painovektoria. [7]

Vektorikvantisaatiotekniikassa tarkastellaan dataa M. Syöteavaruuteen R^n sijoitetaan äärellinen joukko A painovektoreita $\mathbf{w}_i \in R^n, i = 1, 2, \ldots, N$ [6]. Jokaista havaintoa eli syötevektoria $\mathbf{v} \in M$ kuvaa yksi A:han kuuluva voittajapainovektori $\mathbf{w}_j$, joka kuvaa $\mathbf{v}$:tä parhaiten [7]. Painovektoreiden avulla syntyy niin sanottu *koodikirja* (codebook), jonka avulla yritetään tiivistää alkuperäisen datan sisältämä informaatio N:n painovektorin avulla [6].

Painovektori $\mathbf{w}_j$ valitaan siten, että sillä on pienin *etäisyys* (distortion measure) $d(\mathbf{v}, \mathbf{w}_j)$ syötevektoreista $\mathbf{v}$. Vääristymä lasketaan esimerkiksi *neliövirheen* (square error) avulla, eli valitaan painovektori, jolla $\|\mathbf{v} - \mathbf{w}_j\|^2$ on pienin. Näin data M saadaan ositettua alueiksi

$$M_j = \{\mathbf{v} \in M | \|\mathbf{v} - \mathbf{w}_j\| \leq \|\mathbf{v} - \mathbf{w}_i\| \forall_i\},$$

eli *Voronoin monikulmioksi* (Voronoi polygon). [6] Monikulmiossa alue on jaettu siten, että mikä tahansa kohta valitaankin, sitä lähin piste löytyy saman Voronoin solun sisältä [4].

Voronoin monikulmio muodostaa *Delaunayn kolmioinnin* (Delaunay triangulation) (kuva 2), kun Voronoin monikulmion vierekkäisten alueiden neuronit yhdistetään toisiinsa viivoilla. Delaunayn kolmioinnissa yhdistyneet neuronit i ja j pystyvät luomaan keskenään yhteyksiä, jos havaitaan syöte, jolle lähin neuroni on i ja toiseksi lähin on j. [3]

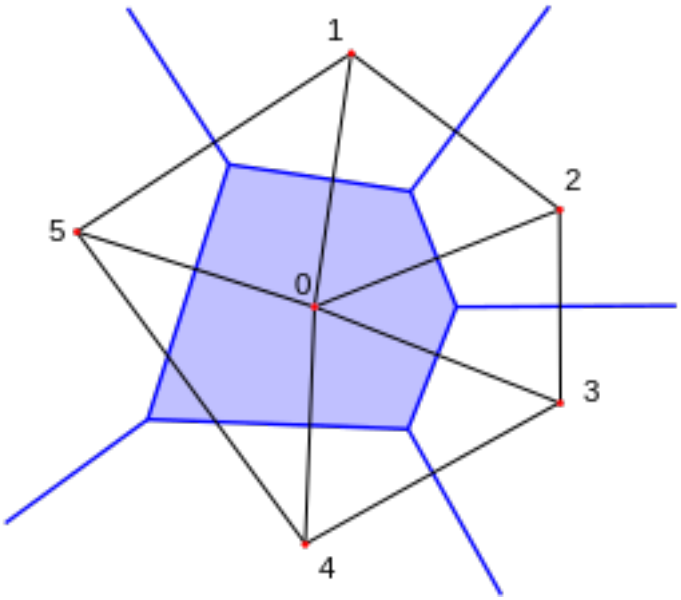
Kuvassa 2 varjostettu alue kuvaa aluetta, jolla havaitut syötteet ovat lähimpänä neuronia 0. Aina kun syöte havaitaan, syötettä lähin neuroni i muodostaa yhteyden toiseksi lähimpään neuroniin. Neuroni i muodostaa yhteyksiä vain sellaisten neuronien kanssa, joiden kanssa sillä on vierekkäiset alueet. [6] Suorat, mustat viivat muodostavat Delaunayn kolmioinnin, joka yhdistää vierekkäiset neuronit toisiinsa [3].

Kohosen itseorganisoituva kartta saattaa joissakin tilanteissa olla huono valinta, koska se vaatii aiempaa tietoa neuroverkon topologiasta [6]. Martinez ja Schulten [6] esittävät, että synaptisia painoja tulisi muokata itsenäisesti, ilman mitään tietoa neuroneiden topologisesta sijoittumisesta neuroverkossa. Neural gas -verkossa synaptiset painot määräytyvät neuroneiden suhteellisista etäisyyksistä toisiinsa syöteavaruudessa [6]. Neural gas -algoritmi vaatii vähemmän etukäteistietoa neuroverkon topologiasta kuin jotkin toiset algoritmit, mutta myös neural gas -algoritmille pitää antaa joitakin tietoja ennen algoritmin suorittamista, muun muassa oppimisaskelten kokonaislukumäärä pitää määritellä etukäteen.

Jokaisella syötevektorilla $\mathbf{v}$ on vääristymien joukko

$$D_v = \{\|\mathbf{v} - \mathbf{w}_i\|, i = 1, 2, \ldots, N\},$$

joka sisältää informaation vastaanottavien alueiden järjestyksestä syötetilassa. Aina kun uusi syöte $\mathbf{v}$ havaitaan, alkioiden järjestys joukossa D_v määrää



Kuva 2: Voronoin monikulmio.

synaptisten painojen $\mathbf{w}_i$ mukauttamisen. Koska alkioiden järjestys joukossa D_v määrää synaptisten painojen muutoksen, neural gas -algoritmin katsotaan toimivan "voittaja-vie-suurimman-osan" -periaatteella. [6] Yksi painovektori ei voita kaikkia muita painovektoreita. Vaikka yksi painovektori onkin lähempänä havaittua syötevektoria kuin muut painovektorit, myös muut painovektorit mukauttavat paikkaansa. Neuroverkon voittajavektori tekee kuitenkin suurimman muutosliikkeen.

Kun uusi syöte havaitaan, neuronit i ja j, joita vastaavat alueet M_i ja M_j ovat vierekkäiset, muodostavat yhteyksiä toisiinsa. Näitä yhteyksiä kuvataan matriisin alkiolla C_{ij}, joka neuronien välisen yhteyden ollessa olemassa on arvoltaan yksi. Jos neuronien i ja j välillä ei ole olemassa yhteyttä, matriisin alkio C_{ij} on arvoltaan nolla. Oppimisprosessin lopussa syntynyt liittyvyysmatriisi C_{ij} kuvaa samankaltaisuuksia, esimerkiksi naapuruussuhteita syötedatan sisällä. [6]

Algoritmissa käytetään myös *Hebbin säännön kaltaista sääntöä* (Hebb-like rule) [6]. Hebbin sääntö on alun perin määritelty koskemaan biologisten aivojen solujen keskenään luomia yhteyksiä [4]. Sääntö määrittelee, että jos neuroni S on riittävän lähellä neuronia T, ja se säännöllisesti tai yhtämittaisesti aktivoi niiden välisen yhteyden, toisessa tai molemmissa neuroneissa tapahtuu kasvua siten, että S:n tehokkuus T:n aktivoimisessa lisääntyy [4]. Sääntöä voi kuitenkin soveltaa myös keinotekoisiin neuroverkkoihin.

Neural gas -algoritmissa Hebbin sääntöä sovelletaan niin, että synaptisen painon muutokseen vaikuttaa sekä havaittu syötevektori $\mathbf{v}$ että jälkikäteen laskettu "aktivoituminen". Jokaisen syötevektorin vastaanottavan neuronin i kohdalla lasketaan neuronin i aktivoituminen $f_i(D_v)$, joka riippuu vääristy-

mien joukosta D_v [6]. Tähän aktivoitumisen määrään vaikuttaa myös muutosten askelkoko ϵ, joka määrätään etukäteen [7]. Askelkoko $\epsilon \in [0, 1]$ kuvaa painovektoreiden muutosten suuruutta [7]. Algoritmin edetessä askelkoko ϵ pienenee, jolloin muutokset verkossa muuttuvat alkutilannetta pienemmiksi [8].

Martinez ja Schulten [6] käyttivät funktiota $f(k_i) = e^{-k_i/\lambda}$ painovektoreiden mukautuksen osana, jossa k_i on k:s neuroni, joka on yhteydessä neuroniin i. Rapautumismuutujalla λ on algoritmin suorituksen alussa suurehko arvo, ja sitä lähdetään pienentämään algoritmin edetessä [7]. Rapautumismuuttuja kuvaa niiden neuroneiden lukumäärää, jotka merkittävästi muuttavat omaa synaptista painoaan kunkin oppimisaskeleen aikana [6].

Neuroverkon jokaisella yhteydellä $i-j$ on ikä t_{ij}. Jos yhteyden ikä ylittää ennaltamäärätyn maksimieliniän T, neuroneiden välinen yhteys poistetaan asettamalla matriisin alkio C_{ij} nollaksi. [6]

Jotta neuroverkko pystyy kuvaamaan neuroneiden välisiä naapuruussuhteita, päivitetään neuroneiden välisiä yhteyksiä matriisissa C, kun uusi syötesignaali havaitaan. Aina kun syötevektori $\mathbf{v}$ havaitaan, muodostetaan yhteys neuronista i_0 neuroniin i_1, joilla i_0:n painovektori $\mathbf{w}_i$ oli lähimpänä syötevektoria $\mathbf{v}$ ja i_1:n painovektori $\mathbf{w}_i$ oli toiseksi lähinnä $\mathbf{v}$:tä. Jos neuroneiden i_0-i_1 välille ollaan muodostamassa yhteyttä, mutta se on jo olemassa, astetaan $t_{i_0 i_1}$ nollaan ja yhteys alkaa vanheta uudelleen. [6]

Neural gas -algoritmia varten tulee siis etukäteen määrittää lopullinen oppimisaskelten lukumäärä t_{max}, neuroverkon muutosten askelkoon ϵ:n lähtöarvo, neuroverkon rapautumismuuttujan λ:n lähtöarvo ja neuroverkon yhteyksien maksimielinikä T. Neural gas -algoritmikin vaatii joitain päätöksiä neuroverkosta etukäteen, mutta se ei silti vaadi yhtä paljon ennakkotietoa kuin jotkin toiset algoritmit.

3 Algoritmin toimintavaiheet

Martinezin ja Schultenin [6] mukaan neural gas -algoritmin vaiheet ovat seuraavanlaiset:

0. Alustetaan painovektorit $\mathbf{w}_i \in R^n$ alkuarvoilla ja asetetaan kaikki arvot C_{ij} nollaksi.

1. Valitaan syötevektori $\mathbf{v}$ datasta M.

2. Jokaiselle neuronille i määritetään k_i verran lähimpiä naapureita j, joilla
$$\|\mathbf{v} - \mathbf{w}_j\| < \|\mathbf{v} - \mathbf{w}_i\|.$$
Tämä voidaan tehdä esimerkiksi määrittelemällä sekvenssi (i_0, i_1, ..., i_{N-1}) neuroneista, joilla
$$\|\mathbf{v} - \mathbf{w}_{i_0}\| < \|\mathbf{v} - \mathbf{w}_{i_1}\| < \cdots < \|\mathbf{v} - \mathbf{w}_{i_{N-1}}\|.$$

3. Mukautetaan painovektoreiden painoja, eli

$$\mathbf{w}_i^{new} = \mathbf{w}_i^{old} + \epsilon \cdot e^{-k_i/\lambda}(\mathbf{v} - \mathbf{w}_i^{old}), \qquad i = 1, 2, \ldots, N.$$

4. Jos $C_{i_0 i_1} = 0$, asetetaan $C_{i_0 i_1} = 1$ ja $t_{i_0 i_1} = 0$. Jos $C_{i_0 i_1} = 1$, asetetaan vain $t_{i_0 i_1} = 0$.

5. Kasvatetaan kaikkien i_0:n yhteyksien ikää asettamalla $t_{i_0 j} = t_{i_0 j} + 1$, kaikille neuroneille j, joilla $C_{i_0 j} = 1$.

6. Poistetaan kaikki i_0:n yhteydet, joiden ikä on kasvanut liian suureksi asettamalla $C_{i_0 j} = 0$, kaikille j, joiden $C_{i_0 j} = 1$ ja $t_{i_0 j} > T$. Jatketaan kohdasta 1.

Algoritmia toistetaan, kunnes oppimisaskelten määrä saavuttaa ennaltamäärätyn maksimimäärän [3]. Martinez ja muut [7] testasivat maksimiiteraatioiden määrän vaikusta algoritmin tuottamiin tuloksiin ja totesivat, että algoritmi ei enää paranna tuloksiaan huomattavasti, vaikka oppimisaskelia suoritettaisiin enemmän kuin 100 000.

Algoritmin aikaa vievin vaihe on naapurien määrittäminen neuronille i [6]. Martinez ja Schulten [6] huomauttavat, että tämä ei kuitenkaan ole suuri ongelma, koska λ:n pienentyessä lajittelua ei tarvitse suorittaa loppuun asti. He myös mainitsevat, että neuronien ei tarvitse tietää, mitkä muut neuronit reagoivat aktiivisemmin syötteeseen, neuronien tulee vain tietää aktiivisempien neuronien lukumäärä [6].

4 Neural gas -algoritmin erityispiirteitä

Witoelar ja muut [13] vertasivat neural gas -algoritmin toimintaa ”voittaja vie kaiken” -algoritmiin. He vertasivat muun muassa virhearvojen muuttumisen nopeutta ja totesivat, että neural gas -algoritmi on sietokykyisempi neuroneiden alkuperäisen sijainnin suhteen kuin ”voittaja vie kaiken” -algoritmi. Neural gas -algoritmi myös saavutti parhaat virhearvot asymptoottisesti. ”Voittaja vie kaiken” -algoritmi pärjäsi hyvin, jos neuronit alustettiin lähelle tiheän datan alueita. [13]

Algoritmin käyttämä askelkoko ϵ muuttuu samassa suhteessa ja tahdissa rapautumismuuttujan λ kanssa. Molempien arvojen muutos lasketaan samanlaisella kaavalla

$$\epsilon(t) = \epsilon_i(\epsilon_f/\epsilon_i)^{t/t_{max}},$$

jossa ϵ_i on askelkoon lähtöarvo, ϵ_f on askelkoon loppuarvo, t on oppimisaskelten lukumäärä ja t_{max} on ennaltamäärätty oppimisaskelten maksimilukumäärä [7]. Rapautumismuuttujan kaava on samanlainen, ϵ:n tilalle sijoitetaan vain λ vastaavilla arvoillaan. Eli ϵ_i tilalle sijoitetaan λ_i, ja niin edelleen.

Muuttujien lähtö- ja loppuarvot ovat hyvin erisuuruiset, esimerkiksi λ:n lähtöarvo voi olla 10, kun ϵ:lla se voi olla 0,5 [7].

Kun rapautumismuuttujan λ arvo on aluksi suuri, muutokset kohti syötesignaalia ovat suurempia. Kun rapautumismuuttujan arvoa aletaan pienentää, lopulta vain syötesignaalin lähintä painovektoria mukautetaan. [3] Rapautumismuuttujan pienentämisellä myös estetään painovektoreita jäämästä ansaan epäedullisiin asemiin [7].

Algoritmilla on kaksi eri mahdollisuutta päivittää neuroverkon yhteyksien ikiä. Yksi tapa on asettaa kaikkien yhteyksien ikä vastaamaan läpikäytyjen oppimisaskelten eli iteraatiokierrosten lukumäärää yhteyden perustamisen jälkeen. Tämä kuitenkin vaatii synkronoidun päivityksen kaikkiin yhteyksiin jokaisella oppimisaskeleella. Toinen ja vähemmän aikaa vievä tapa on kasvattaa yhdellä oppimisaskeleella vain sen neuronin i_0 yhteyksiä, joka on lähinnä havaittua syötettä. Kunhan jokaisen neuronin $i \in A$ todennäköisyys tulla valituksi "voittajaneuroniksi" on yhtä suuri, molemmat yhteyksien ikiä laskevat tavat tuottavat keskimäärin samanlaiset tulokset. [6]

Neuroverkkoon halutaan vain niiden neuronien väliset yhteydet, jotka ovat syötedatan läheisyydessä. Kauempana olevat neuronit ovat neuroverkon topologian kannalta hyödyttömiä, ja niitä kutsutaankin usein *kuolleiksi yksiköiksi* (dead units). Tämän vuoksi neuroneiden välisiä yhteyksiä poistetaan, jos kyseinen yhteys ei ole aktivoitunut riittävän pitkään aikaan. [3]

Martinez ja muut [7] ehdottavat rapautumismuuttujan λ pienentämistä eksponentiaalisesti, esimerkiksi aloittamista lähtöarvosta 10 ja sen pienentämistä algoritmin oppimisaskelten myötä arvoon 0,01. Suuri rapautumismuuttujan λ alkuarvo johtaa suurempaan *kustannusfunktion* (cost function) arvoon, koska algoritmi vaikuttaa yhdellä oppimisaskeleella suurempaan määrään neuroneita. Kun λ:n arvoa aletaan pienentää algoritmin edetessä, myös kustannusfunktion arvo pienenee. Rapautumismuuttujan pienenemisen vauhti vaikuttaa myös osaltaan siihen, päästäänkö algoritmilla optimaalisiin tuloksiin, vai vain niiden lähelle. Tähän kuitenkin vaikuttaa myös esimerkiksi oppimisaskelten kokonaislukumäärä t_{max}. [7]

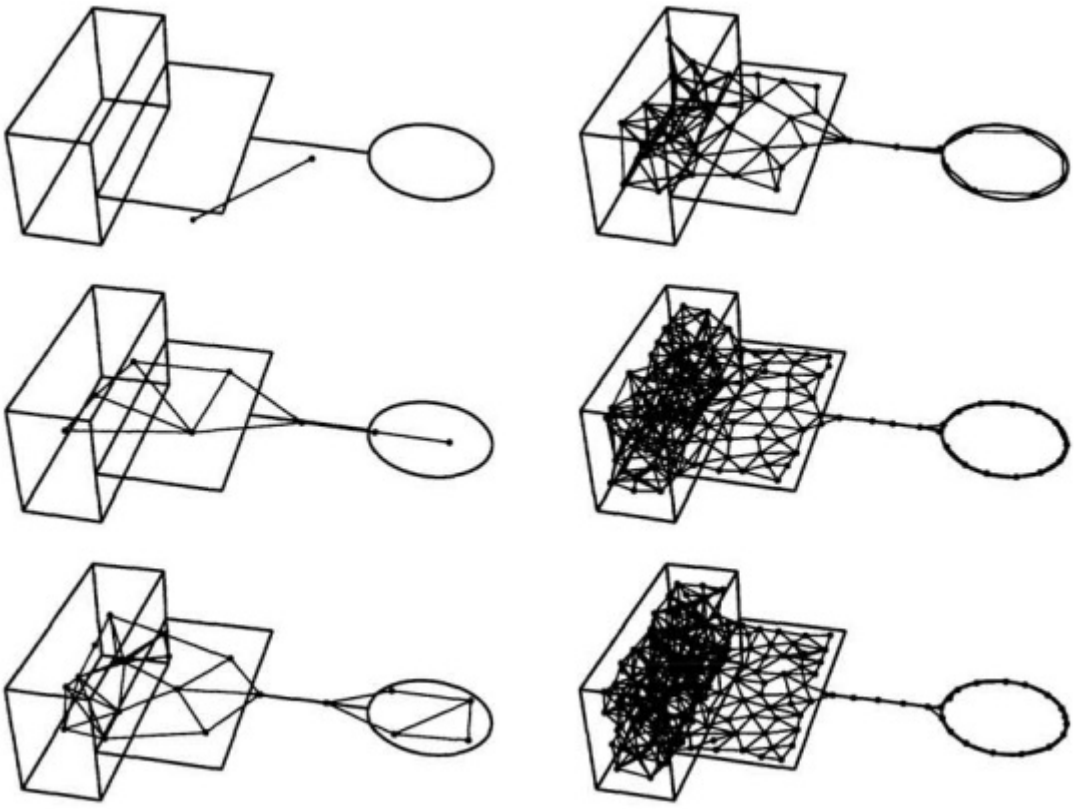
Neural gas -algoritmia on myös sovellettu käyttämään vakioarvoista rapautumismuuttujaa λ, jolloin sen ei anneta muuttaa arvoaan algoritmin edetessä ja pienentyä nollaan [12]. Parigi ja muut [12] vertailivat neuroverkkojen käyttäytymistä, kun neural gas -algoritmille annettiin eri suuruisia rapautumiskertoimia. He totesivat, että jopa pienellä rapautumiskertoimen arvolla neural gas levittäytyy hyvin syötealueelle. Kuitenkin suuremmilla arvoilla algoritmi suppeni tehokkaammin ja tasaisemmin. Rapautuumismuuttujalla λ suurempi arvo on vähintään 10, kun esimerkiksi arvolla 6 algoritmi hajaantui huomattavasti. [12]

Neural gas -algoritmia voidaan myös käyttää *aikasarjojen ennustamiseen* (time-series prediction). Ennustamisessa on tavoitteena ennakoida muuttujan x arvo lyhyen ajan päähän tulevaisuuteen [1]. Martinez ja muut [7] vertasivat neural gas -algoritmin ja k:n keskiarvon menetelmän suorituskykyä

aikasarjan ennustamisessa. He toteavat, että neural gas-algoritmilla päästään jopa optimaalisiin tuloksiin syöteavaruutta ositettaessa ja että neural gas -neuroverkko vaatii jopa kymmenen kertaa vähemmän synaptisia painoja saavuttaakseen saman eron ennusteessa [7].

5 Kasvava neural gas -algoritmi

Yhtenä neural gas -algoritmin heikkoutena voidaan pitää sitä, että optimaalisen parametrien pienenemisvauhdin määrittämiseksi lopullinen oppimisaskelten lukumäärä on tiedettävä etukäteen. Fritzke [3] kehitti neural gas -algoritmia, ja nimesi tämän uuden version kasvavaksi neural gas -algoritmiksi. Tässä algoritmissa yksikään parametri ei muuta arvoaan algoritmin toimimisen aikana, ja oppiminen ja kasvaminen on mahdollista mallin mukaisesti, kunnes jokin tietty suorituskriteeri täyttyy [3].



Kuva 3: Kasvavan neural gas -algoritmin vaiheet [3].

Kuvassa 3 havainnollistetaan kasvavan neural gas -algoritmin vaiheita. Aluksi on olemassa vain kaksi neuronia (ylävasen). Keskellä vasemmalla on tilanne 600 oppimisaskeleen jälkeen, jolloin neuroneita on jo enemmän. Seu-

raavissa kuvissa on tilanne 1800, 5000, 15000 ja lopulta 20000 oppimisaskeleen jälkeen. [3]

Kasvavan neural gas -algoritmin toimintaperiaate on samankaltainen neural gas -algoritmin kanssa. Neuroverkossa on joukko A neuroneita, joita kutsutaan myös solmuiksi. Jokaisella neuronilla $c \in A$ on neuroniin liittyvä painovektori $\mathbf{w}_c \in R^n$. [3]

Näitä neuroneita yhdistää joukko N yhteyksiä eli särmiä. Toisin kuin neural gas -algoritmilla, näillä yhteyksillä ei ole painoja, vaan niiden ainoa tarkoitus on määrittää neuroverkon topologista rakennetta. [3]

Merkittävä ero neural gas -algoritmiin on se, että n-ulottuvuuksellisia syötesignaaleja on mahdollisesti ääretön määrä, ja ne noudattavat jotakin tuntematonta tiheysfunktiota $P(\boldsymbol{\xi})$ [3]. Algoritmin myötä minimaalinen neuroverkko kasvaa, ja uusia neuroneita lisätään peräkkäin käyttäen vektorikvantisaatiotekniikkaa [11].

Kasvavan neural gas -algoritmin toimintavaiheet ovat seuraavanlaiset [3]:

0. Valitaan satunnaiset neuronit a ja b, joilla $\mathbf{w}_a, \mathbf{w}_b \in R^n$.

1. Muodostetaan syöte $\boldsymbol{\xi}$, joka noudattaa tiheysfunktiota $P(\boldsymbol{\xi})$.

2. Haetaan $\boldsymbol{\xi}$:tä lähin ja toiseksi lähin neuroni. Merkitään näitä s_1:llä ja s_2:lla.

3. Kasvatetaan neuronista s_1 lähtevien yhteyksien ikää.

4. Olkoon
$$\Delta error(s_1) = \|\mathbf{w}_{s_1} - \boldsymbol{\xi}\|^2,$$
syötesignaalin ja syöteavaruuden lähimmän neuronin välisen etäisyyden neliö.

5. Siirretään s_1 ja sen välittömät topologiset naapurit kohti $\boldsymbol{\xi}$:tä kertoimien ϵ_b ja ϵ_n verran, lopullisen etäisyyden muutoksen ollessa
$$\Delta\mathbf{w}_{s_1} = \epsilon_b(\boldsymbol{\xi} - \mathbf{w}_{s_1}) \text{ ja } \Delta\mathbf{w}_n = \epsilon_n(\boldsymbol{\xi} - \mathbf{w}_n),$$
kaikille s_1 välittömille naapureille n.

6. Jos neuroneiden s_1 ja s_2 välillä on yhteys, asetetaan yhteyden ikä nollaan. Jos yhteyttä ei ole olemassa, luodaan se.

7. Poistetaan yhteydet, joiden ikä on suurempi kuin a_{max}. Jos tämän jälkeen on neuroneita, joista ei lähde ainoatakaan yhteyttä, poistetaan myös nämä neuronit.

8. Jos tähän mennessä luotujen syötteiden lukumäärä on parametrin λ monikerta, luo uusi neuroni seuraavasti:

(a) Määritetään neuroni q, jolla on suurin kasaantunut virhe.

(b) Haetaan q:n välittömistä naapureista se neuroni f, jolla on suurin virhearvo, ja luodaan uusi neuroni r näiden puoliväliin:

$$\mathbf{w}_r = 0,5(\mathbf{w}_q + \mathbf{w}_f).$$

(c) Luodaan yhteydet yhdistämään uusi neuroni r neuroneihin q ja f. Poistetaan alkuperäinen yhteys q:n ja f:n väliltä.

(d) Pienennetään q:n ja f:n virhearvoja kertomalla ne vakiolla α. Alustetaan r:n virhearvo samaksi kuin q:n uusi virhearvo.

9. Pienennetään kaikkia virhearvoja kertomalla ne vakiomuuttujalla d.

10. Jos lopetusehto (esimerkiksi neuroverkon koko tai jokin suorituskyky) ei ole vielä täyttynyt, jatketaan kohdasta 1.

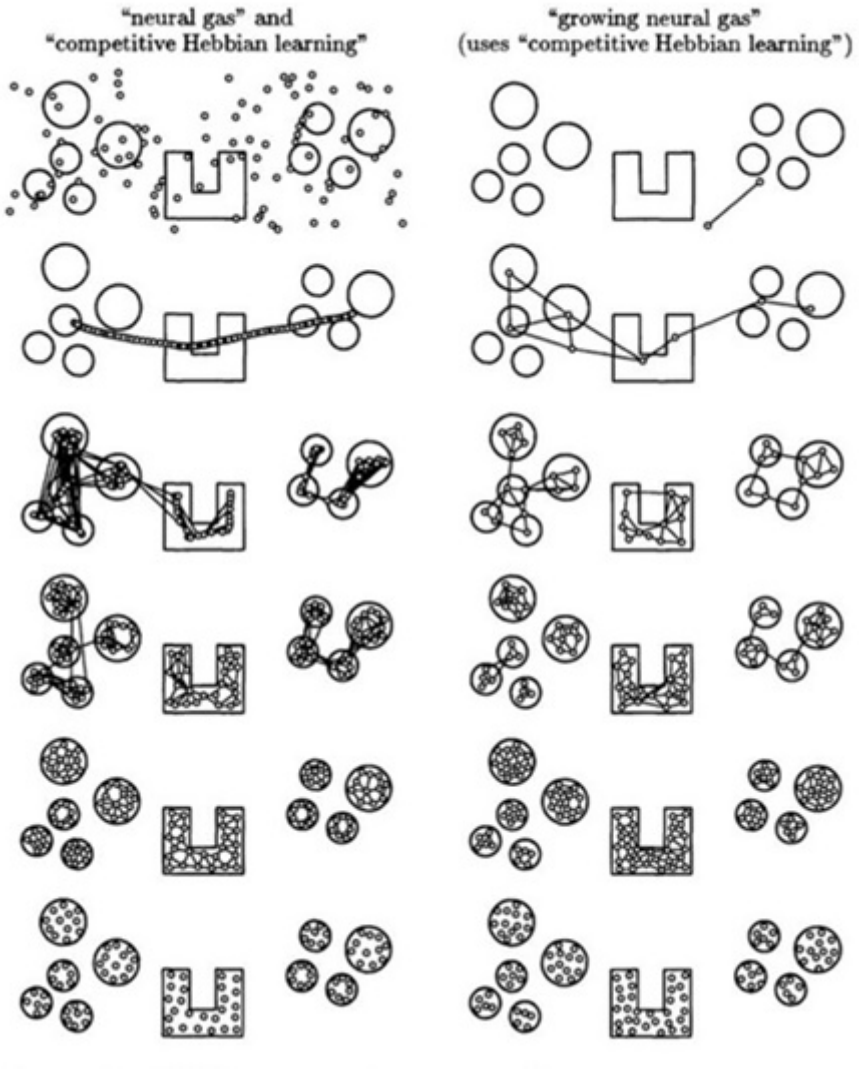
Algoritmissa oppimisaskel johtaa yleiseen liikkeeseen kohti aluetta, joista syötesignaalit tulevat. Lisäämällä ja poistamalla yhteyksiä, malli yrittää luoda ja seurata Delaunayn kolmiointia. Painovektoreiden mukautuminen tekee siitä liikkuvan kohteen. Vanhentuneiden yhteyksien poistaminen vaaditaan, jotta Delaunayn kolmiointi pysyy todenmukaisena. [3]

Fritzke [3] mainitsee, että kasvavan neural gas -algoritmin yksi tärkeä tavoite on ulottuvuuksien vähentäminen. Tarkoitus onkin löytää matalaulottuvuuksinen aliavaruus, joka sisältää suurimman osan tai kaiken syötedatasta [3]. Tällöin tuloksena on yksinkertaisempi esitys, mutta kaikki oleellinen informaatio on edelleen saatavilla.

6 Neural gas ja kasvavan neural gas -algoritmien vertailu

Yksi algoritmien huomattavimmista eroista on se, että kasvava neural gas -algoritmi aloittaa vain kahdella satunnaisesti valitulla neuronilla [3]. Neural gas -algoritmi sen sijaan tarvitsee heti alussa kaikki neuronit, vaikka niiden paikat ovatkin sattumanvaraisia [6]. Jos neural gas -algoritmilla halutaan sadan neuronin neuroverkko, on algoritmille annettava heti nämä sata neuronia. Kasvava neural gas -algoritmi sen sijaan pystyy kasvamaan itse, eikä neuroverkon neuronien lukumäärää tarvitse määrittää etukäteen. Fritzke [3] huomauttaa, että neuronien optimaalisen määrän löytäminen etukäteen voi olla haasteellista.

Kuva 4 näyttää vierekkäin neural gas ja kasvavan neural gas -algoritmien vaiheet, kun algoritmeja sovelletaan samanlaiseen syötedataan. Molempien algoritmien tuloksena on 100 neuronin neuroverkko. Alimmalla rivillä kumpikin algoritmi on suorittanut 10000 oppimisaskelta. [3]



Kuva 4: Neural gas ja kasvavan neural gas -algoritmien vertailu [3].

Algoritmit myös lopettavat toimintansa eri tavoin. Neural gas -algoritmille annetaan oppimisaskelten lukumäärä, ja kun tämä määrä askelia on suoritettu, algoritmi lopettaa toimintansa [7]. Kasvavalle neural gas -algoritmille voi sen sijaan antaa erilaisia lopetusehtoja, joiden täyttymisen algoritmi tarkistaa aina oppimisaskeleen lopussa. Tällainen ehto voi olla esimerkiksi neuroverkon koko tai jokin muu ominaisuus, jonka neuroverkko täyttää, jolloin algoritmi lopettaa toimimisensa. [3]

Neural gas -algoritmissa askelkoko ϵ ja rapautumismuuttuja λ pienentävät arvojaan algoritmin edetessä. Kasvavassa neural gas -algoritmissa sen sijaan niillä on vakioarvot. [3] Martinez ja Schulten [6] aloittivat esimerkiksi rapautumismuuttujan pienentämisen melko pienistä arvoista ($\lambda < 50$). Fritzke [3] sen sijaan testasi kasvaa neural gas -algoritmia jopa niinkin suurella

rapautumisvakiolla kuin 100.

7 Algoritmien käyttö kolmiulotteisiin objekteihin

Molempia algoritmeja on testattu muun muassa kolmiulotteisella Standford bunny -hahmolla [11, 12]. Parigi ja muut [12] testasivat neural gas -algoritmia kolmiulotteiseen malliin sekä pienenevällä rapautumismuuttujalla että vakioarvoisella rapautumismuuttujalla. Pienenevällä rapautumismuuttujalla neural gas -algoritmi pystyi rekonstruoimaan Stanford bunny -hahmon alkuperäisen muodon hyvin. Tämä kuitenkin vaati tietyt olosuhteet ja oikeansuuruiset parametrit. Kun rapautumismuuttuja λ muutettiin vakioarvoiseksi, algoritmi hävitti Stanford bunny -hahmon yksityiskohdat. Esimerkiksi jäniksen korvat eivät enää olleet kuopallaan, vaan korvien pinta oli tasainen. [12]

Orts-Escolano ja muut [11] testasivat kasvavaa neural gas -algoritmia samaan kolmiulotteiseen malliin. He vaikeuttivat neuroverkon toimintaa myös lisäämällä liikkeen kolmiulotteiselle mallille. He huomasivat, että käyttämällä vakioarvoista rapautumismuuttujaa, neuroverkko reagoi liian hitaasti kolmiulotteisen objektin liikkeisiin. Tämän takia he eivät käyttäneet kasvavan neural gas -algoritmin alkuperäistä versiota. He muuttivat rapautumismuuttujan vakioarvoisesta pienenevaksi, kuten neural gas -algoritmissa. [11]

Orts-Escolano ja muut [11] myös lisäsivät kolmiulotteisen ruudukon Stanford bunny -hahmon ympärille nopeuttaakseen algoritmin oppimisaskelta [11]. Koska lähimmän neuronin löytäminen on myös kasvavan neural gas -algoritmin aikaa vievin vaihe, he halusivat nopeuttaa sitä mahdollisimman paljon. Hakuavaruuden osittaminen ja indeksoiminen kolmiulotteisella ruudukolla nopeutti hakutuloksia huomattavasti. [11]

Toisessa tutkimuksessaan Orts-Escolano ja muut [10] testasivat kasvavaa neural gas -algoritmia kolmiulotteisiin hahmoihin, joita ei oltu esikäsitelty ja esimerkiksi hahmon taustaa ei oltu poistettu. He muokasivat algoritmia tälläkin kerralla, lisäten algoritmin painovektoreiden ulottuvuuksia, jotta algoritmi pystyisi ottamaan huomioon myös kolmiulotteiden mallin värit automaattisesti. Algoritmi suoritti oppimisaskeleet normaalisti, mutta niiden jälkeen algoritmi laski neuroneiden värit läheisimpien syötteiden väreistä. [10]

Toinen Orts-Escolanon ja muiden [10] tekemä muutos oli painovektoreiden luoman harvan rungon muuttaminen tiheämmäksi verkoksi. Tämä mahdollisti kasvojen kuvaamisen algoritmilla [10]. Orts-Escolano ja muut [10] halusivat välttää varsinaisen jälkiprosessoinnin. Heidän tavoitteenaan oli, että algoritmi pystyy käsittelemään kasvot ja hahmon värit itsenäisesti. He pääsivät melko hyviin tuloksiin erilaisilla kolmiulotteisilla malleilla. Osa niistä kuitenkin sisälsi aukkoja, koska algoritmi ei pystynyt reagoimaan aivan jokaiseen hahmon muotoon optimaalisesti. Osa näistä aukoista johtui myös

neuroverkon painovektoreiden maksimieliniästä. Sen kasvattaminen ei kuitenkaan parantaisi tuloksia, koska painovektoreiden vanheneminen varmistaa sen, että algoritmi mukautuu hyvin ja että neuroneiden väliset yhteydet ovat oikeasti merkittäviä. [10]

8 Yhteenveto

Koneoppimiseen on haettu inspiraatiota ihmisen oppimisesta. Tätä on sovellettu muun muassa neuroverkkojen kehityksessä. Neuroverkot ovat nykypäivänä oleellinen osa koneellista tiedonkäsittelyämme. Neuroverkot pystyvät oppimaan ympäristöstään ja luomaan tarkempia tuloksia. Ne mahdollistavat myös sen, että ihmisen ei enää tarvitse valvoa ja ennalta määrätä kaikkea, mitä järjestelmän tulee pystyä tekemään.

Neural gas -algoritmi toteuttaa neuroverkon ohjaamatonta oppimista. Algoritmi pystyy pienillä ennakkotiedoilla ja -määrityksillä käsittelemään saamansa syötedatan tehokkaasti, luoden optimaalisen esityksen syötedatasta. Neural gas -algoritmi myös saavuttaa pieniä virhevääristymiä.

Neural gas -algoritmin käyttämä vektorikvantisaatiotekniikka esittää suuren määrän syötevektoreita pienellä määrällä tulostevektoreita. Syöteavaruuteen sijoitetaan painovektoreita, joiden avulla syötedatan sisältämä tieto pystytään esittämään yksinkertaisemmin.

Algoritmi aloittaa satunnaisesti sijoitetuilla painovektoreilla, joita mukautetaan aina uuden syötevektorin myötä. Painovektorit luovat yhteyksiä keskenään, kun kaksi painovektoria ovat syötevektoria lähin ja toiseksi lähin painovektori. Painovektoreiden ja syötevektorin välisiä virhemuuttujan arvoja vertailtaessa muodostuu Voronoin monikulmio. Sen avulla pystytään luomaan Delaunauyn kolmiointi, joka kuvastaa alueita syöteavaruudessa, jotka pystyvät luomaan yhteyksiä keskenään.

Algoritmin rapautumismuuttuja ja askelkoko määrittävät mukautettavien painovektoreiden lukumäärän sekä muutoksen suuruuden. Molemmat aloittavat suurehkoilla arvoilla, joita pienennetään algoritmin edetessä. Vaikka muuttujat ovat hyvin erisuuruisia, ne pienenevät samassa suhteessa toisiinsa. Algoritmin loppuvaiheissa rapautumismuuttuja on pienentynyt niin paljon, että ainoastaan syötevektorin lähintä painovektoria mukautetaan.

Neural gas -algoritmi vaatii edelleen jonkin verran tietoa syötedatan topologiasta, jotta oikea määrä neuroneita pystytään asettamaan neuroverkkoon ennen algoritmin suorittamista. Kasvava neural gas -algoritmi ratkaisee tämän ongelman sillä, että neuroverkko aloittaa vain kahdella neuronilla, ja algoritmi suorittaa neuroneiden lisäystä aina tarvittaessa.

Kasvava neural gas -algoritmi eroaa neural gas -algoritmista myös siinä, että sen rapautumismuuttuja ja askelkoko eivät muuta arvojaan algoritmin suorituksen aikana. Sen sijaan ne pysyvät vakioarvoisina läpi prosessin, ja esimerkiksi rapautumismuuttujan arvo saattaa olla hyvinkin suuri.

Molemmilla algoritmeilla on tehty kaksi- ja kolmiulotteisten objektien analysointia. Varsinkin kasvava neural gas -algoritmi on antanut lupaavia tuloksia jopa liikkuvien kolmiulotteisten objektien seuraamisessa. Kasvavaa neural gas -algoritmia on myös käytetty robottien puheen analysoinnissa ja luomisessa. Algoritmilla onkin monia hyödyllisiä käyttökohteita. Algoritmin käyttöä näissä ja muissa tilanteissa tuleekin tutkia lisää, ja algoritmia parantaa tarvittaessa toimimaan entistä tehokkaammin ja tarkemmin.

Viitteet

[1] Christopher M. Bishop. 1995. *Neural Networks for Pattern Recognition.* Clarendon Press.

[2] Pedro Domingos. 2012. A few useful things to know about machine learning. *Commun. ACM* 55, 10, 78-87.

[3] Bernd Fritzke. 1995. A growing neural gas network learns topologies. *Adv. Neural Inf. Process. Syst.* 7, 625-632.

[4] Simon Haykin. 1994. *Neural Networks: A Comprehensive Foundation.* Macmillan College.

[5] Anil K. Jain, and Jianchang Mao. 1996. Artificial neural networks: a tutorial. *Computer* 29, 3, 31-44.

[6] Thomas Martinetz, and Klaus Schulten. 1991. A "neural gas" network learns topologies. In: K. Mäkisara, O. Simula, J. Kangas, and T. Kohonen (eds.), *Artificial Neural Networks.* Elsevier Science Publishers B.V., 397-402.

[7] Thomas M. Martinetz, Stanislav G. Berkovich, and Klaus J. Schulten. 1993. "Neural-gas" network for vector quantization and its application to time-series prediction. *IEEE Trans. Neural. Netw.* 4, 4, 558-569.

[8] Thomas Martinetz, and Klaus Schulten. 1994. Topology representing networks. *Neural Networks* 7, 3, 507-522.

[9] Ryszard S. Michalski, and Yves Kodratoff. 1990. Research in machine learning. In: Yves Kodratoff and Ryszard Michalski (eds.), *Machine Learning: An Artificial Intelligence Approach, Volume 3*, Morgan Kauffmann Publishers, Inc., 3-30.

[10] Sergio Orts-Escolano, Jose Garcia-Rodriguez, Vicente Morell, Miguel Cazorla, and Juan Manuel Garcia-Chamizo. 2014. 3D colour object reconstruction based on growing neural gas. In: *Proc. of the 2014 International Joint Conference on Neural Networks (IJCNN)*, 1474-1481.

[11] Sergio Orts-Escolano, Jose Garcia-Rodriguez, Vicente Morell, Miguel Cazorla, Marcelo Saval, and Jorge Azorin. 2015. Processing point cloud sequences with growing neural gas. In: *Proc. of the 2015 International Joint Conference on Neural Networks (IJCNN)*, 1-8.

[12] Giacomo Parigi, Andrea Pedrini, and Marco Piastra. 2015. Some further evidence about magnification and shape in neural gas. In: *Proc. of the 2015 International Joint Conference on Neural Networks (IJCNN)*, 1-8.

[13] Aree Witoelar, Michael Biehl, Anarta Ghosh, and Barbara Hammer. 2008. Learning dynamics and robustness of vector quantization and neural gas. *Neurocomputing* 71, 7-9, 1210-1219.

Robotit osana pöytätennistä

Riku Autio

Tiivistelmä.

Robotit ja tietotekniikka ovat tulleet osaksi pöytätennistä viimeisten muutamien vuosikymmenten aikana. Tutkielmassa tarkastellaan lajin ja robotiikan kehityksen vaiheita ja tämän myötä pöytätennistä pelaavien robottien toimintaa. Lisäksi esille nostetaan robotteihin liittyviä ongelmia niin tekniikan kuin lajinkin näkökulmasta. Tutkielmassa pyritään antamaan vastaus kysymykseen, pystyykö robotti pelaamaan pöytätennistä ihmisen kanssa ja samalla tasolla kuin ihminen. Päävertailukohtana käytetään huippupelaajia, jotka pelaavat pöytätennistä ammatikseen.

Pöytätennistä pelaava robotti koostuu havainnointi- ja hallinnointijärjestelmästä sekä mekaanisesta rakenteesta. Robotin toiminnan perustana toimii konenäkö. Ongelmana pöytätennisrobottien käytössä on muun muassa se, että ne eivät pysty huomioimaan kierteitä pallon liikkeessä. Ongelmista huolimatta ei ole poissuljettua, etteivätkö tulevaisuudessa ihmiset voisi harjoitella ihmisten lisäksi myös pöytätennistä pelaavien robottien kanssa.

Tulevaisuudessa roboteista on mahdollisesti tulossa yhä merkittävämpi osa pöytätennistä, jos kehitys saadaan jatkumaan. Kehityksen jatkumiselle ei ole estettä, kunhan suurimmat ongelmat onnistutaan poistamaan ja yhä paremmin pystytään osoittamaan robotin tuomat hyödyt lajille ja sen kehitykselle.

Avainsanat ja -sanonnat: pöytätennis, robotiikka, havainnointijärjestelmä, stereonäkö, hallinnointijärjestelmä.

1. Johdanto

Pöytätennis on laajalti harrastettu nopeatempoinen laji, josta tuli olympialaji vasta vuonna 1988 Soulin kisoissa [Olympic.org]. Alkeellisista ajoista on tultu tähän päivään ja lajin pariin on tuotu koko ajan enemmän ja enemmän teknologiaa. Näistä yhtenä esimerkkinä voidaan pitää robottien mukaantuloa. Vaikka pöytätenniksestä tuli olympialaji varsin myöhään, tutkimuksia siitä, pystyisikö robotti pelaamaan ihmisen kanssa pöytätennistä, on tehty jo 1980-luvun alusta lähtien [Liu *et al.* 2014].

Tutkimukseni tarkoitus on selvittää, kuinka pitkälle pöytätennistä pelaavat robotit ovat jo kehittyneet. Samalla tulen pohtimaan, voisiko robotteja hyödyntää aktiivisessa harjoittelussa ja pystyvätkö ne pelaamaan pöytätennistä ihmisten kanssa ja yhtä hyvin kuin ihmiset. Selvitän pöytätennistä pelaavan robotin

toimintamenetelmiä ja rakennetta. Merkitykselliseksi tutkimuksen tekee robottien mahdollisuus tuoda harjoitteluun monipuolisuutta ja toisaalta vielä kehitettävien asioiden havaitseminen.

Käyn luvussa 2 tarkemmin läpi pöytätenniksen historiaa ja lajin ominaisuuksia sekä välineiden kehittymistä. Tämän jälkeen luvussa 3 tarkastelen robotteja yleisellä tasolla ja annan niille määritelmän, jotta luvussa 4 päästään tarkastelemaan pöytätennistä pelaavien robottien rakennetta ja toimintamenetelmiä. Neljännessä luvussa käyn läpi pöytätennisrobottien kehityksen vaiheita selventääkseni, miten nykytilanteeseen ollaan päästy. Robottien osakomponenttien ominaisuuksia on paranneltu, jotta robotti pystyisi yhä aidommin pelaamaan pöytätennistä [Liu *et al.* 2014]. Lisäksi esittelen myös uusimpia tällä hetkellä käytössä olevia rakenteita ja toimintamenetelmiä, joita robotti käyttää pystyäkseen pelaamaan pöytätennistä tämän hetkisellä tasollaan.

Lajin ja robotiikan tuntemus auttaa hahmottamaan pöytätennistä pelaaviin robotteihin liittyviä ongelmia, joita esittelen luvussa 5. Tarkastelen, miten välineiden kehittyminen on lisännyt haasteita robottien toiminnalle ja toisaalta mitä haasteita marginaalinen aika aiheuttaa. Luvussa 6 puolestaan esittelen pöytätennistä pelaavien robottien tulevaisuuden näkymiä. Nostan esiin omia ajatuksiani siitä, miten kehitys auttaa lajia ja kuinka robotteja voisi hyödyntää harjoittelussa.

2. Pöytätennis lajina ja sen historiaa

Alkuun pöytätennistä ei mielletty edes urheiluksi, vaan sitä pelattiin päivällisen jälkeisenä ajanvietteenä. Pöytätennistä pelattiin ensimmäisen kerran 1880-luvulla Englannissa, jolloin pallona käytettiin samppanjapullon korkkia ja mailoina sikarilaatikon kansia [Olympic.org]. Kuvassa 1 on esitetty lajin alkuaikoja Englannista.

Kuva 1. Pöytätenniksen alkuajoilta Englannista [Killerspin.com].

Noista ajoista pöytätennis on kehittynyt hurjasti. Itse asiassa pöytätennis on nykyään osallistujamäärältään suurin urheilulaji kaikki lajit huomioon ottaen [olympic.org]. Tätä voidaan pitää jopa yllättävänä tietona, kun ottaa huomioon, että esimerkiksi jalkapallo on todella iso laji maailmalla. Suurimmat vaikuttavat tekijät ovat Kiinan suuri harrastajamäärä ja lajin merkittävä suosio muissakin Aasian maissa.

Pöytätennis on nopea ja tekninen laji, joka voidaan jakaa fyysiseen ja mentaaliseen osaan. Niin kuin monessa muussakin urheilulajissa usein henkinen kovuus ratkaisee lopputuloksen tiukassa paikassa. Viimeisten vuosien aikana Kiina on hallinnut kilpapöytiä lähes ylivoimaiseen tapaan ja tämän hetken parhaana pelaajana voidaankin kiistattomasti pitää Kiinan Ma Longia, joka voitti 2015 maailmanmestaruuden, 2016 olympiakultaa ja on tämän hetken maailmanlistan ykköspelaaja [ittf.com 2017a].

2.1. Mailojen kehitys merkittävä tekijä lajin kehittymisessä

Mailat ovat kehittyneet todella paljon ja muuttaneet lajia samalla entistä enemmän huippu-urheiluksi ja samalla myös paljon fyysisemmäksi lajiksi. Alkuun pelattiin suunnilleen välineellä kuin välineellä, joka vain sattui käteen osumaan. Mailat saattoivat olla erikokoisia ja erinäköisiä. Tosin ei nykypäivänäkään mailan kokoa tai muotoa rungon osalta ole millään tavalla rajoitettu. Silti kaikki käyttävät hyvin samankokoista mailaa, koska sen on todettu olevan paras. Rungolla siis tarkoitetaan pöytätennismailan pääasiassa puista osaa, jonka päälle kumit liimataan. Runko koostuu lavasta ja kahvasta. Kahva on rungon osa, josta pidetään kiinni ja lapa on puolestaan osa, jolla palloa pyritään lyömään. Jotta pingismailasta saadaan pelattava, niin rungon lapaan liimataan kummallekin puolelle kumi. Kumien tulee olla eri väriset ja värien tulee olla musta ja punainen. Kumeilla aikaan saadaan ja toisaalta mahdollistetaan pöytätennikselle ominaiset kierteet. Kumeja rungon päällä punnitaan ja mitataan nykyään monilla tavoilla ja välineillä. Joskus tuo tuntuu liialliselta pikkutarkkuudelta, mutta todellisuudessa millinkymmenyksenkin paksuusero kumien välillä vaikuttaa kumin ominaisuuksiin merkittävästi.

Kehitys alkoi näkyä 1900-luvun alussa, kun mailoihin puun päälle laitettiin hiekkapaperia, jotta palloon voitaisiin saada jonkinlaista kierrettä aikaiseksi [Allabouttabletennis.com]. Kehitys jatkui, kun 1920-luvulla puumailojen pintaan laitettiin ensimmäistä kertaa kumit. Ne olivat lyhytnäppyläiset kumit, joilla kuitenkin edelleen pystyi pelaamaan lähes ainoastaan puolustuspeliä. Vasta 1960-luvulla mailoihin alettiin laittaa nykyisinkin vakioituneita 2 mm paksuja hyökkäyskumeja, jotka lasketaan olevan optimaalisen paksuisia hyökkäyspeliin. Aiemmin kokeiltiin paksumpiakin kumeja, mutta ne eivät lopulta osoittautuneet kovin hyviksi. Viimeisimpänä muutoksena pidetään tuoreli-

mauksen lopettamista vuonna 2007 eli kumit alettiin liimata vain vesipohjaisilla liimoilla, jotka eivät sisällä lainkaan haitallisia kemikaaleja [Allabouttabletennis.com].

Vaikka kumien kehitys onkin ollut suurin tekijä mailojen muutoksessa, niin ei sovi unohtaa, että rungotkin ovat mennet eteenpäin ja puun joukkoon on alettu lisätä esimerkiksi hiilikuitua, jotta maila nopeutuisi ja toisaalta myös kevenisi. Painopistettä on myös pyritty tuomaan enemmän mailan kärkeä kohti. Nykyään on niin monenlaisia mailoja tarjolla, että on lähes mahdoton edes tietää kaikkia erilaisia kombinaatioita.

2.2. Pallojen muutos tuonut oman lisänsä lajiin

Pöytätennispallot ovat väriltään oransseja tai valkoisia. Vaikka värejä onkin yhä kaksi, lähes kaikki kilpailut nykyään pelataan valkoisella pallolla. Ennen saatettiin pelata oranssilla pallolla hallin ollessa hyvin vaalea. Nykyisin monet kansainväliset kisat pelataan punaisella tai jonkin muun värillisellä gerfloralustalla, joten valkoinen pallo on valikoitunut pelipalloksi. Tästä syystä kaikki valmistajat panostavat enemmän valkoisten pallojen valmistukseen kuin oranssien. Palloissa voidaan huomata pieniä eroja merkkien välillä. Toiset saattavat olla esimerkiksi hieman nopeampia kuin toiset.

Aivan alkua lukuun ottamatta pallojen osalta suurimmat muutokset pöytätenniksen historiassa ovat varmasti pallon koon kasvaminen ja materiaalin muuttuminen selluloidista muoviksi. Pallon halkaisija siis kasvoi aluksi 38 millimetristä 40 millimetriin. Lopulta pallo kasvoi vielä nykyiseen kokoonsa 40+ millimetriin, kun pallot muuttuivat selluloidista muovisiksi. Vaikkei heti uskoisikaan, niin kahden millimetrin muutos pallon halkaisijassa vaikuttaa peliin huomattavasti. Pelaajilta alettiin vaatia yhä enemmän voimaa ja toisaalta mailoilta nopeutta ja kimmoisuutta.

3. Robotiikasta yleisesti

Robotti nousi terminä esiin Capekin veljesten kirjoituksissa 1900-luvun alkupuolella [Hockstein *et al.* 2007]. Kirjoitukset olivat tuohon aikaan enemmänkin kannanotto nopeaan kehitykseen. Capekin veljekset halusivat tuoda kantansa esiin kirjoittamalla, kuinka robotit tulisivat tulevaisuudessa valtaamaan ihmiskunnan. Robotit siis alun perin esiintyivät vain kirjoituksissa, kunnes niitä alettiin myös toteuttaa. Ensimmäisen ihmistä muistuttavan autonomisesti toimivan robotin kehitti 1900-luvun puolivälissä Englannissa William Grey Walter [Bladin 2006].

Robotiikasta ja roboteista on tullut tietotekniikan lisääntymisen myötä entistä merkittävämpi osa monia aloja. Robotit ovat jopa vieneet osan ihmisten

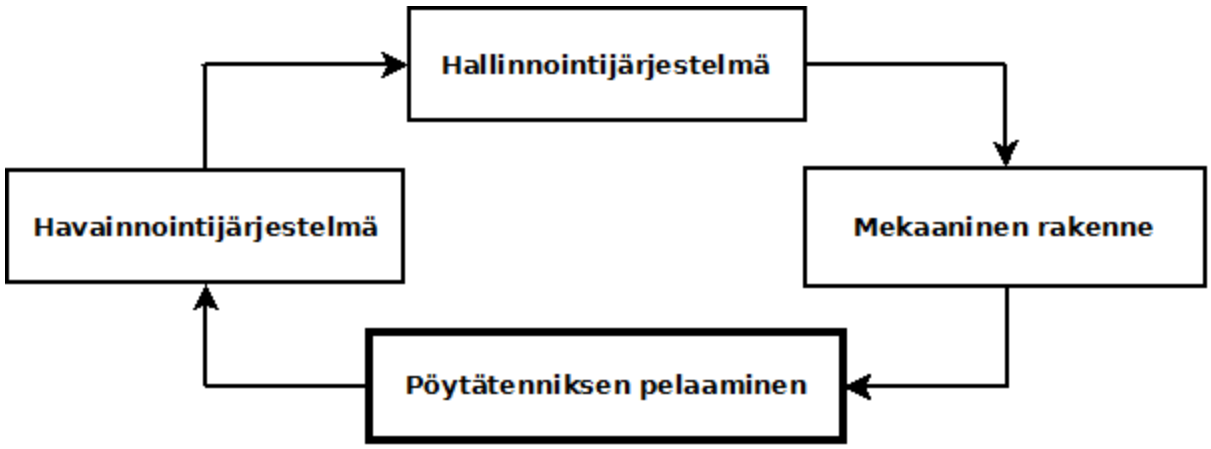
työpaikoista tekemällä työt heidän puolestaan nopeammin ja tarkemmin. Latomben [1991] mukaan robotiikan tärkein tavoite on luoda itsenäisesti toimivia robotteja, jotka suorittavat hyvin tarkkaan määriteltyjä, vaikeita ja yksityiskohtaisia toimintoja [Latombe 1991; Frese and Hirschmüller 2015].

Robotteja on monenlaisia ja monen tasoisia ja niistä kaikista tulen tarkastelemaan pöytätennikseen liittyviä robotteja. Robotit eivät vielä pöytätennikses-sä ole vallanneet pöytien toista puolta harjoituksissa. Tähän paneudun kuitenkin tarkemmin luvussa 6. Pöytätennisrobotit voidaan karkeasti ottaen jakaa kahteen ryhmään eli pöytätennistä pelaaviin robotteihin ja palloja heittäviin robotteihin. Pelaavalla robotilla tarkoitetaan robottia, joka pystyy palauttamaan ihmisen sitä kohti lyömiä palloja. Palloja heittävä robotti puolestaan ainoastaan heittää halutulla kierteellä ja voimakkuudella palloja tiettyihin pisteisiin pöydällä. Tämä robotti ei siis mitenkään reagoi ihmisen lyöntiin, vaan heittää ainoastaan uuden pallon. Tarkastelen kuitenkin ainoastaan pöytätennistä pelaavia robotteja.

4. Pöytätennistä pelaavan robotin toiminta

Pelaavien pöytätennisrobottien tutkiminen on lisääntynyt viimeisten vuosien aikana. Siitä on jopa tullut todella suosittua varsinkin kiinalaisten tutkijoiden keskuudessa [Wang *et al.* 2012]. Robottien kehittäminen koetaan Wangin ja muiden [2012] mukaan kiinnostavaksi ja monisävyiseksi. Kiinassa on erikseen järjestetty robottien välisiä pöytätenniskilpailuja. Tämä on varmasti osaltaan edistänyt pöytätennisrobottien kehitystä.

Pöytätennisrobotin toiminta voidaan jakaa pääpiirteittäin kolmeen osa-alueeseen, jotka ovat havaintojärjestelmä, hallinnointijärjestelmä ja mekaaninen rakenne [Acosta *et al.* 2003]. Näiden saumattomasta yhdistelmästä syntyy ihmisen kanssa pöytätennistä pelaava robotti. Toiminta perustuu sykliin, joka on kuvattu kuvassa 2. Pääidea on, että ihminen lähettää pallon kohti robottia. Tämän jälkeen havainnointijärjestelmä havainnoi palloa ja antaa hallinnointijärjestelmälle tarkan tiedon pallon sijainnista. Hallinnointijärjestelmä puolestaan välittää käskyn robotille ja mekaaninen rakenne toteuttaa lyönnin, jolla pallo palautuu ihmisen pöytäpuolelle. Eri vaiheita ja osa-alueiden toimintaa on pyritty kehittämään jatkuvasti. Tarkastelen näitä osa-alueita ja niiden kehitystä tarkemmin kohdissa 4.1, 4.2 ja 4.3.



Kuva 2. Pöytätennisrobotin toiminnan perusta [Acosta *et al.* 2003].

4.1. Havainnointijärjestelmä

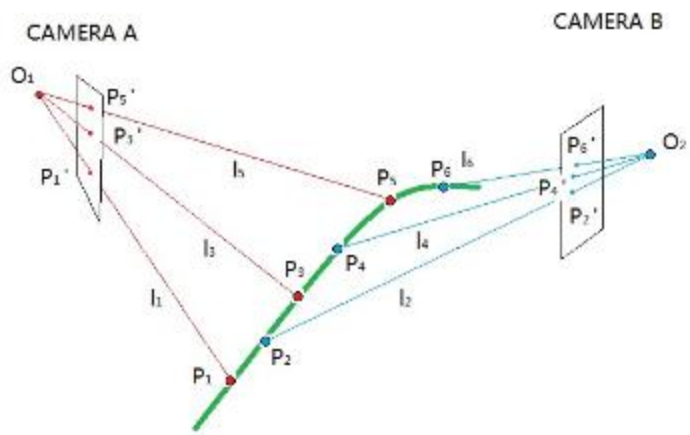
Pääasiassa palloa havainnoidaan kahdella erillisellä kameralla, jotka kuvaavat pelitapahtumaa eli käytännössä pallon sijaintia ja lentorataa tietyn matkan päästä kahdesta eri kulmasta. Kahdella kameralla pyritään kattamaan koko pöydän alue. Näin toimi myös Andersson luodessaan ensimmäisen ihmisen kanssa pöytätennistä pelaavan robotin [Liu *et al.* 2013a;b]. Kahta kameraa tulee käyttää, jotta voidaan mahdollistaa stereonäkö. Stereonäöllä tarkoitetaan kykyä havainnoida kolmiulotteisia asioita, esineitä ja rakenteita. Tämä puolestaan mahdollistaa etäisyyksien hahmottamisen. Havainnointijärjestelmästä on kuitenkin monenlaisia versioita, joista käyn muutamia läpi. Kamerat ovat kehittyneet ja nopeutuneet historian saatossa. Osa käytetyistä tekniikoista on toisia tarkempia ja sen myötä usein myös kalliimpia. Yleissääntönä voidaan kuitenkin pitää, että ilman kameraa robotilla ei ole silmiä, joilla havainnoida palloa.

Alkuun ajateltiin, että kahta kameraa käytettäessä tulee varmistua, että kuvat otetaan samaan aikaan. Tätä varten tarvitaan erillinen tahdistin [Acosta *et al.* 2003]. On kuitenkin selvää, että jos kuvat synkronoivasta tahdistimesta päästäisiin eroon, niin kuvien käsiteltävä aika pienenisi merkittävästi. Acosta ja muut [2003] kehittivät tähän oman ratkaisunsa. He käyttivät vain yhtä kameraa, jolloin tahdistinta ei tarvittu. Vain yhtä kameraa käytettäessä pallon sijainti tulee laskea kuvista pallon ja varjon avulla. [Acosta *et al.* 2003] Ratkaisun etuina voidaan pitää sen halpuutta, mutta toisaalta nopeus ja tarkkuus eivät puolestaan riittäneet vastaamaan todellista tilannetta pöytätenniksen pelaamisesta. Lisäksi valaistuksen merkitys on todella iso, jotta varjo erottuu pöydänpinnasta riittävän selvästi. Pöydälle saattaa myös helposti syntyä katvealueita, joissa varjoa ei kunnolla voida havaita.

Tahdistimen poistamisesta päätettiin hetkellisesti luopua, koska tulokset olivat heikkoja. Kahden kameran synkronointiin käytettiin kameroiden tulo- ja

lähtösignaaleja [Yang *et al.* 2011]. Näin kuvat saatiin melko nopeasti synkronoituna tietokoneelle. Yang ja muut [2011] tutkivat kahden robotin välistä pöytätenniksen pelaamista, jolloin havaintojärjestelmään lisättiin kolmas kamera. Samaa menetelmää voitaisiin käyttää ihmisen kanssakin pelaamiseen. Kolmannen kameran on tarkoitus tarkkailla robotin sijaintia ja välittää tiedot tietokoneelle. Näin tietojen avulla robotin tulisi itse liikkua oikeaan kohtaan, eikä ainoastaan liikuttaa kättään lyödäkseen palloa.

Liu ja muut [2014] ratkaisivat kameroiden synkronointiin liittyvän ongelman. Kehitysaskeleen myötä kameroita ei enää tarvinnut synkronoida keskenään, jotta saataisiin kahdesta samanaikaisesta kuvasta muodostettua 3D-kuva. Uusi 200 kehystä sekunnissa ottava järjestelmä tuo robottien kyvyn pelata pöytätenistä paljon lähemmäs lajin todellista nopeutta. Aikaisemmat kamerat olivat pystyneet ottamaan vain noin 100 kehystä sekunnissa. Liun ja muiden [2014] kehittämän algoritmin avulla kamerat lähettivät ottamansa kuvan prosessoituna hallinnointijärjestelmälle. Hallinnointijärjestelmässä kuvat laitetaan peräkkäin, mistä muodostetaan 3D-kuvaus pallon lentoradasta. [Liu *et al.* 2014] Tätä on havainnollistettu kuvassa 3.



Kuva 3. Kameroiden muodostama 3D-lentorata [Liu *et al.* 2014].

Kaksi kameraa kuitenkin edelleen tarvitaan, ja sen on todettu tällä hetkellä olevan paras tapa havainnoida palloa ja muodostaa lentoradan kuvaus. Uuden algoritmin myötä myös aikaa onnistuttiin säästämään, kun vältettiin kuvien edestakainen lähettely niiden synkronoimiseksi. Liu ja muut [2014] osoittivat myös, että uusi havainnointitapa on vakaampi ja onnistumisprosentti suurempi aiempiin tapoihin verrattuna. Uudella 3D systeemillä pystyttiin välttämään synkronoinnin epäonnistumisesta aiheutuvat virheet ja pallon lentoradan vääristymät.

Kamerat erottavat pallon pöydästä värin perusteella. Kameroiden tehtävä on erotella kuvista pikselit väreittäin ja muodostaa näin havainto pallosta. Pal-

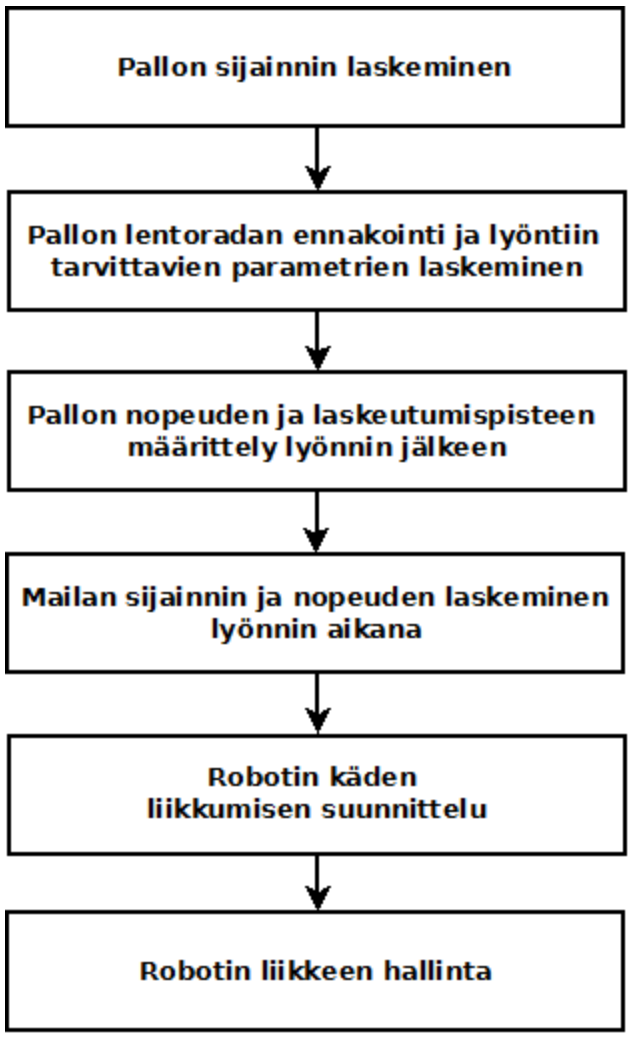
lon koordinaatit tulee koko ajan lähettää hallinnointijärjestelmään päivitettynä, jotta robotti osaa tehdä tarvittavia toimenpiteitä pallon lyömiseksi. Ilman tämän vaiheen onnistumista robotti ei onnistu lyömään palloa oikeasta kohdasta oikeaan aikaan. Jiang ja muut [2017] ovat kehitelleet algoritmin, jonka avulla kohteita voitaisiin havaita niiden värinän perusteella. Tätä algoritmia ja kuvausjärjestelmää voisi mahdollisesti käyttää myös pöytätennistä pelaavan robotin yhteydessä. Järjestelmä pystyy todella tarkkaan nopeasti määrittämään värisevän kappaleen sijainnin käyttämällä jopa 1000 kehystä sekunnissa ottavaa kameraa [Jiang *et al.* 2017]. Paljaalla silmällä värinää ei tietenkään voi havaita, mutta tällaisella tekniikalla se olisi mahdollista. Reaaliaikaisuus paranisi ja tämän myötä myös tulokset olisivat tarkempia. On sanomattakin selvää, mitä nopeammin ja tarkemmin robotille saadaan kuvaus lentoradasta, sitä enemmän jää aikaa suunnitella ja toteuttaa lyönti.

4.2. Hallinnointijärjestelmä

Havainnointijärjestelmän tiedot tulee jotenkin välittää robotille. Tämä on hallinnointijärjestelmän yksi päätehtävistä. Toinen merkittävä tehtävä on laskennan toteuttaminen. [Acosta *et al.* 2003] Hallinnointijärjestelmä on siis suurelta osin vastuussa robotin toiminnan oikeellisuudesta. Toisaalta ilman tarkkoja havaintoja ja kuvia tarvittavia liikkeitä ei voida määrittää.

Acostan ja muiden [2003] mukaan kontrolloinnin tasoja voidaan määrittää kaksi, matalan tason kontrollointi tai korkean tason kontrollointi. Matalan tason kontrolloinnilla tarkoitetaan sitä, että pallo saadaan palautettua tiettyyn pisteeseen vastustajan puolelle. Korkean tason kontrolli puolestaan tarkoittaa pallon tarkempaa sijoittamista vastustajan pöytäpuolikkaalla. [Acosta *et al.* 2003] Näin ollen pallo ei tulisikaan aina samaan kohtaan, vaan paikkaa pystyttäisiin vaihtamaan lyömällä esimerkiksi kaksi palloa keskelle pöytää ja yksi kämmennurkkaan. Todellisen harjoittelun kannalta paikan vaihtaminen on tärkeää.

Jotta robotti pystyisi kameroiden ottamien kuvien perusteella lyömään palloa onnistuneesti, tulee tietokoneen välissä suorittaa monta tehtävää. Tehtävät on kuvattu karkeasti kuvassa 4.



Kuva 4. Hallinnointijärjestelmän tehtävät [Yang *et al.* 2010].

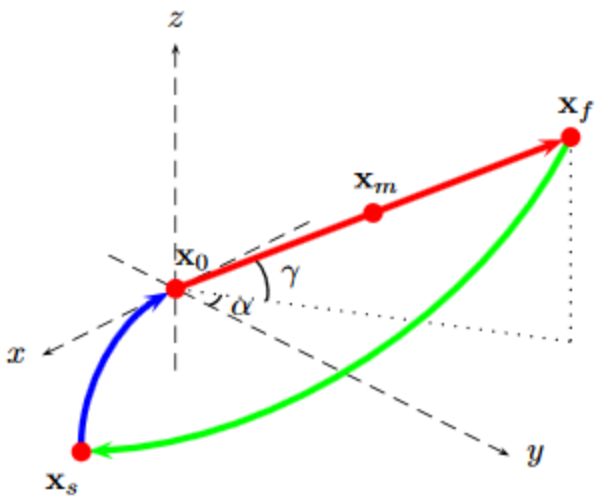
Aluksi koneen tulee määrittää pallon sijainti 3D-kuvana, josta pallon lentorataa voidaan lähteä ennustamaan. Kun lentorata on onnistuneesti saatu kuvattua 3D-muodossa, voidaan määrittää lyöntiin tarvittavat parametrit. Parametreja ovat esimerkiksi nopeus, jolla pallo lentää kohti robottia ja aika, joka pallolla kestää lentää kohtaan, jossa robotin tulee palloa lyödä.

Kun robotti on saanut tarkan kuvan pallon sijainnista, lentoradasta, nopeudesta ja ajasta, voidaan määrittää haluttu nopeus, jolla pallon tulisi osua vastapuolen kenttään. Lisäksi määritetään sijainti, johon pallon tulisi lyönnin jälkeen osua. Tämä voidaan siis määrittää vaihtuvaksi korkean tason kuvauksella tai samaksi yhdeksi tietyksi pisteeksi matalan tason kuvauksella, jotka jo aiemmin mainittiin.

Seuraavaksi tietokone laskee mailan sijainnin ja nopeuden koko lyönnin ajalta. Kun tarvittavat tiedot on kerätty muuttujiin, voidaan suunnitella robotin käden liikerata, minkä jälkeen liike toteutetaan hallitusti suunnitelman mukaan. Matsushima ja muut [2005] esittävät, että robotin mailan liikuttamispää-

tös voidaan perustaa robotin itseoppimiseen. Tätä samaista oppimisperustaista menetelmää ovat myös Liu ja muut [2013a] käyttäneet. Menetelmä perustuu kerättyyn dataan. Jokaisen lyönnin data tallennetaan, ja tallennettuja tietoja robotti sitten käyttää hyödykseen liikerataa suunnitellessaan. Robotti käyttää liikeradan suunnittelussa apuna sisääntulo- ja ulostulokarttoja. Karttoja on kolme, jotka korreloivat kolmea fysikaalista tapahtumaa. Ensimmäinen kartta kuvaa mailan ja pallon osumisajan ja niiden sijainnin sekä nopeuden osumishetkellä. Toinen kartta puolestaan kuvastaa pallon nopeuden muutokset ennen mailan osumista ja sen jälkeen. Kolmas kartta sen sijaan kertoo suhteen pallon nopeudesta osumishetkellä ja sen pudotessa vastustajan puolelle. Lisäksi kolmas kartta ilmoittaa lentoajan osumishetkestä pallon laskeutumiseen vastustajan puolella. [Matsushima *et al.* 2005] Kyseistä karttojen antamaa dataa robotti voi hyödyntää myös myöhemmissä lyönneissä ja näin korjata lyöntiparametreja niin, että haluttu nopeus ja sijainti vastapuolen pöydässä voidaan saavuttaa entistä tarkemmin.

Lyöntitapahtuman liikkumisen vaiheet voidaan jakaa kolmeen vaiheeseen, joita ovat valmistautumisvaihe, lyöntivaihe ja lopetusvaihe [Yang *et al.* 2010; Liu *et al.* 2013a]. Vaiheita havainnollistetaan kuvan 5 avulla.



Kuva 5. Robotin lyönnin vaiheiden kuvaus [Liu *et al.* 2013a]

Valmistautumisvaihe on kuvattu kuvassa 5 sinisellä nuolella, lyöntivaihe punaisella nuolella ja lopetusvaihe vihreällä nuolella. Valmistautumisvaihetta ennen robotti siirtää käden kohtaan, josta lyöntiliikkeen tulee alkaa, jotta pallon lyöminen onnistuu. Kuvassa 5 tämä on X_S. Kun oikea kohta on löydetty ja robotti saa käskyn aloittaa lyönti, niin valmistautumisvaiheessa mailan nopeus kiihdytetään tarvittavaan nopeuteen, minkä jälkeen ollaan kuvan 5 pisteessä X_0. Tämän jälkeen robotin käsi jatkaa liikettään tasaisella nopeudella lyönnin

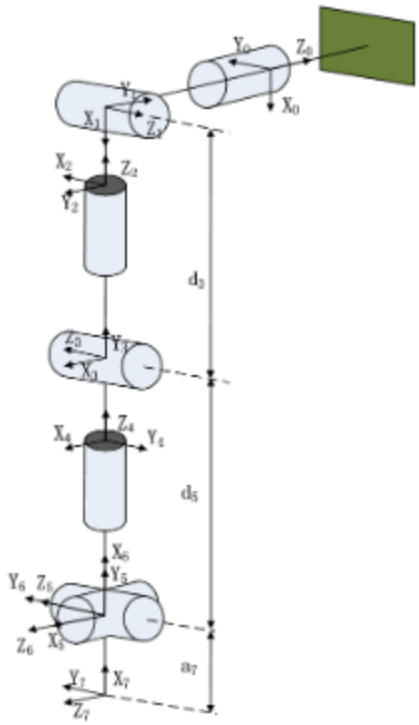
läpi kuvan 5 pisteeseen X_f ja suorittaa näin lyöntivaiheen. Lyönnin tulisi tapahtua arvioidussa osumakohdassa, joka on lyöntiliikkeen keskellä kuvan 5 pisteessä X_m. Näin robotti on suorittanut lyöntiin tarvittavat vaiheet, minkä jälkeen seuraa vielä lopetusvaihe. Lopetusvaihe on tavallaan myös uuden lyönnin esivalmisteluvaihe, sillä lopetusvaiheessa robotti siirtää käden takaisin uuteen laskettuun aloituspisteeseen X_S. Tämän jälkeen robotti on taas valmis aloittamaan uuden lyönnin käskyn saadessaan.

4.3. Mekaaninen rakenne

Mekaaninen rakenne kattaa robotin rungon, käden ja mailan, jolla palloa lyödään [Acosta *et al.* 2003]. Robotti saa hallinnointijärjestelmältä käskyn, jonka mukaan se sijoittaa mailan oikeaan paikkaan ja lyö sopivalla voimakkuudella, jotta pallo saadaan verkon toiselle puolen suunnilleen haluttuun kohtaan.

Mekaanisia rakenteita on monenlaisia. Osa roboteista muistuttaa ihmisiä ja osassa on vain käsi ja maila, eikä robotti siis näytä ihmiseltä. Toimintaperiaatetta tämä ei kuitenkaan varsinaisesti muuta.

Mekaaninen rakenne on kehittynyt tasaisesti ja liikkuvuutta on lisätty eri suuntiin ja vapausasteiden määrää on lisätty. Vapausasteilla tarkoitetaan robotin kädessä olevien nivelten määrää. Vuonna 1993 kehitettiin kolmen suuntaan liikkuva kolmen vapausasteen robottikäsi [Yang *et al.* 2010]. Käsi liikkui ainoastaan ylös- ja alaspäin, eteen- ja taaksepäin ja oikealle ja vasemmalle. Näin ollen mailan kulmaa ei tuolloin vielä pystytty säätämään. Tähän tuli muutos, kun kehitettiin viiteen suuntaan liikkuva käsi [Yang *et al.* 2010]. Liikkuvuutta lisättiin niin, että robotti pystyi kallistamaan mailaa pysty- ja vaakasuunnassa. Robotin käden liikkumissuunnat eivät enää olleet vain lineaarisia, vaan käsi liikkui myös pyörivällä liikkeellä. Tällä hetkellä kehittynein rakenne on seitsemän vapausastetta omaava ja seitsemään eri suuntaan liikkuva käsi [Liu *et al.* 2013a]. Robotin käden liikkeet vastaavat aika hyvin ihmisen käden liikkeitä. Kuvassa 6 on esitetty seitsemän vapausastetta omaavan robotin nivelten välinen rakenne.



Kuva 6. Robotin käden rakenne (7 vapausastetta) [Liu *et al.* 2013a]

Mülling ja muut [2013] kehittivät pöytätennistä pelaavan robotin toimintaan hieman uudenlaisen lähestymistavan, jonka mukaan robotti oppii lyönnit ihmisen esimerkillä. Tässä systeemissä on myös käytetty seitsemän vapausastetta omaavaa robottikättä. Menetelmässä ihminen ohjaa robotin kättä, kun robotti pelaa toista ihmistä vastaan. Näin robotti oppii lyönnit ihmisen lyömänä, kun lyönnit tallennetaan lyöntejä sisältävään kirjastoon. Kun robotti seuraavan kerran pelaa ihmistä vastaan, tulisi robotin vallitsevien olosuhteiden ja informaation nojalla osata valita kirjastosta oikea lyönti ja toteuttaa se. [Mülling *et al.* 2013] Systeemin heikkous on kuitenkin siinä, että harjoitusdataa ihmisen kanssa tulee olla paljon, jotta pelaaminen onnistuu. Todellisuudessa harvoin oikeassa pelissä tulee harjoitellun lyönnin kanssa täysin identtistä tilannetta, jossa kaikki muuttuvat tekijät olisivat täysin samat. Lisäksi saattaa tulla tilanteita, joissa lyönnin opettaminen ei onnistu riittävän hyvin.

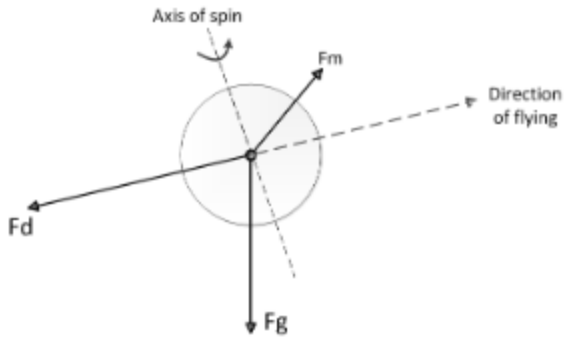
5. Pöytätennistä pelaavan robotin ongelmia

Pöytätennistä pelaavat robotit ovat vuosikymmenien aikana kehittyneet paljon ja niiden taitotaso on lähentynyt ihmisten tasoa. Tästä huolimatta edelleen on monia ongelmia, jotka tulee ratkaista, jotta robotti pystysi pelillisesti antamaan ihmiselle vastusta. Näitä ongelmia tulen seuraavaksi tarkastelemaan tarkemmin. Osa ongelmista ei varsinaisesti estä robottien käyttöä ja pelaamista, mutta hankaloittaa niitä merkittävästi.

Kierteet ovat nykyään merkittävä osa pöytätennistä. Kierteiden kasvaneeseen merkitykseen ovat vaikuttaneet luvussa 2 esittelemäni mailojen ja pallojen kehitys. Varsinkin kumien kitkaisuuden lisääntyminen on lisännyt kierteitä peliin. Monesti kova kierrelyönti on jopa vaikeampi palauttaa kuin todella kova lyönti. Yksi merkittävimpiä ongelmia onkin juuri kierteet ja kierteiden huomioiminen. Tämä ilmenee monella tavalla. Robotin on ensinnäkin todella vaikea lukea kierrettä ja sen voimakkuutta ja toisaalta tämän seurauksena robotin on todella vaikea lyödä kierteiseen palloon onnistuneesti. Kierteen vaikutusta ja havaitsemista on viimeisten vuosien aikana tutkittu todella paljon ja sitä on yritetty ottaa huomioon, mutta kehitystyö on edelleen kesken.

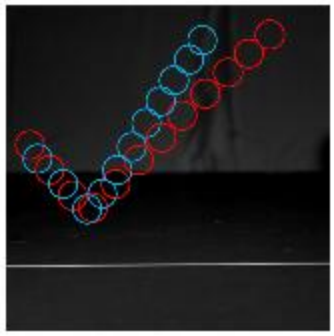
Ennen kuin paneudutaan varsinaiseen ongelmaan, tarkastellaan sitä, kuinka kierre vaikuttaa pelin aikana pallon liikkumiseen. Nämä ovat samalla myös niitä fysikaalisia ilmiöitä, joita robotin tulisi havainnoida.

Kierre siis tuo palloon yhden voiman enemmän ja se vaikuttaa näin pallon lentorataan. Kuvassa 7 on esitetty alakierteiseen palloon vaikuttavat voimat ilmalennon aikana. Lennon aikana palloon vaikuttaa siis painovoiman F_g ja ilmanvastuksen F_d lisäksi myös Magnus-voima F_m. Magnus-voima vaikuttaa pallon lentorataan nostavasti, jos pallossa on alakierre ja toisaalta laskevasti, jos pallossa on yläkierre. Alakierteinen pallo siis lentää huomattavasti pidemmälle kuin yläkierteinen pallo. Robotin tulee huomioida tämä huomattava ero lentoradoissa laskentaa tehdessään.



Kuva 7. Alakierteiseen palloon vaikuttavat voimat [Zhang *et al.* 2014]

Kierre ei ainoastaan vaikuta pallon lentorataan vaan se vaikuttaa myös pallon pomppuun. Alakierteisen pallon pomppu on paljon matalampi kuin yläkierteisen pallon. Konkreettinen ero voidaan havaita kuvasta 8. Kuvassa punaisten pallojen muodostama linja on alakierteisen pallon ja sininen linja yläkierteisen pallon. On myös hyvä huomioida, että ero kasvaa, jos kierteet ovat voimakkaampia ja kapenee puolestaan kierteen ollessa vähäisempi.



Kuva 8. Kierteen vaikutus pallon pomppuun [Nakashima *et al.* 2010]

Varsinainen ongelma on, kuinka havaita kierteen tuomat muutokset ja vaikutukset lentorataan. Lähes ainoita tapoja todentaa kierre on verrata kuvista kierteettömän ja kierteellisen pallon ratoja, jotta tiedetään mihin suuntaan kierre vaikuttaa. On silti todella vaikeaa sanoa selkeästä erosta huolimatta, kuinka voimakas kierre on. Koska voimakkuutta ei tarkkaan voida määrittää, on myös mahdollista, että mailan kulma ei ole aivan optimaalinen, jotta pallo saataisiin onnistuneesti vastustajan puolelle. Virheen mahdollisuus ja toisaalta suuruus epätarkkuuksien myötä usein kasvaa. Hallinnointijärjestelmään on kehitetty menetelmä, jonka avulla robotin ei tarvitse laskea pallon sijaintia aina uudestaan uusien tietojen avulla vaan robotti pystyy hyödyntämään myös historiatietojaan [Liu *et al.* 2015]. Liu ja muut [2015] onnistuivat osoittamaan, että järjestelmä säästää tietokoneelta arvokasta aikaa. Jotta jatkossa kierre saadaan huomioitua entistä paremmin, tulee varmasti tarkastella tällaisia ratkaisuja, sillä uusien kuvien vertailu historiatietoihin on tällä hetkellä yksi vain muutamasta vaihtoehdosta, jolla kierre voidaan tulkita.

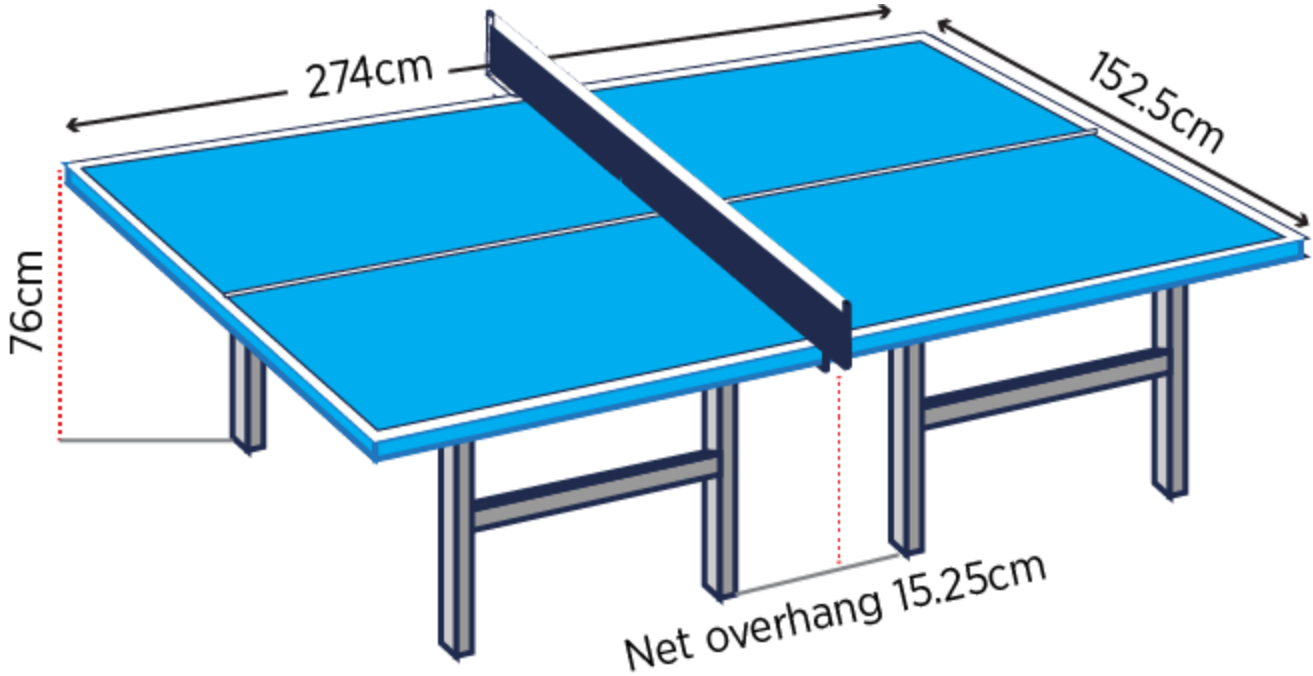
Kierteisiä palloja on myös onnistuneesti palautettu pöytään, mutta vielä ei voida todellakaan puhua siitä tarkkuudesta, joka vaadittaisiin ihmisen kanssa tasaväkiseen pelaamiseen. Zhao ja muut [2015] kehittivät liikemallin pyörivälle pöytätennispallolle ja mallin avulla onnistuivat lyömään myös kierteisiin palloihin. Vaadittu kuvaussysteemi oli kuitenkin erittäin kallis [Zhao *et al.* 2015]. Zhang ja muut [2015] puolestaan hyödynsivät pallossa olevaa merkkiä kierteen havaitsemiseen. Menetelmän kannalta haastavaa on se, ettei merkki välttämättä ole nähtävissä selvästi jollain puolella palloa, jos kierre on todella voimakas. Lisäksi sivukierre saattaa sotkea menetelmän tarkkuutta. Ongelmia siis vielä riittää, vaikka onnistumisiakin on saatu.

Kierre ei ole ainoa ongelma, joka robottien toimintaan liittyy. Toinen ongelma on myös seurausta luvussa 2 esittelemästäni välineiden kehityksestä. Kehityksen myötä peli on tullut nopeammaksi ja näin ollen robottien on vaikeampi pelata pöytätennistä yhtä hyvin kuin ihmiset. Vaikka pallot ovatkin

suurentuneet ja sen myötä tulleet hitaammiksi, niin mailojen kehitys on ollut niin hurjaa, että kokonaisuutena peli on nopeutunut. Pallo voi tulla niin kovaa, että robotilla on vain kymmenesosa sekunnista aikaa reagoida pallon tuloon ja lyödä se takaisin [Liu *et al.* 2014]. Tämä asettaa todella suuria vaatimuksia robotille ja sen toiminnalle. Robotin tulee saada kuvatiedot havaintojärjestelmältä riittävän nopeasti, jotta lentorata voidaan ennustaa ja tämän jälkeen toteuttaa lyönti riittävän nopeasti. Tämän hetken havainnointijärjestelmät pystyvät parhaimmillaan ottamaan 900 kehystä sekunnissa. Tämä ei kuitenkaan aina riitä pallon kovan nopeuden vuoksi. Pallo muuttuu nopeuden kasvaessa kuvassa enemmän ellipsimäiseksi ja hämärtyy, eikä enää muistuta niin selkeästi ympyrää [Liu *et al.* 2013b]. Pallon sumentuminen kuvassa vaikuttaa pallon sijainnin määrittämisen tarkkuuteen.

Osin edellisestä syystä ja kierteiden huomioimisen myötä tietokoneiden tulee jatkossa pystyä laskemaan nopeammin sekä käsittelemään suurempaa määrää dataa. Tästä johtuen onkin syytä muistaa algoritmeja luodessa tarkastella myös aikakompleksisuutta, jolla siis tarkoitetaan ajan suhdetta kasvavaan datan määrään. Algoritmien on jatkossa oltava entistä nopeampia, jotta laskenta saadaan suoritettua todella marginaalisessa ajassa.

Tämän hetken robotit eivät liiku jaloillaan pöydän päässä, vaan ainoastaan liikuttavat kättään ja kurottavat osuakseen palloon. Tästä seuraa yksi huomattava mekaaniseen rakenteeseen liittyvä haaste robottien halutulle toiminnalle. Pallot eivät pelatessa tule aina pöydän päästä yli, vaan ne voivat tulla yli myös pöydän sivusta. Robotin tulisi ylettää näihinkin palloihin. Pöydän päällä tarkoitetaan kuvassa 9 olevaa 152,5 senttimetriä pitkää sivua. Toisaalta pallo voi myös jäädä pöydän päälle. Tämä tarkoitta, että pallo pomppaisi useamman kerran pöydän päällä ennen kuin ylittää pöydän reunan. Robotin tulisi osata siis myös hahmottaa, milloin pallo jää pöydän päälle ja lyödä sitä ennen toista pomppua. Sääntöjen mukaanhan pallo ei saa pompata omalla puolella kuin kerran ennen kuin se tulee lyödä. Toisaalta palloa ei myöskään saa lyödä suoraan ilmasta. Robotin käsi on kehittynyt kolmen vapausasteen kädestä seitsemän vapausasteen käteen, mutta silti koko pöydän aluetta on vaikea kattaa. Käden kehityksestä huolimatta saattaa tulla tilanteita, jolloin robotin käsi ei silti taivu lyömään palloa halutulla tavalla.



Kuva 9. Pöytätennispöydän mitat [ittf.com 2017b].

Ongelma voidaan toki ratkaista pidentämällä robotin kättä, mutta pidentämällä kättä liikenopeus kärsii. Jotta robotti pelaisi täysin kuin ihminen, tulisi sen osata reagoida moneen erilaiseen lyöntiin. Tämä on kuitenkin todella ongelmallista.

Rakenteiden kehittyessä yhä paremmiksi on hyvä huomioida, että systeemi on myös hyvin kallis. Hinta vaikeuttaa menetelmän kulutuskäyttöön ottamisessa ja tulevaisuudessa hintapuoleen on syytä keskittyä. Kuinka saadaan riittävä tarkkuus sopivaan hintaan, jotta kuluttajat ovat valmiita sen maksamaan? Lisäksi systeemi on moniosainen ja työläs asentaa toimintakuntoon, mikä hankaloittaa sen treenikäyttöön ottamista. Järjestelmän asentaminen vaatii alan osaamista ja perehtymistä aiheeseen.

6. Pöytätennistä pelaavien robottien tulevaisuuden näkymät

Kuten Parisi [2014] toteaa, ihmistä muistuttavat robotit voivat olla tulevaisuudessa mahdollisia. On siis myös mahdollista, että tulevina vuosina kehitetään todella hyviä ihmisen kanssa pöytätennistä pelaavia robotteja, joita voidaan käyttää harjoitusolosuhteissa hyödyksi. Olen hieman skeptinen asian suhteen, sillä kehitystyötä on vielä paljon, jotta tähän päästään. Kehityksessä pitäisi tapahtua merkittävä hyppy eteenpäin, jotta robottien kehitys ottaisi lajin kehityksen kiinni. Aika lopulta näyttää, saako tekniikka lajin kehityksen kiinni.

Tarkastellaan seuraavaksi mahdollisuuksia, joita täydellisesti kehitetyt robotit voisivat tuoda lajiin. Toisaalta tarkastellaan myös, kuinka robottien pöytätennikseen mukaan tulo kehittää lajia ja siinä samalla robotiikkaa itsessään.

Robotit mahdollistaisivat yksinään harjoittelun. Esimerkiksi Suomessa on tällä hetkellä niin vähän pelaajia, ettei harjoitusvastustajaa aina ole helppo löytää. Tosin harrastajamäärät Suomessa ovat tällä hetkellä jopa pienoisessa nousussa. Jos olisi täydellistä pöytätennistä pelaava robotti, niin harjoittelu korkealla tasolla olisi mahdollista yhä useammin. Itsekseen harjoitellessa ei tarvitsisi jatkossa tyytyä vain harjoittelemaan syöttöjä ja lyömään palloa seinään, mikä ei vastaa pelitilannetta.

Toisinaan saattaa myös käydä niin, että harjoitusvastustajien taso ei enää riitä huipputasolla harjoitteluun. Kun pelaaja kehittyy riittävästi, on hänen lähdettävä hakemaan muualta parempia vastustajia. Robotti voisi poistaa tämänkin ongelman ainakin teoreettisella tasolla. Jos kaikkien saatavilla olisi yhtä hyvä robotti, joka on maailman paras pöytätenniksen pelaaja, ei harjoitusvastustajaa enää tarvitsisi etsiä. Aina olisi tarjolla parasta mahdollista harjoitusta. Toisaalta voidaan miettiä, kuinka mielekästä pelaajan näkökulmasta olisi pelata täysin reagoimatonta robottia vastaan päivästä toiseen, kun ystävyyssuhteita ja sosiaalista kanssakäymistä ei enää juuri syntyisi. Lajiin saatettaisiin kuitenkin saada lisää tasoa ja tasoerot maiden välillä kaventuisivat, jos kaikille olisi tarjolla yhtä hyvää harjoitusta. Ehkä kehityksen myötä Kiina ei enää olisi niin selkeästi muita parempi. On kuitenkin otettava huomioon, etteivät tasoerot tule pelkästään paremmasta harjoittelusta ja paremmista harjoittelukavereista. Kaikki tuskin olisivat yhtä hyviä pelaajia yli-ihmismäisen robotinkaan ansiosta, mutta ainakin erot voisivat kaventua ja lajin kiinnostus kasvaisi tasaisuuden myötä lisää.

Täydellisen pöytätennistä pelaavan robotin kehittäminen palvelisi varmasti myös robotiikan kehitystä. Pöytätennis on niin vaativa laji, että sitä pelaavan robotin osia voitaisiin varmasti hyödyntää muussakin robotiikassa. Lisäksi robotit tulisivat entistäkin arkipäiväisemmiksi, jos ne olisivat kulutustavaraa ja markkinoilla kaikkien saatavilla. Toisaalta olisiko tämän jälkeen oikeasti vaarana luvussa 3 mainitsemieni Capekin veljesten huoli robottien maailmanvalloituksesta?

Robottien toimintaan vaativia komponentteja tullaan tulevaisuudessakin kehittämään. On mielenkiintoista seurata, kehittyykö jokin robotin toimintamenetelmän osa-alue muita merkittävämmin. Ja toisaalta onnistuuko kehittyneiden komponenttien yhdistäminen vanhoihin komponentteihin ilman ongelmia? Robotin kehitys kuitenkin perustuu komponenttien kehittymiseen

yhdessä. Toisen on pakko kehittyä, jos toinenkin kehittyy, muutoin kokonaisuus helposti rikkoutuu.

7. Yhteenveto

Pöytätennisrobottien rakenne koostuu siis pääasiassa havainnointijärjestelmästä, hallinnointijärjestelmästä ja mekaanisesta rakenteesta, jolla käytännössä tarkoitetaan robotin kättä ja mailaa. Tutkimuksen myötä rakenteen kehityksestä ja toisaalta kehityksen tarpeesta saatiin melko hyvä kuva. Kamerat eivät enää ota vain muutamia kehyksiä pallon seuraamiseksi, vaan kehysten määrä on noussut lähes 1000 kehykseen. Tietokoneiden sisältämät algoritmit ovat kehittyneet ja robotin käden rakenteeseen on tullut lisää niveliä ja sen myötä liikkuvuutta.

Kehitysaskelista huolimatta kaikilla osa-alueilla tulee vielä kehittyä, jotta robotti kykenisi korkealla tasolla pelaamaan yhtäjaksoisesti pöytätennistä ihmisen kanssa. Varsinkin kierteen huomioimisessa ja siihen reagoimisessa on vielä pitkä matka ihmisen taitotasoon. Ihmisen etuna on kyky havainnoida myös vastustajan liikkeitä, mihin robotti ei tällä hetkellä lainkaan kiinnitä huomiota. Robotti luottaa täysin pallosta saatavaan informaatioon. Jotta kierteet saadaan robottien peliin paremmin mukaan, tulee kierteen havaitsemismenetelmiä vielä kehittää ja jatkossa luoda tarkempia menetelmiä.

Lyöntien onnistumisprosentit ja varsinkin lyöntien laatu ei vielä vastaa ihmisen vastaavia. Tässä tarkastelussa on hyvä huomioida, että vertaan onnistumisprosentteja maailman parhaisiin pelaajiin. On kuitenkin hyvä muistaa, ettei robotista täysin virheetöntä saa, eikä se ehkä ole tarkoituskaan. Jos robotti aina palauttaa pallon takaisin, niin ainoaksi vaihtoehdoksi jää, että ihminen joko väsyy tai tekee jossain kohtaa virheen. Pidemmän päälle harjoittelun tuloksellisuuden myötä on merkittävää, ettei aina koe itse tekevänsä virhettä.

Pöytätennistä pelaavat robotit ovat kehittyneet vuosikymmenien kehityksen myötä merkittävästi. Vaikka robotit eivät vielä olekaan suoraan harjoituskäyttöön otettavissa ja markkinoille kuluttajien ostettavaksi tuotavissa, niin lähemmäksi ollaan kuitenkin päästy. En tästä kehityksestä huolimatta henkilökohtaisesti usko, että robotit tulevat lähivuosien aikana kehittymään niin paljon, että tavoite harjoituskäytöstä saavutettaisiin. Kehitystyötä tehdään todella paljon varsinkin Kiinassa, mutta painopiste keskittymisessä ei ainakaan vielä ole näkyvästi ollut markkinoille tuotavan kulutustuotteen kehityksessä. Toisaalta tämä on ymmärrettävää, kun ottaa huomioon, kuinka paljon kehitystyötä vielä tulisi tehdä, jotta robotit kykenisivät pelaamaan kuin ihmiset. Tutkimusta tehdessäni mieleen nousi monesti myös kysymys, onko tuon tavoitteen saavuttaminen oikeasti edes mahdollinen. Aika varmasti antaa vastauksia tä-

hän ja mahdollisesti vuosien kehityksen jälkeen käytössämme on myös itsenäisesti huipputasolla pöytätennistä pelaavia robotteja.

Viiteluettelo

L. Acosta, J. J. Rodrigo, J. A. Mendez, G. N. Marichal and M. Sigut. 2003. Ping-pong player prototype. *IEEE Robotics & Automation Magazine* 10, 4, 44–52.

Allabouttabletennis. *History of table tennis.* http://www.allabouttabletennis.com/table-tennis-racket-history.html. Checked 10.5.2017

P. F. Bladin. 2006 W. Grey Walter, pioneer in the electroencephalogram, robotics, cybernetics, artificial intelligence. *Journal of Clinical Neuroscience* 13, 2, 170-177.

U. Frese and H. Hirschmüller. 2015. Special issue on robot vision: what is robot vision?. *Journal of Real-Time Image Processing* 10, 4, 597-598.

N. G. Hockstein, C. G. Gourin, R.A. Faust and D. J. Terris. 2007. A History of robots: from science fiction to surgical robotics. *Journal of Robotic Surgery* 1, 2, 113–118.

ITTF. 2017a. *World rankings 05/2017.* http://www.ittf.com/rankings/. Checked 10.5.2017

ITTF. 2017b. *Handbook 45th edition.* http://dd7978b90kfgn.cloudfront.net/2016/12/2017_ITTF_Handbook.pdf. Checked 10.5.2017

M. Jiang, Q. Gu, T. Aoyama, T. Takaki and I. Ishii. 2017. Real-time vibration source tracking using high-speed vision. *To appear in IEEE Sensors Journal* PP, 99, 1

J-G. Latombe. 1991. *Robot Motion Planning*. Kluwer Academic Publishers.

H. Liu, Z. Li, B. Wang, Y. Zhou and Q. Zhang. 2013a. Table tennis robot with stereo vision and humanoid manipulator I: system design and learning-based batting decision. In: *Proc. of the 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO).* 905–910.

H. Liu, Z. Li, B. Wang, Y. Zhou and Q. Zhang. 2013b. Table tennis robot with stereo vision and humanoid manipulator II: visual measurement of motion-blurred ball. In: *Proc. of the 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO).* 2430–2435.

J. Liu, Z. Fang, K. Zhang and M. Tan. 2014. Improved high-speed vision system for table tennis robot. In: *Proc. of the 2014 IEEE International Conference on Mechatronics and Automation.* 652-657.

J. Liu, Z. Fang, K. Zhang and M. Tan. 2015. A New data processing architecture for table tennis robot. In: *Proc. of the 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO).* 1780–1785.

Killerspin 1.4.2015. *Table Tennis through time (A brief history of Ping Pong)* https://www.killerspin.com/blog/table-tennis-through-time-a-brief-history-of-ping-pong/. Checked 10.5.2017.

M. Matsushima, T. Hashimoto, M. Takeuchi and F. Miyazaki. 2005. A learning approach to robotic table tennis. *IEEE Robotics & Automation Magazine* 21, 4, 767–771.

K. Mülling, J. Kober, O. Kroemer and J. Peters. 2013. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research* 32, 3, 263–279.

A. Nakashima, Y. Ogawa, Y. Kobayashi and Y. Hayakawa. 2010. Modeling of rebound phenomenon of a rigid ball with friction and elastic effects. In: *Proc. of the 2010 American Control Conference.* 1410–1415.

Olympic. *Table tennis equipment and history.* https://www.olympic.org/table-tennis- equipment-and-history. Checked 10.5.2017.

D. Parisi. 2014. *Future Robots: Towards a Robotic Science of Human Begins.* John Benjamins Publishing Company.

Q. Wang, X. Zhang and D. Xu. 2012. Human behaviour imitation for a robot to play table tennis. In: *Proc. of the 24th Chinese Control and Decision Conference (CCDC).* 1482–1487.

P. Yang, D. Xu, H. Wang and Z. Zhang. 2010. Control system design for a 5-DOF table tennis robot. In: *Proc. of the 11th International Conference on Control Automation Robotics & Vision.* 1731–1735.

P. Yang, D. Xu, Z. Zhang, G. Chen and M. Tan. 2011. A Vision system with multiple cameras designed for humanoid robots to play table tennis. In: *Proc. of the 2011 IEEE International Conference on Automation Science and Engineering.* 737–742.

Y. Zhang, Y. Zhao, R. Xiong, Y. Wang, J. Wang and J. Chu. 2014. Spin observation and trajectory prediction of a ping-pong ball. In: *Proc. 2014 IEEE International Conference on Robotics & Autiomation.* 4108–4114.

Y. Zhang, R. Xiong, Y. Zhao and J. Wang. 2015. Real-time spin estimation of ping-pong ball using its natural brand. In: *Proc. 2015 IEEE Transactions on Instrumentation and Measurement.* 64, 8, 2280–2290.

Y. Zhao, Y. Zhang, R. Xiong and J. Wang. 2015. Optimal state estimation of spinning ping-pong ball using continuous motion model. In: *Proc. 2015 IEEE Transactions on Instrumentation and Measurement.* 64, 8, 2208–2216.

Riikka Finnilä

Suomen kielen konekääntäminen

Tiivistelmä.
Konekäännöstä on kehitetty pitkään, mutta se on yhä vaikeasti ratkaistava ongelma. Luonnolliset kielet ovat monimutkaisia ja keskenään erilaisia, joten konekäännöstä on tutkittu enimmäkseen tietyt kieliparit kerrallaan. Niinpä konekäännöstutkimus on pirstaloitunutta.

Konekäännösmenetelmiä on kehitetty lukuisia, ja nykyisin niitä yhdistellään paljon eri tarkoituksiin. Tällä hetkellä laajimmin käytetään tilastollisia menetelmiä, joihin on yhdistetty erilaisia tekoälyn keinoja kuten koneoppimista ja neuroverkkoja.

Suomen kieli on konekääntäjille erityisen vaikea, sillä suomi on voimakkaasti taipuva kieli: perusmuotoisiin sanoihin voidaan liittää valtavasti erilaisia liitteitä ja päätteitä, sanoja voidaan liittää yhdyssanoiksi ja sanoja voidaan johtaa edelleen uusiksi sanoiksi. Yksittäinen sanamuoto voi sisältää paljon kieliopillista ja merkityksellistä tietoa. Monissa muissa kielissä sama informaatio voidaan kuvata useamman sanan yhdistelmillä. Tämä tekee kääntämisen vaikeaksi, kun yksittäisille sanoille ei ole suoria yksittäisiä vastineita kohdekielessä. Tämän ongelman ratkaisemiseksi suomen kielen analyysiin pureudutaan morfologisten jäsentimien avulla. Ne ovat sääntöjä, koneoppimista ja erilaisia algoritmeja hyödyntäviä järjestelmiä, jotka pilkkovat taivutusmuotoja pienempiin osiin, morfeihin, joiden avulla käännös helpottuu.

Avainsanat ja -sanonnat: konekääntäminen, käännösteknologia, suomen kieli.

1. Johdanto

Koneen suorittamasta automaattisesta käännöksestä on haaveiltu tietokoneiden keksimisestä lähtien [Somers 2003]. Käytännössä tutkimus alkoi yleistyä maailmalla 1950-luvulla, ja aina 1980-luvun alkupuolelle asti useat tutkijat uskoivat tietokoneiden oppivan pian kääntämään mitä tahansa tekstejä yhtä hyvin kuin ihminen [Hutchins 2003; Somers 2003]. Sittemmin on ymmärretty, että luonnollisten kielten moniulotteisuus ja eri kielten väliset erot ovat hankala tietojenkäsittelyongelma. Tultaessa 2000-luvulle useimmat tutkijat olivat jo sitä mieltä, ettei koneen ole mahdollista oppia ihmiskääntäjän tasoiseen käännöstyöhön lähitulevaisuudessa [Hutchins 2003].

Konekääntämistä käytetään nykyisin laajalti, vaikkei käännös useimmiten edelleenkään ole virheetöntä. Tiedonjano voittaa, kun valitaan huonon käännöksen ja tietämättömyyden välillä, joten heikompiakin käännöksiä käytetään ainakin silmäilevään tiedonhakuun. Konekäännösjärjestelmiä käytetään myös mm. alustavina menetelminä, ja ihminen jälkieditoi käännöksen mahdollisimman virheettömäksi. Toisaalta, myös koneita ohjelmoidaan yhä enemmän korjaamaan yleisimmin toistuvia käännösvirheitä kuten väärää sanajärjestystä tai puuttuvia artikkeleja [Hutchins 2003].

Onnistumisiakin on toki alalla koettu. Esimerkiksi Kanadassa 1980-luvulla kehitetty, säätiedotteita englannista ranskaan kääntävä MÉTÉO-järjestelmä toimi jo varhain poikkeuksellisen virheettömästi, sillä se käsittelee niin suppeaa erityissanastoa, että koneella on vähemmän mahdollisuuksia virheisiin [Somers 2003; Hutchins 2003]. Nykyisin erittäin hyviä tuloksia on saavutettu mm. neuroverkkoteknologian yhdistämisellä tilastollisiin käännösmenetelmiin.

Ihmisten väliseen kommunikointiin käytettävät luonnolliset kielet ovat kuitenkin hankalia analysoitavia algoritmeille. Kone esimerkiksi ymmärtää helposti yksiselitteisiä merkkijonoja, kun taas ihminen voi tulkita yksittäisen sanan tai lauseen monella eri tapaa. Luonnollisen kielen tulkintaan voivat vaikuttaa mm. konteksti ja kulttuuri, ja lisäksi samalla tavalla kirjoitettu sana voi tarkoittaa monia eri asioita. Esimerkiksi suomenkielisessä tekstissä yksittäinen sana `voita` voi olla voittaa-verbin imperatiivi- eli käskymuoto, tai maidosta kirnutun tuotteen, voin (substantiivi), partitiivimuotoa: `Levitin leivälle` **`voita`**. Ihminen hahmottaa lauseyhteydestä heti, kumpaa tarkoitetaan, mutta koneelle pitää erikseen ohjelmoida sanaston lisäksi esimerkiksi lauseenjäsenet, jotta se osaisi valita oikein substantiivin ja verbin väliltä.

Suomen kielen asema käännöstieteissä ei ole maailman valtakielten kaltainen. Pieniväestöisen maamme kieli suomi ei pärjää englannin, mandariinikiinan tai espanjan kaltaisille kielille, joita puhuu äidinkielenään sadat miljoonat ihmiset, ja joiden kääntämistä on kiihdyttänyt esimerkiksi pitkään jatkunut kansainvälinen kaupankäynti. Toisaalta se, että suomalaisten kielitaito on kansainvälisessä vertailussa erittäin hyvää, osaltaan vähentää tarvetta kääntää valtakieliä suomeksi. Monet suomalaiset ymmärtävät joko englantia, espanjaa, venäjää tai ranskaa, ja todennäköisesti myös aasialaisten valtakielten osaaminen lisääntyy koko ajan. Toisaalta suomalaisen yhteiskunnan kansainvälinen suuntautuminen ja mm. teknologinen edistyksellisyys myös lisäävät käännösten tarvetta.

Seuraavassa luvussa kerron konekääntämisestä ja sen eri menetelmistä yleisesti, kääntämisen erilaisista käyttökohteista, konekäännöksen globaaleista haasteista luonnollisen kielen käsittelyssä, sekä konekäännöksen laadunarvioinnista. Luvussa 3 kuvaan suomen kielen erityispiirteitä, keskeisiä suomen kielen konekäännökseen liittyviä ongelmia sekä menetelmiä niiden käsittelyyn. Yleinen tai

suomen kielen kielioppi ei kokonaisuudessaan mahdu tämän tutkielman laajuuteen, vaan käännösongelmiin liittyvät kieliopilliset asiat pyrin havainnollistamaan lähinnä esimerkkien avulla. Neljännessä luvussa käyn läpi kotimaisen konekäännöstutkimuksen nykytilaa. Tutkielman lopussa on pohdintaa ja yhteenveto.

2. Konekääntäminen

Konekääntäminen määriteltiin tieteenalan alkuvuosina tietokoneen suorittamaksi tekstin automaattiseksi kääntämiseksi joltakin luonnolliselta kieleltä toiselle luonnolliselle kielelle [Hutchins 1986]. Myöhemmin se sai myös seuraavan määritelmän: ”Konekäännös on mikä tahansa prosessi, joka koostuu lähdekielisen tekstin analysoinnista, tämän tekstin muuntamisesta ja kohdekielisen tekstin tuottamisesta” [Sager 1994]. Joissain kielissä konekääntämisestä käytetään termiä *automaattinen kääntäminen* [Hutchins 2005].

Läheinen termi on myös käännösteknologia, joka on kuitenkin laajempi termi sisältäen myös ihmiskääntäjien käyttämät apuvälineet ja sovellukset. Tässä tutkielmassa ei kuitenkaan käsitellä ihmisen suorittamaa tietokoneavusteista kääntämistä, vaan keskitytään tietokoneen yksin tuottamaan automaattiseen käännökseen. Työstä on lisäksi rajattu pois puheentunnistukseen liittyvät aiheet, vaikka nykyisin tehdään käännöksiä myös ääninauhalta tai puheesta. Tämän työn laajuus ulottuu kuitenkin vain kirjoitettujen tekstien kääntämiseen.

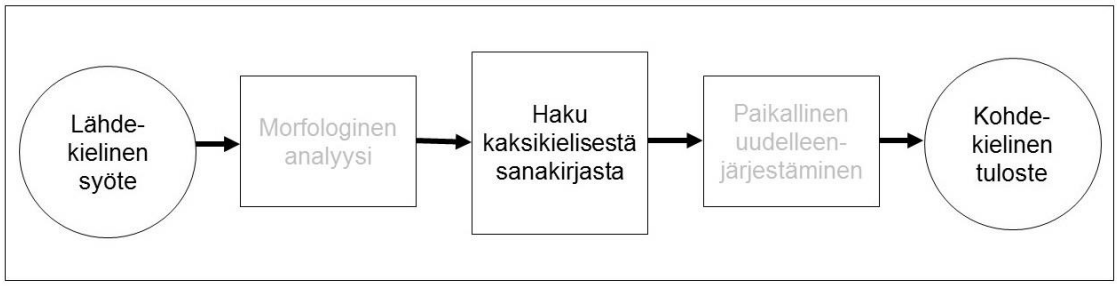
Kohdassa 2.1 käyn läpi konekääntämisen historiaa ja eri menetelmien kehittymistä. Kohdassa 2.2 kuvaan joitakin konekäännöksen tarpeita ja käyttökohteita. Monet keskeiset ongelmat konekäännöksessä ovat yhteisiä useille, tai jotkut jopa kaikille luonnollisille kielille. Kohdassa 2.3 kerron näistä konekäännöksen yleisimmistä globaaleista haasteista ja kohdassa 2.4 hieman käännöksen laadunarvioinnista.

2.1. Konekääntämisen menetelmät ja niiden kehittyminen

Tavallisesti konekäännösprosessissa on kolme päävaihetta: se alkaa lähdekielen analyysistä, jonka jälkeen analysoidun tekstin merkitys siirretään toiselle kielelle ja lopuksi muodostetaan varsinainen kohdekielinen teksti. Tätä voi lähestyä muutamilla eri strategioilla.

Ensimmäisissä konekäännösjärjestelmissä käytettiin ns. *suoraa käännöstä*, jolloin järjestelmä piti suunnitella jokaista kieliparia varten erikseen: sanastoa ja syntaksia analysoitiin minimimäärä, jotta selvitettiin monitulkintaisuudet, kohdekielen vaihtoehtoiset sanat sekä oikea sanajärjestys. Suoran käännöksen menetelmä esitetään kuvassa 1. Kolmivaiheinen prosessi yritettiin käsitellä yhtenäisenä, monimutkaisena ongelmana, eikä merkittäviä tuloksia syntynyt [Hutchins 2003]. Näitä alan alkuvuosina kehitettyjä järjestelmiä kutsutaan *sääntöpohjaisiksi*

*(*tai *sääntöperustaisiksi) konekääntäjiksi*, koska ne perustuivat niihin syötettyihin lähtö- ja kohdekielten kielioppisääntöihin [Hutchins 2005].



Kuva 1. Suoran käännöksen konekäännösarkkitehtuuri [Hutchins and Somers 1992]. Harmaalla merkityt vaiheet olivat alkeellisia, eikä niillä ollut suurta merkitystä käännöksen laatuun.

Konekäännöstä tutkittiin yhä monipuolisemmin 1970- ja 1980-luvuilla ja tuolloin yleisin käännösstrategia oli ns. *epäsuora käännös* [Hutchins 2003]. Tämän aikakauden käännösjärjestelmät perustuivat täsmällisemmin kolmivaiheisen käännösprosessin jaotteluun, eli käännösongelman pilkkominen useampaan pienempään osaan teki koko prosessin koneille helpommaksi käsitellä [Somers 2003]. Käsiteltävät kielet analysoitiin semanttisella, syntaktisella sekä morfologisella tasolla ja niiden välille luotiin jonkinlainen esitysmuoto (esimerkiksi ns. *interlingua, yleinen kielestä riippumaton universaali välikieli*), jolla pystyttiin kuvaamaan tekstin merkitys [Hutchins 2003]. Somers [2003] kutsuu 1970- ja 1980-luvuilla kehitettyjä konekäännösjärjestelmiä ”toisen sukupolven konekääntäjiksi”. Nämä järjestelmät olivat *korpusperustaisia menetelmiä,* ja niiden joukosta kehittyivät hieman myöhemmin *tilastolliset* ja *esimerkkipohjaiset konekääntäjät* [Hutchins 2005]. Korpus tarkoittaa konekääntäjän esimerkkinään käyttämää mallitekstien joukkoa.

Tilastollinen konekääntäminen käsittelee kääntämistä koneoppimisongelmana [Lopez 2008]. Kone oppii kääntämään, jos se on ”opetellut” suuren määrän aiemmin käännettyjä tekstejä, joita kutsutaan siis korpuksiksi tai rinnakkaisteksteiksi. Käytännössä kone osaa valita tilastollisesti todennäköisimmät käännökset vertaamalla sanoja, lauseen osia tai kokonaisia lauseita aiempiin käännöksiin. Lähdetekstin segmentille etsitään todennäköisin käännös hyödyntämällä sekä kaksikielistä rinnakkaistekstikorpusta että yksikielistä kohdekielen korpusta yhdessä jonkinlaisen kielimallin kanssa [Lopez 2008].

Myös esimerkkipohjaiset konekääntäjät käyttävät kaksikielistä esimerkkitekstimassaa, mutta tilastollisen laskennan sijaan rinnakkaisteksteille tehdään *tekstinlinjaus* (engl. text alignment) eli ne linjataan rinnakkain niin, että vastaavat

sisällöt ovat kohdakkain [Hutchins 2005]. Esimerkkipohjaisessa menetelmässä siis yhdistellään rinnakkaisteksteistä löytyviä lauseen osia sellaisenaan kieliopin mukaisiksi lauseiksi.

Nykyisin tilastolliset menetelmät hallitsevat konekääntämisen kehitystä, sillä suurten sähköisten yksi- ja kaksikielisten korpusten, prosessiin soveltuvien avoimen lähdekoodin apusovellusten ja vakiintuneiden laadunarviointimenetelmien saatavuus on kasvanut. Tilastollisten menetelmien kehitys ei myöskään vaadi tekijöiltään käsittelemiensä lähde- ja kohdekielten perusteellista ymmärtämistä. [Lopez 2008] Toisaalta tilastollisiin menetelmiin yhdistetään yhä myös sääntöjen käyttöä. Esimerkiksi syntaktinen analyysi rakenteellisesti erilaisten kielten kääntämiseen, morfologinen analyysi voimakkaasti taipuvien kielten käsittelyyn tai viittaussuhteiden ongelmien käsittely voivat vaatia laajaa sääntöjen soveltamista.

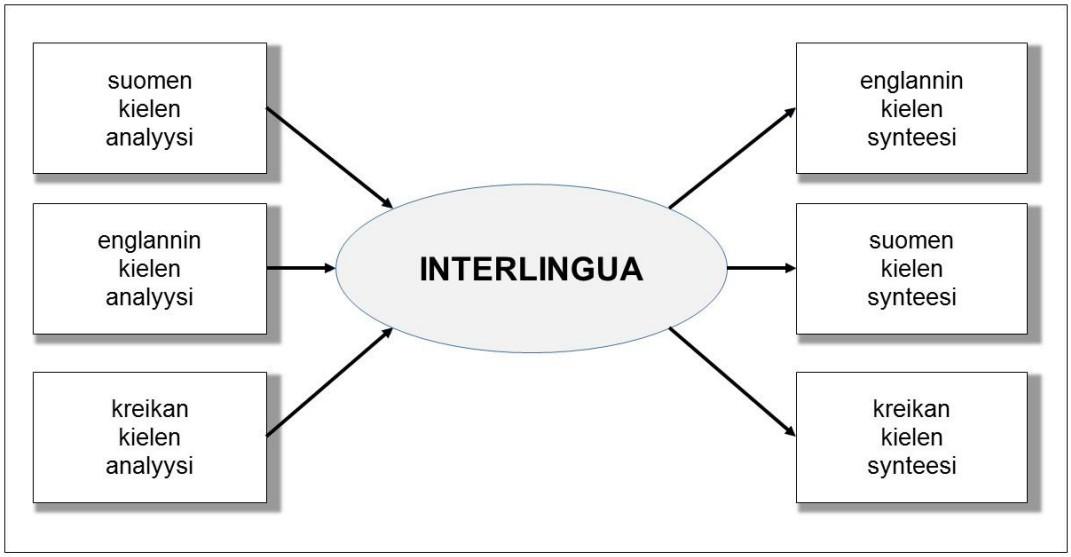
Viimeisimpänä konekäännösteknologiana on otettu käyttöön neuroverkot. Neuroverkkojen avulla on saatu tehostettua tilastollista konekääntämistä, sillä ne käyttävät paljon vähemmän tietokoneen muistia kuin perinteisemmät menetelmät [Luong et al. 2015]. Neuroverkkojen soveltaminen konekääntämiseen on suhteellisen helppoa, joten se on saanut nopeasti lisää suosiota tutkimus- ja kehityskohteena [Luong et al. 2015]. Luvussa 4 nähdäänkin, että neuroverkkoja on alettu tutkia myös suomen kielen konekääntämisen parantamiseksi. Neuroverkkomenetelmällä väitetään voivan pian toteuttaa myös monikielinen kääntäjä, jota ei tarvitsisi räätälöidä erikseen tietylle kieliparille [Firat et al. 2016].

Yhtenä ajankohtaisena ongelmana konekääntämisen alalla pysyttelee järjestelmien laajentaminen eri kielille. Käännösohjelmistojen laajennus uusille kieliparelle vaatii paljon kehittämistä jokaista kieliparia kohden, sillä luonnolliset kielet voivat olla keskenään hyvin erilaisia, ja joillekin kielille riittävän suuren sanaston luominen voi viedä yllättävän paljon aikaa [Somers 2003]. Vuosina 1990–2007 kehitettiinkin lähes yksinomaan kielipareja, joissa toinen kieli oli englanti. Muut kieliparit jäivät kaupallisen valtakielen varjoon, vaikka toki joitakin järjestelmiä oli esimerkiksi Euroopan ja Aasian sisällä naapurimaiden kesken. [Hutchins 2010]

Esimerkiksi interlinguaa käyttävät menetelmät kehitettiin, koska toivottiin olevan mahdollista kehittää täysin universaali välikieli, jonka kautta tekstejä olisi voitu kääntää miltä tahansa kieleltä kaikille muille kielille. Vaikka luonnolliset kielet eroavatkin valtavasti ns. *pintarakenteeltaan*, niiden uskottiin kuitenkin jakavan yhteisen *syvän rakenteen*, joka voitaisiin mallintaa jonkinlaisena säännöllisenä ja ennustettavana interlingua-esityksenä [AlAnsary 2011]. Interlingua-arkkitehtuuri esitetään kuvassa 2.

Kielten moninaisuus on kuitenkin osoittanut, että kaikkiin kieliin sopeutuvaa mallia lienee mahdotonta luoda [Austermühl 2001]. Täysin yleismaailmallinen esitysmuoto tekstin merkitykselle on utopistinen ajatus. Sellaisen kehitykseen

liittyvät samat ongelmat kuin keinoälyn laajempaankin kehitykseen. Kielen kuvaukseen vaadittaisiin kyky mallintaa ihmismielen *käsitys maailmasta* (engl. world knowledge) [Dorr et al. 2004]. Lisäksi kielten välisiä rakenne- ja sanastopoikkeamia on monia erilaisia, ja niiden kaikkien kuvaaminen universaalisti on hankalaa [Dorr et al. 2004]. Pisimmälle kehittynyt interlingua-malli on Yhdistyneiden Kansakuntienkin edistämä *Universal Networking Language* (UNL). YK perusti UNL:n kehitykselle oman voittoa tavoittelemattoman yhdistyksen, ja tutkimusprojektin missio on purkaa kielimuurit kaikkien luonnollisten kielten väliltä. Kehitystä on jo tehty ainakin 17 kielelle. [AlAnsary 2011]

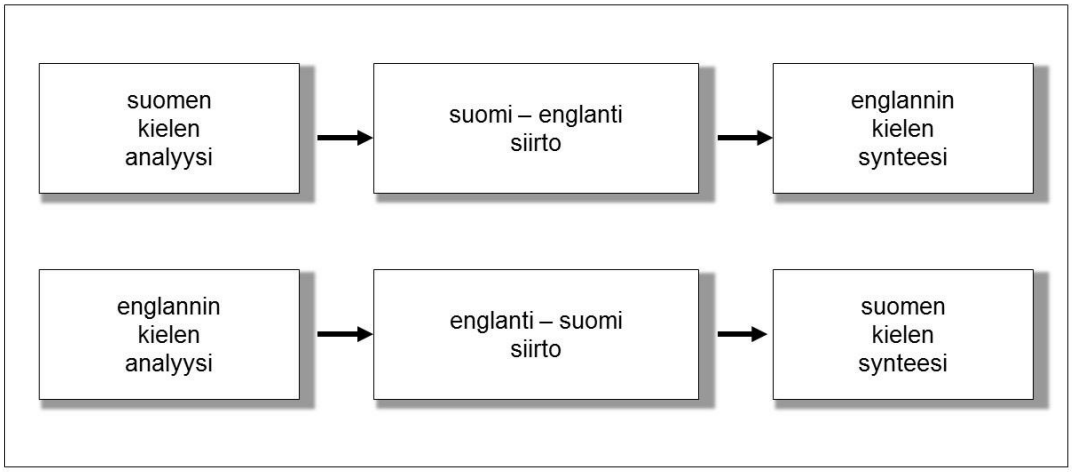


Kuva 2. Interlingua-arkkitehtuuri kuuden kieliparin tapauksessa [Austermühl 2001].

Konekäännösmenetelmien tarkka jaottelu on hankalaa. Konekääntämisen menetelmät jaettiin pitkään kolmeen peruskategoriaan, jotka olivat jo mainitut sääntöperustaiset, esimerkkipohjaiset sekä tilastolliset menetelmät. Menetelmistä on kuitenkin nykyisin hyvin paljon erilaisia versioita. Sääntöperustainen konekäännös voi olla myös välivaiheita käyttävä, vaikka varsinkin vanhimmat järjestelmät toteuttivat suoraa käännöstä [Hutchins and Somers 1992]. Esimerkkipohjainen menetelmä puolestaan yhdistää kieliopillisia keinoja tilastolliseen valintaan, ja sen vuoksi kyseistä menetelmää on vaikea lokeroida [Hutchins 2005]. Viime vuosikymmeninä on ollut yhä yleisempää tutkia ja kehittää *hybridimenetelmiä*, joissa kielioppisäännöt yhdistyvät tilastollisiin tai esimerkkipohjaisiin käännösmenetelmiin [Hutchins 2003; Costa-Jussà and Fonollosa 2015]. Hybridikääntäjä ei oikeastaan olekaan itsessään erillinen menetelmä, vaan käsite viittaa käännösohjelmiin, joissa on yhdistelty eri menetelmiä yhteen järjestelmään. Nykyaikaiset konekääntäjät ovatkin usein nimenomaan hybridikääntäjiä, jotka

ovat keskenään hyvin erilaisia järjestelmiä. Niistä kukin yhdistelee kääntämisen työkaluja sovellusalueensa tarpeen mukaan.

Austermühl [2001] käyttääkin menetelmistä erilaista jaottelua. Hän puhuu käännösstrategian sijaan arkkitehtuurista ja jakaa käännösjärjestelmät *suoraan arkkitehtuuriin, siirtoarkkitehtuuriin* ja *interlingua-arkkitehtuuriin*. Suoran käännöksen arkkitehtuuri nähtiin kuvassa 1 ja interlingua-menetelmä kuvassa 2. Kuvassa 3 esitetään siirtoarkkitehtuuri. Tässäkin menetelmässä joudutaan siirtämään tekstin merkitys yhdellä kieliparilla kerrallaan. Nykyjärjestelmissä siirtovaihe perustuu yleisimmin tilastollisiin menetelmiin. Myös Dorr [2006] jakaa konekäännösmenetelmät suoraan käännökseen, siirtomenetelmään ja interligua-menetelmään. Menetelmien erona on lähtökielen analyysin tarkkuus ja se, miten laajalti ne pyrkivät kieliriippumattomaan merkityksen ja tekstisisällön esitystapaan lähde- ja kohdekielen välillä [Dorr et al. 2004].



Kuva 3. Siirtoarkkitehtuuri [Austermühl 2001].

2.2. Konekääntämisen käyttökohteet

Alun perin konekääntäjistä toivottiin täydellisiä käännösautomaatteja, mutta nykyisin tiedetään tehtävän onnistumisen olevan hyvin tapauskohtaista. Kehnoillakin käännöksillä voi kuitenkin olla käyttöarvoa. Koneen tuottamaa haparoivaa käännöstä kutsutaan *raakakäännökseksi*, joka ei ole julkaisukelpoinen teksti. Eräs käyttökohde raakakäännökselle on alustavaan tiedonhakuun tarkoitettu silmäily, jossa lukija sietää virheellistä ja epäselvää kieltä. Tällaisen katsauksen perusteella voidaan selvittää esimerkiksi alustavaa taustatietoa jostakin aiheesta tai päättää, tarvitaanko tekstistä uusi, julkaisukelpoinen käännösversio. Nuutilan [2005] mukaan raakakäännöksiä tehdäänkin usein myös sellaisista teksteistä, joita ei muutoin käännettäisi ollenkaan. Raakakäännös voi myös toimia ihmiskääntäjän alustavana käännöksenä.

Lähdetekstin rajoittamisen on huomattu parantavan käännöslaatua tehokkaasti [Somers 2003]. Konekäännös soveltuu kääntäjäksi sitä paremmin, mitä yksinkertaisempaa ja termistöään toistavaa tekstiä käännetään. Jo johdannossa mainitsemani kanadalainen MÉTÉO-järjestelmä on hyvä esimerkki tästä. Vastaavasti Systran-nimistä järjestelmää käytetään menestyksekkäästi kääntämään mm. teknisten laitteiden käyttöohjeita, jotka on kirjoitettu yksinkertaistetulla kielellä, jossa toistuvat täsmälleen samat termit läpi dokumentin [Hutchins 2003]. Tekniset dokumentit tai muut samankaltaisina toistuvat, yksinkertaiset tekstit eivät motivoi ihmiskääntäjiä, joten ne ovat loistavia käyttökohteita konekääntäjille. Myös teollisuuden varaosalistojen tms. kääntämiseen konekäännös soveltuu erittäin hyvin, kun voidaan sivuuttaa lausetason semantiikka.

Tietojenkäsittelyn ja ohjelmistoalan kehittyminen 1990- ja 2000- luvuilla kiihdytti myös konekääntämisen laadun parantumista. Yksi nopeimmin kasvaneista konekäännössovelluksista olikin ohjelmistojen sekä verkkosivustojen lokalisointi. Eri aloille kehittyi monia järjestelmiä, jotka hyödynsivät MÉTÉO-järjestelmän tavoin *rajattua kieltä* (engl. sublanguage), eli konekäännös mahdollistettiin käyttämällä yksiselitteiseksi ja viralliseksi rajattua sanastoa ja alueelle tai dokumenttityypille sopivaa tyyliä tai lauserakenteita. [Hutchins 2010]

Internet on vaikuttanut konekäännöksen kehitykseen vahvasti. Kansainvälisen verkkoviestinnän myötä kehittyivät myös online-kääntäjät, joiden suosio on ollut suurta, vaikka niiden käännökset olivatkin aluksi heikkoja. Mielenkiinto niitä kohtaan Internetissä on osaltaan osoittanut, että kysyntää nopeille käännöksille on niin paljon, että laadusta ollaan valmiita tinkimään. Online-kääntäjien kuluttajasuosiosta huolimatta konekäännösalan tutkijakunta on syystä tai toisesta jättänyt ne melko pienelle huomiolle. [Hutchins 2010]

Globalisaation myötä tarve kääntämiselle kasvaa koko ajan, eivätkä ihmiset ehdi tekemään kaikkea tarvittavaa käännöstyötä. Koneilla teetettävän käännöstyön tarpeet voivat vaihdella kansainvälisen liike-elämän tarpeista yksityishenkilöiden kansainväliseen viestintään. Erilaisiin tarpeisiin tarvitaan erityyppisiä konekäännösjärjestelmiä. Suuret kansainväliset organisaatiot, pienet erityisalat, ammattikääntäjät ja kuluttajat tarvitsevat kaikki erilaisia työkaluja. Konekääntäminen lisääntyy koko ajan ja laajenee uusille alueille kuten patentteihin, ravintoloiden ruokalistoihin, sosiaaliseen mediaan, kehitysapuun ja television tekstityksiin. Käännösalan ammattilaisetkin tuntuvat lämmenneen konekäännökselle: kääntäjät käyttävät konekäännöstä ja muita kieliteknologisia apuvälineitä säännöllisesti työnsä eri vaiheissa.

2.3. Tietokone ja luonnolliset kielet

Useimmiten konekääntämisen ongelmat juontuvat luonnollisten kielten luontaisista monitulkintaisuuksista sekä sanaston ja rakenteen vastaavuuksien puutteesta eri kielten välillä [Hutchins 2003]. Luonnollisten kielten sujuva kääntäminen vaatii ihmiseltäkin enemmän kuin vain kahden kielen taitamista. On ymmärrettävä ainakin kulttuurierot, ja lisäksi hienovaraiset vivahteet: onko käsiteltävä teksti luonteva, vakuuttava, kiinnostava, elegantti, yksiselitteinen tai poeettinen? Tekstejä ei voi kääntää sanasta sanaan, sillä eri kielissä käytetään erilaisia rakenteita ja sanajärjestyksiä, kieliopeissa voi olla erilaisia sanojen sukuja, sijamuotoja, poikkeussääntöjä sekä synonymiaa. Onnistunut käännös vaatii siis luovuutta, jota tietokoneilta tunnetusti vielä puuttuu.

Yksi suurimmista ongelmista konekääntämisessä liittyy sanastolliseen ja rakenteelliseen monitulkintaisuuteen [Austermühl 2001; Hutchins 2003]. Sanaston tasolla monitulkintaisuus liittyy tietenkin sanoihin, joilla voi olla monia merkityksiä kuten jo mainittu `voita`, tai vaikkapa `kuusi` (numero ja puulaji). Esimerkiksi englannin kielen sana `light` voi kääntyä suomeksi esimerkiksi joko substantiivina `valo`, adjektiiveina `kevyt` tai `vaalea`, tai verbeinä `sytyttää` tai `valaista`. Lisäksi kielissä voi olla useita sanoja jollekin käsitteelle, jolle toisessa kielessä on vain yksi sana. Esimerkiksi englannin verbi `know` kääntyy suomeksi joko `tietää jotakin` tai `tuntea joku` (`henkilö`).

Monitulkintaisuuksien tulkitsemiseen on toki apukeinoja. Järjestelmään voidaan esimerkiksi merkitä sanojen sanaluokat, jolloin kääntäjä tietää, kääntääkö se substantiivia, adjektiivia vai verbiä. Joissain tapauksissa aiheen rajaus auttaa. Otetaan esimerkiksi englannin kielen sana `bank`: finanssitaloudesta kertovaa tekstiä kääntäessä voidaan poistaa käännösvaihtoehto `joenranta` ja jättää järjestelmän sanastoon vain `pankki`.

Lisäksi sanoille voidaan ohjelmoida järjestelmään ominaisuuksia, kuten *nestemäinen, naaraspuolinen, elollinen* tai *inhimillinen.* Edelleen voidaan merkitä esimerkiksi, että verbin `juoda` kanssa käytetään vain elollisia subjekteja. Tällaisetkin merkinnät ovat kuitenkin monimutkaisia, eikä niiden koodaaminen ole ongelmatonta. Eräs kuuluisa esimerkki on lause `The box is in the pen.` Sanan `pen` täytyy tässä tarkoittaa karsinaa tai leikkikehää, eikä kynää. Tällaisia yleismaailmallisia käsityksiä siitä, minkälaisten substantiivien *sisällä* voi olla mitäkin substantiiveja, on hankalaa, ellei mahdotonta selittää tietokoneille. [Somers 2003]

Rakenteellinen monitulkintaisuus puolestaan viittaa lauserakenteisiin. Esimerkiksi englannin kielessä prepositiofraaseja voidaan liittää eri asemiin lauseessa. Hutchins ja Somers [1992] käyttävät esimerkkinä lausetta `The man saw the horse with the telescope.` Tässäkin tapauksessa käännökseen tarvitaan käsitys maailmasta, jonka perusteella päätellään, ettei hevonen voi tavallisesti käyttää teleskooppia.

Vaikeuksia konekääntäjille aiheuttavat myös idiomit eli sanonnat, joiden merkitys ei ole kirjaimellinen, kuten `potkaista tyhjää` (engl. `kick the bucket`). Ne pitää kääntää kokonaisina, eikä sanasta sanaan, ja niitä voidaankin sisällyttää järjestelmien sanakirjoihin. Käytännössä kuitenkin usein päädytään tekstien esikäsittelyyn, jossa sanonnat poistetaan kokonaan ja korvataan yksiselitteisillä lauseilla [Austermühl 2001].

Lisäksi konekääntäjälle ongelmallisia ovat viittaussuhteet, eli pronominit ja muut sanat, joilla viitataan edeltävien lauseiden asioihin [Austermühl 2001]. Verrataan vaikkapa lauseita `Apina söi banaanin, koska se oli nälkäinen.` ja `Apina söi banaanin, koska se oli kypsä.` Jälleen nähdään, että ihmisen on helppo tulkita, että todennäköisimmin apina oli nälkäinen ja banaani kypsä, mutta konekääntäjälle asia voi olla monimutkaisempi. Englantiin kääntäessä ongelmaa ei synny, koska englannissa käytetään vastaavanlaista neutraalia viittausta `it`. Lukuisissa muissa kielissä viittaava sana valitaan kuitenkin laajemmasta valikoimasta, sillä sanoilla voi olla sukuja. Esimerkiksi saksassa `se` kääntyisi joko `er` (**der** Affe) tai `sie` (**die** Banane). Tässä tilanteessa järjestelmä voisi hyödyntää esimerkiksi merkintää siitä, että vain elollinen olento voi olla nälkäinen.

Virkkeiden väliset viittaussuhteet voivat olla hankalampia. Somers [2003] antaa esimerkin `The soldiers killed the women. They were buried the next day.` Jälleen `they` voi kääntyä esimerkiksi ranskaksi `ils` (viitaten sotilaisiin) tai `elles` (viitaten naisiin). Tämänkin tulkintaan tarvitaan käsitys siitä, että tappamisen objektia seuraa kuolema, jota seuraa edelleen hautaaminen. Tällaisen viittauksen koodaaminen konekääntäjään voi olla erittäin hankalaa. Laadukas kielenkääntäminen tuntuukin vaativan ihmisaivojen kognitiota. Nuutilan [2005] mukaan konekääntämisen ongelmat olisikin jo ratkaistu, jos ihmisaivojen toiminta pystyttäisiin mallintamaan.

Mainitsemieni ongelmien lisäksi konekäännöksessä voi esiintyä monia muitakin ongelmia, varsinkin tiettyjen kieliparien kesken, jos käsitellään rakenteellisesti todella erilaisia kieliä. Eräänä esimerkkinä voidaan käyttää saksan kielen eriäviä yhdysverbejä, joiden etuliite erotetaan perustilanteessa verbistä. Verbin taipuvaa pääosaa käytetään normaalilla paikallaan, mutta etuliite sijoitetaan lauseen loppuun viimeiseksi sanaksi. Nämä verbit tarkoittavat eri asioita ilman etuliitettä ja sen kanssa, joten lauseen merkitys muuttuu täysin, mikäli etuliite jää tulkitsematta. Esimerkiksi saksan verbi `rufen` tarkoittaa yksinään `huutaa`, mutta eriävä yhdysverbi `anrufen` tarkoittaa `soittaa puhelimella`. Tällaisetkin ongelmat on usein ratkaistu esikäsittelemällä tekstiä yksiselitteisemmäksi [Austermühl 2001].

Syntaktisista suhteista voidaan analysoida lukuisia asioita. Subjektin, verbin ja objektin välinen järjestys, mihin substantiiviin adjektiivi viittaa, lauseenvastikkeet, erilaiset fraasit ja sivulauseet ovat esimerkkejä syntaktisista suhteista, joille on vuosien saatossa kehitetty erityyppisiä jäsentimiä. Koska konekääntäjiä on kehitetty tietyille kielipareille, ovat eri kielten erityispiirteet motivoineet lukuisia erilaisia jäsentimiä. Niistä on sittemmin luotu jokaiseen järjestelmään sopivimpia yhä moninaisempia yhdistelmiä [Hutchins 2003]. Vielä 1960-luvulla epäiltiin, että syntaktisia monitulkintaisuuksia olisi liikaa, että laadukas konekääntäminen ikinä realisoituisi. Myöhemmin kuitenkin tekoälytutkimuksen edistys rohkaisi tietopohjaisten menetelmien tutkimusta, varsinkin rajattujen aihepiirien käännöksessä. Vaikkei konekäännös vielä nykyäänkään ole ihmiskääntäjän työn veroista, on tekoälyn keinoja otettu käyttöön kielioppisääntöjen, korpusten ja tilastollisten menetelmien rinnalle. Järjestelmien valtava kirjo on kuitenkin kehittynyt todella monimuotoiseksi. [Hutchins 2003; Costa-Jussà and Fonollosa 2015]

2.4. Konekäännöksen laadun arvioinnista

Konekäännösten laadun arviointi on monimutkaista. Laatuun vaikuttaa mm. tekstin pituus, tekstin aiheen laajuus, sanaston laajuus sekä lähde- ja kohdekielten ominaisuudet. Fiolan [2014] mukaan laatu riippuu aina arvioijasta, tämän käyttämistä painotuksista sekä käännettävän tekstin tehtävästä ja tarkoituksesta. Lisäksi hän huomauttaa, että käännöksen laatu riippuu myös lähdetekstin laadusta siinä mielessä, että ovathan esimerkiksi liian pitkät ja polveilevat virkkeet ja harvinainen sanasto konekääntäjien yleisiä kompastuskiviä.

Konekäännösten laadunarviointia kuitenkin tarvitaan, jotta tutkimuksissa voidaan osoittaa menetelmien välisiä eroja. Alun perin arviointia suorittivat ihmiset. Työmäärä ja työn hitaus kuitenkin herättivät mielenkiinnon automaattista laadunarviointia kohtaan. Papineni ja muut [2002] kehittivät automaattiseen arviointiin työkalun nimeltä BLEU, joka oli edullinen, nopea, kielestä riippumaton ja korreloi hyvin ihmisten antamien arvioiden kanssa. BLEU vakiinnutti asemansa alalla nopeasti ja se on yhä yksi laajimmin käytettyjä arviointimenetelmiä. Toisaalta sen on huomattu olevan epäluotettava voimakkaasti taipuvien kielten käännösten arvioinnissa, ja siitä onkin kehitetty myös laajennuksia. Eräs sellainen on LeBLEU, joka on suunniteltu nimenomaan voimakkaasti taipuvien kielten käsittelyyn [Virpioja and Grönroos 2015].

3. Suomen kielen konekääntäminen

Jokaisella luonnollisella kielellä on erityispiirteensä, jotka vaikuttavat konekääntämiseen. Millaista suomen kieli on ja miten koneet sen näkevät? Ja minkälaisilla keinoilla suomen kieltä mallinnetaan? Näihin kysymyksiin yritän vastata tässä

luvussa. Kohdassa 3.1 kerron suomen erityispiirteistä ja kohdassa 3.2 menetelmistä niiden käsittelemiseksi.

3.1. Suomen kielen erityispiirteet

Jo johdannossa mainitsemastani `voita`-esimerkistä on nähtävissä eräs suomen kielelle erityinen ominaisuus, ns. rikas morfologia. Tämä tarkoittaa samaa asiaa merkitsevän sanan lukuisia taivutusmuotoja: esimerkiksi verbi `voittaa`: `voitin`, `voitimmekohan`, `voittaakseen`, tai substantiivi `voi`: `voin`, `voissa`, `voiksikin` jne. Lukuisten mahdollisten taivutuspäätteiden ja -liitteiden johdosta suomen kielen nominilla voi olla jopa 2000 eri muotoa, verbillä peräti yli 12 000 [Koskenniemi et al. 2012]. Yhdistelmiä syntyy valtavasti, koska erilaiset liitteet ja päätteet riippuvat sijasta, luvusta sekä persoonasta ja lisäksi voidaan liittää vielä esimerkiksi kysymyspäätteitä. Mahdollisten liiteyhdistelmien määrä voi näin nousta suureksi. Esimerkiksi sana `halu+tu+imm+i+lla+mme+ko` koostuu seitsemästä eri osasta.

Lisäksi suomen kielen sanoja voidaan johtaa (`kirja` → `kirjasto`), ja johdosten lisäksi (myös useamman kuin kahden sanan muodostamia) yhdyssanoja on poikkeuksellisen paljon. Näin ollen suomen kielen sanat ovat myös suhteellisen pitkiä. Sanat muokkautuvat myös astevaihtelun ja vokaaliharmonian mukaan. Astevaihtelulla tarkoitetaan sanan vartalon konsonanttien muutoksia, kuten esimerkiksi pu**k**u → pu**v**un, ken**k**ä → ken**g**ästä, ran**t**a → ran**n**at jne. Vokaaliharmonia puolestaan määrittää, käytetäänkö sanojen päätteissä vokaaleja `a` ja `o` vai `ä` ja `ö`.

Sanajärjestys on suomessa yleisimmin SVX (subjekti, verbi, määreet), mutta muitakin sanajärjestyksiä käytetään ja niillä voidaan ilmaista esimerkiksi erilaisia painotuksia tai tutun ja uuden tiedon suhdetta. Vapaan sanajärjestyksen mahdollistaa mm. se, että lauseen tekijän ja kohteen tunnistaa sijapäätteestä:

`Poika osti kirjan.` ←→ `Kirjan poika osti.`

Suomen kielessä ei ole lainkaan kieliopillista sukua eikä artikkeleita. Niiden puuttuminen erottaa suomen monista kielistä, esimerkiksi englannista, saksasta, espanjasta ja ranskasta. Lisäksi suomen kielestä puuttuvat prepositiot, joiden tilalla siis käytämme erilaisia sijapäätteitä, esimerkiksi englanti-suomi-käännökset **`to`** `school` = `koulu`**`un`**, **`from`** `school` = `koulu`**`sta`**.

Voimakkaan taipumisen lisäksi on muitakin kääntämiseen vaikuttavia tekijöitä. Esimerkiksi kielen pieni puhujamäärä (maailmanlaajuisessa mittakaavassa) vaikeuttaa korpuspohjaisten menetelmien toimivuutta, sillä käytettävissä olevia tekstiaineistoja on suhteellisen vähän. Internet on toki luonut suuria tekstimassoja konekääntäjien käyttöön, mutta suomen kielen osalta se ei valitettavasti pidä paikkaansa, sillä käännösesimerkiksi kelpaavaa suomenkielistä asiatekstiä on kuitenkin suhteellisen vähän verkossa.

Kielemme on siten ollut hyvin pienessä roolissa maailman käännöstyössä. Euroopan Unionin jäsenyyden myötä suomen kielen asema kuitenkin parani, sillä suomi oli ensimmäistä kertaa kansainvälisen yhteisön virallinen kieli [Koskenniemi et al. 2012]. Suomen kielen kääntäminen lisääntyi merkittävästi, vaikkei suomi olekaan unionin työkieli, vaan suomeksi käännetään lähinnä esimerkiksi lainsäädäntöä.

3.2. Menetelmät suomen konekäännöksen taustalla

Sanojen voimakkaan taipumisen vuoksi suomen kielen kieliteknologiassa on keskitytty laajalti erilaisten morfologisten jäsentimien kehittämiseen. Jäsennin lisää halutun lingvistisen informaation sanojen yhteyteen. Yhtenä isona alkusysäyksenä suomen kielen konekäännöstutkimuksen kehittymiselle olivat Koskenniemen 1980-luvun julkaisut suomen kielen morfologisesta mallinnuksesta (kts. esim. [Koskenniemi 1983; 1984; Koskenniemi and Church 1988]).

Suomi on siis erityisen voimakkaasti taipuva kieli, ja taivutusmuodoista voidaan analysoida suurin osa syntaktisesta tiedosta, kuten selvittää lauseen objekti. Vain vähän taipuvissa kielissä, kuten englannissa, syntaktinen tieto pitää useimmiten kirjata käännösjärjestelmiin erillisinä sääntöinä. Suomen kielen analyysissä sääntöjä voidaan vähentää, koska taivutusmuotojen kuvaus riittää usein muodostamaan oikeellisen lauseen [Koskenniemi 1983]. Koska kielikohtaisia sääntöjä tarvittiin vain vähän, Koskenniemi [1983] esitti morfologiseen analyysiin ja synteesiin soveltuvan kaksitasoisen mallinsa sopivan lukuisille muillekin kielille, myös voimakkaasti taipuville. Mallia kokeiltiinkin pian soveltaa ainakin englannin, japanin, romanian, ranskan, ruotsin, kreikan, saamen, islannin ja arabian kielille [Koskenniemi 1984].

Koskenniemen [1983; 1984] kehittämä algoritmi tarjosi kielestä riippumattoman alustan taivutusmuotojen käsittelyyn useissa sovelluksissa kuten konekääntämisessä, tiedonhaussa ja oikolukuohjelmissa. Monikäyttöisen siitä tekee mm. sen soveltuminen sekä tekstin analyysiin että synteesiin [Koskenniemi and Church 1988]. Erilaisia äärellistilaisia morfologisia jäsentimiä hyödynnetäänkin nykyisin konekäännöksen lisäksi useissa muissakin kieliteknologian sovelluksissa [Pirinen 2011; 2015].

Europarlamentin dokumentteja alettiin hyödyntää monikielisinä korpuksina 2000-luvulla, sillä se oli luonteva lähde tekstimassoille, joissa toistuu sama sisältö lukuisilla eri kielillä. Koehn [2005] keräsi Europarlamentin verkkojulkaisuista rinnakkaistekstikorpuksen Europarl, joka sisälsi tekstejä yhdellätoista EU:n virallisella kielellä, ja siitä kiinnostuttiin laajalti kieliteknologian alalla. Korpusta käytettiin opetusdatana tilastolliselle konekääntämisjärjestelmälle, ja sitä kokeiltiin jopa 110 kieliparille, jotta nähtäisiin mahdollisimman laajalti sen vaikutuksia.

Mukaan pääsi ensimmäistä kertaa myös harvoin (jos koskaan) tutkittuja kieliparеja kuten mm. suomi-kreikka. Eri kieliparien käännöslaadut vaihtelivat rajusti, mikä osoitti, etteivät kaikkien kielten kääntämiseen sovellu samat menetelmät. [Koehn 2005]

Suomen kielen kääntäminen todettiin tuossa kokeilussa erityisen haastavaksi. Sen pääteltiin johtuvan kielen rikkaasta morfologiasta, mistä johtuu myös valtava sanaston koko. Suomen kielen sanasto onkin noin viisinkertainen englantiin verrattuna, koska samaa merkitsevistä sanoista on lukuisia erilaisia taivutusmuotoja. Tämä johtaa edelleen harvinaisen datan ongelmaan tilastollisessa kääntämisessä, sillä yksittäiset sanat esiintyvät teksteissä harvemmin. [Koehn 2005]

4. Ajankohtainen kotimainen tutkimus

Kotimaisen konekäännöstyön perustana ovat yhä erilaiset morfologiset jäsentimet. Suomen kielen taivutus vaatii erityisiä toimia, jotta oikeat vastaavuudet löytyvät vieraista kielistä varsinkin silloin, kun käännettävän kieliparin toinen kieli on rakenteeltaan hyvin erilainen.

Pirinen [2008; 2011; 2015] on kehittänyt Koskenniemen kaksitasoiseen malliin perustuvasta äärellistilaisesta morfologisesta jäsentimestä avoimen lähdekoodin toteutuksen nimeltään OMorFi. Sen avulla pyrittiin laajempaan yhteistyöhön eri kielten ja tutkimusryhmien kesken koko kieliteknologian alalla, ja siksi jäsennin suunniteltiinkin helposti muokattavaksi eri sovellusalueiden tarpeiden mukaan [Pirinen 2011]. OMorFi on sääntöihin perustuva jäsennin, jota onkin käytetty paljon sääntöpohjaisen konekäännöksen pohjana. Nykymuodossaan OMorFi on kehittynyt oikeastaan tietokannaksi (engl. lexical database tai lexicographical database), joka sisältää suomen kielen sanaston taipumisen säännöt ja muun kielitieteellisen informaation, jonka avulla kieltä voidaan prosessoida digitaalisesti. Tietokantaa on helppo päivittää ja ylläpitää, niin että se onnistuu myös kielitieteilijöiltä ja muilta projektiin osallistuvilta. Lisäksi se on suunniteltu niin robustiksi, että sitä käyttävät sovellukset eivät häiriinny päivityksistä tai muiden sovellusten omiin tietoihinsa tekemistä muutoksista. [Pirinen 2015]

Tilastollisen konekäännöksen tarpeisiin on kehitetty myös toisenlaisia työkaluja morfologian käsittelyyn. Eräs menetelmä on Morfessor-algoritmi, joka etsii tekstistä sanojen sijaan *morfeja* eli sanan osia kuten vartalo ja päätteet [Virpioja 2012]. Tilastollinen konekääntäminen on yleensä keskittynyt sanojen, fraasien tai jopa kokonaisten lauseiden käsittelyyn. Tämä lähestymistapa ei kuitenkaan ole tehokas voimakkaasti taipuville kielille. Tähän ongelmaan on etsitty apua Morfessorista, joka hajottaa taipuneita sanoja vartaloon ja sen liitteisiin. Konekääntäminen vaatii lähdetekstin analyysia, mitä varten on ensimmäiseksi valittava sopiva *leksikaalinen perusyksikkö*, joka esimerkiksi englannin kielessä on yksittäinen

sana, mutta suomen kielessä sanan sijasta tarvitaan pienempi yksikkö, siis morfi [Virpioja 2012].

Morfessor käyttää ohjaamatonta tai osittain ohjattua koneoppimista, ja se on todettu tehokkaaksi ja hyödylliseksi työkaluksi voimakkaasti taipuvien kielten tilastollisessa konekäännöksessä [Virpioja 2012; Virpioja et al. 2013]. Se oppii raakatekstidatan avulla pilkkomaan sanat morfeiksi. Morfessor on julkaistu vapaalla ohjelmistolisenssillä, joten se on käytettävissä vapaasti kuten Omorfikin [Virpioja et al. 2013]. Siitä on myös muokattu erilaisia versioita kuten Morfessor FlatCat. FlatCat-versiota puolestaan on pyritty tehostamaan hyödyntämällä takaisinkytkeytyviä neuroverkkoja (engl. recurrent neural networks tai RNN). Viimeisimmissä tutkimuksissa on käytetty täysin ohjaamatonta oppimista, minkä johdosta menetelmää voitaisiin hyödyntää myös muille voimakkaasti taipuville kielille ilman lisäkonfiguraatioita tai järjestelmämuutoksia. [Grönroos et al. 2015]

Kolmas avoimen lähdekoodin periaatteella luotu työkalu on FinnPos. Tämäkin koneoppimiseen perustuva menetelmä merkitsee tekstiin morfien merkityksiä ja sanojen perusmuotoja, joiden avulla konekääntäjä löytää oikeat vastaavat sanat taipumattoman kohdekielen sanastosta. Perusmuotoisten sanojen merkintä (engl. lemmatization) perustuu osittain OMorFi-järjestelmän analyysiin, mutta sitä on tehostettu lisätoiminnoilla [Silfverberg et al. 2016]. FinnPosin on todettu olevan tehokkaampi ja tarkempi morfologian jäsentämisessä ja merkinnässä kuin monen muun eurooppalaisen menetelmän [Silfverberg et al. 2016].

Parhaiden konekääntäjien etsimistä jatketaan yhä tutkimalla erilaisia hybridimenetelmiä, joissa yhdistyvät mm. sääntöjen, todennäköisyyslaskennan, hakurobottien, koneoppimisen ja neuroverkkojen keinot [Rubino et al. 2015, Tiedemann et al. 2016]. Tietojenkäsittelyn yhä kehittyessä tutkijoiden keinovalikoima saattaa jopa laajentua entisestään. Kaiken kaikkiaan käännöslaatu paranee koko ajan. Tällä hetkellä neuroverkkojen yhdistäminen tilastollisiin menetelmiin näyttäisi erittäin lupaavalta menetelmältä.

Eräs selkeä laatuun vaikuttava seikka on opetusdatan laajuus. Tutkimukset ovat todentaneet useasti, että opetusdatan eli konekääntäjän käytössä olevien tekstikorpusten laajentaminen parantaa huomattavasti erilaisten tilastollista laskentaa hyödyntävien järjestelmien käännöslaatua [Tiedemann et al. 2016].

5. Pohdintaa

Tämän hetken suosituin ilmainen online-kääntäjä Google Translate on osoittanut suosiollaan nopean konekäännöksen olevan erittäin tarpeellinen työkalu nykymaailmassa, käännöslaadusta riippumatta. Svobodan [2014] mukaan Googlen kääntäjällä käännetään jopa miljardi kirjoitusta päivässä, mikä vastaa tekstimassaltaan miljoonaa kirjaa päivässä.

Google-kääntäjän laatu onkin parantunut huimasti viime vuosina. Ohjelma hyödyntää nykyisin tekoälyn ja neuroverkkojen keinoja, ja sen on raportoitu kääntävän helppoja asiatekstejä englannista hyväksi selkeäksi suomeksi. Suurimmilla kielillä tekoälyyn siirryttiin jo vuonna 2016, ja suomen osalta teknologia päivitettiin huhtikuussa 2017. Päivityksen jälkeen käännös arvioitiin erittäin laadukkaaksi neljä kertaa useammin kuin vanhan teknologian aikana. [Puttonen 2017]

Konekääntäminen aiheuttaa toisaalta myös huolta. Tilastollisen konekäännöksen voidaan nähdä köyhdyttävän kieltä, kun se tekee aina kaikkein todennäköisimmät sanavalinnat. Svobodan [2014] mielestä on epätodennäköistä, että kone muuttuisi ihmisen kaltaiseksi, mutta sen sijaan ihminen saattaa muuntaa omaa kieltään köyhemmäksi, jos kaikki tekstit muokattaisiin koneen helposti ymmärtämään muotoon. Lisäksi hän on huolissaan skenaariosta, jossa kaikki käännös olisi saatavilla suoraan lisätyssä todellisuudessa kuten vaikkapa älylaseissa: Mitä jos kielimuurien murtuminen johtaakin entistä syvempiin kulttuurien välisiin kuiluihin, sillä ihmisillä ei enää olisi motivaatiota perehtyä vieraisiin kulttuureihin? Google Translaten suunnittelupäällikkö Macduff Hughes ei puolestaan usko, että konekäännös vähentää ihmisten uteliaisuutta opetella vieraita kieliä, sillä kasvokkain keskustelu on ihmiselle luontevin ja kiehtovin tapa kommunikoida [Puttonen 2017].

Konekäännös on vakiinnuttanut paikkansa uutena teknologiana, mutta kaikkeen kääntämiseen se ei kuitenkaan sovellu. Kaunokirjallisuuteen konekäännöstä ei voi hyödyntää, sillä täsmällisen tilastollisen toiston sijaan kaunokirjallisuuden tarkoitus on päinvastoin luoda uusia kerronnan ja kielen keinoja. Pitäisin kaunokirjallisuuden konekääntämistä jopa erittäin huonona ajatuksena. Surullinen esimerkki oli Suomalaisen kirjakaupan valikoimista löytynyt konekäännetty Kuningas Lear, jossa ”Kuningas on kauhea rage” [Määttänen 2016].

6. Yhteenveto

Konekääntämistä on kehitetty pitkään, mutta se on osoittautunut hankalaksi tietojenkäsittelyongelmaksi. Vaikka teknologia ei yllä yhtä tarkkoihin ja luoviin käännöksiin kuin ihminen, on konekäännös kuitenkin vakiinnuttanut paikkansa maailmassa. Se on ylivoimainen työkalu nopeaan tietoon vieraskielisistä lähteistä ja se mahdollistaa jokaiselle internetin käyttäjälle kansainvälisen kommunikoinnin lukuisilla kielillä. Konekäännös sopii erityisen hyvin selkeästi rajattuihin asiateksteihin, mutta konekääntäjä ei osaa huomioida kirjoitusvirheitä tai tekstilajin tyyliä.

Suomen kielen kääntäminen on erityisen hankalaa, sillä suomen kieli on poikkeuksellisen voimakkaasti taipuva kieli. Se erottaa suomen useimmista niistä val-

takielistä, joille käännöstarpeita eniten on ollut. Taipuneiden sanamuotojen käsittelyyn on luotu morfologisia jäsentimiä, joiden avulla taivutusmuotoihin ja niiden lukuisiin liitteisiin sisältyvä kieliopillinen tieto saadaan purettua konekääntäjän käyttöön. Pienen kansan kielellä ei ole myöskään saatavilla vielä kovin suuria esimerkkitekstimassoja, joiden avulla tilastolliset käännösmenetelmät paranisivat huomattavasti. Maailman valtakielillä konekäännös toimii jo varsin hyvin, sillä viime vuosina paljon tutkitut neuroverkot ovat parantaneet tilastollisten menetelmien käännöslaatua merkittävästi. Suomen kielen osalta saamme vielä odottaa, vaikka käännöslaatu paraneekin jatkuvasti.

Viiteluettelo

AlAnsary, S. 2011. Interlingua-based Machine Translation Systems: UNL versus Other Interlinguas. In: *Proceedings of the 11th International Conference on Language Engineering*.

Austermühl, Frank. 2001. *Electronic Tools for Translators* (Vol. 2). Routledge.

Costa-Jussà, M. R., & Fonollosa, J. A. R. 2015. Latest trends in hybrid machine translation and its applications. *Computer Speech & Language, 32*(1), 3–10.

Dorr, B. J., Hovy, E. H., & Levin, L. S. 2006. Machine translation: Interlingual methods. In K. Brown (Ed.), *Encyclopedia of Language & Linguistics (second edition)*. Elsevier.

Fiola, Marco A. 2014. Machine Translation vs. Human Translation: The Good, the Bad and the Useless. In: *Man Vs. Machine?: The Future of Translators, Interpreters and Terminologists: Proceedings of the XXth FIT World Congress Volume 1*. 38–45.

Firat, O., Cho, K., Sankaran, B., Yarman Vural, F. T., & Bengio, Y. 2016. Multi-way, multilingual neural machine translation. *Computer Speech & Language* (In Press, Corrected Proof).

Grönroos, Stig-Arne, Virpioja, Sami and Kurimo, Mikko. 2015. Tuning phrase-based segmented translation for a morphologically complex target language. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, 105–111.

Hutchins, John 1986. *Machine Translation: Past, Present, Future*. Ellis Horwood.

Hutchins, John. 2003. Machine translation. In: *Encyclopedia of Computer Science (4th edition)*, 1059–1066.

Hutchins, John. 2005. Towards a definition of example-based machine translation. In *Machine Translation Summit X, Second Workshop on Example-Based Machine Translation*. 63–70.

Hutchins, John. 2010. Machine translation: A concise history. *Journal of Translation Studies*, 13, 1-2, 29–70.

Hutchins, John, & Somers, Howard. 1992. *An Introduction to Machine Translation* (Vol. 362). Academic Press.

Koehn, P. 2005. Europarl: A parallel corpus for statistical machine translation. In: *MT Summit* Vol. 5, 79-86.

Koskenniemi, Kimmo. 1983. Two-Level Model for Morphological Analysis. In *IJCAI* Vol. 83, 683–685.

Koskenniemi, Kimmo. 1984. A general computational model for word-form recognition and production. In: *Proceedings of the 10th International Conference on Computational Linguistics*, 178–181.

Koskenniemi, K., & Church, K. W. 1988. Complexity, two-level morphology and finnish. In: *Proceedings of the 12th Conference on Computational Linguistics-Volume 1*, 335–340.

Koskenniemi, Kimmo, Georg Rehm, and Hans Uszkoreit. 2012. *The Finnish Language in the Digital Age.* Springer.

Lopez, Adam. 2008. Statistical Machine Translation. *ACM Computing Surveys*, 40, 3, 1–49.

Määttänen, Juuso. 2016. Suomalaisen kirjakaupan konekäännetty Shakespeare hämmästyttää netissä: ”Kuningas on kauhea rage”. Helsingin Sanomat 25.8.2016. Saatavilla: http://www.hs.fi/kulttuuri/art-2000002917673.html [viitattu 15.5.2017]

Nuutila, P. 2005. *Rough Machine Translation in the Communication Process*. Licenciate Thesis. University of Tampere. School of Modern Languages and Translation Studies.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. 2002. BLEU: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. 311-318.

Pirinen, Tommi. 2008. *Suomen kielen äärellistilainen morfologinen jäsennin avoimen lähdekoodin resurssein.* Pro gradu -tutkielma, Helsingin yliopisto, Yleisen kielitieteen laitos.

Pirinen, Tommi A. 2011. Modularisation of finnish finite-state language description—towards wide collaboration in open source development of a morphological analyser. In: *Proceedings of the 18th Nordic Conference of Computational Linguistics (NODALIDA)*, 299–302.

Pirinen, Tommi A. 2015. Omorfi—Free and open source morphological lexical database for finnish. In: *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA)*, (109) 313–315.

Puttonen, Mikko. 2017. Testasimme uuden Google-kääntäjän – tekoälyn avulla helppo asiateksti kääntyy nyt selväksi suomeksi, kaunokirjallisuus ei edelleenkään. Helsingin Sanomat 20.4.2017. Saatavilla: http://www.hs.fi/tiede/art-2000005178390.html [Viitattu 3.5.2017]

Rubino, R., Pirinen, T., Espla-Gomis, M., Ljubešic, N., Ortiz Rojas, S., Papavassiliou, V., ... & Toral, A. 2015. Abu-MaTran at wmt 2015 translation task: Morphological segmentation and web crawling. In: *Proceedings of the Tenth Workshop on Statistical Machine Translation,* 184-191.

Sager, J. C. 1994. *Language Engineering and Translation: Consequences of Automation* (Vol. 1). John Benjamins Publishing.

Silfverberg, M., Ruokolainen, T., Lindén, K., & Kurimo, M. 2016. FinnPos: an open-source morphological tagging and lemmatization toolkit for Finnish. *Language Resources and Evaluation*, *50*(4), 863-878.

Somers, H. (Ed.). 2003. *Computers and Translation: A Translator's Guide* (Vol. 35). John Benjamins Publishing.

Svoboda, Tomáš. 2014. Man and Machine: Translation in the Era of Augmented Reality. In: *Man Vs. Machine?: The Future of Translators, Interpreters and Terminologists : Proceedings of the XXth FIT World Congress Volume 1.* 93–99.

Tiedemann, J., Cap, F., Kanerva, J., Ginter, F., Stymne, S., Ostling, R., & Weller-Di Marco, M. 2016. Phrase-Based SMT for Finnish with More Data, Better Models and Alternative Alignment and Translation Tools. In *Proceedings of the First Conference on Machine Translation*. Vol. 2. 391–398.

Virpioja Sami. 2012. *Learning Constructions of Natural Language: Statistical Models and Evaluations.* Doctoral dissertation, Aalto University.

Virpioja, S., Smit, P., Grönroos, S., & Kurimo, M. 2013. *Morfessor 2.0: Python implementation and extensions for morfessor baseline*. Aalto University Publication Series SCIENCE + TECHNOLOGY 25/2013.

Virpioja and Grönroos. 2015. LeBLEU: N-gram-based Translation Evaluation Score for Morphologically Complex Languages. In: *Proc. of The Tenth Workshop on Statistical Machine Translation (WMT15),* 411-416.

Luolastojen proseduraalinen generointi videopeleissä

Rama Hannula

Tiivistelmä

Videopelien suosio on jatkuvassa kasvussa ja siksi tarvitaan uusia tapoja tuottaa pelisisältöä entistä kustannustehokkaammin. Sisällön generoiminen proseduraalisesti on yksi ratkaisu tähän ongelmaan, ja luolasto on eräs sisällönmuoto, joka soveltuu erityisen hyvin tuotettavaksi proseduraalisesti. Tässä tutkielmassa käsittelen luolastojen proseduraalista generointia videopeleissä yleisesti ja esittelen joitakin erilaisia siihen käytettyjä menetelmiä. Yksinkertaisin näistä menetelmistä on sisällön valinta, jossa luolasto kootaan valmiiksi tehdyistä palasista. Soluautomaateilla taas on mahdollista tuottaa luolastoja, jotka vaikuttavat luonnon muovaamilta. Avaruuden ositusta hyödyntämällä voidaan tuottaa hyvin jäsenneltyjä luolastoja. Evoluutioalgoritmit tarjoavat parhaat mahdollisuudet lopputuloksen hallintaan, mutta eivät ole yleensä riittävän tehokkaita käytettäväksi pelaamisen aikana.

Avainsanat ja -sanonnat: Proseduraalinen sisällöntuotanto, videopelit, luolastot.

1. Johdanto

Videopelit ovat alati kasvava viihteen laji, ja yhä useammat ihmiset eri ikäryhmistä pelaavat videopelejä säännöllisesti. Uusi teknologia tarjoaa kehittäjille mahdollisuuksia yhä monimutkaisempiin ja hienostuneempiin ratkaisuihin. Teknologian kehittymisen ja pelaamisen suosion kasvun myötä videopelien vaatimukset ja samalla myös niiden kehittämiseen käytetyn rahan määrä on kasvanut huomattavasti. Suurten pelitalojen kehittämien pelien tekemisessä on mukana satoja työntekijöitä ja budjetit liikkuvat miljoonissa euroissa.

Pelien sisällön tuottaminen on yksi merkittävimmistä kuluista pelien kehityksessä. Tuottamalla sisältöä proseduraalisesti eli automaattisesti algoritmien avulla sen sijaan, että työntekijät tuottaisivat sisältöä manuaalisesti, voitaisiin vähentää tätä kuluerää huomattavasti. [Hendrikx *et al.* 2013] Lisäksi monipuolisesti käytetty proseduraalinen sisällöntuotanto mahdollistaa lähes äärettömän määrän mahdollisuuksia ja tuo siten peliin ennalta-arvaamattomuutta ja uudelleenpelattavuusarvoa, joita voidaan pitää useissa peleissä haluttavina ominaisuuksina.

Luolastot ovat eräs peleissä yleisesti esiintyvä ympäristö, joka soveltuu erinomaisesti tuotettavaksi proseduraalisesti. Luolastojen tarkasti jäsennelty rakenne on vahvasti sidoksissa pelattavuuteen, jolloin luolaston proseduraali-

sella generoinnilla voidaan saada aikaan merkittäviä vaikutuksia pelikokemukseen. Lisäksi on melko helppoa suunnitella algoritmeja, jotka kykenevät tuottamaan mielekkäitä luolastoille tyypillisiä rakenteita.

Tässä tutkielmassa tarkastelen erilaisia luolastojen proseduraaliseen generointiin käytettyjä menetelmiä, jotka ovat sisällön valinta, soluautomaatit, avaruuden ositus ja geneettiset algoritmit. Luvussa 2 määrittelen proseduraalisen sisällöntuotannon käsitteen ja käsittelen sen etuja ja ongelmia. Luvussa 3 esittelen luolaston videopelien kontekstissa. Luvussa 4 esittelen menetelmiä luolastojen proseduraaliseen generointiin.

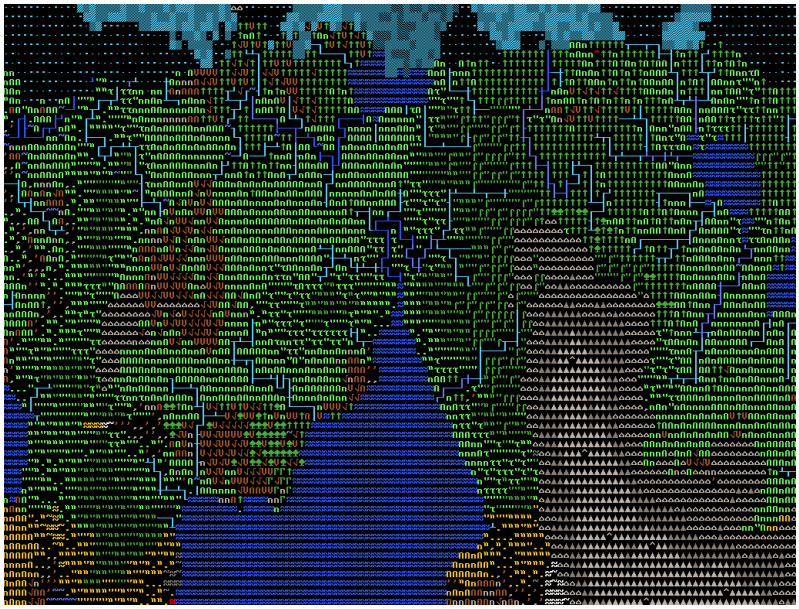
2. Proseduraalinen sisällöntuotanto videopeleissä

2.1. Proseduraalinen sisällöntuotanto

Pelin *sisältö* (content) koostuu lähes kaikesta, mitä pelissä on, kuten esimerkiksi tekstuureista, äänistä, kentistä ja juonesta. Sisällöksi ei lasketa pelimoottoria, vaan se on pikemminkin pohja, jolle sisältö rakentuu. Myöskään tekoälyä ei pidetä sisältönä. [Shaker *et al*. 2016]

Proseduraalinen sisällöntuotanto (procedural content generation) peleissä tarkoittaa pelisisällön automaattista tuottamista algoritmisesti rajoitetun tai epäsuoran käyttäjäsyötteen pohjalta. Proseduraalisesta sisällöntuotannosta käytetään joskus myös hieman virheellisesti termiä *satunnaisgenerointi* (random generation), mutta vaikka satunnaisuus onkin usein mukana proseduraalisessa sisällöntuotannossa, ei se ole välttämätöntä. Tärkeää on, että käyttäjällä ei ole suoraa valtaa tuotettuun sisältöön, vaan algoritmi perustuu ainakin osittain satunnaisuuteen tai käyttäjän epäsuoraan syötteeseen, kuten pelaajan aiempaan toimintaan pelissä. [Togelius *et al*. 2011a]

Yhtenä ääriesimerkkinä proseduraalisesta sisällöntuotannosta voidaan pitää Tarn ja Zach Adamsin kehittämää Dwarf Fortress -peliä, jossa luodaan proseduraalisesti kokonainen maailma, sen historia ja sitä asuttavat olennot ja sivilisaatiot. Pelaaja voi ainoastaan säätää maailman luomiseen vaikuttavia parametreja, kuten sivilisaatioiden lukumäärää, maailman kokoa ja maaston piirteitä, mutta maailma luodaan muuten satunnaislukugeneraattoria käyttäen. Kuvassa 1 on eräs Dwarf Fortress -pelissä tuotettu maailmankartta tilassa ennen sivilisaatioiden luomista.



Kuva 1. Dwarf Fortress -pelissä satunnaislukugeneraattoria ja pelaajan antamia parametreja käyttämällä luotu maailmankartta.

Vaikka annetut parametrit olisivat täsmälleen samat siemenlukua lukuun ottamatta, syntyy silti joka kerta erilainen maailma. Toisaalta aivan identtinen maailma voidaan tuottaa varmuudella, jos käytetään täysin samoja parametreja ja siemenlukua.

2.2. Proseduraalisen sisällöntuotannon etuja ja ongelmia

Kun sisältöä voidaan tuottaa proseduraalisesti, ei tarvita enää yhtä paljon työntekijöitä tuottamaan sisältöä manuaalisesti. Koska sisällön tuottaminen manuaalisesti on tyypillisesti kallista ja aikaa vievää, näin voidaan säästää pelien valmistuskustannuksissa ja nopeuttaa kehitystä [Shaker *et al*. 2016]. Lisäksi pienemmätkin pelitalot voivat tarjota suuria määriä sisältöä, johon heillä ei perinteisillä menetelmillä olisi varaa.

Proseduraalisella sisällöntuotannolla voidaan tuottaa lähes rajattomasti erilaista sisältöä. Näin voidaan varmistaa, että jokainen pelikerta on erilainen ja ainutlaatuinen. Kun pelit pystyvät jatkuvasti tarjoamaan uutta ja sen vuoksi myös kiinnostavaa sisältöä, niiden uudelleenpelattavuusarvo kasvaa. Proseduraalisen sisällöntuotannon tarjoama mahdollisuus ennalta-arvaamattomuuteen voi myös muuttaa pelin pelaamista merkittävästi, sillä pelaaja ei voi enää toimia muistin varassa, vaan joutuu reagoimaan ja sopeutumaan jatkuvasti uusiin tilanteisiin [Smith 2014]. Tällaista pelikokemusta on vaikea saada aikaan muulla tavalla.

Proseduraalisen sisällöntuotannon heikkous on sen huono hallittavuus. On vaikeaa kehittää sellaista menetelmää, joka pystyy joka kerta tuottamaan sa-

man tasoista sisältöä kuin ihmissuunnittelija. Menetelmän kehittämisen vaikeus vaihtelee tuotettavan sisällön tyypin ja laajuuden mukaan.

3. Luolastojen generointi

Luolasto (dungeon) on erityisesti rooli- ja seikkailupeleissä usein esiintyvä labyrinttimainen ympäristö, jossa pelaaja seikkailee. Luolasto voi koostua sekä luonnon muovaamista että ihmiskäden tekemistä rakennelmista, jotka ovat yhteydessä toisiinsa. [van der Linden *et al*. 2014] Yksittäistä tällaisen rakennelman luomaa tilaa kutsutaan huoneeksi ja useita huoneita yhdistävää reittiä käytäväksi. Luolaston sisällä on yleensä erilaisia haasteita, pulmia ja palkintoja pelaajalle, kuten esimerkiksi ansoja, vihollisia ja aarteita [van der Linden *et al*. 2014]. Pelaajalla on vapaus tutkia luolastoa vapaasti, mutta koska pelaajan on edettävä luolaston rakenteita ja haasteita mukaillen, ne tarjoavat hyvin jäsennellyn pelinkulun. [van der Linden *et al*. 2014] Yleensä pelaajan on siis kuljettava luolaston eri osien läpi tietyssä järjestyksessä voidakseen läpäistä sen.

Luolastojen proseduraalisella generoinnilla tarkoitetaan topologian, geometrian ja peliobjektien generointia. Yleensä generointimenetelmä koostuu kolmesta osasta. Ensiksi tarvitaan yksinkertaistettu esitysmalli, jolla voidaan esittää luolaston yleiskuvaus. Yleiskuvaus on abstrakti esitys luolaston rakenteesta ja sen sisältämistä peliobjekteista, eikä se ota kantaa siihen, miten lopullinen luolastogeometria tuotetaan. Toiseksi tarvitaan menetelmä, joka pystyy generoimaan yleiskuvauksia yksinkertaistetun esitysmallin pohjalta. Kolmanneksi tarvitaan menetelmä, joka luo lopullisen luolastogeometrian ja luolaston sisällön yleiskuvauksen pohjalta. [Shaker *et al*. 2016] Yleiskuvauksen muuttaminen geometriaksi on vahvasti yksilöllistä jokaisen pelin kohdalla, ja tämän takia sitä ei ole mielekästä tarkastella tässä tutkielmassa.

Luolastojen tarkkaan jäsennelty rakenne mahdollistaa hyvin hallittavien luolastogeneraattorien laatimisen. Tämä takia luolastot soveltuvat erityisen hyvin tuotettavaksi proseduraalisesti. Yksinkertaisimmillaan luolaston yleiskuvauksen voisi tuottaa luomalla huoneita täysin satunnaisiin sijainteihin, ja yhdistelemällä niitä sitten satunnaisesti käytävillä. Lähes aina on kuitenkin tarpeen varmistaa, että kaikki luolaston huoneet ovat yhteydessä toisiinsa ja saavutettavissa, niin että koko luolaston läpi on mahdollista kulkea. Hyvän generointialgoritmin tulee siis aina tuottaa läpäistävissä olevia tasoja, jottei pelaaja voi jäädä luolastoon jumiin.

4. Menetelmiä

Luolaston proseduraaliseen tuottamiseen ei ole olemassa yhtä menetelmää, joka olisi selvästi muita parempi, vaan erilaiset menetelmät tuottavat erilaisia

tuloksia ja toimivat eri tilanteissa. Luolastojen proseduraaliseen generoimiseen on käytetty hyvin paljon erilaisia menetelmiä ja niiden yhdistelmiä, joista useimmat ovat hyvin pelikohtaisia. Tässä luvussa käsitellään joitakin yleisellä tasolla toimivia menetelmiä, joita voidaan soveltaa monenlaisiin tilanteisiin.

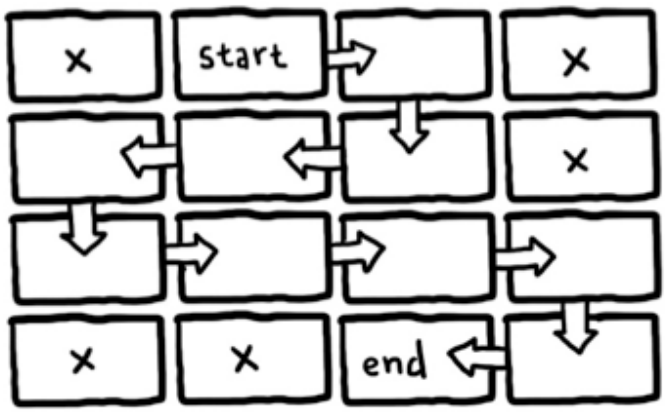
Menetelmät voidaan jakaa karkeasti kahteen osaan: *konstruktiivisiin* (constructive) ja *hakuperustaisiin* (search-based). Konstruktiiviset menetelmät tuottavat tuloksen heti ensimmäisellä ajokerralla ja ovat siten yleisesti ottaen nopeampia. Tämän takia niillä on usein mahdollista tuottaa luolastoja ajonaikaisesti. Konstruktiivisilla menetelmillä on kuitenkin usein taipumus olla vaikeammin hallittavia.

Hakuperustaiset algoritmit pyrkivät tuottamaan optimaalisen ratkaisun optimointiongelmaan. Ne perustuvat ajatukseen siitä, että ongelmaan on olemassa riittävän hyvä ratkaisu jossakin ratkaisujen joukossa, ja se voidaan lopulta löytää iteroimalla ja muokkaamalla mahdollisia ratkaisuja, pitäen hyvät muutokset ja poistaen huonot. Hakuperustaiset menetelmät eivät yleensä sovellu tuottamaan luolastoja ajon aikana hitautensa takia, mutta tarjoavat enemmän mahdollisuuksia lopputuloksen hallintaan. [Shaker *et al*. 2016]

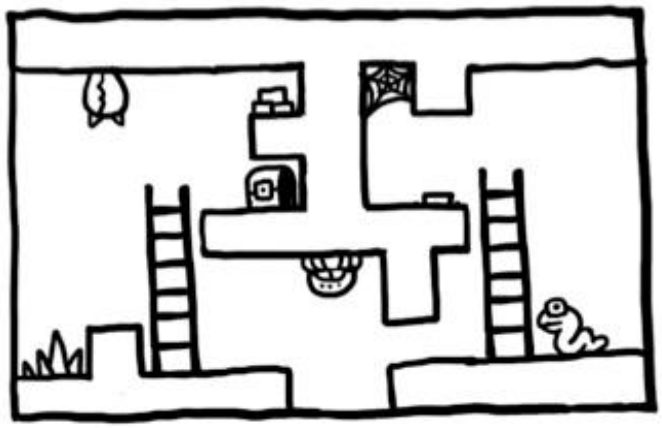
4.1. Sisällön valinta

Sisällön valinta on hyvin yksinkertainen konstruktiivinen tapa tuottaa luolastoja. Siinä generointialgoritmi rakentaa luolaston liittämällä yhteen suunnittelijan valmiiksi tekemiä palasia. [Smith 2014] Palasten manuaalinen suunnittelu mahdollistaa sen, että suunnittelijoilla on erittäin hyvä kontrolli tuotettuun sisältöön. Suunnittelijat voivat esimerkiksi tuottaa valmiita huoneita sisältöineen, joita generointi algoritmi valitsee useita muodostaen kokonaisen luolaston. Tällöin pelaajille voidaan tarjota ihmisen suunnittelemia pulmia ja haasteita samalla hyödyntäen proseduraalista sisällöntuotantoa. Koska palaset on tällöin valmiiksi tuotettu ja generointialgoritmilla on valintaa tehtäessä aina rajallinen määrä vaihtoehtoja, täytyy tällöin palasia olla generointialgoritmin saatavilla riittävän useita tuomaan vaihtelua. Muutoin pelaaja saattaa alkaa havaita toistoa tuotetussa sisällössä, ja pelin uudelleenpelattavuusarvo kärsii. Tämän takia sisällön manuaalista tuottamista tarvitaan enemmän kuin useimmissa muissa menetelmissä.

Esimerkiksi Spelunky-pelissä täytetään kuvan 2 mukaisia neljä kertaa neljä kokoisia ruudukoita kuvan 3 mukaisilla valmiiksi suunnitelluilla huoneilla. Huoneet valitaan muuten satunnaisesti, paitsi että ne muodostavat aina reitin sisäänkäynnin ja uloskäynnin välille.



Kuva 2. Spelunky-pelin luolastorakenne koostuu ruudukosta huoneita [Make Games 2011].



Kuva 3. Spelunky-pelin yksittäinen, valmiiksi suunniteltu huone [Make Games 2011].

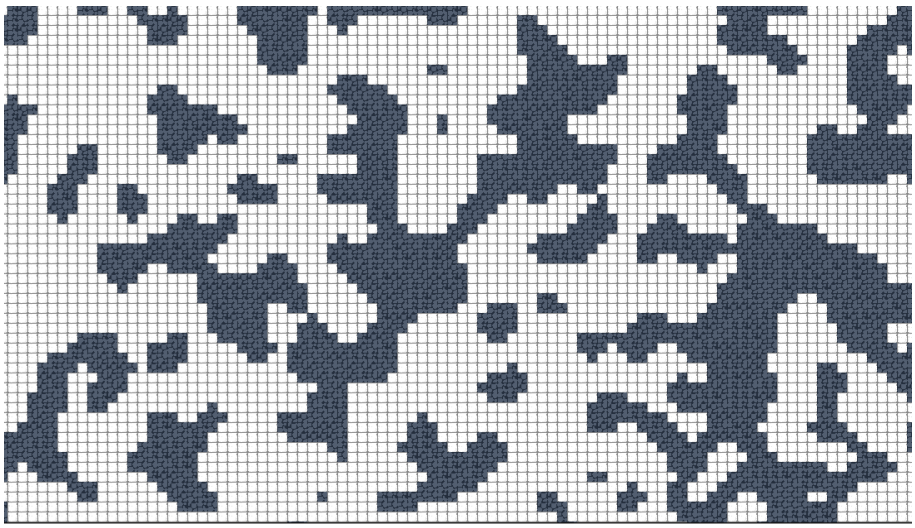
4.2. Soluautomaatit

Soluautomaatit (cellular automata) ovat diskreettejä laskentamalleja, joilla voidaan mallintaa itseorganisoituvia järjestelmiä. Soluautomaatti koostuu verkosta soluja, jonka jokaisella solulla on jokin tila äärellisestä määrästä tiloja. Yksinkertaisessa esimerkissä verkko voisi olla kaksiulotteinen taulukko, ja solulla voisi olla kaksi mahdollista tilaa: 0 ja 1. Jokainen solu on ajanhetkellä nolla ($t = 0$) aloitustilassa. Soluautomaatin tila kehittyy diskreetein askelin siten, että jokaisella askelella lasketaan uusi solujen sukupolvi ($t + 1$) jonkin säännön mukaan. Jokaiselle solulle on olemassa viereisten solujen joukko, naapurusto, joka on yleensä mukana solun tilan määräävässä säännössä. Aloitustila ja käytetty sääntö vaikuttavat merkittävästi siihen, mikä solujen tila on jollakin ajanhetkellä t.

Monille tuttu esimerkki soluautomaateista on John Conwayn määrittelemä Game of Life -peli. Se muodostuu kaksiulotteisesta ruudukosta soluja, jossa jokaisella solulla voi olla yksi kahdesta tilasta: elävä tai kuollut. Jokaista solua ympäröi kahdeksan naapurisolua, joiden perusteella lasketaan solun tila seuraavassa sukupolvessa.

Soluautomaatteja voidaan käyttää luolastojen topologian konstruktiiviseen generointiin. Johnson ja muut [2010] esittävät soluautomaatteihin perustuvan ratkaisun kaksiulotteisten luolastojen generointiin, jossa solut sijaitsevat kaksiulotteisessa ruudukossa, ja niillä voi olla kaksi tilaa: tyhjä ja kivi. Jokaista solua ympäröi kahdeksan naapurisolua, ja jokainen solu tietää naapurustossaan olevien kivisolujen lukumäärän. Pohjaruudukko alustetaan tyhjillä soluilla ja täytetään sitten sattumanvaraisesti kivisoluilla tiettyyn täyttymisasteeseen asti. Tämän jälkeen soluautomaatin tilaa iteroidaan n kierrosta eteenpäin ja jokaisella kierroksella jokaisen solun uusi tila lasketaan seuraavasti: solu on kiveä, jos sitä ympäröivien kivisolujen lukumäärä on suurempi tai yhtä suuri kuin joku etukäteen sovittu vakio, muussa tapauksessa solu on tyhjä. [Johnson *et al.* 2010]

Johnsonin ja muiden menetelmällä voidaan tuottaa kuvan 4 mukaisia luolastorakenteita, jotka muistuttavat luonnon muovaamia luolia. Menetelmä on myös hyvin nopea ja sen vuoksi sitä voidaan käyttää hyvin laajojenkin alueiden tuottamiseen ajon aikana. Menetelmän hallittavuus on heikko, sillä vaikka muodostuvien luolamaisten tilojen avoimuutta voidaan säädellä muuttamalla täyttöastetta, se ei sellaisenaan takaa useimmille peleille olennaista yhteyttä eri luolaston osien välille. Koska alustava täyttäminen tapahtuu satunnaisesti ja parametrien vaikutukset lopputulokseen voivat olla hyvinkin merkittäviä, on myös hyvin vaikeaa hallita syntyvien alueiden määrää ja kokoa.



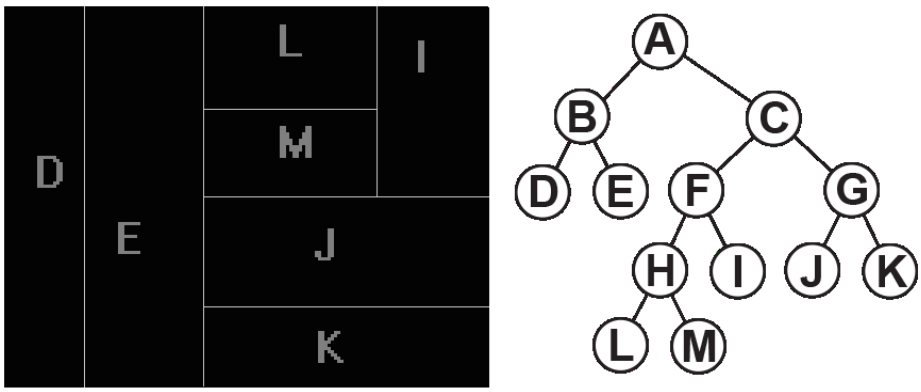
Kuva 4. Johnsonin ja muiden säännöillä generoidun luolaston osa. Valkoiset ruudut ovat tyhjiä ja siniset kiveä.

Yhteys kahden toisistaan erillisen alueen välille voidaan luoda soluautomaatin lopullisen tilan saavuttamisen jälkeen. Yksinkertaisimmillaan alueiden kahden toisiaan lähimmän solun välille kaivetaan ennalta määrätyn levyinen tunneli. [Johnson *et al.* 2010]

4.3. Avaruuden ositus

Avaruuden ositus (space partitioning) on menetelmä, joka jakaa avaruuden useaan osioon hypertason avulla siten, että jokainen avaruuden piste sijaitsee tarkalleen yhdessä tällaisessa osiossa. Avaruus jaetaan useimmiten rekursiivisesti aina vain pienempiin ja pienempiin osioihin, ja rekursion muodostaman hierarkian ansiosta avaruus voidaan näin esittää puurakenteena. Jokaisella sisäsolmulla on siis aina joukko lapsia, jotka ovat muodostuneet solmun jakamisesta. Avaruuden ositusta voidaan käyttää myös apuna luolaston topologian luomiseen konstruktiivisesti. [Shaker *et al*. 2016]

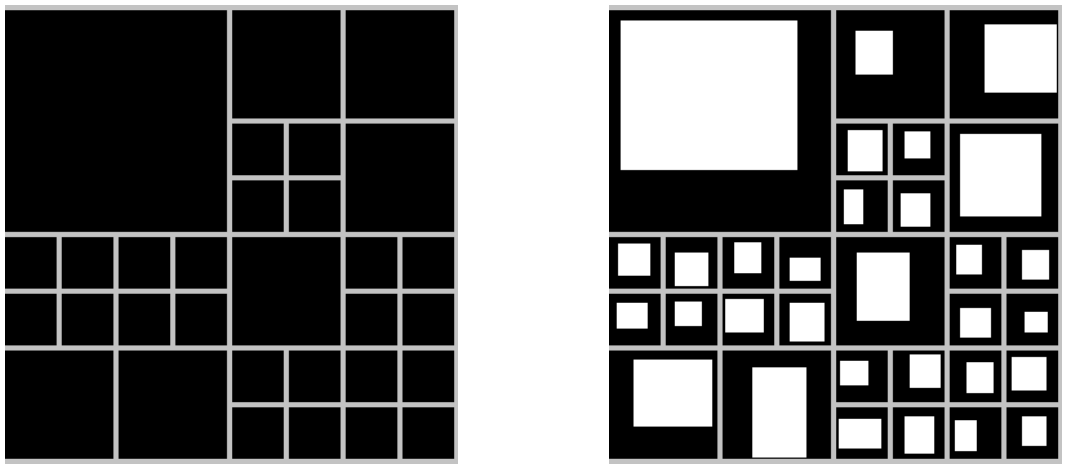
Binäärinen avaruuden ositus (binary space partitioning) on yleisin avaruuden ositukseen käytetty menetelmä. Se jakaa avaruuden rekursiivisesti aina kahteen osioon muodostaen näin binääripuun, jota kutsutaan BSP-puuksi. Kuvassa 5 näkyy binäärisesti ositettu avaruus ja sitä vastaava BSP-puu, jonka jokainen lehtisolmu vastaa yhtä osiota. Algoritmin erilaiset variantit käyttävät erilaisia sääntöjä avaruuden jakamiseen hypertason avulla. Quadtree-menetelmä esimerkiksi jakaa avaruuden aina neljään osioon muodostaen puun jonka jokaisella sisäsolmulla on aina neljä lasta. [Shaker *et al*. 2016]



Kuva 5. Vasemmalla binäärisesti ositettu avaruus ja oikealla sitä vastaava BSP-puu [Shaker *et al*. 2016].

Avaruuden ositusta voidaan käyttää luolastojen generoinnissa jakamaan luolasto sopivan kokoisiin osioihin, joista voidaan muodostaa luolaston huoneita. Puun juurisolmu vastaa koko luolaston aluetta, ja se jaetaan rekursiivisesti osioihin, kunnes jokin lopettamisehto toteutuu. Ehto voi olla esimerkiksi yksittäisen osion ennalta määrätyn minimikoon alittuminen tai osioiden maksimaalisen kokonaismäärän ylittyminen. [Shaker *et al*. 2016]

Kun puu on luotu jollakin ositusmenetelmällä, sen lehtiin eli jakamattomiin osioihin voidaan luoda huoneita [Shaker *et al*. 2016]. Kuvassa 6a näkyy quadtree-menetelmällä tasaisesti keskeltä jakamalla ositettu luolasto ja kuvassa 6b jokaiseen osioon on luotu yksi huone. Avaruuden ositusta käytettäessä huoneet eivät voi sijoittua päällekkäin, mikäli jokaiseen osaan luodaan maksimissaan vain yksi huone [Shaker *et al*. 2016]. Jos kaikkiin osiin luodaan huone, saadaan luolastolle aikaan kuvan 6b kaltainen suorakaiteen muotoinen rakenne, joka mukailee alkuperäisen juurisolmun alueen muotoa. Luolaston muotoon voidaan kuitenkin tuoda enemmän vaihtelua luomalla vain osaan lehtisolmuista huoneita. Myös muodostuvien osien kokoihin voidaan lisätä vaihtelua valitsemalla jokaisella kierroksella esimerkiksi satunnaisesti, missä suhteessa hypertaso jakaa alueen osiin. Näin muodostuvien alueiden mittasuhteet eivät ole enää yhtä suorasti sidoksissa juurisolmun mittasuhteisiin.



(a) (b)

Kuva 6. a) Avaruuden osiot b) Osioihin luodut huoneet [Shaker *et al*. 2016].

Avaruuden osituksella tuotettujen huoneiden yhdistämiseen voidaan käyttää mitä tahansa satunnaista tai sääntöpohjaista menetelmää. Shaker ja muut [2016] esittävät BSP-puun hierarkiaa hyödyntävän tavan käytävien muodostamiseen siten, että käytävä muodostetaan aina kahden sellaisen osion välille, jotka jakavat saman vanhemman. Näin todennäköisyys käytävien päällekkäisyyteen on pienempi kuin esimerkiksi jos käytävät muodostettaisiin täysin satunnaisesti. [Shaker *et al*. 2016]

4.4. Evoluutioalgoritmit

Evoluutioalgoritmit (evolutionary algorithms) ovat hakuperustaisia optimointialgoritmeja, jotka pohjautuvat evoluutioteoriaan ja luonnonvalintaan. Ne pe-

rustuvat itse *hakualgoritmiin* (search algorithm), *sisällön esitykseen* (content representation) ja *kelpoisuusfunktioon* (fitness function). Perusidea on se, että ominaisuuksiltaan olosuhteisiin hyvin sopivat yksilöt selviytyvät ja lisääntyvät, kun taas huonosti sopivat kuolevat pois. [Shaker *et al*. 2016] Ei ole takeita, että evoluutioalgoritmit lopulta tuottavat riittävän hyvän tuloksen, ja siksi ne eivät yleensä sovellu käytettäväksi ajonaikana [Togelius *et al.* 2011b].

Sisällön esitys kuvaa yhden tuotettavan esiintymän ominaisuudet eli kromosomin tai genotyypin, ja miten sen pohjalta voidaan tuottaa lopullinen esiintymä eli fenotyyppi. Sisällön esitykselle tärkeää on korkea *paikallisuus* (locality), mikä tarkoittaa sitä, että pienet muutokset kromosomiin aiheuttavat vain pieniä muutoksia fenotyyppiin ja kelpoisuusarvoon. Tärkeää on myös se, että valittu esitysmuoto pystyy esittämään kaikki kiinnostavat ratkaisut. [Togelius *et al.* 2011b]

Luolaston tapauksessa esitysmuoto voisi olla esimerkiksi suoraan ruudukko, jonka jokainen solu voisi edustaa esimerkiksi seinää, vapaata tilaa, ovea tai hirviötä. Vähemmän suorasti esitys voisi olla vaikkapa lista seinien sijainneista, suunnista ja pituuksista. Vielä epäsuoremmin sisällön voisi kuvata esimerkiksi listana halutuista ominaisuuksista, kuten huoneiden, ovien ja hirviöiden lukumäärät, reittien pituudet ja kuinka paljon luolasto haarautuu. Kaikista epäsuorin tapa olisi pelkkä satunnaislukusiemen, mutta sillä ei ole lainkaan paikallisuutta ja sen haku käyttäytyy kuin satunnaishaku. [Togelius *et al.* 2011b]

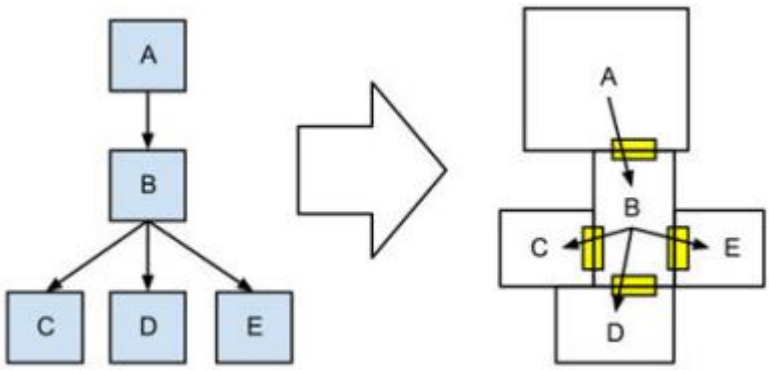
Kelpoisuusfunktio arvioi joko sisällön genotyyppiä tai sen perusteella tuotetun fenotyypin kelpoisuutta, eli kuinka sopiva se on peliin [Togelius *et al.* 2011b]. Se voi pyrkiä arvioimaan esimerkiksi luolaston läpäistävyyttä, kokoa ja pelikokemukseen vaikuttavia ominaisuuksia, kuten haastetasoa ja hauskuutta. Kelpoisuusfunktiot voidaan jakaa kolmeen avainluokkaan: *suoriin* (direct), *simulaatiopohjaisiin* (simulation-based) ja *interaktiivisiin* (interactive) kelpoisuusfunktioihin [Togelius *et al.* 2011b]. Shakerin ja muiden [2016] mukaan hyvän kelpoisuusfunktion kehittäminen on usein vaikein hakuperustaisen menetelmän kehittämisen osa. Pelikokemukseen vaikuttavia ominaisuuksia kuten pelaamisen hauskuutta, immersiota ja haastetasoa on hyvin vaikea arvioida algoritmisesti.

Evoluutioalgoritmi toimii siten, että se muodostaa ensin ensimmäisen sukupolven populaation esimerkiksi satunnaisesti, ja muodostaa sitten toistuvasti populaation kelpoisuusfunktion kannalta parhaista ehdokasratkaisuista seuraavan sukupolven, kunnes saadaan tuotettua riittävän hyvä ehdokasratkaisu tai ennalta määrätty iteraatioiden määrä tulee täyteen. Valintavaiheessa käydään läpi populaation jokaisen ehdokasratkaisun kelpoisuusfunktion antama kelpoisuus. Huonoimmat ehdokasratkaisut hylätään ja parhaiden annetaan

lisääntyä., jolloin päästään jatkuvasti kohti parempaa tulosta. Lisääntyminen eli uusien ehdokasratkaisujen tuottaminen tapahtuu yleensä joko mutaatiolla tai risteytyksellä tai molemmilla. Mutaatiossa jonkin aiemman ehdokasratkaisun kromosomia muutetaan pienissä määrin satunnaisesti, kun taas risteytyksessä kahden ehdokasratkaisun kromosomit sekoitetaan keskenään. [Shaker *et al*. 2016]

Valtchanov ja Brown [2012] esittävät geneettistä algoritmia käyttävän tavan generoida luolastoja proseduraalisesti. Luolaston rakennetta kuvaava kromosomi eli genotyyppi on kuvan 7 vasemmanpuoleisen kuvion mukainen puurakenne, jossa jokainen puun solmu vastaa huonepohjaa ja ovea, joka yhdistää solmun huoneen sen vanhempaan. Puurakenne takaa sen, että kaikki graafin solmut ovat yhteydessä toisiinsa ja irrallisia huoneita ei voi syntyä. [Valtchanov & Brown 2012]

Kromosomin esittämän yleiskuvauksen perusteella luodaan luolaston kuvan 7 oikeanpuoleisen kuvion mukainen fenotyyppi eli ilmentymä käymällä läpi jokainen puun solmu leveyshaulla aloittaen juurisolmusta. Jokaisen solmun huone yritetään asettaa kiinni vanhempansa ensimmäiseen vapaaseen oveen. Mikäli asetettava huone törmäisi johonkin jo aiemmin asetettuun huoneeseen tai asetettavan huoneen solmun vanhemmalla ei ole vapaita ovia, solmu ja kaikki sen lapset karsitaan genotyypistä. Myös liian kaukana keskustasta olevat huoneet ja niiden lapset karsitaan. Poistamalla epäsopivat solmut suoraan genotyypistä voidaan lisätä monimuotoisuutta välttämällä mutaatioiden syntymistä sellaisiin huoneisiin, jotka joka tapauksessa poistettaisiin varsinaisen luolaston luomisen yhteydessä. [Valtchanov & Brown 2012]



Kuva 7. Valtchanovin ja Brownin [2012] puuesitys muutetaan luolaston topologiaksi.

Valtchanov ja Brown [2012] käyttävät sekä mutaatiota että risteytystä uuden sukupolven luomiseen. Puurakenteen tapauksessa risteytystä on käytetty niin, että molemmat ehdokasratkaisut vaihtavat keskenään jonkin aligraafin. Mutaatioon on käytetty kolmea erilaista operaatiota: puun kasvattaminen, puun kar-

siminen ja puun solmujen muokkaaminen. Valinta tehdään jakamalla populaatio lohkoihin ja järjestämällä lohkojen yksilöt niiden kelpoisuuden mukaan ja korvaamalla sitten lohkon huonompi puolisko paremman puoliskon jälkeläisillä. [Valtchanov & Brown 2012]

5. Yhteenveto

Tuottamalla luolastoja proseduraalisesti voidaan säästää sisällöntuotannon kuluissa. Siten voidaan myös tuottaa lukumäärällisesti huomattavan paljon perinteistä enemmän erilaisia luolastoja, ja näin jokainen pelikerta voi olla erilainen. Generointialgoritmit ovat kuitenkin usein vaikeasti hallittavia ja siksi luolastojen proseduraaliseen generoimiseen ei ole olemassa yhtä menetelmää, joka olisi kaikissa tilanteissa paras, vaan sopiva menetelmä valitaan pelin erilaisten tarpeiden ja tavoitteiden mukaan. Useimmissa peleissä tärkeää on varmistaa erityisesti luolaston läpäistävyys, jottei pelaaja voi jäädä luolastoon jumiin. Tässä tutkielmassa esittelin neljä erilaista menetelmää luolastojen generointiin.

Sisällönvalinta on esittelemistäni menetelmistä yksinkertaisin, mutta siitä huolimatta hyviä tuloksia tuottava menetelmä, jonka suurin heikkous on sen suuri tarve sisällön manuaaliseen tuottamiseen. Koska kenttäsuunnittelijat tuottavat valmiit palaset, saadaan peliin helposti ihmisten suunnittelemia pulmia ja haasteita.

Soluautomaatit soveltuvat hyvin esimerkiksi luonnon muovaamien luolastojen tuottamiseen, mutta niiden tuloksia on melko vaikea hallita. Erilaisia sääntöjä voidaan kehittää loputtomiin erilaisiin tarkoituksiin, ja soluautomaatin alkutila määrää suoraan lopullisen tuloksen. Menetelmä itsessään ei takaa huoneiden yhtenäisyyttä, vaan huoneiden väliset käytävät täytyy tuottaa erikseen tai alkutila täytyy määrittää sellaiseksi, että yhteydet syntyvät varmasti.

Avaruuden ositusta käyttämällä voidaan tuottaa huonepohjaisia luolastoja, joissa huoneet eivät voi mennä päällekkäin. Binääristä avaruuden ositusta käytettäessä huoneiden väliset käytävät voidaan tuottaa hyödyntämällä BSP-puun hierarkiaa.

Evoluutioalgoritmeilla pyritään tuottamaan mahdollisimman hyvä luolasto luonnonvalintaa mukaillen luomalla ensin joukko täysin satunnaisia luolastoja ja antamalla niistä parhaiden lisääntyä, kunnes saadaan aikaan yksi riittävän hyvä luolasto. Evoluutioalgoritmit tarvitsevat kuitenkin toimiakseen kelpoisuusfunktion, jonka kehittäminen voi olla haastavaa, koska luolastojen vaikeusastetta ja hauskuutta on hyvin vaikea todeta laskennallisesti. Hyvä kelpoisuusfunktio mahdollistaa kuitenkin hyvän hallittavuuden.

Viiteluettelo

Mark Hendrikx, Sebastiaan Meijer, Joeri van der Velden and Alexandru Iosup. 2013. Procedural content generation for games. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 9, 1-22.

Lawrence Johnson, Georgios N. Yannakakis and Julian Togelius. 2010. Cellular automata for real-time generation of infinite cave levels. In: *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, ACM, 3.

Roland van der Linden, Ricardo Lopes and Rafael Bidarra. 2014. Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(1), 78-89.

Make Games. 2011. The Full Spelunky on Spelunky. http://makegames.tumblr.com/post/4061040007/the-full-spelunky-on-spelunky. Checked 9.5.2017.

Noor Shaker, Julian Togelius and Mark J. Nelson. 2016. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer.

Gillian Smith. 2014. Understanding procedural content generation: a design-centric analysis of the role of PCG in games. In: *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, ACM, 917-926.

Julian Togelius, Emil Kastbjerg, David Schedl and Georgios N. Yannakakis. 2011. What is procedural content generation?: Mario on the borderline. In: *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, ACM, 3.

Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley and Cameron Browne. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 172-186.

Valtchanov Valtchan and Joseph Alexander Brown. 2012. Evolving dungeon crawler levels with relative placement. In: *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering*, ACM.

Miksi pelaajat ostavat virtuaalista sisältöä videopeleissä?

Markus Järvinen

Tiivistelmä
Virtuaalisen sisällön myymisestä ja ostamisesta on tullut iso osa pelialan liikevaihtoa. Virtuaalisen sisällön tarjoaminen on yleistynyt melkein kaikissa peligenreissä, mutta se näkyy erityisesti verkkopeleissä, ilmaispeleissä sekä virtuaalimaailmoissa. Yleisimmät ostomotiivit virtuaaliselle sisällölle ovat esteetön pelaaminen, sosiaalinen vuorovaikutus, kilpailullisuus, taloudelliset syyt ja lisäsisällön avaaminen pelaajalle itselleen ja lapsilleen. Ostohalukkuuteen vaikuttaa lisäksi kauppapaikka, jossa ostotapahtuma suoritetaan. Siinä keskeistä on, kuinka luotettavaksi palvelu koetaan ja kuinka helpoksi ostoprosessi on tehty. Koska videopeleissä myytävästä virtuaalisesta sisällöstä on tullut iso osa peliteollisuuden liikevaihtoa, on pelintekijöiden tärkeä ymmärtää ja tutkia pelaajien ostomotiiveja. Näin he voivat tehdä entistä parempia pelejä ja pelimekaniikkoja sekä lisätä virtuaalisen sisällön myyntiä.

Avainsanat ja -sanonnat: Virtuaalinen sisältö, ostomotiivit, ilmaispeli, virtuaalimaailma

1. Johdanto

Videopelit ovat kasvaneet merkittäväksi osaksi viihdeteollisuutta. Peliteollisuuden myyntitulot olivat vuonna 2016 yli 91 miljardia [SuperData, 2016]. Peleihin liittyvän virtuaalisen sisällön ja pelien lisäsisällön ostaminen ovat nykyään yksi suurimmista internetin välityksellä tapahtuvista kulutuksen muodoista [Hamari *et al.*, 2017]. Internetin, sosiaalisen median ja mobiililaitteiden yleistyminen on kasvattanut pelialaa merkittävästi.

Ensimmäiset virtuaalisen sisällön ostot tehtiin vuonna 1999 pelaajalta toiselle suoritetussa vaihdossa. Pelaajat laittoivat heidän saamaansa virtuaalimaailman sisältöä myyntiin eBay-huutokauppapalveluun ja muut pelaajat pystyivät tekemään niistä tarjouksia oikealla rahalla [Lehdonvirta, 2009].

Pelintekijät myyvät virtuaalista sisältöä suoraan pelaajalle, mutta pelaajalta pelaajalle tapahtuvia kauppoja tehdään myös paljon. Lisäksi virtuaalista sisältöä myydään erilaisissa palveluissa. Esimerkiksi IRC-Galleria-kuvapalvelussa on mahdollista ostaa kuvapalveluun liittyvää virtuaalista sisältöä [Lehdonvirta, 2009]. Tässä tutkielmassa kuitenkin keskitytään enemmän peleissä tarjottavan virtuaalisen sisällön ostomotiiveihin.

Virtuaalisen sisällön tarjoaminen ja kauppaaminen on yleistynyt internetissä viime vuosina, mutta se on jo pitkään näkynyt erityisesti ilmaispeleissä. Ai-

kaisemmin pelinsisäiset ostot olivat harvassa ja kun peli julkaistiin, siihen ei välttämättä tullut enää uutta sisältöä, vaan se oli valmis kokonaisuus. Internetin yleistymisen myötä pelintekijöillä on nyt mahdollisuus päivittää peli jälkikäteen, lisätä uutta sisältöä ja korjata pelin pelimekaniikkoja. Siksi nykyään jo pelejä suunniteltaessa mietitään, millä eri tavoilla saadaan pelaajat tekemään mahdollisimman paljon pelin sisäisiä ostoja. Ne muodostavat yhä suuremman osan pelin kokonaisliikevaihdosta [Hamari *et al.*, 2017].

Pelit pyritään suunnittelemaan niin, että pelaaja kokee tarpeelliseksi tai suorastaan välttämättömäksi hankkia jatkuvasti uutta virtuaalista sisältöä peliin. Pelintekijät myös säännöllisesti julkaisevat ja esittelevät uutta sisältöä peleihin, mikä houkuttaa pelaajia ostamaan niitä. Tämän takia voi todeta, että pelien sisäisissä ostoissa ei ole enää kyse vain pelaajan omasta halusta ostaa uutta pelisisältöä, vaan myös itse peliin rakennetuista tarpeista tehdä maksullisia lisäostoksia [Hamari *et al.*, 2017].

Tässä tutkielmassa on tarkoitus selvittää, miksi pelaajat tekevät pelin sisäisiä ostoja. Millä perusteilla pelaajat käyttävät suuriakin rahasummia virtuaalisen sisällön hankintaan eli mitkä ovat heidän ostomotivaationsa? Lisäksi tarkastellaan, mitkä muut tekijät vaikuttavat erilaisen virtuaalisen sisällön hankintaan.

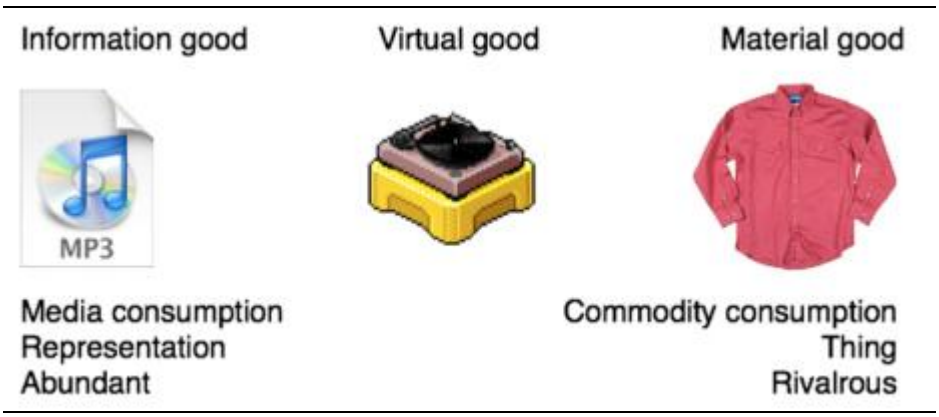
Luvussa 2 esitellään, mitä virtuaalinen sisältö on ja millaisia ominaisuuksia sillä on. Sen jälkeen luvussa 3 pohditaan, mitkä tekijät vaikuttavat virtuaalisen sisällön arvoon. Luku 4 käsittelee virtuaalisen sisällön ostomotiiveja. Tämän jälkeen pohditaan ostopaikkojen merkitystä virtuaalisen sisällön hankinnassa, sekä ostoprosessiin liittyviä huolenaiheita. Lopuksi vielä käydään läpi johtopäätökset tutkielman pääkohdista.

2. Virtuaalinen sisältö

Ennen kuin paneudutaan syvemmin pelaajan motivaatioon ostaa sisältöä peleihin, on tärkeää määritellä, mitä virtuaalinen sisältö (virtual goods) on. Virtuaalinen sisältö on digitaalista sisältöä, joka esiintyy pelissä tai virtuaalisessa ympäristössä, johon se on luotu [Lehdonvirta, 2009]. Tähän ei lasketa konkreettisia tuotteita tai palveluita, joita erinäiset pelit ja sivustot tarjoavat, kuten esimerkiksi vaatteita tai fanituotteita. Virtuaalista sisältöä on esimerkiksi peleissä esiintyvät pelihahmot, aseet, huonekalut, tehosteet, valuutat, ajoneuvot ja asusteet. Pelaaja voi ostaa virtuaalista sisältöä oikeaa rahaa tai pelin sisäistä valuuttaa käyttämällä joko pelin käyttöliittymässä tai jonkin kolmannen osapuolen palvelun kautta.

Lehdonvirran [2009] mukaan virtuaalisen sisällön ero muuhun digitaaliseen sisältöön on sen kilpailullisuus (rivalrous). Virtuaalimaailmassa oleva pai-

ta voi olla yhdellä hahmolla, mutta jos se siirretään toiselle hahmolle, sen alkuperäinen omistaja ei voi käyttää sitä enää. Tämä eroaa perinteisestä digitaalisesta sisällöstä siten, että esimerkiksi internetistä ladattu ääni- tai kuvatiedosto voi olla samanaikaisesti usealla henkilöllä, jos sisällön alkuperäinen omistaja on sen päättänyt jakaa eteenpäin. Erilaisia tuotteita on vertailtu kuvassa 1.



Kuva 1. Tuotteiden erot [Lehdonvirta, 2009]

Monet virtuaaliset tuotteet ovat inspiroituja tosielämän tuotteista ja tavaroista, mutta virtuaalisella sisällöllä ei aina ole olemassa tosielämän vastakappaletta. Virtuaalimaailmassa vaatteet voivat kuvastaa reaalimaailman vaatteita, mutta niillä voi olla myös muita ominaisuuksia. [Lehdonvirta, 2009].

2.1 Ei-virtuaalisten tuotteiden ostoperusteet

Yksi tapa ymmärtää kuluttajien käytöstä virtuaalisten tuotteiden suhteen on tarkastella, miten he toimivat perinteisen, ei-virtuaalisten tuotteiden osalta. Tällöin sosiologisesta perspektiivistä avautuu kolme näkökulmaa, kuinka tavaroita yleensä ostetaan. Nämä ovat käytännöllinen näkökulma, tunnepohjainen näkökulma ja tavaroiden hyödyntäminen sosiaalisina huomionosoituksina [Lehdonvirta, 2009].

Käytännöllisestä näkökulmassa katsottuna tuotteet palvelevat täyttämään ihmisten perustarpeet ja ne ovat yleensä konkreettisia tuotteita. Markkinoinnin termein voidaan sanoa, että niillä on toimintaan ja suorituskykyyn liittyviä ominaisuuksia [Lehdonvirta, 2009].

Tunnepohjaisesta kulmasta tarkasteltuna tuotteet ja niiden kulutus voidaan nähdä hedonistisena prosessina, jossa etsitään nautintoa ja mielihyvää. Tällöin tuotteiden visuaaliset, esteettiset ja nautinnolliset ominaisuudet korostuvat. Tuotesuunnittelussa muoto, maku, väri ja tyyli ovat tärkeitä tekijöitä [Lehdonvirta, 2009].

Kolmanneksi kuluttajat nähdään myös tiedottajina, jotka ilmaisevat käyttämillään tuotteilla statusta, luokkaa, ryhmään kuulumista, erilaisuutta tai

omaa identiteettiään. Esimerkiksi ylemmän sosiaaliluokan henkilö voi yrittää erottaa itsensä alemman luokan edustajista statusesineillä. Toinen esimerkki voisi olla keräilyesineiden hankkiminen. Ne ovat usein turhia ja epäkäytännöllisiä, mutta tarpeeksi harvinaisia ja erityisiä, jotta kuluttaja niitä tavoittelee. Brändituotteiden käyttäminen on myös yksi tapa erottua sosiaalisesti ja ilmaista omaa identiteettiään. [Lehdonvirta, 2009].

Seuraavaksi pohdin, voiko pelien virtuaalisen sisällön ostoperusteita tarkastella näistä kolmesta eri näkökulmasta. Ovatko ne tältä osin samankaltaisia kuin perinteiset ei-virtuaaliset tuotteet?

2.2 Virtuaalisen sisällön toiminnalliset ominaisuudet

Pelit tarjoavat nykyään hyvin laajan valikoiman erilaista virtuaalista sisältöä, jota pelaaja voi ostaa oikealla rahalla tai pelin sisäistä valuuttaa käyttämällä. Virtuaalinen sisältö voi olla monenlaista riippuen siitä, onko kyseessä esimerkiksi mobiilipeli, ilmaispeli tai jokin massiivinen virtuaalimaailma. Ne voivat olla asioita, joilla pelaaja voi nopeuttaa edistymistään pelissä, hankkia lisäsisältöä peliin, ostaa kosmeettisia tavaroita pelihahmoille tai asioita, jotka tekevät pelaajasta tai pelihahmosta jollain tapaa paremman. Näyttää siis siltä, että myös videopelien virtuaalituotteet voidaan jaotella ominaisuuksiensa pohjalta vastaaviin kolmeen luokkaa kuten perinteisetkin kulutustavarat: toiminnallisiin, esteettisiin ja sosiaalisiin ominaisuuksiin [Lehdonvirta, 2009]. Seuraavana tarkastellaan yksityiskohtaisemmin pelien maksullisia virtuaalituotteita tämän kolmijaon pohjalta.

Kun virtuaalinen sisältö vaikuttaa pelin etenemiseen, pelihahmon taitotasoon tai pelin tapahtumiin, voidaan ajatella, että sillä on toiminnallisia ominaisuuksia [Lehdonvirta, 2009]. Osassa peleistä on mahdollisuus tehostaa (boost) pelaajan etenemistä. Pelaaja voi esimerkiksi rahalla nopeuttaa pelin edistymistä, parantaa hahmoansa, nopeuttaa ajastimia tai avata sisältöä, jonka saisi muuten vain pelaamalla. Toiminnallisia ominaisuuksia sisältävät myös valuutat ja varusteet, jotka vahvistavat pelihahmoa.

Massiivisissa monen pelaajan verkkoroolipeleissä (Massively Multiplayer Online Role-playing Game, MMORPG), kuten Ultima Online ja World of Warcraft, suurin syy oikealla rahalla ostamiseen on pelaajan suorituskyvyn parantaminen pelissä. Vahvat hahmot ovat arvokkaampia kuin heikot hahmot, paljon vahinkoa tekevät miekat ovat arvokkaampia kuin tehottomat miekat ja nopeat ratsut ovat halutumpia kuin hitaat ratsut ja niin edelleen [Lehdonvirta, 2009].

Toiminnallista virtuaalisisältöä tarjotaan pelaajille erityisesti yksinpeleissä ja ilmaispeleissä. Niissä ostettavan sisällön avulla voi helpottaa pelissä etene-

mistä [Lehdonvirta, 2009]. Ilmaispeleille on tyypillistä, että pelaajan edettyä tiettyyn vaiheeseen pelissä, hän ei voi jatkaa etenemistään maksamatta pelintekijän ennalta määrittämää summaa. Näitä pelejä kutsutaan freerium-peleiksi.

Monen pelaajan pelattavissa verkkopeleissä toiminnallista sisältöä tarjotaan vähemmän. Se antaisi liian suuren edun pelaajille, jotka ovat valmiita käyttämään paljon rahaa peliin. Tämä johtaisi pelin kilpailullisen idean katoamiseen. Näitäkin pelejä on kuitenkin olemassa: niitä kutsutaan pelikulttuurissa maksa voittaaksesi (pay-to-win) -peleiksi [Lehdonvirta, 2009].

2.3 Virtuaalisen sisällön tunneperäiset ominaisuudet

Virtuaalisen sisällön tunneperäiset ominaisuudet ovat yleensä asioita, jotka eivät muuta peliä muulla kuin visuaalisella tavalla [Lehdonvirta, 2009]. Niillä voi tuottaa esteettistä mielihyvää, kun pelaaja voi muokata esimerkiksi pelaajan hahmoa: sen varusteita, aseita, värejä, hiustyylejä tai kasvonpiirteitä. Nämä eivät tee pelaajasta tai pelihahmosta parempia, vaan ne muuttavat vain visuaalisia asioita pelin sisällä ja näin vahvistavat pelaajan erilaista identiteettiä näkyvästi.

Esteettistä sisältöä ilmenee hyvin monessa pelissä, mutta erityisesti virtuaalimaailmoissa ja verkkopeleissä. Näissä peleissä pelihahmon ulkonäköä ja esineiden kosmeettista ilmettä pidetään erityisen tärkeänä, sillä kaikki pelaajat voivat nähdä muiden pelihahmot ja esineet. Esimerkiksi World of Warcraftissä pelaajat voivat ostaa aseisiinsa lumouksia, jotka parantavat pelaajan aseen tehoa ja joiden voidaan ajatella sisältävän toiminnallisen ominaisuuden. Yksi suosituimmista on ’minor beastslayer’ -lumous. Sen tarjoama lisäteho aseeseen on hyvin heikko, mutta se antaa hienon punaisen hehkun pelaajan aseelle ja tästä johtuen on yksi suosituimmista lumouksista, jota pelaajat pelin sisällä kauppaavat toisilleen [Lehdonvirta, 2009].

Pelaaja saattaa perustella esineen hankinnan sen käytännöllisillä ominaisuuksilla, mutta todelliset syyt saattavat olla tunneperäiset tai sosiaaliset. Tätä voidaan verrata esimerkiksi tehokkaiden elitististen urheiluautojen ostamiseen. [Lehdonvirta, 2009]. Niitä usein sanotaan ostettavan järkiperustein, vaikka ostoperuste saattaa olla hedonistinen tai statuksellinen.

2.4 Virtuaalisen sisällön sosiaaliset ominaisuudet

Peleihin ostettavan virtuaalisen sisällön sosiaaliset ominaisuudet vaikuttavat myös samankaltaisilta kuin perinteisissä kulutustuotteissa. Pelaaja voi osoittaa kuuluvansa johonkin ryhmään pelissä, kerätä harvinaisia keräilyesineitä tai omistaa pelin historiaan tai pelin ulkopuoliseen kulttuuriin liittyvää sisältöä. Ultimate Online-verkkopelissä pelintekijät olivat luoneet virtuaalimaailmaan

hevosen jätöksiä, joita löytyi pelin hevostalleilta. Pelin hevoset eivät kuitenkaan tuottaneet jätöstä lisää, joten ajan kuluessa, kun pelaajat olivat keränneet kaikki jätökset, niiden suosio kasvoi ja niistä tuli haluttua keräilytavaraa. Vaikka hevosten jätöksistä ei ollut pelaajalle mitään toiminnallista hyötyä, niitä kaupattiin useilla sadoilla dollareilla [Lehdonvirta, 2009].

Toisinaan virtuaalinen sisältö saattaa ulottua pelin ulkopuoliseen kulttuuriin. Vuoden 2006 jalkapallon maailmanmestaruuskisojen aikaan kuvapalvelu IRC-galleria myi kilpailuihin liittyvää virtuaalista sisältöä. Kyseisten tuotteiden ostot ylittivät kaiken muun virtuaalisisällön myynnin palvelussa. Monessa verkkopelissä on myös kausittaista sisältöä, kuten joulu- tai halloween-aiheisia tuotteita [Lehdonvirta, 2009].

3. Virtuaalisen sisällön hinnan ja arvon määrittely

Konkreettisten asioiden hintoihin vaikuttavat monenlaiset tekijät, kuten tuotantokulut, logistiikka, verot, tarjonta, kysyntä, markkinointi ja maailman talous. Virtuaalisen sisällön hinta ja arvo eivät ole verrattavissa fyysisiin esineisiin ja asioihin. Tämä johtuu siitä, että virtuaaliset tuotteet ovat käytännössä pikseleitä ja ohjelmointikieltä, eivätkä ne useimmiten kulu käytössä kuten oikean elämän tavarat. Kun jokin asia on ohjelmoitu, se ei voi loppua kesken. Siitä huolimatta, pelien virtuaalisella sisällöllä on hinta ja sen arvoa voidaan mitata. Arvon määrittely on kuitenkin hankalaa.

Virtuaalisen sisällön arvo määrittyy sen perusteella, kuinka hyvin se palvelee käyttäjää verrattuna pelin muihin vastaaviin tuotteisiin sekä ympäröivään peliympäristöön. Virtuaalisen sisällön myyntiarvoa voi analysoida aiemmin mainitun reaalimaailman tai virtuaalimaailman tuotteen kolmen ominaisuuden pohjalta. Käytetään esimerkkinä jonkin pelin miekkaa. Ensinnäkin käytännöllisyyden näkökulmasta miekka voi olla hyvin tehokas. Jotta voidaan ajatella, että miekka on tehokas, täytyy pelissä olla olemassa vähemmän tehokas miekka, johon sitä voidaan verrata [Lehdonvirta, 2009].

Toisekseen tunteiden ja esteettisten ominaisuuksien näkökulmasta miekka voi olla erityisen tyylikkäästi muotoiltu verrattuna pelin muihin miekkoihin. Se tietysti nostaa miekan haluttavuutta ja sitä kautta sen arvoa. Kolmanneksi sosiaalisesta näkökulmasta miekka voi olla harvinainen tai alkuperältään poikkeava keräilykappale verrattuna muihin miekkoihin. Sen omistamalla saa muiden pelaajien arvostusta [Lehdonvirta, 2009].

Virtuaalisen sisällön alkuperäisen hinnan määrittää pelintekijä tai toinen pelaaja. Tämän jälkeen pelaaja itse arvioi, onko hän valmis maksamaan määritetyn hinnan kyseisestä sisällöstä. Toisin sanoen virtuaalisen sisällön lopullinen arvo riippuu siitä, kokeeko pelaaja virtuaalisen sisällön hinnan arvoiseksi.

Peleissä, joissa pelaajat myyvät toisilleen virtuaalista sisältöä, pelaajat luovat itse virtuaaliselle sisällölle arvon ja saattavat rakentaa näin pelin ympärillä toimivat myyntimarkkinat. Yksi esimerkki tällaisesta pelistä on Counter-Strike: Global Offensive. Kyseisessä pelissä pelaajat voivat avata laatikoita, joista he saavat erilaisia aseita. Aseissa on eri harvinaisuusluokan maalipintoja, jotka voivat olla upouusia tai kuluneita. Aseissa voi olla myös laskuri joka näyttää, kuinka monta vihollista pelaaja on ampunut kyseisellä aseella. Näiden lisäksi aseen arvoon vaikuttaa, kuinka yleisesti kyseistä asetta pelin sisällä käytetään. Mitä harvinaisemman esineen pelaaja omistaa, sitä korkeampi sen myyntihinta yleisesti on.

Virtuaalisen sisällön markkina-arvoon vaikuttavat monet sosiaaliset ominaisuudet. Näitä ovat mm. esineen ikä, millaisissa tilanteissa esinettä on käytetty ja aikaisemmat omistajat, kuten julkisuuden henkilöt [Lehdonvirta, 2009]. Vuonna 2014 pohjoisamerikkalainen Counter-Strike-joukkue iBUYPOWER hävisi pelin tahallaan saadakseen rahallista hyötyä. Kun vuonna 2015 asia tuli julkisuuteen, neljä joukkueen jäsentä saivat pelikiellon Valven järjestämiin turnauksiin ja joukkue hajosi pian tämän jälkeen. Tästä seurasi, että pelissä myytävät iBUYPOWER-virtuaaliset tarrat muuttuivat hyvin arvokkaiksi. Aikaisemmin ne maksoivat muutaman kympin, mutta skandaalin jälkeen niiden arvo nousi tuhansiin dollareihin. [Dot Esports, 2015].

4. Miksi pelaajat ostavat pelien tarjoamaa sisältöä?

Erilaisia pelejä on markkinoilla todella paljon ja ne tarjoavat hyvin laajan skaalan virtuaalista sisältöä, jota pelaajat voivat itselleen hankkia. Tämän lisäksi on olemassa erilaisia käyttöliittymiä ja palveluja, joiden avulla pelaaja voi ostaa virtuaalista sisältöä. Pelaajat ovat eri ikäluokista, kulttuureista ja heidän taloudelliset tilanteensa vaihtelevat suuresti. Tästä johtuen erilaisia ostomotiiveja on hyvin paljon ja lopullinen ostopäätös on monen tekijän summa.

Tässä luvussa käydään läpi yleisimpiä syitä, jotka vaikuttavat virtuaalisen sisällön ostamiseen. Tarkastelussa nojataan kahteen empiiriseen tutkimukseen, joista toinen on keskittynyt pelaajien ostomotiiveihin ilmaispeleissä ja toinen käsittelee virtuaalimaailmoissa tapahtuvaa virtuaalisen sisällön myymistä ja ostamista.

Hamari ja muut [2017] tutkivat pelaajien motiiveja ostaa virtuaalista sisältöä ilmaispeleissä. Dataa kerättiin internetissä tapahtuvan kyselyn avulla. Kysely tapahtui erilaisten sosiaalisten medioiden välityksellä ja se toteutettiin yhteistyössä kolmen suuren pelilehden kanssa. Dataa kerättiin 519 vastauksesta. Vastaajista sukupuoleltaan yli 91% oli miehiä ja iältään lähes 95% oli alle 40-

vuotiaita. Eniten oli edustettuna 20-29-vuotiaat, mikä johtuu todennäköisesti siitä, että suurin osa näiden pelilehtien lukijoista kuului tähän ikähaarukkaan.

Barnes ja Guo [2009] käyttivät neljää fokusryhmää tutkiessaan virtuaalisen sisällön myymistä ja ostamista virtuaalimaailmoissa. Osallistujia oli yhteensä 24 ja kaikki olivat kotoisin Kiinasta. Heistä 20 oli miehiä ja 4 naisia. Iän keskiarvo oli 21 vuotta. Jokainen osallistuja pelasi peliä, joka oli luokiteltu Kiinan kymmenen suosituimman massiivisen monen pelaajan verkkoroolipelin joukkoon. Osallistujien käyttämä peliaika virtuaalimaailmoissa vaihteli kuudesta kuukaudesta kahteen vuoteen. Osallistujista noin 33.3% oli työssä käyviä koulusta valmistuneita henkilöitä ja loput noin 67% oli yliopisto-opiskelijoita.

4.1 Sosiaaliset syyt

Edellä mainitut tutkimukset viittaavat siihen, että yksi suurimmista tekijöistä virtuaalisen sisällön ostamisessa on sosiaaliset motiivit. Kun pelaaja ostaa virtuaalista sisältöä, hänen motiivinsa liittyy jollain tapaa muihin ihmisiin ja heidän väliseen vuorovaikutukseen.

Hamari ja muut [2017] katsovat, että ylipäätään pelaamisessa ja kaikessa siihen liittyvässä keskeistä on kavereiden kanssa pelaaminen, personointi, lahjojen antaminen sekä erikoistapahtumiin osallistuminen. Koska verkkopeleissä ollaan vuorovaikutuksessa muiden pelaajien kanssa, Hamari ja muut [2017] ovat sitä mieltä, että sosiaaliset motiivit ovat yksi yleisimmistä syistä ostaa virtuaalista sisältöä verkkopeleissä.

Barnes ja Guo [2009] huomasivat myös, että tutkimukseen osallistujat kokivat sosiaaliset syyt tärkeänä. Sosiaaliset syyt ovat vahva kannustin virtuaalisen sisällön ostamiselle, ja se näkyy monella tavalla. Usein ystävät tai pelikaverit suosittelivat erilaisia ostoja, jotka vaikuttivat lopulliseen ostopäätökseen. Ennen kuin pelaaja suorittaa ostoja, hän saattaa konsultoida ystäviään, minkä avulla hän saa tarvittavaa tietoa sekä hyväksyntää ostokselleen. Pelaaja voi myös ostaa tehostimia pysyäkseen ystäviensä tasolla tai halutessaan suoriutua hyvin pelissä, jossa hän toimii osana isompaa ryhmää. Muiden auttaminen lahjoja lähettämällä on myös hyvin tyypillinen tapa erityisesti verkossa pelattavissa peleissä [Hamari *et al.*, 2017].

Oman pelihahmon personointi on yksi yleisimmistä verkkopelien pelimekaniikoista. Peleissä, joissa on mahdollista nähdä muut pelaajat, pelaajat kokevat personoinnin tärkeäksi ja ovat valmiimpia maksamaan virtuaalisisällöstä [Hamari *et al.*, 2017]. Nykyään suurin osa verkkopeleistä mahdollistaa personoinnin, sillä se ei useimmiten muuta peliä muulla kuin visuaalisella tavalla, eikä näin anna etua pelaajille, jotka käyttävät peliin oikeaa rahaa.

4.2 Esteetön pelaaminen

Esteettömällä pelaamisella tarkoitetaan tilannetta, jossa pelaaja tekee pelin sisäisiä ostoja, jotta hän voisi jatkaa pelaamista ilman keskeytyksiä tai odotteluaikoja. Tähän sisältyy ajastimien nopeutus, toiston välttäminen, lopun saavuttaminen, pelin jatkaminen ja saavutuksien suojeleminen [Hamari *et al.,* 2017].

Ilmaispeleissä on tyypillistä, että pelintekijä luo keinotekoisia rajoitteita peliin, jonka myötä pelaajalle luodaan tarve ja halu ostaa virtuaalista sisältöä. Tämän takia sisällön ostaminen on myös kohdannut vastustusta pelaajilta ja pelintekijöiltä, koska keinotekoiset rajoitteet antavat kuvan, että peli on vain rahantekoa varten tehty, eikä välttämättä aina palvele pelaajia parhaalla mahdollisella tavalla [Hamari *et al.*, 2017]. Jotta pelaaja suostuu maksamaan virtuaalisesta sisällöstä, hänen täytyy kokea peli miellyttävänä. Kun pelaaja nauttii pelistä, hän todennäköisemmin ostaa virtuaalista sisältöä, joka helpottaa hänen etenemistään, kun hän kohtaa haasteellisia tilanteita pelissä [Hamari *et al.*, 2017]. Barnesin ja Guon [2009] tutkimuksessa esteetön pelaaminen ei noussut merkittäväksi virtuaalisen sisällön ostomotiiviksi. Tämä johtunee siitä, että suurin osa peleistä, joissa pelataan virtuaalimaailmoissa, ei sisällä samanlaisia pelaamisen estävää pelisuunnittelua.

Esimerkkeinä keinotekoisista rajoitteista ovat pelit kuten Clash of Clans, HayDay ja Boom Beach. Pelaajan on tarkoitus kasvattaa omaa vahvuuttaan erilaisten rakennusten ja resurssien avulla. Näissä peleissä eteneminen on nopeaa aluksi, mutta kun pelaaja on päässyt tarpeeksi pitkälle, alkaa edistyminen hidastua. Tämä näkyy esimerkiksi hyvin pitkissä rakennusajoissa, joita voi nopeuttaa tai ohittaa käyttämällä pelin sisäistä valuuttaa. Pelin sisäistä valuuttaa saa kuitenkin hyvin hitaasti pelaamalla, jolloin nopein ja helpoin tapa on ostaa sitä oikealla rahalla. [Supercell, 2017].

4.3 Kilpailulliset tekijät

Kilpailullisilla tekijöillä tarkoitetaan sitä, että pelaajan ostomotivaatiot liittyvät kilpailullisiin näkökohtiin, kuten parhaaksi tuleminen, saavutuksien esittely ja kavereille kehuskeleminen [Hamari *et al.,* 2017]. Hamarin ja muiden [2017] tutkimuksessa kilpailulliset tekijät eivät nousseet kovin tärkeiksi ilmaispelien kohdalla. Tämä tarkoittaa, että pelaajat eivät pidä maksa-voittaaksesi -pelejä hyvinä. Ne koetaan epäreiluiksi, koska pelaajat voivat pelissä pärjäämisen sijaan maksaa tiensä huipulle.

Barnesin ja Guon [2009] tutkimuksessa osallistujat pitivät tärkeänä, että heidän hahmonsa eivät olleet liian huonoja muiden pelaajien hahmoihin verrattuna. Erityisesti peleissä, joissa pelaajat saavuttivat vahvan immersion tai flow-tilan ja peli tarjosi sopivasti haastetta pelaajalle, erilaisten hahmojen, esineiden

ja tasojen ostaminen oikealla rahalla tuotti mielihyvää ja se koettiin tarpeelliseksi.

4.4 Taloudelliset tekijät

Taloudellisia tekijöitä ovat erikoistarjoukset, hyvän pelin tukeminen, kohtuullinen hinnoittelu ja harrastukseen panostaminen. [Hamari *et al.*, 2017]. Hamarin ja muiden [2017] tutkimuksessa huomattiin, että taloudelliset syyt ovat hyvin tärkeä ostomotiivi. Pelaajat, jotka yrittävät käyttää mahdollisimman vähän rahaa virtuaaliseen sisältöön, saattavat kuitenkin tarjousten myötä päätyä ostamaan niitä [Hamari *et al.*, 2017].

Barnesin ja Guon [2009] tutkimuksessa henkilökohtaiset resurssit vaikuttivat hyvin paljon virtuaalisen sisällön ostamiseen. Rajallisen ajankäytön takia oli tärkeää ostaa virtuaalista sisältöä, joka nopeutti omaa etenemistään pelin sisällä. Muutamat osallistujat sanoivat, että eivät voi ostaa pelin tarjoamaa sisältöä oman taloudellisen tilanteen takia. Pelintekijät ovatkin huomanneet, että erikoistarjoukset ja rajoitetun ajan myynnissä olevat tuotteet ovat usein hyvä tapa saada pelaajat ostamaan virtuaalista sisältöä. Tämän takia monissa peleissä on päivittäisiä vaihtuvia tarjouksia, jonka avulla saadaan houkuteltua pelaajia ostamaan virtuaalista sisältöä. Moniin peleihin on luotu vuoden ympäri oleva julkaisusykli, jonka avulla pelintekijät pitävät pelin kiinnostavana ja liikevaihdon tasaisen lisäsisältöä hyödyntämällä.

4.5 Sisällön avaaminen ja lisäsisällön tarjoaminen lapsille

Sisällön avaaminen on tärkeä ostomotiivi riippumatta siitä, millaisesta pelistä on kyse. Kun pelaaja pitää pelistä, hän on usein valmis ostamaan virtuaalista lisäsisältöä siihen [Lehdonvirta, 2009].

Lapsille suunnatuissa videopeleissä myydään myös virtuaalista sisältöä. Alaikäiset eivät itse voi virtuaalista sisältöä ostaa, jolloin heidän vanhempansa suorittavat ostokset. Pelintekijät ovat yleensä tietoisia asiasta ja siksi yrittävät suunnitella pelin ottaen tämän rajoituksen huomioon.

5. Ostopaikkojen merkitys sisällön ostamisessa

Pelaajilla on käytössään useampi kanava virtuaalisen sisällön hankkimiseen. Ostotapahtuma voi tapahtua pelin sisällä tai kolmannen osapuolen palvelussa, kuten internetsivustolla. Seuraavaksi tarkastellaan, miten pelaajat valitsevat kanavan, mistä he ostavat virtuaalista sisältöä.

Barnesin ja Guon [2009] tutkimuksessa melkein kaikki osallistujat kokivat tärkeäksi, että palvelu, jossa transaktio tapahtuu, on toteutettu mahdollisimman selkeästi. Oston yhteydessä ei tule olla liian montaa vaihetta. He totesivat,

että usein nämä ostotapahtumat on toteutettu niin, ettei ostotapahtuma vaadi liikaa vaivannäköä pelaajalta ja että se on tehtävissä helposti ja nopeasti.

Monet pelaajat kokevat, että pelinsisäisessä käyttöliittymässä suoritetut ostot ovat usein yksinkertaisempia tehdä kuin esimerkiksi kolmannen osapuolen tarjoamalla internetsivustolla. Kun verrataan pelin sisällä tapahtuvaa ostosta johonkin kolmannen osapuolen internetsivustoon, osallistujat kokivat mieleiseksi, että sisältö on heidän käytössään välittömästi. Kun pelaaja ostaa virtuaalista sisältöä verkkosivulla olevasta palvelusta, saattaa ilmetä viivästyksiä tai odotusaikoja, mikä ei houkuta ostoksiin. [Barnes and Guo, 2009].

Kun peli on tarpeeksi laadukkaasti tehty ja sen tarjoama ostoalusta on toimiva, pelaajat kokevat palvelun myös luotettavampana. Osa vastaajista koki, että internetsivustot eivät ole niin luotettavia ostopaikkoja kuin pelin sisäiset ostopaikat [Barnes ja Guo, 2009]. Vaikka pelien sisäiset ostopaikat koetaan luotettavampina ja ostoprosessi on usein helpompi kuin muualla, silti muilla ostopaikoilla on kuitenkin tärkeä rooli virtuaalisen sisällön kaupankäynnissä.

Usein sisältö on edullisempaa, kun pelaajat itse myyvät virtuaalista sisältöä. Tämä johtuu siitä, että pelin sisäisessä ostopaikassa pelintekijä saattaa ottaa osuuden myydyn tuotteen hinnasta. Yksi esimerkki tästä on Steam-kauppapaikka, jossa palvelun tarjoaja saa 15% tuotteen myyntihinnasta.

Barnes ja Guo [2009] huomasivat myös, että toisinaan peleissä on todella harvinaista virtuaalista sisältöä, jonka arvo on käytännössä korkeampi kuin pelin sisäinen kauppapaikka mahdollistaa. Aikaisemmin Steam-kauppapaikalla korkein mahdollinen tuotteen myyntihinta oli 400 euroa. Tästä seurasi, että monia yli 400 euron arvoisia tuotteita myytiin muilla internet-sivustoilla, joissa ei ollut asetettu mitään rajaa virtuaalituotteiden myyntihinnalle. Maksimimyyntihinta on ajan kuluessa noussut 400 eurosta 1800 euroon. Osa vastaajista sanoi, että heille on tullut tavaksi ostaa virtuaalista sisältöä pelin tarjoamasta kauppapaikasta sen sijaan, että he edes yrittäisivät etsiä samoja tuotteita pelin ulkopuolelta [Barnes and Guo, 2009].

Kun pelaajalla on tiedossa haluamansa virtuaalinen tuote, hän valitsee oikean ostoalustan, vertailee tuotteita, niiden ominaisuuksia, hintaa ja myyjän luotettavuutta. Tehtyään vertailut hän tunnistaa parhaan vaihtoehdon ja suorittaa oston.

Tässä ostoprosessissa sosiaalisilla tekijöillä on suuri vaikutus. Barnesin ja Guon [2009] mukaan monet pelaajat hakivat neuvoa ystäviltään ja jakoivat ostokokemuksiaan keskenään. Luottamus ostettuun virtuaaliseen tuotteeseen ei ollut tärkeä tekijä, sillä tosielämään verrattuna pelaajan ei tarvitse pelätä, että tuote ei palvele tai toimi niin kuin he toivoisivat. On toki mahdollista, ettei tuo-

te vastaa täysin heidän odotuksiaan, mutta tehtyään tarpeeksi taustatutkimusta pelaajat usein tietävät tarkalleen, mitä he saavat vaihdossa.

Luottamukseen voi kuitenkin vaikuttaa missä ja kenen kanssa ostotapahtuma suoritetaan. Barnesin ja Guon [2009] tutkimuksessa ilmeni, että osallistujat kokivat mieluisammaksi käydä kauppaa tuttujen myyjien kanssa sekä heille tutuissa palveluissa. Tämä voi viestiä, että aivan kuten oikeassa elämässä, internetissä tapahtuvissa kauppa tapahtumissa pelätään huijatuksi tulemista. Osa pelaajista osti myös virtuaalista sisältöä myydäkseen kyseisen sisällön korkeampaan hintaan myöhemmin [Barnes and Guo, 2009]. Virtuaalisten tuotteiden hinnat vaihtelevat kysynnän ja tarjonnan mukaan, ja niiden avulla pelaajakin voi tehdä rahallista voittoa.

6. Pohdintaa

Aikaisemmin videopelit olivat pienelle yleisölle suunnattua viihdettä, mutta internetin, sosiaalisen median ja mobiilipelien myötä siitä on tullut suuri ja kasvava liiketoimi. Hyvä puoli on, että pelien yleistymisen myötä tarjontaa on runsaasti ja jokaiselle löytyy jotakin. Koska virtuaalisesta sisällöstä on tullut suuri osa pelialan liikevaihtoa, näkyy se myös pelisuunnittelussa ja pelien sisään rakennetuissa pelimekaniikoissa. Vaikka pelaajan ei yleensä ole pakko ostaa virtuaalista sisältöä, virtuaalisen sisällön monipuolinen tarjonta, markkinointi ja myynti vaikuttavat varsinaiseen pelaamiseen ja siitä saatavaan kokemukseen.

Hyvin todennäköistä on, että pelaaja kokee jäävänsä jostain paitsi, jos hänellä ei ole mahdollisuutta tai halua käyttää oikeaa rahaa virtuaaliseen sisältöön. Aikaisemmin peli oli jo julkaistaessa valmis kokonaisuus, jota ei pystytty päivittämään tai muokkaamaan jälkikäteen. Silloin pelaaja maksoi pelistä ostaessaan sen, mutta hänen ei tarvinnut jälkikäteen käyttää peliin rahaa ja hänellä oli pelin koko sisältö käytettävissä. Internetin myötä kuitenkin pelin päivittäminen on tehty mahdolliseksi ja se näkyy virtuaalisen sisällön kasvavassa tarjonnassa sekä myös videopelien julkaisuissa. Isotkin pelistudiot saattavat julkaista keskeneräiseltä tuntuvia pelejä, joita he sitten ajan saatossa muokkaavat ja parantavat. Tämä kuitenkin koetaan pelaajien keskuudessa hyvin negatiiviseksi. Siksi pelin ostajan täytyy olla hyvin tietoinen, onko hänen ostamansa videopeli tarpeeksi viimeistelty.

Peleille on usein asetettu hyvin tarkka julkaisupäivä, eikä pelintekijöillä aina ole mahdollisuutta lykätä pelin julkaisua, minkä takia viimeistelemättömiä videopelejä tuodaan markkinoille. Tällaisten pelien julkaiseminen on peliyhtiöille riskialtista, sillä videopelien tekeminen on erittäin kallista ja huonot myyntiluvut voivat ajaa pelistudion konkurssiin.

Koska kalliiden videopelien tekemisessä piilee aina iso taloudellinen riski, mobiilipelit ja sosiaalisissa medioissa pelattavat pelit ovat kasvaneet entistä suosituimmiksi pelimarkkinoilla. Ne ovat edullisempia tehdä ja virtuaalisen sisällön myynnillä nekin voivat olla hyvin tuottavia kustannuskuluihin nähden. Ilmaispelit ovat riskittömiä pelaajalle, sillä ne eivät maksa mitään. Tämän takia ilmaispelit nauttivat suurista pelaajakunnista ja koittavat saada liikevaihtoa mainosten ja virtuaalisen sisällön avulla.

Koska isojen ja kalliiden videopelien myyntituotot ovat elinehto peliyhtiöille, pelintekijät eivät uskalla ottaa riskejä ja tehdä innovatiivisia pelejä. Tästä johtuen monia pelisarjoja jatketaan hyvin pitkään, eivätkä videopelit muutu edellisiin osiinsa nähden järin paljon. Näin peliyhtiöt takaavat tasaisen tulonlähteen, kun he julkaisevat videopelejä, jotka ovat pelaajille tuttuja ja turvallisia.

Pelialalla useimmiten suurimmat muutokset pelisarjoihin näyttää tapahtuvan silloin, kun pelin myyntiluvut alkavat olla huonompia edellisiin osiin nähden. Näin pelaajien ostokäyttäytyminen osin vaikuttaa, millaisia pelejä tuotetaan markkinoille. Koska pelaajat äänestävät lompakoillaan, virtuaalista sisältöäkään ei olisi niin paljon tarjolla, elleivät peliyhtiöt kokisi sitä kannattavana liiketoimintana.

Kun videopeli on tehty hyvin ja pelaaja pitää siitä, hän todennäköisemmin myös käyttää rahaa pelin tarjoamaan virtuaaliseen sisältöön [Lehdonvirta, 2009]. Tästä johtuen, videopelit eivät voi liikaa luottaa virtuaalisen sisällön myymiseen, vaan pelin itsessään täytyy olla hyvä ja mieluisa kokemus pelaajalle. Peliyhtiö Sulake teki vuonna 2000 teki kilpailullisen verkkopelin nimeltä Snow War. Myöhemmin siihen lisättiin tulomalli, jossa pelaajat pystyivät tekstiviesteillä ostamaan erilaisia virtuaalisia tuotteita, kuten esimerkiksi isompia lumipalloja ja isompia takkeja. Käyttämällä rahaa pelinsisältöön saattoi pärjätä pelissä paremmin. Se ei kuitenkaan ollut pelaajien mukaan toivottu lisä ja sai pelaajilta huonon vastaanoton. Lopputulos oli, että vuoden kuluttua peliä ei enää pelattu ja se lopetettiin [Lehdonvirta, 2009].

Näyttää siltä, että jos ostettavalla pelisisällöllä on liian suuri vaikutus pelissä onnistumiseen, se helposti vähentää pelin suosiota. Runsaasti maksa voittaaksesi -ominaisuuksia sisältävät pelit eivät ole pelaajien mielestä hyviä, sillä ne koetaan epäreiluiksi [Lehdonvirta, 2009]. Hamarin ja muiden [2017] toteuttamassa tutkimuksessa kävi myös ilmi, että pelaajat eivät koe kilpailullisuutta kovin tärkeänä motiivina virtuaalihyödykkeiden ostamiselle. Tutkimustulokset tukevat siis näkemystä, että pelaajat arvostavat pelaamisella ja siihen liittyvällä osaamisella saavutettua pärjäämistä ja voittamista selvästi enemmän kuin rahalla hankittua.

7. Yhteenveto

Virtuaalisella sisällöllä tarkoitetaan videopeleissä käytettäviä asioita ja esineitä, kuten aseita, huonekaluja, valuuttoja ja tehosteita [Hamari, 2017]. Virtuaalisen sisällön ominaisuudet voidaan jakaa kolmeen ryhmään, toiminnallisiin, tunneperäisiin ja sosiaalisiin [Lehdonvirta, 2009]. Virtuaalisella sisällöllä voi olla yksi tai useampi näistä ominaisuuksista. Virtuaalisen sisällön arvo määräytyy näiden ominaisuuksien myötä sekä pelintekijän luomien pelimekaniikkojen avulla. Osassa videopeleistä erityisesti sosiaaliset ominaisuudet korostuvat, kun pelaajat itse määrittävät sisällön hinnan ja pelin ympärillä pyörivän talouden.

Virtuaalisen sisällön myynnistä on tullut iso osa pelialan liikevaihtoa. Tästä johtuen lähestulkoon kaikissa videopeleissä esiintyy ostettavaa virtuaalista sisältöä riippumatta siitä, onko kyseessä ilmaispeli, mobiilipeli, massiivinen verkkoroolipeli tai yksinpeli. Tämän seurauksena pelisuunnittelu on muuttunut viime vuosina. Peliyhtiöt hakevat liikevaihtoa entistä enemmän kytkemällä peleihin houkuttelevaa tai pelaamisen kannalta lähes välttämätöntä virtuaalista sisältöä. Pelintekijät tiedostavat, että pelin täytyy olla lähtökohdaltaan viihdyttävä, jotta pelaajat jatkavat pelaamista ja ovat halukkaita tekemään siihen liittyviä sisältöhankintoja.

Erään tutkimuksen mukaan 48% ilmaismobiilipelien liikevaihdosta tulee 0.19% pelaajakunnasta [Swrve, 2016]. Tämä tarkoittaa, että häviävän pieni osuus pelaajista kuluttaa paljon rahaa näihin peleihin. Ilmaispeleissä suurin ostomotiivi liittyy esteettömään pelaamiseen [Hamari, 2017]. Ilmaispeleissä on tyypillistä, että niihin on luotu keinotekoisia rajoitteita, jotka hidastavat tai estävät pelaamista, minkä myötä esteettömän pelaamisen tärkeys korostuu.

Näyttää siltä, että virtuaalituotteiden sosiaalisilla ominaisuuksilla on suuri vaikutus niiden hankintaan. Tämä näyttää vielä erityisesti korostuvan peleissä, joissa ollaan vuorovaikutuksessa muihin pelaajiin.

Tutkimuksissa kilpailulliset tekijät eivät nousseet niin tärkeiksi ostomotiiveiksi. Vaikka pelaajilla olisi mahdollisuus rahaa käyttämällä pärjätä paremmin ja saada etumatkaa muihin, Hamarin ja muiden [2017] mukaan kilpailullisuutta ei koettu järin tärkeänä tekijänä ostojen taustalla. Tähän vaikuttaa se, että rahalla pärjääminen ja voittaminen koetaan ymmärrettävästikin epäreiluna. Lisäksi ilmiötä selittänee se, että maksa voittaaksesi -peleissä pelaajat joutuisivat käyttämään suuria summia rahaa peliin, mikä pelimarkkinoilla johtaa helposti kohtuuttomuuksiin.

Pelaajien on helpompi järkeillä ostoksensa kohtuullisella hinnoittelulla ja kun pelintekijät hoitavat pelin tukemisen, PR-puolen ja asiakaspalvelun hyvin, on pelaaja valmiimpi tukemaan peliä rahallisesti. Kaikissa peleissä lisäsisällön

ja lapsille sisällön hankkiminen koettiin tärkeäksi, eikä se ole suoranaisesti sidottu peligenreihin tai alustoihin.

Voidaan todeta, että pelikulttuuri on kokenut suuren murroksen viimeisen 20 vuoden aikana. Valmiista peleistä ollaan siirrytty peleihin, joita myyntiin tultua jatkuvasti muokataan ja kehitetään. Pelien jatkojalostuksessa keskeisessä roolissa on niihin ostettavat virtuaaliset tuotteet. Virtuaalisessa sisällössä pelaajia kiinnostaa erityisesti tuotteiden tunneperäiset ja esteettiset näkökohdat. Ne tuottavat sellaisinaan pelaajille mielihyvää ja lisäävät pelinautintoa. Lisäksi tuotteiden sosiaaliset seikat ovat tärkeitä, sillä niillä pelaaja voi erottua massasta ja saada arvostusta muilta pelaajilta.

Ylipäätään pelien virtuaalisen sisällön kaupankäynnille on muodostunut omat markkinat, joilla pelaajat voivat myös tehdä keskenään kauppaa niin halvemmilla bulkkitavaroilla kuin kalliimmilla keräilyharvinaisuuksilla. Näin pelimaailma on saanut uusia ulottuvuuksia, joiden kehittelyssä vain taivas on rajana – vaikkei ehkä sekään virtuaalitodellisuudessa.

Viiteluettelo

Dot Esports. 2015. https://dotesports.com/counter-strike/match-fixing-counter-strike-ibuypower-netcode-guides-1256. Checked 20.5.2017.

Guo Y and Barnes S. 2009. Virtual item purchase behavior in virtual worlds: an exploratory investigation. *Electronic Commerce Research* 9, 77-96.

Hamari J, Alha K, Järvelä S, Kivikangas JM, Koivisto J and Paavilainen J. 2017. Why do players buy in-game content? an empirical study on concrete purchase motivations. *Computer Human Behavior* 68, 538-546.

Hsiao K and Chen C. 2016. What drives in-app purchase intention for mobile games? an examination of perceived values and loyalty. *Electronic Commerce Research and Applications* 16, 18-29.

Hsu C and Lu H. 2007. Consumer behavior in online game communities: A motivational factor perspective. *Computer Human Behavior* 23(3), 1642-1659.

Lehdonvirta V. 2009. Virtual item sales as a revenue model: Identifying attributes that drive purchase decisions. *Electronic Commerce Research* 9(1), 97-113.

Liquipedia.net.http://wiki.teamliquid.net/counterstrike/North_American_match_fixing_scandal. Checked 23.5.2017.

Supercell. 2017. http://supercell.com/en/ Checked 23.5.2017.

SuperData. 2016. Market Brief — Year in Review 2016. https://www.superdataresearch.com/market-data/market-brief-year-in-review/. Checked 23.5.2017.

Swrve. 2016. The Swrve monetization report. https://www.swrve.com/ images/uploads/resources/swrve-monetization-report-2016.pdf. Checked 23.5. 2017

Virtuaalitodellisuus ja graafisten käyttöliittymien uusi ulottuvuus

Jussi Karhu

Tiivistelmä.
VR-teknologioiden nopea yleistyminen on tuonut mukanaan kirjavan joukon erilaisia graafisia käyttöliittymiä. Kolmiulotteisissa virtuaaliympäristöissä ei ole vielä samanlaisia vakiintuneita käyttöliittymäelementtejä ja käytänteitä kuin kaksiulotteisissa graafisissa käyttöliittymissä.

Tässä tutkielmassa esitetään erilaisia tapoja toteuttaa graafisia käyttöliittymiä virtuaaliympäristöissä. Lisäksi tutkielmassa kuvataan erilaisia 3D-ympäristöön soveltuvia, markkinoilla olevia VR-laitteita hyödyntäviä vuorovaikutusmenetelmiä.

Avainsanat ja -sanonnat: Virtuaalitodellisuus, VR, käyttöliittymät, GUI, 3DUI.

1. Johdanto

Virtuaalitodellisuus (VR) ei ole käsitteenä uusi. Ensimmäisen päähän kiinnitettävän näyttölaitteen (HMD) kehitti Ivan Sutherland jo 1960-luvulla [Sutherland 1968]. Useiden vuosikymmenien ajan kalliita VR-järjestelmiä kehitettiin ja käytettiin lähinnä puolustuslaitoksissa ja tutkimuskeskuksissa. 2010-luvun puolivälin jälkeen virtuaalitodellisuus on kuitenkin vihdoin noussut myös suurten massojen tietoisuuteen. Merkittävä syy tähän on suurten teknologiayritysten, kuten Facebookin ja Googlen, valtavat panostukset kyseisen teknologian kehittämiseen sekä markkinoille tulleet kuluttajille suunnatut HMD-laitteet. Yritysten kiinnostus VR-teknologiaan on kovassa kasvussa. Monet sosiaalisen median palvelut tukevat jo 360 asteen videoiden jakamista ja maailman suurimmat mediayhtiöt tuottavat niihin kilpaa sisältöä.

Sisällöntuotanto virtuaalitodellisuuteen ei ole aivan yksinkertaista. Valtaosa tällä hetkellä tuotetusta sisällöstä on erilaisilla erikoiskameroilla kuvattuja 360-videoita ja -kuvia. Hyvällä kuvanlaadulla, kolmiulotteisella äänellä ja tarinankerronnalla on mahdollista saavuttaa vahva *immersio* eli läsnäolon tunne. Vain kuvaamalla tuotettu materiaali on kuitenkin luonteeltaan hyvin staattista, eikä se mahdollista vapaata liikkumista tilassa tai interaktiota virtuaalisen ympäristön (VE) kanssa. Toinen mahdollisuus on luoda virtuaalinen ympäristö käyttämällä 3D-mallinnusta ja hyödyntämällä esimerkiksi jonkinlaista pelimoottoria. Näin toimimalla voidaan antaa käyttäjän liikkua ja olla vuorovaikutuksessa virtuaalisen ympäristön kanssa täysin vapaasti käytetyn laitteiston ja fyysisen ympäristön sallimissa rajoissa.

Virtuaalitodellisuus tuo oman haasteensa myös graafisten käyttöliittymien suunnitteluun. Kaksiulotteisissa käyttöliittymissä olemme saavuttaneet jo melko vakiintuneita käytänteitä. Suurin osa tietokoneilla käytettävistä käyttöliittymistä noudattaa yhä XEROX PARCissa 1980-luvulla kehitettyä WIMP (window-icon-menu-pointer) -tyyliä [Dix 2009]. Kaksiulotteisia käyttöliittymiä on helppo viedä myös virtuaalitodellisuuteen upottamalla ne johonkin tasaiseen tasoon, kuten virtuaaliympäristön seinään, tai peleistä tutulla HUD-tyylillä lukitsemalla niiden asento ja sijainti virtuaalisen kameran eteen. Kyky liikkua vapaasti virtuaaliympäristössä mahdollistaa kuitenkin hyvin erilaistenkin graafisten käyttöliittymien toteuttamisen. Aiemmin totutusta hyvin paljon poikkeavia syöttölaitteita käytettäessä tutut käyttöliittymäelementit, kuten liukusäätimet ja valintaruudut, eivät välttämättä enää olekaan kaikista toimivimpia ratkaisuja.

Tämän tutkielman tarkoitus on esitellä erilaisia ratkaisuja kolmiulotteisiin graafisiin käyttöliittymiin (3D GUI) alan kirjallisuutta hyväksi käyttäen. Tutkielman on tarkoitus vastata kysymykseen ”Kuinka toteuttaa graafisia käyttöliittymiä virtuaalitodellisuudessa?”. Virtuaalitodellisuudella voidaan viitata hyvin monenlaisiin teknologisiin ratkaisuihin ja virtuaaliympäristöjä voidaan esittää monenlaisilla laitteilla, mutta tässä tutkielmassa keskitytään erityisesti erilaisilla HMD-laitteilla esitettyyn virtuaalitodellisuuteen ja interaktioihin jo markkinoilla olevien syöttölaitteiden avulla. Tutkielmassa keskitytään graafisiin käyttöliittymiin, minkä vuoksi esimerkiksi yksi immersiivisten kokemusten olennaisimmista tekijöistä, kolmiulotteinen ääni, jätetään kokonaan huomioimatta.

Seuraavassa luvussa esitellään erilaisia näyttö- ja syöttölaitteita, joilla virtuaaliympäristöjä voidaan esittää. Luku 3 käsittelee virtuaalisten objektien valintaa ja käsittelyä. Luvussa 4 esitellään erilaisia graafisia käyttöliittymäelementtejä ja erilaisten toimintojen esittämistä virtuaalitodellisuudessa. Luku 5 sisältää yhteenvedon tutkielmasta.

2. Laitteisto

2.1. Näyttölaitteet

Virtuaaliympäristöjä voidaan esittää hyvin monenlaisilla näyttölaitteilla. Laitteet eroavat toisistaan esimerkiksi näkökentän laajuuden (FOV), resoluution, päivitystaajuuden ja ergonomian osalta. Bowman ja muut [2004] luettelevat jopa yhdeksän erilaista näyttölaitetyyppiä, joita voidaan käyttää virtuaaliympäristöjen esittämiseen. Niihin kuuluvat muun muassa tavalliset monitorit, puolipallonäytöt, erilaiset surround-näytöt ja -projektiot (CAVE) sekä kädessä pidet-

tävät näyttölaitteet. Käytetyt näyttölaitteet vaikuttavat suuresti siihen, minkälaiset käyttöliittymäratkaisut ovat toimivia, joten tässä tutkielmassa keskitytään vain erilaisilla silmikkonäytöillä (HMD) esitettävään virtuaalitodellisuuteen. Myös muita tulostuslaitteita voidaan käyttää virtuaalitodellisuuden esittämiseen. Esimerkiksi kolmiulotteisella äänellä, tunto- ja jopa hajupalautteella voidaan saavuttaa äärimmäisen immersiivisiä kokemuksia. Näitä elementtejä ei kuitenkaan käsitellä tässä tutkielmassa.

Eri laitevalmistajat ovat viime vuosina tuoneet markkinoille lukuisia erilaisia silmikkonäyttöjä. Yleisimmät laitteet voi karkeasti jakaa kahteen kategoriaan: mobiileihin älypuhelimiin kytkettäviin ja tietokoneeseen tai pelikonsoliin kytkettäviin silmikoihin. Lisäksi markkinoille on tulossa täysin itsenäisesti toimivia VR-laitteita, jotka eivät tarvitse lainkaan erillistä puhelinta tai tietokonetta.

Edelliseen kategoriaan kuuluvat muun muassa Samsung Gear VR sekä Google Daydream View. Molemmat vaativat toimiakseen laseihin liitettävän puhelimen, jolloin käytetyn puhelimen ominaisuudet, kuten näytön koko ja resoluutio, vaikuttavat kokemukseen. Gear VR käyttää pään liikkeiden seurantaan silmikon sisältämiä liike- ja asentosensoreita, kun taas Daydream View hyödyntää puhelimien sisäänrakennettuja sensoreita. Molemmilla laitteilla FOV on suhteellisen rajallinen. Ihmisen keskimääräisen näkökentän laajuuden ollessa keskimäärin noin 200 astetta [Bowman et al. 2004], uusin Gear VR kykenee näyttämään kuvaa vain hieman yli 100˚ ja Daydream View hieman yli 90˚ laajuudelta. VR-sovelluksia voidaan käyttää uudemmilla puhelimilla myös monien halvempien silmikkojen, kuten Google Cardboardin, kanssa. Tällöin näytön tarkkuus ja käytettävissä olevat ominaisuudet riippuvat täysin käytetyn puhelimen ominaisuuksissa.

Jälkimmäiseen kategoriaan kuuluvat erilaiset tietokoneisiin kytkettävät HMD-laitteet. Näistä suosituimpia ovat Oculus Rift sekä HTC Vive. Molemmissa laitteissa näytön resoluutio on 1200 x 1080 pikseliä per silmä ja FOV on noin 110˚. Molemmat laitteet päivittävät näyttöjen kuvaa 90Hz taajuudella. Laitteet vaativat toimiakseen suhteellisen tehokkaan tietokoneen, sillä kaikki laskenta suoritetaan sillä.

2.2. Syöttölaitteet

Kaikki aikaisemmin esitellyt näyttölaitteet toimivat myös syöttölaitteina. HMD-pohjainen virtuaalitodellisuus vaatii vähintäänkin silmikon asennon seurantaa, jolloin näytöillä näytettävä kuva voidaan jatkuvasti päivittää vastaamaan käyttäjän pään asentoa. PC- ja konsolipohjaiset VR-järjestelmät, kuten Vive, Rift ja PSVR, seuraavat lisäksi myös käyttäjän liikettä ja mahdollistavat täten pään

pyörittämisen lisäksi myös virtuaalisessa tilassa liikkumisen. Eri valmistajien laitteet käyttävät hieman erilaisia menetelmiä liikkeen seurantaan. Esimerkiksi HTC Vive seuraa käyttäjän liikettä tilan nurkkiin asennettavien infrapunamajakoiden avulla, joiden ansiosta käyttäjä pystyy liikkumaan fyysisessä tilassa jopa 25 m^2:n alueella.

Joissain erityisesti mobiilipohjaisissa ratkaisuissa näyttölaitteena toimiva puhelin voi toimia ainoana käytettävänä syöttölaitteena. Tällaisilla ratkaisuilla mahdollisuudet vuorovaikutukseen virtuaalimaailman kanssa ovat kuitenkin hyvin rajalliset ja haluttaessa virtuaalikokemukseen enemmän vuorovaikutusta, vaaditaan usein lisäksi erillinen syöttölaite.

Syöttölaitteena virtuaalitodellisuudessa voidaan joissain tapauksissa käyttää perinteistä näppäimistö-hiiri -yhdistelmää. Tämä tulee kyseeseen lähinnä tapauksissa, joissa käyttäjä istuu työpöydän ääressä. Tavallinen hiiri mahdollistaa osoittimen liikuttamisen vain kahdessa ulottuvuudessa, jolloin siitä on hyötyä lähinnä siinä tapauksessa, jos kolmiulotteisessa ympäristössä esitetään tietoa kaksiulotteisissa ikkunoissa. Myös perinteisen näppäimistön mukana kantaminen ja sillä kirjoittaminen on hyvin ongelmallista, mikäli kokemus sallii vapaasti tilassa liikkumisen. Tavallisesti silmikkonäyttö peittää ympäröivän maailman, minkä vuoksi käyttäjä ei voi myöskään nähdä näppäimistöä. Jotkin sovellukset hyödyntävät myös pelikonsoleista tuttuja peliohjaimia, mutta virtuaalitodellisuuden tarpeisiin paremmin soveltuvat erilaiset laitteet, jotka mahdollistavat käsien tai ohjaimien liikkeen seurannan ja näin antavat 3D-käyttöliittymissä kaivattuja lisävapausasteita.

Google ja Samsung ovat molemmat tuoneet markkinoille mobiilipohjaisten VR-laitteidensa kanssa yhteensopivat käsiohjaimet. Molemmissa ohjaimissa on kosketuslevy, painikkeita sekä anturit ohjaimen liikkeen seurantaan. Ohjaimen liikkeen tarkka seuranta mahdollistaa sen käytön 3D-ympäristössä esimerkiksi taikasauvatyyppisenä osoittimena tai virtuaalikätenä. Samsungin ohjaimessa on lisäksi liipaisinpainike, joka helpottaa esimerkiksi 3D-objekteihin tarttumista virtuaaliympäristössä.

PC-pohjaisissa VR-järjestelmissä yleisin ratkaisu ovat käyttäjän molempiin käsiin tulevat käsiohjaimet. Tällaisia syöttölaitteita käytetään esimerkiksi HTC Vive ja Oculus Rift -järjestelmissä. Molempien ohjaimissa on liipaisin- ja puristuspainikkeet, jotka mahdollistavat luonnollisen vuorovaikutuksen virtuaalimaailman kanssa. Muutamien muiden painikkeiden lisäksi Viven ohjaimesta löytyy peukalolla käytettävä kosketuslevy. Oculuksen ohjaimessa on painikkeiden lisäksi pieni kosketusanturi peukalon alla sekä ohjaussauva.

Erillisten käsiohjainten lisäksi VR-sovelluksissa voidaan hyödyntää myös muunlaisia menetelmiä käyttäjän käsien seurantaan. Esimerkiksi Leap Motion

on silmikkoon kiinnitettävä laite, joka infrapunakameroiden ja -ledien avulla mahdollistaa käyttäjän käsien ja jopa yksittäisten sormien tarkan seurannan. Sama lopputulos voidaan saavuttaa myös erilaisilla datakäsineillä, jotka rekisteröivät käsien ja sormien liikkeet.

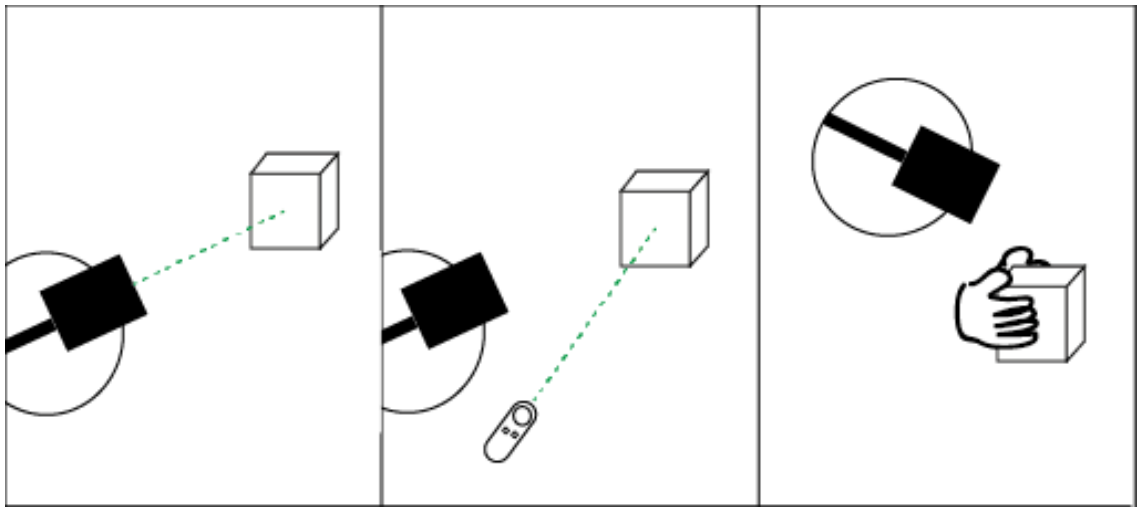
3. Objektien valinta ja käsittely virtuaalitodellisuudessa

3.1. Objektien valinta erilaisilla syöttölaitteilla

Mine [1995] jakaa objektien valintatavat kahteen pääkategoriaan; paikalliseen ja etäältä tapahtuvaan valintaan. Jos objekti on käyttäjän ulottuvilla, on kyse paikallisesta valinnasta ja käyttäjä voi olla objektin kanssa suorassa vuorovaikutuksessa. Kun objekti on kaukana eikä käyttäjä yletä siihen, tarvitaan jonkinlainen etävalintametodi. Kaikki valintatavat koostuvat kahdesta vaiheesta. Ensin tarvitaan jokin mekanismi, jolla osoitetaan valittava objekti ja sen jälkeen signaali, jolla varsinainen valinta vahvistetaan.

Käytetyt syöttölaitteet sanelevat reunaehdot valittaville vuorovaikutusmenetelmille. Jos käytössä on esimerkiksi halpa pahvinen puhelimen kanssa käytettävä silmikko, voi puhelimen asento olla ainoa mahdollinen tapa valita ja aktivoida elementtejä virtuaalimaailmassa. Puhelimen ollessa kiinni silmikossa, kertoo laitteen asento suunnan, johon käyttäjän kasvot osoittavat. Tätä tietoa voidaan käyttää hyväksi objektien valinnassa. Valinta etänä voi tapahtua esimerkiksi niin kutsutulla *ray-casting* -menetelmällä. Virtuaalisesta kamerasta lähetetään virtuaalinen säde sen osoittamaan suuntaan (kuvassa 1 vasemmalla). Tämän jälkeen sovellus saa tiedon, mikäli säde törmää matkallaan johonkin objektiin.

Mikäli mitään painikkeita ei ole käytettävissä, eikä objektien valintaa haluta tehdä välittömästi, voidaan aktivointiin lisätä pieni viive ja tehdä se ajastetusti. Tällöin säteen täytyy osua, eli käyttäjän täytyy kohdistaa kasvonsa, objektiin esimerkiksi parin sekunnin ajan, jonka jälkeen aktivointi tapahtuu. Atienzan ja muiden [2016] mukaan ”katse” pään asennosta on toimiva vuorovaikutustapa ainakin objektien valinnassa. Jos käytössä on edistyneempi laite, joka sisältää katseenseurannan, voidaan käyttäjän katsetta käyttää samoin tavoin. Laite seuraa mihin päin käyttäjä katsoo suhteessa laitteen rintamasuuntaan, ja sovellus lähettää säteen vastaavaan suuntaan virtuaalikamerasta.



Kuva 1. Etänä tapahtuva valinta virtuaalikamerasta ammuttavalla säteellä (vasemmalla) ja ohjaimen mukaan liikkuvasta objektista (keskellä) sekä paikallisesti tapahtuva valinta käsimetaforilla (oikealla).

Jos VR-järjestelmään on kytketty erillinen liikkeen ja asennon tunnistava käsiohjain, on vastaavaa menetelmää mahdollista käyttää sen avulla. Ohjaimen liike- ja asento jäljitellään yksi yhteen virtuaalimaailmassa näkyvällä 3D-mallilla. Malli voi esittää ohjainta sellaisenaan tai se voi näyttää esimerkiksi taikasauvalta. Valintasäde lähetetään ohjainlaitetta vastaavan objektin päädystä, jolloin ohjain toimii ikään kuin laserosoittimena (kuva 1). Tällaisella tekniikalla osoittimen seuranta on usein jatkuvaa ja valinnan vahvistaminen voidaan tehdä esimerkiksi painamalla ohjaimen liipaisinpainiketta. Lasermetafora mahdollistaa myös hieman kauempana sijaitsevien objektien tarkan ja nopean valinnan ja toimii pitkälti samoin kuin hiiren kursori kaksiulotteisissa ympäristöissä.

Toinen erityisesti edistyneempien VR-järjestelmien kanssa paljon käytetty vuorovaikutuskeino on käsimetafora. Tämä tulee kyseeseen siinä tapauksessa, että objekti on käyttäjän ulottuvilla, eli valintatapa on Minen mukaan paikallinen. Siinä käyttäjän toisen tai molempien käsien liikkeet viedään yksi yhteen virtuaalitodellisuuteen käyttäen hyväksi käsiohjaimia tai jotain muuta käsienseurantamenetelmää (kuva 1). Kolmiulotteinen käsi seuraa käsiohjaimien liikkeitä ja asentoa samoin kuin virtuaalikamera HMD:ta. Käyttäjän kurottaessa kätensä metrin eteenpäin, liikkuu virtuaalikäsi vastaavan matkan virtuaalimaailmassa. Objektien valinta tapahtuu tarkkailemalla virtuaalikäden törmäyksiä. Sovellus saa tiedon, jos käsimetafora osuu johonkin objektiin virtuaaliympäristössä. Tämän jälkeen valinta voidaan vahvistaa esimerkiksi käsien seurantaa käytettäessä puristamalla käsi nyrkkiin. Tällainen esineeseen tarttuminen toimii aivan kuten todellisessakin maailmassa, jolloin interaktio on luonnollinen, nopea ja helppo oppia. Käsiohjaimilla valinta voidaan vahvistaa esimerkiksi oh-

jaimen puristus- tai liipaisinpainikkeilla. Tämäkin on hyvin lähellä ihmisen luonnollista tapaa tarttua esineisiin ja täten toimiva ratkaisu. Käsimetafora voidaan korvata myös ilmassa leijuvilla ohjaimilla, jotka jäljittelevät tarkasti käyttäjän käsissä olevia ohjaimia. Tästä on hyötyä käyttäjälle erityisesti ohjaimien eri painikkeiden paikantamisessa.

3.2. Objektien käsittely erilaisilla syöttölaitteilla

Käytetyt syöttölaitteet määrittävät hyvin pitkälti rajat sille, minkälaisia interaktioita voidaan käyttää objektien käsittelyyn virtuaaliympäristössä. Poupyrev ja Ichikawa [1999] jakavat virtuaaliympäristössä tapahtuvan objektien käsittelyn kahteen pääkategoriaan; egosentriseen ja eksosentriseen. Egosentrisessä, eli minäkeskeisessä, vuorovaikutuksessa käyttäjä on sisällä virtuaaliympäristössä, kun taas eksosentrisessä hän katselee virtuaaliympäristöä sen ulkopuolelta. Jälkimmäisessä tapauksessa virtuaalimaailma voidaan esittää esimerkiksi pienoiskoossa. Useissa VR-sovelluksissa näitä menetelmiä yhdistellään sulavasti, jolloin käyttäjä voi tehdä laajemman mittakaavan muutoksia virtuaaliympäristön ulkopuolelta sekä pienempiä ja tarkempia sen sisäpuolella. Suurin osa virtuaaliympäristöissä tapahtuvasta vuorovaikutuksesta on kuitenkin egosentristä [Poupyrev and Ichikawa 1999] ja tutkielmassa käsitellään lähinnä siihen kategoriaan kuuluvia menetelmiä. Käytetty käsittelytapa on useimmiten sidoksissa käytettyyn valintatapaan, vaikkakin erilaisia menetelmiä voidaan yhdistellä melko vapaasti.

Mine [1995] nimeää kolme yleisintä virtuaalisten objektien manipulointiin liittyvää tehtävää: objektin siirtäminen, pyörittäminen ja sen koon muuttaminen. Kaikki eivät laske objektin koon muuttamista täysin omaksi toimenpiteekseen. Esimerkiksi Bowman ja muut [2004] perustelevat tämän sillä, että 3D-objektin muodon muuttaminen tapahtuu käytännössä muita edellä mainittuja toimia, eli valintaa, siirtämistä ja pyörittämistä hyödyntäen. Egosentriset käsittelymenetelmät voidaan jakaa edellä kuvattujen valintamenetelmien tavoin paikallisiin ja etänä tapahtuviin [Mine 1995].

Paikallisesti tapahtuva käsittely, virtuaalisen objektin ollessa käyttäjän käden ulottuvilla, on usein suoraa ja vastaa esineiden käsittelyä todellisessa elämässä. Esimerkiksi liikkeen tunnistavia käsiohjaimia käytettäessä voidaan ohjaimen asento ja liike viedä 1:1 virtuaalisen objektin asentoon ja sijaintiin. Kun käyttäjä on ensin käsi- tai ohjainmetaforallaan valinnut halutun objektin, lähtee se liikkumaan ohjainta liikuttaessa. Käyttäjän liikuttaessa ohjainta metrin eteenpäin, myös virtuaaliobjektin sijainti muuttuu metrin ja käännettäessä ohjain ylösalaisin, myös objekti pyörähtää päälaelleen. Sama idea toimii myös muita käsienseurantamenetelmiä, kuten Leap Motionia tai datakäsineitä, käy-

tettäessä. Useimmiten käyttäjän täytyy pitää objekti valittuna koko toimenpiteen ajan esimerkiksi ohjainta puristamalla tai liipaisinpainiketta painamalla. Käyttäjän irrottaessa otteensa painikkeesta myös virtuaaliobjekti vapautuu, eivätkä ohjaimen tai käsien liikkeet enää vaikuta siihen. Riippuen siitä, millaista fysiikkamallinnusta sovelluksessa käytetään, objekti jää tämän jälkeen joko paikoilleen tai jatkaa matkaansa. Fysiikkamallinnuksen avulla virtuaaliobjekti voidaan esimerkiksi pudottaa, heittää tai laittaa vierimään eteenpäin. Tällainen objektien välitön käsittely on erittäin luonnollista ja vastaa pitkälti sitä, miten olemme tottuneet esineitä käsittelemään.

Suorassa käsiohjatussa vuorovaikutuksessa myös objektien koon muuttaminen voidaan toteuttaa luonnollisella tavalla. Käyttäjän tarttuessa objektiin kahdelta eri sivulta ja liikuttaessa käsiään joko lähemmäs tai kauemmas toisistaan, voidaan objektin kokoa muuttaa vastaavasti, aivan kuten sitä venytettäisiin tai puristettaisiin kasaan [Mine 1995].

Mikäli käytettävissä on ainoastaan pään asento, eli HMD ilman erillisiä ohjaimia, ovat kaikki objektin valintaa monimutkaisemmat toimenpiteet hyvin hankalia. Esimerkiksi pelkällä ray-casting -menetelmällä on käytännössä mahdotonta liikuttaa objektia vapaasti virtuaaliympäristössä [Poupyrev and Ichikawa 1999]. Objektia voidaan liikuttaa virtuaalikameraa ympäröivällä kehällä käyttäjän katseen mukana, mutta sen liikuttaminen lähemmäs ja kauemmas virtuaalikamerasta vaatii erillisen ohjaimen. Tähän toki riittää esimerkiksi Samsung Gear VR:stä löytyvä kosketuslevy. Objektin pyöritys ja koon muuttaminen voidaan myös toteuttaa kosketuslevyä, tai muita erillisiä ohjaimia käyttämällä.

Myös etänä tapahtuva objektin käsittely voidaan käsiohjaimia tai käsien seurantaa käyttämällä toteuttaa lähes samalla tavalla kuin paikallisesti tapahtuva. Yksi tällainen menetelmä on nimeltään Go-Go [Poupyrev et al. 1996]. Siinä käden ulottuvilla olevien objektien käsittely tapahtuu aivan kuten välittömässäkin interaktiossa. Käyttäjän kurottaessa kätensä etäämmällä sijaitsevaa objektia päin, alkaa virtuaalinen käsi ikään kuin venyä, jolloin käyttäjä yltää tarttumaan objektiin. Toinen hieman vastaavanlainen on nimeltään HOMER. Bowmanin ja Hodgesin [1997] esittelemässä menetelmässä yhdistellään ray-casting -menetelmää ja suoraa käsien seurantaan pohjautuvaa vuorovaikutusta. Käyttäjä voi valita etäällä sijaitsevan objektin lähettämällä virtuaalisen säteen, mutta objekti ei kiinnitykään itse säteeseen, vaan käyttäjän käsimetafora kiinnittyy kyseiseen objektiin. Tämän jälkeen käyttäjä voi käsitellä sitä aivan kuin se olisi käden ulottuvilla. Objektin liikuttamisessa käytetään käyttäjän käsien sijainnin suhdetta tämän vartaloon sekä näiden suhdetta virtuaalisen objektin etäisyyteen.

Näiden menetelmien lisäksi objekteja voidaan manipuloida käyttämällä erilaisia fyysisiä tai virtuaalisia ohjaimia. Käsittely voidaan toteuttaa esimerkiksi HTC Viven käsiohjaimien kosketuslevyä tai Oculus Touch -ohjaimien joystick-ohjaimia hyödyntäen. Muutamia virtuaalisia ohjaimia ja käyttöliittymäelementtejä kuvaillaan seuraavassa luvussa.

4. Graafiset käyttöliittymäelementit virtuaalitodellisuudessa

4.1. Kaksiulotteiset käyttöliittymäelementit

Yksi varsin laajasti käytetty menetelmä toteuttaa graafisia käyttöliittymiä virtuaalisessa ympäristössä on yksinkertaisesti sijoittaa perinteinen kaksiulotteinen GUI 3D-ympäristöön. Feiner ja muut [1993] jakavat 3D-ympäristöön sijoitetut 2D-ikkunat kolmeen pääkategoriaan: ympärille sijoitettuihin, maailmaan kiinnitettyihin sekä näkymään kiinnitettyihin ikkunoihin. Näistä ensimmäisessä käyttöliittymäelementit voidaan asetella tasoon, joka sen jälkeen laitetaan esimerkiksi leijumaan määrättyyn kohtaan 3D-tilassa. Käyttäjän liikkuessa virtuaalitilassa ja katsellessa ympärilleen, hän näkee eri osia ikkunasta erilaisista kuvakulmista. Ilmassa leijuva GUI ei kuitenkaan välttämättä tunnu kovin immersiiviseltä. Todentuntuisempi vaihtoehto voi olla GUI:n sijoittaminen johonkin virtuaaliympäristössä olevaan tasoon tai pintaan, esimerkiksi huoneen seinään, jolloin sitä voi ajatella ikään kuin näyttönä. Nämä kuuluvat Feinerin ja muiden jaottelussa maailmaan kiinnitettyihin ikkunoihin.

Tällainen virtuaalinäyttö voidaan kiinnittää myös liikuteltavaan 3D-objektiin, jolloin sitä voidaan käyttää esimerkiksi kuten tablettitietokonetta. Lindemanin ja muiden [1999] mukaan kädessä pidettävät 2D-ikkunat mahdollistavat käyttäjän tehokkaan toiminnan virtuaaliympäristössä. Tällainen tablettimainen näyttö ei peitä käyttäjän näkyvyyttä eikä riko immersiota, mutta on aina käden ulottuvilla ja nopeasti käytettävissä, kun sille on tarvetta. Tutkimuksessaan he käyttivät fyysistä mailamaista esinettä, joka vastasi virtuaalitodellisuudessa näkyvää kosketusnäyttöä. Vastaava toteutus voitaisiin tehdä myös käyttämällä oikeaa tablettitietokonetta, jolloin laitteen kosketusnäyttö voitaisiin hyödyntää syöttölaitteena. Tämä vaatisi tietysti sekä tabletin että käyttäjän käsien seurannan, jotta käyttö onnistuisi luontaisesti.

Feinerin ja muiden mainitsema kolmas kategoria on näkymään kiinnitetyt ikkunat. Tässä käyttöliittymäelementtien sijainti on sidoksissa virtuaalikameraan, jolloin ne voidaan pitää aina käyttäjän näkyvillä. Ikkuna liikkuu mukana käyttäjän liikkuessa ja kääntäessä päätänsä. Tällainen HUD-tyyppinen ratkaisu on tuttu esimerkiksi videopeleistä, jossa sitä käytetään hyvin yleisesti. Virtuaalitodellisuudessa näkökentässä kiinteästi pysyvä näyttö saattaa kuitenkin vä-

hentää immersiivisyyden tunnetta, sillä se ei vastaa kokemustamme todellisuudesta. Näkymään kiinnitetyt käyttöliittymäelementit myös peittävät käyttäjän näkyvyyttä 3D-ympäristöön, minkä vuoksi niitä kannattaa käyttää hyvin harkiten, tai mahdollistaa ikkunan piilottamisen halutessa esimerkiksi näkökentän alapuolelle.

Soveltuvat vuorovaikutustavat riippuvat täysin kaksiulotteisen käyttöliittymän toteutuksesta. Kohtuullisen suuria painikkeita ja muita käyttöliittymäelementtejä voidaan käsitellä suoraan esimerkiksi käsi- tai ohjainmetaforalla. Tällöin vuorovaikutus on samanlaista kuin käyttäisi kosketusnäyttöä. Aivan kuten kosketusnäytöilläkin, voivat hyvin pienet kohteet, kuten esimerkiksi valintalaatikot, olla ongelmallisia. Toinen yleisesti käytetty tapa on ray-casting -menetelmä. Laserosoitinmetafora mahdollistaa myös hieman pienempiin käyttöliittymäelementteihin osumisen.

Kaksiulotteisten graafisten käyttöliittymien tuominen kolmiulotteiseen ympäristöön ei kuitenkaan aina ole toimivin mahdollinen ratkaisu. 2D-ympäristöissä hyviksi ja tehokkaiksi todetut elementit, kuten liukusäätimet, eivät välttämättä toimi yhtä tehokkaasti kolmiulotteisessa ympäristössä. Kaksiulotteista liukusäädintä voi käyttää vain oikeasta suunnasta katsottaessa [Patterson 2016], mutta esimerkiksi sivusta katsoessa sen käyttö on mahdotonta.

4.2. Kolmiulotteiset käyttöliittymäelementit

Siirryttäessä kaksiulotteisista ympäristöistä kolmiulotteisiin, on täysin luonnollista, että kehittäjät alkavat kokeilla myös erilaisia kolmiulotteisia käyttöliittymäelementtejä. Yksi yleisimmistä lähestymistavoista on esittää tai hieman muokata todellisesta maailmasta tuttuja elementtejä [Bowman et al. 2004]. Erityisesti erilaisissa simulaatioissa tutut fyysiset elementit halutaan kopioida hyvin todenmukaisesti virtuaaliympäristöön. Bowmanin ja muiden mukaan visuaalinen realismi ei välttämättä ole tärkeää. Huomattavasti tärkeämpää on simulaation vuorovaikutuksen todenmukaisuus. Visuaalista todenmukaisuutta rajoittaa myös käytetty teknologia. Esimerkiksi nykyisten HMD-laitteiden resoluutio on vielä sen verran rajallinen, ettei niillä ole vielä mahdollista näyttää täysin aidolta näyttäviä mallinnuksia.

Järjestelmän tilaa voidaan muuttaa käyttämällä esimerkiksi kolmiulotteisia vipuja, painikkeita ja liukusäätimiä. Yhdessä suoraan käsien seurantaan tai käsiohjaimiin perustuvan vuorovaikutuksen kanssa voidaan saavuttaa ympäristöön sulautettu käyttöliittymä, jota käyttäjä osaa käyttää helposti ilman erillistä ohjausta tai opettelua. 3D-maailmaan sijoitetun käyttöliittymän ongelma on kuitenkin se, että se ei välttämättä ole aina tarvittaessa käyttäjän ulottuvilla. Lisäksi Pattersonin [2016] mukaan esimerkiksi kolmiulotteinen liukusäädin voi

olla ongelmallinen, mikäli sitä joudutaan käyttämään jostakin muusta kuin optimaalisesta katselukulmasta. Erityisesti usean käyttäjän sovelluksissa tällaista elementtiä voi olla mahdoton esittää niin, että sen käyttö olisi helppoa kaikille eri suunnista katsoville käyttäjille.

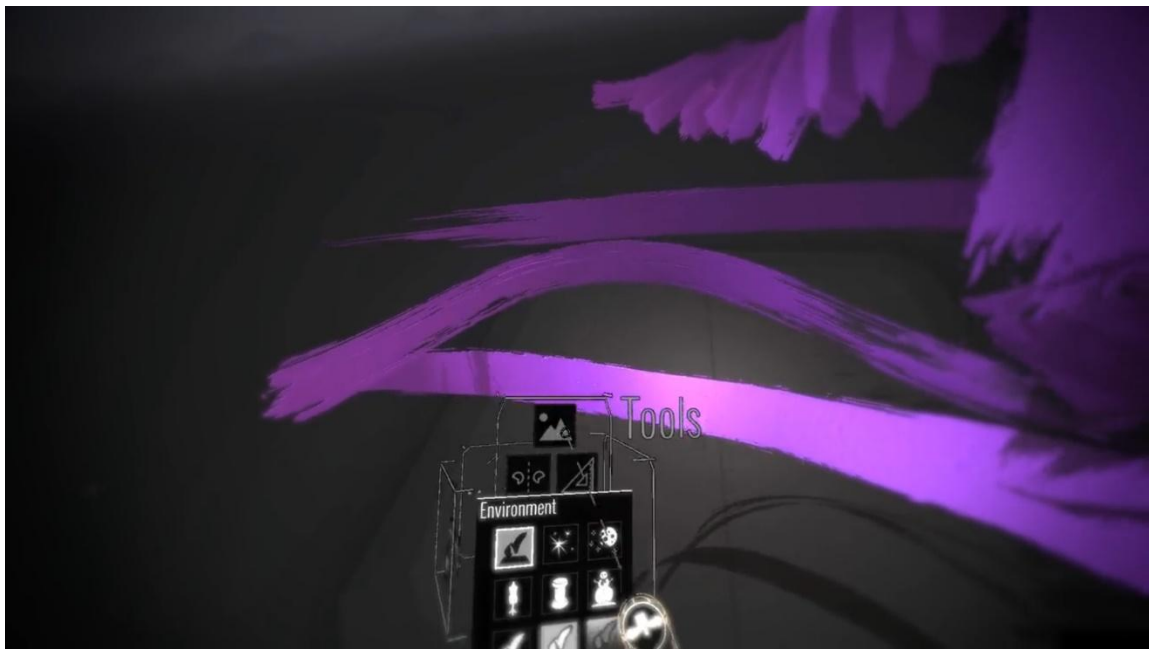
Kuva 2. Pakastin muuttuu kassakaapiksi säädintä vääntämällä. [Jobsimulator-game.com]

Sen sijaan, että yritettäisiin tarkkaan matkia todellista maailmaa, voidaan tosielämästä lainata vain yksittäisiä esineitä tai ideoita ja soveltaa niitä virtuaaliympäristöön ja kyseiseen sovellukseen paremmin soveltuvaksi [Bowman et al. 2004]. Kun virtuaalimaailmassa ei tarvitse noudattaa fysiikan lakeja, voidaan pieneenkin tilaan mahduttaa paljon. Esimerkiksi Job Simulator -pelissä käyttäjä voi käsimetaforalla säätimeen tarttumalla ja sitä kääntämällä vaihtaa pakastimen jääkaapiksi tai kassakaapiksi (kuva 2). Fysiikan lakien pätemättömyys mahdollistaa myös sen, että todellisuudessa raskaidenkin käyttöliittymien, kuten vipujen ja salpojen, käyttö onnistuu vastuksen puutteen vuoksi myös sellaisilta käyttäjiltä, jotka liikkuvat huonommin.

Virtuaaliympäristöön voidaan kehitellä myös täysin uudenlaisia käyttöliittymäelementtejä. Patterson [2016] esittelee kolmiulotteisen kuplasäätimen, joka muuttaa kokoa samaan tahtiin kuin se muuttaa haluttua arvoa. Tällä säätimellä voidaan asettaa ja esittää tietyllä asteikolla olevia arvoja aivan kuten liukusäätimellä. Arvoasteikon yläraja voidaan esittää esimerkiksi kuplan ympärillä olevalla toisella, läpinäkyvällä, kuplalla. Kuplasäätimen etu perinteiseen kaksi- tai kolmiulotteiseen liukusäätimeen verrattuna on se, että symmetrisen muotonsa vuoksi se näyttää samalta kaikista suunnista.

4.3. Käsi- tai ohjainmetaforaan sidotut käyttöliittymät

Käyttöliittymän pitäisi tehdä käyttäjälle näkyväksi ja selväksi, mitä hän voi milloinkin tehdä [Dix 2009], mutta graafisilla käyttöliittymäelementeillä ei haluta estää käyttäjää näkemästä ympäröivää virtuaalista maailmaa. Varsinkin kun näkökenttä on vielä nykyisillä HMD-laitteilla melko rajallinen, halutaan valikot ja painikkeet pitää piilossa silloin, kun niitä ei tarvita. Yksi tässä hyväksi koettu menetelmä on kiinnittää graafinen käyttöliittymä esimerkiksi käyttäjän käsimetaforaan. Käsimetafora toimii objektien valinnassa ja käsittelyssä aivan kuten edellä on kuvattu, mutta siihen on lisäksi kiinnitetty graafinen käyttöliittymä. Käyttöliittymän saa näkyviin esimerkiksi katsomalla kämmentänsä tai nostamalla ranteensa eteensä aivan kuin katsoisi rannekelloa. Vuorovaikutus käyttöliittymän kanssa tapahtuu vapaana olevaa kättä käyttäen tai katseella ray-casting -menetelmällä.



Kuva 3. Tilt Brushissa toinen ohjainmetafora muuttuu tarpeen mukaan työkaluvalikoksi. [Tiltbrush.com]

Toinen vaihtoehto on kytkeä graafinen käyttöliittymä langattomaan käsiohjaimeen. Esimerkiksi HTC Viven käsiohjainta käytettäessä ohjainmetaforaa voidaan muokata tarpeen mukaan. Googlen Tilt Brush -sovelluksessa toinen ohjainmetafora muuttuu kädessä pidettäväksi työkaluvalikoksi (kuvassa 3). Käyttäjä voi selata erilaisten työkalujen ja asetusten välillä joko kääntelemällä kädessään olevaa käsiohjainta tai pyörittämällä työkaluvalikkoa Viven käsiohjaimen kosketuslevyä käyttäen. Eri puolilla työkaluvalikkoa käyttäjä voi muun muassa vaihtaa siveltimen tyyppiä tai käytettyä väriä. Vuorovaikutus tapahtuu ray-casting -menetelmällä toista ohjainmetaforaa osoittimena käyttäen.

Jos käytössä on liikeohjaimien sijaan esimerkiksi Leap Motionilla tai datakäsineillä tapahtuva käsien seuranta, voidaan myös yksittäisiin sormiin kiinnittää käyttöliittymäelementtejä. Yksi tällainen on Bowmanin ja Wingraven [2001] esittelemä TULIP-valikko. Tässä molemmat kädet toimivat valikkoina, joissa kummassakin on kerrallaan enintään neljä valittavaa kohdetta. Kohteiden valinta tapahtuu viemällä saman käden peukalo ja haluttu sormi yhteen. Jos kohteita on enemmän kuin kahdeksan, voidaan esimerkiksi pikkurilliin yhdistää valinta, joka vaihtaa näkyvissä olevat valintakohteet. Bowmanin ja Wingraven mukaan tällainen valikko on erittäin helppo ja nopea käyttää. Moniin muihin VR-ympäristöissä käytettyihin käyttöliittymiin verrattuna tämän menetelmän etu on se, että käyttäjän opittua valintakohteita vastaavat sormet, voidaan valintoja tehdä helposti myös näkökentän ulkopuolella valikkoon katsomatta.

5. Yhteenveto

Virtuaalisissa, kolmiulotteisissa ympäristöissä tapahtuva vuorovaikutus poikkeaa melko paljon siitä, mihin käyttäjät ovat kaksiulotteisissa työpöytäympäristöissä tottuneet. Soveltuvat käyttöliittymäobjektien valinta- ja käsittelymenetelmät riippuvat täysin käytetyistä syöttölaitteista sekä käyttöliittymän graafisesta toteutuksesta. Useimmat ratkaisut yhdistelevät erilaisia vuorovaikutusmenetelmiä, kuten ray-castingia ja käsien tai käsiohjainten seurantaan perustuvaa suoraa vuorovaikutusta.

VR-sovelluksissa voidaan käyttää hyvin monenlaisia graafisia käyttöliittymiä. Niin kaksi- kuin kolmiulotteisiakin GUI-elementtejä voidaan kiinnittää tiettyyn sijaintiin, käyttäjän näkymään, erilaisiin virtuaaliympäristössä oleviin pintoihin ja objekteihin sekä käsi- ja ohjainmetaforiin. Jotkin 2D-käyttöliittymistä tutut elementit voidaan tuoda sellaisenaan tai kolmiulotteiseksi muutettuna myös virtuaaliympäristöön. Myös tosielämästä tuttuja esineitä ja ideoita kopioimalla voidaan saada aikaiseksi helposti ymmärrettäviä käyttöliittymiä. Monissa tapauksissa paras ratkaisu on kuitenkin suunnitella täysin uusia, 3D-ympäristöön tarkoitettuja käyttöliittymäelementtejä.

Viiteluettelo

Atienza, Rowel, Blonna, Ryan, Saludares, Maria I., Casimiro, Joel and Fuentes, Vivencio. 2016. Interaction techniques using head gaze for virtual reality. In: *Proc. of the Region 10 Symposium (TENSYMP)*, 110-114.

Bowman, Doug A. and Hodges, Larry F. 1997. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In: *Proc. of the 1997 Symposium on Interactive 3D Graphics*, 35-38.

Bowman, Doug, Kruijff, Ernst, LaViola, Joseph J Jr, and Poupyrev, Ivan. 2004. *3D User Interfaces: Theory and Practice*. Addison-Wesley.

Bowman, Doug, Wingrave, Chadwick. 2001. Design and evaluation of menu systems for immersive virtual environments. In: *Proc. of IEEE Virtual Reality 2001*, 149-156.

Dix, Alan 2009. *Human-Computer Interaction.* Springer.

Feiner, Steven, MacIntyre, Blair, Haupt, Marcus AND Solomon, Eliot. 1993. Windows on the world: 2D windows for 3D augmented reality. In: *Proc. of the 6th annual ACM symposium on User interface software and technology*, 145-155.

Jobsimulatorgame.com. 2017. *http://jobsimulatorgame.com/press/.* Checked 23.05.2017.

Lindeman, Robert, Sibert, John and Hahn, James. 1999. Hand-held windows: towards effective 2D interaction in immersive virtual environments. In: *Proc. of IEEE Virtual Reality*, 205-212.

Mine, Mark. 1995. Virtual environment interaction techniques. UNC Chapel Hill Computer Science Technical Report TR95-018.

Patterson, Dale. 2016. 3D spirals, bubbles and sliders: setting range values in multi-user 3D environments. In: *Proc. of the Australasian Computer Science Week Multiconference*, 49-59.

Poupyrev, Ivan, Billinghurst, Mark, Weghorst, Suzanne and Ichikawa, Tadao. 1996. The Go-Go interaction technique: non-linear mapping for direct manipulation in VR. In: *Proc. of the 9th Annual ACM Symposium on User Interface Software and Technology*, 79-80.

Poupyrev, Ivan and Ichikawa, Tadao. 1999. Manipulating objects in virtual worlds: Categorization and empirical evaluation of interaction techniques. *Journal of Visual Languages & Computing 10*, 19-35.

Sutherland, Ivan E. 1968. A head-mounted three dimensional display. In: *Proc. of the Fall Joint Computer Conference*, 757-764.

Tiltbrush.com. 2017. *https://www.tiltbrush.com/.* Checked 22.05.2017.

Satunnaismetsä

Eero Mustalahti

Tiivistelmä.
Satunnaismetsä on koneoppimismenetelmä, jota käytetään datan tilastolliseen luokitteluun ja regressioanalyysiin. Se toimii luomalla satunnaisella tavalla useita päätöspuita, jolloin luokittelun tapauksessa puut äänestävät uuden havainnon saamasta luokasta ja regressioanalyysissä puiden antamien arvojen keskiarvo annetaan uuden havainnon lopulliseksi arvoksi. Tässä tutkielmassa esitellään satunnaismetsä-algoritmin toimintaa näyttämällä, miten satunnaismetsä tuottaa päätöspuut saamansa datan perusteella. Lisäksi havainnollistetaan algoritmin tuottaman valmiin satunnaismetsän käyttöä uusien havaintojen luokittelussa tai regressioanalyysissä. Algoritmin toimintaan perehdytetään yleisen tason selityksen ja yksityiskohtaisemman perehdytyksen lisäksi esimerkin avulla. Myös satunnaismetsän tarkkuuden testaamista ennustevirheen avulla käsitellään, kuten myös sen hyödyntämistä satunnaismetsän parantelemisessa. Satunnaismetsään oleellisesti liittyvä päätöspuun käsite esitellään ennen varsinaista satunnaismetsä-algoritmia.

Avainsanat ja -sanonnat: Satunnaismetsä, koneoppimisalgoritmi, päätöspuu, CART, luokittelu, regressioanalyysi.

1. Johdanto

Koneoppimisalgoritmi (machine learning algorithm), lyhyemmin *oppimisalgoritmi*, on algoritmi, joka saamansa *opetusdatan* (training data) perusteella oppii toimimaan halutulla tavalla uuden, ennestään havainnoimattoman datan parissa. Algoritmi siis oppii opetusaineiston perusteella, miten sen tulee toimia uuden aineiston kanssa.

Koneoppimisalgoritmeilla on erilaisia lähestymistapoja siihen, miten ne oppivat toimimaan saamansa datan perusteella. Yksi tapa on *päätöspuuoppiminen* (decision tree learning). Tähän koneoppimisalgoritmien kategoriaan kuuluu *satunnaismetsä* (random forest), jota käytetään *luokitteluun* (classification) ja *regressioanalyysiin* (regression analysis).

Nykyisenlaisen satunnaismetsän on kehittänyt Leo Breiman [2001]. Jo Breimania ennen eräänlaisen algoritmin *satunnaiselle päätösmetsälle* (random decision forest) oli kehittänyt Tim Kam Ho [1995], joka käytti satunnaisen päätösmetsän luomisessa *satunnainen aliavaruus -menetelmää* (random subspace method) [Ho, 1998]. Breimanin satunnaismetsä-algoritmi on laajennettu versio tästä algoritmista, jossa käytetään myös Breimanin keksimää *bootstrap-aggregointia* (bootstrap

aggregating), lyhyemmin *bagging*. Satunnaismetsä-algoritmia tullaan käsittelemään tässä tutkielmassa Breimanin [2001] alkuperäisen esityksen perusteella, mutta myös Friedmanin ja muiden [2009] kirjoittamaa satunnaismetsän kuvausta tullaan hyödyntämään.

Satunnaismetsän toimintatapa on muodostaa opetusdatan perusteella useita, satunnaisella tavalla muodostettuja päätöspuita, jotka täten yhdessä muodostavat ”satunnaisen metsän”. Tämä päätöspuiden kokoelma on malli, jonka avulla voidaan ennustaa uuden datan arvoja luokittelulla tai regressiolla. Luokiteltaessa uusi havainto ajetaan se kaikkien muodostettujen päätöspuiden läpi. Lopuksi suoritetaan äänestys havainnon saamasta luokasta, eli kukin päätöspuu määrittää havainnolle luokan, ja yleisin päätöspuiden määrittämä luokka annetaan havainnon lopulliseksi luokaksi. Myös regressioanalyysissä uusi havainto ajetaan kaikkien puiden läpi, jonka jälkeen kaikkien puiden antamien arvojen keskiarvo lasketaan, ja tämä keskiarvo on lopullinen *ennuste* (prediction) havainnon arvosta.

Ennen satunnaismetsä-algoritmin toimintaan perehtymistä esitellään sen kannalta keskeinen käsite *päätöspuu* (decision tree) luvussa 2. Tämän jälkeen kerrotaan satunnaismetsän toiminnasta yleisellä tasolla luvussa 3 ja algoritmin toimintaan perehdytään tarkemmin luvussa 4. Luvussa 5 valotetaan satunnaismetsän toimintaa konkreettisen esimerkin avulla. Luvussa 6 kerrotaan, miten satunnaismetsän ennustetarkkuus on mahdollista määrittää, ja millä tavoin tätä tietoa voidaan hyödyntää satunnaismetsän parantamisessa. Lopuksi kerrotaan, miksi ja missä yhteyksissä satunnaismetsää on sovellettu.

2. Päätöspuu

Päätöspuu on keskeinen käsite satunnaismetsässä. Tässä luvussa tarkastellaan sellaisia päätöspuita, joita algoritmissa käytetään.

2.1. Päätöspuun käyttötarkoitus

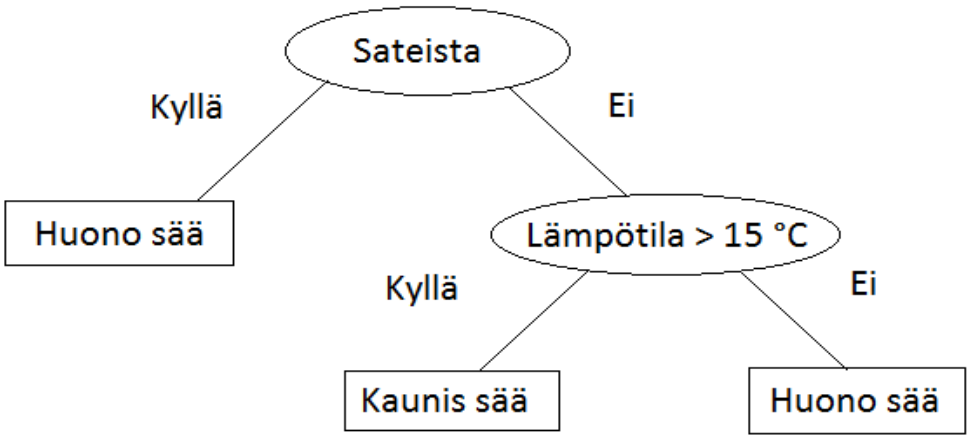
Päätöspuuoppiminen (decision tree learning) on koneoppimisessa yksi tapa, jolla voidaan oppia tekemään päätelmiä datasta. Päätöspuu on *ennustava malli* (predictive model), jonka avulla voidaan datan ominaisuuksien perusteella ennustaa *kohdemuuttujan* (target variable) *kohdearvo* (target value). Päätöspuun avulla voidaan siis päätellä havainnon yhden ominaisuuden arvo havainnon muiden ominaisuuksien perusteella. Selittävät muuttujat eli ominaisuudet, joiden avulla selitetään kohdemuuttujan arvoa, ovat *riippumattomia muuttujia* (independent variable). Päätöspuissa näitä ominaisuuksia voidaan kutsua myös *syötemuuttujiksi* (input variable).

Selitettävä muuttuja eli kohdemuuttuja, jonka arvoa päätöspuun avulla selvitetään, on *riippuva muuttuja* (dependent variable). Riippuen riippuvan muuttujan tyypistä, päätöspuuta voidaan kutsua *luokittelupuuksi* (classification tree) tai *regressiopuuksi* (regression tree). Jos muuttuja on kategorinen eli sellainen, joka jakaa tarkasteltavat havainnot luokkiin, on kyseessä luokittelupuu. Kategorisia muuttujia voivat olla esimerkiksi opiskelijan tutkinto-ohjelma tai kansanedustajan puolue. Jos muuttuja on kvantitatiivinen, jatkuva-arvoinen reaaliluku, niin kyseessä on regressiopuu. Tällaisia muuttujia voivat olla esimerkiksi asunnon hinta tai henkilön pituus.

Puutyyppejä voidaan yhteisesti kutsua *luokittelu- ja regressiopuiksi* (Classification and Regression Trees), lyhyemmin CART, joka on siis sateenvarjotermi näille kahdelle puutyypille. Tässä tutkielmassa päätöspuihin viitatessa tarkoitetaan nimenomaan CART-tyylisiä puita.

2.2. Rakenne ja käyttö

Kuva 1 on esimerkki päätöspuusta. Päätöspuu koostuu sisäsolmuista, lehtisolmuista ja kaarista. Sisäsolmut ovat kuvan esimerkissä ellipsejä, lehtisolmut suorakaiteita ja kaaret solmujen välisiä viivoja. Jokaisella sisäsolmulla on kaksi lapsisolmua, joten päätöspuu on *binääripuu* (binary tree).



Kuva 1. Esimerkki päätöspuusta.

Kukin sisäsolmu vastaa syötemuuttujaa, jossa testataan kyseisen syötemuuttujan arvoa. Sisäsolmusta lähtevät kaaret kuvaavat syötemuuttujassa tapahtuvan testauksen totuusarvoa. Yhtä kaarta seurataan testauksen palauttaessa toden, toista testauksen palauttaessa epätoden. Kaaret voivat päätyä sisäsolmuun, jossa jälleen testataan jonkin muuttujan arvoa, tai lehtisolmuun.

Lehtisolmut vastaavat kohdemuuttujan arvoja, joihin päädytään syötemuuttujien arvojen perusteella. Polku päätöspuun juuresta lehtisolmuun kertoo, millä syötemuuttujien arvoilla päädytään kyseiseen kohdemuuttujan arvoon. Aja-

malla uusi havainto päätöspuun läpi eli testaamalla sen ominaisuuksia puun sisäsolmuissa ja kulkemalla kaaria tulosten mukaan päädytään siis johonkin ennusteeseen havainnon kohdearvosta.

2.3. Päätöspuun rakentaminen

Päätöspuun avulla voidaan päätellä uuden havainnon kohdearvo, mutta ennen tätä on itse päätöspuu muodostettava. Päätöspuun muodostamiseen tarvitaan valmis aineisto. Aineistossa on havaintojen muuttujien arvojen, joiden avulla halutaan muodostettavassa päätöspuussa arvioida kohdemuuttujan arvo, lisäksi myös itse kohdemuuttujan arvo. Tämän aineiston avulla opitaan, miten ominaisuudet vaikuttavat kohdemuuttujan arvoon ja täten päätöspuu rakentuu sen mukaan, miten ominaisuuksien erilaiset arvot muuttavat kohdemuuttujan arvoa. Tämä aineisto toimii siis päätöspuulle opetusdatana.

Päätöspuu rakennetaan valitsemalla kunkin solmun kohdalla syötemuuttuja ja sopiva jakokohta tässä muuttujassa. Solmun jakaminen tarkoittaa sitä, että solmulle luodaan kaksi uutta lapsisolmua. Jakaminen tapahtuu pilkkomalla opetusaineistoa *osajoukoiksi* (subset). Kun solmu jaetaan, jaetaan myös solmussa olleet havainnot kahteen osaan lapsisolmujen kesken. Jakaminen alkaa siis koko opetusaineiston jakamisella kahteen osajoukkoon, ja jatkuu rekursiivisesti näiden osajoukkojen jakamisella. Syötemuuttujien arvoista valitaan jakajaksi se, joka parhaiten jakaa havaintojen kohdearvot eri joukkoihin. Tämä valinta tehdään jokaisessa solmussa erityisen funktion avulla, joka laskee jokaisen mahdollisen jakokohdan kustannuksen. Luokittelussa kustannus lasketaan *gini-kertoimella* (Gini index), ja regressiossa *jäännösneliösummalla* (sum of squared errors) [Breiman *et al.* 1984]. Jakokohdaksi valitaan kohta, jolla on paras eli alhaisin mahdollinen kustannus.

Jakamista jatketaan rekursiivisesti lehtisolmuissa niin kauan, kunnes todetaan, että solmun jakaminen ei tuottaisi enää lisäarvoa kohdearvon ennustamiseen tai kun jakamisen yhteydessä muodostuvassa osajoukossa olisi liian vähän havaintoja jäljellä. Lopulta päätöspuu on muodostettu ja valmis uusien havaintojen kohdearvojen ennustamiseen.

3. Satunnaismetsän toiminta

Satunnaismetsä on menetelmä, jonka avulla ennustetaan uusien havaintojen kohdearvoja. Satunnaismetsän tuloksena on joukko päätöspuita, jotka ovat muodostettu satunnaisella tavalla ja ovat täten hyvin riippumattomia toisistaan. Tämän päätöspuiden suuren määrän, satunnaisuuden ja toisistaan riippumattomuuden vuoksi on oletettavissa, että useimmat päätöspuut ennustavat havaintojen kohdearvot oikein.

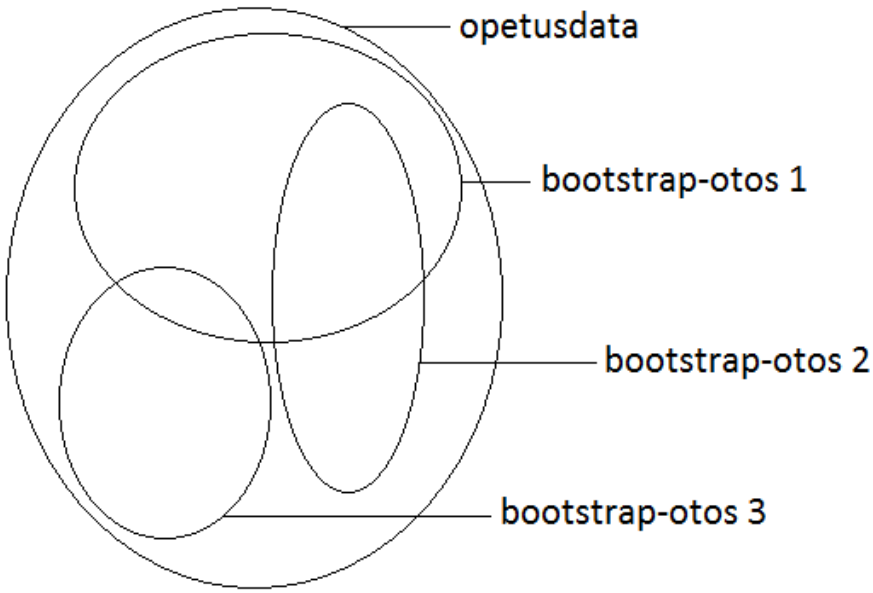
Havainnollistetaan satunnaismetsän toimintaa esittämällä, miten algoritmi rakentaa joukon päätöspuita käyttäen datan ja muuttujien satunnaista valitsemista. Selitetään myös, miten valmista satunnaismetsää käytetään luokittelun tai regressioanalyysin tekemiseen. Satunnaismetsän toiminnan voi jakaa seuraaviin vaiheisiin:

- Satunnaismetsän rakentamiseen
    - Datan satunnainen valitseminen
    - Muuttujien satunnainen valitseminen
- Valmiin satunnaismetsän käyttämiseen.

3.1. Datan satunnainen valitseminen

Satunnaismetsä-algoritmi aloittaa jokaisen päätöspuun rakentamisen vaiheella, jossa muodostetaan *bootstrap-otos*. Tätä prosessia, jossa päätöspuita rakennetaan näitä otoksia tehden ja hyödyntäen, kutsutaan bootstrap-aggregoinniksi.

Bootstrap-otos muodostetaan opetusdatasta. Opetusdatasta tehdyt otokset ovat yhtä suuria kuin opetusdata itse. Niissä kaikissa on siis yhtä paljon havaintoja kuin alkuperäisessä opetusaineistossa. Opetusdatasta otokseen mukaan otettavat havainnot valitaan satunnaisesti siten, että jokaisella havainnolla on yhtä suuri todennäköisyys tulla valituksi. Otokseen valittu havainto palautetaan takaisin opetusdataan. Täten sama havainto voi tulla useaan kertaan arvotuksi samaan otokseen, ja myös muiden otosten valitsemaksi. Otosten kesken voi siis ilmetä päällekkäisyyttä. Kuva 2 havainnollistaa opetusdatasta tehtyjä bootstrap-otoksia. Kunkin otoksen avulla rakennetaan yksi päätöspuu, joten otoksia tulee muodostettua yhtä paljon kuin päätöspuita.



Kuva 2. Bootstrap-otosten muodostaminen opetusdatasta.

3.2. Muuttujien satunnainen valitseminen

Yksittäisen bootstrap-otoksen muodostamisen jälkeen kasvatetaan päätöspuu. Data, jota käytetään päätöspuun kasvattamiseen, on siis yhdestä bootstrap-otoksesta. Päätöspuun rakentaminen noudattaa normaalia CART-menetelmää sillä erotuksella, että jokaisen solmun kohdalla, jossa päätetään, minkä muuttujan ja muuttujan arvon kohdalla jako tehdään, ei valita parasta mahdollista jaon tekevää muuttujan arvoa kaikkien aineiston muuttujien arvojen kesken, vaan muodostetaan osajoukko, johon valitaan satunnaisesti vain osa muuttujista. Tästä osajoukosta valitaan paras mahdollinen muuttuja ja muuttujan arvo jaon tekemiseen.

Tämä vaihe muistuttaa edellistä datan satunnaista valitsemisvaihetta sikäli, että kuten siinä valittiin satunnaisella tavalla koko aineistosta havaintoja muodostettavaan otokseen, myös tässä vaiheessa valitaan satunnaisella tavalla kaikkien muuttujien joukosta muuttujia osajoukkoon. Keskeisiä eroja on kuitenkin kaksi. Bootstrap-otoksessa on yhtä paljon havaintoja kuin aineistossa, josta se muodostetaan. Muuttujien joukko, joka arvotaan ja josta valitaan paras jaon tekevä muuttuja, on kuitenkin aina pienempi kuin kaikkien muuttujien joukko. Lisäksi bootstrap-otoksessa sama opetusaineiston havainto voi esiintyä useaan kertaan. Arvottuun muuttujien joukkoon ei valita samaa muuttujaa useaan kertaan.

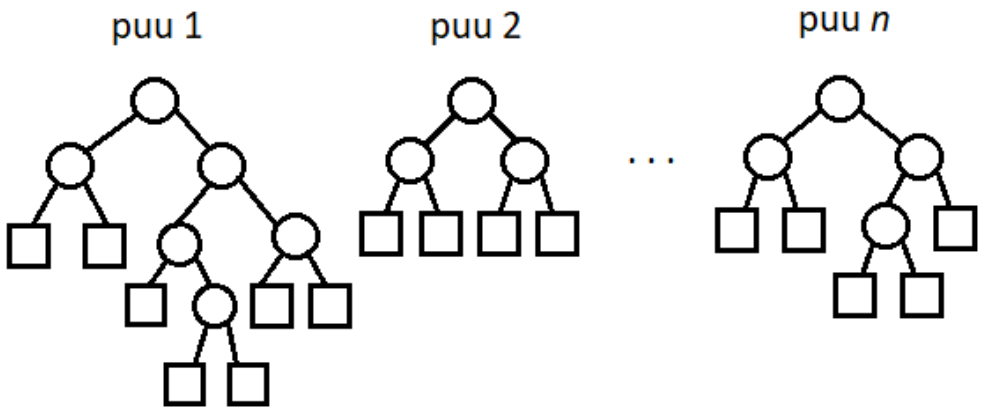
Tin Kam Ho [1998] nimesi tämän muuttujien satunnaisen valitsemisen satunnainen aliavaruus -menetelmäksi. Breiman [2001] itse tosin käyttää tästä nimitystä *satunnainen syötteiden valinta* (random input selection) soveltaessaan tätä satunnaismetsä-algoritmissaan.

3.3. Satunnaismetsän käyttäminen

Satunnaismetsä-algoritmin tuloksena on syntynyt satunnaismetsä-malli (kuva 3), jota voidaan käyttää uusien havaintojen kohdearvojen ennustamiseen. Riippuen kohdemuuttujan tyypistä ennustamisessa voi olla kyse luokittelusta tai regressioanalyysistä.

Sekä luokittelussa että regressioanalyysissä uusi havainto ajetaan kaikkien satunnaismetsän puiden läpi, jolloin jokainen puu antaa oman ennusteensa kohdearvosta. Kategorisen kohdemuuttujan tapauksessa eli luokittelussa päätöspuut äänestävät tämän jälkeen oikeasta kohdearvosta. Havainnon lopulliseksi kohdearvoksi määritellään se arvo, jonka päätöspuut antoivat useimmin. Kvan-

titatiivisten muuttujien tapauksessa eli regressioanalyysissa taasen puiden antamista kohdearvoista lasketaan keskiarvo, joka määritellään havainnon lopulliseksi kohdearvoksi.



Kuva 3. Tyylitelty esimerkki satunnaismetsä-mallista.

4. Algoritmi

Seuraava Friedmanin ja muiden [2009] pseudokoodi kuvaa satunnaismetsä-algoritmin toimintaa:

1. b = 1:stä B:hen tee seuraavaa:
    a. Muodosta n kokoinen bootstrap-otos opetusdatasta.
    b. Kasvata satunnaismetsän puu T_b bootstrap-otoksen datasta toistamalla rekursiivisesti seuraavia kohtia jokaiselle lehtisolmulle, kunnes vähimmäiskoko solmulle m_{min} on saavutettu.
        i. Valitse F muuttujaa satunnaisesti p:stä muuttujasta.
        ii. Valitse paras muuttuja/jakokohta F:n joukosta.
        iii. Jaa solmu kahteen lapsisolmuun.
2. Tulosta kokoelma puita $\{T_b\}_1^B$.

Kohdassa 1 algoritmi rakentaa satunnaismetsän puut ja kohdassa 2 tulostaa puut, jolloin satunnaismetsä-malli on käytettävissä. Kohdan 1 a-vaiheen menetelmää kutsutaan bootstrap-aggregoimiseksi, jossa opetusdatasta muodostetaan satunnainen bootstrap-otos. Kohdan 1 b-vaiheessa tästä otoksesta muodostetaan päätöspuu käyttäen satunnainen aliavaruus -menetelmää.

Perehdytään tässä luvussa algoritmin toimintaan tarkemmin näissä vaiheissa. Lisäksi kerrotaan, miten algoritmin tuloksena muodostetun satunnaismetsä-mallin avulla määritellään kohdemuuttujan arvo uusille havainnoille.

4.1. Bootstrap-aggregointi

Jokaisen päätöspuun rakentaminen alkaa muodostamalla bootstrap-otos. Koska kutakin otosta vasten rakennetaan yksi päätöspuu, on muodostettavien otoksien määrä B yhtä suuri kuin muodostettavien puiden määrä.

Otokset muodostetaan opetusaineistosta

$$X = (x_1, \dots, x_n),$$

joka on kokoa n. Yksittäinen havainto on x_i ja aineiston viimeinen n:s havainto on x_n. Opetusaineistosta muodostettu bootstrap-otos

$$X_b^* = (x_1^*, \dots, x_n^*), \text{ missä } b = 1, \dots, B,$$

on myös kokoa n. Otoksen havainnot on arvottu käyttäen yksinkertaista satunnaisotantaa palauttaen. Havainnot on siis poimittu aineistosta arpomalla siten, että jokaisella havainnolla on yhtä suuri todennäköisyys tulla valituksi. Jokaisen poiminnan jälkeen havainnon valituksi tulemisen todennäköisyys pysyy samana, koska arvottu havainto palautetaan takaisin aineistoon. Palauttamisen vuoksi aineistossa vain kerran esiintyvät havainnot voivat tulla useaan kertaan valituiksi otokseen.

4.2. Satunnainen aliavaruus -menetelmä

Kunkin bootstrap-otoksen muodostamisen jälkeen, ennen seuraavan bootstrap-otoksen muodostamista, rakennetaan päätöspuu juuri muodostettua bootstrap-otosta vasten. Otos siis toimii rakennettavan päätöspuun opetusaineistona.

Opetusaineisto, josta bootstrap-otokset muodostetaan, on muotoa

$$(y, Y) = (y_1, y_2, \dots, y_p, Y).$$

Riippuva muuttuja Y on kohdemuuttuja, jonka arvoa vektorin y muuttujat y_i, joita on p kappaletta, pyrkivät selittämään. Kasvatettava päätöspuu tulee arvioimaan Y:n arvoa joidenkin ominaisuuksien y_i perusteella.

Päätöspuun kasvattaminen noudattaa normaalia CART-menetelmää sillä erotuksella, että solmussa parasta jaon tekevää jonkin muuttujan arvoa ei valita kaikkien muuttujien joukosta. Sen sijaan muuttujista arvotaan osajoukko, jonka koko on F. Tästä osajoukosta valitaan edelleen se muuttujan arvo, joka tuottaa parhaan jaon. Osajoukkoon siis valitaan muuttujat arpomalla, mutta toisin kuin bootstrap-otoksen muodostamisessa, muuttujan poimimisen jälkeen sitä ei palauteta takaisin. Muuttuja ei siis voi tulla useaan kertaan valituksi osajoukkoon. Kuitenkin osajoukon muuttujat palautetaan takaisin, kun solmu on jaettu osajoukosta valitun parhaan jaon tekevän muuttujan arvon perusteella. Näin ollen

seuraavassa solmussa muodostettavan osajoukon muuttujat valitaan jälleen kaikkien muuttujien joukosta.

Muuttujien osajoukon koko *F* on vakio, joka pysyy samana koko algoritmin suorituksen ajan. Ennen algoritmin suoritusta se on kuitenkin määritettävissä. Yleensä *F*:lle määritettävä arvo on *p*:n ominaisuuden tapauksessa $\sqrt{p}$, joskus niinkin alhainen kuin 1 [Friedman *et al.* 2009]. Kuitenkin aina on $F < p$.

4.3. Uuden havainnon kohdearvon määrittäminen

Kun *B* kappaletta päätöspuita on valmistettu, on satunnaismetsä-malli valmiina käytettäväksi. Menettely, jolla uuden havainnon x' kohdemuuttujan arvo määritellään, riippuu kohdemuuttujan tyypistä. Kategorisen kohdemuuttujan eli luokittelun tapauksessa sovelletaan äänestystä, kvantitatiivisessa kohdemuuttujassa eli regressioanalyysissä ennusteiden keskiarvoa. Molemmissa tapauksissa x' ajetaan ensin kaikkien satunnaismetsän päätöspuiden läpi.

Luokittelussa x':n kohdemuuttujan arvoksi eli luokaksi määritellään päätöspuiden antamien luokkien moodi, eli se luokka, joka esiintyy useimmin päätöspuiden antamien luokkien joukossa. Voidaan siis sanoa, että päätöspuut ”äänestävät” oikeasta luokasta. Funktio

$$C_{rf}^{B}(x) = majority\ vote\ \{C_b(x)\}_1^B$$

kuvaa tätä menetelmää $C_b(x)$ ollessa *b*:nen satunnaismetsän puun tarjoama luokka.

Regressioanalyysissä x':n kohdemuuttujan arvo saadaan laskemalla päätöspuiden antamien ennusteiden keskiarvo. Funktio

$$f_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

kuvaa menetelmän $T_b(x)$ ollessa *b*:nen satunnaismetsän puun antama arvo.

5. Esimerkki satunnaismetsän toiminnasta

Havainnollistetaan satunnaismetsän toimintaa esimerkin avulla. Taulukossa 4 on kuvitteellinen aineisto, jossa on havaintoja potilaista ja kustakin potilaasta mitattujen ominaisuuksien, kuten ruumiinlämmön ja verenpaineen, arvoja. Potilaiden määrä on *n*, ja erilaisia mitattuja ominaisuuksia on useita, *p* kappaletta. Näiden lisäksi taulukkoon on merkitty potilaalle annettu lääketieteellinen diagnoosi sairaudesta. Tällaista aineistoa voitaisiin haluta käyttää apuna diagnoosin ennustamisessa uusille potilaille, joiden ominaisuudet on mitattu, mutta joiden diagnoosia ei ole määritelty. Sovelletaan satunnaismetsää tähän tehtävään.

Koska tässä tilanteessa kohdemuuttuja diagnoosi on tyypiltään kategorinen muuttuja, on kyse luokittelusta.

	1	*2*	*...*	*p*	
	Ruumiinlämpö	Verenpaine	...	Hemoglobiini	DIAGNOOSI
x_1	37,8	Kohonnut	...	150	Influenssa
x_2	37,1	Normaali	...	160	Keuhkokuume
x_3	36,0	Tyydyttävä	...	150	Influenssa
⋮	⋮	⋮	⋱	⋮	⋮
x_n	35,5	Normaali	...	140	Anemia

Taulukko 4. Esimerkki opetusaineistosta.

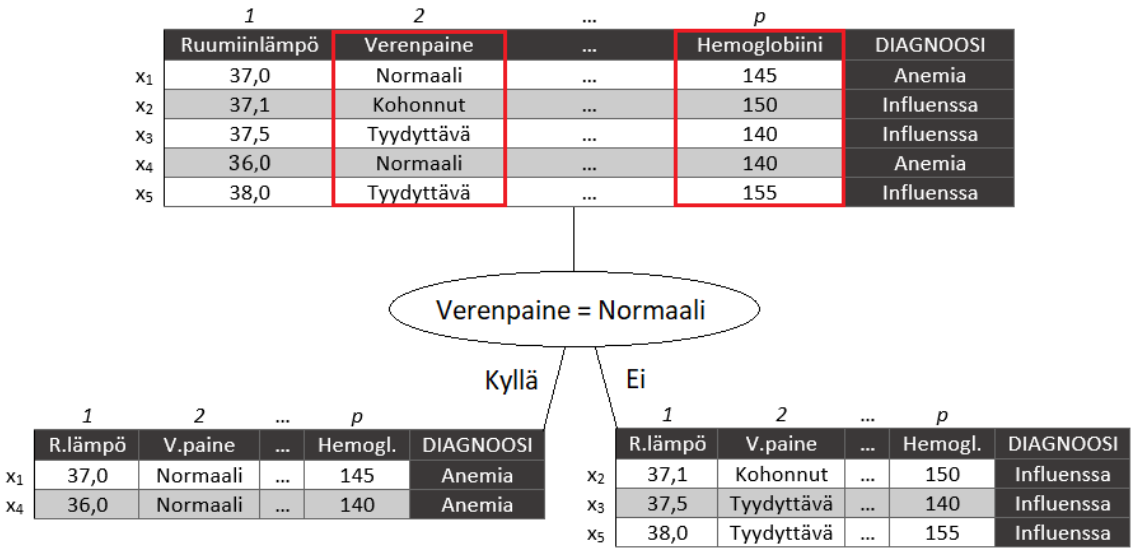
Potilaishavainnoista koostuva aineisto toimii opetusdatana, jonka avulla rakennetaan joukko päätöspuita, kaiken kaikkiaan B kappaletta. Kukin päätöspuu rakennetaan erikseen. Yksittäisen päätöspuun rakentaminen aloitetaan tekemällä bootstrap-otos opetusaineiston potilashavainnoista. Kuten luvussa 4 kerrottiin, bootstrap-otoksessa havaintoja on yhtä paljon kuin opetusaineistossa, mutta opetusaineistossa oleva havainto voi esiintyä otoksessa useaan kertaan, koska otos tehdään yksikertaisella satunnaisotannalla takaisin palauttaen. Taulukko 5 havainnollistaa mahdollista opetusaineistosta muodostettua bootstrap-otosta, jossa havainto x_5 on tullut valituksi ainakin kaksi kertaa.

	1	*2*	*...*	*p*	
	Ruumiinlämpö	Verenpaine	...	Hemoglobiini	DIAGNOOSI
x_5	35,0	Alhainen	...	162	Anemia
x_{12}	38,5	Kohonnut	...	161	Influenssa
x_5	35,0	Alhainen	...	162	Anemia
⋮	⋮	⋮	⋱	⋮	⋮
x_{41}	35,5	Normaali	...	150	Keuhkokuume

Taulukko 5. Opetusaineistosta muodostettu bootstrap-otos.

Bootstrap-otoksen potilashavaintojen avulla rakennetaan yksi päätöspuu. Rakentaminen aloitetaan jakamalla havainnot kahteen osajoukkoon. Ensin kaikkien muuttujien joukosta valitaan satunnaisesti tietty määrä muuttujia. Satunnaisesti valittujen muuttujien määrä F on vapaasti asetettava parametri, olkoon se tässä esimerkissä kaksi. Näistä kahdesta muuttujasta ja niiden arvoista jakofunktio valitsee parhaan mahdollisen jaon tekevän kohdan, jonka perusteella havainnot jaetaan kahteen osaan. Tätä jakamista jatketaan rekursiivisesti molemmissa muodostetuissa osajoukoissa, kunnes jakaminen ei tuota enää lisäarvoa ennustamiselle, tai jaettavassa osajoukossa ei ole tarvittavaa vähimmäismäärää m_{min} havaintoja. Tämä on myös vapaasti asetettava parametri, olkoon se tässä esimerkissä viisi.

Kuva 6 on yksi mahdollinen tilanne, jossa yhden jakamisen tuloksena muodostetun osajoukon havainnot jaetaan vielä kahteen osaan. Tämä on mahdollista, koska havaintoja on viisi eli vähimmäismäärä jakamisen suorittamiseksi. Kaikkien muuttujien joukosta on arvottu muuttujat ”verenpaine” ja ”hemoglobiini”. Näistä kahdesta muuttujasta ja niiden arvoista on jakofunktion avulla valittu jaon määrittäväksi kohdaksi muuttujan ”verenpaine” arvo ”normaali”. Potilashavainnot on siis jaettu kahteen osaan sen perusteella, onko niissä verenpaine normaali vai ei.

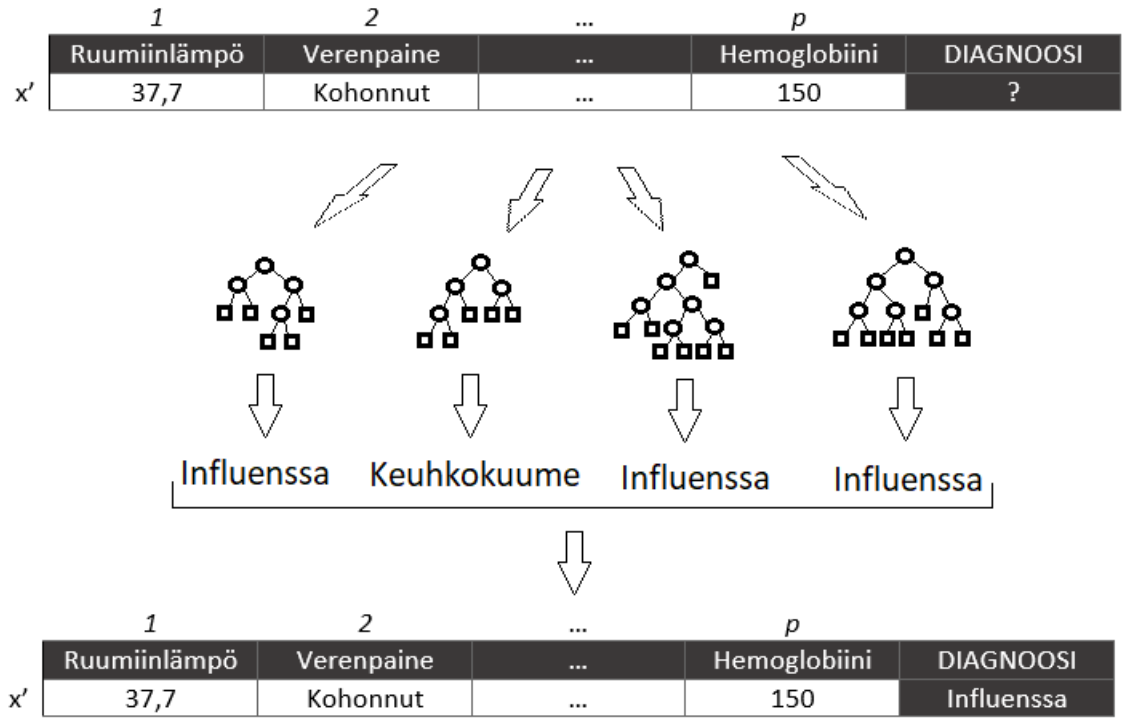


Kuva 6. Havaintojen jakaminen kahteen osaan.

Kun päätöspuita on rakennettu tarvittava määrä, voidaan muodostettua satunnaismetsää käyttää uusien potilaiden diagnoosien ennustamiseen. Kuva 7 esittää, kuinka uusi havainto potilaasta mitattuine ominaisuuksineen ajetaan kaikkien päätöspuiden läpi, jolloin kukin puu antaa oman luokkansa, ja yleisin puiden antama luokka asetetaan ennusteeksi potilaan diagnoosista. Tässä esimerkissä puita on vain neljä, mutta yleensä niitä muodostetaan ainakin satoja.

Esimerkissä satunnaismetsää käytettiin luokittelussa, mutta satunnaismetsän käyttäminen regressioanalyysissä ei eroa siitä paljoakaan. Oletetaan, että kategorisen diagnoosimuuttujan sijaan esimerkin kohdemuuttuja olisikin potilaan ikä, joka on kvantitatiivinen, jatkuva-arvoinen reaaliluku. Päätöspuiden rakentaminen tapahtuisi samalla tavalla kuin luokittelussa, erilaista jaon hinnan laskevaa jakofunktiota lukuun ottamatta. Satunnaismetsä-mallin käyttö kuitenkin

eroaa siten, että puiden antamien ikien keskiarvo lasketaan, ja tämä annetaan potilaan iän ennusteeksi.



Kuva 7. Uuden havainnon kohdearvon ennustaminen.

6. Satunnaismetsän ennustevirhe ja sen hyödyntäminen

Aivan kuten muutkin luokittelua ja regressioanalyysiä tekevät koneoppimismenetelmät, satunnaismetsä ei välttämättä ole virheetön ennusteissaan uusien havaintojen kohdearvoista. Siksi onkin oltava mittari, jonka avulla voidaan testata muodostetun satunnaismetsä-mallin luotettavuutta. Tähän tarkoitukseen käytetään *yleistysvirhettä* (generalization error), jota voidaan kutsua myös *ennustevirheeksi* (prediction error). Se ilmaisee, kuinka tarkasti algoritmi pystyy ennustamaan kohdearvot uusille havainnoille. Ennustevirheen riittävä minimoiminen on sitä mittaaville koneoppimisalgoritmeille tärkeä tavoite, sillä mitä alhaisempi ennustevirhe, sen tarkempi algoritmi.

Ennustevirheen määrittämiseen on erilaisia tapoja. Yksi tapa, jota myös Breiman [2001] on hyödyntänyt satunnaismetsien tarkkuuksien arvioimisissa, on jakaa saatavilla oleva, havaintojen kohdearvot sisältävä valmis aineisto opetusaineistoon ja testiaineistoon. Opetusaineistoa käytetään normaalisti ennustavan mallin muodostamiseen ja testiaineistolla testataan mallin tarkkuutta ajamalla testiaineiston havainnot mallin läpi ja katsomalla, kuinka monen testiaineiston havainnon kohdearvon malli ennusti oikein ja kuinka monen ei. Tämä tapa laskea ennustevirhe voi olla ongelmallinen muun muassa siksi, että jos saatavilla oleva aineisto on pieni jaettaessa se kahteen osaan opetus- ja testiaineistoon voi

opetusaineistosta jäädä pois ennustamisen kannalta merkittäviä havaintoja. Vastaavasti testiaineistosta voi jäädä testaamisen kannalta tarpeellisia havaintoja pois.

Satunnaismetsä pystyy kuitenkin arvioimaan omaa ennustevirhettään *out-of-bag -virheen* (out-of-bag error) avulla siten, että erillistä testiaineistoa ei tarvita. Tällöin satunnaismetsä käyttää ennustevirheen määrittämisessä samoja havaintoja, joita käytetään algoritmin opetusaineistossa. Tässä luvussa esitellään, miten satunnaismetsä laskee out-of-bag -virheen ja miten sitä voidaan hyödyntää satunnaismetsässä päätöspuiden tarpeellisen määrän ja muuttujien merkitsevyyksien arvioimisessa.

6.1. Out-of-bag -virhe

Out-of-bag -virheellä, jota voidaan kutsua myös *out-of-bag -estimaatiksi* (out-of-bag estimate), estimoidaan satunnaismetsän ennustevirhettä. Sen laskemisessa hyödynnetään bootstrap-aggregointia, jota käytetään jokaisen päätöspuun rakentamisen alussa. Breimanin [2001] mukaan tehtäessä bootstrap-otos opetusdatasta päätöspuun T_b rakentamista varten noin kolmasosa havainnoista jää pois otoksesta. Otoksen ulkopuolelle jääneitä havaintoja kutsutaan *out-of-bag -dataksi* tai *out-of-bag -havainnoiksi*. Näitä havaintoja ei siis käytetä puun T_b rakentamisessa. Kun päätöspuu T_b on rakennettu, kukin sen rakentamisessa käyttämätön, bootstrap-otoksen ulkopuolelle jäänyt havainto ajetaan puun läpi, jolloin puu antaa näille oman ennusteensa luokasta. Kun puita on rakennettu tarpeeksi monta, jokaiselle opetusaineiston havainnolle x_i saadaan testiluokka noin joka kolmannesta päätöspuusta. Kun algoritmi on lopettanut suorituksensa, katsotaan, mikä on havainnon x_i yleisin saama testiluokka j. Nyt out-of-bag -virhe on virheellisesti annettujen luokkien, eli kun j ei ole sama kuin x_i:n todellinen luokka, suhde havaintojen määrään. Esimerkiksi jos satunnaismetsä on antanut sadasta havainnosta kolmelle väärän luokan, on out-of-bag -virhe 3 %.

6.2. Päätöspuiden ihanteellinen määrä

Satunnaismetsä-mallin päätöspuiden määrä B on vapaasti asetettava parametri, jonka arvo riippuu opetusdatan koosta ja luonteesta. Yleisesti on niin, että mitä suurempi B:n arvo eli mitä enemmän satunnaismetsässä on puita, sitä tarkempi on satunnaismetsän antama ennuste uusille havainnoille. Voi kuitenkin olla epäselvää, kuinka monta päätöspuuta on rakennettava, jotta saavutettaisiin tyydyttävä tarkkuus satunnaismetsä-mallin antamille ennusteille. Tämän selvittämiseen voidaan hyödyntää out-of-bag -virhettä.

Kun algoritmi rakentaa päätöspuita satunnaismetsä-mallia varten, voidaan out-of-bag -virheen avulla päätellä, onko malli jo tarpeeksi tarkka käytettäväksi, vai pitäisikö rakentaa vielä lisää puita tarkkuuden parantamiseksi. Laskemalla

out-of-bag -virheen arvo sopivin väliajoin, esimerkiksi joka sadannen puun rakentamisen jälkeen, nähdään, milloin tarkkuus ei parane enää merkittävästi ja täten löydetään optimaalinen päätöspuiden määrä B satunnaismetsä-malliin. Puita ei siis ole liian vähän, jolloin mallin antamien ennusteiden tarkkuus olisi huono, mutta ei myöskään liian paljon, jolloin oltaisiin tehty paljon työtä saavuttamatta kuitenkaan merkittävää parannusta ennustetarkkuuteen.

6.3. Muuttujien merkitsevyys

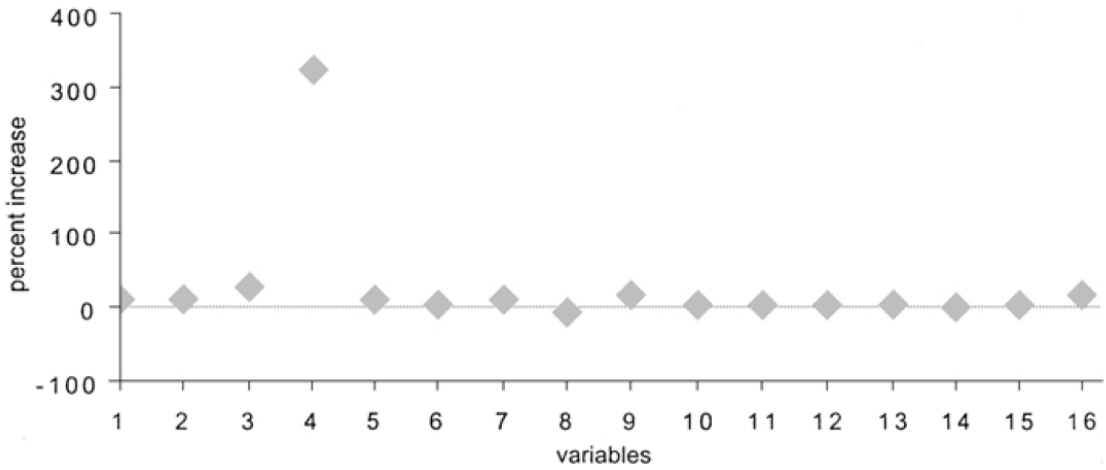
Joskus on tarpeen ymmärtää, mitkä muuttujat lisäävät parhaiten kohdearvojen ennustamisen tarkkuutta. Esimerkiksi luvun 5 esimerkissä voisi olla tärkeää saada selville, mitkä potilaan ominaisuudet ovat tärkeitä diagnoosin ennustamisen kannalta ja mitkä taas eivät tarjoa siihen riittävää hyötyä. Satunnaisessa metsässä ennustamisen kannalta merkittävien muuttujien selvittämisessä voidaan myös hyödyntää out-of-bag -virhettä. Määriteltäessä muuttujien merkitsevyyksiä muodostetaan normaalilla tavalla lasketun out-of-bag -virheen lisäksi monta erilaista out-of-bag -virhettä, yksi jokaista muuttujaa kohti.

Oletetaan, että syötemuuttujia on p kappaletta. Jokaisen satunnaismetsän päätöspuun rakentamisen jälkeen muuttujan y_i arvot puun rakentamisessa käyttämättömissä havainnoissa permutoidaan satunnaisesti. Out-of-bag -havaintojen arvot muuttujan y_i kohdalla siis järjestetään uudelleen satunnaiseen järjestykseen. Permutoinnin jälkeen out-of-bag -havainnot ajetaan juuri rakennetun puun läpi, ja kunkin out-of-bag -havainnon x_i saama luokka tallennetaan. Tätä prosessia toistetaan jokaiselle $y_i = 1, 2, \ldots, p$. Kaikkien päätöspuiden rakentamisen jälkeen kunkin havainnon x_i yleisintä saamaa luokkaa muuttujan y_i ollessa satunnaisesti permutoitu verrataan x_i:n todelliseen luokkaan, jolloin saadaan virheellisesti annettujen luokkien suhde havaintojen määrään. Näin ollaan saatu p kappaletta out-of-bag -virheitä, joissa kussakin on ollut eri muuttuja y_i satunnaisesti permutoituna.

Vertailemalla out-of-bag -virhettä muuttujan y_i ollessa satunnaisesti permutoituna normaaliin out-of-bag -virheeseen, missä muuttujien arvot ovat koskemattomia, saadaan selville, paljonko ennustevirheessä on tapahtunut kasvua y_i arvojen sekoittamisen seurauksena. Jos ennustevirhe ei ole kasvanut huomattavasti, on y_i todennäköisesti ennustamisen kannalta hyvin merkityksetön muuttuja. Jos taas ennustevirheessä on tapahtunut huomattavaa kasvua, on kyseessä luokan ennustamisen kannalta merkityksellinen muuttuja.

Breimanin [2001] esittämässä esimerkissä out-of-bag -virheiden ja muuttujien arvojen satunnaisen permutoinnin hyödyntämisessä on opetusaineistona käytetty Yhdysvaltojen 435 kongressiedustajaa. Syötemuuttujia on 16, jotka ilmentävät edustajien kyllä- ja ei-ääniä 16 asiakysymyksessä. Kohdemuuttujana on edus-

tajan puolue, eli edustajan luokka on joko republikaani tai demokraatti. Merkitsevien muuttujien selvittämiseksi satunnaismetsän muodostamisen yhteydessä määriteltiin tavallisen out-of-bag -virheen lisäksi 16 muuta out-of-bag -virhettä, joissa kussakin oli eri syötemuuttujan arvot satunnaisesti permutoituna. Näin saatiin selville, montako prosenttia yksittäisen muuttujan y_i arvojen sekoittaminen kasvatti ennustevirhettä tavalliseen, muuttujien arvot koskematta jättäneeseen out-of-bag -virheeseen verrattuna. Testin tulokset ovat kuvassa 8. Kuten kuvasta käy ilmi, ennustevirhe kasvoi kolme kertaa korkeammaksi, kun muuttujan 4 arvot permutoitiin satunnaiseen järjestykseen, kun taas muiden muuttujien osalla huomattavaa kasvua ei syntynyt. Voidaan siis päätellä, että äänestyspäätös asiakysymyksessä 4 on merkittävä ominaisuus ennustettaessa kongressiedustajan puoluetta.



Kuva 8. Breimanin [2001] muuttujien merkitsevyystestin tulokset.

Merkitseviltä vaikuttavien muuttujien löytämisen jälkeen voidaan vielä varmistaa niiden merkitsevyys ennustamisen kannalta muodostamalla uusi päätösmetsä niin, että vain niitä käytetään mallin rakentamisessa. Näin voidaan verrata vain valituin muuttujin rakennetun satunnaismetsän ennustevirhettä kaikilla muuttujilla rakennetun satunnaismetsän ennustevirheeseen. Jos valituin muuttujin rakennetun satunnaismetsä-mallin ennustevirhe on lähellä alkuperäisen, kaikilla muuttujilla rakennetun satunnaismetsä-mallin ennustevirhettä, voidaan muuttujien merkitsevyyttä ennustamisen kannalta pitää varmistettuna. Esimerkiksi edellä esitellyssä Breimanin [2001] esimerkissä kongressiedustajien äänestyskäyttäytymisestä muuttujan 4 merkitsevyys varmistettiin muodostamalla uusi satunnaismetsä vain sitä käyttäen. Näin muodostetun satunnaismetsä-mallin ennustevirhe oli lähes sama kuin kaikilla muuttujilla muodostetun satunnaismetsän ennustevirhe. Asiakysymyksestä 4 äänestäminen siis erotti republikaanit demokraateista melkein yhtä hyvin kuin kaikista 16 asiasta äänestäminen.

7. Yhteenveto

Satunnaismetsä on tehokas ja suosittu menetelmä moniin luokittelun ja regressioanalyysin tehtäviin. Se osaa hyödyntää perinteistä päätöspuuoppimista siten, että niiden kielteisiä ominaisuuksia vähennetään reilusti. Esimerkiksi ylisovittumisen vaara on huomattavasti pienempi kuin yksittäisellä päätöspuulla. Monissa ongelmissa satunnaismetsän suorituskyky on samankaltainen erääseen toiseen luokittelu- ja regressiotehtäviä suorittavaan *boostaukseen* (boosting), mutta siihen verrattuna satunnaismetsän opettaminen ja säätäminen on yksinkertaisempaa [Friedman *et al.* 2009].

Koska satunnaismetsä on suosittu työkalu, se on käytössä monessa analytiikkaohjelmistossa. Esimerkiksi suosittuun tilastolliseen ohjelmistoon R on saatavilla satunnaismetsän toteuttava paketti [Friedman *et al.* 2009]. Satunnaismetsälle on olemassa monenlaisia sovelluskohteita, esimerkiksi ekologiassa [Cutler *et al.* 2007].

Viiteluettelo

Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1, 5–32.

Leo Breiman, Jerome Friedman, Charles J Stone and Richard A Olshen. 1984. *Classification and Regression Trees*. CRC Press.

D. R. Cutler, Thomas C. Edwards, Karen H. Beard, Adele Cutler, Kyle T. Hess, Jacob Gibson and Joshua J. Lawler. 2007. Random forests for classification in Ecology. *Ecology* 88, 11, 2783–2792.

Jerome Friedman, Trevor Hastie and Robert Tibshirani. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Tin Kam Ho. 1995. Random decision forests. In: *Proc. of 3rd International Conference on Document Analysis and Recognition* 1, 278–282.

Tin Kam Ho. 1998. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, 8, 832–844.

C4.5-algoritmin toiminta

Miikka Mäki

Tiivistelmä

C4.5-algoritmi on luokittelualgoritmi, joka tuottaa informaatioteorian perusteella päätöspuita haje- ja saantilaskelmien avulla. Päätöspuut luodaan ennalta määriteltyjen luokkien mukaan, eli kyseessä on ohjattuun koneoppimiseen pohjautuva algoritmi. Algoritmille syötetään vaatimusten mukainen opetusjoukko, josta algoritmi pilkkoo syötetyt datatietueet osajoukoiksi parhaan saannin mukaan laskevasti. Osajoukoista algoritmi luo mahdollisimman tarkan ja yksinkertaisen päätöspuun, jonka avulla uusia datatietueita voidaan luokitella. Tässä tutkielmassa esitellään C4.5-algoritmin toimintavaiheita ja toiminnan periaatteita.

Avainsanat: ohjattu koneoppiminen, algoritmit, päätöspuut, dataluokittelijat.

1. Johdanto

Viime vuosikymmenten aikana koneoppiminen (machine learning) on kasvattanut merkitystään informaatioteknologian alueella: koneoppimisesta on tullut merkittävä, vaikkakin usein kätketty osa jokapäiväistä elämäämme [Smola *et al.* 2008]. Käytettävissä olevan datan määrän kasvaessa entistä älykkäämpien data-analyysimenetelmien kehittämistä pidetään tärkeänä tavoitteena teknologisen kehityksen kannalta [Smola *et al.* 2008]. Koneoppiminen on osoittautunut tehokkaaksi suurten datamassojen analysointimenetelmäksi, jossa tietokone kykenee optimoimaan toimintaansa oppimalla tietoa vastaanottamastaan datasta.

Ihmisten ja tietokoneiden välistä oppimista tarkastellessa voidaan kuitenkin havaita selkeitä eroja: ihminen kykenee järkeilemään ja käyttämään loogista päättelyä oppiessaan, mutta tietokone ei tähän kykene. Jos kuitenkin pohdimme kaikkein perusteellisimpia oppimisen keinoja, voidaan huomata, että tietokone kykenee varsin samankaltaiseen oppimisprosessiin kuin ihminenkin: tunnistamalla tilanteen, jossa olemme olleet aikaisemmin (olemme tarkastelleet dataa aiemmin), kokeilemalla jotakin toimintatapaa (lähettämällä tulosteen) ja onnistumalla (toimintatapa oli oikea) tai epäonnistumisen sattuessa kokeilemalla ensi kerralla toisenlaista toimintatapaa [Smola *et al.* 2008]. Nykyaikana koneoppimisella toteutetaan lukuisia erilaisia sovelluksia: koneoppimista käytetään esimerkiksi verkkohakukoneissa, suodatus-ja käännössovelluksissa ja nimikokonaisuuksien tunnistuksessa [Smola *et al.* 2008].

Koneoppimisessa tietokoneen toiminnan tarkkuutta pyritään mukauttamaan ja tehostamaan käyttämällä algoritmeja, joiden avulla se kehittyy vastaamaan ihanteellista, toivottua toimintatapaa [Marsland 2015]. Algoritmi oppii ja kehittyy vastaanottamansa opetusaineiston perusteella ja pyrkii siitä luotujen ennakkoesimerkkien mukaan päättelemään todennäköisimmän tulevaisuuden toimintamallin.

Algoritmin tarkkuutta voidaan mitata vertailemalla algoritmin päättelemän toimintamallin oikeellisuutta oikeisiin vastauksiin, joihin algoritmia pyritään ohjaamaan [Marsland 2015]. Koneoppimisen menetelmä voidaan jakaa kolmeen peruskomponenttiin [Quinlan 2014]: tiedon esitystapa, jolla määritellään tiedon oletettu tila, käytettävän algoritmin arviointi, jolla tietoa haetaan oletetusta tilasta ja haun ominaisuuksien optimointi, jolla hakua ohjataan kohti toivottua tulosta.

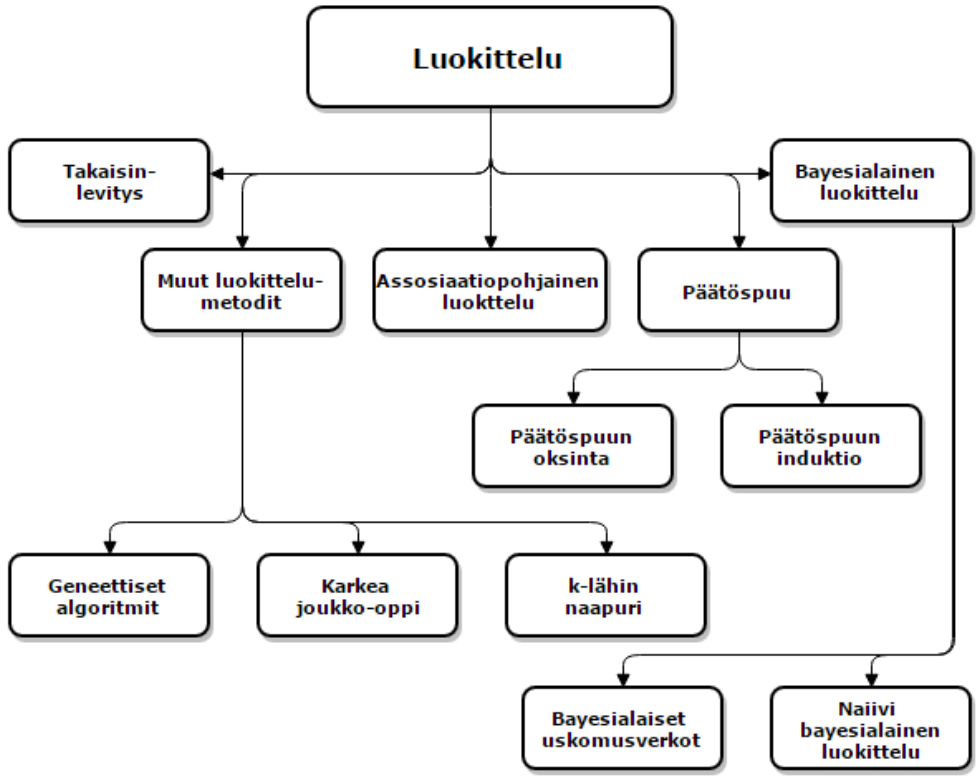
Tutkielmani on kirjallisuuskatsaus C4.5-algoritmin toimintaan. Tärkeimpinä aineistoina toimivat C4.5-algoritmin luoneen J. Ross Quinlanin julkaisut, joissa hän käsittelee algoritminsa toimintaa ja optimointimahdollisuusia monesta koneoppimisen eri näkökulmasta. Quinlan kehitti C4.5-algoritmin vuonna 1993, jolloin hän yhdisti aikaisemmin kehittämänsä luokittelualgoritmin, ID3:n, tehokkaampaan ja joustavampaan C4.5-versioon. C5.0 on C4.5-algoritmin kehittyneempi versio.

Toisessa luvussa käsittelen yleisesti päätöspuita, joita C4.5-algoritmi käyttää käsitellyn tiedon luokittelijana. Kolmannessa luvussa käsittelen C4.5-algoritmin toiminnan periaatteita: esittelen algoritmin pseudokoodin, eri toimintavaiheet eri ja erityispiirteet. Neljännessä luvussa esittelen C4.5-algoritmin toimintaa käytännön esimerkin avulla. Viidennessä luvussa vertailen C4.5-algoritmin toimintaa ja ominaisuuksia sen kehittyneempään versioon, C5.0:aan. Viimeisessä varsinaisessa luvussa käsittelen C4.5-algoritmin toimintaan liittyviä ongelmia ja mahdollisia tulevaisuuden tutkimusmahdollisuuksia.

2. Päätöspuut

Luokittelu on yksi tärkeistä koneoppimisalgoritmien tehtävistä. Luokittelu koostuu datatietueiden sijoittamisesta luokka-arvoihin käyttäen opetusaineiston (training set) avulla muodostettua luokittelijaa. Luokittelijan avulla algoritmi arvioi/ennustaa uusien datatietueiden luokan [Ilyes *et al.* 2008]. Toisin sanoen algoritmi määrittelee luokittelijan opetusaineistosta, jonka avulla vertaillaan opetusaineiston tietueiden attribuutteja ja sijoitettavien datatietueiden attribuutteja keskenään. Vertailulla voidaan ennustaa todennäköisin luokka uudelle tietueelle.

Yksi koneoppimisalgoritmien käyttämistä luokittelumenetelmistä on päätöspuut. Päätöspuu on vuokaaviota mukaileva puurakenne, joka koostuu kolmesta peruselementistä: puun solmut vastaavat joko algoritmille syötetyn opetusaineiston attribuutteja, attribuuttien oksia/reunoja tai puun lehtiä, jotka vastaavat määriteltyjä luokkia [Ilyes *et al.* 2008]. Päätöspuita pidetään yhtenä suosituimmista koneoppimisalgoritmien luokittelumenetelmistä [Ilyes *et al.* 2008]. Kuvassa 1 havainnollistetaan hierarkiapuun avulla muita koneoppimisessa käytettäviä luokittelumenetelmiä.



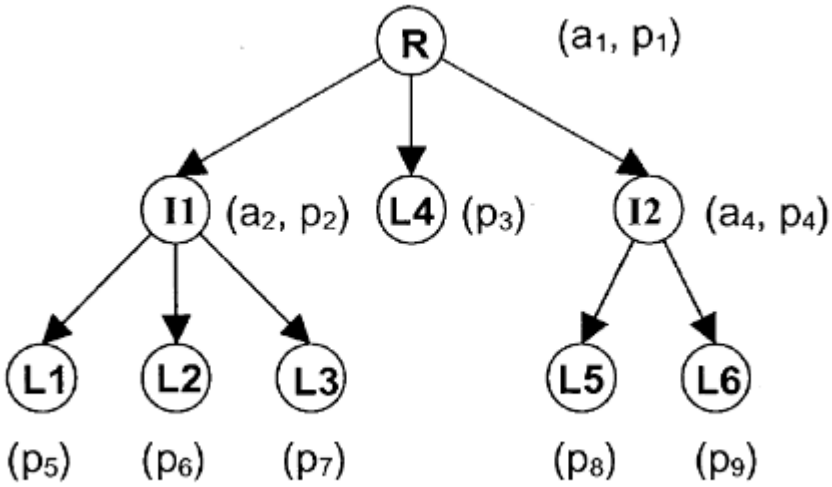
Kuva 1: Luokittelija-algoritmien hierarkiapuu [Pandya *et al.* 2015].

Wu ja muut [2008] arvioivat C4.5-algoritmin parhaaksi tiedonlouhinta-algoritmiksi vuonna 2008. C4.5-algoritmin lisäksi datan luokitteluun ja esittelyyn päätöspuita käyttäviä algoritmeja ovat esimerkiksi C4.5-algoritmia edeltävä versio, ID3-algoritmi, ja CART-algoritmi.

Ilyes ja muut [2008] kuvailevat päätöspuiden synnyn kaksivaiheiseksi: Rakennusvaiheessa syötetystä opetusaineistosta poimitaan valintametodien avulla tyhjälle puulle sopivimmat testiattribuutit päätöspuun solmukohtiin. Attribuuttien sopivuuden mittaamiseen käytetään saanti- ja hajelaskemia, joiden avulla voidaan määritellä eri attribuuttien hyvyys luokitteluvaiheessa. Tarkoituksena on valita attribuutti, joka palauttaa mahdollisimman suuren saannin (gain) etsintävaiheessa luotujen osajoukkojen välillä, jotta päätöspuun mukaan sijoi-

tettavien instanssien luokkien määrittely tapahtuisi mahdollisimman tarkasti. Avaan valikointimetodien toimintaa tarkemmin seuraavassa luvussa, jossa tutkitaan C4.5-algoritmin toiminnan periaatteita.

Luokitteluvaiheessa datatietuetta luokitellessa kuljetaan pitkin päätöspuuta ja vertaillaan solmukohdissa tarkasteltavien attribuuttien arvoja luokiteltavan tietueen attribuuttien arvoihin. Päätöspuussa edetään arvojen mukaan niin kauan, kunnes kohdataan puun lehti. Päätöspuun lehti määrittelee luokan uudelle syötteelle. Kuvassa 2 on esimerkki päätöspuusta (a edustaa attribuutteja, p edustaa luokkia).



Kuva 2: Esimerkki päätöspuusta

3. C4.5-algoritmin toiminnan periaatteet

C4.5-algoritmi on tietokoneohjelmien kokoelma, joka luo induktiivisen yleistyksen periaatteella päätöspuita käyttäviä luokittelumalleja syötetystä opetusaineistosta, joka sisältää spesifejä tapauksia [Quinlan 1996a]. Opetusaineiston sisällön tulee vastata algoritmin vaatimuksia, jotta luokittelu onnistuisi tehokkaasti. Esittelen seuraavaksi Quinlanin [2014] määrittelemät vaatimukset opetusjoukolle, joka syötetään algoritmille:

1. Analysoitavan datan tapausten informaatio tulee olla ilmaistu kiinnitettynä kokoelmana joko diskreettejä tai numeerisia arvoja sisältävinä attribuutteina.
2. Kategoriat, joihin tapaukset pyritään luokittelemaan, tulee olla ennalta määriteltyjä luokkia. Tämä tarkoittaa, että C4.5-algoritmi toteuttaa ohjatun koneoppimisen periaatteita. Ohjaamattomat koneoppimisalgoritmit kykenevät

itse etsimään ryhmät opetusaineistosta, joille määritellään luokat ja joiden mukaan datatietueita luokitellaan. Luokkien tulee myös olla diskreettejä ja tarkkaan rajattuja: luokkia ei tule olla enemmän kuin tapauksia.

3. Algoritmille tulee tarjota riittävästi opetusaineistoa, jotta induktiivisen yleistyksen avulla datasta voidaan erottaa siinä esiintyviä toimintarakenteita. Datan määrän tarve on verrannollinen ennalta määriteltyjen luokkien ja attribuuttien määrään: mitä monimuotoisempi datasarja on, sitä enemmän dataa tulee tarjota, jotta algoritmi kykenee rakentamaan luotettavia luokittelumalleja.
4. C4.5 vaatii ”loogisia” luokittelumalleja: luokittelijat tulee olla muodossa, jossa ne kyetään esittämään luonnollisesti päätöspuussa. Toisin sanoen aritmeettiset luokittelijat, jotka perustuvat attribuuttien arvojen yhdistelyyn ja vertailuun, eivät ole mahdollisia.

3.1 Periaatteet

C4.5-algoritmi käyttää ”hajota ja hallitse” (divide and conquer) -menetelmää datan pilkkomiseksi osajoukkoihin. Olkoon *joukko* $C = \{C_1, C_2, \ldots, C_k\}$ joukko tapauksia, jotka tulee luokitella. Algoritmi toimii seuraavasti [Quinlan 1996b, 78]:

- Mikäli C kohtaa lopetuskriteerin, C:n päätöspuu on lehti, joka koostuu C:n yleisimmästä luokasta. Lopetuskriteereitä voivat olla joko vain yhden luokan kohtaaminen joukossa tai se, että jokaisella opetusaineiston instanssilla on erillinen luokka.
- Jaetaan aineisto C osajoukkoihin $C_1, C_2, \ldots, C_k$ kriteerin $T = \{T_1, T_2, \ldots, T_k\}$ avulla, jossa C_i sisältää T_i:n saaneet havainnot. Koko joukon C puu sisältää testijoukon T juurenaan. Puun jokainen solmukohta on tulos T_i, jotka määritellään soveltamalla samaa toimintatapaa rekursiivisesti C_i:n tapauksille.

Tarkoituksena on pilkkoa T osajoukkoihin, jotka johtavat yhdestä luokasta koostuviin tapauskokoelmiin. Puun solmukohdat jakavat puun oksiin, joissa otetaan huomioon jokainen mahdollinen testitulos. Näin voimme tarkasti määrittää tapauksille omat luokkansa.

3.2 Arviointitestit

Arviointitestien tarkoituksena on löytää mahdollisimman tehokas testidatan osittelumenetelmä päätöspuun luontia varten. Quinlan [2014] osoittaa, että määritelläkseen mahdollisimman tehokkaan ja yksinkertaisen päätöspuun opetusaineistosta C4.5-algoritmi hyödyntää arviointitestejä, joiden avulla pyritään määrittämään ihanteellinen järjestys tapauksien attribuuttien arvottamisessa.

Tämän saavuttamiseksi C4.5 käyttää saantikriteeriä (gain criterion): saantilaskelmien avulla C4.5 määrittää luotettavimman attribuutin, jonka luokan etsimisessä esiintyy vähiten hajetta (entropy).

Saantilaskelmien avulla algoritmi kykenee määrittämään tehokkaimman ja tarkimman etenemisjärjestyksen tapausten mukaan. Näin vältytään luomasta useita päätöspuita ja vertailemasta niitä keskenään. Useiden päätöspuiden luontimenetelmällä toteutettavien päätöspuiden määrä voisi kasvaa suhteessa opetusaineiston suuruuteen sellaisiin lukemiin, että optimointi hidastuisi merkittävästi. Saantilaskelmiin perustuvia algoritmeja kutsutaan ”ahneiksi” algoritmeiksi (nonbacktracking, greedy): kun testi on päätetty sisältämään valitusta joukosta testitapauksia, jolle on laskettu paras mahdollinen saanti, päätöksessä pysytään ja jätetään muut mahdollisuudet tarkastelematta. Päätöksen tulee siis olla tarkka [Ingargiola 1996a].

Arviointitestien kriteerit toimivat mekanismina, jonka avulla algoritmi kykenee arvottamaan esitetyt testit paremmuusjärjestykseen, jotta parhaimman tuloksen palauttava testi voidaan valita tarkasteltavaksi. Luokittelija-algoritmit käyttävät tämän kaltaisia arviointimekanismeja hyödykseen, jotta niiden toiminta olisi mahdollisimman tarkkaa ja tehokasta. Testi usein sisältää vain yhden opetusaineistosta valitun attribuutin; näin päätöspuut pysyvät selkeinä eivätkä paisu liian vaikeiksi ymmärtää.

C4.5:n arviointimekanismi käyttää kolmea eri testityyppiä [Quinlan 2014]: Yleisin menetelmä on valita yksi diskreetti attribuutti, jolla on yksi tulos ja oksa jokaista mahdollista vaihtoehtoa varten. Toinen vaihtoehto on monimutkaisempi testi, jossa diskreetin attribuutin arvot jaetaan tuloksen mukaan ryhmittäin, jossa joka ryhmällä on yksi tulos arvon sijaan. Tämä testi tulee erikseen määrittää käytettäväksi algoritmia käynnistäessä. Viimeistä testiä käytetään numeraaliarvoille, kun arvoja vertaillaan binäärisesti. Jokainen testityyppi arvotetaan saman laskukaavan mukaan, jossa mitataan attribuutin saantisuhde (gain ratio). Esittelen saantilaskelmat seuraavassa kohdassa.

3.3 Saantikriteeri

Oletetaan, että mahdollinen testi, jolla on *n* tulosta, jakaa testijoukon *T* osajoukkoihin $T_1, T_2, \ldots, T_n$. Arvioidaksemme testijoukon ilman jokaisen mahdollisen T_i:n jaon erillistä tarkastelua käytettävissämme on testijoukossa sisältyvien luokkien jakautuneisuus *T*:ssä ja sen osajoukoissa. Olkoon joukko S minkä tahansa opetusdatan joukko, ja olkoon $freq(C_i, S)$ niiden joukon S tapausten määrä, jotka kuuluvat luokkaan C_i. Käytetään myös standardinotaatiota, jossa |S| osoittaa tapauksien määrän joukossa *S* [Quinlan 2014].

Saantilaskelmilla pyritään jakamaan osajoukot mahdollisimman ”puhtaaseen” muotoon. Toisin sanoen, mikäli luokkajako olisi kaksijakoinen, (esimerkiksi luokat: ”positiivinen” ja ”negatiivinen”) ihanteellisessa tilanteessa osajoukot koostuisivat joko ainoastaan positiivisista tai negatiivisista luokkatapauksista, tai suurimmasta positiivisten tapausten joukosta. Saantikriteerin käyttämät laskukaavat ovat seuraavat [Quinlan 2014]:

Valitaan yksi satunnainen tapaus tapausjoukosta *S* ja päätetään, että se kuuluu johonkin luokkaan C_j. Tällä tapauksella on todennäköisyys

$$\frac{freq(C_j,S)}{|S|} \tag{1}$$

ja informaation määrä, jonka kaava välittää, on

$$-log_2\left(\frac{freq(C_j,S)}{|S|}\right). \tag{2}$$

Oletetun informaation löytämiseksi summaamme luokat suhteessa niiden lukumäärään joukossa *S*. Saadaan

$$info(S) = -\sum_{j=1}^{k} \frac{freq(C_j,S)}{|S|} \times log_2\left(\frac{freq(C_j,S)}{|S|}\right). \tag{3}$$

Tiedon tarpeen keskimääräinen lukumäärä saadaan soveltamalla laskukaavaa *info*(T) opetustapauksiin, jotta tapauksen luokka joukossa *T* voidaan määrittää. Määre vastaa joukon *S* hajetta (entropy).

Samankaltaista mittausta sovelletaan tapaukseen, jossa *T* on ositettu *n* tulosten mukaan testissä *X*. Oletettu tiedontarve saadaan osajoukkojen painotettuna summana, joka on

$$info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times Info(T_i). \tag{4}$$

Määrä

$$gain(X) = info(T) - info_X(T) \tag{5}$$

mittaa informaatiota, joka saadaan osittamalla *T* testijoukko *X*:n mukaisesti. Laskutoimitusten perusteella algoritmin saantikriteeri valitsee maksimaalisen saannin palauttavan testijoukon.

Tässä menetelmässä on kuitenkin perustavanlaatuinen ongelma: palautettuun informaation vaikuttaa merkittävästi mahdollisten tulosten määrä ja menetelmä palauttaa maksimaalisen arvon, mikäli jokaisessa osajoukossa T_i on ainoastaan yksi tapaus. Palautetun tuloksen epävarmuutta voidaan tasapainottaa osittamalla ja normalisoimalla tulos edelleen. Quinlan [2014] osoittaa, että normalisoimalla suurista osajoukoista koostuvan testin saantia palaute ”puhdistuu” tarkemmaksi. Kaava

$$split\ info(X) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times log_2\left(\frac{|T_i|}{|T|}\right) \tag{6}$$

kuvaa potentiaalista informaatiota, joka muodostetaan jakamalla T n:ään osajoukkoon, jossa saanti vastaa saman joukon luokitteluun tarvittavaa tietoa. Näin ollen

$$gain\ ratio(X) = \ gain(X)/\ split\ info(X) \qquad (7)$$

kuvaa hyödyllisen tiedon määrää, jota voidaan käyttää osajoukkojen luokitteluun. Saantikriteeri pyrkii valitsemaan osajoukon, jolla on suurin saanti, jotta split-suhde ei jää pieneksi ja luokittelun kannalta epävarmaksi.

3.5 C4.5-algoritmin pseudokoodi

Algoritmissa 1 esittelen C4.5-algoritmin pseudokoodin. Pseudokoodia käytetään tietokoneohjelmien koodin perusrakenteen esittelyyn, missä ei oteta huomioon eri ohjelmointikielien syntaksieroja:

Algoritmi 1: C4.5(D) [Wu 2008]

Syöte: attribuutti-arvoinen datasarja D

1. Puu = {}
2. **jos** D on "puhdas" TAI jokin muu päätösehto kohdataan **niin**
3. lopeta
4. **lopeta jos**
5. **kaikille** attribuuteille a ∈ D **tee**
6. Laske informaatio laskukriteereiden mukaan
7. **lopeta kohden**
8. a_{paras} = Paras attribuutti laskentakriteereiden mukaan
9. Puu = Luo päätössolmu, joka testaa a_{paras} juuressa
10. D_v = asetetut osadatat D:stä a_{paras}:n perustuen
11. **kaikille** D_v **tee**
12. Puu_v= C4.5(D_v)
13. Liitä Puu_v vastaavaan Puun haaraan
14. **lopeta kohden**
15. **palauta** Puu

4. Päätöspuun luonti C4.5-algoritmilla

Seuraavaksi esitän käytännön esimerkin [Quinlan 2014], jossa C4.5-algoritmille syötetään opetusjoukko, ja josta algoritmi luo aikaisemmassa luvussa mainittujen kriteerien mukaisesti päätöspuun ennalta määrättyjen luokkien mukaan.

4.1 Esimerkin käsittely

Olkoon opetusaineistona sarja Jarkon tennisharjoituksista. Opetusaineiston tietueilla on neljä attribuuttia (sääennuste, lämpötila, kosteus, tuulisuus) ja kaksi luokkaa (pelataan ja ei pelata), ks. taulukko 1.

Sääennuste	Lämpötila(F)	Kosteus(%)	Tuulisuus?	Luokka
Aurinkoinen	75	70	kyllä	Pelataan
Aurinkoinen	80	90	kyllä	Ei pelata
Aurinkoinen	85	85	ei	Ei pelata
Aurinkoinen	72	95	ei	Ei pelata
Aurinkoinen	69	70	ei	Pelataan
Pilvinen	72	90	kyllä	Pelataan
Pilvinen	83	78	ei	Pelataan
Pilvinen	64	65	kyllä	Pelataan
Pilvinen	81	75	ei	Pelataan
Sateinen	71	80	kyllä	Ei pelata
Sateinen	65	70	kyllä	Ei pelata
Sateinen	75	80	ei	Pelataan
Sateinen	68	80	ei	Pelataan
Sateinen	70	96	ei	Pelataan

Taulukko 1: Esimerkissä käytettävä harjoitusdatajoukko [Quinlan 2014]

Kaavan (3) avulla voimme määritellä luokat suhteessa niiden lukumäärään joukossa T määrittääksemme joukon luokan, kuten edellisessä luvussa on esitetty. Tämä luku vastaa siis joukon T hajetta, toisin sanoen keskimääräistä informaationtarvetta, jolla joukon T luokka kyetään määrittämään. Opetusaineistossa on kaksi luokkaa, *pelataan* (9/14) ja *ei pelata* (5/14):

$$info(T) = \frac{9}{14} \times log_2\left(\frac{9}{14}\right) - \frac{5}{14} \times log_2\left(\frac{5}{14}\right) = 0.94.$$

Käytetään attribuuttia *sääennuste* jakamaan joukko T kolmeen osajoukkoon (sääennusteella on kolme eri arvoa) kaavan (4) mukaisesti:

$$\begin{aligned} info_{sääennuste}(T) = \quad & \frac{5}{14} \times \left(\frac{2}{5} \times log_2\left(\frac{2}{5}\right) - \frac{3}{5} \times log_2\left(\frac{3}{5}\right)\right) \\ & + \frac{4}{14} \times \left(\frac{4}{4} \times log_2\left(\frac{4}{4}\right) - \frac{0}{4} \times log_2\left(\frac{0}{4}\right)\right) \\ & + \frac{5}{14} \times \left(\frac{3}{5} \times log_2\left(\frac{3}{5}\right) - \frac{2}{5} \times log_2\left(\frac{2}{5}\right)\right) = 0.694. \end{aligned}$$

Saantikaavan (5) mukaisesti voimme nyt laskea saannin attribuutille *sääennuste*:

$$gain(X) = 0.94 - 0.694 = 0.246.$$

Jos olisimme valinneet tarkasteltavaksi attribuutin *sääennuste* sijaan attribuutin *tuuleeko*, olisimme saaneet arvot $info_{tuuleeko}(T) = 0.892$ ja $gain(X) = 0.94 - 0.892 = 0.048$. Attribuutti *sääennuste* palauttaa siis suuremman saannin, ja C4.5-algoritmi valitsee sen ensimmäiseksi tarkasteltavaksi attribuutiksi päätöspuun rakennusprosessissa.

4.2 Tapauksien osittaminen

Osoitetaan seuraavaksi osajoukot attribuuttien ja luokkien mukaan mahdollisimman ”puhtaana” (vain yhden luokan esiintymiä osajoukossa) taulukoiden 2-6 tapaan.

Ennuste = **Aurinkoinen**:

Kosteus < 76:

Sääennuste	Lämpötila(F)	Kosteus(%)	Tuulisuus?	Luokka
Aurinkoinen	75	70	kyllä	Pelataan
Aurinkoinen	69	70	ei	Pelataan

Taulukko 2: Osajoukko 1 attribuutin sää*ennuste* mukaan [Quinlan 2014]

Kosteus > 75

Sääennuste	Lämpötila(F)	Kosteus(%)	Tuulisuus?	Luokka
Aurinkoinen	80	90	kyllä	Ei pelata
Aurinkoinen	85	85	ei	Ei pelata
Aurinkoinen	72	95	ei	Ei pelata

Taulukko 3: Osajoukko 2 attribuutin sää*ennuste* mukaan [Quinlan 2014]

Ennuste = **Pilvinen**

Sääennuste	Lämpötila(F)	Kosteus(%)	Tuulisuus?	Luokka
Pilvinen	72	90	kyllä	Pelataan
Pilvinen	83	78	ei	Pelataan
Pilvinen	64	65	kyllä	Pelataan
Pilvinen	81	75	ei	Pelataan

Taulukko 4: Osajoukko 3 attribuutin sää*ennuste* mukaan [Quinlan 2014]

Ennuste = **Sateinen**

Tuulisuus = **Kyllä**

Sääennuste	Lämpötila(F)	Kosteus(%)	Tuulisuus?	Luokka
Sateinen	71	80	kyllä	Ei pelata
Sateinen	65	70	kyllä	Ei pelata

Taulukko 5: Osajoukko 4 attribuutin sää*ennuste* mukaan [Quinlan 2014]

Tuulisuus = **Ei**

Sääennuste	Lämpötila(F)	Kosteus(%)	Tuulisuus?	Luokka
Sateinen	75	80	ei	Pelataan
Sateinen	68	80	ei	Pelataan
Sateinen	70	96	ei	Pelataan

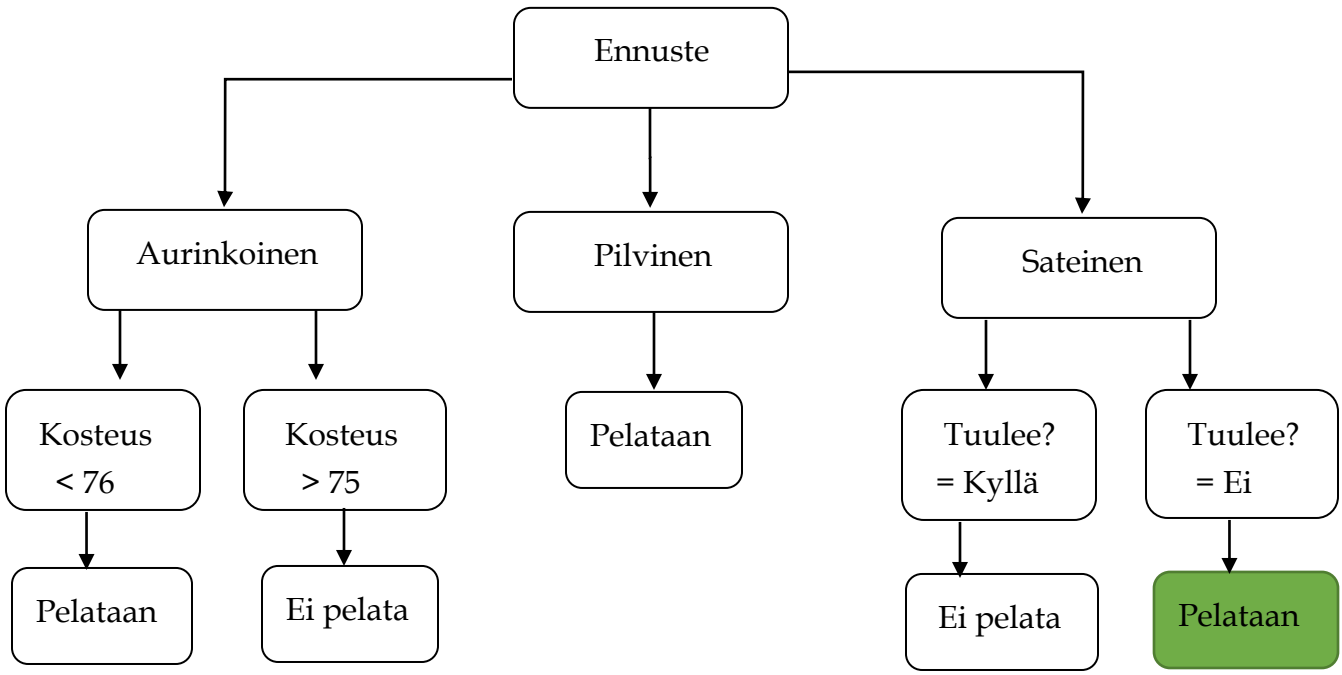
Taulukko 6: Osajoukko 5 attribuutin sää*ennuste* mukaan [Quinlan 2014]

Ositetun opetusaineiston avulla algoritmi luo päätöspuun rekursiivisesti attribuuttien saannin mukaan. Näin ollen voimme luokitella uuden datatietueen luokan luodun päätöspuun avulla, kulkemalla ylhäältä alaspäin, edeten solmukohdissa tietueen attribuuttien arvon mukaan. Jos uusi tietue, jolle tulisi selvittää luokka, olisi taulukon 7 kaltainen.

Sääennuste	Lämpötila(F)	Kosteus(%)	Tuulisuus?	Luokka
Sateinen	85	90	ei	?

Taulukko 7: Datatietue, joka tullaan luokittelemaan luodun päätöspuun perusteella

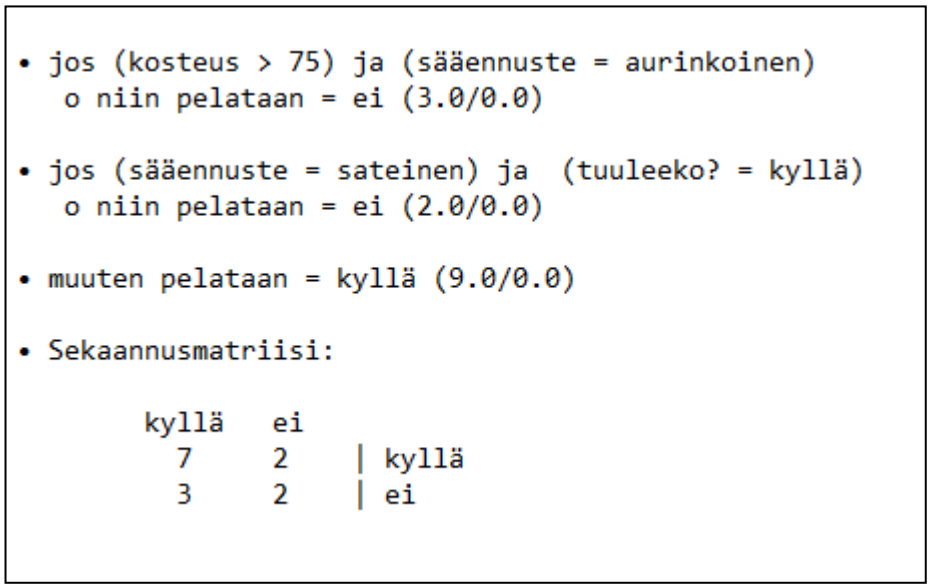
Tietueelle asetetaan päätöspuun (kuva 3) perusteella luokka: Pelataan.



Kuva 3: Opetusjoukon perusteella luotu päätöspuu

4.3 Sääntöjoukot

Suurista datasarjoista luodut päätöspuut voivat usein kasvaa varsin suuriksi ja vaikeiksi ymmärtää. C4.5 tarjoaa ongelmaan ratkaisun kykenemällä luomaan yksinkertaisista ”jos, niin” (*if, else*) -säännöistä koostuvan järjestämättömän sääntöjoukko-luokittelijan. Sääntöjoukon avulla algoritmi kykenee vertailemaan luokiteltavien datatietueiden attribuutteja luotuihin sääntöihin ja luokittelemaan tietueita ehtojen perusteella [Quinlan 1998a]. Luokitteluvaiheessa useitakin ehtoja saatetaan käyttää, mikäli niiden ehdot täyttyvät. Mikäli säännöt ennustavat tietueelle eri luokkia, algoritmi käyttää sääntöjen välillä äänestysmekanismia, jossa luotettavampi attribuutti vastaanottaa korkeamman pisteytyksen. Äänestämisen jälkeen luodaan sekaannusmatriisi ja asetetaan tietueelle todennäköisempi luokka [Quinlan 1998a]. Jos esimerkiksi käyttäisimme taulukon 1 datasarjaa sääntöjoukon luomiseen, sääntöjoukko olisi kuvan 4 kaltainen.



Kuva 4: Taulukosta 1 luotu sääntöjoukko

5. C4.5:n ja C5.0:n vertailu

Tässä luvussa käsittelen tarkemmin C5.0-algoritmia, joka on J. Ross Quinlanin vuonna 1997 luoma kehittyneempi versio C4.5-algoritmista [Quinlan 1998a]. Havainnollistaakseni algoritmien eroja ja uuden sukupolven ominaisuuksia vertailen C4.5:n kahdeksatta versiota C5.0:n versioon 2.07 esimerkkitapauksien avulla. Käsittelen myös tarkemmin C5.0:n ominaisuuksia, jotka ovat sille ominaisia. Algoritmien vertailuun on käytetty Intelin C-kääntäjää samoilla optimisointiasetuksilla ja testaus on suoritettu samalla järjestelmällä. Vertailuun käytetään kolmea opetusjoukkoa: *uni* (105 908 tapausta), *tulot* (199 523 tapausta) ja *metsä* (581 012 tapausta) [Hettich *et al.* 1999].

5.1 Lisätty toiminnallisuus

C5.0:aan on lisätty useita erilaisia toimintoja, jotka nostavat algoritmin toiminnan tehokkuutta. Yksi uusista ominaisuuksista on luokiteltavien tapausten arvottaminen painomääritteen avulla. Jotkut tapaukset voivat olla erilaisten luokittelukriteerien vuoksi enemmän tai vähemmän tärkeitä kuin toiset, joten tapausten arvottaminen tärkeyden mukaan vähentää algoritmin mahdollisia virheluokitteluja, ja näin ollen virheprosenttia [Quinlan 1998b]. Toinen uusi ominaisuus on muuttujien luokitteluvirheiden arvottaminen. C5.0 kykenee arvottamaan oletettujen ja oikeiden luokkien parit ja arvioimaan mahdollisten virheiden kriittisyyttä. C4.5:ssä jokainen virhe käsitellään tasavertaisena. Tällä

keinolla algoritmi kykenee vähentämään luokitteluvirheiden riskiä [Quinlan 1998b].

C5.0 käyttää lisättyjä datatyyppejä C4.5:n datatyyppien lisäksi attribuuttien merkinnässä, kuten aikaleimoja, tapausmerkintöjä ja järjestettyjä diskreettejä attribuutteja. C5.0 Kykenee myös luomaan uusia attribuutteja, jotka toimivat syötettyjen attribuuttien funktioina. Algoritmi kykenee myös karsimaan vähemmän relevantteja attribuutteja datasarjasta ennen varsinaisia luokittelu- ja ositteluvaiheita. Mikäli datasarjassa on tuhansia tapauksia, jotkin attribuutit voivat olla luokittelun kannalta merkityksettömiä. Näin ollen voidaan tehostaa algoritmin toimintaa entisestään. C5.0 on myös helppokäyttöisempi aikaisempaan versioonsa verrattuna: algoritmin vaihtoehtoja on yksinkertaistettu ja yhdistetty toisiinsa sulavamman käyttäjäkokemuksen takaamiseksi [Quinlan 1998b].

C5.0 on useassa eri vertailussa tarkempi, nopeampi ja kevyempi kuin versio C4.5 [Quinlan 1998a]. C5.0:lla on alhaisempi virheprosentti datatietueiden luokitteluvaiheessa, luo tarkempia lehtirakenteita päätöspuihin ja käsittelyaika on selkeästi nopeampi (ks. taulukko 8). Tehokkuus on suurempi, sillä C5.0 kykenee luomaan pienempiä päätöspuita, kykenee hallitsemaan muistinvarausta tehokkaammin ja myös poistaa automaattisesti hyödyttömiä attribuutteja ennen osittelu- ja luokitteluvaiheita [Pandya *et al.* 2015].

	Virheprosentti	lehdet	aika(sek.)
Uni C4.5	27.7%	3,546	3
Uni C5.0	27.0%	2,160	1
Tulot C4.5	5.0%	264	5
Tulot C5.0	4.9%	122	1
Metsä C4.5	6.1%	10,169	62
Metsä C5.0	6.1%	9,185	4

Taulukko 8: Luotujen päätöspuiden vertailua [Quinlan 1998b]

5.2 Edistäminen

Toinen merkittävistä ominaisuuksista C5.0:ssa, joita ei C4.5:ssä ole lainkaan, on mukautuva edistys (adaptive boosting). Tarkoituksena on luoda useampi luokittelija opetusaineistosta yhden sijaan. Uutta tapausta luokitellessa jokainen luokittelija ”äänestää” tapauksen todennäköisimmän luokan. Äänten perusteel-

la valitaan todennäköisin luokka tapaukselle [Quinlan 1998b]. Useamman luokittelijan luonti samasta datasarjasta tapahtuu keskittymällä ensimmäisen kierroksen virheisiin luokitteluvaiheessa. Seuraavassa kierroksessa luokittelija keskittyy tarkemmin virheellisiin luokkiin luokiteltuihin tapauksiin, jolloin luotu luokittelija usein poikkeaa edellisestä. Kierroksia toistetaan algoritmille asetettujen kertojen verran. Tällä tavalla usean päätöspuun avulla muodostettu luokittelumenetelmä antaa keskimääräisesti tarkemman ja tehokkaamman palautteen [Quinlan 1998b]. Taulukko 9 osoittaa datasarjojen virheprosentit ennen edistämistä ja kymmenen edistyskierroksen jälkeen.

	uni	tulot	metsä
C5.0 päätöspuut	27.0%	5.0%	6.1%
Edistetyt C5.0 päätöspuut	24.6%	4.6%	3.4%
C5.0 säännöt	26.0%	5.0%	5.9%
Edistetyt C5.0 säännöt	24.5%	4.5%	3.4%

Taulukko 9: Testitapausten virheprosentti ennen edistämistä ja sen jälkeen [Quinlan 1998b]

6. C4.5-algoritmin ongelmia

Wun ja muiden [2008] mukaan monet tiedonlouhinnan ja koneoppimisen tutkijat pitävät päätöspuita ”ratkaistuna ongelmana”, eli luokittelutapana, jonka ongelmat on suurimmaksi osaksi ratkaistu, eikä syvempää tutkimusta ole enää tarpeellista toteuttaa. Wun ja muiden [2008] mukaan C4.5- algoritmista löytyy kuitenkin useita eri puutteita, jotka liittyvät etenkin luotujen päätöspuiden rakenteisiin ja luotettavuuteen. Taulukosta 8 on helppo havaita selkeä ero virheprosenteissa, jotka algoritmi palauttaa käsitellessään toisistaan poikkeavia datasarjoja. Virheprosenttien vaihtelu selittyy C4.5-algoritmin heikkoudesta ennustaa ennen näkemätöntä dataa verrattuna dataan, josta päätöspuu on alun perin luotu. Toinen vaikuttaja on myös C4.5:n spesifisyys: algoritmi saattaa olla huolimatta hyvinkin suuren osan tarjotusta datasta, sillä algoritmin ehdot pitävät niitä merkityksettöminä luokittelun kannalta (ks. uni-datasarjan virheprosentti taulukossa 8).

Toinen merkittävä C4.5:n ongelma ovat monimutkaiset päätöspuut. Suurten datasarjojen kohdalla algoritmi saattaa palauttaa varsin monimutkaisia päätöspuita, joita on vaikea ymmärtää. Olisiko monimutkaisen puurakenteen

mahdollista pilkkoa pienempiin ja ymmärrettävämpiin osiin? Mikäli monimutkainen päätöspuu olisi mahdollista pilkkoa ja arvottaa puun alhaalta ylöspäin kulkien, tarpeettoman monimutkaisista päätöspuista voitaisiin luoda useampi, pienempi ja helpommin sisäistettävämpi päätöspuu, joka kuitenkin vastaa alkuperäistä puuta [Wu *et al.* 2008].

7. Yhteenveto

Koneoppiminen on kehitetty käsittelemään ympärillämme olevia yhä kasvavia datamääriä mahdollisimman automaattisesti ja itseoppivasti. Koneoppimisen oppimisprosessissa mukaillaan ihmiselle ja eläimille ominaisia oppimismetodeja. Yksi tehokkaista ja suosituista suurien datamäärien analysointi- ja luokittelurakenteista ovat päätöspuut. Päätöspuu on vuokaavioita mukaileva tietorakenne, jossa solmukohdat vastaavat syötetyn datasarjan ominaisuuksien mukaan luotuja ehtoja ja puun lehdet luokkia, joihin tietoa tullaan luokittelemaan. C4.5-algoritmi luo päätöspuita sille syötetystä opetusaineistosta, ja kykenee datan ominaisuuksista oppimiensa tietojen perusteella luokittelemaan uusia datatietueita ennalta määriteltyihin luokkiin.

C4.5-algoritmin toiminta perustuu informaatioteoriaan, jossa käytetään haje- ja saantilaskelmia luokittelun tarkentamiseksi ja tehostamiseksi. C5.0 on C4.5:n kehittyneempi versio, jossa on lukuisia uusia luokittelua helpottavia ja tehostavia ominaisuuksia. C4.5- ja C5.0-algoritmeja pidetään monipuolisina ja onnistuneina luokittelumenetelminä, jotka eivät enää juurikaan haasta koneoppimisen ja tiedonlouhinnan asiantuntijoita spesifimmän tutkimuksen harjoittamiseen. Algoritmeissa on kuitenkin myös heikkouksia.

Viitteet

Hettich, Seth, and Bay, Stephen D. 1999. The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.

Ilyes, Jenhani, Amor, Nahla Ben and Elouedi, Zied. 2008. Decision trees as possibilistic classifiers. *International Journal of Approximate Reasoning* 48.3: 784-807.

Ingargiola, Giorgio. 1996. Building classification models: ID3 and C4.5. Available 5. Available in WWW-format: http://www. cis. temple. edu/~ ingargio/ cis587/readings/id3-c45. html.

Marsland, Stephen. 2015. *Machine Learning: An Algorithmic Perspective*. CRC Press.

Pandya, Rutvija, and Pandya, Jayati. 2015. C5.0 Algorithm to improved decision tree with feature selection and reduced error pruning. *International Journal of Computer Applications*, 117.

Quinlan, J. Ross. 1996a. Bagging, boosting, and C4. 5. In: *AAAI/IAAI*, Vol. 1. 725-730.

Quinlan, J. Ross. 1996b. Improved use of continuous attributes in C4. 5. *Journal of Artificial Intelligence Research 4*, 77-90.

Quinlan, J. Ross. 1998a. C5. 0: An Informal Tutorial. RuleQuest.

Quinlan, J. Ross. 1998b. Is See5/C5 better than C4.5? Rulequest Research. http://www.rulequest.com/see5-comparison.html

Quinlan, J. Ross. C4. 5: *Programs for Machine Learning*. Elsevier, 2014.

Smola, Alex, and S.V.N. Vishwanathan. 2008. *Introduction to Machine Learning*. Cambridge University Press.

Wu, Xindong, et al. 2008. Top 10 algorithms in data mining. *Knowledge and Information Systems* 14.1, 1-37.

Monen agentin polunetsintä

Aapo Nikkilä

Tiivistelmä.
Monen agentin polunetsintä on ongelma, jossa monelle eri agentille pyritään löytämään yksilölliset reitit graafissa aloitussolmuista tavoitesolmuihin niin, ettei yhtä useampi agentti ole samassa solmussa samaan aikaan. Tässä tutkielmassa tarkastelen ongelman ratkaisuun erilaisia lähestymistapoja, kuten yhdistetyn ja erotetun polunetsinnän sekä niiden vahvuuksia ja heikkouksia.

Avainsanat ja -sanonnat: Polunetsintä, graafi, heuristiikka, hierarkkisuus.

1. Johdanto

Polunetsintä (pathfinding) on ongelma, jossa pyritään löytämään peräkkäisten tilojen polku määrätystä lähtötilasta toiseen päätöstilaan. Polunetsintää tarvitaan muun muassa logistiikassa, robotiikassa ja videopeleissä, joissa polunetsinnällä pyritään löytämään agentille vapaa, kulkemiskelpoinen ja yleensä mahdollisimman lyhyt reitti omasta sijainnistaan johonkin tavoiteltuun sijaintiin. Polunetsinnän hakualueena on graafi, jossa polkua etsitään solmuja ja niiden välisiä kaaria pitkin.

Usein reaaliaikaisissa sovelluksissa tietokoneen resurssit ovat käytössä monessa muussa tehtävässä, kuten videopeleissä grafiikan tai tekoälyn prosessoinnissa. Polunetsinnälle jäävät resurssit ovat siis usein pienet. Siksi on tarpeen kehittää mahdollisimman tehokkaita polunetsintäalgoritmeja.

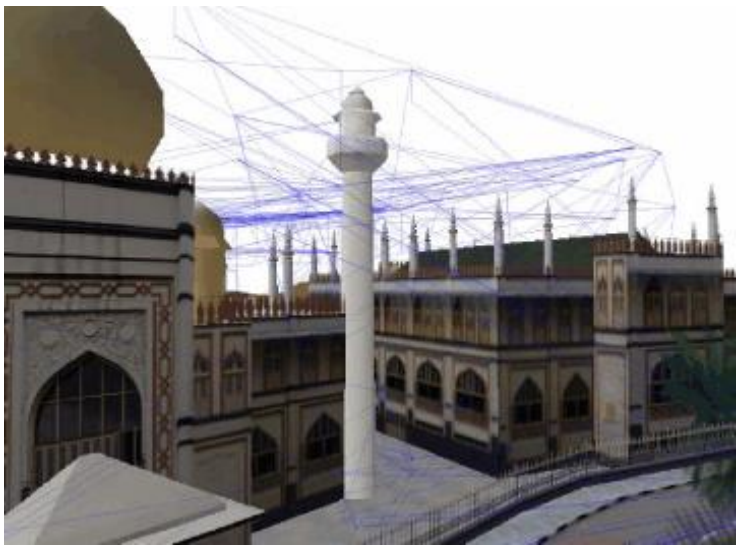
Monen agentin polunetsintä (multi-agent pathfinding) on ongelma, jossa monelle eri agentille pyritään löytämään reittinsä omiin määränpäihinsä yhteisessä ympäristössä ilman agenttien välisiä törmäyksiä [Silver 2005]. Käytettäessä yksinkertaisia, staattiseen ympäristöön suunniteltuja algoritmeja polkuja jouduttaisiin laskemaan kokonaisuudessaan aina uudestaan agenttien välisten törmäysten ilmetessä. Siksi monen agentin polunetsintään on kehitetty tehokkaampia menetelmiä ratkaisemaan nimenomaan tätä ongelmaa.

Tässä tutkielmassa käsittelen monen agentin polunetsintään liittyviä ongelmia ja lähestymistapoja niiden ratkaisemiseen. Kontekstia pohjustaakseni aloitan luvussa 2 katsastuksella muutamaan polunetsinnässä käytettävään tietorakennetoteutukseen. Luvuissa 3 ja 4 käsittelen heuristiikan ja hierarkkisuuden merkitystä polunetsinnässä. Lopulta luvuissa 5 ja 6 käsittelen lähestymistapoja monen agentin polunetsintään, kuten yhdistetyn ja erotetun lähestymistavan eron, joiden yhteydessä esittelen hieman tarkemmin kahden näitä lähestymistapoja noudattavan algoritmin toimintaa.

2. Polunetsinnässä käytettäviä tietorakenteita

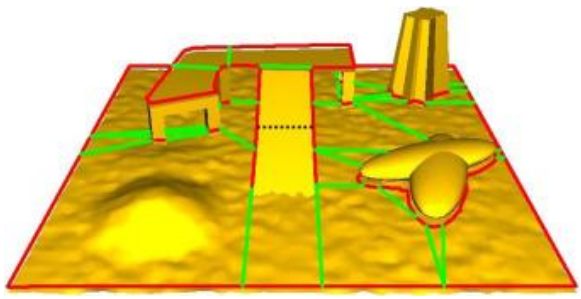
Polunetsintäalgoritmit toimivat graafin tapaisissa tietorakenteissa. Geneeriselle, korkean tason algoritmille siis yleensä riittää, että sille kerrotaan aloitustila, tavoiteltu päätetila ja tilojen väliset yhteydet. Todellisuudessa sovelluksessa käytettävä tietorakenne on kuitenkin usein yksinkertaista graafia monimuotoisempi. Tässä luvussa esittelen muutaman sellaisen yleisesti käytetyn tietorakenteen.

Fyysisen maaston esityksen sisältävässä sovelluksessa, kuten videopelissä, maastonmuotoja kuvaamaan voitaisiin muodostaa näkölinjoihin perustuva graafi (esim. Wardhanan ja muiden [2013] *waypoint graph*). Tällaisen graafin solmut vastaavat sijainteja maastossa ja kahden solmun välillä on kaari, mikäli solmujen välissä ei ole kulkemista estäviä esteitä. Graafi on painotettu, ja sen kaaren painoksi katsotaan kaaren solmujen välinen etäisyys, joka voidaan laskea solmujen koordinaattien perusteella.



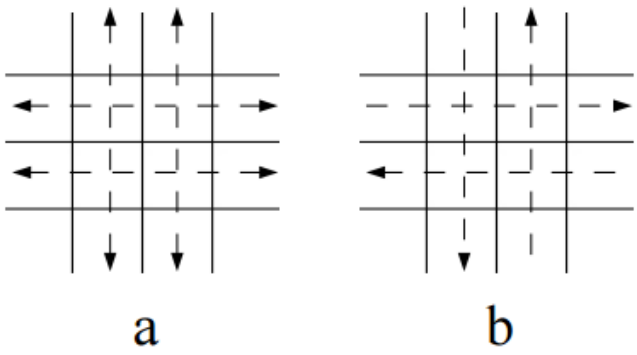
Kuva 1. Näkölinjoihin perustuva graafi [Wardhana *et al.* 2013]. Siniset linjat kuvaavat graafin kaaria.

Toinen, yleisesti etenkin videopeleissä käytetty ympäristön geometriaa mukaileva tietorakenne on *navigaatioverkko* (navigation mesh) (esim. [Oliva and Pelechano 2013]). Navigaatioverkko koostuu geometrisista tasoista ja niiden välisistä yhteyksistä. Tasot kuvaavat alueita, joiden sisäpuolella kulkeminen on esteetöntä, ja kahden tason välillä on yhteys, mikäli tasojen välillä on yhteinen särmä, eli tasolta voidaan kulkea toiselle (ks. kuva 2). Polunetsinnässä navigaatioverkon tasot katsotaan solmuiksi ja niiden väliset yhteydet kaariksi.



Kuva 2. Olivan ja Pelechanon [2013] automaattisesti generoitu navigaatioverkko kolmiulotteisessa ympäristössä. Punaiset ja vihreät linjat ovat tasojen särmiä, joista vihreät ovat kahdelle tasolle yhteisiä, eli yhdistävät kaksi tasoa.

Geneeriset polunetsintäalgorimit voidaan yleensä toteuttaa toimimaan millaisessa graafissa tahansa. Kuitenkin jotkin polunetsintäalgoritmit asettavat rajoitteita graafin rakenteelle. Esimerkiksi Wangin ja Botean [2008] FAR-algoritmi on suunniteltu toimimaan nimenomaan ruudukossa. Ruudukossa jokainen ruutu vastaa graafin solmua, ja vierekkäisten (ja joskus myös kulmittaisten) ruutujen välillä on kaari. FAR merkitsee jokaisen ruudukon rivin ja sarakkeen yksisuuntaiseksi tavoitteenaan välttää agenttien välisiä törmäyksiä keskenään vastakkaissuuntaisessa liikkeessä [Wang and Botea 2008] (ks. kuva 3).



Kuva 3. (a) Tavallinen ruudukko, jossa kulkeminen on sallittu jokaiseen neljään suuntaan. (b) FAR-algoritmin vaatima riveittäin ja sarakkeittain suunnattu ruudukko. [Wang and Botea 2008]

3. Heuristiikka polunetsinnässä

Tehokkaille polunetsintäalgoritmeille keskeinen tavoite on pyrkiä tarkastelemaan mahdollisimman pientä määrää solmuja polun löytämiseksi. Jokaisessa algoritmin silmukassa valittaessa seuraavaksi tarkasteltavaa solmua sen pitäisi siis

osata valita mahdollisimman hyvällä menestyksellä sellainen solmu, joka kuuluu lopulta löydettyyn polkuun. Heuristisen arvioinnin avulla solmujen tiedettyjä ominaisuuksia voidaan hyödyntää sellaisen solmun löytämiseen.

Usein polunetsinnässä heuristista arviointia käytetään solmujen välisten tuntemattomien polkujen pituuksien arvioimiseen (esim. A* [Hart *et al.* 1968]). Hyvän heuristisen funktion muodostamiseen voidaan käyttää hyväksi tietämystä ympäristöstä, jossa polkua etsitään. Sovelluskohteissa, joissa graafi mallintaa luvun 2 tietorakenteiden kaltaisesti sijainteja maastossa, voidaan puhua solmujen välisistä etäisyyksistä. Tällöin yksinkertainen valinta heuristiseksi funktioksi on sellainen, joka palauttaa solmujen välisen etäisyyden linnuntietä, eli toisin sanoen suorinta lyhintä mahdollista tietä solmusta toiseen huolimatta siitä, onko solmujen välillä kaarta tai polkua.

Kahden solmun P ja Q välistä etäisyyttä voidaan mitata esimerkiksi niiden välisellä euklidisella etäisyydellä

$$d(P, Q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2},$$

missä $P = (p_1, \ldots, p_n)$ ja $Q = (q_1, \ldots, q_n)$ ovat n-ulotteisen avaruuden pisteitä. Euklidisen etäisyyden sijaan vaihtoehtoinen etäisyyden mitta on Manhattan-etäisyys

$$d(P, Q) = \sum_{i=1}^{n} |p_i - q_i|.$$

Manhattan-etäisyys on sopiva etäisyyden mittaamiselle etenkin ruudukkomallisessa graafissa, jossa kulkeminen on rajoitettu vain vaaka- ja pystysuuntaiseksi.

A*-algoritmi

A* [Hart *et al.* 1968] (ks. algoritmi 1) on tehokas laajalti käytössä oleva polunetsintäalgoritmi. Algoritmi takaa *täydellisyyden* (completeness), eli lupaa löytää reitin kahden solmun välillä, mikäli sellainen on olemassa. Tulos on myös aina *optimaalinen* (optimal), eli lyhin mahdollinen reitti, mikäli algoritmissa käytetty heuristiikka on *pätevä* (admissible). Heuristiikka on pätevä, mikäli se ei koskaan yliarvioi lyhimmän solmujen välisen polun pituutta.

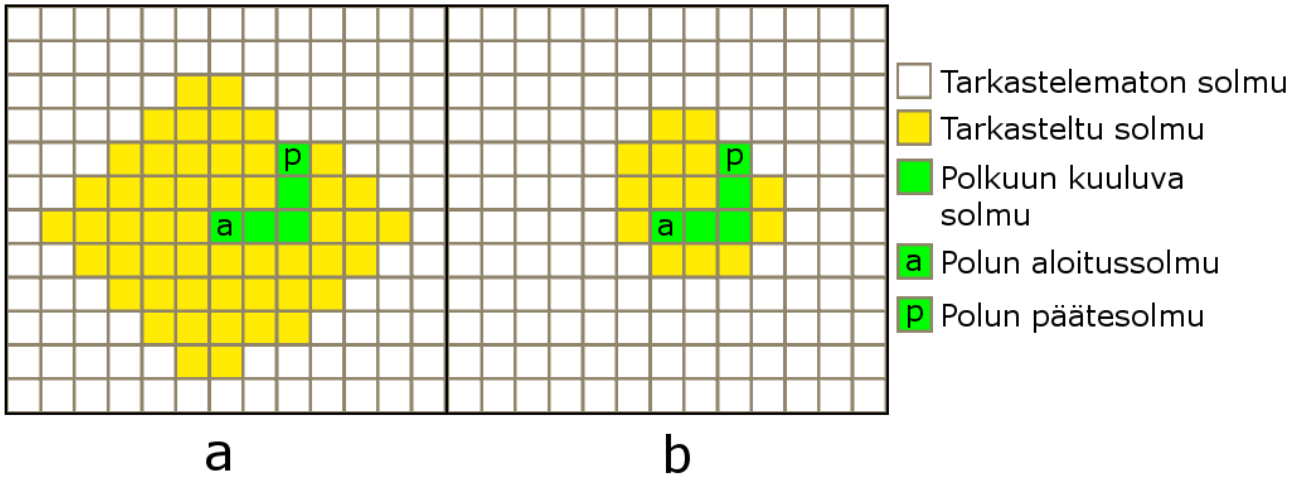
Algoritmille määritellään heuristiikka, jonka mukaan se valitsee seuraavaksi tarkasteltavan solmun jokaisella kierroksellaan. Hyvän heuristiikan valinta on oleellista algoritmin tehokkuuden kannalta. Monet A*-algoritmiin pohjautuvat monimutkaisemmat algoritmit pyrkivätkin tehokkuuteen muun muassa hyvän heuristiikan muodostamisella.

```
A*-algoritmi(aloitussolmu a, päätesolmu p)
    Merkitään a avoimeksi
    g[a] <- 0
    f[a] <- Heuristiikka(a, p)
    Kunnes yksikään solmu ei ole avoinna:
        Solmu s <- Solmu, jolla on pienin f-arvo
        Jos s == p:
            Palautetaan löydetty polku, joka voidaan rakentaa seuraamalla
            jokaiseen solmuun merkittyä edeltäjää. Lopetetaan algoritmin
            suoritus.
        Merkitään s suljetuksi
        Jokaiselle solmun s naapurille n, joka ei ole suljettu:
            Merkitään n avoimeksi
            x <- g[s] + Solmujen s ja n välisen kaaren pituus
            Jos x < g[n]:
                g[n] <- x
                f[n] <- g[n] + Heuristiikka(n, p)
                Merkitään s solmun n edeltäjäksi
    Palautetaan virhe: Solmujen a ja p välille ei ole polkua.
    Lopetetaan algoritmin suoritus.
```

Algoritmi 1. A* [Hart *et al.* 1968]

Silmukassaan algoritmi valitsee avoimista solmuista seuraavaksi tarkasteltavan solmun s heuristiikan avulla valitsemalla sen avoimen solmun, jolla on pienin siihen mennessä laskettu f-arvo. Solmun f-arvo lasketaan lisäämällä heuristisen funktion palauttama luku solmun g-arvoon, joka on aloitussolmun a ja solmun s välisen polun lyhin siihen mennessä laskettu pituus.

A* johtaa heuristiikan valinnan puitteissa mahdollisimman pieneen tarkasteltujen solmujen määrään. Ilman heuristiikkaa A* hakisi solmuja sokeasti [Holte *et al.* 1996] ja näin vastaisi toiminnaltaan alkuperäistä Dijkstran [1959] polunetsintäalgoritmia, joka ei arvioi, kuinka lähellä päätesolmua tarkasteltava solmu on. Kuvasta 4 ilmenee hyvin, miten tarkasteltujen solmujen määrä pienenee, kun algoritmissa käytetään sopivaa heuristiikkaa. Kuvassa 4a heuristiikkaa ei ole, eli voidaan sanoa, että heuristinen funktio palauttaa aina saman arvon riippumatta sille annetuista parametreista. Kuvassa 4b taas heuristisena funktiona on käytetty solmujen eli ruutujen välistä Manhattan-etäisyyttä.



Kuva 4. Vertailu erään A*-algoritmin toteutuksen ajosta yhdelle agentille esteettömässä ruudukossa ilman heuristiikkaa (a) ja sen kanssa (b).

A*-algoritmia käytetään edelleen laajalti. Sovelluksissa, joissa polunetsinnälle on monipuoliset vaatimukset, kuten dynaamisesti muuttuva ympäristö tai monta huomioonotettavaa agenttia, on kuitenkin A*-algoritmia älykkäämmille menetelmille tarvetta. Näissäkin monimutkaisemmissa menetelmissä, joita tässä tutkielmassa tarkastelen, A* tai sen kaltainen algoritmi on usein käytössä jollain tasolla (esim. WHCA* [Silver 2005] ja FAR [Wang and Botea 2008]).

4. Hierarkkinen polunetsintä

Hierarkkisessa polunetsinnässä hakualueesta muodostetaan abstrakti korkeamman tason kuvaus. Esimerkiksi ruutuihin jaetussa maailmassa neljän vierekkäisen ruudun voidaan katsoa muodostavan yhden ison, abstraktin ruudun, joka kattaa näiden neljän ruudun alueen [Sturtevant and Buro 2005]. Abstrahointi ei kuitenkaan aina tarkoita tällaista solmujen klusterointia, vaan se voi olla mitä tahansa hakualueen yksinkertaistamista tai yleistämistä. Monen agentin polunetsintäalgoritmissa hierarkkisuus voi tarkoittaa esimerkiksi hakualueen abstrahointia niin, ettei siinä oteta muita agentteja huomioon. Esimerkiksi luvussa 6 tarkastelemissani Silverin [2005] sekä Sharonin ja muiden [2015] menetelmissä on käytössä tämänkaltainen idea.

Abstrahoinnin avulla polunetsinnän hakualuetta voidaan tarkastella korkeammalta tasolta, jonka käsittely on alkuperäisen graafin käsittelyä kevyempää vähäisemmän monimutkaisuuden ansiosta. Korkean tason tiedolla ei kuitenkaan voida suoraan muodostaa tarkkoja polkuja alimmalle, todelliselle tasolle. Korkean tason mallia käytetään apuna hahmottelemaan polku, jota käytetään apuna matalimman tason polunetsinnässä. Abstrahoitua hakualueen kuvausta voidaan käyttää hyväksi muun muassa heuristiikkaa hyödyntävän algoritmin,

kuten A*:n heuristiikkafunktion muodostamiseen [Holte *et al.* 1996]. Polkua etsiessä lähtöpisteestä tavoitepisteeseen voidaan siis laskea polku karkeasti mutta nopeasti korkealla tasolla, ja tähän korkean tason polkuun voidaan viitata heuristiikkafunktiossa, jossa arvioidaan pisteen etäisyyttä tavoitepisteeseen matalalla tasolla. Esimerkiksi Silverin [2005] algoritmissa (ks. WHCA* kohdassa 6.2) abstrahoidussa hakualueessa lasketun polun pituutta käytetään suoraan heuristiikkana varsinaisen polun etsimisessä.

Sovelluksissa, joissa agentti kulkee polunetsinnällä löydettyä reittiä rajatulla nopeudella, on mahdollista, että koko polkua ei pyritä löytämään kokonaan välittömästi, vaan sen sijaan polkua rakennetaan sitä mukaa, kun sillä edetään. Näin polun laskenta jaetaan pitkälle aikavälille, ja samalla lomitetaan toiminta ja laskenta ja niin vähennetään polun uudelleenlaskemista dynaamisen ympäristön muuttuessa [Sturtevant and Buro 2005]. Tällainen toteutustapa kuitenkin vaatii sen, että polkua lähdetään alun perin etsimään oikeaan suuntaan. Sturtevantin ja Buron [2005] Partial-Refinement A* (PRA*) -algoritmi käyttää abstrahointia täyttääkseen tämän vaatimuksen muodostaen ensin abstrahoidussa hakualueen mallissa korkean tason polun, jota käytetään ohjaamaan matalan tason polunetsintää.

Hakualue voidaan menetelmästä ja tarpeesta riippuen abstrahoida useammalla kuin yhdellä tasolla. Se voidaan sovelluskohteesta ja abstrahoinnin luonteesta riippuen tehdä manuaalisesti tai automaattisesti. Esimerkiksi käsin suunniteltua videopelimaailmaa mallintavasta graafista voidaan muodostaa abstrakti malli manuaalisesti esimerkiksi muodostamalla solmuista klustereita. Mikäli kenttä on proseduraalisesti generoitu, on vastaava abstrahointi luultavasti tarpeen suorittaa automaattisesti.

5. Yhteistyön taso monen agentin polunetsinnässä

Silver [2005] ehdottaa monen agentin polunetsinnälle kolme erilaista mahdollista yhteistyön tasoa. *Yhteistoiminnallisessa* (cooperative) polunetsinnässä jokainen agentti tietää kaikkien muiden agenttien suunnitelmat. Polunetsinnän tavoitteena on tällöin yleensä minimoida kaikkien polkujen kokonaiskustannus, jolloin on tavallista, että yksittäinen agentti kulkee lyhimmän reitin sijaan pidemmän reitin välttääkseen yhteentörmäyksen toisen agentin kanssa [Bnaya *et al.* 2013]. *Ei-yhteistoiminnallisessa* (non-cooperative) polunetsinnässä agentit eivät tiedä muiden agenttien suunnitelmista, vaan agentit joutuvat ennakoimaan muiden agenttien toimintaa. *Antagonistisessa* (antagonistic) polunetsinnässä agentit pyrkivät tavoitteisiinsa estäen samalla muiden agenttien pääsyä omiin tavoitteisiinsa. [Silver 2005]

Polunetsinnän vaatimukset voivat muodostua monista tekijöistä. Kuitenkin ensimmäisenä rajoitteena on se, onko tieto muista agenteista ensinkään saatavilla. Esimerkiksi videopeliin on yleensä mahdollista toteuttaa yhteistoiminnallinen polunetsintäalgoritmi, koska teknisestä näkökulmasta jokaisen agentin tiedot ovat täysin saatavilla kaikille muillekin agenteille. Sen sijaan esimerkiksi sovelluksessa, jossa agenttien polunetsintä on hajautettu useampaan agenttikohtaiseen prosessointiyksikköön, on agenttien välinen tiedon jakaminen rajoitettua.

Useimmat tässä tutkielmassa käsitellyt monen agentin polunetsintämenetelmät ovat yhteistoiminnallisia. Bnaya ja muut [2013] esittävät kuitenkin ratkaisun poikkeavaan lähestymistapaan, jossa agentit toimivat itsekkäästi pyrkien vain parhaaseen mahdolliseen yksilölliseen polunetsinnän tulokseen perinteisin menetelmin, kuten A*-algoritmin avulla. Ehdotettu verotusjärjestelmä kannustaa agentteja välttämään törmäyksiä asettamalla tiettyjen solmujen tai kaarien läpi kulkemiselle ylimääräisiä kustannuksia. Lisäkustannuksia luodaan iteratiivisesti agenttien välisten törmäysten tapahtuessa.

Useat yhteistoiminnalliset monen agentin polunetsintämenetelmät tuovat polkuihin mukaan aikaulottuvuuden, mikä tarkoittaa sitä, että polun solmut sidotaan johonkin ajankohtaan (esim. WHCA* [Silver 2005] ja FAR [Wang and Botea 2008]). Siten polkua laskettaessa solmu voidaan katsoa vapaaksi, mikäli se ei kuulu mihinkään muuhun polkuun tietyllä ajankohdalla. Kun agentin polku kertoo sen ajankohdan, jolle kukin sen solmuista on varattu, voivat muut agentit ottaa aiemmin lasketun polun huomioon jo omassa polunetsintävaiheessa. Mikäli solmujen ajankohdat eivät ole tiedossa, törmäykset voidaan havaita vasta, kun agentit todella törmäävät kulkiessaan laskettuja polkujaan.

Puhtaasti graafissa toimivassa polunetsintäalgoritmissa agentit liikkuvat solmusta toiseen niin, että agentti varaa aina yhden solmun kerrallaan ja liike solmusta toiseen tapahtuu välittömästi. Kuitenkin useissa sovelluskohteissa, kuten videopeleissä, todellinen aika ja polkua etsivän toimijan liike ei ole diskreettiä vaan jatkuvaa. Siksi aikaa on tarkasteltava diskretisoituna esimerkiksi Silverin [2005] WHCA*-algoritmia varten.

6. Yhdistetty ja erotettu monen agentin polunetsintä

Monen agentin polunetsinnän menetelmät voidaan luokitella eri kategorioihin sen perusteella, onko ongelman ratkaisu jaoteltu osiin. Sharon ja muut [2015] luokittelevat ratkaisut *keskitettyihin* (centralized) ja *hajautettuihin* (distributed). Keskitetyssä polunetsinnässä kaikki laskenta suoritetaan yksittäisessä prosessointiyksikössä. Hajautetussa polunetsinnässä taas jokainen agentti suorittaa oman polunetsintänsä itsenäisesti omassa yksikössään ja agentit keskustelevat

keskenään erinäisten viestintäkanavien kautta. Tässä tutkielmassa keskityn menetelmiin, jotka kuuluvat keskitetyn polunetsinnän kategoriaan.

Keskitetty polunetsintä voidaan edelleen jaotella *yhdistettyihin* (coupled) ja *erotettuihin* (decoupled) menetelmiin [Sharon *et al.* 2013]. Yhdistetyssä lähestymistavassa monen agentin polunetsinnän ongelma muotoillaan yksittäiseksi ongelmaksi, jossa yksi päätöksentekijä laskee kaikkien agenttien polut samanaikaisesti. Erotetussa lähestymistavassa jokaisen agentin polut lasketaan erikseen käsitellen niitä itsenäisinä ongelmina.

Lähteestä riippuen edellä mainitsemilleni termeille on toisinaan toisistaan eroavia määritelmiä. Tässä tutkielmassa noudatan kuitenkin tätä termistöä niin kuin Sharon ja muut [2013, 2015] ovat ne määritelleet.

6.1. Yhdistetty lähestymistapa

Yhdistetyssä polunetsinnässä monen agentin polunetsinnän ongelma muotoillaan yksittäiseksi ongelmaksi, jossa kaikkien agenttien polunetsinnän suorittaa yksittäinen päätöksentekijä. Voidaan siis esimerkiksi sanoa, että yhdistetyn monen agentin polunetsintäalgoritmin kukin tila koostuu kaikkien agenttien sijainneista ja niiden ajankohdista yhdessä. Tällöin ongelma on ratkaistavissa lähes millä tahansa geneerisellä polunetsintäalgoritmilla, kuten esimerkiksi A*-algoritmilla, joka tyypillisesti nähdään yhden agentin polunetsintäalgoritmina. A*-algoritmilla päästäisiin optimaalisiin tuloksiin, mutta se olisi epäkäytännöllisen tehoton suurella määrällä agentteja, koska sen kompleksisuus kasvaisi eksponentiaalisesti agenttimäärän kasvaessa [Sharon *et al.* 2015].

Yhdistetty lähestymistapa vaaditaan, mikäli halutaan optimaalisia tuloksia [Khorshid *et al.* 2011]. Menetelmä voi kuitenkin olla yhdistetty ja silti tuottaa muita kuin optimaalisia tuloksia tehokkuuden nimissä. Esimerkiksi Khorshidin ja muiden [2011] Tree-based Agent Swapping Strategy (TASS) on yhdistetty polunetsintämenetelmä, joka löytää agenteille polut puussa polynomisessa ajassa.

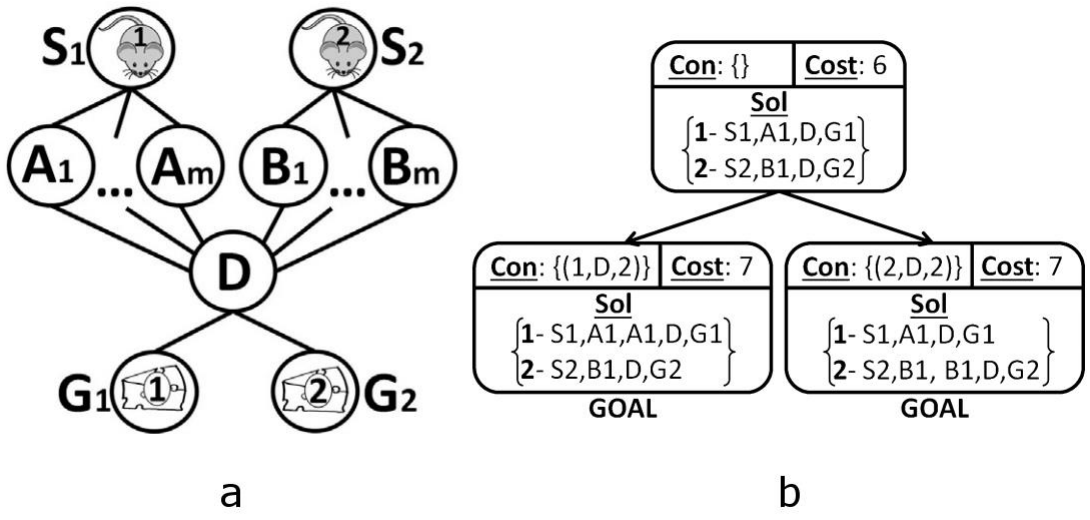
Conflict Based Search

Sharonin ja muiden [2015] Conflict Based Search -algoritmi (CBS) ratkaisee monen agentin polunetsinnän erottamalla sen kahdelle tasolle. Matalalla tasolla agenteille etsitään polkuja tavallisella yhden agentin polunetsintäalgoritmilla. Korkealla tasolla ratkaistaan agenttien väliset konfliktit, eli matalalla tasolla sattuneet agenttien polkujen risteymät. Konfliktien perusteella muodostetaan uusia rajoitteita, joiden rajoittamana matalalla tasolla polkuja lasketaan uudelleen, jottei samaa törmäystä tapahdu seuraavalla polunetsintäkerralla. Konfliktien rat-

komista, rajoitteiden muodostamista ja matalalla tasolla polkujen uudelleenlaskemista suoritetaan toistuvasti, kunnes konfliktiton ratkaisu ongelmaan on löytynyt.

Kuvassa 5 on esimerkki CBS-algoritmin ajon ensimmäisestä kierroksesta korkealla tasolla. Kuvan 5b rajoitepuun solmut sisältävät konflikteja (”Con”), kustannuksia eli yhteenlaskettuja polkujen pituuksia (”Cost”) ja ratkaisuja eli kaikkien agenttien polkuja yhdessä (”Sol”). Uusien konfliktien tapauksessa solmulle luodaan lapsia, joissa konflikti ratkaistaan keskenään eri tavoin. Esimerkissä juuressa ilmenee konflikti solmussa D, jossa kaksi agenttia suunnittelevat olevansa samaan aikaan. Konflikti on ratkaistu luomalla juurelle kaksi lasta, joista vasemmanpuolimmaisessa agentille 1 lasketaan uusi konfliktin välttävä polku, mutta agentin 2 polkua ei muuteta. Oikeanpuoleisessa lapsessa sama tehdään toisin päin. Eri tavoin ratkotuista ratkaisuista suositaan sitä, jonka yleiskustannus on pienin. Esimerkin tapauksessa rajoitepuun toisella tasolla ei enää ilmene konflikteja, joten algoritmin suoritus päättyy.

Vaikka CBS onkin korkealla tasolla yhdistetty polunetsintämenetelmä, sen voidaan nähdä yhdistävän erotettua ja yhdistettyä lähestymistapaa siinä mielessä, että matalan tason polunetsintä suoritetaan erotetun lähestymistavan kaltaisesti jokaiselle agentille erikseen. Algoritmissa on kuitenkin korkealla tasolla yksittäinen päätöksentekijä, joka pyytää agentteja laskemaan uusia polkuja, joten algoritmin lähestymistapa on yhdistetty.



Kuva 5. **(a)** Aloitustilanne graafissa, jossa agenteille 1 ja 2 tulee löytää vastaavasti reitit S1->G1 ja S2->G2. **(b)** CBS-algoritmin korkean tason rajoitepuu ajon jälkeen. [Sharon *et al.* 2015]

6.2. Erotettu lähestymistapa

Erotetussa lähestymistavassa monen agentin polunetsintää käsitellään yksilöllisesti agenttikohtaisina ongelmina. Erotettu polunetsintäalgoritmi siis suoritetaan jollekin tietylle agentille, ja muiden agenttien huomioonottaminen toteutetaan erinäisillä rajoitteilla. Tämä tarkoittaa yleensä sitä, että agenttien välillä on jaettua tietoa, eli agentit tietävät muiden agenttien suunnitelmista, jotta törmäykset voidaan välttää.

Tätä lähestymistapaa noudattavat menetelmät eivät pyri optimaalisiin tuloksiin, vaan sen sijaan tehokkuuteen optimaalisuuden kustannuksella. Usein erotetut polunetsintämenetelmät eivät myöskään takaa täydellisyyttä [Sharon *et al.* 2013]. Erotettu lähestymistapa on tyypillinen valinta nopeaa suoritusaikaa vaativissa sovelluksissa, joissa agenttien määrä on suuri. Esimerkiksi videopelissä, jossa pelihahmojen polunetsintä täytyy voida suorittaa reaaliajassa, tuloksen optimaalisuus ei usein ole välttämätöntä.

Erotetut polunetsintämenetelmät kykenevät parempaan skaalautuvuuteen kuin yhdistetyt menetelmät. Tällaisten algoritmien suoritusaikaa ja tuloksen laatua on kuitenkin vaikeaa tai jopa mahdotonta arvioida täsmällisesti. Samoin on myös vaikeaa arvioida etukäteen, onko polunetsinnälle olemassa ratkaisua. [Wang and Botea 2009]

Windowed Hierarchical Cooperative A*

Silverin [2005] Windowed Hierarchical Cooperative A* (WHCA*) on A*-algoritmiin perustuva, agenttikohtaisesti suoritettava erotettu monen agentin polunetsintäalgoritmi. Algoritmi välttää agenttien väliset törmäykset agenttien välillä jaetulla varaustaululla, joka sisältää agenttien suunnitellut sijainnit aika-avaruudessa. Varaustauluun siis merkitään agenttien lasketut polut ja jokaisen polun solmun ajankohta.

WHCA* käyttää hierarkkisuutta heuristiikan muodostamiseen. Varsinaisen A*-algoritmilla suoritettavan, agentteja välttelevän polunetsinnän heuristiikaksi lasketaan ensin lyhin polku hakualueen sellaisessa abstrahoidussa mallissa, jossa muita agentteja ei oteta huomioon. Sen polun pituus olisi täsmällinen heuristiikka, jos muita agentteja ei olisi lainkaan. Se on siis selkeästi pätevä heuristiikka, sillä kun muut agentit otetaan huomioon graafin yhteyksien lisärajoitteina, lopullinen lyhin polku voi olla vain yhtä pitkä tai pidempi kuin se, joka laskettiin ottamatta muita agentteja huomioon.

Algoritmi ei pyri optimaalisiin tuloksiin, eikä takaa täydellisyyttä. Algoritmi on kuitenkin tarpeeksi nopea, jotta sitä voidaan käyttää esimerkiksi reaaliaikaisessa videopelissä. Näin WHCA* täyttää erotetun lähestymistavan tyypillisimmät piirteet niin heikkouksiltaan kuin vahvuuksiltaankin.

7. Yhteenveto

Tässä tutkielmassa tarkastelin erilaisia lähestymistapoja ja menetelmiä monen agentin polunetsinnän ongelmaan. Oleellisin huomio on, että mikään menetelmä ei ole muita parempi kaikissa tapauksissa. Menetelmää valitessa ensin on oltava tiedossa, minkälaista tietoa toiminnoistaan agentit voivat jakaa keskenään. Ongelman vaativuuteen vaikuttavat keskeisimpinä hakuavaruuden kompleksisuus ja agenttien määrä. Niiden puitteissa voidaan päättää, vaaditaanko polunetsinnältä optimaalisuutta ja täydellisyyttä vai tavoitellaanko paremmin skaalautuvaa suoritustehokkuutta.

Viiteluettelo

Zahy Bnaya, Roni Stern, Ariel Felner, Roie Zivan and Steven Okamoto. 2013. Multi-agent path finding for self interested agents. In: *Proceedings of the Sixth Annual Symposium on Combinatorial Search 2013*, 38-46.

E. W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1, 269-271.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 2, 100-107.

Robert C. Holte, M. B. Perez, R. M. Zimmer, and A. J. MacDonald. 1996. Hierarchical A*: Searching abstraction hierarchies efficiently. In: *Proceedings of the AAAI-96*, 530-535.

Mokhtar M. Khorshid, Robert C. Holte and Nathan Sturtevant. 2011. A polynomial-time algorithm for non-optimal multi-agent pathfinding. In: *Proceedings of the Fourth International Symposium on Combinatorial Search, SoCS 2011*, 76-83.

R. Oliva, N. Pelechano. 2013. NEOGEN: Near optimal generator of navigation meshes for 3D multi-layered environements. *Computers and Graphics* 37, 5, 403-412.

Guni Sharon, Roni Stern, Meir Goldenberg and Ariel Felner. 2013. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence* 195, 470-495.

Guni Sharon, Roni Stern, Ariel Felner and Nathan R. Sturtevant. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219. 40-66.

David Silver. 2005. Cooperative pathfinding. In: *Proceedings of the AIIDE'05*, 117-122.

Nathan Sturtevant and Michael Buro. 2005. Partial pathfinding using map abstraction and refinement. In: *Proceedings of the AAAI-05*, 1392-1397.

Ko-Hsin Cindy Wang and Adi Botea. 2008. Fast and memory-efficient multi-agent pathfinding. In: *Proceedings of the ICAPS'08*, 380-387.

Ko-Hsin Cindy Wang and Adi Botea. 2009. Tractable multi-agent path planning on grid maps. In: *Proceedings of IJCAI* 9, 1870-1875.

Nicholas Mario Wardhana, Henry Johan and Hock Soon Seah. 2013. Enhanced waypoint graph for surface and volumetric path planning in virtual worlds. *The Visual Computer* 10, 1051-1062.

Kryptografian ja steganografian yhdistäminen uskottavan kiistettävyyden aikaansaamiseksi

Akseli Piilola

Tiivistelmä

Ongelma pelkän kryptografian käyttämisessä datan salaamiseksi on se, että salatun datan olemassaolon todistaminen on yleensä helppoa. Kun datan tiedetään olevan olemassa, voidaan datan haltijaa erilaisin keinoin painostaa luovuttamaan salausavaimet tai muutoin avaamaan tarkastelun kohteena oleva data. Ongelman ratkaisuksi tässä tutkielmassa ehdotetaan kryptografian yhdistämistä steganografiaan siten, että salatun datan olemassaolon luotettava todistaminen muuttuu lähes mahdottomaksi. Tutkielmassa tarkastellaan erityisesti LSB-kuvasteganografiaa, sen havaitsemista ja havainnoinnin estämistä. Tutkielman lopussa ehdotetaan käytännön ratkaisua ja algoritmeja tiedon salaamiseksi ja piilottamiseksi siten, että saavutetaan ns. uskottava kiistettävyys piilotetun datan olemassaolon suhteen.

Avainsanat ja -sanonnat: Steganografia, LSB-kuvasteganografia, steganalyysi, RS-analyysi, datan piilottaminen, uskottava kiistettävyys.

1 Johdanto

Tiedon salaaminen siten, että sitä salauksen purku ilman oikeaa avainta on käytännössä mahdotonta, on nykyaikaisilla menetelmillä kohtuullisen helppoa. Joskus voi kuitenkin tulla vastaan tilanne, jossa salattua tietoa ei saisi löytyä ollenkaan. Tällöin pelkkä salaaminen ei ole riittävä toimenpide, vaan salaamisen lisäksi tieto täytyy myös voida piilottaa. Tässä tutkielmassa keskitytään datan piilottamiseen LSB-kuvasteganografian ja kryptografian yhdistelmällä siten, että salatun datan olemassaolo on vaikeaa tai lähes mahdotonta todistaa. Käytännössä tämä tarkoittaisi esimerkiksi sitä, että kukaan ei voisi pakottaa datan hallussapitäjää antamaan salausavainta tai muutoin avaamaan hallussaan olevaa dataa. Tällainen tilanne voisi konkretisoitua esimerkiksi kuljetettaessa arkaluontoista dataa valtioiden rajojen yli, tai muissa vastaavissa tilanteissa, joissa joudutaan tekemisiin korkean tason viranomaisten kanssa. Etenkin tiettyjen valtioiden rajoittaessa tiedon vapaata liikkuvuutta on hyvin tärkeää kehittää menetelmiä, joiden avulla ihmiset voivat kommunikoida keskenään huomaamattomasti ja luotettavasti.

Varsinaisena tutkimusongelmana tässä tutkielmassa on sellaisen kryptografiaa ja LSB-kuvasteganografiaa hyödyntävän menetelmän löytäminen, jolla saa-

vutetaan uskottava kiistettävyys piilotetun datan olemassaolon suhteen. Menetelmän tulee siis paitsi pystyä piilottamaan dataa kuvatiedostojen sisään LSB-tekniikalla, myös olla vastustuskykyinen tunnettuja havaitsemistekniikoita kohtaan.

Tutkielman edetessä tutustutaan LSB-kuvasteganografian toimintaan, sen havaitsemiseen sekä nykyisten havaitsemistekniikoiden estämiseen. Tutkielman lopussa muodostetaan käytännön menetelmä kryptografian ja steganografian yhdistelmällä siten, että uskottava kiistettävyys saadaan aikaiseksi. Esimerkkikuvissa on datan piilottamiseen käytetty itse kehitettyä yksinkertaista LSB-algoritmia (ks. liitteet 1 ja 2), ja datan salaamiseen Javan standardikirjaston Cipher-luokan AES-lohkosalausmenetelmää 128-bittisellä avaimella.

2 Käsitteitä

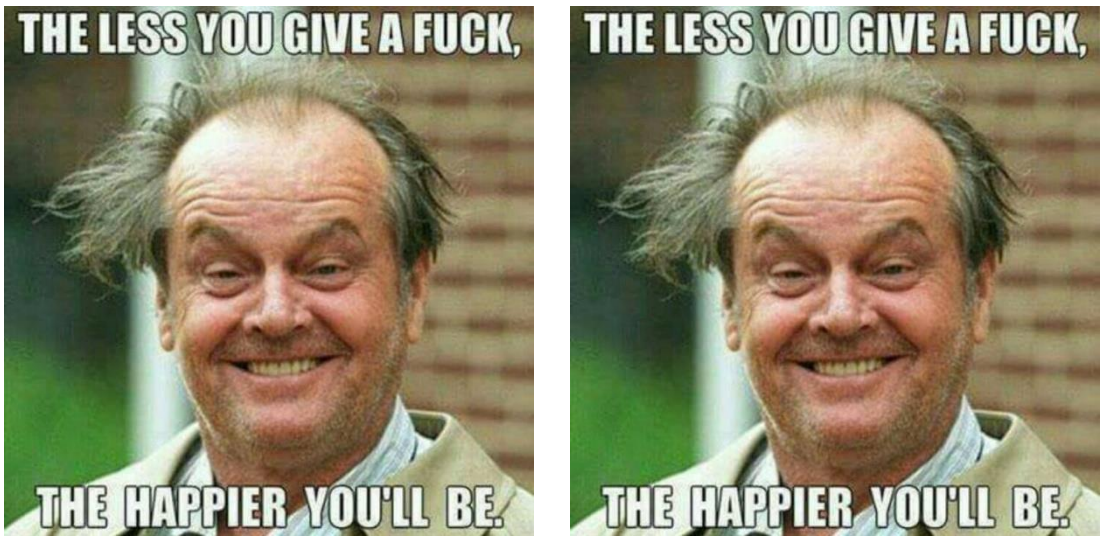
Steganografia (engl. *steganography*) tarkoittaa tiedon piilottamista. Sana juontaa juurensa kreikkalaisista termeistä stegan'os (στεγανός), suomennettuna ”katettu, peitetty tai suojeltu”, sekä graphein (γράφειν), suomennettuna ”kirjoitus". Kansankielellä steganografia on siis salakirjoituksen muoto, jossa salakirjoituksen kohteena oleva teksti tai data varsinaisen salaamisen sijasta piilotetaan siten, että sitä on mahdollisimman vaikea löytää. Vaikka steganografiassa tietoa voidaan piilottaa periaatteessa aivan minne tahansa, niin useimmat nykyaikaiset toteutukset käsittävät datan piilottamista digitaalisesti esimerkiksi kuva-, ääni- tai videotiedoston sisälle. Toisin kuin kryptografia, steganografia itsessään ei sisällä tiedon salaamista. Useimmissa käytännön toteutuksissa käytetään kuitenkin steganografian ja kryptografian yhdistelmää, jossa data ensin salataan ja vasta sitten piilotetaan.

Steganalyysi (*engl. steganalysis*) on tieteenala, joka keskittyy steganografian eri muotojen tutkimiseen sekä piilotetun tiedon löytämiseen. Kryptografiassa vastaava termi on nimeltään kryptoanalyysi. Käytännössä steganalyysillä pyritään murtamaan samoja suojauksia, joita steganografialla yritetään saavuttaa. Steganalyysiä käytetään siis myös ”hyvään”, eli haavoittuvuuksien etsimiseen olemassa olevien algoritmien kehittämiseksi).

Uskottava kiistettävyys (engl. *plausible deniability*) tarkoittaa tässä yhteydessä sitä, että henkilö voi uskottavasti kiistää hallussaan olevien tiedostojen sisältävän piilotettua dataa. Tämä on tietyin edellytyksin mahdollista steganografian ja kryptografian yhdistelmän oikeanlaisella toteutuksella, kuten myöhemmin osoitetaan. Käytännössä tämä tarkoittaa siis sitä, että kukaan ei voi riittävän luotettavasti osoittaa piilotetun datan olemassaoloa.

3 LSB-kuvasteganografia

Tässä tutkielmassa keskitytään steganografian osalta kuvasteganografiaan, ja vielä tarkemmin LSB-tekniikkaan. LSB-kuvasteganografia on steganografian muoto, jossa dataa piilotetaan kuvatiedoston sisälle korvaamalla pikseleiden väriarvojen vähiten merkitsevät bitit (engl. *least significant bit*) piilotettavan datan biteillä. Vähiten merkitsevän bitin muuttaminen muuttaa pikselin väriä niin vähän, ettei muutosta käytännössä pysty ihmissilmällä havaitsemaan.



Kuva 1: Vasemmalla alkuperäinen kuva ja oikealla stegokuva, johon on piilotettu dataa lähes 100% täyttöasteella.

Kuvassa 1 on esimerkki kuvaparista, jossa toiseen on LSB-tekniikalla piilotettu dataa lähes 100% täyttöasteella. Kuvat näyttävät silmämääräisesti samoilta.

True color RGB -väriskaalalla pikselin väriarvo kuvataan 24-bittisellä luvulla. Luvun ensimmäiset kahdeksan bittiä kertovat punaisen värin arvon, toiset kahdeksan bittiä vihreän värin arvon ja kolmannet kahdeksan bittiä sinisen värin arvon. 32-bittisellä RGBA-skaalalla lukuun lisätään vielä kahdeksan bittinen alpha-arvo, joka kertoo värin läpinäkyvyyden. Tästä eteenpäin puhuttaessa pikseleistä, tarkoitetaan nimenomaan 32-bittisen RGBA-väriarvon omaavaa pakkaamattoman kuvatiedoston (bitmap) pikseliä.

RGBA-väriskaalan bittijärjestys saattaa hieman vaihdella järjestelmästä ja sovelluksesta riippuen. Pääsääntöisesti kuitenkin alpha-arvo on joko ensimmäisenä tai viimeisenä. Värien järjestyksellä ei käytännössä muuten ole väliä, mutta alpha-arvon sijainti täytyy LSB-steganografiaa ja jatkoa ajatellen tuntea. Kuvassa 2 nähdään RGBA-arvon esitys bitteinä. Bitit 16, 8 ja 0 ovat LSB-bittejä, jotka voidaan korvata piilotettavan datan biteillä.

Pituus	8								8								8								8							
Kohde	Alpha								Red								Green								Blue							
Bitti#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Kuva 2: RGBA-arvo bittiesityksenä.

LSB-steganografiassa voidaan nimensä mukaisesti jokaisen pikselin RGBA-väriarvoista korvata erikseen jokaisen värin vähiten merkitsevä (vasemmalta katsoen viimeinen ja lukuarvoltaan pienin) bitti. Mahdollisiin alpha-arvon bitteihin ei kosketa ollenkaan, sillä satunnainen läpinäkyvyyden muuttaminen olisi hyvin helppo havaita jo kaikkein yksinkertaisimmillakin koneellisilla analyyseillä. Korvaaminen rajataan jokaisen värin viimeiseen bittiin jo menetelmän nimenkin perusteella, minkä lisäksi käytännössä enemmän merkitsevien bittien korvaaminen aiheuttaisi ennen pitkää myös kuvan näkyvän vääristymisen. Käytännössä LSB-menetelmä siis mahdollistaa maksimissaan kolme bittiä tallennustilaa jokaista kuvatiedoston pikseliä kohden. Tällöin esimerkiksi 1024*768 kokoisen kuvan sisään olisi mahdollista piilottaa maksimissaan n. 288 kilotavua dataa. Kuvassa 3 on esitetty kahdeksan peräkkäistä pikseliä bittimuodossa. Luettaessa LSB-bitit rivi riviltä vasemmalta oikealle, saadaan tulokseksi bittijono 1111 1110 1101 0001 1001 1110.

Pituus	8								8								8								8							
Kohde	Alpha								Red								Green								Blue							
Pikseli #1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	0	1	0	1	0	1	1	1	0	1	0	1	0	0	1	1
Pikseli #2	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	1	1	1	1	0	1	1	1	0	1	0	1
Pikseli #3	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1	1	0	1	1	0	1	0	1	0	1
Pikseli #4	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	0	1	0	0	0	1	1
Pikseli #5	1	1	1	1	1	1	1	1	0	1	1	1	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0
Pikseli #6	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	1	0	0	0	1	0	0	1	0	1	1	1	1	1	0	0
Pikseli #7	1	1	1	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0	1	0	1	0	1	1
Pikseli #8	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	1	0	0	0	1	0	1	1	1	0	1	0	0	1	0	0
Bitti#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Kuva 3: Esimerkki datan piilottamisesta LSB-menetelmällä.

Kuvaa, johon data piilotetaan, kutsutaan säiliökuvaksi (engl. *cover photo*). Säiliökuvien yhteydessä voidaan puhua täyttöasteesta, joka kertoo LSB-tasolta prosentuaalisesti korvattujen bittien määrän. Mikäli kuvan jokaisen värin LSB-tason jokainen bitti korvataan piilotettavan datan biteillä, on täyttöaste 100%. Luonnollisesti mitä isompaa täyttöastetta käytetään, sitä enemmän kuva myös vääristyy. Vaikka näitä vääristymisiä ei silmämääräisesti kuvaa katsomalla huomaisikaan, saattaa liian iso täyttöaste tietyissä tapauksissa helpottaa piilotetun tiedon havaitsemista.

4 LSB-Steganografian havaitseminen

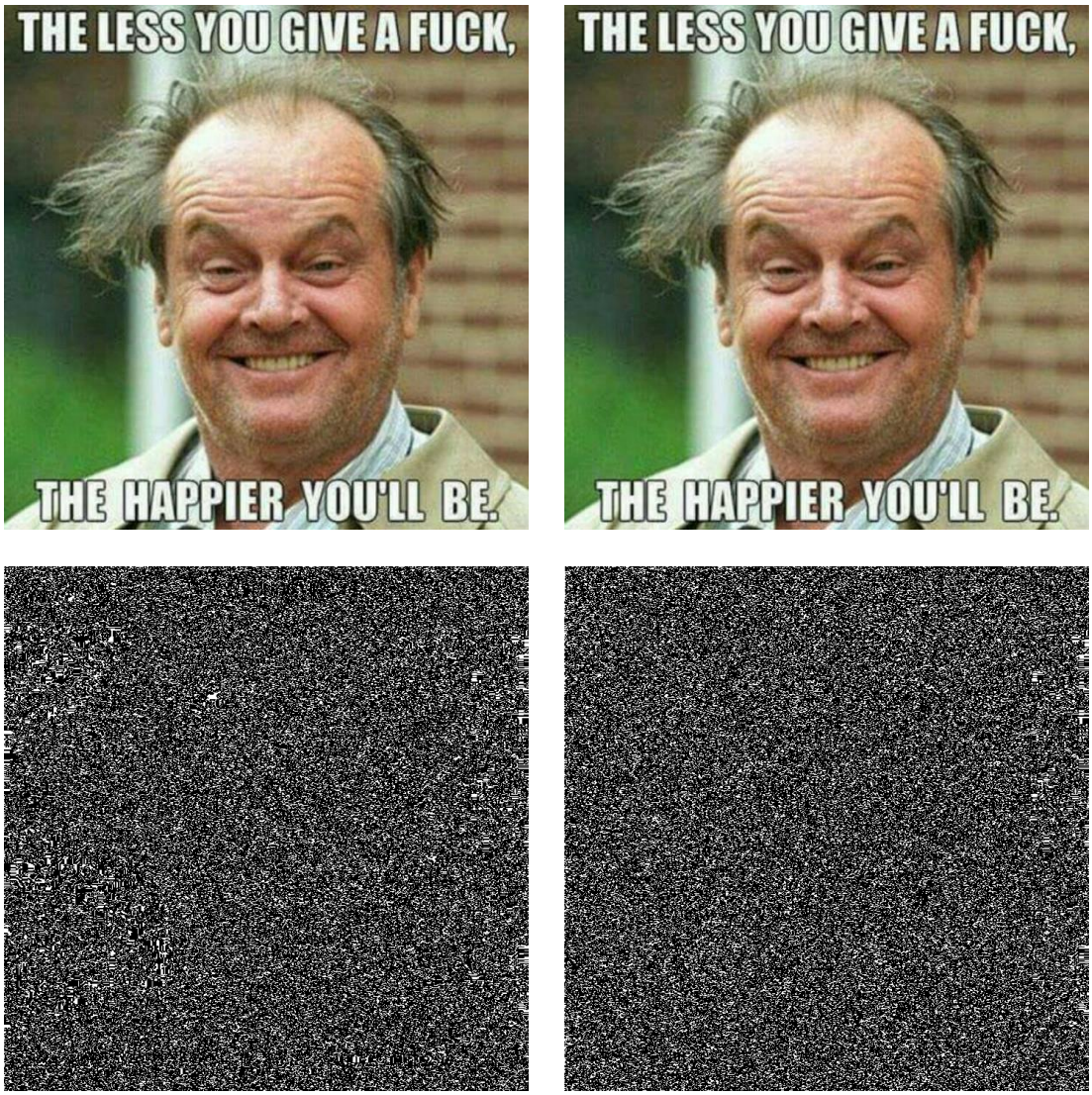
Käytännössä LSB-steganografian havaitseminen on mahdollista analysoimalla säiliötiedostoa (engl. *cover image*). Havaitsemiseen on olemassa useita erilaisia analysointitekniikoita, jotka voidaan jakaa kahteen eri ryhmään: visuaalinen steganalyysi ja kvantitatiivinen steganalyysi. Visuaaliseen analysointiin kuuluvat kaikki tekniikat, joissa kuvatiedostoa tarkastellaan manuaalisesti ja esimerkiksi verrataan muihin kuvatiedostoihin, kun taas kvantitatiiviseen analyysiin kuuluvat tekniikat, jotka tutkivat bittitasojen satunnaisuutta tai sen hajontaa, tasojen suhdetta toisiinsa tai esimerkiksi väriarvojen tilastollista vaihtelua.

4.1 Suorat poimintayritykset

Varsinaisen analysoinnin sijaan kuvatiedostosta voidaan suoraan yrittää saada irroitetuksi piilotettua dataa. LSB-tason bittejä voidaan koota eri kaavoilla ja yksinkertaisesti kokeilla, vastaako saatu dataa mitään valmista tiedostoformaattia. Tällaisella analyysillä kuvasta on kohtuullisen helppo kaivaa esimerkiksi selväkielisenä piilotettua tekstiä, tai yleisesti käytettyjä tiedostoformaatteja. Kyseinen menetelmä on myös yksinkertainen ja hyvin helppo automatisoida. Vaikka se saattaakin olla erittäin tehokas yksinkertaisimpia steganografia-algoritmeja vastaan, on tämäntyyppinen analyysi kuitenkin helppo estää hieman kehittyneemmillä algoritmeilla.

4.2 Visuaalinen analyysi

Havaitsemistekniikoista helpoin, ja tietyissä tapauksissa myös hyvin toimiva, on visuaalinen analyysi, eli useimmiten kohdetiedoston vertaaminen mahdolliseen alkuperäiseen tiedostoon. Vertailu voidaan suorittaa esimerkiksi vertaamalla kuvien bittitasoja keskenään.



Kuva 4: Vasemmalla alkuperäinen kuva ja sen LSB-taso, ja oikealla stegokuva ja sen LSB-taso.

Kuten kuvasta 4 huomataan, voidaan LSB-tasoja vertailemalla jo silmämääräisesti havaita, että toisen kuvan LSB -tasoa on muokattu. Mikäli eroavaisuuksia ei löydetä muista tasoista, antaa se suoria viitteitä steganografisten menetelmien käytöstä, sillä normaalissa kuvan muokkaamisessa myös muita tasoja olisi todennäköisesti muokattu enemmän tai vähemmän.

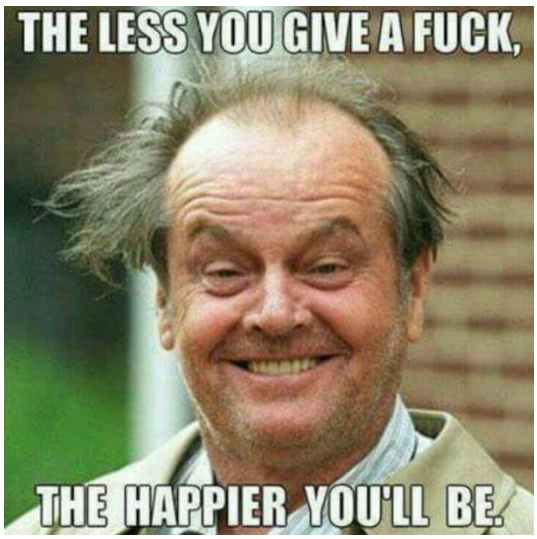
Tämän kaltainen tutkiminen vaatii kuitenkin käytännössä aina vertailukohteekseen myös alkuperäisen tiedoston, sillä hyvälaatuisen kuvatiedoston LSB-taso on yksistään tarkasteltuna hyvin satunnaisen oloinen.

4.3 Kvantitatiiviset analyysit

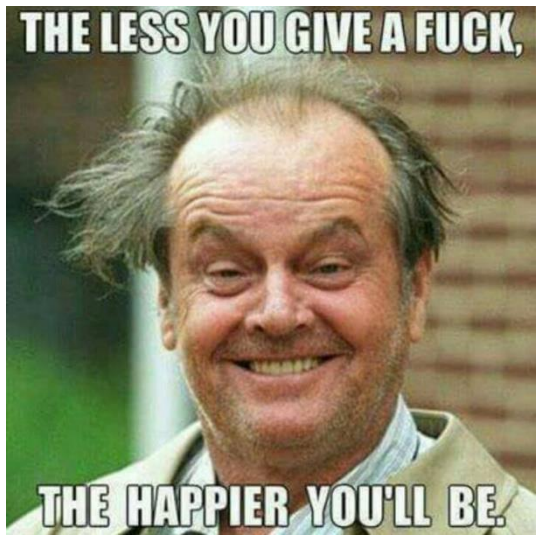
Fridrich ja muut [2003] esittelivät vuosituhannen alussa erittäin tehokkaan, joskin monimutkaisen, kvantitatiivisen analysointitekniikan LSB-steganografiaa

vastaan. Tekniikasta käytämme jatkossa nimitystä RS-analyysi, ja tässä tutkielmassa keskitymmekin nimenomaan ko. tekniikkaan, sillä se on selvästi tehokkain käytettävissä olevista kvantitatiivisista analysointimenetelmistä.

Analyysi perustuu siihen väittämään, että kuvan jokainen bittitaso on jollain tavalla yhteydessä myös viereisiin bittitasoihin. Tämä pätee myös LSB-tasoon, vaikka se silmämääräisesti tarkasteltuna näyttäisikin hyvin satunnaiselta. Kun LSB-tason bittejä korvataan salattavalla datalla, menettää taso yhteytensä seuraavaan tasoon. RS-analyysillä huomataan tämä yhteyden menetys ja voidaan jopa antaa arvio siitä, kuinka iso osa LSB-tason biteistä on korvattu piilotettavan tiedoston biteillä. Käytännössä oikein tehty RS-analyysi on erittäin tehokas LSB-steganografiaa vastaan. Tämä johtuu siitä, että korvattaessa LSB-tason bittejä muuttuu taso näennäisen satunnaisuuden sijasta oikeasti satunnaiseksi.




```
--- VSL Log ---
15:50:06: Starting experiment
15:50:36: Input number 0: Starting 0 iteration, Input -> LSB-RS
...
15:50:08: Red: Estimated message length (in bytes): 205.43833953559496
...
15:50:08: Green: Estimated message length (in bytes): 504.2616920143205
...
15:50:08: Blue: Estimated message length (in bytes): 159.59523390810895
15:50:08: Average result: 289.76508848600815
15:50:08: result: Estimated message size [B]:289.76508848600815
15:50:08: Input number 0 stopped.
15:50:08: Experiment finished
```




```
--- VSL Log ---
15:50:36: Starting experiment
15:50:36: Input number 0: Starting 0 iteration, Input -> LSB-RS
...
15:50:38: Red: Estimated message length (in bytes): 205.43833953559496
...
15:50:38: Green: Estimated message length (in bytes): 143071.67095814095
...
15:50:38: Blue: Estimated message length (in bytes): 159.59523390810895
15:50:38: Average result: 47812.234843861545
15:50:38: result: Estimated message size [B]:47812.234843861545
15:50:38: Input number 0 stopped.
15:50:38: Experiment finished
```

Kuva 5: Virtual Steganographic Laboratory -sovelluksella tehty RS-analyysi alkuperäiselle kuvalle (vas.) ja stegokuvalle (oik.).

Kuvassa 5 vasemmalla on alkuperäinen kuva ja oikealla kuva, jonka pikseleiden vihreän värin LSB-tasolle on piilotettu dataa koko kuvan alueelle. Piilotetun datan koko on n. 50kt. RS-analyysin tuloksista nähdään, että vihreän värin ”estimated message length” on selvästi muita värejä korkeampi, ja analyysin antama lopullinen arvio piilotetun datan määrästä onkin hyvin lähellä oikeaa arvoa.

5 Havaitsemisen estäminen

Siinä missä suoran poiminnan yritys joko onnistuu tai ei, antavat muut analysointimenetelmät käytännössä jonkinlaisen todennäköisyyden piilotetun datan olemassaololle tuottaen samalla myös tietyn määrän ”false positive” -tuloksia. Luotettavan havainnoinnin estämiseksi tarvitaan järjestelmä, joka suoran poiminnan estämisen lisäksi myös tiputtaa kehittyneempien analyysien antaman todennäköisyyden tarpeeksi pieneksi suhteessa väärien havainnointien määrään, jolloin piilotetun datan olemassaoloa ei käytännössä voida tarpeeksi luotettavasti osoittaa.

5.1 Suoran poiminnan estäminen

Suorat poimintayrityksen voidaan helposti estää salaamalla data tarpeeksi vahvalla salausalgoritmilla. Tästä seuraa se, että vaikka data saataisiinkin irrotettua kuvasta, on lopputulemana pelkkää satunnaista bittimössöä. Näin ollen hyökkääjä ei ilman oikeaa avainta voi mitenkään tietää, onko poimintayritys onnistunut vai ei. Salausalgoritmiksi voi valita minkä tahansa sellaisen algoritmin, joka antaa riittävän salauksen tuottaen näennäisesti satunnaisen jonon bittejä. Tämän tutkielman esimerkkikuvissa on käytetty AES-lohkosalausmenetelmää, mutta käytännössä moni muukin moderni ja turvalliseksi havaittu salausmenetelmä kelpaisi aivan yhtä hyvin. Suoraa poimintaa voisi vaikeuttaa myös muilla tavoilla, kuten esimerkiksi käyttämällä datan piilottamiseen tietynlaisella satunnaislukugeneraattoriin perustuvalla kaavalla vain osaa kuvatiedoston pikseleistä. Datan oikeaoppinen salaaminen kuitenkin estää suorat poiminnat täysin, joten muita tapoja ei tässä kohtaa tarvitse tämän enempää tarkastella.

5.2 Visuaalisen analyysin estäminen

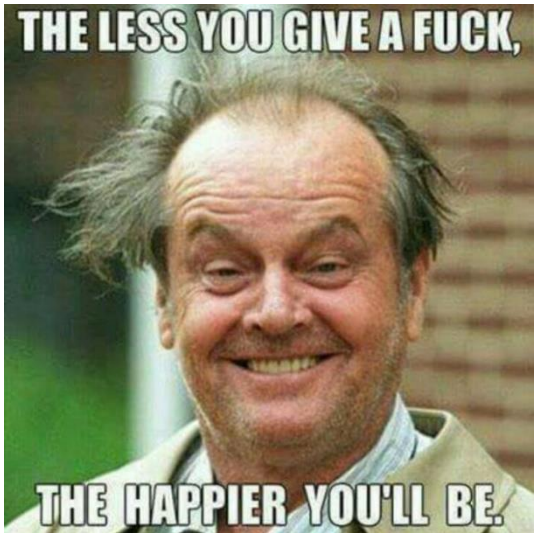
Visuaalinen analyysi voidaan estää muuntamalla piilotettava data mahdollisimman satunnaiseen muotoon ja käyttämällä itsetuotettua laadukasta säiliökuvaa, josta ei ole olemassa kopioita. Datan muuntaminen satunnaiseksi toteutettiin jo edellisessä luvussa käyttämällä salausalgoritmia, joka kelpaa tässäkin luvussa. Säiliökuvan tulisi myös olla värikäs ja monimuotoinen, joten korkealaatuisia valokuvia tulisi käyttää esimerkiksi itse tehtyjen piirrosten sijaan. Käytännössä säiliökuvaksi valitaan siis itse otettu valokuva, josta ei ole olemassa varmuuskopioita. Huomiona esimerkiksi internetistä otettu kuva on tässä kohdassa todella huono idea, vaikka se täyttäisikin muut laatuvaatimukset, sillä käytännössä kuka tahansa voi tällöin löytää alkuperäisen kuvan ja visuaalisella analyysillä todeta muokatun kuvan sisältävän piilotettua dataa.

5.3 Kvantitatiivisen analyysin estäminen

RS-analyysin estäminen on hieman vaikeampaa kuin visuaalisen analyysin ja suoran poiminnan estäminen, muttei silti suinkaan mahdotonta. Vaikka RS-analyysi osaakin arvioida kuvaan piilotetun datan määrän hyvin tarkasti, antaa se kohdekuvan tyypistä ja laadusta riippuen hyvin vaihtelevia tuloksia sellaisten kuvien kohdalla, joihin ei ole piilotettu lainkaan dataa. Tämä johtuu siitä, että RS-analyysi perustuu bittitasojen väliseen yhteyteen, ja mitä syvemmälle bittitasoissa liikutaan, sen hauraammaksi tämä yhteys muuttuu. Viimeisellä tasolla (LSB-tasolla) yhteys on jo hyvin hauras, ja tasossa esiintyykin luonnostaan kohtia joista analyysi ei löydä haluttua yhteyttä. Kutsumme tätä LSB-tason luonnolliseksi kohinaksi.

Luonnollisen kohinan olemassaolosta seuraa sellainen johtopäätös, että jokaiseen kuvaan on mahdollista piilottaa pieni määrä dataa siten, että piilotettu data ei luotettavasti erotu RS-analyysissä alkuperäisen kuvan kohinasta.

Mitä isompi luonnollinen kohina kuvalla on, sen enemmän LSB-tasosta löytyy kohtia, joilla ei ole yhteyttä seuraavaan bittitasoon, ja sitä isomman arvion (EMS-arvon) muokkaamaton kuva saa piilotetun datan määrästä RS-analyysiä tehtäessä. Mitä isompi EMS-arvo kuvalla alun perin on, sitä isomman määrän dataa kuvaan voi piilottaa ilman, että piilotetun datan määrä erottuu kohinasta. Käytännössä tavallisten kuvien kohdalla näiden kohtien lukumäärä on kohtuullisen vähäinen, ja jo erittäin pienelläkin täyttöasteella piilotettu data erottuu analyysissä selvästi kuvan luonnollisesta kohinasta. Itseasiassa Ker [2004] esittää RS-analyysin onnistumisen keskimääräiseksi todennäköisyydeksi vielä 20% kuvan LSB-tason täyttösuhteen ollessa 1%. Keskimääräinen todennäköisyys on kuitenkin hyvin harhaanjohtava luku, sillä luonnollisen kohinan määrään voidaan vaikuttaa säiliökuvan valinnalla, ja kohinan kasvaessa havaitsemisen todennäköisyyden suhde LSB-tasojen täyttösuhteeseen muuttuu luonnollisesti pienemmäksi. RS-analyysin perustuessa siihen, että bittitason satunnaistaminen aiheuttaa yhteyden menetyksen viereisiin tasoihin, voidaan päätellä jo valmiiksi mahdollisimman satunnaisen LSB-tason omaavan kuvan olevan potentiaalinen hyväksi säiliökuvaksi. Esimerkiksi korkealaatuisilla ja pakkaamattomilla kuvilla LSB-taso on lähtökohtaisesti satunnaisempi kuin huonolaatuisilla ja pakatuilla kuvilla.



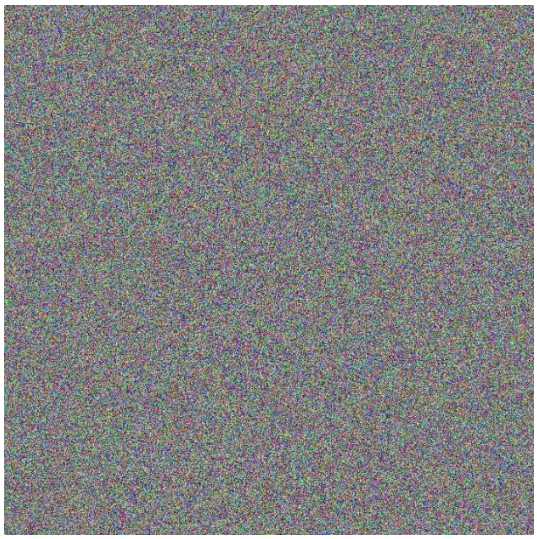
```
15:50:06: Starting experiment
15:50:06: Input number 0: Starting 0 iteration, Input -> LSB-RS
...
15:50:08: Red: Estimated message length (in bytes): 205.43833953559496
...
15:50:08: Green: Estimated message length (in bytes): 504.2616920143205
...
15:50:08: Blue: Estimated message length (in bytes): 159.59523390810895
15:50:08: Average result: 289.76508848600815
15:50:08: result: Estimated message size [B]:289.76508848600815
```

```
15:51:46: Starting experiment
15:51:47: Input number 0: Starting 0 iteration, Input -> LSB-RS
...
15:51:48: Red: Estimated message length (in bytes): 8753.845100274111
...
15:51:48: Green: Estimated message length (in bytes): 5269.369821487398
...
15:51:48: Blue: Estimated message length (in bytes): 7286.951930893805
15:51:48: Average result: 7103.388950885105
15:51:48: result: Estimated message size [B]:7103.388950885105
```

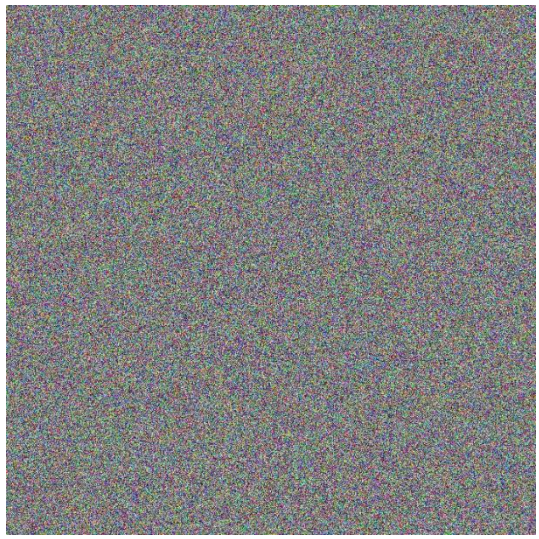
Kuva 6: Normaalilaatuisen (vas.) ja korkealaatuisen (oik.) alkuperäiskuvan RS-analyysit.

Kuvassa 6 vertaillaan normaalilaatuisen ja korkealaatuisen kuvan eroa RS-analyysille relevantin luonnollisen kohinan suhteen. Kuvat ovat alkuperäiskuvia, eli kumpikaan kuva ei sisällä piilotettua dataa. Molempien kuvien resoluutio on sama, mutta vasemmanpuoleisen kuvan laatua on heikennetty pakkaamalla se välissä JPEG-muotoon. RS-analyysin mukaan pakattu kuva sisältää n. 290t piilotettua dataa, kun taas pakkaamattoman kuvan kohdalla vastaava luku on n. 7100t. Pelkästään kuvan laadun muuttaminen aiheutti siis sen, että luonnollinen kohina, ja siten myös alkuperäisen kuvan EMS arvo, lähes 25-kertaistui.

Havaittiin siis, että laadukkaaseen ja pakkaamattomaan kuvaan voidaan todennäköisesti piilottaa moninkertaisesti enemmän dataa ilman, että RS-analyysi havaitsee sitä. Ideaa voidaan viedä vielä pitemmälle käyttämällä täysin satunnaisesti generoitua säiliökuvaa.



```
--- VSL Log ---
15:38:01: Starting experiment
15:38:01: Input number 0: Starting 0 iteration, Input -> LSB-RS
...
15:38:03: Red: Estimated message length (in bytes): 217213.0690135383
...
15:38:03: Green: Estimated message length (in bytes): 198144.91935262695
...
15:38:03: Blue: Estimated message length (in bytes): 213916.09547248643
15:38:03: Average result: 209758.02794621722
15:38:03: result: Estimated message size [B]:209758.02794621722
15:38:03: Input number 0 stopped.
15:38:03: Experiment finished
```

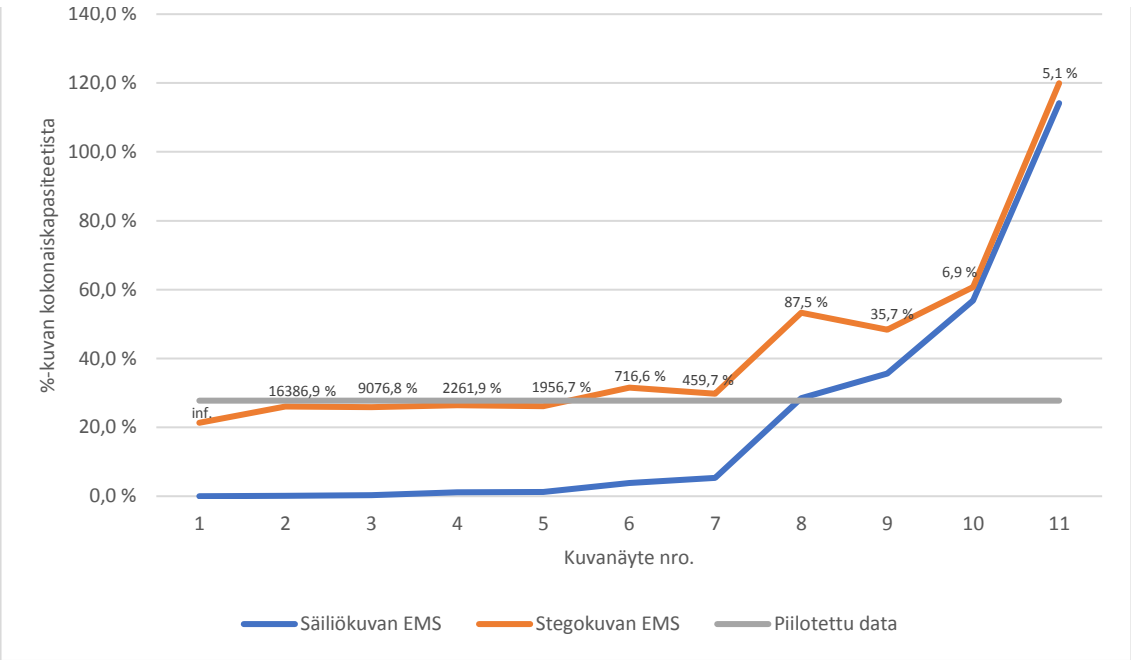


```
--- VSL Log ---
15:41:11: Starting experiment
15:41:11: Input number 0: Starting 0 iteration, Input -> LSB-RS
...
15:41:13: Red: Estimated message length (in bytes): 217213.0690135383
...
15:41:13: Green: Estimated message length (in bytes): 230053.52768477637
...
15:41:13: Blue: Estimated message length (in bytes): 213916.09547248643
15:41:13: Average result: 220394.23072360037
15:41:13: result: Estimated message size [B]:220394.23072360037
15:41:13: Input number 0 stopped.
15:41:13: Experiment finished
```

Kuva 7: RS-analyysi satunnaisesti generoidulle kuvalle ja sen stegoversiolle.

Kuvassa 7 vasemmalla täysin satunnaisesti generoitu kuva, ja oikealla samasta kuvasta tehty stegokuva, jonka vihreän värin LSB-tasolle on piilotettu n. 50kt dataa tason täyttöasteen ollessa yli 90%. Kuvien alapuolelta löytyvät kuville tehtyjen RS-analyysien tulokset. Kuten kuvasta nähdään, väittää RS-analyysi jo alkuperäisen kuvan sisältävän huiman määrän piilotettua dataa. Stegokuvan kohdalla kokonaisarvio piilotetun datan määrästä eroaa n. 5% alkuperäiseen tiedostoon nähden, vaikka kuvan LSB-tasojen kokonaistäyttöaste kasvoi nollasta noin kolmeenkymmeneen prosenttiin. Tuloksista huomataan myös, että vaikka dataa piilotettiin kuvaan noin 50kt, kasvoi arvio piilotetun datan määrästä vain noin 10,5kt. Tämä johtuu siitä, että satunnaisesti generoidun kuvan bittitasot ovat jo lähtökohtaisesti erittäin löyhässä yhteydessä toisiinsa. Vaikka bittejä korvattaisiin iso määrä, on käytännössä yhtä todennäköistä rikkoa jo olemassa oleva yhteys kuin luoda uusi yhteys. Teoriassa tällaisen kuvan EMS-arvo voi aivan yhtä hyvin laskea kuin nousta datan piilottamisen yhteydessä.

Näiden tulosten perusteella voidaan todeta, että isomman EMS-arvon omaavat säiliökuvat pystyvät häiritsemään tai jopa kokonaan estämään RS-analyysin luotettavan toiminnan. Huomataan myös, että stegokuvan EMS-arvo kasvaa hitaammin kuin alkuperäisen kuvan. Tätä tarkastellaan tarkemmin kuvassa 8.



Kuva 8: Vertailu alkuperäisen kuvan ja stegokuvan EMS-arvoista suhteessa kuvatiedoston LSB-tasojen kokonaiskapasiteettiin.

Kuvassa 8 tarkastellaan EMS-arvoja ennen ja jälkeen datan piilottamisen suhteessa kuvatiedoston LSB-tasojen kokonaiskapasiteettiin. Otannassa on mukana 11 erilaista kuvaa, jotka on järjestetty vasemmalta oikealle alkuperäisen kuvatiedoston EMS-arvojen mukaan. Kaikkien kuvien täyttösuhde on noin 30%, minkä lisäksi kaaviossa on harmaalla fontilla esitetty EMS-arvojen prosentuaalinen ero kunkin kuvan kohdalla. Nähdään, että alkuperäisen ja stegokuvan EMS-arvojen ero laskee logaritmisesti alkuperäisen kuvan EMS arvon noustessa suhteessa kuvan kokonaiskapasiteettiin.

Esimerkkitarkastelussa arvojen väliset erot alkavat kuvassa 8 olla kohtuullisen pieniä kohdassa, jossa alkuperäisen kuvan EMS-arvo on noin 60% kuvan kokonaiskapasiteetista. Tällaisesta kuvasta tehdyn stegokuvan EMS-arvo eroaa alkuperäisestä kuvasta alle 7%. Tarkkaa arvoa sille, missä vaiheessa RS-analyysin tulokset muuttuvat epäluotettaviksi on toki vaikeaa laskea, mutta aikaisemmissa tutkimuksissa noin 5% eroavaisuutta on pidetty riittävän pienenä [Marçal and Pereira 2005].

Päättelystä saadut arvot eivät näin pienellä otannalla ole missään nimessä universaalisti päteviä, mutta ne antavat kuitenkin vahvoja viitteitä siitä, että EMS-arvojen väliset erot laskevat logaritmisesti alkuperäisen kuvan EMS-arvon noustessa suhteessa kuvan kokonaiskapasiteettiin. Tästä voimme päätellä, että valitsemalla oikeanlainen säiliökuva suhteessa haluttuun täyttösuhteeseen, voidaan RS-analyysin luotettava toiminta estää, tai ainakin heikentää sen tulosten luotettavuutta merkittävästi.

Tavoitteena olevaan 5% eroavaisuuteen pääseminen vaatii kuitenkin esimerkissä olleella 30%:n täyttösuhteella käytännössä satunnaisesti generoidun säiliökuvan. Mikäli säiliönä halutaan käyttää oikeita valokuvia, joudutaan täyttösuhdetta pienentämään. Tämä aiheuttaa sen, että ko. metodi soveltuu huonosti yleispätevään steganografiseen käyttöön. Satunnaisesti generoitujen kuvien liikuttelu edestakaisin on jo itsessään vähintäänkin epäilyttävää, ja täyttösuhteen liika pienentäminen taas syö tehokkuutta ja vaatii suuria datamääriä.

Toisaalta tarkasteltu metodi voisi soveltua hyvin tilanteisiin, jossa siirrettävän datan määrä on pieni, tai dataa täytyy siirtää kertaluontoisesti. Käytännössä metodin käyttöönotto vaatisi kuitenkin tutkimustyötä suurella datamäärällä, jotta voitaisiin muodostaa luotettavat funktiot piilotettavan datan maksimipituudelle suhteessa kuvatiedoston kokonaiskapasiteettiin ja EMS-arvoon.

5.4 Kvantitatiivisten analyysien estäminen kehittyneillä algoritmeilla

Marçal ja Pereira [2005] kehittivät hyvin pian RS-analyysin julkistamisen jälkeen toimivan algoritmin sen ohittamiseksi. Algoritmi perustuu säilökuvien histogrammien käsittelyyn ennen ja jälkeen datan piilottamisen. Histogrammimuunnoksilla pyritään rikkomaan pikseleiden väliset suhteet ennen datan piilottamista ja palauttamaan ne mahdollisimman luonnollisesti takaisin datan piilottamisen jälkeen. Pikselien väliset suhteet siis tavallaan synnytetään uudestaan datan piilottamisen ja bittitasojen manipuloimisen jälkeen. Tästä seuraa se, että kuvassa pikselien väliset suhteet vaikuttavat luonnollisilta, eivätkä pikseleiden statistisia suhteita tarkastelevat analyysit osaa epäillä piilotetun datan olemassaoloa. Data saadaan myöhemmin palautettua kuvasta suorittamalla koko operaatio päinvastaisessa järjestyksessä. Kyseisellä algoritmilla saatiin noin 90%:n täyttöasteella olevien stegokuvien RS-analyysin antama EMS-arvo pienennettyä reilusti alle 5%:n suhteessa kuvan kokonaiskapasiteettiin. Algoritmin kompastuskivenä voidaan kuitenkin pitää sitä, että toimiakseen oikein algoritmin täytyy kohdistaa erilaisia muutosfunktioita erilaisille kuville. Etukäteen ei myöskään voida tietää, minkä tyyppiset muutokset toimivat kullekin kuvalle, joten käytettyjä muutosfunktioita ei voi mitenkään päätellä lopullisesta stegokuvasta. Tämä aiheuttaa käytännössä sen, että piilotetun datan palauttamiseksi täytyisi tietoa käytetyistä muutosfunktioista jotenkin tallentaa stegokuvaan varsinaisen datan piilottamisen jälkeen. Tämä rikkoo uskottavan kiistettävyyden periaatetta, sillä kaiken kuvaan piilotettavan datan tulisi olla ulkopuolisesti tarkasteltuna riittävän satunnaista. Vaikka tieto olisi kooltaan kuinka vähäinen, niin ei-satunnaisen tiedon tallentaminen ennalta sovittuihin kohtiin aiheuttaa aina sen ongelman, että kuvia analysoivan tahon on teoriassa mahdollista todentaa kuvan sisältävän

piilotettua dataa pelkästään näitä ennalta sovittuja tallennuskohtia tarkastelemalla.

6 Toimivan algoritmin rakentaminen

Toimivassa toteutuksessa tulee data ensin salata salausalgoritmilla ja tämän jälkeen piilottaa salattu data kuvatiedostoon LSB-menetelmällä. Dataa palautettaessa poimitaan ensin salattu data kuvan LSB-tasosta, ja tämän jälkeen avataan se salauksen purkuun tarkoitetulla algoritmilla.

Valmiin algoritmin toiminta voidaan jakaa neljään kohtaan:

1. Salataan data m avaimella k_0, käyttäen salausfunktiota E_e, josta saadaan salattu data

$$c = E_e(m, k_0). \quad (1)$$

2. Piilotetaan salattu data säiliökuvaan p_{cover} LSB-steganografiafunktiolla S_e. Näin saadaan stegokuva

$$p_{stego} = S_e(c, p_{cover}). \quad (2)$$

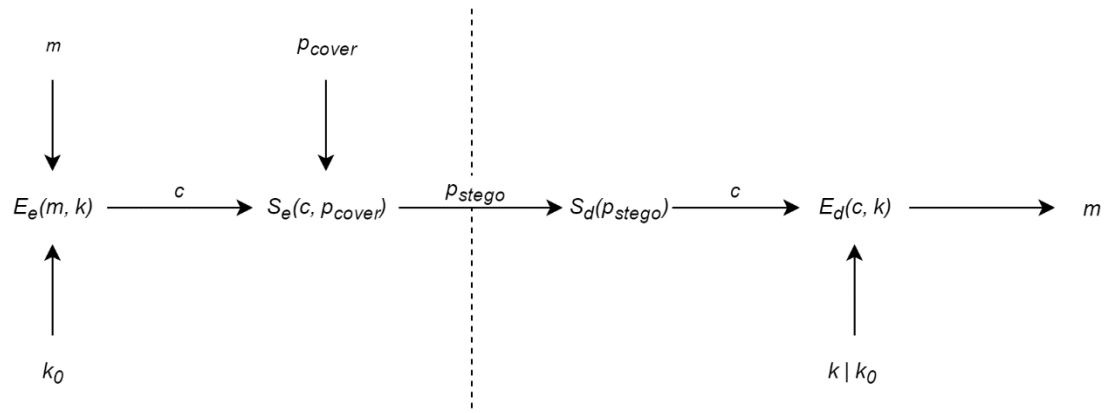
3. Puretaan salattu data stegokuvasta p_{stego} funktiolla S_d, jolloin saadaan salattu data

$$c = S_d(p_{stego}). \quad (3)$$

4. Puretaan salattu data funktiolla E_d käyttäen avainta k (symmetristä salausta käytettäessä $k = k_0$), ja palautetaan alkuperäinen data

$$m = E_d(c, k). \quad (4)$$

Algoritmin tehokkuuteen ja piilotetun datan havaittavuuteen vaikuttavat kaavoissa 1-4 sekä kuvassa 9 näkyvät funktiot E ja S, sekä säiliökuva p_{cover}. Valitsemalla nämä kolme oikein, voidaan uskottava kiistettävyys saavuttaa.



Kuva 9: Algoritmin toimintaperiaate ja vaiheet.

6.1 Salausalgoritmin E valitseminen

Salausalgoritmin valinnalla pyritään suojaamaan data suoria poimintayrityksiä, sekä visuaalista analyysiä vastaan. Uskottavan kiistettävyyden aikaansaamiseksi voidaan valita mikä tahansa salausalgoritmi, joka riittävän suojauksen lisäksi tuottaa salatusta datasta näennäisen satunnaisen bittijonon. Suositeltavin salausalgoritmi lienee symmetrisen salauksen tarpeeseen AES-lohkosalausmenetelmä ja asymmetrisen salauksen tarpeeseen oikealla tavalla toteutettu RSA-salaus. Suositellut avainten pituudet ovat vähintään 128-bittiä AES-salaukseen ja 2048-bittiä RSA-salaukseen [German Federal Office for Information Security 2017]. Varmasti riittävänä avaimen pituutena voidaan ainakin toistaiseksi pitää AES-salauksen kohdalla 256-bittistä avainta, ja RSA salauksen kohdalla 3072-bittistä avainta [United States National Institute of Standards and Technology 2015].

6.2 Säiliökuvan p_{cover}valitseminen

Uskottavan kiistettävyyden saavuttamiseksi täytyy myös joko valita sellainen LSB-algoritmi, jota ei havaita kvantitatiivisilla analyyseillä, tai sellainen säiliökuva, jonka luonnollinen kohina on riittävän suuri estämään kyseisten analyysien luotettavan toiminnan. Pieniä ja satunnaisia datamääriä varten helpompi ja luotettavampi tapa on käyttää riittävän suuren luonnollisen kohinan omaavia säiliökuvia yhdistettynä tarpeeksi pieneen täyttöasteeseen. Ongelmaksi tässä metodissa muodostuu kuitenkin tallennettavan tietomäärän rajallisuus suhteessa säiliökuvan kokoon – viestien tulee joko olla lyhyitä tai säiliökuvan isoja. Kohdassa 5.4 mainitun RS-analyysille vastustuskykyisen algoritmin [Marçal and Pereira 2005] käyttö parantaa säiliökuvasta saatavaa hyötysuhdetta, mutta tuo mukanaan myös ongelmia. Käytännössä sovelluksen tulisi pystyä ennalta analysoimaan säiliökuvia ja testaamaan niille oikeanlaiset muutosfunktiot, minkä jälkeen informaatio käytetyistä muutosfunktioista tulisi säilöä näennäisesti satunnaisena lukuna tehtyyn stegokuvaan. Yleispätevän RS-analyysille immuunin algoritmin puuttuessa lienee perinteisen LSB-algoritmin käyttäminen yhdessä mahdollisimman korkean luonnollisen kohinan omaavan kuvan kanssa paras ratkaisu. Valitaan siis säiliökuvaksi mahdollisimman hyvälaatuinen, itse otettu valokuva, tai koneellisesti satunnaisgeneroitu kuva. Kuvan valitsemisen jälkeen kuvan EMS-arvo tulee mitata, ja mitatun arvon perusteella laskea kuvan maksimaalinen täyttösuhde siten, että RS-analyysi ei löydä piilotettua dataa.

6.3 Steganografia-algoritmin *S* rakentaminen

Kohdissa 6.1 ja 6.2 mainittujen ehtojen täyttämisen jälkeen voimme valita algoritmiksi *S* perinteisen LSB-algoritmin, joka kirjoittaa dataa suoraviivaisesti kuvan jokaisen värin LSB-tasolle valitulla täyttösuhteella. Tähän käyttötarkoitukseen toimii tutkielman esimerkkikuvissa käytetty yksinkertainen algoritmi, jonka pseudokoodit löytyvät liitteistä 1 ja 2.

Datan piilottamisen suorittavalle funktiolle annetaan parametreiksi piilotettava data, kaksiulotteinen pikselitaulukko, haluttu avain, sekä täyttösuhdelukua kuvaava ”max_offset”-muuttuja. Funktio käy annetun pikselitaulukon LSB-tasoja (eli jokaisen 32-bittisen kokonaisluvun bitit 0, 8 ja 16) läpi rivi riviltä vasemmalta oikealle hypäten jokaisen kirjoitetun bitin jälkeen satunnaisen välimatkan eteenpäin. Eteenpäin hypättävä välimatka lasketaan ottamalla aina seuraavan satunnaislukugeneraattorista saadun kokonaisluvun ja funktiolle annetun max_offset-parametrin jakojäännös. Koska satunnaislukugeneraattori alustetaan annetulla avaimella, saadaan dataa purkaessa samalla avaimella myös samat välimatkat. Max_offset-arvo siis kertoo, joskin epäsuorasti, kuinka isoa osaa LSB-tason biteistä käytetään datan piilottamiseen (1 = 100%, 2 ≈67%, 3 ≈50%, 4≈40% jne…) Vaikka datan hajauttaminen tällä tavalla ei suoranaisesti vaikutakaan esimerkiksi RS-analyysin toimintaan, saavutetaan sillä silti hienoinen etu analyysin tuloksia tulkittaessa, sillä mikäli kaikki analyysin mielestä ”epäilyttävät” pikselit ovat yhdessä jonossa heti kuvatiedoston alussa, on tämä omiaan lisäämään epäilyksiä piilotetun datan olemassaolosta.

Datan purkamisen suorittava funktio ottaa parametreikseen kaksiulotteisen pikselitaulukon, avaimen, sekä täyttösuhteen kertovan muuttujan. Purkamisen onnistumiseksi avaimen ja täyttösuhteen täytyy saada samat arvot kuin dataa piilotettaessa.

7 Yhteenveto

Uskottavan kiistettävyyden saavuttaminen on mahdollista käyttämällä perinteistä LSB-algoritmia (Liitteet 1 ja 2), sekä valitsemalla oikeanlainen salausalgoritmi ja alkuperäinen säiliökuva, jonka luonnollinen EMS-arvo on riittävän suuri suhteessa piilotettavan datan kokoon. Salausalgoritmiksi suositellaan symmetrisen salauksen osalta AES-lohkosalausta vähintään 128-bittisellä avaimella, ja asymmetrisen salauksen kohdalla RSA-salakirjoitusmenetelmää vähintään 3072-bittisellä avaimella. Hyväksi säiliökuvaksi kelpaavat parhaiten korkean luonnollisen kohinan omaavat, mahdollisimman hyvälaatuiset, pakkaamattomat tai jopa kokonaan koneellisesti generoidut kuvat. Säiliökuvan tulee myös aina olla uniikki, ja alkuperäinen kuva tulee tuhota stegokuvan luomisen yhteydessä. Ku-

van maksimaalisen täyttösuhteen siten, että RS-analyysi ei löydä piilotettua dataa, voi arvioida alkuperäisen säiliökuvan EMS-arvosta. Datan piilottamisen jälkeen myös luodun stegokuvan EMS-arvo tulee myös mitata, ja sitä verrata alkuperäisen säiliökuvan arvoon. Näin voidaan varmistua siitä, että datan piilottaminen ei aiheuttanut liian isoja muutoksia arvossa.

Viiteluettelo

J. Fridrich, M. Goljan, D. Hogea and D. Soukal. 2003. Quantitative steganalysis of digital images: estimating the secret message length. *Multimedia Systems* 9, 288–302.

J. Fridrich, M. Goljan and Rui Du. 2001. Detecting LSB steganography in color, and grayscale images. *IEEE Multimedia* 8, 22–28.

German Federal Office for Information Security (BSI). 2017. Cryptographic Mechanisms: Recommendations and Key Lengths, BSI TR-02102-1.

A. Gupta and R. Garg. 2016. Detecting LSB Steganography in Images. Online at: www.rahuldotgarg.appspot.com/data/steg.pdf.

Manveer Kaur at al. 2014. Review of Various Steganalysis Techniques. *International Iournal of Computer Science and Information Technologies* 5.2.

Andrew Ker. 2004. Quantitative Evaluation of Pairs and RS Steganalysis. SPIE EI'04. Oxford University Computing Laboratory.

A. Marçal, and P. Pereira. 2005. A steganographic method for digital images robust to RS steganalysis. Proc of the *International Conference Image Analysis and Recognition,* 1192-1199.

T. Nair et al. 2012. Genetic Algorithm to Make Persistent Security and Quality of Image in Steganography from RS Analysis. *ArXiv preprint at www.arXiv.org*, 1204.2616.

United States National Institute of Standards and Technology. 2015. Commercial National Security Algorithm (CNSA) Suite Factsheet, MFS-U-OO-814670-15.

Liite 1 – LSB-Algoritmi "lsb_encode"

```
lsb_encode(message: byte[], pixels: 32bit integer[][],
                key: String, max_offset: integer):
    len: 32bit integer
    data: byte[message.length + 8]
    len ← data.length
    data[0…7] ← len
    len ← len + 8
    data[8…len] ← message
    rng: Random number generator ← init(key)
    offset: integer ← next integer from rng modulo max_offset

    B: integer ← 0   // Tavulaskuri, kertoo nykyisen paikan data-taulukossa
    b: integer ← 0   // Bittilaskuri, kertoo nykyisen bittinumeron tavussa

    For i=0 to i= pixels.height -1:
        For j=0 to j= pixels.width-1:
            For k=1 to k=3:
                offset ← offset-1

                // Hypätään bittien yli, kunnes offset-arvo on alle nolla
                If offset < 0:
                    offset ← next integer from rng modulo max_offset

                    // Asetetaan pikselitaulukon bitti i,j seuraavan data-
                    // taulukossa olevan bitin mukaan
                    If (7-b)th bit in data[B] is 1:
                        pixels[i][j] ← set (24-8*k)th bit to 1
                    Else:
                        pixels[i][j] ← set (24-8*k)th bit to 0
                    End if

                    b ← b+1

                    If b>7:
                        b ← 0
                        B ← B+1
                    End if
                End if
                If B >= len:
                    return pixels
                End if
            End for
        End for
    End for
```

Liite 2 – LSB-Algoritmi “lsb_decode”

```
lsb_decode(pixels: 32bit integer[][], key: String, max_offset: integer):
    len: integer ← pixels.width*pixels.height*3/8
    data: byte[len]
    rng: Random number generator ← init(key)
    offset: integer ← next integer from rng modulo max_offset

    temp_byte: byte ← 0;// Väliaikainen tavu, toimii kuten puskurit
    B: integer ← 0          // Laskuri nykyiselle paikalle data-taulukossa
    b: integer ← 0          // Laskuri nykyiselle bittinumerolle tavussa

    For i=0 to i=pixels.height-1:
        For j=0 to j=pixels.width-1:
            For k=1 to k=3:
                offset ← offset-1
                // Hypätään bittien yli, kunnes offset-arvo on alle nolla
                If offset < 0:
                    offset ← next integer from rng modulo max_offset
                    // Luetaan bitti pikselitaulukosta puskuriin
                    If (24-8*k)th bit in pixels[i][j] is 1:
                        temp_byte ← set (7-b)th bit to 1
                    Else:
                        temp_byte ← set (7-b)th bit to 0
                    End if

                    b ← b + 1

                    // Tavu on kokonainen
                    If b > 7:
                        // Ensimmäinen tavu kertoo datan pituuden
                        If B==0:
                            len ← temp_byte + 8
                        End if
                        // Tallennetaan puskuritavu data-taulukkoon
                        data[B] = temp_byte
                        B ← B + 1
                        b ← 0
                        temp_byte ← 0
                    End if
                End if
                If B >= len:
                    return data[8…len];
                End if
            End for
        End for
    End for
```

Pörssikauppaa käyvät algoritmit

Niklas Pulli

Tiivistelmä.
Tämä tutkielma käsittelee pörssikauppaa käyviä algoritmeja. Tutkielmassa tutustutaan algoritmisen kaupankäynnin osiin sekä siihen, millaisia strategioita ja pörssimarkkinoiden analysointimenetelmiä se sisältää. Lisäksi tutkielmassa esitellään muutamia algoritmeja ja keskitytään algoritmisen kaupankäynnin hyviin ja huonoihin puoliin.

Avainsanat ja -sanonnat: Pörssikauppa, algoritmit, markkinoiden analysointimenetelmät, HFT-kauppa

1. Johdanto

Pörssikauppaa on käyty jo vuosisatojen ajan. Viimeisten vuosikymmenten aikana se on kuitenkin muuttunut valtavasti. Kehittynyt telekommunikaatio ja tietokoneteknologia ovat mahdollistaneet globalisoituneet, dynaamiset ja monimutkaiset taloudelliset markkinat. Näillä markkinoilla kaupankäyntiä käydään tietokoneohjelmien avulla ja enenevissä määrin algoritmisen kaupankäynnin sovelluksilla. [Nuti *et al.* 2011] Algoritminen kaupankäynti on kasvanut nopeasti rahoitusvälineiden keskuudessa. Algoritmiseksi kaupankäynniksi kutsutaan kaupankäynnin eri muotoja, joissa käytetään pitkälle kehitettyjä algoritmeja automatisoimaan joko täysin tai joltain osin kaupankäyntiä. [Treleaven *et al.* 2013]

Algoritmisia kaupankäyntijärjestelmiä suunniteltaessa on oltava tarkka tietämys markkinoiden mikrorakenteesta. Ohjelman on tiedettävä, kuinka yksityiskohtainen prosessi toimii, kuinka kaupankäynti tapahtuu ja kuinka ostopäätökset vaikuttavat markkinoihin. [Nuti *et al.* 2011] Algoritmisen kaupankäynnissä äärimmäisen tärkeää on myös ymmärtää markkinoiden toiminta. Toukokuussa 2010 Yhdysvaltain pörssi romahti, viiden minuutin aikana aiheutui 600 miljardin tappiot. Tapahtuman syynä oli puutteelliset tiedot äärimmäisen nopeasta HFT-kaupasta (high frequency trading). [Nuti *et al.* 2011]

Tämän tutkielman luvussa 2 kerrotaan, mikä on pörssi ja miten se toimii. Seuraavaksi luvussa 3 käsitellään erilaisia menetelmiä markkinoiden analysoimiseen. Luvussa 4 esitellään erilaisia strategioita algoritmiselle pörssikaupalle. Sen jälkeen luvussa 5 esitellään algoritmisen kaupankäyntijärjestelmä eri osat. Luvussa 6 esitellään algoritmeja yleisesti ja tutustutaan muutamiin algoritmeihin yksityiskohtaisemmin. Lopuksi luvussa 7 tarkastellaan algoritmien käytön hyötyjä ja haittoja pörssikaupassa.

2. Pörssikaupan toiminta

Jotta algoritmista pörssikauppaa voidaan käsitellä ja ymmärtää tarkemmin, on ensin selvitettävä, kuinka pörssi toimii.

Pörssissä käydään kauppaa osakkeilla, raaka-aineilla, valuutoilla, futuureilla eli erilaisilla johdannaisilla ja optioilla. Myytävien hyödykkeiden pörssikurssi eli hinta määräytyy kysynnän ja tarjonnan mukaan. [Wikipedia 2017]

Nuti ja muut [2011] esittävät pörssikaupan etenevän seuraavasti. Pörssimeklarit tekevät osto- ja myyntitoimeksiannot keskitettyyn tilauskirjaan (centralized order book). Tilauskirjassa samaa osaketta koskevat toimeksiannot jaetaan kahteen osaan: ostotoimeksiannot toiselle puolelle ja myyntitoimeksiannot toiselle puolelle. Tilauskirja järjestelee osakkeiden toimeksiannot hinta- ja aikajärjestykseen siten, että yleisimmin ensiksi tulleet osto- ja myyntimääräykset käsitellään ensimmäisinä eli hyödynnetään niin kutsuttua first-in-first-out -menetelmää. Keskitetty tilauksia ajava kaupankäyntijärjestelmä (order-driven trading system) pyrkii jatkuvasti yhdistämään osto- ja myyntimääräykset.

Koska hyödykkeiden hinta määräytyy kysynnän ja tarjonnan eli osto- ja myyntitoimeksiantojen mukaan, voidaan päätellä, että tilauskirjassa jonossa olevilla osto- ja myyntimääräyksillä on eri hinnat riippuen paikasta jonossa.

Vaikka osto- ja myyntimääräykset tulisi järjestellä yleisimmin aikajärjestykseen tilauskirjaan, näin ei aina tapahdu. Nutin ja muiden [2011] mukaan myyntitoimeksiannot järjestellään hinnan mukaan nousevaan järjestykseen. Ostotoimeksiannot puolestaan järjestellään laskevaan järjestykseen. Mikäli useammalla osto- tai myyntitoimeksiannolla on sama hinta, perustuu järjestys silloin toimeksiannon ajankohtaan. Osto- ja myyntitoimeksiannot tapahtuvat, kun ostajan ja myyjän hinnat kohtaavat. Kaupankäyntijärjestelmä suorittaa jatkuvasti eri osakkeisiin liittyvien toimeksiantojen vertailua ja pyrkii löytämään sopivat osto- ja myyntimääräykset.

3. Markkinoiden analysointimenetelmät

Markkinoiden analysoinnin tavoitteena on ymmärtää markkinoiden käyttäytymistä, jotta osto- ja myyntipäätösten tekeminen olisi mahdollista. Markkinoiden analysointiin käytetään yleisimmin teknistä analyysiä (technical analysis) ja perusteanalyysiä (fundamental analysis). [Cavalcante *et al.* 2016] Markkinoiden analysoinnissa käytetään myös kvantitatiivista analyysiä (quantitative analysis) ja sekoitettua analyysiä (blending analysis), joka on sekoitus teknisestä analyysistä ja perusteanalyysistä [Nuti et al. 2011; Hu et al. 2015]. Näiden analysointimenetelmien tarkoituksena on ymmärtää pörssissä tapahtuvia hintojen muutoksia sekä ennustaa tulevaisuuden suuntia [Cavalcante *et al.* 2016].

Algoritminen kaupankäyntijärjestelmä hyödyntää edellä mainittuja analysointimenetelmiä. Mikäli jokin menetelmistä havaitsee sijoitusmahdollisuuden, lähettää se kaupankäyntisignaalin (trading signal). [Nuti *et al*. 2011] Kaupankäyntisignaalin avulla tehdään päätös siitä, mitä osakkeita vaihdetaan ja milloin [Treleaven *et al*. 2013].

3.1. Perusteanalyysi

Perusteanalyysiä hyödynnettäessä analysoinnin kohteena on yritys, sekä yrityksen sisäiset ja ulkoiset tekijät [Cavalcante *et al.* 2016]. Perusteanalyysin avulla pyritään selvittämään, onko yrityksen osakkeen arvo arvioitu liian suureksi tai liian pieneksi. Näin voidaan muodostaa kaupankäynnin säännöt, joiden mukaan määritetään, milloin osakkeet tulisi myydä tai ostaa. [Hu *et al.* 2015]

Perusteanalyysissä arvioidaan muuttujia, jotka vaikuttavat osakkeen arvoon [Treleaven *et al*. 2013]. Tekijät, joita perusteanalyysissä ennen sijoituspäätöstä tarkastellaan, ovat yrityksen koko, pääoman määrä, tulot, menot, varat, velat ja muut yrityksen taloudelliseen tilanteeseen vaikuttavat tekijät. Yrityksen taloudellisten tunnuslukujen lisäksi analyysissä tarkastellaan myös suhdanteita, lainsäädäntöä ja talousuutisia sekä sosiaalisia kanavia. [Cavalcante *et al.* 2016]

Hu ja muut [2015] jakavat perusteanalyysin kolmeen osaan, joissa yrityksen tilannetta arvioidaan erilaisista olennaisista näkökannoista. Nämä analyysit ovat makrotaloudellinen analyysi, toimiala-analyysi ja yritysanalyysi.

Makrotaloudellisessa analyysissä arvioidaan makrotaloudellisen ympäristön vaikutusta yrityksen tulevaisuuden tuottoihin. Tässä analyysissä merkittäviä makrotaloudellisia mittareita ovat esimerkiksi bruttokansantuote ja kuluttajahintaindeksi.

Toimiala-analyysin avulla arvioidaan yrityksen arvo. Arvon määrittely perustuu yrityksen asemaan toimialallaan ja toimialan tulevaisuuden näkymiin. Lisäksi arvioidaan yrityksen tuloja ja menoja suhteessa muihin toimialalla toimiviin yrityksiin.

Yritysanalyysin avulla analysoidaan yrityksen sen hetkistä tilannetta, jotta voidaan arvioida yrityksen sisäinen arvo. Arvioinnissa hyödynnetään pääasiassa yrityksen taloudellisia raportteja. Perusteanalyysi luo kaupankäyntisignaalin, kun nykyinen varallisuudenarvo eroaa käyvästä arvosta [Nuti *et al.* 2011].

3.2. Tekninen analyysi

Perusteanalyysi keskittyy analysoimaan yritystä ja sen tilannetta yleisesti ja suhteessa ympäröivään maailmaan. Teknisen analyysin avulla puolestaan pyritään tunnistamaan historiatietojen perusteella tekijöitä, joiden avulla pystytään

ennustamaan markkinamuutoksia tulevaisuudessa [Cavalcante *et al.* 2016]. Tekijöiksi, joita tulisi tarkastella, he nimeävät hintojen kehityksen, vaihdannan volyymin ja pörssikaupan vaihdannan vaikutusvoiman. Myös Hu ja muut [2015] toteavat teknisen analyysin tarkastelevan osakekurssien ja volyymin muutoksia sekä peilaavan näitä aiempaan saatavilla olevaan tietoon arvopaperimarkkinoista. Teknisen analyysin tarkoituksena on siis tutkia pörssissä tapahtuvien muutosten vaikutusta. Näiden tietojen perusteella pyritään ennustamaan pörssin muutoksia ja näin saamaan apua sijoituspäätösten tekoon. Tekninen analyysi luo kaupankäyntisignaalin, kun uusi pörssikurssien suuntaus on tunnistettu ja signaali päättyy, kun suuntaus loppuu [Nuti *et al.* 2011].

Hu ja muut [2015] jakavat teknisen analyysin tarkasteltavat tekijät kahdeksaan ryhmään. Nämä ryhmät ovat pörssi-ilmapiiri (sentiment), rahoitusvirta (flow-of-funds), raakadata, kehityssuunnat, vauhti (momentum), volyymi, kiertokulku ja epävakaus (volatility).

Pörssi-ilmapiiri kuvailee erilaisten pörssikauppaan osallistuvien käytöstä. Pörssi-ilmapiiriä tarkastellaan, koska ajatellaan, että sijoittajat käyttäytyvät eri tavalla markkinoiden käännekohdissa. Pörssi-ilmapiiriin vaikuttavia mittareita ovat asiantuntija-yksityishenkilö-tunnusluku (expert-public ratio), lyhyen ajan koron suhdeluku sekä myyntioptioiden ja osto-optioiden välinen suhde.

Rahoitusvirta taas on mittari, joka tutkii useiden sijoittajien taloudellista asemaa. Mittarin avulla esi-arvioidaan sijoittajien vahvuuksia osakkeiden osto- ja myyntitapahtumissa ja pyritään omaksumaan niistä parhaat ominaisuudet. Rahoitusvirtaa tarkasteltaessa analysoitava data kerätään sijoitusrahastojen tai suurten sijoittajien pääomasta ja edellä mainittujen tekemistä uusista sijoitustapahtumista.

Raakadata sisältää pörssikurssien sarjat ja vaihtokurssien mallit. Pörssikurssisarjoja hyödynnetään tavallisesti aikasarjojen analysointiin tai suuntausten arviointiin yhdessä muiden mittarien kanssa. Vaihtokurssimallit voivat heijastaa pörssi-ilmapiiriä, joka voi aiheuttaa lyhytaikaista vilkastumista.

Kehityssuuntiin perustuva mittari puolestaan käsittelee pörssikursseja ja sen tarkoituksena on jäljentää pörssikurssien kehityssuuntia. Poliittiset ja taloudelliset tapahtumat vaikuttavat pörssikursseihin. Sijoittajat voivat ansaita mallintamalla ilmiöiden kehityssuunnat.

Vauhti on myös pörssikursseihin perustuva mittari. Sen avulla arvioidaan pörssikurssin muutosten nopeus ja nopeuden paikkansapitävyys, mikäli kehityssuunnanvaihtuminen pörssikursseissa tapahtuu. Vauhdin mittaaminen perustuu oletukseen, jonka mukaan pörssikurssit käyvät läpi muodollista kiertoa ja kierto havainnollistuu kurssien heilahteluina.

Volyymiin perustuva mittari heijastaa sijoittajien innostusta ostajille ja myynnille. Tämä innostus ennakoi pörssikursseihin muutoksia. Mittari perustuu oletukseen, jonka mukaan hinnan muutokset määräytyvät ostajien ja myyjien sijoitushalukkuudesta. Sijoituspäätöksiä tavallisesti kasvatetaan noususuhdanteessa ja pienennetään laskusuhdanteissa.

Kiertokulun mittaaminen perustuu siihen, että pörssikurssit muuttuvat jaksoittain. Pitkät jaksot voivat kestää yli kymmenen vuotta, mutta se koostuu monenlaisista lyhyemmistä jaksoista. Lyhyemmät jaksot voivat kestää muutamia päiviä tai viikkoja. Kiertokulun analysointi on tärkeää, koska sen avulla on mahdollista selvittää pörssikurssien asema kyseisellä ajanjaksolla. Näin ollen voidaan ennustaa tulevaisuuden muutokset.

Epävakauden mittaria käytetään tarkastelemaan osakkeiden arvojen vaihteluväliä. Sitä voidaan käyttää riskin arviointiin sekä tunnistamaan tuen ja vastustuksen tasot.

3.3. Kvantitatiivinen analyysi

Kvantitatiivinen analyysi tarkastelee varallisuushintoja satunnaisesti matemaattisten ja tilastollisten analyysien pohjalta. Tarkoituksena on löytää sopiva kaava kuvaamaan satunnaisuutta. Kvantitatiivinen analyysi onkin hallinnut rahoitusalaa viimeisten vuosikymmenten ajan. Sen avulla on muodostettu perusta portfolioteorialle, johdannaisinstrumenttien hinnoittelulle ja riskin hallinnalle. [Nuti *et al*. 2011]

Vaikka perusteanalyysissä ja teknisessä analyysissä hyödynnetään myös matemaattisia ja tilastollisia menetelmiä, keskittyvät ne tarkastelemaan varallisuudenhinnan ja siihen liittyvän tiedon välistä determinististä suhdetta. Sen sijaan kvantitatiivinen analyysi keskittyy varallisuudenhinnan satunnaiseen vaihteluun. Tämän seurauksena kvantitatiivinen analyysi on yhteydessä johdannaistuotteiden hinnoitteluun, kuten optioiden ja swap-kaupan hinnoitteluun, joiden käypä hinta on riippuvainen varallisuuden satunnaisesta arvosta. [Nuti *et al*. 2011]

Algoritmisessa kaupankäyntijärjestelmässä kvantitatiivinen analyysi luo kaupankäyntisignaalin, kun nykyinen varallisuudenarvo eroaa varallisuudenarvon käypäarvosta. Tarkoituksena on tehdä voittoa hinnoittelun epätehokkuuden kustannuksella. Puhutaan niin kutsutusta tilastollisesta arbitraasista. [Nuti *et al*. 2011] Tilastollinen arbitraasi on osakekaupan strategia, joka käyttää aikasarja menetelmiä tunnistaakseen keskinäistä väärinhinnoittelua osakkeiden välillä [Treleaven *et al.* 2013].

4. Algoritmisen kaupankäynnin strategioita

Robottikauppa tarjoaa mahdollisuuksia käyttää kaupankäynnin strategioita osakkeiden vaihtamiseen hyödyntämällä monimutkaista laskentaa, dataa, tutkimusta ja nopeutta [Yadav 2015]. Strategiat ovat hyödyllisiä markkinoiden laadun kannalta. Algoritmit reagoivat nopeasti uuteen tietoon ja ne voivat yllättävästi parantaa esitetyn tiedon laatua. Lisäksi etenkin HFT-kauppa (high-frequency trading) eli äärimmäisen nopeaa kauppaa harjoittavat pitävät osakemarkkinat likvideinä eli helposti rahaksi muutettavina. Tämä johtuu siitä, että HFT-kauppaa käyvä algoritmi on jatkuvasti valmiina ostamaan ja myymään osakkeita.

Yadav [2015] esittelee viisi eri strategiaa algoritmikaupalle, jotka ovat käynnistyskauppa (trigger trade), arbitraasi, markkinatakaus, ennakkotilaus ja toimeksiantojen purkaminen.

Käynnistyskaupalla tarkoitetaan robottikauppaa, jossa tietokoneohjelmat lähettävät sähköisiä toimeksiantoja osakkeiden vaihtamiseksi ennalta asetettujen määräysten mukaisesti. Tietokoneet tarkkailevat osakkeiden hintojen kehitystä ja ostavat tai myyvät osakkeita, kun saavutetaan ennalta määritetty käynnistyshinta (trigger price).

Arbitraasi-strategian avulla pyritään löytämään osakemarkkinoilta väärinhinnoiteltuja osakkeita. Algoritmien avulla etsitään markkinoilta arvopaperikauppa haluttuun hintaan. Lisäksi niiden avulla voidaan tutkia markkinoilta osakkeita, joiden hinnat poikkeavat toisistaan. Esimerkiksi jossain pörssissä saman osakkeen arvo voi olla eri kuin toisessa ja tällaisia tilanteita algoritmi pyrkii löytämään. Voidaan olettaa, että hinnat tulevat lähestymään toisiaan, jolloin algoritmi voi ostaa halvemmalla ja lopulta myydä kalliimmalla ja näin tuottaa voittoa. Arbitraasi-strategian mukaan algoritmi pyrkii siis löytämään väärinhinnoitellun osakkeen tai sitten löytämään eri pörssien välillä pieniä hintaeroja, joiden avulla voidaan tehdä voittoa.

Markkinatakausstrategiaa hyödynnettäessä pyritään pitämään yllä markkinoiden likviditeettiä ja järjestystä. Erityisesti nopeaa HFT-kauppaa käyvät yritykset toteuttavat markkinatakausta. Yritykset ostavat ja myyvät osakkeita nopealla vauhdilla, sillä niillä on kyky suorittaa jopa tuhansia osakekauppoja millisekuntien aikana. Tämä on mahdollista ensiluokkaisen teknologian ja nopeuden avulla. Nopean kaupan algoritmien avulla muilla sijoittajilla on mahdollisuus päästä halvemmilla hinnoilla markkinoille, sekä saada nopeamman täytäntöönpanon ja kilpailukykyisemmät maksut.

Ennakkotilausstrategian avulla algoritmit pyrkivät selvittämään investointirahastojen mahdollisia suurempia toimeksiantoja ja tekemään sen avulla voittoa. Algoritmin nopean toiminnan avulla pystytään ostamaan osakkeet ennen

investointirahastoa, jolloin suuren vaihdannan takia osakkeiden arvo nousee ja ne voidaan myydä edelleen hieman kalliimmalla joko investointirahastolle tai jollekin muulle sijoittajalle.

Toimeksiannon purkamisen strategialla tarkoitetaan toimintaa, jossa algoritmit purkavat isompia toimeksiantojaan pienempiin osiin. Tämä tapahtuu sen jälkeen, kun algoritmi on saanut selville tiedon suuremman sijoitusrahaston halusta osallistua suurempaan kauppaan. Tämän jälkeen algoritmi voi hyödyntää edellä mainittua ennakkotilaus strategiaa.

Lisäksi Treleaven ja muut [2013] esittelevät algoritmisen kaupankäynnin strategioiksi myös impulssistrategian (momentum) ja keskiarvon palautumisen strategian (mean reversion).

Impulssistrategialla tarkoitetaan menettelyä, jossa kaupankäynninstrategia seuraa kurssikehitystä ja pyrkii tekemään tuottoa markkinoiden kehityssuunnan jatkumisella. Algoritmi omaksuu kasvun seuratun osakkeen hinnoissa ja tekee voittoa ostamalla kallistuvia osakkeita ja vastaavasti myymällä osakkeen, jos sen hinnankehitys on laskeva.

Keskiarvoon palautumisen strategia puolestaan olettaa, että hinnat palaavat takaisin keskiarvoonsa. Näin ollen, jos osakkeen hinnat poikkeavat keskiarvostaan, algoritmi tekee toimeksiantopäätöksen niitä kohtaan.

5. Algoritmisen kaupankäyntijärjestelmän osat

Pörssien toiminta on monimutkaista ja jokaisen toimeksiannon taustalla on valtava määrä erilaisten tietojen analysointia ja käsittelyä. Treleaven ja muut [2013] esittävät algoritmisen kaupankäyntijärjestelmän koostuvan viidestä eri tasosta. Nämä tasot ovat tiedon saaminen ja puhdistaminen, kauppaa edeltävä analyysi (pre-trade analysis), kaupankäyntisignaali, kaupan toimeenpano ja kaupan jälkeinen analyysi.

Algoritminen kaupankäyntijärjestelmä on monitasoinen sekä sen toimintaan ja tehokkuuteen vaikuttavat monet eri tekijät pörssidatassa, kansantaloudessa ja politiikassa. Kaupankäyntijärjestelmät ovat myös hyvin erilaisia riippuen siitä, millaista strategiaa halutaan toteuttaa. Strategia, riskinmäärä, painotukset, kustannukset ja hintojenkehitys mahdollistavat valtavan määrän erilaisia kombinaatioita, joiden mukaan järjestelmän suunnittelija voi algoritmisen kaupankäyntijärjestelmän toteuttaa.

5.1 Tiedon saaminen ja puhdistaminen

Kaupankäyntijärjestelmän ensimmäisellä tasolla hankitaan tietoa ja puhdistetaan se käsiteltävään muotoon myöhempiä tasoja varten. Tiedon puhdistamisella on tärkeä rooli kaupankäyntijärjestelmän menestymisen kannalta, kos-

ka se helpottaa ja nopeuttaa saatavilla olevan tiedon käsittelyä. Kaupankäyntijärjestelmän kannalta oleellinen tieto sisältää rahoituksellista tietoa, kansantaloudellista tietoa sekä uutistietoa. Rahoituksellinen tieto sisältää hintatietoja rahoituksellisista instrumenteista pörssissä ja sähköisissä viestintäverkoissa. Kansantaloudellinen tieto puolestaan on tietoa, joka kuvastaa valtioiden taloudellista tilaa ja johon kuuluvat työttömyyden mittarit, bruttokansantuote, korkoprosentit sekä kansallinen politiikka. Uutistiedot sen sijaan sisältää pörssi-ilmapiiriä koskevaa tietoa, joka on hankittu sosiaalisen median palveluista, RSS-syötteistä ja uutispalveluista. Uutistiedot jaetaan vielä reaaliaikaiseen dataan ja historiatietoihin. Reaaliaikainen data käsittää kaikki syötteet pörssistä eri kanavien kautta, kuten sähköiset viestintäverkot, uutispalvelut tai sosiaalinen media. Historiatiedot puolestaan sisältää aiemmin kertynyttä ja varastoitua rahoituksellista, kansantaloudellista ja sosiaalista tai uutisista peräisin olevaa tietoa. Historiatiedot on puolestaan jaettu vielä raakadataan ja puhdistettuun tietoon. Raakadatalla tarkoitetaan tietoa, joka on käsittelemätöntä tietoa suoraan lähteestä ja siksi voi sisältää virheitä tai voi olla analysoimatonta. Puhdistettu tieto puolestaan on käsiteltyä tai siitä on poistettu virheellinen ja ylimääräinen tieto. Näiden edellä mainittujen tietojen saaminen on äärimmäisen tärkeätä kaupankäyntialgoritmien tutkimiselle, suunnittelulle ja päätöksenteolle. Tiedon puhdistaminen on aikaa vievää, mutta sen saaminen on välttämätöntä kaupankäynninalgoritmien toiminnalle.

5.2 Kauppaa edeltävä analyysi

Ensimmäisen tason jälkeen toisena on kauppaa edeltävä analyysi. Kauppaa edeltävä analyysin sisältää laskennallisia ja analyyttisia toimintoja kolmen eri mallin avulla. Nämä mallit ovat alfamalli, riskimalli ja transaktiokustannusmalli. Alfamalli on matemaattinen malli, joka on suunniteltu ennustamaan tulevaisuuden rahoitusinstrumenttien käyttäytymistä. Riskimalli puolestaan arvioi rahoitusinstrumenttien riskille altistumisen tasoa. Lopuksi transaktiokustannusmalli laskee potentiaalisia kustannuksia rahoitusinstrumenttien vaihdannassa. Kaikki kauppaa edeltävän analyysin mallit hyödyntävät markkinoidenanalysoinnin menetelmiä. Erityisesti alfamalli analysoi reaaliaikaista- ja historiallista tietoa näiden markkinoinnin analysoinnin menetelmien avulla löytääkseen potentiaalisia kaupankäyntimahdollisuuksia. Alfamalli jaetaan tavallisesti kahteen vaihtoehtoiseen lähestymistapaan. Nämä lähestymistavat ovat teoreettinen lähestymistapa ja empiirinen lähestymistapa. Teoreettista lähestymistapa käytettäessä ohjelmoijat valitsevat hypoteesin, jonka mukaan he olettavat osakkeiden käyttäytyvän. Empiirisen lähestymistavan mukaan ohjelmoijat sen sijaan käyttävät algoritmia tiedonlouhintaan. Algoritmin avulla he pyr-

kivät tunnistamaan kaavan saatavilla olevan tiedon perusteella ilman, että he tekevät oletuksia osakkeiden käyttäytymisestä.

Alfamallin rakentaminen ja muuttujien asettaminen on monimutkainen tehtävä. Monet tekijät vaikuttavat algoritmin toimintaan. Näitä tekijöitä ovat ennustuksen tavoitteet, kuten suunta, suuruusluokka, kesto ja todennäköisyys. Lisäksi ennustuksen aikahorisontilla on suuri merkitys eli ennustetaanko sekunteja, päiviä vai viikkoja. Myös osakkeiden sekoitukset, saatavilla oleva tieto ja muuttujien asettelu mallissa vaikuttavat algoritmin toimintaan.

Riskimalli puolestaan keskittyy tavoitellun osakesalkun riskeihin. Lisäksi riskimalli tutkii merkityksellisiä tekijöitä, jotka vaikuttavat pörssi-ilmapiiriin ja arvioi rahoitusinstrumenttien nykyistä ja tulevaa arvoa. Näitä asioita tutkimalla malli pyrkii määrittelemään yksittäisten osakkeiden ja koko osakesalkun riskin määrän. Riskimallin kaksi lähestymistapaa ovat riskin rajamäärä ja riskin rajatyyppi. Riskin rajamäärän avulla rajoitetaan riskin kokoa, jotta voidaan hallita kurssiriskipositioita vipuvoiman tai epävakaisuuden ja hajaantumisen laskennan avulla. Riskin rajatyyppi puolestaan pyrkii poistamaan kaikki kurssiriskipreemiot.

Transaktiokustannusmalli toimii siten, että se laskee kaikki mahdolliset transaktiokustannukset. Transaktiokustannuksiin kuuluu muun muassa välityspalkkiot, hävikki ja markkinavaikutus. Edellä mainitun kolmen komponentin avulla voidaan luoda kaupankäyntisignaalin generointi, jonka myötä kolmannen tason eli kaupankäyntisignaalin taso saa syötteenä alfamallin, riskimallin ja transaktiokustannusmallin tulokset. Näiden tulosten perusteella valitaan paras mahdollinen osakesalkku, jotta voidaan maksimoida tuotot rajoitetulla riskillä ja minimoiduilla kaupankäynnin kuluilla.

5.3 Kaupankäyntisignaali

Seuraavaksi kaupankäyntisignaalin taso sisältää osakesalkun rakennuksen mallin (portfolio construction model). Osakesalkun rakennuksen malli ottaa edellisen tason malleilta tulokset syötteenä ja näiden tulosten perusteella rakentaa laadullisesti ja määrällisesti parhaimman osakesalkun. Myös osakesalkun rakennuksen malli jaetaan kahdeksi eri malliksi, sääntöpohjaisiksi malleiksi ja optimointimalleiksi. Sääntöpohjaiset mallit ovat heuristisia vaatimuksia, joiden pohjalta osakesalkku kootaan. Strategiasta riippuen sääntöpohjainen malli voi käsitellä tasavertaista aseman painotusta (equal position weighting), tasavertaista riskin painotusta (equal risk weighting), alfan käytön painotusta (alpha-driven weighting) tai päätöspuupainostusta (decision-tree weighting). Strategiasta riippuen voidaan siis painottaa eri asioita osakesalkun kokoamisessa. Optimointimalli puolestaan käyttää algoritmeja yhdessä kohdefunktioiden

kanssa, jotka iteroivat osakesalkun. Iteroinnin avulla pyritään löytämään osakesalkku, joka antaa esimerkiksi parhaan tuottoprosentin. Kun osakesalkun rakennuksen malli on saanut koottua parhaan mahdollisen osakesalkun, annetaan se syötteenä seuraavalle tasolle.

5.4 Kaupantoimeenpano ja kaupan jälkeinen analyysi

Kaupantoimeenpanon tasolla toimeenpano malli suorittaa päätöksiä muun muassa pörssistä, kaupankäynnin strategiasta ja toimeksiannon tyypistä. Tärkein päätös on kuitenkin kaupankäynnin strategian valinta. Toimeenpanon mallin on siis päätettävä, missä pörssi toimeksiannot halutaan toteuttaa. Tämä päätös on tehtävä, sillä monet osakkeet ovat markkinakelpoisia useissa eri pörsseissä. Toimeksiannon tyyppiä valittaessa järjestelmä jakaa markkinoiden toimeksiannot valitun strategian mukaisten sääntöjen mukaan.

Viimeisellä tasolla, kaupan jälkeisen analyysin tasolla, verrataan tehdyn kaupan odotettua ja todellista toimintaa. Tämän analyysin avulla voidaan kehittää hinnan ja toiminnan mittaamista.

6. Algoritmit

Yadavin [2015] mukaan algoritminen kaupankäynti reflektoi perusteellisesti kaikkia kaupankäynnin näkökulmia, sillä reflektointi tapahtuu etukäteen ohjelmoitujen ohjeistusten avulla tietokoneellisesti. Jotta kauppaa käyvän algoritmin ohjelmointi olisi mahdollista, on ohjelmoijalla oltava tarkka käsitys siitä, kuinka markkinat toimivat ja minkälaista strategiaa ohjelman halutaan noudattavan. Siksi pohjatyö ennen ohjelmointia on tehtävä huolella.

Ennen algoritmien ohjelmointia pörssikauppiaiden on laadittava yksityiskohtainen strategia osakkeiden myynnille ja ostamiselle. Tämän jälkeen ohjelmoijat voivat strategian mukaisesti rakentaa tietokoneistetun algoritmin tai algoritmien sarjan toteuttamaan strategiaa pörssimarkkinoille. Vaikka algoritmi toimiikin itsenäisesti, on ihmisillä suuri rooli sen toiminnassa. Strategian luonti ja ohjelmointi ovat ihmisten toteuttamaa, sekä sen lisäksi ihmiset valvovat ohjelman toimintaa. [Yadav 2015] Algoritmit voivat olla luonteeltaan ja toimintatavoiltaan hyvin erilaisia. Ne voidaan suunnitella toimimaan pitkän tai lyhyen aikajakson perusteisesti. Esimerkiksi algoritmista riippuen, algoritmi voi toteuttaa 1000 toimeksiantoa joko tunnissa tai muutamien kuukausien aikana. [Burr 2014]

Osakemarkkinat ovat monimutkainen kokonaisuus, jossa monet tekijät vaikuttavat markkinoiden muutokseen [Mabu *et al.* 2015]. Ja siksi, kuten Burr [2014] toteaa, algoritmien on huomioitava lukuisia tekijöitä, jotta ne saavat selvitettyä parhaan mahdollisen keinon, ajan ja kokoluokan toimeksiantojen to-

teuttamiseksi. Tästä syystä markkinoiden suuntauksia ennustetaan tavallisesti perusteanalyysin ja teknisen analyysin avulla. Osakekauppamallit ja osakehintojen ennustaminen pohjautuvat laskennalliseen älyyn (computational intelligence), joka taas kuuluu tavallisesti tekniseen analyysiin.

Monet algoritmisen kaupankäynnin menetelmät käyttävät pehmeän laskennan (soft computing) ja koneoppimisen algoritmeja kuten neuroverkkoja, geneettisiä algoritmeja, geneettistä ohjelmointia, tukivektorikonetta (support vector machine ja tiedonlouhintaa [Mabu *et al.* 2015; Cavalcante *et al.* 2016]. Neuroverkkojen avulla voidaan etsiä epälineaarista toimintaa aikasarjoista ilman tilastollista oletusta datasta ja siksi sitä käytetäänkin aikasarjojen ennustamiseen. Aikasarjalla tarkoitetaan numeerisen datan havainnoinnin jaksoa, jonka aikana tietoa on tallennettu. Tiedonlouhintaa hyödynnetään etenkin perusteanalyysissa, jossa algoritmin on käytävä läpi valtava määrä tekstimuodossa olevaa informaatiota ja poimittava niistä tietoa tulevaisuuden taloudellisten arvojen ennustamisen tueksi.

Algoritmeja voidaan hyödyntää pörssikaupassa moneen eri tarkoitukseen ja jokainen niistä on erilainen riippuen siitä, minkälaista strategiaa algoritmin suunnittelija on halunnut toteuttaa. Seuraavaksi esitellään yksi algoritmisessa pörssikaupassa yleisimmin käytetty kaupankäynnin muoto. HFT-kaupalla tarkoitetaan autonomista kaupankäynninsovellusta, jonka strategia perustuu lyhyisiin osakkeiden hallussapidon jaksoihin. Sen jälkeen esitellään muutamia erilaisia algoritmeja, joiden toiminta perustuu erilaisiin kaupankäynninstrategioihin.

6.1 HFT-Kauppa

HFT-kauppa (high-frequency trading) on algoritmisen pörssikaupan erityisalue, jossa tietokoneistettu kaupankäynnin strategia perustuu äärimmäisen lyhyisiin osakkeiden hallussapidon jaksoihin. Jaksot ovat pituudeltaan vain muutamia sekunteja tai jopa millisekunteja [Treleaven *et al.* 2013].

HFT-kauppa on äärimmäisen kallista ja siitä syystä yksittäisillä sijoittajilla ei ole mahdollista osallistua siihen. HFT-kauppa perustuu nopeuteen, jonka avulla HFT-kauppaa käyvät yritykset ovat nopeampia kuin muut. Jotta nopeus olisi mahdollista, on yrityksen pienennettävä viivästysaikaa yrityksen ja pörssin välillä. Tämä on mahdollista olemalla samassa paikassa (co-location) tai saamalla suora yhteys markkinoille (direct market access). Samassa paikassa olemisella, tarkoitetaan, että yritys laittaa itsensä markkinoille, jolloin se saa varmistettua viiveettömän tilan. Tällöin yrityksen on mahdollista tehdä toimeenpanoja niin nopeasti kuin algoritmi niitä tuottaa. Suoralla yhteydellä markkinoille tarkoitetaan puolestaan sähköisen kaupankäynnin välineiden hy-

väksikäyttöä, jolloin HFT-kauppaa käyvällä yrityksellä on mahdollisuus päästä suoraan pörssiin. Näiden lisäksi HFT-yritykset voivat saada nopeamman yhteyden pörssiin käyttämällä mikroaaltosignaaleja, jotka ovat tavallista kaapeli- ja satelliittisignaalia nopeammat. Mikroaalloilla liikkuvan signaalin hyödyntäminen on paljon nopeampaa, koska kaapelit luovat signaaleille vastustusta ja hidastavat sähköisten signaalien liikettä, kun taas mikroaaltojen liikkuessa ilmassa, vastus ei ole niin suurta. [Burr 2014]

HFT-kauppa on erittäin merkittävä osa pörssikauppaa, sillä jo vuonna 2010 56 % Yhdysvaltojen pörsseissä tapahtuvista osakevaihdoista tapahtui sen avulla. Vastaava luku Euroopassa oli 38 %. [Biais and Woolley 2011] HFT-kaupan odotetaan edelleen yleistyvän, sillä vuonna 2012 osakekaupoista jo yli 60 % tapahtui sen avulla. HFT-kauppa tekee voittonsa hyötymällä pienistä ja lyhytkestoisista kurssieroista, joiden avulla ne tekevät voittoa vain pennejä jokaista vaihdettua osaketta kohden. Nopeudesta johtuen HFT-kauppaa käyvät yritykset tuottavat kuitenkin valtavia summia, sillä sen nopeudesta ja luonteesta johtuen, se tekee tuhansia tai jopa miljoonia toimeksiantoja päivässä. [Burr 2014]

6.2 Muita algoritmeja

Algoritmiset kaupankäyntijärjestelmät voivat olla hyvin erilaisia, riippuen halutusta strategiasta, riskin määrästä, kustannuksista, painotuksista ja hintojen kehityksestä. Luvussa 5 esiteltiin algoritmisen kaupankäyntijärjestelmän perusrakenne, jonka pohjalta järjestelmät jossain määrin rakennetaan. Tässä luvussa esitellään seuraavaksi impulsseihin perustuva algoritmi ja parivaihdannan algoritmi.

Treleaven ja muut [2013] esittelevät impulsseihin perustuvan algoritmisen kaupankäyntijärjestelmän. Impulssistrategialle on tyypillistä seurata hintojen kehitystä ja ostaa osakkeita, kun kehitys on nouseva ja myydä osakkeita kun, kehitys on laskeva. Algoritmi hyödyntää niin kutsuttua yksinkertaista liukuvan keskiarvon tunnuslukua viimeisten 60 minuutin ajalta tunnistaakseen laskevat ja nousevat hintakehitykset. Impulsseihin perustuvan algoritmin ensimmäinen vaihe on rakentaa perusrakenne käsittelemään valtavia määriä reaaliaikaista ja historiallista tietoa. Seuraavaksi kauppaa edeltävän analyysin menetelmät järjestelevät syöttötiedot ja laskevat tilastolliset tunnusluvut, jotka tunnistavat ja rekisteröivät poikkeavuudet tiedoissa. Tunnistettavia poikkeavuuksia voivat olla esimerkiksi kurssin muutosherkkyys, kehityssuunnat tai keskiarvosta poikkeaminen. Seuraavassa vaiheessa riskimalliin kuuluvalla kaupankäyntisignaalin tasolla lasketaan riskin tunnusluvut ehdolla oleville osakkeille ja osakesalkuille. Laskennassa hyödynnetään riskin arvon laskentaa. Riskin arvon pudotessa alle rajariskin, ennalta määritetyt säännöt astuvat voimaan. Tämän

algoritmin osalla se tarkoittaa kaupankäyntijärjestelmän toimintojen pysäyttämistä, jonka jälkeen algoritmi tulee käynnistää uudelleen manuaalisesti. Mikäli riskin arvo ei ylitä ennalta määriteltyä rajaa, toteutetaan tässä vaiheessa kaupankäynninsignaali, joko myyntitoimeksiannolle tai ostotoimeksiannolle. Toimeksianto riippuu siitä, onko osakkeen kehityssuunta nouseva vai laskeva. Lopuksi kaupan toimeksiannon tasolla päätetään pörssi, jossa toimeksianto tehdään. Päätös perustuu likviditeetinarvostukseen eli valitaan kauppapaikka, jossa toimeksiannot saadaan mahdollisimman nopeasti tehtyä. Viimeiseksi järjestelmä analysoi toteutuneen kaupan ja arvioi eron ennustetun ja lopullisen hinnan välillä. Lopullinen hinta sisältää muun muassa välityspalkkiot ja verot. Analysoinnin avulla selvitetään kyseiselle osakkeelle allokoitavien rahojen määrä tulevaisuutta varten.

Deshpande ja Barmish [2016] esittelevät parivaihdannan algoritmin (pairs-trading algorithm). Se hyödyntää aikaisempia riippuvuussuhteita osakkeiden välillä ja pyrkii siten löytämään tilastollisen arbitraasin. Tilastollisessa arbitraasissa tehdään voittoa väärinhinnoittelun avulla. Algoritmin toiminta perustuu siihen, että algoritmi pyrkii löytämään kaksi osakketta, joiden hinnat poik.-keavat niiden historiallisesti havaitusta tavallisesta suhteesta. Oletetaan, että näiden osakkeiden hinnat palaavat takaisin historiallisesti havaittuun suhteeseen.

7. Algoritmikaupan hyötyjä ja haittoja

Algoritmikauppaa pidetään taloudellisesti tehokkaana ja tuottavana, koska tietokone voi suorittaa osakkeiden osto- ja myyntitapahtumia, sekä lisäksi havaita markkinoilla ja uutisissa tapahtuvia muutoksia huomattavasti ihmistä nopeammin. Algoritmikauppa vähentää myös pankkien ja rahoitusyhtiöiden toimintakustannuksia. Ajatellaan myös, että algoritmikauppa parantaa pörssimarkkinoiden toimintaa. Algoritmikaupan hyötyinä voidaan pitää, että se auttaa pitämään pörssikurssit vakaina suurten valuuttojen välillä, auttaa sijoittajia hajauttamaan sijoituksia eri pörssien välillä ja lisäksi sen avulla voidaan pienentää ihmisen omaa havainnointikykyä. [Biais and Woolley 2011]

Perusteluna algoritmikaupan mielipiteiden jakautumiselle voidaan varmasti pitää sen vaikutusta tavallisen pörssikaupan nopeaan muutokseen. Siksi on ymmärrettävää, että sillä on myös vastustajansa.

Robottikauppaa vastustavat perustelevat algoritmikaupan haittoina olevan pörssikaupan manipulointi, epäsuotuisa valikoituminen (adverse selection), epätäydellinen kilpailu ja systeeminen riski. Algoritmikauppa on nopeaa ja sitä käyvät useat tietokoneet samanaikaisesti ympäri maailmaa. Tästä syystä algoritmikaupan vastustajat uskovat, että osa koneista on naamioitu johtamaan

harhaan muita sijoittajia ja muita robottikauppaa käyviä algoritmeja. [Biais and Woolley 2011] On ymmärrettävää, että kilpailu pörssikaupassa on kovaa ja se houkuttelee kaupankävijöitä käyttämään kyseenalaisia keinoja pärjätäkseen paremmin kilpailussa.

Suhteellisen pieni määrä algoritmikauppaa käyvistä sijoittajista tekee suuren osan pörssikaupan liikevaihdosta. Tästä syystä he saattavat osallistua markkinoiden manipuloimiseen käyttäen jotain kolmesta manipuloinnin keinosta. Nämä markkinoiden manipuloinnin keinot ovat tilkitseminen (stuffing), savustaminen (smoking) ja huijaaminen (spoofing). Tilkitsemisellä tarkoitetaan sitä, että algoritmit tekevät pörssissä kauppoja, joita on hankala käsitellä. Tämä vaikeuttaa tavallisten pörssimeklareiden työtä, jolloin he eivät saa selkeää kuvaa markkinatilanteesta. [Biais and Woolley 2011] Tilkitsemisen lainmukaisuutta voidaan pitää kyseenalaisena, sillä vuonna 2013 FBI aloitti toimenpiteet sen ehkäisemiseksi [Burr 2014]. Savustamista harjoittaessa algoritmikauppaa käyvät johtavat muita sijoittajia harhaan julkaisemalla erilaisia tiedotteita markkinatilanteesta. Savustamisen tavoitteena on hämätä muita sijoittajia epämääräisten tiedotteiden ja ostopäätösten avulla. Tarvittaessa ostopäätökset perutaan, mikäli julkaisut eivät saa aikaan robottikauppaa käyvän toivomaa lopputulosta. Tämä on mahdollista vain siksi, että kone on riittävän nopea toimimaan ja lisäksi sen on tehtävä ostot riittävän suurilla summilla, jotta ostopäätökset herättäisivät mielenkiintoa muiden sijoittajien silmissä. [Biais and Woolley 2011]

Epäsuotuisa valikoituminen tarkoittaa tässä yhteydessä sitä, että robottikaupan harjoittajilla on parempi tieto vallitsevasta markkinatilanteesta kuin muilla pörssikauppaa käyvillä. Esimerkiksi algoritmikauppaa käyvillä on 20 prosenttia parempi käsitys markkinoilla tapahtuvista hintamuutoksista kuin tavallisilla sijoittajilla. Tämä johtuu siitä, että tietokoneet saavat hintamuutokset selville huomattavasti nopeammin kuin ihmiset. Näin ollen algoritmikauppaa käyvillä on mahdollisuus säännellä markkinoilla vallitsevia hintoja. [Biais and Woolley 2011] Tämä tuntuu epäoikeudenmukaiselta muita sijoittajia kohtaan, koska kaikille ei ole tarjolla samanlaista tietoa sen hetkisestä markkinatilanteesta. Biais ja Woolley [2011] esittävät algoritminkaupan haittana myös epätäydellisen kilpailun. Heidän mukaansa epäsuotuisan valikoitumisen seurauksena algoritmikaupan hintojen sääntely johtaa epätäydelliseen kilpailuun pörssikaupassa. Burr [2014] toteaa, että epätäydellisen kilpailun ja epäsuotuisan valikoitumisen ehkäisemiseksi on kehitetty sovellus, joka pienentää robottikaupan tehokkaan tiedonsaannin tuomaa aikaetua suhteessa tavallisiin sijoittajiin.

Systeeminen riski taas perustuu samanlaiseen sijoittamistapaan algoritmikauppaa käyvien välillä. Tämä saattaa johtaa siihen, että kaikki robottikauppaa käyvät koneet tekevät huonoja sijoituspäätöksiä, jotka johtavat markkinoiden

heikentymiseen tai pahimmassa tapauksessa pörssikurssien romahtamiseen. [Biais and Woolley 2011]

Yhteenveto

Tässä tutkielmassa on esitelty algoritmisen kaupankäyntijärjestelmän osat sekä niiden hyödyntämät strategiat ja markkinoiden analysointimenetelmät. Kuten Algoritmista kaupankäyntiä hyödynnetään erityisesti markkinoiden käyttäytymisen analysointiin ja ennustamiseen [Cavalcante *et al.* 2016].

Coobs [2016] toteaa, että kaupankäyntijärjestelmien tietämystä on alettu kasvattaa aiemmin sattuneiden taloudellisten kriisien jälkeen. Tästä syystä myös valvontaa ja läpinäkyvyyttä on lisätty osakemarkkinoilla. Burr [2014] puolestaan esittää, että tämän vuosikymmenen alussa sattuneiden pörssikriisien jälkeen talousvalvojat Yhdysvalloissa ja myös muualla maailmassa ovat luoneet sääntömääräisiä lakeja, jotka lisäävät valvojien määräysvaltaa. On hyvä, että algoritmista pörssikauppaa tutkitaan jatkossa, jotta vältyttäisiin vuoden 2010 kaltaisilta pörssiromahduksilta. Pörssiromahdukset vaikuttavat erittäin voimakkaasti talouteen niin kansallisesti kuin kansainvälisestikin ja niistä toipuminen vie kauan aikaa.

Jatkossa algoritmisia kaupankäyntijärjestelmiä voisi käsitellä enemmän myös ohjelman tasolla. Vielä ohjelmistojen sisältöön liittyvää tietoa ei kuitenkaan juuri ole saatavilla, vaan sisältö keskittyy enemmänkin kaupankäyntijärjestelmien teoriaan ja laskennallisiin seikkoihin. Tämä on kuitenkin ymmärrettävää, sillä ala on erittäin kilpailtua ja sen ympärillä liikkuu valtavat määrät rahaa.

Viiteluettelo

Bruno Biais and Paul Woolley. 2011. High frequency trading. Manuscript, Toulouse University, IDEI, 27 pages.

Andrew C. Burr. 2014. Canceling the order: How high frequency traders are disrupting the derivatives market, and what the regulators can do to stop them. *Express0*.

Rodolfo C. Cavalcante, Rodrigo C. Brasileiro, Victor L. F. Souza, Jarley P. Nobrega and Adriano L. I. Oliveira. 2016. Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications* 55, 194-211.

Nathan Coombs. 2016. What is an algorithm? financial regulation in the era of high-frequency trading. *Economy and Society* 45, 2, 278-302.

Atul Deshpande and B. Ross Barmish. 2016. A general framework for pairs trading with a control-theoretic point of view. 2016. In: *Proc. of IEEE Conference on Control Applications*, 761-766.

Yong Hu, Kang Liu, Xiangzhou Zhang, Lijun Su, E.W.T. Ngai and Mei Liu. 2015. Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing* 36, 534-551.

Shingo Mabu, Masanao Obayashi and Takashi Kuremoto. 2015. Ensemble learning of rule-based evolutionary algorithm using multi-layer perceptron for supporting decisions in stock trading problems. *Applied Soft Computing* 36, 357-367.

Giuseppe Nuti, Mahnoosh Mirghaemi., Philip Treleaven and Chaiyakorn Yingsaeree. 2011. Algorithmic trading. *Computer* 44, 11, 61-69.

Philip Treleaven, Michal Galas, and Vidhi Lalchand. 2013. Algorithmic trading review. *Communications of the ACM* 56, 11, 76-85.

Wikipedia. 2017. Pörssi. https://fi.wikipedia.org/wiki/P%C3%B6rssi. Luettu 25.5.2017.

Yesha Yadav. 2015. How algorithmic trading undermines efficiency in capital markets. *Vanderbilt Law Review* 68, 6, 1607-1671.

Ihmispelaajan toimintoihin reagoiva tekoäly ja vaikeustason skaalaus reaaliaikaisissa strategiapeleissä

Enni Salmi

Tiivistelmä

Reaaliaikaiset strategiapelit ovat nousseet yhdeksi suosituimmista peligenreistä viime vuosikymmenien aikana, mutta ongelmana on edelleen pelien tekoälyjen yksinkertaisuus. Eksperttipelaajat joutuvat pelaamaan vain toisia ihmispelaajia vastaan, sillä tekoäly ei tarjoa heille tarpeeksi haastetta. Tässä tutkielmassa esitellään joitakin tekoälyjä, joista on pyritty tekemään eri tilanteisiin sopeutuvia ja pelaajan taitotasoon mukautuvia käyttäen esimerkiksi koneoppimisen tekniikoita ja dynaamista vaikeustason skaalausta.

Avainsanat ja -sanonnat: strategiapelit, tekoäly, koneoppiminen.

1. Johdanto

Reaaliaikaiset strategiapelit (real-time strategy games, lyhyesti RTS-pelit) ovat lisänneet suosiotaan viimeisten vuosikymmenien aikana. Jotkut pelaajat omistautuvat peleille täysin ja oppivat jatkuvasti uusia strategioita ja hyviä menetelmiä menestyä peleissä. Tässä suhteessa pelien tekoäly on ihmispelaajia huomattavasti jäljessä. Kuinka voimme kehittää reaaliaikaisten strategiapelien tekoälyä niin, että ne osaisivat tarjota haastetta kokeneillekin pelaajille käyttäytymättä kuitenkaan liian armottomasti aloittelijaa vastaan?

Flow-tilan ylläpitäminen on erityisen tärkeää pelin kiinnostavuuden ylläpitämiseksi. Liian helppo peli ei anna tarpeeksi haastetta ja sitä kautta onnistumisen iloa, kun taas liian vaikean pelin pelaaminen voi helposti johtaa turhautumiseen. Yksinkertainen, tietyistä toimintasekvensseistä koostuva tekoäly ilman koneoppimista tai strategista suunnittelua tekee samat ratkaisut useasti, ja pelaajan on liian helppo oppia ennustamaan sen valitsemat toiminnot tietyssä tilanteessa [Aiolli and Palazzi 2008]. Ihannetilanteena voidaan siis pitää sitä, kun peli on mukautettu juuri pelaajan taitotasoa vastaavaksi.

Tämän tutkielman luvuissa 2 ja 3 käsittelen sellaisten tekoälyjen rakentamista, jotka osaavat reagoida vastustajan tekemiin valintoihin sekä ottaa huomioon vastustajan strategian ja taitotason. Tämän lisäksi tarkastelen luvussa 4 hieman vaikeustason skaalausta ja muiden peligenrejen peleissä käytettyjä menetelmiä.

2. RTS-pelit

Reaaliaikaiset strategiapelit ovat strategiapelien alalaji, jonka ominaispiirteisiin kuuluvat useista eri asioista huolehtiminen samanaikaisesti reaaliajassa sekä mahdollisuus voittaa vastustajat eri strategioita hyödyntäen. Peleissä on useita erilaisia *yksiköitä* (units), kuten erilaisia olentoja ja rakennuksia, jotka suorittavat erilaisia toimintoja. Pelin toiminnot ovat eri yksiköille annettavia komentoja, jotka voidaan jakaa useisiin eri osa-alueisiin kuten talouden ylläpitämiseen, tutkimusten tekemiseen parempien aseiden, rakennusten ja muiden vastaavien rakentamisen mahdollistamiseksi sekä taktisten hyökkäysten suorittamiseen vastustajan joukkoja vastaan [Weber *et al.* 2011]. RTS-peleissä hyvä suoriutuminen vaatii pelaajalta kykyä erikoistua näihin osa-alueisiin olemalla vastustajaansa nopeampi ja taktisempi samalla edeten kohti lopullista tavoitetta, joka yleensä on vastustajan armeijan voittaminen.

RTS-peleissä pelaajalla on yksi tai useampi vastustaja, joka voi olla joko tekoälyn tai ihmisen ohjaama. Tämän lisäksi useimmissa RTS-peleissä on myös mahdollista liittoutua muutaman muun pelaajan kanssa. Lopullinen tavoite on saavuttaa aseellinen tai alueellinen ylivoima vastustajiin nähden [Balla and Fern 2009]. Tärkeä osa pelissä selviytymistä on sekä puolustuksellisen että hyökkäyksellisen suunnittelun taitaminen. Kaikki pelaajat liikkuvat samanaikaisesti ja useat toiminnot, kuten rakentaminen, vaativat jonkin verran aikaa valmistuakseen [Ontañón *et al.* 2013]. Pelin reaaliaikaisuus vaatii pelaajalta nopeita päätöksiä, sillä jokainen tuhlattu sekunti antaa vastustajalle aikaa edetä pidemmälle. Tämän lisäksi useimmissa RTS-peleissä pelaaja näkee vain yksiköidensä kartoittamaan alueen kartasta, jolloin osan yksiköistä lähettäminen tiedustelijoiksi on tärkeää suuremman näkökentän saavuttamiseksi.

RTS-pelit ovat kasvattaneet suosiotaan vuosien myötä, ja niin massiivisen tunnettuja moninpelejä kuin niiden kloonejakin on kehitetty paljon. Tässä tutkielmassa keskityn erilaisiin tekoälyihin Wargus- ja StarCraft- peleissä (kuvat 1a ja 1b). Wargus on suositusta Warcraft II -pelistä tehty modi (sanasta *modification*, tarkoittaa peruspelistä tehtyä muunnelmaa), jonka avointa lähdekoodia voi muokata vapaasti tehden siitä erityisen hyvän pohjan erilaisten tekoälyjen kokeiluun [Aha *et al.* 2005]. Wargus on hyvin tyypillinen reaaliaikainen strategiapeli, jossa on aiemmin mainitut ominaisuudet: armeijan ohjaaminen, taloudellinen kehitys, maailman kartoitus sekä erilaisten rakennusten ja yksiköiden rakentaminen.

StarCraft on Blizzard Entertainmentin vuonna 1998 julkaisema suosittu RTS-peli, jossa pelaajan on valittava rotunsa kolmesta eri vaihtoehdosta: terraneista, protosseista ja zergeistä [Ontañón *et al.* 2013]. Jokaisella rodulla on omat

vahvuutensa: terranit ovat tasapainoinen valinta monikäyttöisyytensä vuoksi, protossit ovat vahvoja varjopuolenaan hitaat rakennusajat ja zergit ovat fyysisiltä ominaisuuksiltaan heikkoja, mutta rakentamisen nopeus mahdollistaa suuren armeijan johtamisen jo varhaisessa vaiheessa. Rodun valinnalla on merkitystä strategian valintaan ja tähän palaan myöhemmin luvussa 3.

Wargusin avoin lähdekoodi ja StarCraftista järjestettävät tekoälykilpailut ovat tehneet näistä kahdesta pelistä hyvän välineen erilaisten tekoälyjen ja niiden toimivuuden testaamiseen. Luvussa 3 keskityn ihmisen toimintoihin reagoiviin tekoälyihin sekä niiden kehittämisen haasteisiin ja luvussa 4 vaikeustason skaalaukseen muun muassa dynaamisen tekoälyn avulla.

Kuva 1a: Wargus [Stratagus]. Kuva 1b: StarCraft [StarCraft].

3. Ihmisen toimintaan reagoivat tekoälyt

Tekoälyn ohjelmoiminen RTS-peleille on osoittautunut ongelmalliseksi niissä esiintyvien useiden eri strategioiden ja toimintojen vuoksi. Eksperttipelaajien pelaamista rajoittaa pakko pelata ihmispelaajia vastaan, sillä tekoäly helposti ennakoitavine strategioineen ei välttämättä tarjoa heille tarpeeksi vastusta [Ontañón *et al.* 2013]. Pelin vaikeuttamiseksi pelin tekoälylle on joskus myös kehitetty epäreiluiksi koettuja ominaisuuksia, kuten koko kartan näkeminen jo pelin alussa, jolloin tekoäly tietää ihmispelaajan tekemisistä ilman tiedustelua.

3.1. Hallittavat osa-alueet

Tässä kohdassa selitän tarkemmin RTS-peleissä menestykseen tarvittavat osa-alueet, joista mainitsin lyhyesti edellisessä luvussa. Eksperttipelaajilla on tietyt strategiat kunkin osa-alueen tehokkaaseen hallintaan, joten tekoälyn kehityksessä olisi otettava nämä huomioon ja pyrittävä soveltamaan niitä tekoälyn käyttöön.

Mikrohallinta

Mikrohallinta (micromanagement) tarkoittaa useiden eri hyökkäysten ja toimintojen samanaikaista hallintaa. Kohdattaessa vastustajan armeija tehokkain suunnitelma on usein useiden eri yksiköiden usuttaminen yhden vihollisyksikön kimppuun [Weber *et al.* 2011]. Tiettyyn viholliseen keskittyminen vähentää tämän tekemää vahinkoa ja vähän energiaa omaaviin vihollisiin keskittyminen pienentää vihollisjoukon kokoa nopeasti. Energia kuvastaa yksiköiden elinvoimaa: kun energialukema putoaa nollaan, yksikkö tuhoutuu tai kuolee. Mikrohallinnan ohella tärkeää on *reaktiivinen kontrolli* (reactive control), jolla tarkoitetaan useiden eri yksiköiden samanaikaista ohjausta ja huomioon ottamista, kuten vihollisen tulen sekä esteiden ja ansojen välttämistä [Ontañón *et al.* 2013].

Maaston analysointi

Maastoa voidaan hyödyntää RTS-peleissä taktisen etulyöntiaseman saavuttamiseksi. Strategisesti järkevien paikkojen valinta rakennuksille ja armeijalle sekä reitin valinta hyökkäykselle ovat kaikki osa maaston analysointia. Useissa peleissä metsässä sijaitseviin yksiköihin on hankalampi osua tuliaseilla, mikä helpottaa. Tämän lisäksi esimerkiksi StarCraftissa alhaalla maastossa sijaitsevat yksiköt eivät näe ylämaastoon, joten myös korkeussuhteita voidaan hyödyntää hyökkäysten suunnittelussa sekä puolustuksessa.

Tiedustelu

Tiedustelu (*scouting*) on pelihahmojen lähettämistä oman reviirin ulkopuolelle näkyvyyden parantamiseksi. Tiedustelulla on myös tärkeä rooli strategian valitsemisessa, sillä tiedustelijoiden avulla voidaan kartoittaa vihollisen hallitsema alue ja saada tietoa esimerkiksi siitä, minkä yksikköjen ja parannusten rakentamiseen tämä keskittyy.

Toimintojen asettaminen jonoihin

Koska RTS-peleissä pelaajan on pystyttävä keskittymään useisiin eri osa-alueisiin samanaikaisesti, toiminnot voidaan asettaa jonoon, jolloin edellisen toiminnon tultua suoritetuksi siirrytään automaattisesti seuraavaan. Tätä hyödyntäen pelaajat pystyvät määräämään esimerkiksi rakentamiskomennot tärkeysjärjestykseen ilman, että heidän tarvitsee jokaisen valmistuneen rakennuksen jälkeen valita erikseen seuraava.

Ihmisen toimintaa matkiva ja siihen reagoiva tekoäly olisi siis sellainen, joka pystyisi hallitsemaan aiemmin mainitsemani osa-alueet eksperttipelaajan tavoin, osaisi mukautua vastustajan strategiaan oppien häviöistään, pystyisi määrittämään rakennuskomennot tärkeysjärjestykseen, osaisi matkia vastustajaansa eikä tarvitsisi huijausta kuten koko alueen näkemistä ilman tiedustelua tarjotakseen ihmispelaajalle haastetta [Weber *et al.* 2011]. Tällaisia tekoälyjä on pyritty rakentamaan, ja niitä on testattu muun muuassa Wargus- ja StarCraft-peleissä, mutta niiden kehittämisessä on omat ongelmansa, kuten massiivinen toimintojen määrä.

3.2. Tekoälytestaukset StarCraftissa

Weber ja muut [2011] tutkivat ihmisenkaltaisen tekoälyn rakentamista StarCraftissa, jonka he valitsivat sen ammattitaitoisen pelaajakunnan tarjoamien pelitoisintojen vuoksi. Tutkimusta varten he käyttivät EISBot-agenttia, jonka ytimenä toimiva reaktiivinen *suunnittelija* (planner) soveltuu erityisen hyvin reaaliajassa toimivien toimintojen suorittamiseen hallinnoiden tavoitteita ja toimintojen suorittamista. Suunnittelijan ulkoiset komponentit on toteutettu käyttäen koneoppimista ja *tapauskohtaista järkeilyä* (case-based reasoning) ja yhdistetty toisiinsa työmuistia käyttämällä. Agentti on jaettu viiteen manageriin, joista kukin huolehtii tietystä pelattavuuden osa-alueesta: strategia-, tuotto-, rakennus-, taktiikka- ja tiedustelumanageriin.

Työmuistin tehtävänä on tallentaa uskomuksia peliympäristöstä mukaan lukien ennakointeja vastustajan käyttämistä strategioista ja yksiköiden sijainneista. Tämän lisäksi toimintojen suunnittelusta vastaavat komponentit tallentavat työmuistiin toimintojen sekvenssejä, jolloin agentti pystyy suorittamaan toiminnot strategisesti tehokkaassa järjestyksessä. EISBotia testattiin 250 eri pelaajaa vastaan, ja taisteluista 33 % päättyivät botin voittoon vastaten täten amatöörikilpapelaajan tasoa.

Ontañón ja muut [2013] tutkivat eri menetelmillä kehitettyjä ihmisen toimintaa matkivia tekoälyjä StarCraft-tekoälykilpailussa. Vuonna 2010 ensimmäistä kertaa järjestetty AIIDE StarCraft AI Competition kerää erilaisia tekoälyjä kilpailemaan toisiaan vastaan turnauksen tavoin järjestetyissä otteluissa. Vuonna 2011 kilpailun voitti Skynet-botti, joka valitsi strategiansa ottelun alussa vastustajan rodun (terranit, protossit, zergit) sekä otteluun arvotun kartan perusteella. Botti kokosi armeijansa mahdollisimman suureksi ennen hyökkäystä ja piti puolensa aggressiivisia vastustajia vastaan vahvojen mutta kömpelöiden yksiköiden tehokkaalla hallinnalla. Ottelussa ammattitason pelaajaa vas-

taan Skynet kuitenkin hävisi heikkoutenaan vaihtaa strategiaa kesken pelin alussa valitun strategian osoittauduttua tehottomaksi.

Shantia ja muut [2011] hyödynsivät neuroverkkoja StarCraftia varten kehittämässään tekoälyssä. Neuroverkot ovat ohjelmoinnissa käytetty keinotekoisista neuroneista koostuvia kokonaisuuksia, jotka käyttävät olemassa olevaa dataa muuttujien välisten riippuvuussuhteiden oppimiseen. Tutkijoiden tavoitteena oli kehittää metodi, joka käyttäisi vahvistusoppimisalgoritmia vihollisjoukkojen muuttuviin strategioihin sopeutumiseen sekä uusien taktiikoiden löytämiseen kokemukseen perustuen. *Vahvistusoppiminen* (reinforcement learning) on koneoppimisen tekniikka, jossa agentti tutkii ympäristön tilaa ja toimii sen mukaisesti pyrkien lopputulokseen, joka on arvoltaan mahdollisimman positiivinen.

Shantia ja muut [2011] käyttivät työssään useita monen kerroksen neuroverkkoja, joista jokainen on erikoistunut tiettyyn toimintotyyppiin. Neuroverkoille syötetään pelin arvioimisen kannalta tärkeää informaatiota, kuten tietoa aseiden tilasta, vihollista vastaan hyökkäävien omien yksiköiden tai liittolaisten lukumäärästä, vihollisten ja omien yksiköiden energiasta sekä ympäristön vaarallisuudesta. Päätöksenteon koodaamiseen he käyttivät tietorakenteena graafia. Koska olisi aikaa vievää ottaa huomioon kaikki ympäristön tarjoama informaatio kerralla, päätöksenteko on jaettu kahteen eri vaiheeseen: kun vihollisjoukot ovat kaukana, päätöksenteko perustuu ainoastaan energian ja vahingon laskemiseen ja kun joukot ovat lähellä, neuroverkot ottavat huomioon kaiken informaation yksikön ympärillä. Jokainen verkko analysoi tietoa spesifistä taistelukentän ominaisuudesta, kuten sijainnista tai aseiden kantomatkasta.

Koska pelissä on useita eri toimintoja, joita suoritetaan sekvensseissä dynaamisessa ympäristössä, tehokas tapa varmistaa agentin käyttäytymisen kehittyminen pelin aikana on käyttää vahvistusoppimista. Neuroverkot määrittävät kullekin toiminnolle arvon, jonka perusteella tekoäly pystyy valitsemaan kulloiseenkin tilanteeseen sopivan toiminnon. Pelisession jälkeen tekoäly päivittää tilan ja toiminnon välisiä suhteita tallentavan funktionsa perustuen pelin aikana suoritettujen toimintojen lopputuloksiin. Tekoäly ei siis unohda pelisession aikana valittujen eri toimintojen painoarvoja tietyissä tilanteissa, vaan päivittää järjestelmänsä, jotta session avulla saatua kokemusta voidaan hyödyntää seuraavissa otteluissa.

Vahvistusoppimisen algoritmina Shantia ja muut käyttivät SARSA-algoritmia (State-Action-Reward-State-Action). Algoritmin perusidea on ennustaa seuraavan kierroksen tilanne sen hetkisen tilanteen (S_1), agentin valitseman toiminnon (A_1), toiminnon seurauksen (R), toiminnon jälkeisen tilan (S_2) ja sen jälkeen valitun toiminnon (A_2) perusteella ja siten valita mahdollisimman tehokkaaseen tulokseen johtavat toiminnot. Toiminnon seuraus (R) laskettiin

energian määrän, jäljellä olevien omien pelihahmojen määrän sekä jäljellä olevien vihollisten määrän perusteella käyttäen kaavaa

R = tehtyVahinko – saatuVahinko + 60 × voitetutViholliset.

Toiminnot ja hahmot, joiden seurauksen arvot olivat suurimpia, olivat siten strategisesti parhaita.

Tekoälyä testattiin sekä kolme vastaan kolme että kuusi vastaan kuusi -skenaarioissa, joissa vastaan taistelevat hahmot olivat kaikki luokaltaan *merijalkaväen sotilaita* (marines). Kolme vastaan kolme -tilanteissa SARSA-algoritmia käyttävä tekoäly voitti 70 % otteluista, täten menestyen huomattavasti perinteistä tekoälyä paremmin.

3.3. Tekoälytestaukset Wargus-pelissä

Aha ja muut [2005] tutkivat *tapauskohtaista suunnitelman valitsemista* (case-based plan selection) reaaliaikaisissa strategiapeleissä t*apauskohtaisen taktikon* (Case-based Tactician, CaT) avulla. CaT keskittyy alusta lähtien pelin voittamiseen, ei niinkään tiettyjen alitoimintojen suorittamiseen. Reaaliaikaisten strategiapelien suuren toimintamäärän vuoksi tietyllä hetkellä mahdollisten toimintojen joukko, eli *valinta-avaruus* (decision space) voidaan laskea kaavalla [Aha *et al.* 2005]

$$O(2^W(A \times P) + 2^T(D+S) + B(R+C)).$$

Kaavassa *W* on *työläisten* (workers) sen hetkinen määrä, *A* työläisten toimintavaihtoehtojen määrä, *P työpaikkojen* (workplaces) keskimääräinen lukumäärä, *T* eri yksiköiden määrä, *D* liikuttavien suuntien lukumäärä, *S* yksiköiden asenne kuten vartioiva tai hyökkäävä, *B* rakennusten lukumäärä, *R* keskimääräinen tutkimuskohteiden määrä rakennuksessa ja *C* keskimääräinen rakennusten rakentamiseen valittujen hahmojen määrä. Vaihtoehtojen määrän vähentämiseksi Aha ja muut käyttivät tapauskohtaista suunnitelman valitsemista, jonka tarkoituksena on valita taktiikka supistetusta valinta-avaruudesta.

Valinta-avaruutta supistettiin käyttämällä tilakohtaista taktiikkakirjastoa ja tilaristikkoa. Tilaristikko on abstraktio tila-avaruudesta, jossa eri tiloja voidaan kuvata solmuina ja siirtymiä tilasta toiseen nuolilla. Esimerkiksi tietyn rakennuksen, kuten myllyn, rakentamisen jälkeen yhtenä mahdollisena siirtymänä on siirtyä rakentamaan keskustornia. Uuteen ristikon tilaan siirryttäessä CaT hakee eri tapaukset ja tallentaa arvot sen hetkisestä pelitilanteesta suunnitelman valitsemisen helpottamiseksi. Näin tekoäly pystyy rakentamaan eri rakennukset ja yksiköt strategisessa järjestyksessä.

CaTia testattiin kahdeksaa eri vastustajaskriptiä vastaan, joiden strategiat pyrittiin ottamaan huomioon vastaten niihin mahdollisimman tehokkaasti. Kaikki vastustajaskriptit poikkesivat toisistaan monipuolisen vastustajajoukon

takaamiseksi. Sadan ottelun jälkeen taktiikkansa tilanteen perusteella valitsevan CaTin voittoprosentti oli 82,4 %. Johtopäätelmänä tästä voidaan siis pitää, että tekoälyn tehokkaampi oman ja vastustajan tilanteen tallentaminen ja sen perusteella jatkotoimintojen valitseminen johtaa perinteisemmän tekoälyn häviöön.

Balla ja Fern [2009] lähestyivät RTS-pelien tekoälyä *automatisoidun taktisen hyökkäyssuunnittelun* (automated tactical assault planning) näkökulmasta. Tutkimuksen tavoitteena oli kehittää automaattinen toiminnot valitseva mekanismi, joka osaa ohjata armeijan joukkoja ja johdattaa ne tekemään tehokkaita hyökkäyksiä tiettyjä vastustajan joukkoja vastaan. Hyökkäysten tehokkaaseen suorittamiseen liittyy useita huomioitavia tekijöitä, kuten ympäristön dynaamisuus ja taipumus muuttua nopeasti, useat eri lopputulokseen vaikuttavat toimijat, tietyn ajan vaativat toiminnot sekä eri yksikkötyyppien väliset erot attribuuteissa, kuten vahvuudessa ja nopeudessa. Sisäänrakennetun mekanismin tehtävänä olisi voittaa vihollisjoukot samalla maksimoiden omien joukkojen jäljelle jäävä energia taistelun jälkeen. Tekoälyn kannalta tällaisen näkökulman omaksuminen tarvitsisi kykyä liikuttaa useita eri pelihahmoja tai armeijoita samanaikaisesti, nopeaa reagointia äkillisesti muuttuviin tavoitteisiin ja odottamattomiin tilanteisiin, kykyä järkeillä ottaen huomioon sen hetkinen peliavaruus sekä väliaikaisten toimintojen toteutusta järkevissä sekvensseissä. Näiden lisäksi mekaanisen suunnittelijan tulisi pystyä toimimaan erilaisten hyökkäyksen tehokkuutta mittaavien funktioiden kanssa.

Balla ja Fern [2009] käyttivät testaamiseen UCT-suunnittelualgoritmia. Jokaisen uuden valintatilanteen alussa UCT rakentaa puun, jonka juuri on sen hetkinen tilanne, kaaret toimintoja ja lehdet tilanteita, joihin toiminnon myötä päädytään. Seuraavan toiminnon valitsemisen helpottamiseksi toimintojen tuloksista tallennetaan arvioitu tulos solmuihin. UCT perustaa valintansa aiemmin läpi käytyihin polkuihin suosien valintoja, jotka ovat aiemmin osoittautuneet kannattaviksi. Jokainen solmu tallentaa siinä käytyjen kertojen lukumäärän sekä sen, kuinka monta kertaa tietty toiminto on suoritettu aikaisemmilla kierroksilla. Algoritmi tukee myös eri vaihtoehtojen tutkimista ja testaamista pelin aikana, sillä mikäli valittavissa on aiemmin käyttämättömiä toimintoja, UCT valitsee niistä yhden satunnaisesti. Toiminnon suorittamisen jälkeen siitä seuraava tila lisätään puuhun, mikäli sitä ei siinä vielä ole.

Algoritmin testaamiseksi Warguksessa tutkijat loivat 12 eri peliskenaariota, jotka erosivat toisistaan yksiköiden lukumääriltä sekä niiden sijoitukselta pelin kartalla. Testauksissa käytettiin kahta erilaista UCT-algoritmia: ensimmäisen (UCT(t)) tehtävänä oli yrittää tuhota vihollisjoukot mahdollisimman vähissä pelisykleissä, toinen (UCT(hp)) pyrki maksimoimaan omien yksiköiden jäljellä

olevan energian määrän vihollisen voittamisen jälkeen. Algoritmien tehokkuutta verrattiin viiteen eri perustaktiikkaan: satunnaisten komentojen antamiseen joutilaille joukoille, lähimpänä olevan vihollisyksikön kimppuun hyökkäämiseen ensimmäiseksi, heikoimman vihollisyksikön kimppuun hyökkäämiseen ensimmäiseksi, perinteiseen tekoälyyn sekä kokeneen ihmispelaajan suoritukseen. Tutkimuksesta kävi ilmi, että ihmisen lisäksi ainoastaan molemmat UCT:t kykenivät voittamaan kaikki erityyppiset vastustajansa. Lopputuloksista on myös tärkeää huomata, että algoritmin optimointi vaikuttaa suuresti sen suoriutumiseen tiettyä ominaisuutta mitattaessa. UCT(t):tä ei ollut optimoitu energiapisteiden maksimoimiseen, joten sen lopputulos energian suhteen ei ollut niin hyvä kuin UCT(hp):lla. UCT(hp) puolestaan tarvitsi voiton saavuttamiseen enemmän aikaa kuin ajankäyttöön keskittyvä UCT(t).

Näistä UCT-testeistä voidaan päätellä, että tehokkaan suunnittelun algoritmit suoriutuivat otteluista paljon perinteistä tekoälyä paremmin. Jokaisessa kahdessatoista tapauksessa UCT-algoritmi oli ainoa mekaaninen suunnittelija, joka kykeni aina löytämään voittavan strategian. Tutkimusta varten käytetty prototyyppi ei kuitenkaan ollut vielä tarpeeksi nopea jokapäiväiseen käyttöön eikä toimintojen suorittamiseen tarvittava laskenta-aika optimaalinen.

4. Vaikeustason skaalaus

Vaikeustason skaalaus on pelin automaattista vaikeuden muokkaamista, jonka tavoitteena on asettaa taso pelaajan taitotasoa vastaavaksi. Vaikeudeltaan sopivan tasoinen peli koetaan usein viihdyttävämmäksi kuin liian helppo tai vaikea [Spronck *et al.* 2004]. Useissa peleissä on aloitettaessa mahdollista valita vaikeustaso omien mieltymyksensä mukaan esimerkiksi helposta, normaalitasoisesta ja vaikeasta. Nykyään on lisääntynyt myös mahdollisuus muuttaa vaikeustasovalintaa kesken pelin, jos alussa valittu taso tuntuu liian helpolta tai vaikealta. Vaikeustason korottaminen tällaisten valikoiden kautta johtaa kuitenkin yleensä vain vihollisten ominaisuuksien kuten aseiden tehon ja nopeuden korottamiseen, jonka vuoksi kokeneen pelaajan on edelleen helppo löytää tekoälyn heikot kohdat ja voittaa ottelu [Aiolli and Palazzi 2008].

Tässä luvussa keskityn koneoppimisen, kuten online-oppimisen, hyödyntämiseen vaikeustason skaalaamiseksi. *Online-oppimisella* (online learning) tarkoitetaan koneoppimisen käyttöä pelaajan toimintoihin sopeutumiseksi pelin aikana [Spronck *et al.* 2004]. Erityisesti tarkastelen vaikeustason dynaamista skriptausta sekä adaptiivista vaikeustason skaalausta, joiden avulla peli pystyy muokkaamaan itse vaikeustasoaan pelaajasta tehtyjen havaintojen perusteella. Lopuksi kerron lyhyesti ja pintapuolisesti hiukan muiden peligenrejen tekoäly-

jen parannuksista ja niiden mahdollisista soveltamismahdollisuuksista RTS-peleihin.

4.1. Dynaaminen skriptaus

Dynaaminen skriptaus (dynamic scripting) on videopeleille suunniteltu online-oppimisen tekniikka, joka on laskennallisesti nopea ja tehokas [Spronck *et al.* 2004]. Skripti ylläpitää useita *sääntökokoelmia* (rulebase) pelin eri vihollistyypeistä. Sääntökokoelmat päivittävät sisältämiään painoarvoja pelin aikana kokemustensa perusteella vastaamaan onnistumisen ja epäonnistumisen todennäköisyyttä. Vihollisen kohtaamisen jälkeen sääntöjen painoarvoja muutetaan vastaamaan sitä, kuinka iso osuus tietyllä säännöllä oli lopputuloksen saavuttamiseksi. Onnistumiseen johtaneiden sääntöjen painoarvoa kasvatetaan, kun taas epäonnistumiseen johtaneiden lasketaan. Myös jäljelle jäävien sääntöjen painoarvoja muokataan tarvittaessa, sillä painoarvojen kokonaissumman on pysyttävä muuttumattomana koko pelin ajan.

Painoarvoja käytetään muun muassa omien yksiköiden toiminnan arvioimiseen. Toiminnon painoarvon suuruus mitataan *sopivuusfunktiolla* (fitness function), joka perustuu siihen, voittiko vai hävisikö, kuoliko vai selvisikö kyseinen yksikkö, kuinka paljon tälle jäi energiaa taistelun jälkeen ja kuinka paljon viholliseen onnistuttiin tekemään vahinkoa.

Painoarvojen käytön tarkoituksena on ikään kuin listata mahdolliset valinnat tehokkuusjärjestykseen, jonka avulla tekoäly pystyy valitsemaan tiettyyn tilanteeseen kannattavimman toiminnon tai strategian. Tämän lisäksi dynaamisen skriptauksen avulla voidaan tarvittaessa myös heikentää tekoälyn strategiaa. Koska taisteluiden tulokset tallennetaan, on helposti nähtävissä, mikäli tuloksissa esiintyi epätasaisuutta: vastustaja jatkuvasti häviää tai voittaa. Mikäli dynaamista skriptausta käyttävän tekoälyn vastustaja jatkuvasti häviää, painoarvojen suuruuksia pienennetään, kun taas tekoälyn hävitessä niitä vastaavasti kasvatetaan. Vaihtoehtoisesti voidaan käyttää dynaamisen skriptauksen menetelmää, jossa taktiikan painoarvon kasvaessa maksimiarvon ohi tekoäly ei pysty enää valitsemaan sitä, vaan siirtyy käyttämään heikompia strategioita, kunnes painoarvojen tasapainotus laskee maksimiarvon ohittaneen toiminnon painoarvoa. Tavoitteena on siis mahdollisimman tasavertaisen pelitilanteen saavuttaminen toimintojen tehokkuutta arvioimalla.

4.2 Adaptiivinen vaikeustason skaalaus

Hagelbäck ja Johansson [2009] kehittivät tekoälyn, joka käytti adaptiivista vaikeustason skaalausalgoritmia pelihahmojen vahvuuden säätelemiseen. Testausta varten he käyttivät Open Real Time Strategy (ORTS) -nimistä avoimen lähdekoodin RTS-pelimoottoria ja tankkitaisteluskenaariota. Tankkitaistelun alussa jokaisella pelaajalla on viisi tukikohtaa ja 50 tankkia. Pelin tavoitteena on tuhota kaikki vihollisten tukikohdat samalla ylitsepääsemättömiä jyrkänteitä sekä maastossa liikkuvia lampaita varoen.

Tankkitaistelussa on neljä erilaista skenaariota, jotka on otettava huomioon tekoälyä suunniteltaessa: liikkuviin objekteihin törmäämisen varominen, vihollisjoukkojen metsästys, jyrkänteiden varominen sekä omien tukikohtien puolustaminen. Tämän mahdollistaminen tekoälyn toiminnassa on toteutettu luomalla *strateginen kenttä* (strategic field), joka houkuttelee tekoälyn tankkeja, sekä *karkottava voima* (repelling force), joka asetetaan muun muassa lampaille ja jyrkänteille. Strategisen kentän tehtävänä on johdattaa omat joukot vihollisen alueelle, kun taas karkottavan voiman on tarkoitus pitää tekoälyn tankit tarpeeksi kaukana sillä ympäröidyistä objekteista, jotta tankit pystyisivät etenemään sujuvasti jäämättä jumiin.

Hagelbäck ja Johansson käyttivät testeissään viittä erilaista bottia, joiden jokaisen toiminta perustui edellä mainittuun käyttäytymiseen. Sen lisäksi jokaisen yksikön toimintaa rajoitettiin asettamalla botin vahvuus pelin alussa ja arpomalla jokaisen *ajanjakson* (time frame) alussa jokaiselle pelinappulalle satunnainen numero. Mikäli satunnainen numero on vahvuutta suurempi, pelinappula ei tee mitään kyseisen ajanjakson aikana, jos taas numero on vahvuutta pienempi, pelinappula toimii tavalliseen tapaansa. Kahden botin vahvuus oli vakio, kolmen muun vahvuutta säädeltiin dynaamisesti. Vakiovahvuuksisista toinen oli vahvempi ja toinen heikompi. Vahvuutta säätelevien bottien välillä erot olivat oppimistahdissa.

Vahvuutta säätelevä algoritmi laskee suhteellisen vahvuuden tietokoneen ja ihmispelaajan välillä jokaisen ajanjakson aikana. Nollaa suurempi arvo tarkoittaa pelaajan johtotilannetta, kun taas nollaa pienempi tekoälyn johtoasemaa. Algoritmi muokkaa vahvuuden arvoa tarpeen mukaan, säätäen arvon esimerkiksi hieman nollaa suuremmaksi antaen ihmispelaajalle pienen johtoaseman.

Tutkimuksen koehenkilöitä olivat yliopisto-opiskelijat, jotka pelin pelaamisen jälkeen arvioivat muun muassa pelitilanteiden haastavuutta ja hauskuutta. Tutkimuksen tulokset osoittavat, että pelitilanteeseen sopeutuvat botit koettiin parhaimmiksi vastustajiksi, sillä niitä ei koettu liian helpoiksi eikä liian vaikeik-

si. Vakiovahvuudelliset botit sen sijaan olivat osan mielestä liian vaikeita tai tylsiä vastustajia, koska ne eivät tuntuneet vastaavan pelaajan omaa taitotasoa.

4.3 Mekanismit muissa peligenreissä ja niiden soveltamismahdollisuudet RTS-peleihin

Hunicke ja Chapman [2004] tutkivat dynaamista vaikeustason säätöä erilaisten menetelmien avulla. Heidän tutkimuskohteenaan oli ammuskelupeli Half Life. Tutkimuksessa käytettiin Hamlet-nimistä vaikeustason säätelijää, joka analysoi pelaajan tavaraluetteloa ja toimii sen perustalta vaikuttaen yleisesti pelin vaikeustasoon.

Hamlet on erinäisistä kirjastoista koostuva järjestelmä, joka sisältää funktioita pelaajan tarkasteluun ja pelitilanteen muokkaamiseen. Järjestelmä tallentaa jatkuvasti tilastollista dataa pelitilanteesta ja estimoi pelaajan tulevaisuuden tilanteen sen avulla muokaten pelin asetuksia tarvittaessa. Päätehtävänä on tunnistaa tilanteet, joissa pelaaja jatkuvasti etenee kohti tilannetta, josta ei ole hänelle mitään hyötyä, tai joka ei auta häntä etenemään kohti lopullista tavoitetta millään tavalla. Hamlet tallentaa tietoa pelaajan sen hetkisestä sijainnista pelissä, kuinka usein pelaajan hahmo on kuollut kyseisellä alueella, kuinka monta kertaa kyseinen kohtaus on jouduttu toistamaan sekä kuinka monta kertaa Hamlet on joutunut puuttumaan peliin aiemmin. Huomatessaan pelaajan olevan jumissa samassa kohdassa pitkään järjestelmä reagoi lisäämällä ensiapupakkauksen lähistölle, vahvistamalla pelaajan ammuksia tai heikentämällä vihollisten tekemää vahinkoa. Mikäli näistä muutoksista ei ole apua, järjestelmä voi lisäksi heikentää vihollisten vahvuutta, lisätä vihollisten ruumiista saatujen esineiden määrää tai korvata vihollistyypin helpommalla.

Koska myös RTS-peleissä vihollisten ja pelaajan tekemien hyökkäysten vahvuudella on suuri merkitys lopputuloksen kannalta, voitaisiin Hamletin taktiikkaa ainakin näiltä osin soveltaa myös tähän peligenreen. Haastavan tästä tosin tekee se, että RTS-pelit nimensä mukaisesti toimivat reaaliaikaisesti, ja sen vuoksi mikäli kaikki pelaajan hahmot kuolevat, peli päättyy, eikä hän pysty yrittämään uudestaan taistelua edeltäneestä vaiheesta. Tämän vuoksi järjestelmän olisi hankala tarkkailla, missä pelin kohdissa pelaajalla on jatkuvasti vaikeuksia. Mikäli pelaaja ei kuitenkaan menetä kaikkia pelihahmojaan taisteluissa eikä peli siten pääty, Hamletin kaltainen järjestelmä voisi kiinnittää huomiota siihen, mihin tilanteeseen taistelut tekoälyn kanssa yleensä päättyvät. Mikäli tekoälyn joukot ovat taisteluissa voitokkaampia, voisi järjestelmä reagoida Hamletin tavoin laittamalla vihollisen tuottamaan heikompia hahmoja tai vähentämällä niiden tekemää vahinkoa.

Aiolli ja Palazzi [2008] kehittivät tietokoneella toimivan version Ghosts -pelistä, jonka tekoäly käytti koneoppimista vastustajan voittamiseksi. Ghosts on alun perin lautapelinä pelattu peli, jossa molemmilla pelaajilla on neljä hyvää ja neljä pahaa kummitusta. Pelissä liikutetaan jokaisella vuorolla yhtä kummitusta eteenpäin ja tavoitteena on ryöstää kaikki vastapelaajan hyvät kummitukset tai saada vastustaja ryöstämään kaikki pelaajan pahat kummitukset niiden ruutuun siirtymällä. Toinen tapa voittaa on saada siirrettyä joku omista hyvistä kummituksista vastapelaajan nurkkaruutuun. Pelaajat eivät tiedä mitkä vastustajan kummituksista ovat hyviä ja mitkä pahoja.

Peliä varten kehitetty tekoäly pyrki saamaan selville ihmisvastustajan kummitusten luonteen tämän pelitapaa tarkkailemalla. Riippuen siitä pelaako pelaaja aggressiivisesti tai bluffaamalla, hän liikuttaa samantyyppisiä kummituksia usein samalla tavoin tietyissä tilanteissa. Koneoppimista käyttämällä tavoitteena oli tallentaa tieto vastustajan tekemistä siirroista tietyissä tilanteissa ja käyttää näin saatua tietoa tulevissa otteluissa vastustajan kummituksen tyypin ennustamiseen.

RTS-peleissä vastaavanlaisen tekoälyn käyttäminen vaatisi sen kykyä pystyä arvioimaan vastustajan tekemät valinnat tämän käyttäytymisen ja aikaisempien pelisessioiden perusteella. Pelaajalla saattaa olla tietty strategia, jota hän käyttää jokaisessa tai lähes jokaisessa pelisessiossa, kuten tiettyjen yksiköiden tuottaminen tietyssä järjestyksessä. Mikäli pelaaja on jo voittanut useita kertoja tätä strategiaa hyödyntäen ja haasteen lisääminen olisi siten tarpeen, tekoäly voisi ennustaa vastustajan seuraavaksi tuottamat yksiköt ja pyrkiä tuottamaan yksiköitä, jotka ovat erityisen vahvoja näitä vastaan. Ongelmallisen tästä kuitenkin tekee se, että samaa pelaajaa vastaan tarvittaisiin usea ottelu ennen kuin tekoäly saisi tarpeeksi dataa vastustajan strategian kaavan selvittämiseen ja sen tehokkaaseen voittamiseen.

5. Yhteenveto

Tässä tutkielmassa keskityin lähinnä tekoälyjen yksinkertaisuuteen ja siitä johtuvaan haasteen puuttumiseen eksperttipelaajia vastaan taistellessa. Parempia tekoälyjä, jotka osaisivat valita strategian ihmisvastustajan toiminnot huomioon ottaen, tarvittaisiin kokeneempien pelaajien haastamiseksi. Erilaisia koneoppimisen menetelmiä käyttäen tekoäly on onnistunut tarkkailemaan vastapelaajan toimintoja, ja niihin reagoimalla onnistunut lisäämään haastavuuttaan. Tällaisia tekoälyjä olisi hyvä kehittää lisää ja tulevaisuudessa olisi toivottavaa korvata nykyisin yleiset suoraviivaiset ja ennalta arvattavat tekoälyt strategian vaihtoon ja vaikeustason skaalaamiseen kykenevillä.

Tekoälyjen kehityksessä on kuitenkin myös toinen puoli haastavuuden lisäämisen lisäksi. Järjestelmät ja tekniikat, kuten Hamletin pelinmuokkaus Half-Life-pelissä, saattavat näyttää eksperttipelaajan silmiin huijaukselta. On kuitenkin tärkeää muistaa, että flow-tilan ylläpitäminen on hyvin tärkeää pelin viihdyttävyyden kannalta tarkoittaen myös sitä, että liian haastavaa peliä ei kaikkien mielestä ole mukava pelata. Jokainen pelaaja ei ole eksperttipelaaja eikä välttämättä ikinä sellaiseksi kehity, joten myös dynaaminen pelin helpottaminen olisi hyvä ottaa huomioon, mikäli myös taidoiltaan heikompien pelaajien kiinnostus halutaan pitää pelissä.

Viiteluettelo

Aha David W., Molineaux M. and Ponsen M. 2005. Learning to win: Case-based plan selection in a real-time strategy game. In: *Case-based Reasoning Research and Development*. Springer, 5-20.

Aiolli F. and Palazzi C. 2008. Enhancing artificial intelligence in games by learning the opponent's playing style. In: *New Frontiers for Entertainment Computing*. Springer, 1-10.

Balla R. and Fern A. 2009. UCT for tactical assault planning in real-time strategy games. In: *IJCAI*, 40-45.

Hagelbäck J. and Johansson S. J. 2009. Measuring player experience on runtime dynamic difficulty scaling in an RTS game. *Computational Intelligence and Games, 46-52.*

Hunicke R. 2005. The case for dynamic difficulty adjustment in games. In: *Proc. of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology.* ACM, 429-433.

Ontañón S., Synnaeve G., Uriarte A., Richoux F., Churchill D. and Preuss M. 2013. A survey of real-time strategy game AI research and competition in stacraft. *IEEE Transactions on Computational Intelligence and AI in games*. 5(4), 293-311.

Shantia A., Begue E. and Wiering M. 2011. Connectionist reinforcement learning for intelligent unit micromanagement in starcraft. In: *Proc. of the 2011 International Joint Conference on Neural Networks.*

Spronck P, Sprinkhuizen-Kuyper I and Postma E. 2004. Difficulty scaling of game AI. In: *Proc. of the 5th International Conference on Intelligent Games and Simulation*, 33-37.

StarCraft. StarCraft Wiki. http://starcraft.wikia.com/wiki/StarCraft_II. Checked 20.5.2017.

Stratagus. Stratagus (Wargus). http://anotherguest.blogspot.fi/2011/11/stratagus-wargus-warcraft-2-2105.html. Checked 20.5.2017.

Weber BG, Mateas M. and Jhala A. 2011. Building human-level AI for real-time strategy games. In: *Proc. of the AAAI Fall Symposium: Advances in Cognitive Systems.* Vol 11.

Internetin sosiaalisten verkostojen simulointi agenttipohjaista mallintamista ja simulaatiota käyttäen

Sami-Santeri Svensk

Tiivistelmä

Usealla tieteenalalla on yhä tärkeämpää ymmärtää sosiaalisten verkostojen merkitys ihmisten käyttäytymisessä. Sosiaalisten verkostojen kaltaisten rakenteiden analysointi on niiden kompleksisuuden takia vaikeaa. Agenttipohjainen mallintaminen ja mallin pohjalta suoritettu simulaatio on yleisimpiä menetelmiä sosiaalisten verkostojen tutkimiseen. Sosiaalisen median käytön yleistyessä niiden sisältämät sosiaaliset verkostot ovat tulleet arkipäiväiseksi käyttäjille, ja ne tarjoavat samalla realistista dataa sosiaalisten verkostojen käyttäytymisen tutkimiseen mikro- ja makrotasolla. Tämä tutkielma on kirjallisuuskatsaus agenttipohjaiseen mallintamiseen ja simulointiin.

Avainsanat: Sosiaalinen media, sosiaalinen verkosto, agenttipohjainen mallintaminen ja simuloiminen, Monte Carlo -menetelmät

1. Johdanto

Internetin sosiaaliset verkostot ovat koko ajan tulleet yhä tärkeämmäksi ja näkyvämmäksi osaksi ihmisten ajankäyttöä sosiaalisten verkoston sivustojen kautta. Kiinteänä osana sosiaalisen verkoston sivustoihin kuuluu palveluihin integroitu sosiaalinen verkosto.

Erilaisten sosiaalisten verkostojen sivustojen käyttäjiä on maailmanlaajuisesti satoja miljoonia, ellei jopa miljardeja. Maailmanlaajuisesti suosituimpia sosiaalisen verkoston sivustoja ovat tammikuussa 2017 tehdyn aktiivisia käyttäjiä listaavan tilaston mukaan Facebook, Facebookiin kiinteästi sidoksissa oleva Facebook Messenger, WhatsApp, QQ, WeChat ja QZone, joista Facebook ja WhatsApp yltävät yli miljardin käyttäjän rajaan [statista.com 2017]. Maailmanlaajuisesti suosituimpien sivustojen listaa silmäillessä on huomionarvoista, että useat edellä mainitut sivustot keräävät käyttäjäkuntansa pääasiassa Euroopan ja Pohjois-Amerikan ulkopuolelta.

Ensimmäiset sosiaalisen verkoston sivustoiksi luokiteltavat sivustot ovat jo 1990-luvulta, mutta sosiaalista mediaa ja siihen liittyviä sosiaalisia verkostoja voi pitää 2000-luvun ilmiönä, jolloin niistä tuli valtavirtaa MySpacen ja Friendsterin kaltaisten uraauurtavien sivustojen kautta [boyd and Ellison 2007]. Sosiaalisen verkoston sivustojen käyttäjämäärät kasvavat edelleen huolimatta siitä, että uudemmat sosiaalisen verkoston sivustot syrjäyttävät vanhat sivustot. So-

siaalisen verkoston sivustojen käyttäjämäärien jatkaessa kasvuaan sosiaalisen median mahdollistamien internetin sosiaalisten verkostojen analysoinnista on tullut yhä tärkeämpi tutkimuskohde [Ryczko *et al.* 2017].

Käyttäjien jakaessa ja silmäillessä sosiaalisen verkoston sivuston sisältöä ja käyttäen sosiaalisen verkostojen sivustojen työkaluja keskinäiseen vuorovaikutukseen heidän toimintansa tallennetaan. Tästä tiedosta voidaan sosiaalisten verkostojen sivustojen API-rajapintoja käyttämällä kerätä tietoa käyttäjistä ja käyttää tietoa päätelmien tekoon käyttäjien toiminnasta, käyttäjiä kiinnostavasta sisällöstä, käyttäjien välisistä yhteyksistä ja niissä tapahtuvista muutoksista ajan kuluessa [itk.fi].

Tässä tutkielmassa keskityn käsittelemään mikroblogipalvelu Twitterin simulointia. Tutkielmassani selvitän, miten Twitterin kaltaisia sosiaalisen verkoston sivustoja on mahdollista simuloida keinotekoisilla malleilla ja millaisia ajattelutapoja internetin sosiaalisten verkostojen simulointiin liittyy. Luvussa 4 esitelen simuloimisen tarpeellisuuden eri tieteenalojen näkökulmasta. Painopisteeni tutkielmassa on informaation leviämisessä internetin sosiaalisissa verkostoissa. Siihen liittyen esittelen luvussa 6 keinoja vaikutusvaltaisimpien verkoston yksilöiden havaitsemiseen. Lopuksi luvussa 7 pohdin esittelemiäni keinoja ja tutkimuksissa esitettyjä väitteitä.

2. Terminologiaa

Sosiaalisen verkoston sivusto tarkoittaa verkossa toimivaa palvelua, jossa käyttäjät voivat tuottaa ja jakaa omaa sisältöä ja vastaanottaa muiden käyttäjien tuottamaa ja jakamaa sisältöä. Koska sosiaalisen verkoston sivustot ovat suhteellisen uusi ilmiö, termille ei vielä ole löytynyt vakiintunutta määritelmää. Joskus sosiaalisen verkoston sivustosta puhuttaessa käytetään termiä sosiaalinen media [boyd and Ellison 2007].

boyd ja Ellison [2007] antavat sosiaalisen verkoston sivuille kolme määritelmää: 1) Sosiaalisen verkoston sivustot ovat vuorovaikutteisia web-pohjaisia palveluita, jotka antavat käyttäjien luoda julkisen tai puolijulkisen profiilin palvelussa. 2) Käyttäjä voi määrittää listan käyttäjistä, joiden kanssa käyttäjä on yhteydessä. Toisin sanoen käyttäjän on mahdollista luoda oma sosiaalinen verkosto palvelussa. 3) Käyttäjä voi selata ja käydä läpi omaa ja muiden verkostoa. Palvelun yhteyksien luonne ja nimistö saattavat vaihdella sivustosta riippuen. Obar ja Wildman [2015] nostavat esille näiden kolmen määritelmän lisäksi käyttäjän luoman sisällön.

Graafi on abstrakti tietorakenne, joka koostuu äärellisestä määrästä solmuja, ja niitä yhdistävistä kaarista. Jos graafin sisältämille kaarille asetetaan jokin arvo, puhutaan painotetusta graafista. Graafin kaari voi olla suunnattu tai suun-

taamaton. Suunnatulla kaarella on suunta, jolloin kaarella on lähtö- ja maalisolmu. Yleisenä käytäntönä on, että graafi sisältää vain joko suunnattuja tai suuntaamattomia kaaria. Jos kaarilla on suunta, puhutaan suunnatusta graafista. Tällöin kahden solmun välillä voi olla kaksi kaarta, joiden suunnat ovat toisiinsa nähden päinvastaiset. Vastaavasti jos kaarilla ei ole suuntaa, graafista käytetään nimitystä suuntaamaton graafi. Graafissa olevan solmun naapureiksi luetaan ne solmut, jotka on yhdistetty kaarilla kyseiseen solmuun. Graafista haetaan tietoa hakuoperaatioilla. Yleisimpiä hakuoperaatioita on leveyshaku ja pituushaku. Leveyshaku alkaa ennalta päätetystä solmusta, ja etenee rekursiivisesti jokaiseen käymättömään naapurisolmuun, kunnes jokainen graafin solmu on käyty läpi. Graafiin liittyy oleellisesti lyhyimmän polun etsiminen kahden graafin solmun välille, mutta tätä käsittelen tarkemmin solmun keskeisyyden määrittämisen yhteydessä luvussa 6.

Monte Carlo -menetelmillä tarkoitetaan algoritmeja, jotka perustuvat satunnaisten arvausten tekemiseen. Monte Carlo -menetelmässä arvotaan satunnaislukugeneraattorilla arvot simulaatiossa esiintyville muuttujille niiden vaihteluvälin ja annetun todennäköisyysjakauman mukaan [Kujala 2000]. Jotta Monte Carlo -menetelmän tuottama satunnaismuuttuja olisi riittävän tarkka, simulaatio suoritetaan useita kertoja, tyypillisesti satoja kertoja. Lopuksi Monte Carlo -menetelmän tuottamista satunnaismuuttujien arvoista lasketaan keskiarvo, tai joku muu tilanteeseen sopiva laskutoimitus.

Vaikka Monte Carlo -menetelmä perustuu satunnaislukuihin, sitä käytetään reaalimaailmaa vastaavien tulosten laskemiseen. Esimerkiksi Katzgraber [2011] käyttää Monte Carlo -menetelmää piin arvon laskemiseen. Pseudokoodi piin laskemiseen on kuvassa 1.

```
Algoritmi pii_arvo
osumat ← 0
yritykset ← 10000
laskuri ← 0
while (laskuri < yritykset) do
        x ← random(0,1)
        y ← random(0,1)
        if(x**2 + y**2 < 1) then
        osumat++
laskuri++
done
pii ← 4*osumat /yritykset
```

Kuva 1. Piin arvon laskeminen Monte Carlo -menetelmällä [Katzgraber 2011].

Iteraatiota suoritetaan ennalta-asetettujen yritysten verran. Jos muuttujien x ja y neliöiden summa on enemmän kuin yksi, kasvatetaan osumat-muuttujan arvoa. Lopuksi piin arvo lasketaan pseudokoodin mukaisella tavalla.

Monte Carlo -menetelmän yhteydessä puhutaan usein Markovin ketjusta. Markovin ketjua käyttävä Monte Carlo -menetelmä eroaa Monte Carlo -menetelmästä siten, että uudet tapahtumat linkittyvät aikaisempiin tapahtumiin. Kuvassa 1 esitetyssä pseudokoodissa tämä näkyisi siten, että while-silmukan sisälle lisätään ehtolauseke, jossa arvotun muuttujan arvoa verrataan edelliseen muuttujan arvoon ja edetään sen mukaan [Katzgraber 2011].

3. Sosiaalisen verkoston ominaisuuksia

3.1. Sosiaalisista verkostoista yleisesti

Sosiaalinen verkosto on teoreettinen rakenne, joka koostuu sosiaalisista yksilöistä, kuten ihmisistä ja organisaatioista, sekä verkoston yksilöitä yhdistävistä suhteista. Verkoston yksilöillä on suhteet toistensa kanssa, joka voi olla ystävyyssuhde, jäsenyys esimerkiksi ryhmään tai muu reaalimaailmasta tuttu riippuvuutta kuvaava suhde [Sun *et. al* 2015].

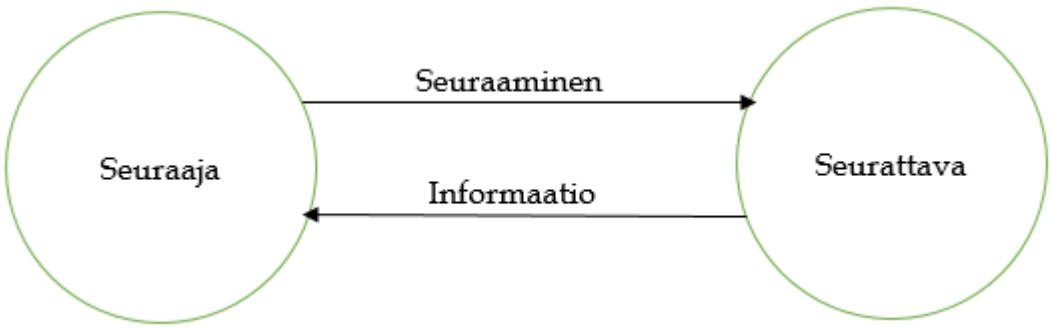
Luonnollinen esitystapa sosiaaliselle verkostolle on graafi, jonka solmut ovat sosiaalisen verkoston yksilöitä ja kaaret yksilöiden välisiä suhteita. Yksinkertaistettuna sosiaalisen verkoston voi kuvata suuntaamattomana graafina, jolloin ensisijaisesti tärkeää on hahmottaa, kuinka verkosto on rakentunut. Reaalimaailmassa kahden yksilön välisen suhteen voi katsoa riippuvan yksilön näkökulmasta, jolloin suhteella voi olla eri merkitys suhteen muodostaville yksilöille. Tällöin suunnattu graafi havainnollistaa yksityiskohtaisemmin käyttäjien välisiä riippuvuussuhteita ja niissä tapahtuvia muutoksia.

3.2. Internetin sosiaaliset verkostot

Internetin sosiaaliset verkostojen sivustot ovat viime vuosina tuoneet reaalimaailman sosiaaliset verkostot internet-ympäristöön. Vuorovaikutuksen keinot verkostossa ovat laajentuneet internetin sosiaalisten verkostojen palveluiden tarjotessa yhä enemmän uusia mahdollisuuksia vuorovaikutukseen [Blanco-Moreno *et al.* 2011]. Internetin sosiaalisten verkostojen myötä mielenkiintoista sisältöä on mahdollista jakaa käyttäjien kesken nopeasti. Myös suurten henkilökohtaisten sosiaalisten verkostojen rakentaminen on internetin sosiaalisten verkostojen helpon yhdistettävyyden myötä mahdollista. Reaalimaailmassa esimerkiksi useiden satojen henkilöiden laajuiset kontaktiverkostot ovat käytännössä mahdottomia [Wang and Taylor 2013].

Käyttäjien väliset suhteet on helpompi hahmottaa internetin sosiaalisissa verkostoissa kuin reaalimaailman sosiaalisissa verkostoissa, koska käyttäjien väliset riippuvuussuhteet ja niiden vahvuuden määrittävät tekijät voidaan havaita ja mitata konkreettisesti. Tällaisia tekijöitä ovat esimerkiksi käyttäjien toisilleen lähettämät viestit tai muu palvelussa tapahtuva vuorovaikutus. Esimerkiksi reaalimaailman kaveruus riippuu yleensä henkilöiden jakamista mielenkiinnon kohteista, eikä toiminnosta, jolla kaveruussuhde voidaan katsoa perustetuksi. Internetin sosiaalisten verkostojen tapauksessa solmujen voi ajatella edustavan yksittäisiä sosiaalisen verkoston sivuston käyttäjiä, ja kaarien heidän välisiä suhteita, joiden merkitys riippuu tarkasteltavasta sosiaalisen verkoston sivustosta.

Twitter on sosiaalisen verkoston sivustoksi luokiteltava palvelu, jossa käyttäjät kommunikoivat korkeintaan 140 merkin mittaisilla ”twiiteillä”. Twiittejä voi levittää eteenpäin uudelleentwiittaamalla tai lainaamalla, niistä voi tykätä ja twiitteihin voi vastata, jolloin vastaukset muodostavat puurakenteen, joka voi sisältää useita haaroja. Käyttäjät voivat seurata muita käyttäjiä (ks. kuva 2), jolloin he näkevät seuraamiensa käyttäjien twiitit ja uudelleentwiitit henkilökohtaisella aikajanallaan. Vastaavasti seuraamisen lopettaminen lopettaa seurattavan jakaman informaation näyttämisen henkilökohtaisella aikajanalla.



Kuva 2. Kahden käyttäjän välinen seuraajasuhde Twitterissä. Seuraamisen jälkeen seuraaja vastaanottaa seurattavan jakamaa informaatiota

Lähtökohtaisesti jokaisen käyttäjän twiitit ovat kaikkien muiden nähtävissä, ellei käyttäjä ole määrittänyt Twitter-profiiliaan yksityiseksi, jolloin vain käyttäjän hyväksymät seuraajat voivat seurata käyttäjää. Twitterissä on myös mahdollista lähettää yksityisviestejä kahden käyttäjän välillä. Yksityisviestit ovat luonnollisesti käyttäjien välisiä, eivätkä ne näy julkisesti keskustelun ulkopuolisille.

Olennaisena osana Twitteriin kuuluvat avainsanat, joiden mukaan käyttäjä voi selata keskusteluaiheita. Twitter poimii käyttäjien twiiteistä avainsanat ja ryhmittelee twiitit avainsanojen mukaisiin listoihin, joiden perusteella on mah-

dollista selailla eri keskusteluaiheita. Erona suosituimpaan sosiaalisen verkoston sivustoon Facebookiin, Twitteriä voi luonnehtia suunnatuksi sosiaalisen verkoston sivustoksi, sillä toisen käyttäjän seuraaminen ei saa seurattua näkemään seuraajiensa jakamaa sisältöä omalle aikajanalleen. Molemmat käyttäjät näkevät toistensa sisällön aikajanallaan, kun kumpikin seuraa toisiaan. Facebookissa seuraaminen tarkoittaa kaverisuhdetta, jonka molempien käyttäjien tulee hyväksyä, jonka jälkeen molemmat käyttäjät näkevät toistensa jakamaa sisältöä omalla aikajanallaan.

4. Internetin sosiaaliset verkostot

4.1. Ongelmia internetin sosiaalisten verkostojen analysoinnissa

Sosiaalisten verkostojen ymmärtäminen on ylipäätään vaikeaa niiden kompleksisen rakenteen ja toiminnallisuuden takia [Zeng *et al.* 2015]. Sosiaaliset verkostot ovat kompleksisia järjestelmiä, mikä tarkoittaa verkoston koostuvan itsenäisesti toistensa kanssa vaikuttavista komponenteista [Macal and North 2010]. Internetin sosiaalisten verkostojen yhteydessä nämä komponentit edustavat sosiaalisen verkoston sivuston käyttäjiä. Sosiaaliset verkostot muuttuvat ajan myötä ja mekanismit, joiden mukaan sosiaaliset verkostot muuttuvat, on tärkeä ymmärtää sosiaalisia verkostoja tutkiessa [Zeng *et al.* 2015].

Sosiaalisia verkostoja on mahdollista mallintaa ja simuloida. Yleinen mallintamismuoto sosiaaliselle verkostolle on graafi, jonka päälle on mahdollista rakentaa ohjelmointikielillä tai simulaatiotyökaluilla simulaatio kuvaamaan graafin sisältämien solmujen välistä vuorovaikutusta. [Zeng *et al.* 2015].

Internetin sosiaalisten verkostojen simulointi rajoittuu yleensä yhteen sosiaalisen verkoston sivustoon, mikä rajoittaa simuloinnin tulosten validisuutta palvelun ulkopuolella. Esimerkiksi Twitter-käyttäjien käyttäytyminen Twitterissä tuskin paljastaa heidän käyttäytymistä reaalimaailmassa, etenkin jos käyttäjä ei ole aktiivinen palvelun käyttäjä. Simulaatioon perustuvia metodeja pidetään kuitenkin tehokkaana tapana ymmärtää sosiaalisten verkostojen kaltaisia monimutkaisia järjestelmiä [Zeng *et al.* 2015].

4.2. Internetin sosiaalisten verkostojen ymmärtämisen merkitys

Sosiaalisten verkostojen ymmärtämisellä olisi suuri merkitys ihmisten käyttäytymisen tutkimiselle. Internetin sosiaaliset verkostot ovat nykyään käytetyin tapa levittää informaatiota, kuten uutisia, uusia ideoita, ja tyypillisenä piirteenä sosiaalisten verkostojen sivustoille, käyttäjien luomaa sisältöä [de C. Gatti *et al.* 2013].

Muun muassa sosiologia ja psykologia hyötyvät sovelluksista, jotka simuloivat sosiaalisia verkostoja ja ennustavat verkoston käyttäytymisen sekä hajautetulla että keskitetyllä tasolla [Blanco-Moreno *et al.* 2011]. Simuloimalla internetin suuria sosiaalisia verkostoja paljastuu informaation etenemisen näissä verkostoissa.

Internetin sosiaalisten verkostojen simuloinnista hyötyy myös liike-elämä, sillä sosiaalisten verkostojen simulointi paljastaa, kuinka mainoskampanjat saavuttavat käyttäjät tehokkaammin internetin sosiaalisia verkostoja pitkin [de C. Gatti *et al.* 2013]. Sosiaalisen verkoston sisältämät suhteet tarjoavat mahdollisuuksia laajemman asiakaskunnan tavoittamiselle. Sosiaalisen verkoston sivustoilla markkinoinnin mielenkiinnon kohteena olisi sivuston sosiaalisen verkoston suosituimmat käyttäjät, joilla on paljon vaikutusvaltaa verkostossa, sen sijaan että mainonta kohdistuisi paljon kuluttaviin käyttäjiin, kuten perinteisessä markkinoinnissa [Zeng *et al.* 2015].

Sosiaalisten verkostojen sisältämiä rakenteita ja yksittäisiä käyttäjiä analysoimalla on mahdollista mitata verkoston yleisiä ominaisuuksia ja ymmärtää verkoston toimintaa kokonaisuutena. Verkoston rakenteesta voi päätellä verkoston dynamiikan ja sen periaatteet informaation leviämisessä [Yang *et al.* 2015].

Koska internetin sosiaalisista verkostoista saatu data on laskennallisesti hankalaa, perinteisten matemaattisten ja tilastotieteellisten mallien käyttö on internetin sosiaalisten verkostojen analysointiin ongelmallista. Perinteiset matemaattiset ja tilastotieteelliset mallit eivät myöskään tarkastele verkostoa luonnollisessa ympäristössä [Bonabeau 2002]. Yleensä ongelmaksi muodostuu myös ajankäyttö, sillä havaintojen saaminen ja johtopäätösten muodostaminen reaalimaailman datasta kestää kauan [Beheshti].

4.3. Sosiaalisten verkostojen simulointi

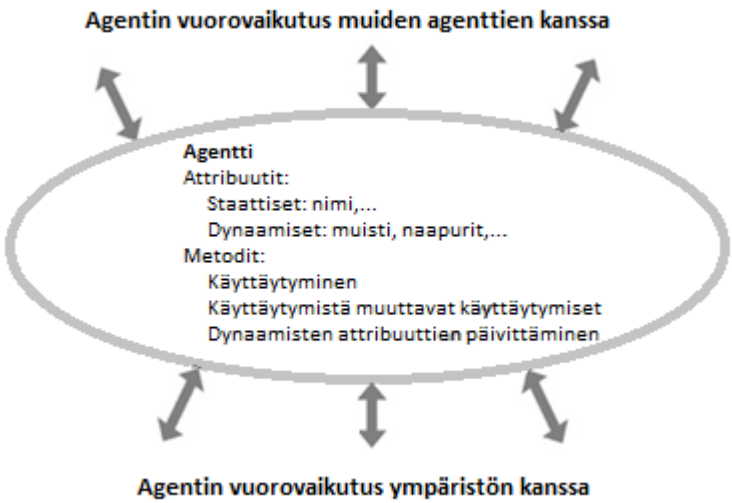
4.3.1. Agenttipohjainen mallintaminen

Vaihtoehtoisena suurten kompleksisten sosiaalisten verkostojen analysoimiseen käytetään agenttipohjaista mallintamista ja simuloimista. Malli on yksinkertaistettu kuvaus koko verkostosta, mutta sisältää laajemman sosiaalisen verkoston pääpiirteet, kuten agenttien ryhmittymisen pienempiin ryhmiin verkoston sisällä ja erikokoiset henkilökohtaiset lähiverkostot [Yang *et al.* 2015].

Agenttipohjaista mallintamista ja simuloimista voi pitää koelaboratoriona reaalimaailman kompleksisten verkostojen analysointiin, mutta tärkein agenttipohjaisen mallintamisen hyöty on sen reaalimaailmaa ja sen ilmiöitä vastaava luonnollinen esitystapa [Bonabeau 2015].

Agenttipohjainen mallintaminen ja simulointi tarkastelee sosiaalista verkostoa hajautetulla tasolla eli verkoston sisältämien yksittäisten agenttien näkökulmasta. Agenttipohjaisen mallintamisen ja simuloinnin tavoite on paljastaa yksittäisen agentin käyttäytyminen ja vuorovaikuttaminen muiden verkoston agenttien kanssa ja näin tutkia, kuinka yksittäisten agenttien käyttäytyminen vaikuttaa koko verkoston käyttäytymiseen [Macal and North 2010].

Tyypillisen agenttipohjaisen mallin ominaisuuksia ovat agenttien joukko, agenttien väliset vuorovaikutuskeinot ja agenttien ympäristö [Macal and North 2010]. Agentit ovat agenttipohjaisessa mallintamisessa ja simuloinnissa autonomisia toimijoita. Ne pystyvät toimimaan itsenäisesti, tekemään itsenäisiä päätöksiä ja toimimaan ilman ulkopuolista vaikutusta [Macal and North 2010].



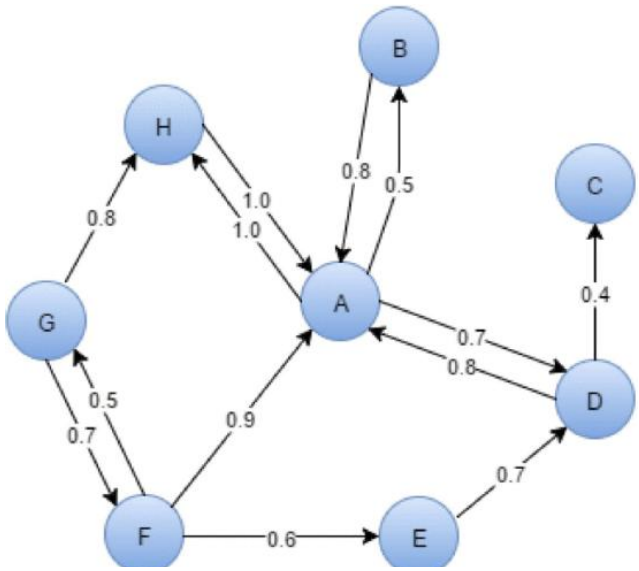
Kuva 1. Agentin määritelmä [Macal and North 2010].

Kuva 3 esittää Macalin ja Northin [2010] agentille antamaa määritelmää, jonka mukaan agentilla on sekä staattisia että dynaamisia attribuutteja. Staattiset attribuutit ovat muuttumattomia, esimerkiksi nimi on staattinen attribuutti. Agentin dynaamiset attribuutit ovat esimerkiksi listoja naapuriagenteista, jotka päivittyvät sen mukaan, mitkä ovat agentin tarkasteltavan hetken naapuriagentit. Agentti sisältää myös metodeja, jotka määrittävät agentin käyttäytymisen, käyttäytymisen muutoksen ja attribuuttien muuttamista koskevat säännöt. Lisäksi agentti vaikuttaa toisten agenttien sekä sen ympäristön kanssa, jossa agentti on. Piskor-Ignatowicz ja Zachara [2016] luettelevat malliin kuuluvan agentin ominaisuuksiksi sen sijainnin, yhteydet muihin agentteihin ja agentin persoonallisuuden. Tämä voidaan kuvata kaavalla A = {L, E, R}, missä L merkitsee agentin sijaintia, E agentin persoonallisuutta ja R kuvaa agentin suhteita toisiin agentteihin. Piskor-Ignatowicz ja Zachara [2016] antavat esimerkkinä

agentin persoonallisuuden, joka vaikuttaa yhteyksien laatuun muiden agenttien kanssa.

Ylipäätään sosiaalisen verkoston simulointi on siis kaksivaiheinen prosessi, jossa ensiksi mallinnetaan sosiaalinen verkosto, jonka jälkeen mallia simuloidaan toteutustavasta riippuen.

Koska yleisin tapa mallintaa sosiaalisia verkostoja on graafi, agenttien voi ajatella olevan graafissa solmuja ja kuvaavan yksilöllistä sosiaalisen verkoston sivuston käyttäjää. Täten agenttien väliset suhteet ovat graafin kaaria, jotka yhdistävät käyttäjät. Kaarilla on mallissa yleensä jokin arvo, joka voi olla solmujen välinen etäisyys tai joku muu kahden solmun välinen arvo. Bahbouhi ja Moussa [2015] käyttävät arvona solmujen edustamien käyttäjien välistä luottamusarvoa, joka kertoo, kuinka paljon agentti luottaa toiseen agenttiin. Bahbouhin ja Moussan [2015] käyttämä graafi on kuvassa 4.



Kuva 2. Suunnattu graafi, jossa kuvattuna painotettujen kaarien arvot [Bahbouhi and Moussa 2015]

Tyypillistä agenttipohjaisissa simulaatioissa on agentteja yhdistävien kaarten arvojen muuttuminen simulaation aikana ja kaarten poistot ja uusien kaarien lisääminen agenttien välille. Kaarten painoarvot määrittyvät agenttien attribuuteista, jotka määrittävät kuinka agentit muodostavat suhteen toisten agenttien kanssa [Piskor-Ignatowicz and Zachara 2016]. Graafi, johon sosiaalinen verkosto mallinnetaan, voi olla simulaation aikana kooltaan muuttumaton tai jatkuvasti muuttuva [Zeng *et al.* 2015].

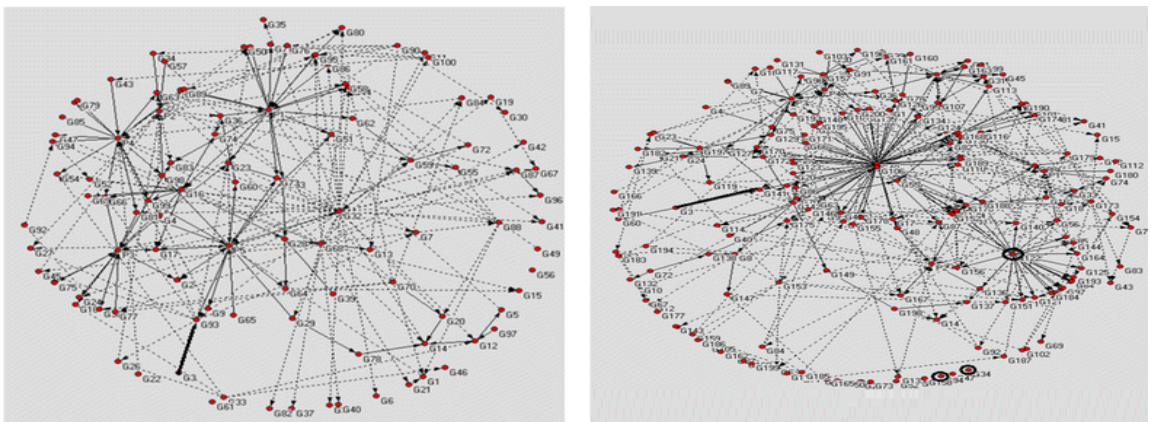
Agenttien toiminta perustuu niille määritettyihin käyttäytymismalleihin, jotka johdetaan agenteille määritetyistä attribuuteista ja metodeista [Macal and North 2010]. Macal ja North [2010] mainitsevat agenttien ominaisuuksiksi autonomisuuden, itseohjautuvuuden, tilan ja sosiaalisuuden. Ensimmäiset kaksi

ominaisuutta korostavat agenttien kykyä toimia itsenäisesti. Kuitenkin agentin sosiaalisuus mahdollistaa agentin olevan kykenevä sopeutumaan ympäristössä tapahtuviin muutoksiin. Edellä mainittujen ominaisuuksien voi katsoa määrittävän agentin tilan, joka Twitterin kaltaista internetin sosiaalista verkostoa simuloidessa voi olla esimerkiksi ”odottaa”, ”kirjoittaa” tai muuta vastaavaa, jolla on luonnollinen vastinpari reaalimaailman tekemisessä.

Agentit ovat vuorovaikutuksessa verkoston agenttien osajoukon kanssa ja tätä osajoukkoa kutsutaan agentin naapurustoksi. Koska agenttipohjainen mallintaminen ja simuloiminen tapahtuu hajautetulla tasolla, yksi agenttipohjaisen mallintamisen pääperiaatteista on, ettei mallissa ole olemassa keskusagenttia, joka ohjaisi agenttien käyttäytymistä. Agentit saavat ainoastaan paikallista informaatiota itseltään tai naapurustostaan [Macal and North 2010].

Simulaatio lähtee liikkeelle agenttien sijoittamisesta ympäristöön, jossa ne jokaisen simulaatiokierroksen aikana kohtaavat muita agentteja ja solmivat uusia suhteita keskenään. Simulaation alussa agentilla on joko satunnaisesti solmittuja suhteita toisiin agentteihin tai ei suhteita ollenkaan [Piskor-Ignatowicz and Zachara 2016].

Agenttipohjaisen mallin simuloiminen edellyttää agenttien toistuvasti suorittavan niille määritellyt tehtävät. Simulointi etenee yleensä aikajanalla portaittain, tapahtumittain, tai diskreettien tapahtumien kautta [Macal and North 2010]. Kuvassa 5 havainnollistetaan sosiaalisen verkoston sisältämän graafin muuttumista kahden ajanhetken välillä.



Kuva 3. Sosiaalisen verkoston rakenne kahtena eri ajanhetkenä [Zeng et al. 2015].

Agenttipohjainen mallintaminen ja simuloiminen on enemmän ajattelutapa kuin teknologia [Bonabeau 2002]. Näin ollen agenttipohjaisia simulaatioita on mahdollista toteuttaa erilaisilla ohjelmointikielillä. Lähes mikä tahansa ohjelmointikieli sopii agenttipohjaisen mallin luomiseen ja simulointiin, mutta useimmin käytetään Python- ja R-kieliä, jotka soveltuvat muita kieliä parem-

min simuloinnin lisäksi myös simulaation ja siitä tehdyn analyysin visualisoimiseen. Bonabeau [2002] huomauttaa, että yleishyödyllisen mallin rakentaminen agenttipohjaista mallia käyttäen on mahdotonta. Mallin on aina palveltava yhtä tarkoitusta, jotta simulaation tulokset ovat tulkittavissa. Yleensä simuloimiseen käytetään valmiita simulointiohjelmia.

4.3.2. Monte Carlo -menetelmät

Monte Carlo -menetelmiä käytetään agenttipohjaisissa simulaatioissa agenttien välisen vuorovaikutuksen satunnaisuuden tuottamiseen. Simuloidessa informaation leviämistä internetin sosiaalisissa verkostoissa Monte Carlo -menetelmillä voidaan esimerkiksi generoida painot sosiaalista verkostoa kuvaavan graafin kaarille, jotka kuvaavat todennäköisyyttä, jolla informaatiota saava agentti levittää informaatiota edelleen, tai jolla todennäköisyydellä agentti muuttaa käyttäytymistä agenttien välisen vuorovaikutuksen seurauksena. Markovin ketjua käyttävät Monte Carlo -menetelmät ovat luonnollisempia simuloimaan agenttien käyttäytymistä, koska ne ottavat huomioon aikaisemmat tapahtumat. Näin voisi ajatella myös reaalimaailmassa, jossa sosiaalisten verkostojen sivustojen käyttäjät eivät tee päätöksiä mielivaltaisesti, vaan aiempiin kokemuksiin pohjautuen.

de C. Gatti ja muut [2013] hyödynsivät Monte Carlo -menetelmää simulaatiossaan muuttamaan agentin tilaa erilaisten todennäköisyyksien mukaan. Snijders [2002] käytti myös Monte Carlo -menetelmää laskiessaan, millä todennäköisyydellä graafin agenttien välille muodostuu kaari.

5. Agenttipohjainen mallintaminen internetin sosiaalisissa verkostoissa

Internetin sosiaalisen verkosto simulointi voi kohdistua keinotekoiseen sosiaaliseen verkostoon tai käyttämällä sosiaalisen verkoston sivustojen API-rajapintoja hakemaan aitoja käyttäjätietoja simulaatiomallin. Keinotekoinen verkosto tarkoittaa verkoston mallintamista tyhjästä, kun taas API-rajapintojen kautta tehtävä mallintaminen mallintaa aidot käyttäjätiedot ja suhteet agenttien välille malliin.

de C. Gatti ja muut [2013] simuloivat Barack Obaman twiittien leviämistä Twitterissä vuoden 2012 Yhdysvaltain presidentinvaalien aikaan. de C. Gatti ja muut [2013] käyttivät Twitterin API-rajapintaa aitojen käyttäjätietojen kuten käyttäjän perustietojen, seuraajasuhteiden ja twiittien keräämiseen muodostaakseen aidon rakenteen malliin. API-rajapintojen avulla muodostetun rakenteen voi katsoa olevan internetin sosiaalisten verkostojen simulointia varten mielekkäämpää, koska simulaation nollatilanteen voi olettaa olevan nykyhetki.

de C. Gatti ja muut [2013] käyttivät käyttäjien keräämiseen lumipallokeräystä, jossa ensiksi valitaan mielivaltainen lähtösolmu, josta aloitetaan leveyshaun suorittaminen. Leveyshaku suoritetaan rekursiivisesti niin kauan, kunnes vierailtujen solmujen suhde ylittää halutun keräyssuhteen. de C. Gatti ja muut [2013] toteuttivat lumipallokeräyksen asettamalla maksimietäisyydeksi lähtösolmusta kaksi ja merkitsemällä kaikki Barack Obaman 23 miljoonaa seuraajaa identifioiduiksi, joista 40000 satunnaista seuraajaa merkittiin kuulluiksi, joista edelleen 10000 seuraajaa merkittiin vierailluiksi. Samalla tavalla edettiin niihin seuraajiin, jotka olivat etäisyydellä kaksi Barack Obamasta. Etäisyydeltä kaksi valittiin yhteensä 40000 käyttäjää. Näiden käyttäjien twiitit otettiin mukaan malliin. Twiitit kerättiin viimeisimmän kuukauden ajalta, ja lopulta poimittuna oli yhteensä noin viisi miljoonaa twiittiä. Kerätyn aineiston perusteella toteutettu simulaatio toteutettiin agenttipohjaisena simulaationa, jossa agentti sisälsi yhden Twitterin käyttäjän käyttäytymismallin. Agenttien ympäristönä oli aidosta internetin sosiaalisesta verkostosta saatu graafi G = (A, R), jossa A on graafin sisältämien solmujen joukko ja R kaikkien solmujen väliset riippuvuussuhteet. Koska riippuen sosiaalisesta verkoston sivustosta toiminnot ja vuorovaikutusmetodit vaihtelevat, simulaatiota varten täytyy määritellä toiminnot agenttien käyttäytymismalliin.

de Gatti ja muut [2013] mainitsevat suurimman osan alan tutkimuksesta kohdistuvan synteettisiin verkostoihin, mikä tekee suurimman osan alan tutkimuksista vähemmän realistisia. Simulaatiomalli itsessään on keinotekoisesti luotu ja jotta se vastaisi Twitter-ympäristöä, sen täytyy sisältää suurin piirtein samat toiminnot kuin oikea Twitter. Nämä toiminnot muodostavat agentin vuorovaikutuskeinot simulaatiossa.

6. Verkoston vaikutusvaltaisimmat käyttäjät

6.1. Keskeisyys-käsitteet

Tutkittaessa tiedon leviämistä verkostoa simuloidessa on oleellista tunnistaa ne agentit, joilla on verkostossa eniten vaikutusvaltaa. Verkoston agentin vaikutusvalta saadaan selville tutkimalla yksittäisen agentin keskeisyyttä verkostossa. Agentin keskeisyyttä voi mitata erilaisilla keskeisyysmitoilla.

Astekeskeisyys kuvaa tietyn agentin suorien yhteyksien määrää muihin agentteihin [wikipedia.org]. Suuntaamattomassa graafissa astekeskeisyys on helppo määrittää sisällyttämällä asteeseen kaikki yhteen solmuun liittyvät kaaret. Suunnatussa graafissa astekeskeisyydellä on olemassa eri tulkintoja, sillä kaaret eivät enää ole vain yhdistämässä solmuja, vaan niillä on myös suunta. Luvussa 1 annoin Twitterille määritelmän ”suunnattu sosiaalisen verkoston

sivusto", joten Twitterissä astekeskeisyydelle on asetettava tulkinta. Suunnatut kaaret kuvaavat informaation leviämissuuntaa, joten tulkinta agentista lähtevien kaarien asteelle on agentin seuraajien lukumäärä. Yleistajuisella kielellä ilmaistuna agentista lähtevien kaarien aste kuvailee sitä, kuinka suosittu agentti on. Agenttiin tulevien kaarien aste kuvaa agentin seurattavien lukumäärää, mikä ei tosin tässä tutkielmassa ole erityisen oleellista.

Muita keinoja mitata agentin keskeisyyttä verkostossa on välikeskeisyys, joka kuvaa verkoston kaikkien agenttien välisten lyhimpiä polkuja, jotka kulkevat tarkasteltavan agentin kautta, ja läheisyyskeskeisyys, joka kuvaa tarkasteltavan agentin kaikkiin muihin agentteihin liittyviä lyhyimpiä polkuja [Yang et al. 2015].

6.2. Floyd–Warshallin algoritmi

Väli- ja läheisyyskeskeisyyksien määrittämiseen käytetään suurissa sosiaalisissa verkostoissa Floyd–Warshallin algoritmia, joka tunnistaa kaikkien verkoston agenttien väliset lyhyimmät polut ajassa $O(x^3)$, missä x on agenttien lukumäärä verkostossa.

Floyd–Warshallin algoritmin toimintaperiaatteena on asettaa kaikkien solmujen väliset lyhyimmät etäisyydet kaksiulotteiseen taulukkoon X[n][n], jossa ensiksi jokainen taulukon alkio alustetaan arvolla ääretön. Tämän jälkeen solmun etäisyys itseensä asetetaan nollaksi. Toisin sanoen taulukon alkio X[i][j] alustetaan arvolla nolla, kun i on yhtä suuri kuin j. Tämän jälkeen sopiviin taulukon soluihin tallennetaan olemassaolevien kaarien pituudet. Kaaren, joka yhdistää solmuja indeksiarvoilla 1 ja 2, pituus tallennetaan taulukon alkioon X[1][2] ja niin edelleen. Taulukon alkiot, joiden välille ei ensimmäisessä vaiheessa löydetty polkua jäävät äärettömäksi. Iteraatioiden aikana tutkitaan, onko taulukon alkion X[i][j] arvo suurempi kuin alkioiden X[i][k] ja X[k][j] summa, missä k on iteraatiokierroksen järjestysnumero, $k \geq 0$. Jos alkio X[i][j] on suurempi, alkio X[i][j] saa arvokseen edellämainittujen alkioiden summan. Muussa tapauksessa alkio säilyttää vanhan arvonsa.

Iteraatioita suoritetaan taulukon koon verran, joten iteraatiossa käytettävien muuttujien k, i, j on oltava keskenään yhtä suuret. Algoritmin päättyessä yhdenkään taulukon alkion arvo ei ole ääretön, ellei graafissa ole solmuja, jotka ovat saavuttamattomissa. Toisin sanoen tällaisilla solmuilla ei ole kaaria yhdistämään sitä muihin graafin solmuihin. Algoritmin tuottama lopullinen taulukko sisältää lyhimmät polkujen etäisyydet graafin solmujen välillä.

6.3. Esimerkkitapauksia

Piskor-Ignatowicz ja Zachara [2016] simuloivat vuoden 2013 Association Press -huijausta, jossa Association Pressin Twitter-tili hakkeroitiin ja sitä käytettiin väärää tietoa Yhdysvaltain presidentin virka-asuntoon Valkoiseen taloon kohdistuvasta pommiuhkasta sisältävän twiitin julkaisemiseen. Twiitti levisi Twitterissä ja vaikutti hetkellisesti rajusti pörssikursseihin.

Piskor-Ignatowicz ja Zachara [2016] mallinsivat aidon agenttien välisen rakenteen Twitteristä ja tulivat lopputulokseen, jonka mukaan korkeimpien välikeskeisyyksien agenttien poistaminen simulaatiosta vähensi merkittävästi väärän tiedon leviämistä, ja näin voidaan päätellä välikeskeisyyden olevan yhteydessä ylipäätään siihen, kuinka informaatio leviää verkostossa. Samankaltaiseen tulokseen päätyivät myös de C. Gatti ja muut [2013]. Heidän tuloksensa mukaan Barack Obaman kymmenen vaikutusvaltaisimman seuraajan poistamisella simulaatiosta saavutettiin alkuperäisen viestin huomattavasti vähäisempi leviäminen ja että nämä kymmenen seuraajaa olivat yhdessä vaikutusvaltaisempia sosiaalisessa verkostossa kuin Barack Obama itse.

7. Pohdintaa

Twitterissä on henkilökohtaisen syötteen lisäksi mahdollista tutkia avainsanoihin perustuvia twiittilistoja. Twitterissä informaatio ei siis välity ainoastaan käyttäjältä käyttäjälle seuraamisen kautta, vaan vuorovaikutus on mahdollista sellaisten Twitterin käyttäjien kesken, joiden välillä ei ole minkäänlaista seuraajasuhdetta. Suurin osa aiheeseen liittyvistä tutkimuksista keskittyivät mallintamaan sosiaalista verkostoa Twitterissä pelkästään seuraajasuhteiden kautta eivätkä tarkastele avainsanojen kautta tapahtuvaa twiittien selailua eikä sitä, kuinka seuraajasuhteiden ulkopuolella tapahtuva vuorovaikutus vaikuttaa informaation leviämiseen ja agentin käyttäytymiseen. Kuten Bonabeau [2002] toteavat, jokaisen mallintamisen on perustuttava tarkkaan rajattuun ongelmaan, joten tämä on eri tutkimusten piirissä. Nykyiset tutkimukset onnistuvat kuitenkin mallintamaan ja simuloimaan informaation leviämisen riittävän kattavasti, jotta simulaatio vastaa realistisella tasolla reaalimaailmaa.

de C. Gatti ja muut [2013] väittävät keinotekoisiin verkostoihin tehtävän tutkimuksen olevan epärealistisempaa kuin aitojen internetin sosiaalisten verkostoista saadun datan simulointi, mutta esimerkiksi suurin hyöty Twitterin API-rajapintoja hyödyntävästä mallintamisesta on vain aitoa internetin sosiaalista verkostoa vastaava rakenne. Malliin täytyy joka tapauksessa syöttää agenttien vuorovaikutuskeinot, jonka jälkeen en näe, että olisi merkittävää, onko rakenne aito vai keinotekoinen.

8. Yhteenveto

Tässä kirjallisuuskatsauksessa käsittelin agenttipohjaista mallintamista ja tarkastelin, kuinka agenttipohjaisella mallintamisella ja simuloinnilla voidaan simuloida informaation leviämistä Twitterin kaltaisissa internetin sosiaalisissa verkostoissa.

Sosiaalisen verkoston kaltaisten kompleksisten verkostojen simuloinnilla on kysyntää usealla tieteenalalla. Sillä saavutetaan tietoa sosiaalisten verkostojen sivustojen käyttäjien välisistä riippuvuussuhteista ja käyttäytymisestä niin käyttäjän, kuin verkoston tasolla. Agenttipohjainen mallintaminen ja simulointi tarjoaa muihin vaihtoehtoihin verrattuna tehokkaan työkalun internetin sosiaalisten verkostojen tutkimiseen. Sen etuihin kuuluu luonnollinen esitystapa, nopeus verrattuna muihin analysointikeinoihin ja simulaatiolla tuotetun datan visualisointi. Agenttipohjainen mallintaminen pystyy mallintamaan todellisen maailman sosiaalisia ilmiöitä paremmin kuin vaihtoehtoiset matemaattiset ja tilastotieteelliset mallit. Sosiaalisten verkostojen lisäksi agenttipohjaista mallintamista ja simulointia on sovellettu eri tieteenaloilla, esimerkiksi ennustamaan pörssissä toimivien agenttien toimintaa, ennustamaan epidemioiden leviämistä ja ymmärtämään muinaisten sivilisaatioiden romahtamista.

Floyd–Warshallin algoritmia käytetään vaikutusvaltaisimpien jäsenten havaitsemiseen verkostosta. Algoritmia käyttäen on kullekin verkoston agentille mahdollista määrittää aste, joka kertoo joko agentin kautta kulkevien kaikkien muiden agenttien välisten lyhyimpien polkujen lukumäärän, tai agentin lyhyimmät polut kaikkiin muihin verkoston agentteihin.

Monte Carlo -menetelmillä tuotetaan satunnaisuutta agenttien väliseen vuorovaikutukseen.

Viiteluettelo

J. E. Bahbouhi and N. Moussa. 2015. Agent-based system simulation of electronic commerce: Effect of cut-link on prisoner's dilemma with Small World topology. In: *Proc. of the 15th International Conference on Intelligent Systems Design and Applications (ISDA)*, 493–498.

R. Beheshti. Agent-based modeling: A simpler way to understand complexity. https://www.coursera.org/learn/systems-science-obesity/lecture/pqYse/agent-based-modeling-a-simpler-way-to-understand-complexity. Checked 3.5.2017.

D. Blanco-Moreno, R. Fuentes-Fernández and J. Pavón. 2011. Simulation of Online Social Networks with Krowdix. In: *Proc. of the International Conference on Computational Aspects of Social Networks (CASoN)*, 13–18.

E. Bonabeau. 2002. Agent-based modeling: Methods and techniques for simulating human systems. *PNAS* 99, 3, 7280–7287.

d. m. boyd and N. B. Ellison. 2007. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication* 13, 1, 210–230.

M. A. de C Gatti, A. P. Appel, C. Nogueira dos Santos, C. S. Pinhanez, P. R. Cavalin and S. Barbosa Neto. 2013. A simulation-based approach to analyze the information diffusion in Microblogging Online Social Network. In: *Proc. of the Simulation Conference (WSC), 2013 Winter*, 1685–1696.

itk.fi. http://itk.fi/2012/ohjelma/tori/150. Luettu 6.3.2017.

H. G. Katzgraber. 2011. *Introduction to Monte Carlo Methods*. Department of Physics and Astronomy, Texas A&M University.

J. V. Kujala. 2000. *Markov Chain Monte Carlo -integroinnin nopeuttamisen tekniikoita.* Pro gradu -tutkielma. Tietotekniikan laitos, Jyväskylän yliopisto.

C. M. Macal and M.J North. 2010. Tutorial on agent-based modelling and simulation. *Journal of Simulation* 4, 3, 151–162.

J. A. Obar and S. Wildman. 2015. Social media definition and the governance challenge: An introduction to the special issue. *Telecommunications Policy* 39, 9, 745–750.

K. Ryczko, A. Domurad, N. Buhagiar and I. Tamblyn. 2017. Hashkat: large-scale simulations of online social networks. *Social Network Analysis and Mining*, December 2017, 7:4.

T. A. B. Snijders. 2002. *Markov Chain Monte Carlo Estimation of Exponential Random Graph Models*. Department of Statistics and Measurement Theory, University of Groningen.

Statista.com. Most famous social network sites worldwide as of January 2017, ranked by number of active users (in millions), https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/. Checked 28.3.2017.

Y. Sun, J. Tang, L. Pan and J. Li. 2015. Matrix Based Community Evolution Events Detection in Online Social Networks. In: *Proc. of the IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, 465–470.

Q. Wang and J. E. Taylor. 2013. Energy saving information cascades in online social networks: An agent-based simulation study. In: *Proc. of the 2013 Winter Simulation Conference*, 3042-3050.

Wikipedia.org. https://en.wikipedia.org/wiki/Centrality. Checked 3.5.2017.

S. Y. Yang, A. Liu and S. Y. K. Mo. 2014. Twitter financial community modeling using agent based simulation. In: *Proc. of the 2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, 63–70.

M. Zachara and C. Piskor-Ignatowicz. 2016. Agent-Based Creation and Simulation of Artificial Social Networks and the Analysis of Their Properties. *Computing in Science & Engineering* 18, 4, 34–41.

R. Zeng, Q. Z. Sheng and L. Yao. 2015. A simulation method for social networks. *Social Network Analysis and Mining*, December 2015, 5:6.

Tietoturvallisuuden johtaminen yrityksessä

Petrus Vasenius

Tiivistelmä.
Tutkielmassa selvitetään, miksi yritysten pitäisi ottaa tietoturvallisuus osaksi niiden yleistä johtamistoimintaa ja turvallisuusjohtamista. Tietoa on kartoitettu ja ohjeistusta on laadittu yleisellä tasolla. Motiivina työn tekemiselle on antaa yritykselle tietoa tietoturvallisuuden merkityksestä ja sen tärkeydestä osana yritysturvallisuutta.

Tietoturva jakautuu kahteen kategoriaan, hallinnolliseen ja tekniseen. Teknistä puolta on yleensä helpompi lähestyä ja sen kustannukset on helpompi perustella. Tietoturvaan liittyvien toimien tarkoituksena on tietoaineistojen, tietojärjestelmien ja palveluiden asianmukainen suojaus siten, että niiden luottamuksellisuuteen, eheyteen ja käytettävyyteen liittyvät riskit otetaan huomioon.

Yrityksille suunniteltujen tietoturvatoimien ja sertifiointien perusta löytyy kansainvälisistä ohjeistuksista. Esimerkiksi ISO 27000 -perheen standardit määrittelevät pitkälti yritykselle kuuluvat tietoturvaohjeet ja käytännöt. Edellä mainittujen lisäksi löytyy muitakin ohjeistuksia, mutta niitä ei ole validoitu, eivätkä ne siten ole hyvä perusta tietoturvatoiminnalle yrityksessä. Suomenkielisistä aineistoista Valtiohallinnon tieto- ja kyberturvallisuuden ohjausryhmän laatimat VAHTI-ohjeistukset ovat päteviä myös yksityisen sektorin käyttöön.

Avainsanat ja -sanonnat: Tietoturvallisuus, yritysturvallisuus, turvallisuusjohtaminen, standardit.

1. Johdanto

Tietoturvallisuus käsitteenä on tuttu lähes jokaiselle tietokoneen ja Internetin käyttäjälle. Se on nykyaikaisissa yrityksissä laaja käsite, ja se sisältää laaditut tietoturvasäännöt, ohjelmistojen valinnan ja tietoturvatietoisuuden. [Kaukver 2012].

Tietoturva jakautuu selkeästi kahteen kategoriaan, hallinnolliseen ja tekniseen. Teknistä puolta on yleensä helpompi lähestyä ja sen kustannukset on helpompi perustella. Hallinnollista tietoturvaa on sen sijaan vaikeampi lähestyä, koska organisaatioissa ei ole siihen tarpeeksi asiantuntemusta, resursseja on liian vähän, ja johto ei yleensä ymmärrä tietoriskejä [Sulosaari 2005]. Tietoturvallisuus on merkittävä osa nykyaikaista yritysturvallisuutta ja vaatii yhä enemmän huomiota yrityksen johdolta.

Tietoturvallisuuden johtaminen osana yrityksen turvallisuusjohtamista ei ole pelkästään yksiselitteisten riskien ennaltaehkäisyä, vaan kokonaisuuden hallintaa, jonka avulla varmistetaan yrityksen strategisten tavoitteiden saavuttaminen. Turvallisuusjohtaminen on organisoitua ja järjestelmällistä johtamista, jolla pyritään ennaltaehkäisemään ihmisiä, ympäristöä, omaisuutta, tietoa ja mainetta vahingoittavia tapahtumia. Turvallisuusjohtamisen on oltava jatkuvaa toimintaa ja kiinteä osa organisaation normaalia johtamisprosessia [Leppänen 2006].

Tämän tutkielman tavoitteena on antaa ohjeistusta yrityksien johdolle ottamaan tietoturvallisuus osaksi yrityksen yleistä turvallisuusjohtamista. Työn tavoitteena on antaa yrityksille tietoa tietoturvallisuuden merkityksestä ja sen tärkeydestä osana yritysturvallisuutta. Luvussa 2 avaan tietoturvan käsitettä ja siihen kuuluvia osa-alueita. Luvussa 3 kirjoitan yritysturvallisuudesta ja miten tietoturvallisuus on osa sitä. Lisäksi käsittelen tietoturvallisuuden johtamista ja siihen liittyviä standardeita. Neljännessä luvussa kerron, mitä on turvallisuusjohtaminen ja mitä lainsäädännöllisiä velvoitteita sillä on. Luku 5 on yhteenveto.

2. Tietoturva

Tietoturvallisuuden käsitteelle ei löydy yksiselitteistä määritelmää. Se on useiden pienempien osa-alueiden kokonaisuus, jonka yhteisvaikutuksella tietoturvallisuus saavutetaan [Sulosaari 2004]. Tietoturvaan ja kyberturvallisuuteen liittyvät kysymykset ovat nykyaikana todella tärkeitä. Ei kuitenkaan pelkästään riitä, että suojellaan kansalaisten, yritysten ja julkishallinnon tietoja [Bishop 2003].

Perinteisesti tietoturvan katsotaan koostuvan kolmesta tiedon perustekijästä: luottamuksellisuudesta, eheydestä ja käytettävyydestä. Luottamuksellisuudella tarkoitetaan tässä yhteydessä sitä, että tieto on vain niiden tahojen käytettävissä, joilla on oikeus tietoon. Eheys on tiedon ja tiedon käsittelymenetelmien oikeellisuuden ja täydellisyyden varmistamista suojautumalla luvattomilta muokkauksilta. Käytettävyydellä tai saatavuudella tarkoitetaan sitä, että tieto ja tietoon liittyvät resurssit ovat auktorisoitujen käyttäjien käytettävissä tarvittaessa [ISO 27002].

Valtiovarainministeriön laatiman VAHTI-tietoturvaohjeiden mukaan tietoturvatoimien tarkoituksena on varmistaa tietoaineistojen, tietojärjestelmien ja palveluiden asianmukainen suojaus siten, että niiden luottamuksellisuuteen,

eheyteen ja käytettävyyteen liittyvät riskit otetaan huomioon. Tietojen, järjestelmien ja palveluiden on oltava luotettavia, oikeita ja ajantasaisia. Ne eivät saa tuhoutua, paljastua tai muuttua hallitsemattomasti asiattoman toiminnan, haittaohjelmien, laitteisto- tai ohjelmistovikojen tai muiden vahinkojen, tapahtumien tai häiriötilanteiden vuoksi [VAHTI 3/2014].

Tietoturvallisuus jaetaan yhdeksään eri osa-alueeseen [Miettinen 2002]:

- Hallinnollinen ja organisatorinen tietoturvallisuus
- Henkilöstöturvallisuus
- Fyysinen turvallisuus
- Tietoliikenneturvallisuus
- Laitteisto- ja ohjelmistoturvallisuus
- Tietoaineistoturvallisuus
- Käyttöturvallisuus
- Yksityisyyden suojaaminen
- Immateriaalioikeuksien suojaaminen.

2.1 Hallinnollinen ja organisatorinen tietoturvallisuus

Hallinnollisella ja organisatorisella tietoturvallisuudella tarkoitetaan yhtä tietoturvallisuuden johtamistoimintoa, ja se on koko organisaation tietoturvatoiminnan lähtökohta. Se muodostuu kokonaisuudessaan johdon hyväksymistä tietoturvaperiaatteista, vastuunjaosta, tarkoitukseen varatuista resursseista sekä riskien arvioinnista. Hallinnollisen tietoturvallisuuden tarkoituksena on luoda organisaatioon tietoturvalliset toimintatavat. Kuvassa 1 on esitelty tietoturvapolitiikan toimintamalli. Sen pohjalta luodut henkilöstön koulutusjärjestelyt sekä ohjeistus-, valvonta- ja tarkastusmenettelyt ovat välttämättömiä tietoturvallisuuden kehittämiseksi ja ylläpitämiseksi [Väistö 2005].

Kuva 1. Tietoturvapolitiikan toimintamalli. [Väistö 2005]

Hallinnollisessa tietoturvallisuudessa on oleellista, että käyttäjät tietävät ja ymmärtävät ne periaatteet, joille organisaation tietoturvallisuus rakentuu. Tätä varten organisaation johdon tulee julkaista organisaation tietoturvapolitiikka. Politiikka jaetaan koko henkilökunnalle. Tietoturvapolitiikan tueksi tulee suunnitella organisaation ohjekokonaisuus ja määritellä vaadittu tietoturvakuvausten taso. Lisäksi laaditaan tietoturvasuunnitelmat, jotka osoittavat organisaatiolle elintärkeät tietojärjestelmät, niiden toipumistoimet sekä vaatimukset poikkeusolojen valmiudelle [Väistö 2005].

2.2 Henkilöstöturvallisuus

Henkilöstöturvallisuudella tarkoitetaan Valtiovarainministeriön laatiman VAHTI-tietoturvaohjeiden mukaan tietoturvassa henkilöstöstä johtuvaa riskien hallintaa. Henkilöstöturvallisuuden perustana on osaava ja sitoutunut henkilöstö, jolle tietoturvavastuut ja -tehtävät on selkeästi kuvattu toimenkuvissa. Lisäksi tarvitaan riittävällä tasolla määriteltynä olevat henkilöstöhallinnon prosessit sekä muut prosessit, joissa kuvataan työtehtävät niin tarkasti, että avainhenkilöriskien syntyminen vältetään [VAHTI 2008].

Keskeisiä asioita ovat työhönottoon, toimenkuvan olennaisiin muutoksiin ja palvelussuhteen loppumiseen liittyvät prosessit ja niistä on tarpeen olla kaikilla osallisilla käytössään sovittu toimintamalli. Tehtävien vaativuudesta tai luottamuksellisuudesta riippuen rekrytoitavan henkilön tausta, sopivuus ja osaaminen on selvitettävät ennen työhönottoa [VAHTI 2008].

Avainhenkilöriskien hallinnassa tunnistetaan toiminnan kannalta avainhenkilöt sekä varmistetaan heidän sopivuutensa organisaation palvelukseen. Suunnittelussa varaudutaan lomiin, poissaoloihin, työnkiertoon ja väliaikaisjärjestelyihin riittävän hyvin sekä lisäksi valmennetaan henkilöstö poikkeusoloihin. Vaaralliset työyhdistelmät tulee tunnistaa ja poistaa, jotta organisaation toiminnan suojaksi rakennettuja menetelmiä ei voida havaitsematta kiertää. VAP-varausten (vapautettu asepalveluksesta sodan aikana) tekeminen on osa riittävien henkilöresurssien varmistamista osana poikkeusoloihin liittyvää valmiussuunnittelua [VAHTI 2008].

2.3 Fyysinen tietoturvallisuus

Fyysinen tietoturvallisuus on tietoaineistojen ja -välineiden sekä niitä ympäröivien tilojen rakenteellista ja toiminnallista suojaamista. Fyysisen tietoturvallisuuden kohteena on turvallisuusluokiteltu tieto kaikissa olomuodoissaan sekä

sen käsittelyprosessit. Leppäsen [2006] mukaan fyysisen tietoturvallisuuden voi jakaa riskien kautta seuraaviin osa-alueisiin:

- Paloriskit (tulipalot ja savuvahingot)
- Vesi- ja kosteusvahingot (vesijohto- ja sprinklerijärjestelmä, kosteusvauriot)
- Sähköriskit (sähköpulssit, sähkökatkot, EMP)
- Ilmastointiriskit (pöly, tuuletus ja lämpötilan kohoaminen)
- Omaisuusrikosriskit (varkaudet, kavallukset, tuhotyöt jne.)
- Tiedustelu- ja vakoiluriskit (salakatselu- ja kuuntelu, luvaton kopiointi, laiton tiedonhankinta).

Fyysisen tietoturvallisuuden hallinnassa korostuu riskien arviointi. Käytännössä fyysinen tietoturvallisuus voidaan ja tulee liittää siis kiinteästi rakenteelliseen turvallisuuteen [Leppänen 2006].

2.4 Tietoliikenneturvallisuus

Tietoliikenneturvallisuudella tarkoitetaan häiriötöntä viestintää, tiedonsiirtoyhteyksien käytettävyyttä, tiedonsiirron suojausta ja salausta, käyttäjien tunnistusta ja verkon varmistamista. Tietoliikenneturvallisuustyön tulos on turvatut tiedonsiirtoyhteydet. Työ kattaa tietoliikenneverkon ja sen laitteiden kokoonpanon, ylläpidon ja muutosten hallinnan [Tietoturva- ja tietosuojapolitiikka 2012].

Tietoliikenneturvallisuus muodostuu kokonaisuudesta, jossa tieto kulkee luotettavasti ja luottamuksellisesti siirtotiellä ja on oikean ja tunnistetun vastaanottajan saatavilla muuttumattomana. Lisäksi sekä lähettäjä että vastaanottaja ovat tunnistettavissa ja todennettavissa. Myös tapahtuman kiistämättömyys on oltava todennettavissa. Leppäsen [2006] mukaan tietoliikenneturvallisuus muodostuu seuraavien osien hallinnasta:

- Lähetysympäristö (riskeinä mm. salakuuntelu ja -katselu)
- Päätelaite (riskeinä tietojen kopiointi, päätelaitteen varkaudet yms.)
- Sisäverkko (riskeinä tietomurrot, sähkökatkot, tietoliikennehäiriöt, salakuuntelu yms.)
- Suojattu yritysverkko (riskeinä palvelinongelmat, reitittimet, tietomurrot yms.)
- Julkinen internet-verkko (riskeinä tietoliikenteen kuuntelu, tietojen kaappaus jne.)
- Vastaanottoympäristö (riskeinä väärät vastaanottajat, salakuuntelu- tai katselu, tietomurto jne.).

2.5 Laitteisto- ja ohjelmistoturvallisuus

Tietovälineet ja ohjelmistot muodostavat oman suojattavan kokonaisuuden. Tietovälinelaitteiston turvallisuus voidaan jakaa seuraaviin osiin: suunnittelu, rakennus tai valmistus, kokoonpano, asennus, käyttöönotto, käytöstä poisto, kunnossapito ja laadun varmistus. Laitteistot on suunniteltava ja rakennettava siten, että ne vastaavat organisaation käyttötarpeita ja -vaatimuksia. Tietoteknisiä laitteistoja ja -ohjelmistoja tulee käsitellä tuotantovälineinä. Niiden hankinnan, huollon ja ylläpidon tulisi olla yhtä huolellista kuin minkä tahansa muunkin laitteen, jonka avulla tuotetaan prosessissa tarvittavia tuloksia [Miettinen 2002].

Laitteistoturvallisuuden ylläpitämisessä on tärkeää laitteistojen käyttötarpeisiin ja vaatimuksiin perustuva hankintapolitiikka. Laitteistojen toiminnan yhdenmukaisuus helpottaa teknisen tuen toimintaa laitteistojen päivityksissä sekä uudelleenhankinnoissa [Leppänen 2006].

Ohjelmistoturvallisuus muodostuu käytössä olevista käyttöjärjestelmistä sekä sovellusohjelmistoista. On kuitenkin hyvä muistaa, ettei mikään ohjelmisto ole koskaan täysin turvallinen. Kaikkien ohjelmistojen tietoturvallisuudessa on puutteita ja aukkoja. Ohjelmistoissa olevat selkeät virheet vaikuttavat niiden toimivuuteen ja vakauteen. Valmisohjelmistot tulevat usein markkinoille keskeneräisinä ja sisältävät paljon virheitä. Näitä pyritään jälkikäteen korjaamaan päivityksillä. Merkittävintä on kuitenkin ohjelmistoissa olevien tietoturva-aukkojen olemassaolo. Tietohallinnon keskittyessä etsimään ja suojaamaan satoja puutteita riittää krakkerille yhden tietoturva-aukon olemassaolo. [Leppänen 2006] Laitteisto- ja ohjelmistokokonaisuuden tulee muodostua seuraavista asioista [Miettinen 2002]:

- Tietoturvaohjeistus, säännöt ja koulutus
- Laitteistokirjanpito ja muu dokumentaatio
- Laitteiden sijoittelu
- Tiloihin pääsy
- Laitekaappien lukitus ja valvonta
- Laitteiden hankinta ja asennukset
- Varajärjestelyt
- Huoltosopimukset ja niiden yhdenmukaisuus tietojärjestelmän tärkeysluokissa
- Ohjelmistoasennukset ja päivitykset
- Ohjelmistojen koodin laadun varmistus ja tuotannossa käytetyt välineet
- Tietoturvaluontien suoritus ennen hankintaa
- Lisenssien hallinta ja asennusohjeet.

2.6 Tietoaineistoturvallisuus

Tietoaineistoturvallisuus kattaa tietoaineistoon liittyvän turvallisuuden. Tietoaineisto voidaan luokitella, jotta sillä voidaan rajata tietojen kokoamiseen, käyttöön, säilyttämiseen, siirtämiseen ja tuhoamiseen liittyviä riskejä. Ensisijainen luokittelukriteeri on se, miten julki tuleva tieto vaikuttaa yrityksen tai organisaation toimintaan [Leppänen 2006].

Julkitulon voi jakaa yleiseen, esimerkiksi median tai viranomaisen rekisterin kautta tapahtuvaa, tai rajattua, jolloin tieto siirtyy vain sellaisille henkilöille, joilla voi olla vaikutusta yrityksen toimintaan. [Leppänen 2006]. Tietoaineiston turvallisuudesta tulee huolehtia koko sen elinkaaren ajan. Elinkaari muodostuu tiedon syntymisestä tai keräämisestä, tiedon käsittelystä, tiedon siirtämisestä, tiedon varastoinnista sekä tiedon hävittämisestä [Leppänen 2006].

2.7 Käyttöturvallisuus

Käyttöturvallisuus kattaa lähes kaikki edellä mainitut osa-alueet. Leppänen [2006] toteaa, että koska ihminen on usein se heikoin lenkki kaikista turvallisuustoimenpiteistä huolimatta, tulee käyttöturvallisuudella varmistaa, että koko henkilöstö hallitsee riittävät tietoturvallisuuden ylläpitämiseen liittyvät toimenpiteet. Käyttöturvallisuuden kokonaisuus koostuu seuraavista osista [Leppänen 2006]:

- Työaseman käyttö
- Tietovälineiden käyttö (tietokoneet, faksi, puhelin, mobiililaitteet yms.)
- Tietoaineiston käyttö
- Liitetietojen käyttö
- Sähköpostin ja internetin käyttö
- Virustorjunta
- Lähiverkko
- Käyttöoikeuksien ja salasanojen hallinta
- Varmuuskopiointi
- Tilojen lukitus ja kulunvalvonta.

2.8 Yksityisyyden suojaaminen

Yksityisyyden suoja on tietoturvallisuuden osa-alue, jossa tarkastellaan erityisesti yrityksen toimintaan liittyvien henkilöiden henkilötietojen suojaamista.

Yksityisyyden suojaamisesta käytetään usein myös nimeä tietosuoja. Yksityisyyden suojaa tarkastellaan tässä lyhyesti omana kokonaisuutena, koska siihen liittyviä asioita säädellään melko tarkasti lainsäädännössä ja tämä asettaa omia erityisvaatimuksia yrityksille niiden käsittelemien henkilötietojen suojauksen suunnittelulle, toteutukselle ja ylläpidolle [Miettinen 2002].

Säädöspohja perustuu tällä hetkellä voimassaolevaan henkilötietolakiin, lakiin yksityisyyden suojasta televiestinnässä ja lakiin teletoiminnan tietoturvasta. Henkilötietolaki on yksityisyyden suojaamisen taustalla oleva yleislaki ja se korvaa aikaisemmin käytössä olleen henkilörekisterilain. Kaksi muuta edellä mainittua lakia niihin liittyvine alempitasoisine säädöksineen tukevat henkilötietolakia [Miettinen 2002].

Tyypillisiä yksityisyyden suojaan liittyviä tietoja ovat esimerkiksi henkilön terveydentilaa, poliittista ja uskonnollista vakaumusta, seksuaalista suuntautumista ja käyttäytymistä ja rikollista menneisyyttä koskevat tiedot. Henkilötunnuksen käsittely kuuluu myös tähän aihepiiriin. Näiden tai muiden vastaavien arkaluontoisten tietojen joutuminen ilman henkilön lupaa ulkopuolisten haltuun voi aiheuttaa yksittäiselle ihmiselle vakavia seuraamuksia. Siksi yrityksen on kiinnitettävä erityistä huomiota näiden tietojen suojaamiseen, jotta tietoja ei voitaisi käyttää henkilön etujen vastaisesti [Miettinen 2002].

2.9 Immateriaalioikeuksien suojaaminen

IPR (Intellectual Property Rights) eli immateriaalioikeuksien suojaaminen tarkoittaa sitä, että yritys tai yksityishenkilö hyödyntää lainsäädännön mahdollisuuksia kohteen omistus- ja hallintaoikeuden suojaamisessa muiden menetelmien lisäksi. Immateriaalioikeuksien suojaamisella pyritään estämään esimerkiksi kohteen tai sen osan luvaton kopiointi tai jäljentäminen. Immateriaalioikeuksien suojaamisella yritys voi pyrkiä erottumaan kilpailijoistaan käyttämällä toiminnassaan juuri sille sopivaa ilmiasua. Sen lisäksi se voi vaatia suojaamiensa tuotteiden käytöstä korvauksia itselleen, jos toinen yritys haluaa käyttää suojattua kohdetta omassa toiminnassaan [Miettinen 2002].

Immateriaalioikeuksien suojaamisen peruskäsitteitä ovat esimerkiksi patentti, hyödyllisyysmalli, mallisuoja, tuotemerkki, tekijänoikeus ja liikesalaisuus. Yrityksen toiminimiin, aputoiminimiin ja Internet domain -nimiin liittyy usein myös suojaamista vaativia immateriaalioikeuksia. Immateriaalioikeuksien suojaamisen on oltava kiinteä osa yrityksen toimintaa ja erityisesti tuotekehitystä, jotta suojaaminen voidaan suorittaa riittävän ajoissa ennen suojattavan kohteen julkistamista. [Miettinen 2002].

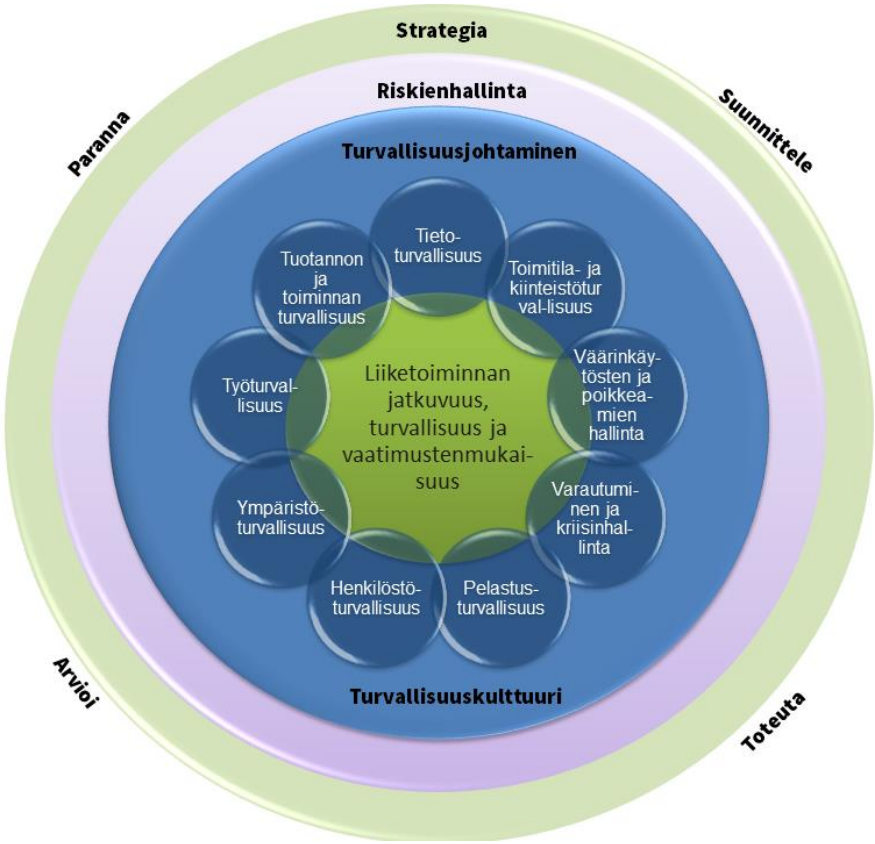
Nykyaikaiseen liiketoimintaan liittyy useimmilla toimialoilla nykyisin Internet. Tämä on tuonut mukanaan yrityksille myös immateriaalioikeuksien suojaamiseen liittyviä asioita, koska yritysten käyttämät niin sanotut Internet domain -nimet joudutaan anomaan kussakin maassa virallisesti. Tämä johtuu siitä, että samaa nimeä ei voida käyttää kuin kerran, jotta tietoliikenteen reititys toimisi Internetissä oikein. Suomessa nimien myöntäjänä toimii Viestintävirasto. Internet domain -nimistä on tullut nykyisin myös kauppatavaraa, kun jotkut yritykset ja yksittäiset henkilöt ovat rekisteröineet haltuunsa suuren joukon suuryritysten Internet domain -nimiä [Miettinen 2002].

3. Yritysturvallisuus

Miettinen [2002] mukaan yritysturvallisuus tarkoittaa turvallisuuden eri osa-alueiden kokonaisvaltaista hallintaa yrityksen toiminnassa. Sen avulla yritys pyrkii varmistamaan liiketoimintansa häiriöttömän jatkuvuuden suojaamalla henkilöstöä, asiakkaita, muita mahdollisia sidosryhmiä, tietoja, omaisuutta ja toimintaympäristöä vahingoilta, väärinkäytöiltä ja rikolliselta toiminnalta.

Yritysturvallisuus on kiinteä osa yrityksen toimintaa ja tukee omalta osaltaan yrityksen liiketoiminnan tulostavoitteita, koska yritysturvallisuus on moniulotteinen asia, sitä pitää tarkastella kokonaiskuvan hahmottamiseksi useista eri näkökulmista
(ks. kuva 2). Näin saadaan muodostettua todellinen kuva siitä, mitä yritysturvallisuus tarkoittaa ja miksi sitä ylläpidetään ja kehitetään. Miettinen [2002] esittelee kolme näkökulmaa:

- **Liiketoiminnallinen tarkastelukulma** edustaa yrityksen ylimmän johdon näkökulmaa yritysturvallisuuteen. Tämä näkökulma antaa vastauksen siihen, miksi yritysturvallisuus on tärkeää juuri minun yritykselleni.
- **Prosessitarkastelunäkökulma** edustaa yrityksen toimintaprosessien edustajien näkökulmaa yritysturvallisuuteen. Tämän tarkastelunäkökulman kautta yritys löytää yrityksen suojattavat kohteet. Suojattavat kohteet voidaan karkeasti jakaa prosesseihin ja resursseihin.
- **Tekninen tarkastelunäkökulma** edustaa perinteistä näkökulmaa yritysturvallisuuden kehittämisessä. Tämän tarkastelunäkökulman kautta yritys pyrkii hahmottamaan, mitä teknisiä ja ei-teknisiä suojuksia sen olisi käytettävä, jotta kaikki yritysturvallisuuden osa-alueet saataisiin asianmukaisesti suojattua.



Kuva 2. Yritysturvallisuuden osa-alueet [Susi 2015].

3.1 Tietoturvallisuus osana yritysturvallisuutta

Tietoturvallisuus on yksi yritysturvallisuuden osa-alueista, ja sen merkitys on keskeinen yrityksen liiketoiminnallisen jatkuvuuden ja kilpailukyvyn turvaamisessa [Koivula 2013]. Jokaisesta yrityksestä löytyy taloudellisesti arvokasta tietoa, jonka menetys ulkopuoliselle aiheuttaa vahinkoa [Koivula 2013].

Elinkeinoelämän keskusliiton [Yritysturvallisuus 2016] mukaan tietoturvallisuudesta on tullut yhä keskeisempi osa organisaation turvallisuutta. Perinteisessä merkityksessä kyse on organisaation tietojen luottamuksellisuuden, käytettävyyden ja eheyden takaamisesta. Teknologinen kehitys on nopeaa, joten ajan tasalla pysyminen edellyttää korostuneesti menetelmien jatkuvaa seuraamista ja toimenpiteiden kehittämistä. Täydellistä turvallisuutta on mahdotonta saavuttaa, joten tietoturvallisuudessa kannattaa panostaa toiminnan jatkuvuuden varmistamiseen.

3.2 Tietoturvallisuuden johtaminen

Kansainväliset tietoturvaohjeet ovat perusta yrityksen tietoturvallisuuden johtamiselle ja sen tietoturvatoiminnan sertifioimiselle. Esimerkiksi ISO 27000 -per-

heeseen (ks. taulukko 1) kuuluvat kansainväliset standardit ISO 27000 – 27011 määrittelevät yritykselle tärkeimmät tietoturvakäytännöt ja ohjeet.

Muita erilaisia ohjeita ovat muun muassa TCSEC, CobiT, GAISP ja SSE-CMM (ks. taulukko 2). Siponen ja Willison [2009] muistuttavat kuitenkin, että kyseiset ohjeet eivät ole yleisiä ohjeita kaikille yrityksille tietoturvallisuuden johtamiseen. Sen sijaan ne auttavat yrityksiä kohti parempaa tietoturvallisuutta tai yrityksen tietyn tietoturvasertifikaatin saamiseksi.

Standardi	**Standardin aihe**
ISO/IEC 27000:2009	Tietoturvallisuuden hallintajärjestelmät. Yleiskatsaus ja sanasto.
ISO/IEC 27001:2005	Tietoturvallisuuden hallintajärjestelmät. Vaatimukset.
ISO/IEC 27002:2005	Tietoturvallisuuden hallintaa koskeva menettelyohje.
ISO/IEC 27003:2010	Tietoturvallisuuden hallintajärjestelmän toteuttamisohjeita.
ISO/IEC 27004:2009	Tietoturvallisuuden hallinta. Mittaaminen.
ISO/IEC 27005:2008	Tietoturvariskien hallinta.
ISO/IEC 27006:2007	Tietoturvallisuuden hallintajärjestelmien auditointi- ja sertifiointielinten vaatimukset
ISO/IEC 27007	Tietoturvallisuuden hallintajärjestelmien auditointiohjeet.
ISO/IEC 27011	Standardiin ISO/IEC 27002 perustuvat tietoturvallisuuden hallintaohjeet tietoliikenneorganisaatioille.

Taulukko 1. ISO 27000 -perheen standardit.

Ohje	**Ohjeen aihe**
TCSEC	Yhdysvaltain puolustusministeriön tietojärjestelmien turvallisuuden arviointiohjeet.
CobiT	Tietoturvallisuuden yleiset kontrolliohjeet.
GAISP	Kansainvälisesti hyväksytyt informaatioturvallisuusohjeet.
SSE-CMM	Tietojärjestelmien turvallisuuden suunnittelun kypsyyttä mittaavat ohjeet.

Taulukko 2. Muut tietoturvaohjeistukset.

Siponen ja Willison [2009] toteavat tietoturvallisuuden johtamisen nykyisten ohjeiden sisältävän kaksi pääasiallista ongelmaa. Ensimmäinen ongelma on, että ohjeet on laadittu yleisellä tasolla. Ongelma on, että yritykset tarvitsevat juuri heidän toiminnalle ja toimintaympäristölle suunnitellut ohjeet. Toinen ongelma on, että ohjeita ei ole validoitu, vaan ne laaditaan yleisen toimintatavan virheitä seuraamalla. Se on epävarma perusta standardille. Vaikka negatiivinen lähtökohta voi olla väärä tapa kehittymiselle, von Solms ja von Solms

[2004] ovat luetelleet kymmenen virhettä, joita yrityksen tietoturvan johtamisessa voidaan tehdä. Ne ovat nähtävillä taulukossa 3.

1. Ei ymmärretä, että tietoturva on yrityksen johdon vastuulla.
2. Ei ymmärretä, että tietoturva on liiketoiminnallinen kysymys eikä tekninen ongelma.
3. Ei ymmärretä, että tietoturvan johtaminen on monimutkainen asia. Yhtä suoraa ratkaisua kaikkiin sen ongelmiin ei ole olemassa.
4. Ei ymmärretä sitä, että tietoturvasuunnitelman tulee perustua jo ennalta tunnistettuihin riskeihin.
5. Ei ymmärretä hyvien kansainvälisten toimintatapojen tärkeyttä tietoturvan johtamisessa.
6. Ei ymmärretä, että yrityksen tietoturvapolitiikan olemassaolo on ehdoton edellytys.
7. Ei ymmärretä, että tietoturvan noudattamisen täytäntöönpano ja valvonta ovat ehdottoman olennaisia yritykselle.
8. Ei ymmärretä, että asianmukainen tietoturvan hallintorakenne on välttämätön yritykselle.
9. Ei ymmärretä tietoturvan keskeistä merkitystä käyttäjille.
10. Ei valtuuteta tietoturvasta vastuussa olevien esimiesten infrastruktuurin, työkalujen ja tukevien mekanismien hoitamaan kunnolla vastuunsa.

Taulukko 3. Kymmenen virhettä yrityksen tietoturvallisuuden johtamisessa [Von Solms ja Von Solms 2004].

4. Turvallisuusjohtaminen

Turvallisuusjohtamiselle ei ole olemassa yhtä yhtenäistä ja yleisesti hyväksyttyä määritelmää. Yleensä turvallisuusjohtaminen on kokonaisvaltaista, sekä lakisääteisen että omaehtoisen turvallisuuden hallintaa, jossa yhdistyy niin menetelmien ja toimintatapojen kuin ihmisten johtaminen. Se sisältää ajatuksen jatkuvasta turvallisuuden ja terveyden edistämisestä työpaikalla. Turvallisuusjohtaminen pitää sisällään jatkuvan suunnittelun, toiminnan ja seurannan [Aluehallintovirasto 2010].

Turvallisuudella tarkoitetaan kokonaisturvallisuutta, joka sisältää sekä perinteisen turvallisuuden (security) että pehmeän turvallisuuden (safety). Termi security sisältää esimerkiksi liiketilaturvallisuuden ja tietoturvallisuuden. Termi safety liittyy sen sijaan muun muassa työturvallisuuteen ja pelastustoi-

mintaan. Kokonaisturvallisuus yrityksessä voidaan jakaa sekä strategiseen että operatiiviseen tasoon. Strateginen taso sisältää kansainvälisen turvallisuuden ja kansallisen lainsäädännön vaatimukset sekä liiketoiminnan turvallisuudelle asettamien vaatimusten näkökulman.

Operatiivinen taso sisältää puolestaan yrityksen sisäisen ja sidosryhmien välisen yhteistyön sekä kokonaisturvallisuuden hoitamisen [Mäkinen 2007].

Tavoitteena omaehtoisessa tai lakisääteisessä turvallisuusjohtamisessa on turvallisuuden ylläpitäminen ja kehittäminen. Lisäksi sen tarkoituksena on parantaa yrityksen tuottavuutta sekä tukea kilpailukykyä. Turvallisuusjohtamisen tulisi olla osa yrityksen päivittäistä johtamistoimintaa [Piisku ja Saari 2007].

Hyvälle turvallisuusjohtamiselle tulee olla olemassa selkeät lähtökohdat: turvallisuuspolitiikan luominen, toimintavelvoitteiden ja -valtuuksien määrittäminen, riskien arviointi, mittaaminen, seuranta ja dokumentointi, osaamisen varmistaminen sekä tiedottaminen. Turvallisuusjohtaminen vaatii myös toimivan palautejärjestelmän, jonka avulla työpaikka pystyy järjestelmällisesti varmistamaan omien käytäntöjensä jatkuvan parantamisen. Työnantajalla tulee aina olla turvallisuuspolitiikka tai -periaatteet, jotka määrittelevät yleisen turvallisuuden päämäärät. Politiikassa ilmenee johdon kannanotto turvallisuustyön merkityksestä. Lisäksi henkilöstön yhteistyön toimintaperiaatteet ja -tavat tulee olla määritelty [Aluehallintovirasto 2010].

Turvallisuusjohtamisen keskeiset elementit organisoinnin osalta ovat toimintajärjestelmien, toimintavastuiden ja -velvollisuuksien määrittäminen sekä riittävien resurssien varaaminen tavoitteiden toteuttamiseksi. Riskien ja toiminnan arvioinnin kattava nykytilanteen kartoitus antaa perustan turvallisuustyölle. Nykytilanteen selvitykseen ja riskien arviointiin on valittavissa erilaisia työkaluja. Tehtyjen toimenpiteiden toteutumista pitää seurata ja turvallisuuden arvioimiseksi pitää valita sopivia mittareita. Osaaminen, oikeat asenteet ja motivaatio tarvitaan myös turvallisuuden saavuttamiseksi ja ylläpitämiseksi. Johtamisen tueksi tarvitaan monipuolista tiedottamista näistä asioista [Häikiö 2009]. Taulukossa 4 on nähtävillä turvallisuusjohtamisen kannalta keskeisimmät tekijät.

Turvallisuusjohtaminen	
Turvallisuuspolitiikka	Sisältää päämäärät Näkyy johdon sitoutuminen Näkyy henkilöstön merkitys turvallisuuden toteuttamisessa
Turvallisuusjohtamisen organisointi	Järjestelmällisten toimintatapojen luominen Kokonaisvastuiden ja velvollisuuksien määrittäminen Linjaesimiesten resurssien varmistaminen
Käytännön toiminta	Riskien arviointi Osaamisen varmistaminen Toimenpiteiden toteutus Tiedonkulun varmistaminen Mittaaminen ja seuranta

Taulukko 4. Turvallisuusjohtamisen keskeiset tekijät [Aluehallintovirasto 2010].

4.1 Turvallisuusjohtaminen ja lainsäädäntö Suomessa

Turvallisuusjohtamisessa on keskeistä myös turvallisuuteen liittyvä lainsäädäntö. Voimassa oleva sekä valmisteilla oleva lainsäädäntö tulisi huomioida yritysten käytännön turvallisuustoiminnassa. Toisaalta yritysten toimintaedellytysten turvaamiseksi tehtävän lakisääteisen turvallisuustoiminnan tulisi olla vain niiden minimitaso. Vasta omaehtoisella ja tavoitteellisella turvallisuustoiminnalla voidaan saada lisäarvoa yrityksen liiketoiminnalle [Piisku ja Saari 2007].

Vaikka suomalaisessa lainsäädännössä ei ole suoranaisesti säädetty turvallisuusjohtamisesta, on joissakin laeissa kuitenkin turvallisuusjohtamiseen liittyviä elementtejä. Esimerkiksi laissa vaarallisten kemikaalien ja räjähteiden käsittelyn turvallisuudesta [390/2005] on säädetty turvallisuusjohtamisjärjestelmän laatimisvelvollisuudesta [Piisku ja Saari 2007].

Laissa esitetyt työnantajan velvollisuudet kuitenkin perustuvat turvallisuusjohtamisen ajatukselle ja turvallisuusjohtamisen peruselementit ovat asioina mukana siinä. Laissa asetettujen velvoitteiden täysmääräinen toteuttaminen edellyttää systemaattista ja pitkäjänteistä työolosuhteiden seurantaa ja toimintatapaa, jolla varmistetaan työntekijöiden turvallisuus ja terveys. Työnantaja on velvollinen selvittämään ja kehittämään työterveyttä ja -turvallisuutta yhteistyössä henkilöstön kanssa [Aluehallintovirasto 2010].

Työturvallisuuslaki [2002/738] säätelee turvallisuuden järjestelmällistä hallintaa yleisellä tasolla. Työnantaja voi kuitenkin valita työpaikalle parhaiten

sopivat tavat ja keinot, joilla se toteuttaa turvallisuuden hallinnan tavoitteita. Laissa [2002/738] määritellyt toiminnan keskeiset elementit ovat:

- Työsuojelun toimintaohjelma, joka voidaan ymmärtää joko yleisluontoisena toimintapolitiikkana tai yksityiskohtaisena toimintaohjelmana
- Haitta- ja vaaratekijöiden tunnistaminen sekä niiden poistaminen tai merkityksen arviointi, eli riskien arviointi
- Työntekijöille järjestettävä opetus ja ohjaus
- Työympäristön ja työyhteisön tilan jatkuva tarkkailu
- Riskien arvioinnin ajan tasalla pitäminen ja toimintaohjelman päivittäminen.

4.2 Tietoturva ja turvallisuusjohtaminen

Valtiohallinnon tieto- ja kyberturvallisuuden johtoryhmän vuonna 2016 suorittaman julkisen hallinnon henkilöstön ja johdon tietoturvabarometrin mukaan asennoituminen tietoturvallisuuteen on erittäin myönteistä. Peräti 60,8 % vastaajista oli sitä mieltä, että tietoturvallisuus mahdollistaa laadukkaan toiminnan ja antaa heidän organisaatiostaan laadukkaan kuvan. Vastaajista 32,3 % oli sitä mieltä, että tietoturvallisuutta tarvitaan, jotta he voivat käsitellä työssään tarvittavia tietoja. Lisäksi 6,7 % ymmärsi tietoturvan merkityksen, vaikka se aiheutti toisinaan ylimääräistä työtä. Ainoastaan 0,2 % piti tietoturvaa toimintaansa haittaavana tekijänä. Tietoturvallisuus on siis suuren enemmistön mielestä laadukkaan toiminnan tarpeellinen mahdollistaja. Tätä havaintoa tukee se, että johto koki tietoturvallisuuden eri osa-alueet pääsääntöisesti erittäin tärkeiksi [Valtioneuvosto 2016].

Vastauksissa korostui puutteita toimitilaturvallisuudessa, ohjelmien ajantasaisuudessa sekä selkeiden roolien ja vastuiden jakamisessa. Salasanojen hallinnan ja sähköpostin käytön toivotaan olevan helpompaa. Salasanojen turvallisuutta voidaan parantaa kohdistetulla ohjeistuksella sekä ottamalla käyttöön kustannustehokkaita salasanojen hallintaohjelmistoja [Valtioneuvosto 2016].

Vastaajista lähes kaikki koki olonsa joko hyvin turvalliseksi tai melko turvalliseksi päätelaitteilla työskennellessään. Heille tärkeät tietoturvallisuuden osa-alueet eivät keskimäärin joko huolestuttaneet olleenkaan tai huolestuttivat vain vähän. Turvallisuuden tunne oli korkea ja huolestuneisuus eri uhista yleensä olematonta tai vähäistä. Turvallisuuden tunne ja huolestuneisuuden puute eivät kuitenkaan ole linjassa vastaajien saaman koulutuksen ja ohjeistuksen määrän kanssa. Ainoastaan 29,7 % vastaajista oli saanut eri tietoturvallisuuden osa-alueisiin keskimäärin riittävästi koulutusta [Valtioneuvosto 2016].

Tuloksien perusteella voidaan todeta, että tietoturvallisuus on otettu osaksi turvallisuusjohtamista. Vaikka koulutusta ja ohjeistusta ei nähdä riittäväksi, se ei siltikään aiheuta heikkoa turvallisuuden tunnetta. Kyse voi olla esimerkiksi siitä, että saatua koulutusta ja ohjeistusta ei välttämättä tunnisteta, tai henkilö on saanut koulutusta ja osaamista vapaa-ajalla. Vastaajat eivät myöskään todennäköisesti tunnista niitä riittämättömään koulutukseen ja ohjeistukseen liittyviä riskejä, joita he kohtaavat työssään. Saattaa olla, että vastaajien luottamus omiin taitoihinsa on korkealla, vaikka he havaitsevatkin saamansa koulutuksen ja ohjeistuksen riittämättömyyden [Valtioneuvosto 2016].

5. Yhteenveto

Tietoturvasta ja kyberturvallisuudesta on tullut yhä keskeisempi osa yritysturvallisuuta. Se on tuonut mukanaan uuden uhkien ulottuvuuden yrityksille ja eroaa huomattavasti tavanomaisemmista rikosuhkista. Vaikka tietoturvallisuus on sisällöltään varsin laaja ja abstrakti käsite, sen osa-alueiden tarkastelun kautta lukija toivottavasti ymmärsi sen tarkoituksen käytännössä. Tietoturvaa kehittämällä ja ylläpitämällä yritys pyrkii suojaamaan tietonsa ja toimintansa jatkuvuuden tulevaisuudessa. Teknologia kehittyy jatkuvasti, joten ajan tasalla pysymiseksi yrityksen on seurattava omia tietoturvamenetelmiä ja kehitettävä toimenpiteitä. Vaikka täydellistä tietoturvaa on mahdotonta saavuttaa, voidaan silti löytää keinoja suurimpien tietoturvauhkien torjumiseen.

Käyttäjille ja yrityksille olennaisia asioita ovat organisaation ajantasainen tietoturvapolitiikka ja tietoturvaperiaatteet, tietoturvavastuiden jako työjärjestyksessä ja tehtäväkuvauksissa, tietoturvaohjeiden laatu ja kattavuus sekä allekirjoitetut vakuutukset tietoturvaohjeiden lukemisesta ja noudattamisesta. Niiden noudattaminen takaa hyvän tietoturvatason saavuttamisen yrityksissä.

Yrityksen tietoturvallisuuteen liittyviä ohjeita löytyy paljon, ja niiden perusta löytyy suurimmiksi osiksi kansainvälisistä ohjeistuksista. ISO 27000 -perheeseen kuuluvat standardit määrittelevät pitkälti tietoturvallisuuden riskien hallintatoimenpiteet ja siihen liittyvien ohjeiden perustan. Muita ohjeistuksia löytyy paljon, mutta osaa niistä ei ole validoitu eivätkä ne siten ole hyvä perusta yrityksen tietoturvallisuuden johtamiselle. Suomenkielisiä ohjeistuksia löytyy, ja esimerkiksi Viestintäviraston julkiselle sektorille laatimat VAHTI-ohjeistukset ovat päteviä myös yksityisen puolen tietoturvatoimintaan.

Viiteluettelo

Aluehallintovirasto. 2010. *Turvallisuusjohtaminen*. Multiprint oy.

Matt Bishop. 2003. What is computer security? *IEEE Security & Privacy*, 99(1), 67-69.

Panu Häikiö. 2009. *Turvallisuusjohtaminen Finaviassa.* Tutkielma. Teknillinen korkeakoulu.

International Organization for Standardization. *ISO 27002 – Information security - Security techniques – Code of practice for information security controls.*

Hanna Kaukver. 2012. *Kohti parempaa tietoturvaa pienyrityksessä*. Opinnäytetyö. Laurea-ammattikorkeakoulu.

Kauppa- ja teollisuusministeriö. 2005. *Laki vaarallisten kemikaalien ja räjähteiden käsittelyn turvallisuudesta 390/2005*.

Noora Koivula. 2013. *Yritysturvallisuuden hallinta – Pk-yritysten tarpeet ja käsitykset.* Opinnäytetyö. Seinäjoen ammattikorkeakoulu.

Juha Leppänen. 2006. *Yritysturvallisuus käytännössä – turvallisuusjohtamisen portfolio*. Talentum.

Juha Miettinen. 2002. *Yritysturvallisuuden käsikirja*. Kauppakaari.

Kalevi Mäkinen. 2007. *Yrityksen strateginen kokonaisturvallisuus*. Edita.

Henna Piisku ja Elviira Saari. 2007. *Turvallisuusjohtaminen ylimmän liikkeenjohdon näkökulmasta*. Opinnäytetyö. Laurea ammattikorkeakoulu.

M. Siponen and R. Willison. 2009. Information security management standards: Problems and solutions. *Information & Management*, 46(5), 267-270.

B. Von Solms and R. Von Solms. 2004. The 10 deadly sins of information security management. *Computers & Security*, 23(5), 371-376.

Sosiaali- ja terveysministeriö. 2003. *Työturvallisuuslaki 738/2002*.

Antti Sulosaari. 2004. *Tietoturva-ammattilaisen osaamistarvekartoitus*. Diplomityö. Tampereen teknillinen yliopisto.

Mika Susi. 2015. *Yritysturvallisuus*. Elinkeinoelämän keskusliitto.

Suupohjan peruspalvelu-liikelaitoskuntayhtymä. 2012. *Tietoturva- ja tietosuojapolitiikka.*

Valtiovarainministeriö. 2016. *Henkilöstön ja johdon tietoturvabarometri*. Valtioneuvoston tietotuki- ja julkaisuyksikkö.

Valtiovarainministeriö. 2014. VAHTI 3/2014 - *Tietoturvallisuuden arviointiohje*. Valtioneuvoston tietotuki- ja julkaisuyksikkö.

Valtiovarainministeriö. 2008. *VAHTI 2/2008 – Henkilöstöturvallisuus osana tietoturvallisuutta.* Valtioneuvoston tietotuki- ja julkaisuyksikkö.

Hannu Väistö. 2005. *Hallinnollinen tietoturvallisuus*. Dataratas.

YTNK. 2016. *Elinkeinoelämän yritysturvallisuusmalli*. Elinkeinoelämän keskusliitto.

Käytettävyyden ja tietoturvallisuuden yhdistäminen käyttäjätodennuksessa

Johanna Ylitervo

Tiivistelmä.
Tutkielmassa käsitellään erilaisia käyttäjätodennuksen menetelmiä, niille ominaisia piirteitä sekä niiden käytettävyyttä ja tietoturvallisuutta. Käyttäjätodennuksen tietoturvallisuuden merkitys kasvaa jatkuvasti digitalisaation edetessä, mutta todennuksen menetelmien koetaan joutuvan tasapainoilemaan tietoturvallisuuden ja käytettävyyden välillä. Ongelmiin ei ole löydetty ratkaisua, vaikka tarve on suuri. Tässä tutkielmassa esitellään erilaisia käyttäjätodennusmenetelmiä, niiden vahvuuksia ja heikkouksia ja pyritään esittämään syitä, miksi käytettävyyden ja tietoturvallisuuden suhde on niin monimutkainen eikä siihen ole löydetty kaikenkattavaa ratkaisua.

Avainsanat ja -sanonnat: Käytettävyys, tietoturvallisuus, käyttäjätodennus.

1. Johdanto

Digitalisaation edetessä yhä enemmän tietoa käsitellään ja säilytetään verkossa. Samaan aikaan kun tiedonsaanti helpottuu, kasvaa riski yksityisten tietojen joutumisesta vääriin käsiin. Tiedon turvaamisen merkitys siis lisääntyy koko ajan, mutta itse käyttäjät nähdään edelleen pahimpana uhkana tietoturvallisuudelle, vaikka siihen pyritään heidän edukseen.

Tässä tutkielmassa esitellään erilaisia käyttäjätodennuksen menetelmiä, niiden tiedon turvaamisen onnistumista, kyseisten menetelmien aiheuttamia haittoja käytettävyydelle sekä näiden käytettävyysongelmien mahdollisia ratkaisuja. Käyttäjien pyrkiessä tehokkaaseen ja miellyttävään työskentelyyn, he eivät aina koe käyttöä hidastavien ja vaikeuttavien tietoturvaprosessien toteuttamista itselleen tarpeeksi hyödyllisinä. Eri käyttäjätunnusten ja salasanojen hallinnasta on tullut jokaiselle arkipäivää, mikä kasvattaa ihmisten kognitiivista taakkaa päivittäisissä askareissa huomattavasti. Ihmisiä on erotettu toisistaan niin sanotun käyttäjätunnistuksen avulla jo kauan ennen kuin tietokoneita oli olemassa, mutta tietokoneiden aikana informaation jaon tapa on muuttunut radikaalisti ja entistä enemmän henkilökohtaista ja erittäin arkaa informaatiota jaetaan näkemättä vastapuolta konkreettisesti.

Luvussa 2 esitellään tutkielmassa käytetyt keskeisimmät käsitteet. Luvussa 3 siirrytään tietoperustaisen käyttäjätodennuksen syvempään käsittelyyn ja kategorian tunnistautumismenetelmistä käsitellään laajimmin todennusta käyttäjätunnuksen ja salasanan avulla. Luvussa 4 käsitellään identiteettiperustaisen käyttäjätodennuksen menetelmää, biometriikkaa, ja luvussa 5 esineperustaisia

tunnistautumismenetelmiä. Kaikissa näissä kolmessa luvussa ensimmäisenä määritellään tarkemmin todennusmenetelmä ja sille ominaisia piirteitä, sen jälkeen käsitellään menetelmän käytettävyyttä ja tietoturvallisuutta. Luku 6 on tutkielman yhteenveto ja siinä esitellään lyhyesti tutkielman sisällöstä saatuja johtopäätöksiä.

2. Käsitteet

2.1. Tietoturvallisuus

Tietoturvallisuuden käsikirjan [Hakala et al. 2006] mukaan klassinen tietoturvallisuuden tiedon arvoon perustuva määritelmä jakaa tietoturvan kolmeen eri osa-alueeseen: *luottamuksellisuuteen* (confidentality), *eheyteen* (integrity) ja *saatavuuteen* (availability). Lähteestä riippuen availability voidaan suomentaa myös käytettävyydeksi, mutta tässä tutkielmassa käytetään saatavuus-suomennosta pyrkien välttämään sekaannuksia *käytettävyyden* (usability) kanssa.

Luottamuksellisuudella taataan tietojen olevan ainoastaan niiden osapuolien käytössä, joilla on tietoihin oikeus. Tämä saavutetaan suojaamalla tiedot salauksella tai estämällä niihin pääsy ilman myönnettyjä käyttöoikeuksia, jotka varmistetaan *käyttäjätodennuksen* (user authentication) avulla.

Tiedon eheys takaa tiedon oikeellisuuden. Tiedot eivät saa sisältää tahallisia tai tahattomia virheitä, mikä voidaan tarkistaa esimerkiksi erilaisin ohjelmistoteknisin ratkaisuin kuten lisäämällä sovelluksiin erilaisia tiedon syöttörajoituksia ja sen tarkisteita. Eheyden kriteeriksi kuuluu myös se, ettei tietoa ole muutettu sen jälkeen, kun sen todennettu luoja taikka käsittelijä on sitä viimeksi tarkoituksellisesti muokannut.

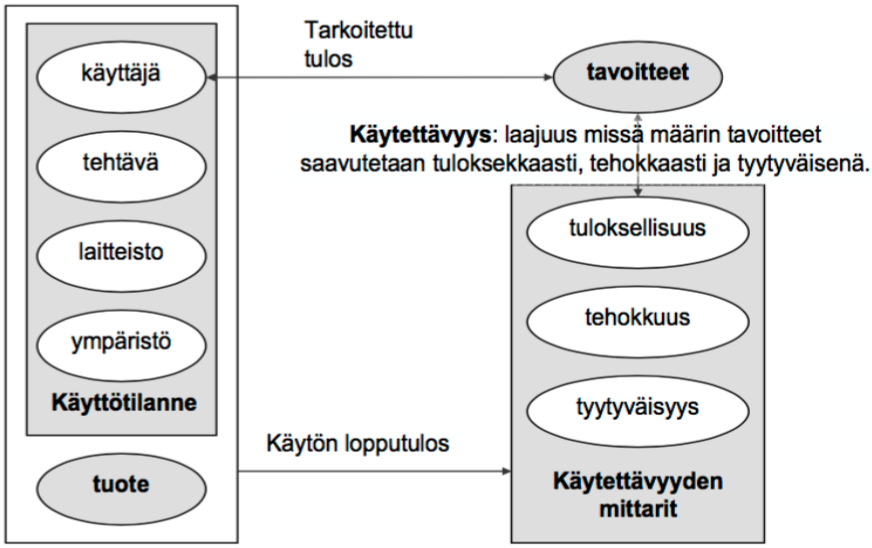
Saatavuutta ylläpidetään huolehtimalla siitä, että käyttäjät saavat tarvitsemansa tiedot käyttöönsä mahdollisimman tehokkaasti ja haluamassaan muodossa. Asiallinen laitteisto sekä tietotyyppiä mahdollisimman hyvin palveleva ohjelmisto ovat saatavuuden onnistumiseen keskeisesti vaikuttavia tekijöitä.

Laajennetussa turvallisuuden määritelmässä [Hakala et al. 2006] klassisen turvallisuuden määritelmän osatekijöihin lisätään *kiistämättömyys* (non-repudiation) sekä *pääsynvalvonta* (access control). Aiempi tietoturvallisuuden määritelmä on koettu puutteellisena nykytilanteessa, koska se jättää liian vähälle huomiolle tiedon tuottajan tai omistajan identiteetin sekä laitteistojen ja tieto- sekä tietoliikennejärjestelmien arvon. Kiistämättömyydellä pyritään takaamaan tiedon luojan tai sen käyttäjän henkilöllisyys niin, ettei siitä voi olla epäilystä. Jos tietoa käytetään väärin, tulee tietojärjestelmän omistajan pystyä todistamaan, kuka tiedon on luonut sekä kuka on käyttänyt sitä. Kiistämättömyys pyritään toteuttamaan käyttämällä erilaisia salausmenetelmiin liittyviä tunnis-

tusmekanismeja sekä biometrisiä tunnisteita, jotka ovat siis käyttäjätodennukseen keskeisesti liittyviä aiheita. Näitä käsitellään laajemmin kohdassa 2.3.

Pääsynvalvonta rajoittaa tietojenkäsittelyn infrastruktuurin käyttöä. Kun luottamuksellisuus suojelee varsinaisiin tietoihin pääsyä, pääsynvalvonta tarkkailee organisaation laitteiden tai tietoliikenneyhteyksien käyttöä omiin tarkoituksiin, olkoon sitten kyse ulkopuolisista tai omista työntekijöistä. Luvattomalla käytöllä tietojärjestelmiä kuormitetaan turhaan, mikä heikentää tietojen saatavuutta sekä altistaa haittaohjelmien leviämiselle. Nämä taasen voivat johtaa eheys- ja luottamusongelmiin. Tietoturvallisuus kattaa siis monia toisistaan riippuvaisia osa-alueita, jotka ovat tiiviissä yhteydessä toisiinsa: kun yksi osa-alue kärsii, voi se korreloida negatiivisesti toiseen osa-alueeseen.

Tietoturvallisuuden osatekijät ovat ymmärrettäviä, mutta tietoturvallisuuden mittaaminen on asia erikseen. O'Gorman [2003] huomauttaa, ettei tietoturvallisuuden vahvuuden arviointiin löydy yksiselitteistä mitta-asteikkoa vaan turvallisuusjärjestelmät ja -tekniikat kuvataan usein joko vahvoina tai heikkoina. Suhteellisia termejä käytettäessä merkitys on usein selvä: ovi, jossa on lukko, on turvallisuudeltaan vahvempi kuin ovi ilman lukkoa. Käyttäjätodennuksen saralla esimerkkinä voidaan käyttää esimerkiksi luottokorttia. Luottokortti yksinään on heikko turvallisuudeltaan, sen helpon varastettavuuden ja heikon kiistämättömyyden osalta. Kortin omistaja voi helposti väittää valheellisesti, ettei käyttänyt korttia, vaikka todellisuudessa näin tapahtui. Kun kortin käyttäjätodennukseen lisätään allekirjoitus, on kortin tietoturvallisuus vahvempi kiistämättömyyden osalta, verrattuna siihen tilanteeseen, jossa allekirjoitusta ei käytetä. Tietoturvallisuuden arviointi tilanteista riippumattomilla tekijöillä on hankalampaa. Tietoturvallisuutta voidaan mitata hyökkäyksen kulujen ja saadun hyödyn suhteella. Jos murtautujalle tulee enemmän kuluja kuin saavutettuja tuloja hyökkäyksestä, voidaan järjestelmää pitää tietoturvallisena. Kuluihin lasketaan rahallisten kulujen lisäksi myös esimerkiksi ajalliset kulut sekä riski vankeusrangaistukseen. Kansainvälinen standardisoimisjärjestö (International Organization for Standardization, ISO) ja kansainvälinen sähköalan standardoitiorganisaatio (International Electrotechnical Commission, IEC) ovat julkaisseet vuonna 2016 ISO/IEC 27004 -standardin tietoturvallisuuden valvontaan, mittaamiseen, analysointiin ja arviointiin, josta aiheesta saa lisää tietoa. Tietoturvallisuutta mitattaessa tulee kuitenkin muistaa, että tietoturvallisesti vahvasta järjestelmästä saadaan nopeasti heikko, jos käyttäjät eivät noudata heille annettuja ohjeita.

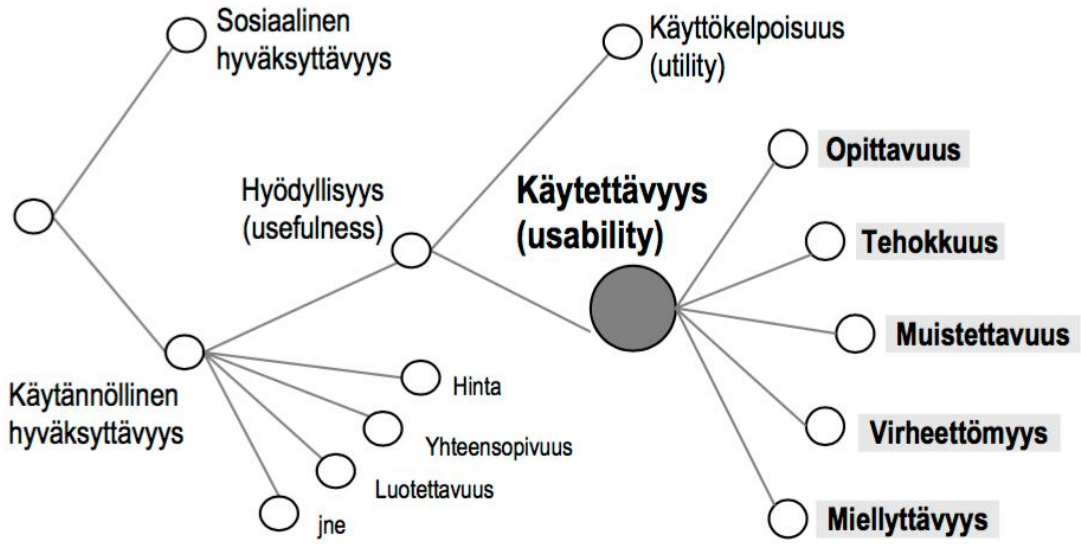


Kuva 1. Käytettävyyden osatekijät ISO 9241-11 standardin mukaan. Suomennos [Ovaska et al. 2005]

2.2. Käytettävyys

Käytettävyyden määritellään ISO 9241-11 -standardin [1998] mukaan kertovan, kuinka ansiokkaasti käyttäjä saavuttaa toivomansa määränpään *tuloksellisesti* (effectiveness), *tehokkaasti* (effiency) sekä käyttäjää *tyydyttävällä tavalla* (satisfaction). Edellä mainittuja attribuutteja tarkastellaan käyttäjän kokemana tietyssä käyttökontekstissa. Käyttökontekstiin sisällytetään käyttäjän henkilökohtaiset ominaisuudet, hänen tekemänsä tehtävä sekä siihen käytettävä laitteisto ja käyttötilanteen ympäristö. *Käyttäjän kokemusta* (user exprience, use experience) käyttötilanteesta sekä hänen aikaansaamaansa tulosta verrataan käyttäjän alkuperäisiin tavoitteisiin, jolloin saadaan tietoa käytön onnistumisesta. Määrittelyn tavoitteena onkin korostaa käytettävyyttä käyttäjän sekä tuotteen vuorovaikutuksen tuloksena ennemmin kuin laitteen ominaisuutena sekä ottaa huomioon erilaisten käyttäjien henkilökohtaiset tarpeet.

Nielsenin käsiterakenne [1993] käytettävyydestä laajentaa ISO 9241-11 -standardia *opittavuuden* (learnability), *muistettavuuden* (memorability) ja *virheettömyyden* (errors) käsitteillä. Nielsen erottaa *käyttökelpoisuuden* (utility) käsitteen käytettävyydestä korostaakseen, että tuotteen käyttökelpoisuus saadaan selville vasta käytännön tehtävissä, eikä sitä voi etukäteen tutkia kuten käytettävyyttä. Opittavuudella Nielsen tarkoittaa sitä, kuinka nopeasti käyttäjä kykenee omaksumaan uuden käyttöliittymän, josta seuraa jatkumo muistettavuuteen eli kuinka käyttäjä muistaa oppimansa käyttökerrasta seuraavaan. Virheettömyydellä tarkoitetaan virheiden mahdollisimman vähäistä määrää käytön aikana.



Kuva 2. Käytettävyyden osatekijät Nielsenin [1993] mukaan. Suomennos [Ovaska et al. 2005].

Kuten kuvasta 2 huomataan, käytettävyys on ainoastaan yksi osa-alue hyväksyttävyydestä. Käytettävyys ja käyttökelpoisuus muodostavat yhdessä hyödyllisyyden, joka kuuluu käytännöllisen hyväksyttävyyden osa-alueeseen.

2.3. Käyttäjätodennus

Käsiteltäessä tietoturvallisuuden määritelmästä muun muassa luottamuksellisuutta, kiistämättömyyttä sekä pääsynvalvontaa kohdassa 2.1, nousi esille käyttäjän tunnistaminen sekä hänen identiteettinsä vahvistaminen. Tässä prosessissa on kyse käyttäjätodennuksesta, jossa ihminen todistaa tietokoneelle todella olevansa väittämänsä henkilö.

Käyttäjätodennusprosessi jaetaan Shireyn [2000] mukaan kahteen eri vaiheeseen: käyttäjän *tunnistamiseen* (user identification), jossa käyttäjä esittää järjestelmälle *tunnisteen* (identifier) sekä sen *varmistukseen* (verification), jossa hän esittää tunnistustekijän, joka vakuuttaa, että käyttäjä on esittämänsä tunnistuksen oikeutettu omistaja. Käyttäjätodennuksessa on siis kyse yksittäisvertailusta, jossa käyttäjän antaman hakutunnisteen varmistetaan olevan sama kuin esimerkiksi järjestelmässä säilytetyn vertailutunnisteen. Käyttäjätunnistuksessa, joka on käyttäjätodennusprosessin ensimmäinen vaihe, henkilön antamaa hakutunnistetta verrataan useisiin eri vertailutunnisteisiin, jolloin kyseessä on seulontavertailu. [Metsämäki 2016]

Käyttäjän mahdolliset tunnistustekijät jaetaan useimmiten kolmeen eri kategoriaan: *jotain mitä käyttäjä tietää*, esimerkiksi salasana, PIN-koodi tai vastaus turvallisuuskysymykseen, *jotain mitä käyttäjällä on*, esimerkiksi avain tai ID-

kortti sekä *jotain mitä käyttäjä on tai tekee* [Shirey, 2000]. Jotain mitä käyttäjä on viittaa biometriseen tunnistukseen, joka perustuu käyttäjän fysiologisiin yksilöllisiin piirteisiin, kuten esimerkiksi sormenjälkeen tai silmän värikalvoon, iirikseen. Tekemisellä käyttäjät voidaan erottaa esimerkiksi puhenäytteen avulla. Kyseinen määritelmä on kuitenkin osaltaan ongelmallinen. O'Gorman [2003] huomauttaa, että esimerkiksi salasana ei ole jotain mitä käyttäjä tietää, vaan jotain mitä hän painaa mieleensä ja sen takia se voi unohtua. Määritelmän aukkojen takia O'Gorman jakaa tunnistusmekanismit mieluummin *tietoperustaisiin* (knowledge-based), *esineperustaisiin* (object-based) sekä *identiteettiperustaisiin tunnistusmenetelmiin* (ID-based authenticators).

Tietoperustaisille tunnistusmenetelmille on tunnusomaista niiden salailu sekä epäselvyys muille kuin itse käyttäjälle. Kategoriaan luetaan salassa pidetyt salasanat kuten myös epäselvät tiedot, jotka eivät ole varsinaisesti salaisuuksia, mutta ne eivät ole tiedossa suurimalle osalle ulkopuolisista. Tällaisia tietoja ovat esimerkiksi käyttäjän lempiväri tai hänen äitinsä tyttönimi. Kategorian heikkoutena on tunnisteen salausvoiman väheneminen joka kerta, kun tunnistetta käytetään.

Esineperustaiset tunnistusmenetelmät kattavat kaikki konkreettisesti fyysisesti omistettavissa olevat esineet. Yleisin esineperustainen tunnistusmenetelmä on fyysinen avain. Turvallisuutta koskeva haittapuoli näissä tunnistautumismenetelmissä on niiden kadottamisen tai varastetuksi tulemisen mahdollisuus. Jos käyttäjä kadottaa avaimensa, johon ei ole liitetty toista erilaista tunnistautumismekanismia, voi varas päästä turvattuun paikkaan yksinkertaisesti käyttämällä haltuunsa saamaan avainta. Esineperustaisissa tunnistusmenetelmien käytössä on kuitenkin se etu, että käyttäjä huomaa helposti, kun tunniste ei ole enää hänen hallussaan ja voi toimia heti asiaan kuuluvala tavalla. Monesti käyttäjä ei välttämättä edes huomaa, jos hänen salasanansa on hakkeroitu, jos siitä ei tule mitään välittömiä ja konkreettisesti huomattavia seurauksia.

Identiteettiperustaiset tunnistemenetelmät käyttävät tunnistautumisessa hyväkseen ihmisen yksilöllisyyttä. Kategoriaan kuuluvat esimerkiksi erilaiset identiteettidokumentit, kuten ajokortti, passi ja luottokortti, sekä biometriikka. Biometriikka pyrkii hyödyntämään ihmisten yksilöllisiä fysiologiaan perustuvia ominaisuuksia, kuten sormen jälkien tai silmän iiriksen yksilöllisiä piirteitä. Identiteettiperustaisten tunnistusmenetelmien vahvuus on niiden väärentämisen tai kopioimisen vaikeudessa. Jos kopiointi kuitenkin onnistuu, on niitä paljon hankalampi korvata kuin tieto- ja esineperustaisissa menetelmissä. Huomattavaa biometriseen tunnistautumiseen liittyen on se, että toisin kuin esimerkiksi salasanat, niiden tunnistuskyky ei ole riippuvainen niiden salassapidosta. Biometriikan käyttö tunnistusmenetelmänä on kuten passin numero:

passin käyttö tunnisteena ei ole riippuvainen passin numeron salaamisesta, vaan siitä, kuinka hankalaa aitoa dokumenttia on väärentää.

Turvallisuuden parantamiseksi edellä mainittuja tunnistusmenetelmiä voidaan käyttää yhdessä eri kategorioista, jolloin kyseessä on *monimenetelmäinen todentaminen* (multifactor authentication). Eri menetelmiä yhdistelemällä saadaan käyttöön eri kategorioiden vahvuudet ja pystytään paikkaamaan niille tyypillisiä heikkouksia. Esimerkiksi pankkikorttiin liitetään yleensä tietoperustainen tunnistusmenetelmä, kuten *tunnusluku* (PIN, Personal Identification Number). Jos pankkikorttia käytettäisiin yksinään esineperustaisena tunnistautumismenetelmänä, olisi se haavoittuvainen väärinkäytöksille johtuen sen varastamisen helppoudesta. Kun tunnistautumisprosessiin lisätään tunnusluku, täytyy hyökkääjän selvittää myös salainen tunnusluku koko suojauksen murtamiseksi. Kun kaksi eri kategorian todentautumismenetelmää yhdistetään, kutsutaan sitä *kaksiosaiseksi todentamismenetelmäksi* (two-factor authentication). Kaksiosainen todentamismenetelmä on yleisemmin käytetty kuin monimenetelmäinen todentautumismenetelmä, jossa käytetään kaikkia kolmea eri tunnistusmenetelmäkategoriaa.

Metsämäen mukaan [2016] käyttäjän tunnistusmenetelmän luotettavuus on riippuvainen tiettyjen ominaisuuksien täyttämisestä. Tunnistusmenetelmän tulee pystyä *yksilöimään* eri henkilöt toisistaan niin, että he ovat selkeästi erotettavissa. Tunnistuksen tulee olla *toistettavissa* niin, että tulokset ovat mahdollisimman samoja kaikilla eri tunnistuksen kerroilla. Tunnistustavan tulee olla myös *esteetön* ottaen huomioon erilaiset käyttäjäryhmät, eikä se saa erotella heitä heidän toimintakykynsä perusteella. Esteettömyyttä sivuavasti tunnistusmenetelmän tulee olla myös *yleisluontoinen* eli sen tulee sopia kaikille erilaisille käyttäjille ja ottaa huomioon sellaisten piirteiden käyttömahdollisuus, jotka ovat ominaisia yhteisesti kaikille ihmisille. Viimeisenä kriteerinä tunnistusmenetelmän tulee olla *hyväksyttävä*. Hyväksyttävyys jaetaan sosiaaliseen ja käytännölliseen hyväksyttävyyteen (vrt. kuva 2) ja tärkeänä alaosana käytännölliseen hyväksyttävyyteen liittyy käytettävyys, jota käsiteltiin syvällisemmin kohdassa 2.2.

3. Tietoperustainen tunnistautuminen: salasanat

Suosituin käyttäjätodennuksen mekanismi on käyttäjätunnus ja siihen liitetty salasana. Oxfordin Tietojenkäsittelytieteiden sanakirjan [2016] mukaan salasana on uniikki merkkijono, joka säilytetään tietojärjestelmässä käyttäjätodennusta varten. Todennusprosessin aikana käyttäjä syöttää salasanan, jonka tulee vastata järjestelmässä säilössä olevaa käyttäjätunnukseen liittyvää vastinetta, ennen kuin käyttäjä päästetään järjestelmään sisään. Kyseisessä määritelmässä mainit-

tu uniikkiuden käsite on harhaanjohtava. Ainutlaatuisuus olisi salasanan tietoturvallisuutta vahvistava ominaisuus, joka kuitenkin toteutuu valitettavan harvoin.

Salasanalla on paljon erilaisia ominaisuuksia, jotka riippuvat järjestelmästä: se voi olla *käyttäjän* (human generated) tai *satunnaisgeneraattorin* (random generator) *luoma*, käyttöajaltaan *kertakäyttöinen* (one-time password), *määräaikainen* (temporary) tai *pysyvä* (static), ja siihen käytettävien merkkien joukko voi vaihdella. Mahdolliseen merkkijoukkoon voi kuulua ainoastaan pienaakkosia, suuraakkosia, erikoismerkkejä tai numeroita sekä näiden kaikkien joukkojen eri yhdistelmiä. Esimerkkinä rajatun merkkijoukon salasanasta toimii esimerkiksi tunnusluku, johon käytetään ainoastaan numeroita. Tunnusluvut ovat yleisiä esimerkiksi pankkien käyttämässä käyttäjätodennuksessa. Salasana voi koostua myös lauseesta (passphrase), jolloin se ei eroa käyttötavoiltaan salasanoista mutta on merkkimäärältään suurempi.

Huolimatta siitä, että salasanojen kuolemista on ennustettu jo pitkään (muun muassa Bill Gates [2004]), säilyttävät ne silti suosionsa käyttäjätodennuksessa. Käyttäjätunnuksen ja salasanan koetaan olevan vähiten kustannuksia aiheuttava käyttäjätodennuksen malli, mutta ne aiheuttavat usein käyttäjille päänvaivaa sekä syventävät kuilua turvallisuusammattilaisten ja käyttäjien välillä. Miksi salasanoja käytetään edelleen runsaasti, vaikka niihin liittyy turvallisuusriskejä ja käytettävyyshaittoja?

3.1. Salasanojen hyvät puolet ja miksi ne pysyvät käytössä

Ennen käytettävyyshaittojen ja tietoturvariskien ruotimista on hyvä pohtia, kuinka salasanat ovat alunperin saavuttaneet suosionsa. Herley ja van Oorschot [2012] toteavat, että salasanat, huolimatta huonoista puolistaan, ovat osaltaan mahdollistaneet internetin nopean leviämisen sekä käytön laajamittaisen kasvun. Tämän hetken yli kolme ja puoli miljardia internetin käyttäjää [Internet World Stats, 2017] pääsevät kirjautumaan sosiaalisen median tileilleen, sähköposteihinsa sekä lukuisiin muihin palveluihin ympäri maailmaa niiden käyttäjätodennuksen ollessa sitoutumaton paikkaan tai aikaan ja sen vaatiessa välineeksi ainoastaan tietokoneen ja selaimen. Salasanojen toimintaperiaate on myös perusteiltaan käyttäjille helposti ymmärrettävä ja niiden käyttöönotto on yksinkertaista automaattisen ja välittömän luomismahdollisuuden ansiosta. Salasanat ovat myös käyttäjien unohduksille anteeksiantavaisia. Jos käyttäjä ei muista enää salasanansa, voi hän klikata ”salasana unohtui” -linkkiä, jolloin hän saa luotua heti uuden salasanan ilman, että se aiheuttaa palveluun kirjautumiseen suuria viivytyksiä. Salasanojen hyödyntäminen on myös lähes ilmaista palveluntarjoajan näkökulmasta. Facebookin käyttäjämäärät kasvoivat mil-

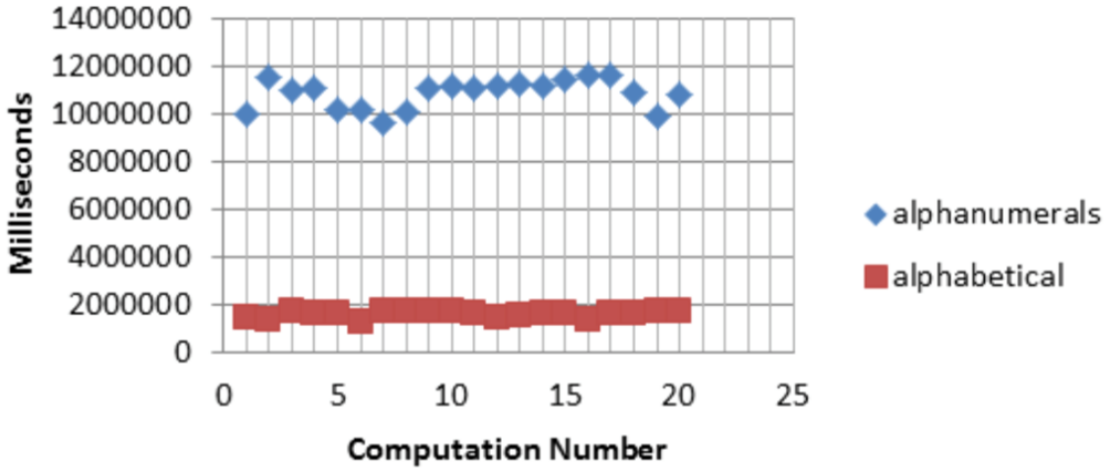
joonaan ennen kuin sen perustajat ottivat vastaan mitään rahoitusta. Tämänkaltainen kuluton käyttäjämäärien mittava kasvu ei olisi ollut mahdollista, jos käyttäjätodennuksesta olisi tullut kuluja aloittelevalle yritykselle.

Edellä mainitut syyt ovat osaltaan vaikuttaneet salasanojen hallitsevaan asemaan käyttäjätodennuksessa. Toisaalta salasanat kuitenkin pitävät paikkansa myös sen takia, että niiden korvausyritykset ovat epäonnistuneet erinäisistä syistä. Herley ja van Oorschot [2013] nostavat esiin vallitsevan piirteen salasanojen syrjäyttämisskenaariossa: mahdolliselle korvaajalle ei aseteta tarpeeksi täsmällisiä vaatimuksia eikä aikaisemmin epäonnistuneista yrityksistä ole otettu oppia.

3.2. Salasanojen tietoturvallisuus ja käytettävyys

Salasanojen hyvät puolet jäävät usein taka-alalle käytettävyysongelmien hallitessa keskustelua, mikä on ymmärrettävää, kun haittoja on paljon enemmän kuin hyviä puolia. Ongelmat voivat johtua monesta eri tekijästä, mutta useimmiten ne liittyvät käyttäjien mukavuudenhaluun ja pyrkimykseen työskennellä mahdollisimman tehokkaasti sekä heidän vähäisestä ymmärryksestä tietoturvallisuudesta.

Salasanojen tietoturvallisuus on riippuvainen monesta eri tekijästä. Sisältönsä puolesta tietoturvallisesti vahvana käsitetty salasana on yhtä turvaton tietojärjestelmähyökkäyksien edessä kuin heikko salasana, jos tietoja ei ole suojattu erilaisilla salauksen keinoilla ennen järjestelmään tallentamista. Salasanat voidaan käsitellä esimerkiksi *kryptaamalla* (encrypting), *tiivistämällä* (hashing) tai yhdistämällä tiivistysalgoritmiin parametriksi lisämerkkijonon, jota kutsutaan *suolaksi* (salted hashing). Kaikkien näiden metodien käyttö turvaa salasanoja tietokantamurtojen varalta. Kryptauksessa käytetään salausavainta, jonka avulla tietue muutetaan funktion kautta salattuun muotoon. Hyökkääjä ei siis saa salasanoja käyttöönsä ilman kyseistä salausavainta. Usein avainta säilytetään kuitenkin samalla palvelimella, joten kryptaus ainoastaan hidastaa salasanojen murtoprosessia eikä tarjoa erityistä suojaa verrattuna kryptaamattomien salasanojen säilytysmuotoon. Tiivistys tarjoaa tähän kryptauksen huonoon puoleen ratkaisun: kun salasana on tiivistetty salattuun muotoon *tiivistysfunktion* (hash function) avulla, on prosessi peruuttamaton eli salasanan alkuperäiseen muotoon muuttamiseen ei ole algoritmia. Salasanojen murtaminen onnistuu ainoastaan kokeilemalla kaikkia mahdollisia vaihtoehtoja ja vertaamalla, mikä murtautujan ehdottama tiivistetty salasana on yhteneväinen tietokantaan tallennetun tiivistetyn muodon kanssa. Tiivistyksestä saadaan tietoturvallisesti vielä vahvempi, jos salasanaan lisätään suola. Suola on satunnaisesti muodostettu uniikki merkkijono, joka on jokaiselle käyttäjälle henkilökohtainen. Suo-



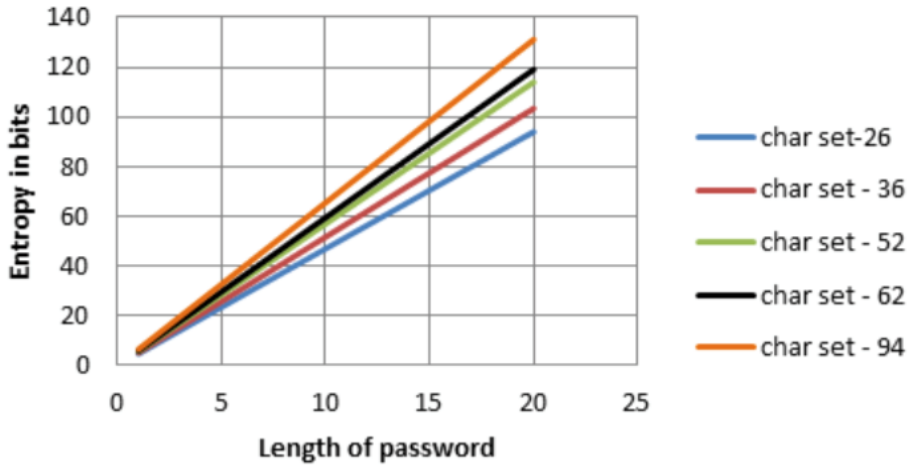
Kuva 4: Pienaakkosellisten ja aakkosnumeeristen salasanojen murtoajat [Chanda 2016]

lamerkkijonoa käytetään tiivistysalgoritmin parametrina salasanan lisäksi. Vaikka suolat säilytettäisiin tietokannassa, on salasanojen murtaminen monimutkaista ja aikaa vievää, koska suolat ovat pitkiä ja uniikkeja merkkijonoja. [Chanda, 2016]

Salasanan sisältö vaikuttaa oleellisesti sen turvallisuuden vahvuuteen. Yksi hyökkäyksen keinoista, *väsytyshyökkäys* (brute-force attack), esimerkiksi kokeilee jokaista mahdollista yhdistelmää tietystä merkkijoukosta eli mitä enemmän mahdollisia merkkien yhdistelmiä salasanassa on, sitä kauemmin sen murtaminen kestää. Chanda [2016] toteutti empiirisen kokeen, jossa hän vertaili tiivistettyjen salasanojen sisällön vaikutusta murtoon käytettyyn aikaan. Tuloksista nähdään selkeästi kuinka ainoastaan yhden merkin lisääminen salasanan pituuteen tai suur- ja pienaakkosten yhdisteleminen vaikeuttaa ja hidastaa salasanan murtoprosessia. Kuvasta 3 nähdään, kuinka salasanan murtoaika pitenee yhden kirjaimen lisäyksellä salasanaan. Testissä käytettiin pienaakkosista koostuvia viiden kirjaimen pituisia salasanoja, jotka on esitetty grafiikassa punaisella ja kuuden pienaakkosen salasanat on kuvattu sinisellä.

Kun viiden pienaakkosen sisältämän salasanan murto vie keskimääräisesti yhden minuutin, on kuuden merkin salasanojen väsytyshyökkäykseen käytetty aika jo yli 23 minuuttia. Merkkien lajilla vielä suurempi vaikutus murtoaikaan. Seuraavassa kokeessa verrattiin kuuden pienaakkosen sisältämän salasanan sekä kuuden merkin salasanan, jossa käytettiin myös numeroita, murtoaikaa toisiinsa. Jo kymmenen merkin lisäys mahdolliseen merkkijoukkoon lähes kuusinkertaisti murtoaikaa, kuten kuvasta 4 huomataan.

Kuuden aakkosnumeerisen merkin sisältämän salasanan nopeimmin suoritettu murto tapahtui yli kahdessa ja puolessa tunnissa.



Kuva 5: Salasanan pituuden ja merkkijoukon suuruus entropiaan [Chanda 2016]

Salasanan turvallisuutta mitattaessa entropialla havainnollistetaan salasanan informaatiosisällön vaikutusta salasanan vahvuuteen. Entropia on informaatioteorian suure, jolla ilmaistaan informaation satunnaisuuden määrää biteissä. Jos esimerkiksi merkkijono voi koostua ainoastaan A-merkistä, on merkkijonon entropia nolla, koska seuraavasta merkistä ei ole epävarmuutta.

Salasanan entropia on riippuvainen salasanan pituudesta sekä käytettävissä olevien merkkien lukumäärästä. Kuvasta viisi huomataan, että salasanan pituus kasvattaa entropiaa bittiä kohden tehokkaammin kuin käytössä olevan merkkijoukon koko.

Salasanojen tietoturvallisuutta voidaan mitata melko yksinkertaisin keinoin edellä esitetyillä tavoilla. Tästä huolimatta käyttäjät eivät usein noudata suosituksia ja heidän on todettu luovan mahdollisimman lyhyitä salasanoja mahdollisimman vähäisellä merkkiryhmien vaihtelulla [Herley and Florencio 2007]. Käyttäjien luomat salasanat sisältävät usein myös jonkun heille läheisen elementin, kuten kotiosoitteen tai lemmikin nimen [Yan et al. 2004], minkä takia ne ovat suuremmassa riskissä tulla hakkeroiduiksi esimerkiksi sanakirjahyökkäyksessä, jossa murrossa kokeillaan listaa sanoista tilin aukaisemiseen. Satunnaisgeneraattorin luomat salasanat ovat vaikeammin murrettavissa, mutta ne ovat käyttäjille hankalia muistettavia. Heikon muistettavuuden takia luontitapaa käytetään harvemmin ja sen käyttö johtaa usein salasanojen muistiinkirjoittamiseen, joka aiheuttaa taas oman tietoturvariskinsä. Vaikka salasanojen tietoturvallisuuden lisääminen olisi yksinkertaista ja pienien muutoksien aikaansaamat tulokset ovat tutkimuksesta havaittuina jokaiselle vaikuttavia, käyttäjät eivät tartu toimenpiteisiin, koska he ovat yleensä tietämättömiä siitä, kuinka salasanojen murtaminen todellisuudessa toimii [Adams and Sasse 1999]. Kun käyttäjät eivät ole tietoisia todellisista tietoturvariskeistä, he keksivät niistä

omia mallejaan ja kehittävät oman käsityksensä mukaisia tietoturvaa lisääviä toimintatapoja.

Salasanojen tietoturvaa parantavien metodien käyttö ja monimutkaisen tietoturvallisen salasanan luonti on teoriassa yksinkertainen ja helposti toteutettava tilanteessa, jossa käyttäjällä on ainoastaan yksi käyttäjätunnus ja salasana käytössään. Todellisuudessa nykyaikana käyttäjillä on useita tunnistautumispareja muistettavanaan eri järjestelmiin ja palveluihin, sekä työ- että henkilökohtaisessa elämässä. Helkalan ja Bakåsin [2012] Norjassa tekemän tutkimuksen mukaan tyypillisellä tietokoneen käyttäjällä oli noin 25 eri käyttäjätunnusta, joihin kaikkiin liittyi salasana ja käyttäjä kirjoitti keskimäärin kahdeksan salasanaa päivässä. Kun käyttäjillä on näin suuri määrä eri tunnistautumistietoja muistettavinaan, alkaa niiden hallinnasta tulla käyttäjille suuri kognitiivinen taakka [Adams and Sasse 1999]. Se johtaa niin sanottuun *salasanauupumukseen* (password fatigue), jolloin käyttäjät alkavat etsiä keinoja helpottaakseen muistitaakkaansa. Käyttäjät voivat esimerkiksi valita saman tai samankaltaisia käyttäjätunnus-salasana pareja useille eri tileille, eri järjestelmiin tai palveluihin [Dhamija and Dusseault 2008]. Kun käyttäjien muistitaakka heidän itsensä kokemana kevenee, on vaikutus todellisuudessa päinvastainen: Wickensin [1992] mukaan ihmisen kyky muistaa samankaltaisia sanoja heikkenee *listainterferenssin* (within-list interference) johdosta. Samankaltaisten käyttäjätunnuksien ja salasanojen käyttö kasvattaa myös virheellisten sisäänkirjautumisyrityksien määrää, kun käyttäjä ei muista välittömästi palvelun käyttäjätunnusta ja salasanaa vaan selvittää kokeillen, mitä variaatiota salasanasta hän käytti tämän sivun kirjautumistietoihin. Muistamishankaluuksien lisäksi tunnistautumistietojen turvallisuus heikkenee niiden ollessa samankaltaisia. Jos hyökkääjä saa hallintaansa yhden tunnistautumistietoparin, on riski muiden samalta pohjalta luotujen tietojen murtamisen onnistumiseen suuri.

4. Identiteettiperustainen tunnistautuminen: biometriikka

Biometrinen tunnistaminen on osa identiteettiin perustuvaa käyttäjätodennuksen menetelmää, jossa tunnistamisessa hyödynnetään ihmisruumiin ainutlaatuisia piirteitä. Suomen Standardisoimisliiton mukaan biometriikalla terminä tarkoitetaan henkilön fysiologisten ja käyttäytymiseen liittyvien ominaisuuksien mittaamista. Biometriikka käsittää erityisesti tunnistuksen tapahtuman henkilön ominaispiirteiden avulla, kun taas biometria-termi tarkoittaa perinteisesti biologista mittaamista. Biometrinen tunnistaminen toteutetaan vertailemalla henkilön antaman hakutunnisteen ja dokumentissa tai tietokannassa olevan vertailutunnisteen yhteneväisyyttä. Biometrisessa mittauksessa käytettyihin fysiologisiin piirteisiin kuuluvat muun muassa sormenjäljet, henkilön kasvot sekä

silmän iiris. Käyttäytymiseen liittyviin ominaisuuksiin sisältyvät opitut tavat, kuten esimerkiksi allekirjoitus tai käynti. Puhenäytteen avulla tunnistaminen luetaan myös käyttäytymiseen liittyviin ominaisuuksiin, koska se on opetellun toiminnan tulos huolimatta siitä, että henkilön puhe-elimistöllä, eli fysiologisella ominaisuudella, on suuri vaikutus puhenäytteen ominaisuuksiin. Tältä näkökulmalta jokaisen biometrisen tunnistustavan voidaan nähdä riippuvan henkilön fysiologisista ominaisuuksista. Selvän monitulkinnallisuuden sekä määritelmän rajojen hämärtyvyyden takia O'Gorman [2003] ehdottaa määrittelytapaa, jossa ei luokitella tunnistustapoja fyysisten ja käytöksellisten ominaisuuksien avulla. Sen sijaan, että määrittelisimme keinot keräystapojen avulla, ehdottaa O'Gorman määrittelyn tapahtuvan mittauksesta saatavan signaalin perusteella. O'Gormanin tavassa jako tehdään *pysyvien* (stable) ja *dynaamisten* (alterable) *biometristen signaalien* (biometric signal) välille. Kategorioiden sisältö ei muutu paljoa verrattuna fyysisten ja käytöksellisten ominaisuuksien väliseen jakoon, mutta joukosta käytetty termi on täsmällisempi ja tilanteeseen sopivampi. Tällä hetkellä kirjallisuudessa fyysisiin ja käytöksellisiin piirteisiin pohjautuva jako on yleisesti käytetympi, mutta tässä tutkielmassa käsitellään pysyviä ja dynaamisia biometrisiä ominaisuuksia.

Pysyvä biometrinen näyte saadaan suhteellisen pysyvästä ihmisen ominaisuudesta. Muuttumista voi kuitenkin tapahtua esimerkiksi ikääntymisen myötä tai jos henkilölle sattuu jokin fysiologiaa muuttava tapahtuma, kuten sairastuminen tai vammautuminen tai hänelle tehdään esimerkiksi plastiikkakirurginen toimenpide. Kategoriaan kuuluu esimerkiksi sormenjäljet ja ihmisen kasvot. Dynaamisiin biometrisiin ominaisuuksiin taas kuuluu ominaisuudet, joiden tunnusmerkit ovat suhteellisen pysyviä, mutta kaksi eri hetkellä kerättyä signaalia ominaisuudesta eivät ole tismalleen samanlaiset. Pysyvien ja dynaamisien biometristen ominaisuuksien ero voidaankin huomata parhaiten niiden keräystavassa.

4.1. Biometristen tunnisteiden keräys, tallennus ja vertailu

Biometristen näytteiden (biometric sample) käyttöönotto vaatii muiden tunnistautumistekniikoiden kaltaisesti ensimmäisenä tunnisteen luonnin ja tallennuksen. Tämä toteutetaan biometriikassa vaiheessa, jota kutsutaan *biometriseksi rekisteröinniksi* (biometric enrolment). Näytteen luku tehdään jokaiselle tunnisteelle tyypillisellä sensorilla. Biometrinen järjestelmä tunnistaa kerätystä näytteestä yksilöivät piirteet, joista muodostetaan automaattisesti arvioimalla tai erottamalla luokitteita. Luokitteet mahdollistavat henkilöiden identiteetin vertailun, koska kahdella eri yksilöllä ei ole täysin samanlaista luokitteiden yhdistelmää. Luokitteista muodostetaan henkilön tunnistamiseen tarvittava malli,

jota kutsutaan *biometriseksi mallineeksi* (biometric template). Biometrinen malline tallennetaan digitalisoidussa muodossaan varsinaisen biometrisen näytteen sijaan. Edellä esiteltyä tapahtumaa kutsutaan *biometriseksi taltiontiprosessiksi* (biometric acquisition process). Biometrinen rekisteröinti sisältää taltiointiprosessin lisäksi myös erilaisia hallinnollisia prosesseja. [Metsämäki 2016].

Rekisteröinnin jälkeen biometrinen tieto on käytettävissä tunnistamisen sekä todentamisen prosesseissa. Tunnistettaessa henkilön kerättyä biotunnistetta, eli hakutunnistetta, vertaillaan useamman kuin yhden henkilön biometrisiin vertailutunnisteisiin, joita voidaan säilyttää esimerkiksi tietokannassa. Kyseessä on siis seulontavertailu, jotta saadaan vertailutuloksia esimerkiksi siitä, löytyykö työntekijän sormenjälki tietokannasta, eli onko hän kyseisen työpaikan työntekijä. Todennuksessa käyttäjän biometristä hakutunnistetta verrataan taas ainoastaan yhteen vertailutunnisteeseen. Prosessissa pyritään yhteen vertailutulokseen, eli tietoon siitä ovatko tunnisteiden omistajat yksi ja sama henkilö. Sormenjälkitodennusta voidaan käyttää esimerkiksi matkapuhelimen lukituksen poistossa pääsykoodin tilalla.

Pysyvien ja dynaamisien tunnisteiden keräys tapahtuu peruspiirteiltään yhteneväisin keinoin ja saaduista mallineista verrataan tiettyjä ominaisuuksia vertailutunnisteeseen. Esimerkiksi sormenjälkitunnistuksessa luokittavia piirteitä, joiden mukaan malline ensinnäkin luodaan ja vertailu suoritetaan, ovat kaaret, silmukat ja kierteet. Sormenjälkien tunnistus tapahtuu vertailemalla mallineesta joko *yksityiskohtia* (minutiae) tai *kuvioita* (pattern) vertailutunnisteeseen [Metsämäki 2016]. Kun pysyvien biometrisien tunnisteiden rekisteröinti ja vertailu onnistuu ominaisuudesta itsestään, dynaamisien näytteiden keräykseen tarvitaan myös näytteelle tyypillinen muuttuja. Jos käyttäjältä kerätään esimerkiksi puhenäyte, täytyy käyttäjälle tarjota myös puheen sisältö. Tästä ominaisuudestaan johtuen dynaamiset biometriset näytteet tarjoavat mahdollisuuden kahdenlaiseen tunnistamiseen: näytettä voidaan verrata sen mallineen perusteella tai vertailussa voidaan käyttää hyödyksi myös muuttujan sisältöä. [O'Gorman 2003]. Esimerkiksi puhenäytteen sisällöksi voidaan vaatia salasana tai tunnuslause, jonka avulla tunnistautumisesta tulee kaksivaiheinen tunnistautuminen, sen sisältäessä myös tietoperäisen tunnistautumisnäytteen samalla kun puhenäytteen luokitteita verrataan tallennettuun mallineeseen.

4.2. Biometriikan käytettävyys, tietoturvallisuus ja muut haasteet

Biometriikalla on tunnistautumiskeinona käytettävyyden suhteen paljon etuja, jotka puuttuvat salasanoilta ja esineperustaiselta tunnistautumiselta. Biometristä ominaisuutta ei voi unohtaa salasanan tavoin, yhdellä henkilöllä on ainoastaan yksi tunnistautumisnäyte, se pysyy aina mukana ja parhaassa tapauksessa

on aina käytettävissä. Sitä on myös hankala varastaa esineperustaisiin tunnistautumismenetelmiin verrattuna. Tästä huolimatta kaikki nämä vahvuudet voidaan kuitenkin nähdä myös heikkouksina. O'Gorman [2003] huomauttaa biotunnisteen olevan kestävä, jokaiselle henkilölle tunnusomainen ja helposti erottuva, mutta erittäin heikosti kaikenlaisista uhkista palautuva. Jos biotunniste varastetaan, sitä ei ole mahdollista korvata uudella. Tunnisteen vahva yksilöllisyys voi myös johtaa tilanteisiin, joissa henkilö joutuu tunnistetuksi ilman suostumustaan. Tästä esimerkkinä voidaan mainita tilanne, jossa julkisen liikenteen palveluissa, esimerkiksi metroissa, hyödynnetään kasvojen tunnistusmenetelmää. Jos tiedot joutuvat väärinkäytöksien kohteeksi, on yksityishenkilön henkilökohtaiset sijaintitiedot sekä kulkurutiinit hyökkääjien saatavilla. Toisaalta samaa menetelmää voitaisiin hyödyntää rikollisten tehokkaammassa etsinnässä. Tämän seikan takia biotunnisteita käsittelevässä keskustelussa on usein vahvasti esillä myös eettinen puoli, mitä muissa tunnistautumismenetelmissä ei jouduta käsittelemään. Biometriikan vahvasti yksilöivän ominaisuuden ja peruuttamattoman luonteen takia, käyttäjien oikeuksien pohtiminen ja huomioonottaminen on erityisen tärkeää. Kenenkään biometristä tunnistetta ei tulisi kerätä ilman henkilön suostumusta ja käyttäjillä pitäisi olla mahdollisuus poistaa tunniste käytöstä heidän niin halutessaan. Biometrinen tunnistaminen voi olla myös erityisryhmiä syrjivä, esimerkiksi vammautuneiden henkilöiden suhteen, ja sen voidaan nähdä riskeeraavan ihmisarvon kunnioittamisen periaatteen, kun ihmisen ominaisuus muutetaan koneella luettavaksi välineeksi, johon ihmisellä ei välttämättä ole täysimittaista hallintaa [Korja 2016]. Tästä johtuen biometrisen tunnistamisen käyttöön tulisi olla perusteltu tarve, eikä sitä tulisi käyttää pelkän tehokkuuden ja helppouden nimissä.

Eettisten ongelmien lisäksi biometrisessä tunnistuksessa käytetyt laitteistot ovat kalliimpia ja aiheuttavat suurempia ylläpitokustannuksia kuin muiden tunnistautumistapojen laitteistot. Ne ovat myös usein alttiimpia virheille, mikä vähentää tunnistautumismenetelmän käytettävyyttä. Biometrisen näytteen keräämiseen tarvitaan laadultaan hyvä sensori, joka voi usein olla alttiina esimerkiksi ympäristön muuttuville olosuhteille, kuten lämpötilojen ja valonmäärän vaihteluihin tai kosteuteen. Esimerkiksi sormenjälkitunnistuksessa märkä sormi voi estää tunnistuksen onnistumisen. Biometrinen rekisteröinti on erittäin tärkeä vaihe todentumis- ja tunnistusprosessien osalta: jos alun perin rekisteröity vertailutunniste on kerätty puutteellisesti tai se on jollain tavalla epäonnistunut, voi jatkossa tapahtuva tunnistus antaa negatiivisen tuloksen, vaikka kyseessä olisi sama henkilö. Rekisteröintivaiheessa käyttäjän tulisi pystyä myös esittämään henkilöllisyytensä aukottomasti. Ensitunnistusta pidetään vahvana, jos käyttäjä joutuu esittämään henkilökohtaisesti ja kasvotusten luotettavan

lähdeasiakirjan (breeder document), joihin kuuluu Suomessa pääasiallisesti jokin poliisin myöntämä henkilöllisyystodistus, kuten passi. [Metsämäki 2016].

Biometristä tunnistautumista voidaan kuitenkin pitää oikein käytettynä erittäin tietoturvallisena tunnistautumistapana. Vaikka biotunnisteiden väärentäminen on mahdollista ja se luultavasti tulee helpottumaan laitteistojen kehittyessä, on biotunnisteiden keräys- ja käsittelytapa tietoturvallisuutta lisäävä. Kohdassa 4.2. esiteltiin biotunnisteen keräys, jossa kerrottiin kuinka biometrisen ominaisuuden yksilöllisistä piirteistä muodostetaan malline, joka muutetaan digitaaliseen muotoon. Mallineen digitaalisesta muodosta ei ole mahdollista luoda enää uudestaan alkuperäistä sormenjälkeä, joka lisää biotunnisteiden käyttöturvallisuutta, varsinkin jos mallinetiedosto salataan ennen tietokantaan tallentamista. Joitakin biotunnisteita on myös erityisen hankala väärentää niiden ominaisuuksien takia. Esimerkiksi ihmisen silmän iiris, sisältää viisi kertaa enemmän yksilöiviä piirteitä kuin sormenjälki ja keskellä silmää sijaitsevan pupillin kokoon vaikuttaa ympäristön valonmäärä. Tämä asettaa väärentäjälle suuren haasteen kun väärennöksen pitäisi vastata luontaisen iiriksen kaltaisesti ympäristön valoisuuteen, eikä väärennös voi olla yksi liikkumaton malline.

5. Esineperustainen tunnistautuminen

Esineperustaisessa tunnistautumismenetelmässä käyttäjä hyödyntää tunnistautumisessa jotain fyysistä esinettä eli *tunnistusvälinettä* (token) [O'Gorman 2003]. Esineen avulla voidaan joko suorittaa koko tunnistautuminen tai sitä voidaan käyttää apuvälineenä tunnistautumisprosessissa. Tunnistusvälinettä voidaan käyttää neljällä eri tavalla: pysyvänä salasanan sisältävänä tunnistusvälineenä (static password token), tahdistavana salasanan tarjoavana tunnistusvälineenä (synchronous dynamic password token), itsetahdistavana tunnistusvälineenä (asynchronous password token) ja haastemenetelmää hyödyntävänä tunnistusvälineenä (challenge response token). Pysyvä tunnistusväline sisältää itsessään käyttäjän tarvitseman ”salasanan” eli keinon, jonka avulla henkilöllisyys voidaan todentaa. Tämänkaltaisia tunnistusvälineitä ovat esimerkiksi pankkikortit, kaukosäätimet autotallin ovelle tai älykortit. Tahdistavissa, itsetahdistavissa ja haastemenetelmällisissä versioissa tunnistusväline on aktiivinen laite, joka tuottaa ja tarjoaa käyttäjälle *kertasalasanoja* (one-time passcode). Tahdistetussa versiossa laite luo uuden kertasalasanan tasaisin väliajoin yhtäaikaisesti masterlaitteen kanssa ja todentuminen palveluun onnistuu ainoastaan syöttämällä kyseisellä hetkellä tunnistuslaitteen tarjoaman salasanan. Itsetahdistetussa menetelmässä uuden kertasalasanan luonti ei ole aikaan sidottu, vaan uusi salasana luodaan, kun käyttäjä ilmoittaa jollain keinolla, esimerkiksi napin painalluksella, tarpeen uudelle salasanalle. Haastemenetelmää hyödyntäviä tunnistusvä-

lineitä voidaan käyttää esimerkiksi niin, että käyttäjä ilmoittaa halustaan kirjautua järjestelmään ja järjestelmä antaa käyttäjälle niin sanotun haastekoodin, jonka käyttäjä syöttää omaan tunnistusvälineeseensä, joka laskee esimerkiksi tietyn algoritmin avulla vastauksen koodiin. Aikaansaatu koodi ilmoitetaan käyttäjälle, joka syöttää koodin järjestelmään, ja sen ollessa oikea saa näin todennettua oikeutensa.

Esineperustaista tunnistautumista ei yleensä käytetä yksinään todentautumiseen, vaan se on usein toinen kaksivaiheiseseen todennukseen kuuluvista menetelmistä [O'Gorman 2003]. Esimerkiksi pankkikortti siruineen on esineperustainen tunnistautumisväline, mutta siihen yhdistetään useimmiten PIN-koodi, jotta sen todentautumisturvallisuutta saataisiin vahvistettua. Tunnistusvälineiden ei nähdä olevan tarpeeksi tietoturvallisia toimiakseen yksinään, mikä johtuu suurimmaksi osaksi yhdestä niitä määrittävästä piirteestä: tunnistusvälineitä on niiden konkreettisen fyysisen olemassa olon takia yleensä helpompi varastaa verrattuna muihin todentautumiskeinoihin, mikä yksinkertaistaisi niiden väärinkäyttöä, jos niihin ei olisi liitetty toista todentautumismenetelmää. Tunnistusvälineitä ei ole yhtä helppo saada massoittain haltuun kuten salasanoja, mutta jos tunnistusvälinettä käytettäisiin yksinään tietoturvallisesti paljon vaativissa tapauksissa, kuten esimerkiksi pankkikortissa, olisi yksittäiset täsmälliset hyökkäykset kertahyödyltään kannattavia suhteutettuna hyökkääjälle aiheutuneisiin kuluihin. Toisaalta tämä vaatisi kuitenkin hyökkääjältä fyysistä läsnäoloa, mikä lisää kiinnijäämisen riskiä. Tunnistusvälineitä voidaan suojata tietoturvan lisäämiseksi mahdollisen varkauden tapahtuessa esimerkiksi käyttämällä välineessä erityistä ulkokuorta, joka estää laitteen ulkopuolisen peukaloinnin tai käyttämällä laitteistoa, joka lamauttaa tunnistusominaisuuden jos laitetta yritetään manipuloida tai sen avulla tehdyt kirjautumisyritykset ylittävät rajatun lukumäärän.

Tietoturvallisuusriskien lisäksi esineperustaisilla tunnistautumistekniikoilla on muutamia käytettävyyttä haittaavia puolia. Jos tunnistusvälinettä haluaa käyttää käyttäjätodennuksessa, on esineen oltava aina fyysisesti käyttäjän hallussa kirjautumishetkellä. Tieto- ja identiteettiperustaiset tunnistusmenetelmät taas kulkevat jatkuvasti käyttäjän mukana ilman vaivannäköä. Esineperustainen tunnistautuminen aiheuttaa myös enemmän laitteiden luonti- ja ylläpitokustannuksia, eivätkä esimerkiksi älykortit ole yhtä joustavia käytössään kuten ajasta ja paikasta saatavuudeltaan riippumattomat salasanat. Edellä mainituista haitoista huolimatta tunnistusvälineillä on monia niiden käytettävyyttä lisääviä hyviä puolia: tunnistusvälineen hallinnan menetys on helpompi huomata verrattuna salasanoihin ja biometrisiin tunnistautumismenetelmiin ja se voi sisältää monia eri salasanoja yhtäaikaisesti. Tämän takia esimerkiksi älykortti on

erittäin tehokas kulkukorttina työpaikalla, jossa henkilöllisyys tulee todistaa päästäkseen sisälle esimerkiksi laboratorioon. Kulkukorttia on yksinkertaista ja tehokasta käyttää ja sen käytössä tapahtuu vähemmän virheitä verrattuna biometrisen tunnisteen käyttöön kulunvalvonnassa. Tunnistusvälineen yhdistäminen toiseen tunnistautumismenetelmään on myös yksinkertaista. Esimerkiksi biometristä tunnistautumiskeinon mallinetta voidaan säilyttää muovikorteissa, optisissa korteissa tai älykorteissa, jolloin käyttäjällä on mahdollisuus kuljettaa mallinetta mukanaan tunnistusvälineenä [Valmisteluasiakirja biotunnisteista 2003]. Näin biotunnistetta pystyttäisiin käyttämään passin tavoin henkilöllisyyden varmentamisessa.

6. Yhteenveto ja johtopäätökset

Jokaisessa tässä tutkielmassa esitellyssä käyttäjätodennusmenetelmässä on sille kategorialle tyypillisten ominaisuuksien mukaiset vahvuudet ja heikkoudet. Salasanat ovat perusperiaatteiltaan yksinkertaisia käyttää, niiden käyttömahdollisuudet ovat rajattomat suhteessa aikaan ja paikkaan, eikä niistä aiheudu ylläpitäjille suuria kuluja. Suurin ongelma salasanojen käytössä on niiden käytettävyyshaitat: yhdellä käyttäjällä on monia eri salasanoja hallittavanaan eri palveluihin. Salasanojen suuri lukumäärä käyttäjää kohti johtaa käyttäjien omiin toimenpiteisiin, joilla he pyrkivät hallitsemaan kognitiivista taakkaansa ja aiheuttavat tahattomasti näillä toimenpiteillään salasanojen tietoturvallisuuden heikkenemisen. Biometrinen tunnistautuminen päinvastoin ei aiheuta käyttäjille muistamisen taakkaa. Jokaisella on ainoastaan yksi uniikki todentautumisväline, joka kulkee käyttäjän mukana jatkuvasti. Biotunnistetta on hankalampi varastaa tai väärentää muihin todennusmenetelmiin nähden, mutta niiden monimutkaisempaan varastettavuuteen ei kannata luottaa liiaksi: biotunnisteita väärennetään ja tullaan väärentämään myös tulevaisuudessa, minkä takia pysyviä biotunnisteita ei tulisi käyttää yksinäisinä todentaumisvälineinä. Biometriikka myös edellyttää eettistä keskustelua ja aiheuttaa ylläpitäjille suurempia kuluja kuin salasanat.

Esineperustaiset tunnistusmenetelmät mahdollistavat monien eri salasanojen säilytyksen samassa säilössä ilman, että käyttäjän tulisi muistaa ne kaikki. Niiden käyttö on tiettyissä käyttötilanteissa tehokasta ja yksinkertaista, ja niiden vaarantuminen tai katoaminen kokonaan on helpompaa havaita kuin muissa tunnistautumismenetelmissä. Tunnistusvälineet on kuitenkin kohtalaisen helppo varastaa ja niiden väärinkäyttö on yksinkertaista, jos todennukseen ei ole liitetty toisen menetelmän käyttöä vahvistukseksi. Tunnistusvälineiden käytön kulut ovat myös verrattavissa biometrisen tunnistautumisen kuluihin ja

niiden käyttöön vaadittavat laitteistot aiheuttavat rajoitteita menetelmän käyttömahdollisuuksiin.

Tavanomainen ja usein käytetty sanonta turvallisuuden ja käytettävyyden vaihtokaupasta käyttäjätodennuksessa yksinkertaistaa todellisuutta. Tilanne on monimutkaisempi kuin yksiulotteinen tila kahden eri tekijän välillä, ja sen sijaan, että yritettäisiin etsiä yhtä kattavaa ratkaisua kaikkiin ongelmiin, tulisi ponnistukset kääntää jokaiseen yksilölliseen tilanteeseen parhaiten sopivan todennusmenetelmän löytämiseen ja sen käyttöön [Herley and Van Oorschot 2012]. Kun Gates [2004] ennusti salasanojen sukupuuttoa, hän ehdotti tilalle älykortteja. Esineperustainen tunnistauminen ei kuitenkaan vastaa kaikkien niiden tilanteiden tarpeisiin, joissa salasanoja tällä hetkellä käytetään. Käyttäjien liikkuvuuden lisääntyessä jatkuvasti on hankalaa kuvitella tilannetta, jossa käyttäjä kantaisi mukanaan älykortinlukijaa lomamatkallaan sen takia, että pääsee kirjautumaan Facebookiin. Älykortit palvelevat käyttäjää hyvin tilanteissa, joissa hän joutuu tunnistautumaan moneen eri järjestelmään lyhyen ajan sisällä, kuten esimerkiksi työpaikalla, mutta ne eivät ole sopivia korvaamaan salasanojen käyttöä jokaisessa tilanteessa.

Jatkuva pyrkimys pois kaikesta salasanojen käytöstä on hidastanut todennusmenetelmän kehitystä ja ratkaisujen etsimistä niihin ongelmiin, joiden takia salasanojen koetaan olevan turvaton todennusmenetelmä. Monet käyttäjien kokemista haasteista salasanojen käytössä, johtuvat heidän puutteellisista tiedostaan ja virheellisistä käsityksistä. Käyttäjillä ei useimmiten ole syvällistä ymmärrystä siitä, millainen on tietoturvallinen salasana tai tietoturvallisesti heikon salasanan käytön aiheuttamista uhkista. Käyttäjät noudattavat turvallisuusohjeita mieluummin, jos uhkat on tehty heille näkyviksi ja tiettäviksi [Adams and Sasse 1999], mutta tällä hetkellä salasanojen käyttöympäristö ei ole käyttäjille motivoiva. Kaikkia salasanoihin liittyviä tietoturvariskejä ei voida eliminoida pelkästään käyttäjiä informoimalla, mutta osasta päästäisiin eroon käyttäjien opastuksella. Sen sijaan, että käyttäjät pakotetaan valitsemaan tietoturvalliset salasanat valmiiksi asetetuilla merkkivaatimuksilla ilman ilmoitettuja syitä, voitaisiin käyttäjille antaa esimerkkitapauksia, kuinka salasanan sisältö vaikuttaa murtoaikaan, kertoa mitä seurauksia salasanan murtamisesta voi koitua sekä antaa tarkkoja ohjeita minkälaiset merkit voivat kertoa sivuston olevan aidon sivun sijaan kalastelusivusto [Herley and Van Oorschot 2012].

Käyttäjätodennusmenetelmäksi tulisi siis valita järjestelmän tarkoitusta ja ominaisuuksia parhaiten palveleva keino. Kaikilla järjestelmillä ei ole samanlaiset vaatimukset tietoturvallisuutta kohtaan (vrt. ilmaissähköpostitili ja potilastietokanta), eikä niitä käytetä samanlaisissa tilanteissa samanlaisiin tarkoituksiin. Varsinkin tietoturvallisuutta paljon vaativien järjestelmien käyttätoden-

nuksessa olisi tarpeellista käyttää kaksivaiheista todennusta [O'Gorman 2003]. Tietoturvallisuuteen pyrkiessä ei tule unohtaa käytettävyyden tärkeyttä: käytettävyyden kärsiessä käyttäjät ryhtyvät toimenpiteisiin, jotka voivat heikentää tietoturvallisuutta merkittävästi [Adams and Sasse 1999]. Vaikka käytössä olisi kuinka vahva todennusmenetelmä, jos sitä käytetään väärin, tietoturvallisuus voi heikentyä mittavasti.

Viiteluettelo

Anne Adams and Martina Sasse. 1999. Users are not the enemy. *Communications of the ACM* 42, 12, 40-46.

Andrew Butterfield and Gerard Ekembe Ngondi. 2016. *A Dictionary of Computer Science (7 ed.)*. Oxford Unversity Press.

Katha Chanda. 2016. Password Security: An Analysis of Password Strengths and Vulnerabilities. *I. J. Computer Network and Information Security* 8, 7, 23-30.

Rachna Dhamija and Lisa Dusseault. 2008. The seven flaws of identity management: usability and security challenges. *IEEE Security & Privacy* 6, 2, 24-29.

William Henry "Bill" Gates III. 2004. Headspeech. RSA Conference.

Lawrence O'Gorman. 2003. Comparing Passwords, Tokens, and Biometrics for User Authentication. In: *Proceedings of the IEEE* 91, 12, 2021-2040.

Kirsi Helkala and Tone Hoddo Bakås. 2014. Extended results of Norwegian password security survey. *Information Management & Computer Security*. 346-357.

Mika Hakala, Mika Vainio ja Olli Vuorinen. 2006. *Tietoturvallisuuden käsikirja*. Docendo.

Cormac Herley and Dinei Florencio. 2007. A large-scale study of web password habits. In: *Proceedings of the 16th International Conference on World Wide Web*, 657-666

Cormac Herley and Paul Van Oorschot. 2012. A Research agenda acknowledging the persistence of passwords. *IEEE Security & Privacy* 10, 1, 28-36.

Juhani Korja. 2016. *Biometrinen tunnistaminen ja henkilötietojensuoja. Tutkimus biometristen tunnisteiden lainsäädännöllisestä asemasta.* Väitöskirja, Oikeustieteiden tiedekunta, Lapin yliopisto.

Markku Metsämäki. 2016. Biometriikka Kalvosarja oppilaitoksille. Suomen Standardisoimisliitto SFS ry.

Jakob Nielsen. 1993. *Usability Engineering*. Academic Press.

Miniwatts Marketing Group. 2017. Internet World Stats. http://www.internetworldstats.com/stats.htm. Checked 8.4.2017.

Saila Ovaska, Anne Aula ja Päivi Majaranta. 2005. *Käytettävyystutkimuksen menetelmät*. Raportti B-2005-1. Tietojenkäsittelytieteiden laitos, Tampereen yliopisto.

R. Shirey. 2000. Internet Security Glossary. https://www.ietf.org/rfc/rfc2828.txt. Checked 5.4.2017.

Suomen Standardisoimisliitto SFS ry. Biometriikka. https://www.sfs.fi/ standardien_laadinta/sfs_n_tekniset_komiteat_ja_seurantaryhmat/it-standardisointi/it_-_aihealueet/biometriikka Checked 30.4.2017.

Valmisteluasiakirja biotunnisteista. 29 artiklan mukainen tietosuojatyöryhmä. Hyväksytty 1.8.2003. Luettu 20.5.2017

C. D. Wickens. 1992. *Engineering Psychology and Human Performance.* Harper Collins.

Jeff Yan, Alan Blackwell, Ross Anderson and Alasdair Grant. 2004. Password memorability and security: Empirical results. *IEEE Security & Privacy Magazine* 2, 5, 25-31.