Puukartat

Käyttöliittymätutkimus – seminaari

Ossi Ahomaa (Ossi.Ahomaa@helsinki.fi)

Helsingin Yliopisto

Tietojenkäsittelytieteen laitos

Helsinki 26.10.2001

Puukartat

Ossi Ahomaa

Seminaariesitelmä
Tietojenkäsittelytieteen laitos
Helsingin Yliopisto
26. lokakuuta 2001, 10 sivua

Puukartat (treemap) on kehitetty hierarkkisen tiedon esittämiseen rajallisella näyttöalueella. Puukarttojen esittämiseen on esitelty monenlaisia algoritmeja, joilla on yritetty ratkaista tuloksen havainnollisuuteen ja helppokäyttöisyyteen liittyviä ongelmia.

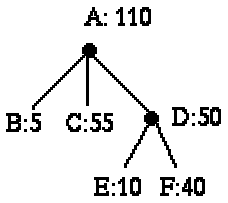
Puukarttojen kehitys lähti liikkeelle kovalevyn sisällön kuvaamisesta. Puukartat ovat kuitenkin yleinen kuvaamistapa, jota voidaan käyttää melkein minkä tahansa hierarkkisen rakenteen sisällön kuvaamiseen kaksiulotteisesti.

Aiheluokat (ACM Computing Reviews 1998): H.5.2

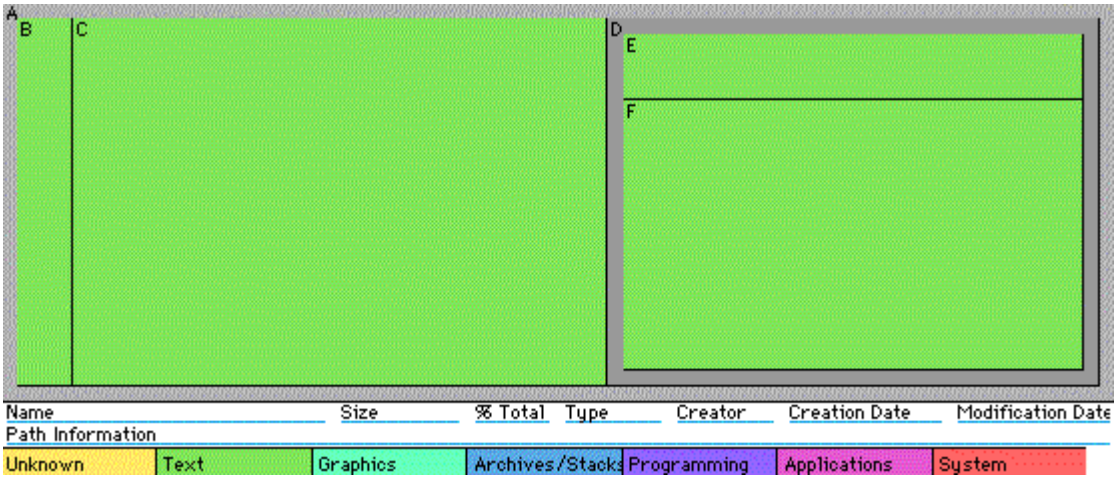
Avainsanat: visualisointi, puukartta, treemap

1 Johdanto

Puukartta (treemap) on tapa esittää puurakenne kaksiulotteisesti. Puukartat muodostuvat suorakulmioista, joiden pinta-ala kuvaa puun alkion painoa. Kaikki yhden alkion jälkeläiset ovat esi-isänsä sisällä. Esimerkiksi puu (kuva 1) Esitettäisiin puukarttana seuraavasti (kuva 2). Puukartta on tehokas tapa esittää suuria puita rajoitetulla alueella, kuten tietokoneen näytöllä [BHW2000].



Kuva 1. Puu



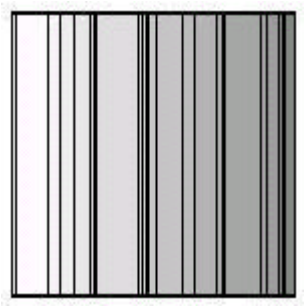
Kuva 2. Puukartta

Alkuperäisen puukartta-visualisointitekniikan kehitti Ben Shneiderman vuonna 1990. Tuolloin hän huomasi tarvitsevansa paremman visualisoinnin tietokoneensa kovalevyn sisällöstä kuin mihin perinteiset puurakenteet pystyivät. Shneidermanin alkuperäisessä ideassa suorakulmioiden suunnat muuttuvat jokaisella puun tasolla, jolloin tietyn polun seuraaminen on helpompaa. Korkeat ja kapeat ja toisaalta leveät ja matalat suorakulmiot aiheuttivat kuitenkin myös ongelmia alueiden merkitsemisessä ja niiden pinta-alojen vertailussa [BHW2000].

Puukarttojen esittämiseen on esitelty monenlaisia algoritmeja, joilla on yritetty ratkaista tuloksen havainnollisuuteen ja helppokäyttöisyyteen liittyviä ongelmia. Yleistäen voidaan sanoa, että havainnollisuuden lisääminen ja algoritmin tuottaman tuloksen reagointi muuttuvaan tietoon ovat vastakkaiset ominaisuudet. Toisin sanoen toisen ominaisuuden lisääminen vähentää toista [SW2001].

2 Puukartta-algoritmit

Alkuperäistä puukartta-algoritmia kutsutaan nimellä slice-and-dice. Siinä sisäkkäisten suorakulmioiden sisältö on jaettu aina joko pysty- tai vaakasuoriin osiin. Jos edeltävä taso oli pystysuora, on sen sisällä oleva osio vaakasuora ja toisin päin. Tämä algoritmi tuottaa tyypillisesti suorakulmioita, joiden sivusuhde on suuri (kuva 3) [BHW2000].



Kuva 3. Slice-and-dice ulkoasu

Tässä yhteydessä sivusuhde on max[korkeus/pituus, pituus/korkeus]. Neliön sivusuhde on siis 1, ja muilla suorakulmioilla sitä suurempi, mitä enemmän korkeus ja leveys eroavat toisistaan. Pitkät ja kapeat sekä lyhyet ja leveät suorakulmiot eivät kuitenkaan ole parhaita mahdollisia visualisointitarkoitukseen, sillä niitä on vaikea merkitä, erottaa ja valita ruudulta. Lisäksi niiden pinta-aloja on vaikea verrata toisiinsa. [BHW2000].

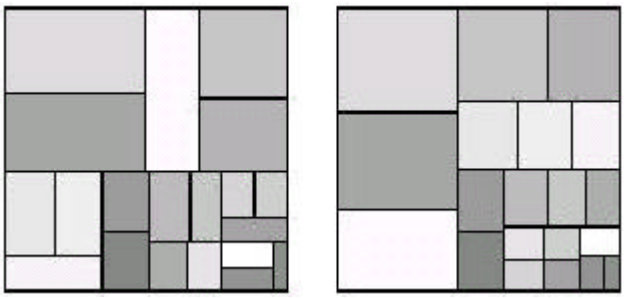
Puukartta-algoritmien vertailuun voidaan käyttää kolmea mittaa [BHW2000]:

- Keskimääräinen sivusuhde
- Ulkoasun muutos tiedon muuttuessa
- Luettavuus

Paras mahdollinen ulkoasu olisi sellainen, jossa keskimääräinen sivusuhde olisi mahdollisimman lähellä yhtä, ja esitettävän tiedon muuttuessa ulkoasu muuttuisi mahdollisimman vähän. Lisäksi tietyn kohteen etsiminen tulisi olla mahdollisimman helppoa ja nopeaa [SW2001].

2.1 Alhaisen sivusuhteen puukartat

Slice-and-dice algoritmista on kehitelty useita muunnoksia. Niiden avulla on pyritty ratkaisemaan puukarttojen esittämiseen liittyviä ongelmia. Sivusuhteen saamiseksi mahdollisimman alhaiseksi on kehitetty kahdentyyppisiä algoritmeja. Ryvästetyt ja neliöidyt puukartat [BHW2000] minimoivat sivusuhdetta jakamalla alueita sopivasti toistensa kanssa samankokoisten kanssa samaan ryhmään (kuva 4).



Kuva 4. Ryvästetty ja neliöity puukartta

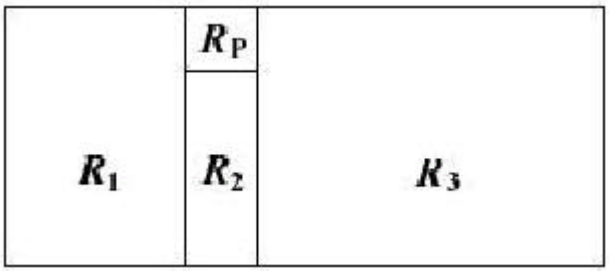
Näilläkin parannuksilla on kuitenkin haittansa. Jos esitettävä data muuttuu vaikkapa vain yhden alkion osalta, saattaa koko puukartan ulkoasu muuttua merkittävästikin. Erityisen haitallinen piirre tämä on järjestelmissä, joiden esittämä data muuttuu nopeassa tahdissa. Tällöin järjestelmän käyttäjä joutuu etsimään kohteen uudestaan aina uuden ulkoasun tullessa näytölle. Toinen alhaisen sivusuhteen puukarttojen huono puoli on se, että ne eivät säilytä järjestystä. Vaikka esitettävä data olisi selvästi järjestettyä esim. ajan mukaan, nämä algoritmit saattavat hajottaa peräkkäiset alkiot eri puolille ulkoasua [SW2001].

2.2 Järjestetyt puukartat

Järjestetyt puukartat on kehitetty parantamaan puukarttojen tuottamaa ulkoasua vastaamaan paremmin lähdeaineistoa. Niiden tavoitteena on saada järjestetyssä datassa peräkkäin olevat alkiot asteltua ulkoasussa vierekkäin. Vaikka tämä tapa ei ole yhtä täsmällinen järjestyksen suhteen kuin alkuperäinen slice-and-dice, se kuitenkin antaa hyviä johtolankoja siitä, missä päin tietty alkio voisi olla [SW2001].

Seuraavassa on esitetty järjestetyn puukartan luontialgoritmi. Syötteenään algoritmi saa suorakulmion R, ja alkiolistan, jonka alkiot on järjestetty indeksin mukaan ja niillä on tietty koko. Algoritmi valitsee jakoalkion, joka sijoitetaan R:n sivulle (kuva 5). Loput

alkiot jaetaan kolmeen muodostuvaan suorakulmioon R1, R2 ja R3. Sama algoritmi toistetaan rekursiivisesti kullekin suorakulmiolle R1 – R3.



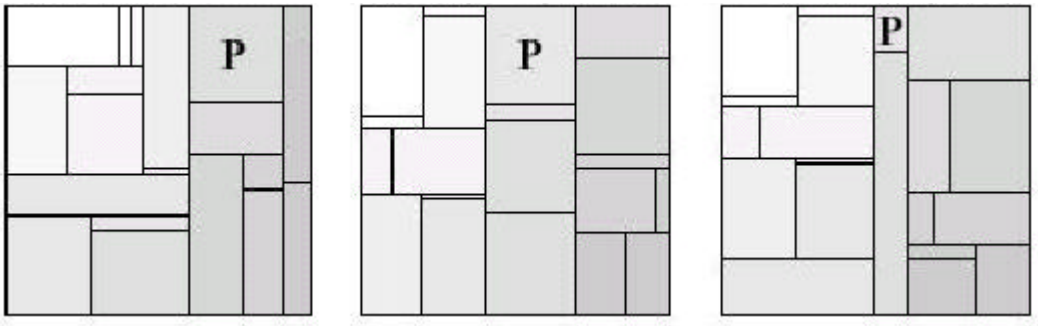
Kuva 5. Jako alueisiin

Algoritmin suoritus loppuu kun suorakulmiossa on neljä alkioita tai vähemmän. Tällöin viimeisten alkioiden asettelu voi tapahtua kolmella eri tavalla: jako(pivot), piha (quad) tai käärme (snake). Pivot jakaa alueen kuten itse algoritmi, pihamallissa muodostuu kaksi aluetta, joissa molemmissa on kaksi tai yksi alkio. Käärme taas jakaa kaikki alkiot joko pysty- tai vaakasuoraan. Eri lopetusehdot toimivat eri tavalla eri tilanteissa, joten on tarpeen laskea minkä lopetusehdon mukaan milloinkin toimitaan.

Nyt järjestetyn puukartan luominen voidaan esittää kuusiaskelisena algoritmina seuraavasti [SW2001]:

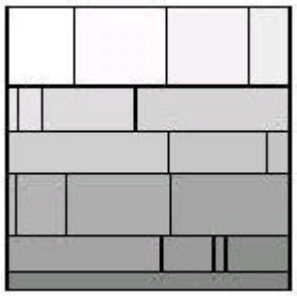
1. Jos alkioiden määrä <= 4, käytä lopetusehtoja ja valitse se, jonka tuottamien sivusuhteiden keskiarvo on pienin. Lopeta.
2. Valitse jakoalkioksi P alkio, jolla on suurin alue.
3. Jos R:n leveys on suurempi tai yhtä suuri kuin korkeus, jaa R neljään suorakulmioon R1, R2, R3 ja Rp kuten (kuvassa 5). Jos korkeus on suurempi kuin leveys, tee samanlainen jako pyöräytettynä suoran x = y mukaan.
4. Laita P suorakulmioon Rp, jonka tarkat mitat saadaan askeleessa 6.
5. Jaa listan alkiot pois lukien P kolmeen listaan L1, L2 ja L3, jotka asetellaan P1, P2 ja P3:een. L1:n alkioiden indeksien on oltava pienempiä kuin P:n indeksin, L2:n indeksit ovat pienempiä kuin L3:n, ja Rp:n jakosuhde on mahdollisimman pieni.
6. Tee askeleet 1-6 kaikille L1-L3 ja R1-R3, jos listat eivät ole tyhjiä.

Jakoalkion valinta vaikuttaa lopulliseen ulkoasuun. Esitetyssä algoritmissa (pivot-by-size) valittiin suurin alkio. Muita vaihtoehtoja ovat esimerkiksi valinnat sen perusteella, mikä alkio jakaa muut alkiot uusiin listoihin määrällisesti (pivot-by-middle) tai pinta-alaltaan (pivot-by-split-size) samankokoisiin osiin (kuva 6) [SW2001].



Kuva 6. pivot-by-size, pivot-by-middle ja pivot-by-split-size

Toinen tapa muodostaa järjestetty puukartta on järjestää alkiot kaistoihin (strips). Itse asiassa kaistoitetut puukartat ovat neliöityjen puukarttojen muunnos. Kaistoitetussa algoritmissa alkiot sijoitetaan järjestyksessä vaaka- tai pystykaistoihin, joiden alkiot ovat yhtä korkeita (jos kaista on vaakasuora) tai leveitä (jos kaista on pystysuora). Sen tuottaman ulkoasun luettavuus on paljon parempi kuin tavallisilla järjestetyillä puukartoilla (kuva 7). Lisäksi sen reagointikyky muuttuvaan dataan ja sivusuhde on verrattavissa muihin järjestettyihin puukarttoihin [BSW2001].



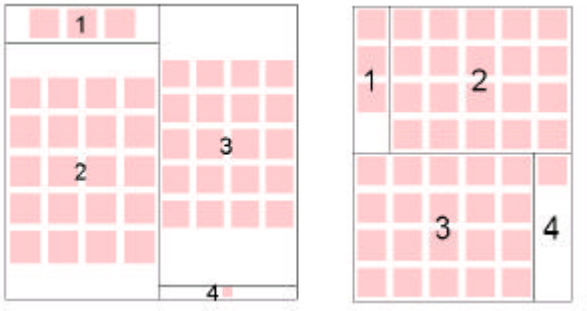
Kuva 7. Kaistoitettu puukartta

Kaistoitettu algoritmi toimii seuraavasti. Aluetta täytetään kaista kerrallaan siten, että ennen uuden alkion lisäämistä kaistaan lasketaan kaistan keskimääräinen sivusuhde. Jos sivusuhde on pienempi kuin ennen lisäämistä, uusi alkio sijoitetaan kyseiseen kaistaan. Jos taas sivusuhde huononee, aloitetaan uusi kaista. Kaistan korkeus tai leveys määräytyy kaistalla olevien alkioiden pinta-aloista [BSW2001].

Kaistoitettu puukartta-algoritmi on itse asiassa neliöidyn puukartta-algoritmin yksinkertaistus. Neliöity muodostaa myös kaistoja, mutta aina siihen suuntaan, jossa sivusuhteiden keskiarvo on mahdollisimman pieni. Kaistoitettu algoritmi tuottaa hiukan huonompia sivusuhteita mutta on järjestetty ja helposti luettava [BSW2001].

2.3 Kvanttipuukartat

Kvanttipuukartat ovat saaneet nimensä siitä, että niiden alkioiden koolla on tietty minimikoko, joka määrää ulkoasun rivien ja sarakkeiden muodon. Kvanttipuukarttojen avulla voidaan esittää paremmin ryhmiä, jotka sisältävät visuaalista tietoa, esimerkiksi valokuvia. Edellä esitetyt algoritmit eivät tuota visuaalisesti helppolukuista ulkoasua esimerkiksi eri hakemistoissa olevien kuvien esittämiseen. Ne muodostaisivat hakemistojen kuvien määrien mukaiset alueet, joihin kuvat sijoitettaisiin. Kuvat eivät kuitenkaan olisi suorissa riveissä, vaan eri alueiden kuvat olisivat eri tavoin aseteltu. Kvanttipuukartat sen sijaan järjestäisivät kuvat siten, että kaikkien ryhmien kuvat ovat järjestetty (kuva 8) [Bederson2001].



Kuva 8. Järjestetty puukartta ja kvanttipuukartta

2.4 Algoritmien vertailua

Puukartta-algoritmien tuottamien ulkoasujen hyvyyden vertailuja on tehty mm. mittaamalla kolmea suuretta satunnaisesta aineistosta [SW2001]. Käytettyjä suureita ovat sivusuhteen keskiarvo, ulkoasun muutos ja luettavuus. Alla olevan taulukko on saatu aineistosta, joka koostuu tasapainoisesta kolmetasoisesta 512 solmun puusta, jonka jokaisella tasolla on kahdeksan solmua. Testissä tehtiin sata koetta, joissa jokaisessa oli sata vaihetta. Alkuperäiset arvot olivat normaalisti jakautuneet, ja joka vaiheessa arvot kerrottiin e^x:llä. X arvottiin normaalijakaumasta, jonka hajonta oli 0.05 ja keskiarvo 0.

Tuloksista voidaan selvästi havaita, että alhaiset sivusuhteet ja hyvä päivittyvyys ovat vastakkaiset ominaisuudet siten, että toisen lisääminen vähentää toista (kuva 9) [BSW2001]. Slice-and-dice:n luettavuus ja muutoksiin reagointikyky ovat omaa luokkaansa, mutta samalla sivusuhde on jopa kymmenen kertaa huonompi kuin muilla algoritmeilla. Toisessa ääripäässä neliöity algoritmi tuottaa todella alhaisia sivusuhteita, mutta toisaalta sen muutos- ja luettavuusarvot ovat slice-and-diceen verrattuna todella huonot. Muut algoritmit sijoittuvat näiden kahden välille. Käytettävä algoritmi onkin valittava sovelluksen käyttötarkoituksen mukaan.

Algorithm	Aspect Ratio	Change	Readability
Slice-and-dice	26.10	0.46	1.0
Pivot-by-middle	3.58	1.21	0.42
Pivot-by-size	3.31	4.14	0.33
Pivot-by-split	3.00	2.37	0.35
Strip	2.83	1.09	0.51
Cluster	1.79	7.67	0.26
Squarified	1.74	8.27	0.26

Kuva 9. Algoritmien vertailua

On todennäköistä, että kaikkia edellä esitettyjä algoritmeja voitaisiin optimoida tuottamaan entistä parempia ulkoasuja. Myös erilaisten algoritmien yhdisteleminen saattaisi parantaa tuloksia. Lisäksi olisi mahdollista etsiä matemaattista mallia, jolla sivusuhteiden ja päivittyvyyden välinen kauppa saataisiin määriteltyä tarkasti. Myös suorakulmioita monimutkaisempien alueiden käyttö voisi olla tarpeen [BSW2001].

3 Sovelluksia

Puukarttojen kehitys lähti liikkeelle kovalevyn sisällön kuvaamisesta [Johnson92]. Useita juuri tähän tarkoitukseen tehtyjä sovelluksia onkin esitelty ja saatu jopa kaupalliseen levitykseen (http://www.miclog.com/dm/1/desc1.htm). Puukartat ovat kuitenkin yleinen kuvaamistapa, jota voidaan käyttää melkein minkä tahansa hierarkkisen rakenteen sisällön kuvaamiseen. Muita esimerkkejä puukarttojen käytöstä ovat mm:

- Osakesalkun seuranta [Wattenberg99]
- Verkkojen hallinta
- Kuvien selaaminen (PhotoMesa)
- Koripallotilastointi
- Analyyttisen päättelymallin kuvaaminen
- Satelliittien hallinta
- Lähdekoodin visualisointi
- Hypertekstin hallinta

4 Yhteenveto

Nykyiset tietojärjestelmät käsittelevät suuria määriä erilaista tietoa. Tiedon visualisointi on erittäin tärkeää, koska tärkeän tiedon löytäminen suuresta tietomassasta voi olla vaikeaa. Puukartat ovat hyvä apu suuren tietomäärän kuvaamisessa, sillä ulkoasu vie vakiomäärän tilaa.

Puukarttoja voidaan käyttää minkä tahansa hierarkkisen tiedon kaksiulotteiseen esittämiseen. Eri puukartta-algoritmeilla on erilaisia ominaisuuksia. Hyvä luettavuus, pieni sivusuhde ja sujuva muuttuvaan tietoon reagointi eivät toteudu samalla algoritmilla. Algoritmeissa on vielä paljon kehitettävää, ja tarkka matemaattinen malli optimaalisen puukartta-algoritmin kehittämiseksi puuttuu.

5 Lähdeluettelo

[Bederson2001]
Bederson, B.B.
Quantum treemaps and Bubblemaps for Zoomable Image Browser.
Proceedings of User Interface and Software Tecnology (UIST 2001) ACM Press

[BHW2000]
Bruls, M., Huizing, K., &van Wijk, J. J.
Squarified treemaps.
In Proceedings of Join Eurographics and IEEE TCVG Symposium on Visualization (TCGV 2000) IEEE Press, pp. 33-42

[BSW2001]
Bederson, B., Shneiderman, B., Wattenberg, M.,
Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies.
University of Maryland, CS-TR-4277 (2001).

[Johnson92]
Brian Johnson.
TreeViz: Treemap visualization of hierarchically structured information.
CHI Conference Proceedings: Striking a Balance, pages 369-370 (1992).

[SW2001]
Shneiderman, B., & Wattenberg, M.
Ordered Treemap Layouts
Tech Report CS-TR-4237, Computer Science Dept., University of Maryland, College Park, MD (2001).

[Wattenberg99]
Wattenberg, M.
Visualizing the Stock Market
In Proceedings of Extended Abstracts of Human Factors in Computing systems (CHI 99) ACM Pre