Teknillinen Korkeakoulu
Informaatio- ja luonnontieteiden tiedekunta

Mat-2.4177 Operaatiotutkimuksen projektityöseminaari: 4. Puustokuvion korjuukelpoisuus ja saavutettavuusanalyysi

Koljonen, Juha-Matti
Matikainen, Juho
Nurmentaus, Joni
Paatsama, Matti
Valvanne, Juha
Östring, Lasse

Projektityö
Espoo, 20. huhtikuuta 2009

Sisältö

Luku 1

Johdanto

1.1 Projektityön tausta

Tietokoneiden laskentakapasiteetin kasvu ja parantuneet mittaustavat ovat avanneet uusia mahdollisuuksia. Suurelle metsäyhtiölle mielenkiintoista on uuden laserkeilausmenetelmän kehittyminen. Laserkeilauksessa lennetään alueen yli ja kaikuja kuuntelemalla voidaan päätellä pinnankorkeus ja pinnan päällä olevan aineksen tiheys. Nykyisin metsävarat kartoitetaan pääosin ammattilaisten silmin metsäalueella kävellen. Tulevaisuudessa tähdätään siihen, ettei metsäpalstalle olisi tarpeellista mennä muuten kuin tarkistamaan aiemmin muodostettuja tuloksia. Tähän mennessä aineistosta on jo kyetty määrittämään puustokuvioita eli pääpuulajiltaan ja puuston tiheydeltään samankaltaisia yhtenäisiä alueita. Lisäksi näille puustokuvioille on pystytty määrittämään todennäköinen toiminta vaihtoehdoista lepotila, harvennushakkuu ja päätehakkuu.

Tämän projektityön tavoitteena on puustokuvioaineistoa käyttäen muodostaa suunnitelma useasta lähekkäisestä puustokuviosta koostuvan leimikon optimaaliselle korjuuajankohdalle. Tämä korjuuajankohdan määrittäminen käsittää kaksi päätehtävää: kelvollisen kulkureitin määritys lähimmälle tielle ja puustokuvioista muodostetun leimikon korjuuajankohtien optimointi.

Aineistona projektityössä on käytössä vektorimuotoinen maaperäkartta-aineisto, vektoripohjainen tiekartta, yllä mainittu puustokuvioaineisto sekä laserkeilauksella muodostettu korkeusaineisto alueesta. Projektityön tarkoitus on ollut tuottaa laskentarutiini, joka edellä mainittua aineistoa apuna käyttäen tuottaa leimikoinnin, reitit leimikon puustokuvioilta tielle ja leimikon

puustokuvioiden korjuuajankohdat.

1.2 Mallin rajoitukset

Rajoituksia mallille muodostavat pääasiassa fyysiset rajoitukset metsäkoneen liikkumiselle. Metsäkoneen on maastossa mahdollista kulkea ylämäkiä, alamäkiä ja sivuttaissuuntaisia kallistuksia vain tiettyihin jyrkkyysasteisiin saakka, riippuen siitä, onko metsäkone tyhjä vai täynnä puuta.

Lisäesteitä kulkemiselle tuovat tietyyppien asettamat rajoitukset. Metsäkoneella ei voida ylittää moottoritietä sekä tiettyjä muita vilkkaasti liikennöityjä tietyyppejä. Tien käsittelyyn vaikuttaa myös se, voidaanko sen viereen varastoida puuta. Mikäli puuta ei voida varastoida ja se voidaan huoletta ylittää, tietä ei huomioida metsäkoneen liikkumisessa.

Myös maaperällä on omat vaikutuksensa kulkukelpoisuuteen. Metsäkoneella voidaan mennä joillekin maalajeille vain tiettynä vuodenaikana. Esimerkiksi suoalueelle voidaan ajaa vain ns. routatalven aikana. Lisäksi maaperäalueen kulkukelpoisuuteen vaikuttaa myös puuston määrä ja se, onko kyseessä pääte- vai harvennushakkuu.

Viimeinen tehtävänasettajalta saatu rajoitus koski leimikoiden muodostamistapaa. Päätehakkuulle leimikot muodostettiin pelkästään vierekkäisistä puustokuvioista, kun taas harvennushakkuulle leimikointi tehtiin asettamalla puustokuvioiden keskipisteiden etäisyyksille tietty yläraja.

Luku 2

Mallin rakentaminen

2.1 Yleisiä asetuksia

Laskentarutiinissa tarkasteltava alue diskretoidaan. Käytetään diskretointiparametrille arvoa r. Laskennassa käytetään r:lle arvoa 10 metriä, mutta malli on toteutettu siten, että arvoa voidaan haluttaessa muuttaa. Tarkasteltavan alueen määrittely tapahtuu siten, että annetaan alueen vasemman yläkulman KKJ-koordinaatit (x_0, y_0). Tämän jälkeen määrätään muodostettavan suorakulmion sivujen pituudet $d(X)$ ja $d(Y)$. Tällöin voidaan muodostaa koordinaattivektorit X ja Y, jotka sisältävät kaikki alueeseen kuuluvat x- ja y-koordinaatit.

$$X := [x_0, x_0 + r, x_0 + 2r, \ldots, x_0 + d(X) - r, x_0 + d(X)] \tag{2.1}$$

$$Y := [y_0 - d(Y), y_0 - d(Y) + r, \ldots, y_0 - r, y_0] \tag{2.2}$$

Yhtälössä (2.2) koordinaatit järjestetty siten, että alkioiden arvot kasvavat alkioiden järjestyksen mukaisesti vasemmalta oikealle. Tämän jälkeen alueen diskretointi voidaan muodostaa matriisiin XY, jonka alkioina on vektoreita. Pelkän alueen diskretoinnin kannalta vektorit sisältävät vain kunkin pisteen koordinaatit, mutta niihin voi tarvittaessa liittää muutakin informaatiota

$$XY := \begin{bmatrix} [x_0, y_0] & \cdots & [x_0 + d(X), y_0] \\ \vdots & \ddots & \vdots \\ [x_0, y_0 - d(Y)] & \cdots & [x_0 + d(X), y_0 - d(Y)] \end{bmatrix} \tag{2.3}$$

Yhtälössä (2.3) määritellään XY-matriisi. Kuten esityksestä nähdään, koor-

dinaatit ovat järjestetty matriisiin siten, että piste (x_0, y_0) on matriisin vasemmassa yläkulmassa, kuten myös maastossa.

2.2 Tienvieruspisteiden etsiminen

2.2.1 Tieaineiston esittely

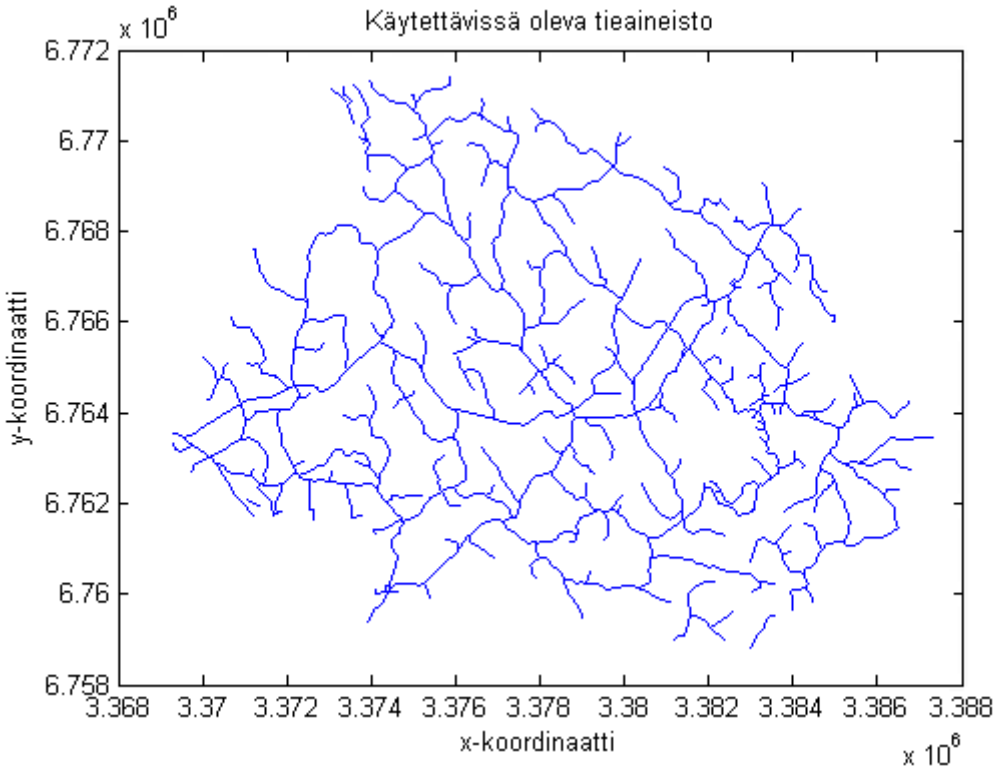
Optimoitaessa puuaineksen kuljetuskustannuksia leimikolta lähimmän käyvän tien viereen tarvitaan tienvieruspisteitä. Tässä tien käypyydellä tarkoitetaan sitä, että kyseinen tie on sellainen, jonka viereen voi varastoida puutavaraa. Esimerkiksi moottoritien varteen ei voida puutavaraa kerätä, mutta seututien varteen voidaan. Eri tieluokkien käypyys on määritelty toimeksiantajan taholta. Annetussa lähtöaineistossa tiepisteet ovat MapInfo-ohjelmiston käyttämässä tiedostoformaatista siten, että tiepisteissä on ilmoitettu kaikkien kulmapisteiden KKJ-koordinaatit. Lisäksi jokaiselle tielle on määrätty tieluokka. Ensiksi aineisto on muunnettava sopivaan muotoon, jotta sitä voidaan käsitellä laskentarutiinissa. Kuvassa 2.1 on esitelty tieaineisto MATLAB-ohjelmiston avulla. Kuvan koordinaatit ovat KKJ-järjestelmästä.

2.2.2 Algoritmin muodostaminen tienvieruspisteiden etsimiseen

Muodostetaan algoritmi käypien teiden vieruspisteiden löytämiseksi. Olkoon teiden lukumäärä N_T. Tällöin merkitään tien l kulmapisteet sisältävää matriisia T_l:llä, missä $l = 1, \ldots, N_T$. Määritellään T_l $2 \times n(l)$ kokoiseksi matriisiksi, missä $n(l)$ on tien kulmapisteiden määrä. Valitaan, että kulmapisteiden x-koordinaatit ovat ensimmäisessä sarakkeessa ja y-koordinaatit toisessa.

$$T_l = \begin{bmatrix} x_l^1 & x_l^2 & \cdots & x_l^{n(l)} \\ y_l^1 & y_l^2 & \cdots & y_l^{n(l)} \end{bmatrix}^T \tag{2.4}$$

Määritellään algoritmin esittämiseen tarvittavia funktiota. Olkoon $OK_T(T_l)$ funktio, joka palauttaa arvon 'TRUE', kun tie T_l kuuluu käypää tieluokkaan ja muutoin funktio palauttaa arvon 'FALSE'. Lisäksi tarvitaan funktio, joka lisää tienvieruspisteet diskretointimatriisin vektoreihin. Olkoon tällainen funktio $ADD([x, y]\, XY)$, joka merkitsee pisteen (x, y) tienvieruspisteeksi matriisiin XY, joka sisältää alueen diskretoinnin. Piste (x, y) määritellään



Kuva 2.1: Lähtöaineisto tiedatasta.

algoritmissa siten, että se aina jokin diskretointipiste, eikä funktion siis tarvitse muuntaa pistettä diskretointiin sopivaksi. Näin voidaan esitellä algoritmi:

$l \longleftarrow 1$
while $l \leq N_T$ do
    if $OK_T(T_l)$ then
        $k \longleftarrow 1$
        while $k < n(l) - 1$ do
            if $\left|\frac{[T_l]_{k+1,2} - [T_l]_{k,2}}{[T_l]_{k+1,1} - [T_l]_{k,1}}\right| > 1$ then
                $y \in Y \cap \left[\min\left\{[T_l]_{k,2}, [T_l]_{k+1,2}\right\}, \max\left\{[T_l]_{k,2}, [T_l]_{k+1,2}\right\}\right]$
                $x(y) = \frac{[T_l]_{k+1,1} - [T_l]_{k,1}}{[T_l]_{k+1,2} - [T_l]_{k,2}}\left(y - [T_l]_{k,2}\right) + [T_l]_{k,1}$
                for $\forall\ y$ do
                    $x_d = X \cap [x(y) - r, x(y))$
                    $x_u = X \cap [x(y), x(y) + r)$
                    $ADD([x_d, y], XY)$

$ADD([x_u, y], XY)$
endfor
else
$x \in X \cap \left[\min\left\{[T_l]_{k,1}, [T_l]_{k+1,1}\right\}, \max\left\{[T_l]_{k,1}, [T_l]_{k+1,1}\right\}\right]$
$y(x) = \frac{[T_l]_{k+1,2} - [T_l]_{k,2}}{[T_l]_{k+1,1} - [T_l]_{k,1}}\left(x - [T_l]_{k,1}\right) + [T_l]_{k,2}$
for $\forall\ x$ do
$y_d = Y \cap [y(x) - r, y(x))$
$y_u = Y \cap [y(x), y(x) + r)$
$ADD([x, y_d], XY)$
$ADD([x, y_u], XY)$
endfor
endif
$k \longleftarrow k + 1$
endwhile
endif
$l \longleftarrow l + 1$
endwhile

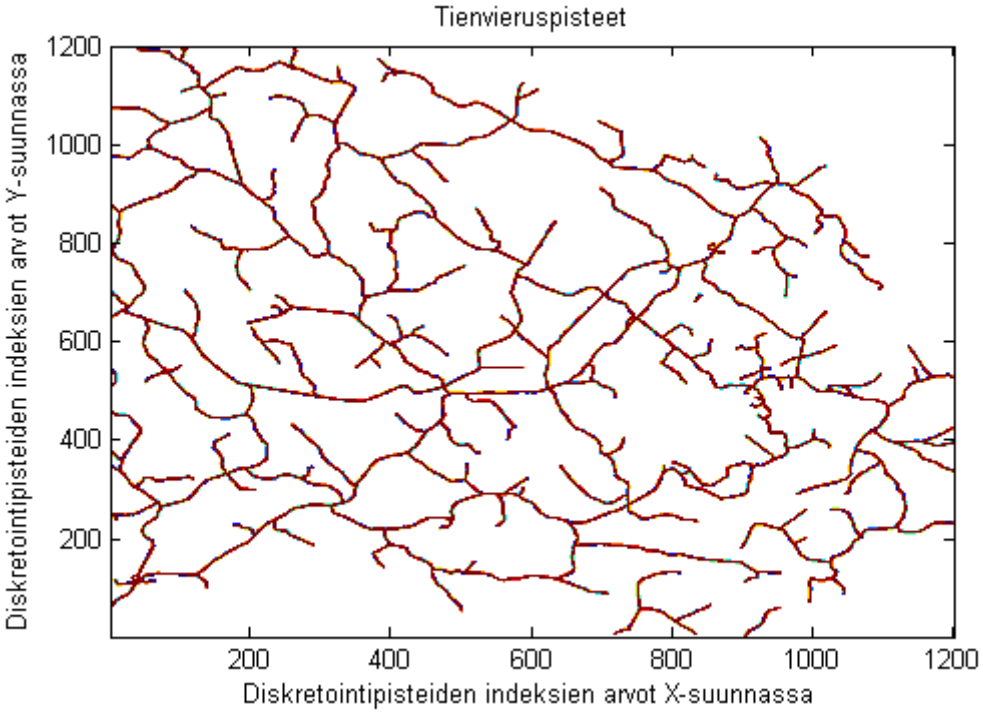
Algoritmissa notaatiolla $[T_l]_{i,j}$ viitataan tien T_l kulmapisteet sisältävän matriisin alkioon (i, j), missä i on rivin indeksi ja j sarakkeen indeksi. Silmukan sisällä algoritmi tarkistaa siis ensimmäisellä if-lauseella, että tarkasteltava tie T_l on käypä. Mikäli näin on, algoritmi valitsee kaksi peräkkäistä tien kulmapistettä kerrallaan. Kulmapisteiden välille algoritmi määrittää janan. Ensiksi tarkistetaan, että onko janan kulmakertoimen itseisarvo suurempi kuin yksi vai ei. Mikäli kulmakertoimen itseisarvo on ykköstä suurempi, algoritmi hakee kaikki janan päätepisteiden y-koordinaattien välille kuuluvat diskretoidut y:n arvot ja määrittää näitä vastaavat x-koordinaattien arvot janalla. Tämän jälkeen algoritmi käy läpi kaikki löydetyt x-koordinaattien arvot ja etsii jokaista arvoa kohden kaksi x:n diskretoitua arvoa, joiden välille[1] x:n arvo osuu. Nämä diskretointiarvot algoritmi tämän jälkeen merkitsee tienvieruspisteiksi matriisiin XY. Toisaalta jos kulmakertoimen itseisarvo olikin pienempi tai yhtä suuri kuin yksi, algoritmi etsii janan päätepisteiden x-koordinaattien välille kuuluvat diskretoidut x:n arvot ja hakee näitä vastaavat y-koordinaattien arvot janan avulla. Seuraavaksi algoritmi käy läpi kaikki löydetyt y-koordinaattien arvot ja hakee kaikille arvoille ne kaksi diskretoitua y:n arvoa, joiden väliin y arvo kuuluu. Vastaavasti, kuin ensimmäisessä tapauksessa, myös nyt algoritmi lisää löydetyt tienvieruspisteet matriisiin XY. Algoritmin toiminnan kannalta on nyt sallittava se, että kulmakertoimen it-

[1] Välin pituus on r.

seisarvo saattaa olla ääretön. Tällöin tietysti epäyhtälö $\infty > 1$ on voimassa, jolloin algoritmi osaa toimia tästä eteenpäin.

2.2.3 Algoritmin toiminnan havainnollistaminen

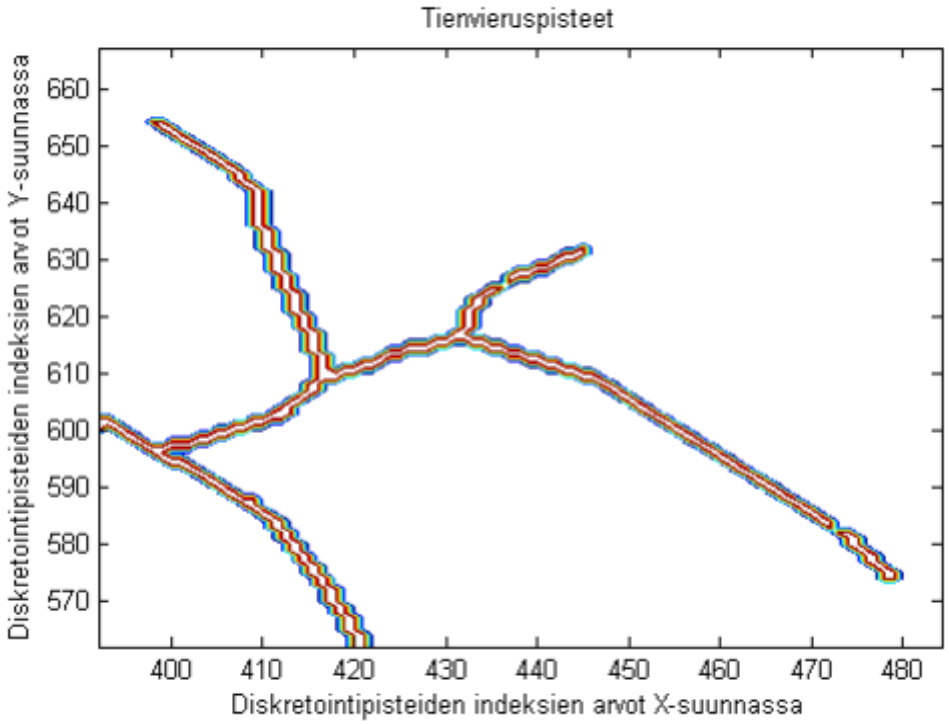
Valitaan tienvieruspisteiden esittelyä varten tarkasteltavan alueen vasemmaksi yläkulmaksi $(x_0, y_0) = (3374019.176, 67770940.444)$, muodostettavan suorakulmion sivunpituuksiksi $d(X) = d(Y) = 12000$ (metriä) ja tarkkuudeksi $r = 10$ (metriä). Tällöin alue on maastossa 12 km kertaa 12 km neliö. Tässä käyvät ja kielletyt tieluokat on erikseen määritelty, mutta osoittautuu, että kaikki tieaineiston sisältämät tiet kuuluvat käypiin tieluokkiin, joten niihin ei tässä kiinnitetä sen enempää huomiota. Piirretään tienvieruspisteet alueella sijaitsevalle tiedatalle.



Kuva 2.2: Tienvieruspisteet koko valitulla alueella.

Kuvassa 2.2 nähdään valitun alueen tienvieruspisteet. Kuvan vaaka- ja pystyakselien arvot kuvaavat diskretointipisteiden indeksejä, jotka juoksevat välillä $[1, 1201]$ molemmissa tapauksissa. Välin yläraja saadaan suoraan laskemalla $d(X)/r + 1$ ja vastaavasti myös y-suunnassa. Näin ollen alue sisältää $1201 \cdot 1201 = 1442401$ diskretoitua pistettä. Algoritmi ei kuitenkaan käy läpi

kuin sellaisia pisteitä, jotka todella ovat tienvieruspisteitä. Lisäksi voidaan todeta tieverkoston olevan samanlainen, kuin kuvassa 2.1. Tässä tarkasteltava alue ei tosin kata koko tieverkoston aluetta.



Kuva 2.3: Suurennos tienvieruspisteistä.

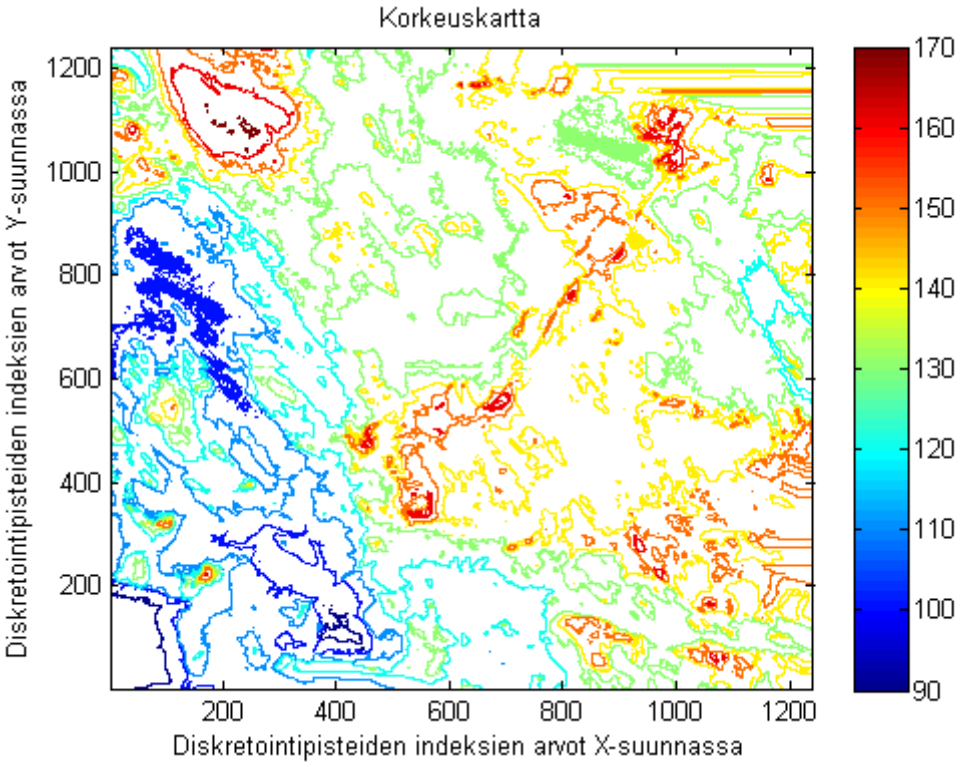
Kuvassa 2.3 on esitetty suurennos kuvan 2.2 tienvieruspisteistä. Kuvaan liittyen voidaan todeta, että itse tie kulkee tienvieruspisteiden välissä ja tienvieruspisteiden valinta tien suunnan mukaan koordinaatistossa vaikuttaa toimivan hyvin. Näitä pisteitä käytetään myöhemmässä vaiheessa leimikolta lähtevän reitin loppupisteinä optimaalisia metsäkuljetusreittejä etsittäessä.

2.3 Ei-käypien reittipisteiden määrittäminen

2.3.1 Korkeusaineiston esittely

Optimaalisten metsäkuljetusreittien etsimisessä on otettava huomioon maaston ja ympäristön asettamia rajoituksia. Käytännössä metsäkuljetukset toteutetaan metsäkoneiden avulla, jotka pystyvät kyllä liikkumaan maastossa,

mutta liian jyrkissä kohdissa ei. Lisäksi metsäkoneella ei voi ylittää tiettyjä tietyyppejä, kuten esimerkiksi moottoriteitä tai muita paljon liikennöityjä väyliä. Käytettävissä on maastoalueen korkeuskartta, jonka tarkkuus on 1 metri ja kappaleessa 2.2 esitelty tieaineisto. Korkeuskartta on jälleen muunnettava käyttökelpoisempaan muotoon harmaasävykuvasta. Muodon valinta kohdistui matriisiesitykseen lausekkeen (2.3) tapaan, jossa matriisin alkiot vastaavat r:n välein diskretoituja pisteitä ja jokaisessa alkiossa on kyseisen pisteen korkeus merenpinnasta metreinä. Merkitään kyseistä matriisia H_{XY}:llä. Valitaan diskretoinnin parametrit jälleen siten, että vasemman yläkulman koordinaatit ovat $(x_0, y_0) = (3374019.176, 67770940.444)$, muodostettavan suorakulmion sivunpituudet ovat $d(X) = d(Y) = 12000$ (metriä) ja tarkkuus on $r = 10$ (metriä). Esitetään korkeuskartta MATLAB-ohjelmiston avulla. Kuvassa 2.4 nähdään alueen korkeuskartta. Kuvan oikeassa reunassa palkki yhdistää kuvassa olevat värit korkeuteen merenpinnasta metreinä. Havaitaan, että kuvan oikeassa yläkulmassa on jonkin verran korruptoitunutta dataa.



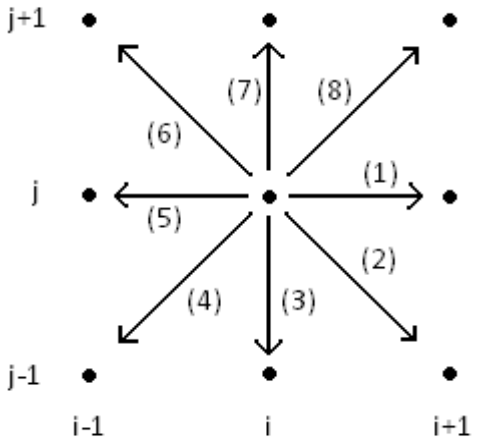
Kuva 2.4: Korkeuskartta alueesta.

2.3.2 Algoritmin muodostaminen liian suurien korkeusvaihteluiden etsimiseksi

Taulukossa 2.1 nähdään suurimmat mahdolliset kallistuskulmat, jotka metsäkoneelle sallitaan riippuen siitä, onko metsäkoneella kuorma päällä vai ei. Sivuttaiskallistus ei riipu metsäkoneen kuormasta. Asteluvut ovat toimeksiantajan valitsemia.

Maksimikulmat asteina	
Täytenä ylämäkeen	$\alpha_1 = 15°$
Täytenä alamäkeen	$\alpha_2 = 25°$
Tyhjänä ylämäkeen	$\alpha_3 = 20°$
Tyhjänä alamäkeen	$\alpha_4 = 30°$
Sivuttaiskallistus	$\alpha_5 = 15°$

Taulukko 2.1: Maksimikallistukset metsäkoneen etenemiselle maastossa.



Kuva 2.5: Diskretoidusta pisteestä (i, j) muihin pisteisiin liikkuminen.

Reittimallissa diskretoidusta pisteestä toiseen siirtyminen sallitaan vain pysty- ja vaakasuunnassa sekä ala- ja yläviistoon. Tällöin yhdestä pisteestä voidaan siirtyä kahdeksaan eri pisteeseen. Kuvassa 2.5 on esitetty sallitut siirtymät diskretoidusta pisteestä eteenpäin. Kuvassa i on diskretointipisteiden indeksi X-suunnassa ja j vastaavasti Y-suunnassa. Lisäksi suunnat ovat numeroitu ja niitä on siis 8 kappaletta. Tässä kahden vaaka- tai pystysuunnassa vierekkäisen diskretointipisteen välinen etäisyys on r. Viistosti vierekkäisten pistei-

den (esimerkiksi kuvassa 2.5 pisteiden (i, j) ja $(i+1, j+1)$) välinen etäisyys sen sijaan on $\sqrt{2}r$ Pythagoraan lauseen nojalla.

Korkeuserot kaikkiin käypiin suuntiin
$\Delta h_1(i,j) = [H_{XY}]_{j,i+1} - [H_{XY}]_{j,i}$
$\Delta h_2(i,j) = [H_{XY}]_{j-1,i+1} - [H_{XY}]_{j,i}$
$\Delta h_3(i,j) = [H_{XY}]_{j-1,i} - [H_{XY}]_{j,i}$
$\Delta h_4(i,j) = [H_{XY}]_{j-1,i-1} - [H_{XY}]_{j,i}$
$\Delta h_5(i,j) = [H_{XY}]_{j,i-1} - [H_{XY}]_{j,i}$
$\Delta h_6(i,j) = [H_{XY}]_{j+1,i-1} - [H_{XY}]_{j,i}$
$\Delta h_7(i,j) = [H_{XY}]_{j+1,i} - [H_{XY}]_{j,i}$
$\Delta h_8(i,j) = [H_{XY}]_{j+1,i+1} - [H_{XY}]_{j,i}$

Taulukko 2.2: Korkeuserot kaikkiin käypiin suuntiin diskretointipisteestä (i, j).

Taulukossa 2.2 on määritelty korkeuserot kaikkiin sallittuihin suuntiin diskretointipisteestä (i, j). Merkintä $[H_{XY}]_{j,i}$ viittaa matriisin H_{XY} rivillä j sarakkeessa i olevaan alkioon. Näin ollen voidaan muodostaa suurin mahdollinen korkeusero, joka kahden diskretointipisteen välillä saa olla, että metsäkone voi edetä niiden välillä:

$$\Delta h_{max}(\alpha; k) := \begin{cases} r\tan(\alpha) & k \in \{1,3,5,7\} \\ \sqrt{2}r\tan(\alpha) & k \in \{2,4,6,8\} \end{cases} \tag{2.5}$$

Olkoon N_X diskretoitujen pisteiden määrä vaakatasossa ja N_Y vastaavasti pystytasossa. Esitellään suppea versio algoritmista, joka käy läpi kaikki sallitut suunnat diskretoiduista pisteistä ja tarkistaa niiden käypyydet korkeuden vaihteluiden suhteen. Esitellään algoritmin yhtälössä (2.5) määritetyn funktion lisäksi tarvitsema funktio. Olkoon $ADD([i, j, k], b, XY)$ funktio, joka lisää arvon b matriisiin XY alkiona j, i[2] olevan vektorin alkioksi k. Tässä siis (i, j) kuvaa diskretointipistettä ja k suuntaa kyseisestä pisteestä. Arvon b määrittely:

- '$b = 0$' vastaa tilannetta, jossa metsäkone voi liikkua suuntaan k pisteestä (i, j)

[2]Matriiseissa on tapana ilmoittaa ensin rivin indeksi ja tämän jälkeen sarakkeen, kun taas yleisesti koordinaattien yhteydessä ilmoitetaan ensin pystytason ja sitten vaakatason määräävä indeksi.

- '$b = 1$' tarkoittaa sitä, että metsäkone voi tyhjänä liikkua suuntaan k pisteestä (i, j)
- '$b = 2$' vastaa sitä, että metsäkone ei voi missään tilanteessa liikkua suuntaan k pisteestä (i, j)

Esitellään suppea versio algoritmista. Tässä suppealla tarkoitetaan sitä, että esiteltävä algoritmi ei käy läpi diskretoidun alueen reunapisteitä. Tämän toteuttaminen sinällään ole monimutkainen tehtävä, mutta sen esittäminen analyyttisessa muodossa ei ole järkevää, mikäli esitys halutaan pitää kompaktina. Olkoon C_{XY} $N_Y \times N_X$-matriisi, jonka kaikki alkiot ovat kahdeksanalkioisia vektoreita, joiden kaikki alkiot ovat nollia.

for $\forall\, i \in [2, N_X - 1]$ do
  for $\forall\, j \in [2, N_Y - 1]$ do
    for $\forall\, k \in \{1, 2, 3, 4, 5, 6, 7, 8\}$
      if $\Delta h_k(i, j) > \Delta h_{max}(\alpha_3; k)$ then
        $ADD([i, j, k], 1, C_{XY})$
        if $\Delta h_k(i, j) > \Delta h_{max}(\alpha_1; k)$ then
          $ADD([i, j, k], 2, C_{XY})$
        endif
      elseif $\Delta h_k(i, j) < -\Delta h_{max}(\alpha_4; k)$ then
        $ADD([i, j, k], 1, C_{XY})$
        if $\Delta h_k(i, j) < -\Delta h_{max}(\alpha_2; k)$ then
          $ADD([i, j, k], 2, C_{XY})$
        endif
      endif
      if $|\Delta h_k(i, j)| > \Delta h_{max}(\alpha_5; k)$ then
        $ADD([i, j, k \oplus 2], 2, C_{XY})$
        $ADD([i, j, k \ominus 2], 2, C_{XY})$
      endif
    endfor
  endfor
endfor

Yllä esitelty algoritmi käy siis läpi kaikki diskretoidun alueen pisteet pois lukien reunapisteet. Reunapisteet on käytävä myös vastaavalla tavalla läpi algoritmissa kuin muutkin pisteet, mutta niille on merkattava diskretoidulta alueelta pois vieville suunnille arvoksi 2, jotta optimaalisia reittejä etsittäessä ei päädyttäisi ulos käyvältä alueelta. Jokaisen pisteen kohdalla algoritmi käy läpi korkeuserot kaikkiin suuntiin tarkastaen ensin ylämäkeen mennessä

tyhjänä ja täytenä, jonka jälkeen tarkastetaan alamäkeen mennessä tyhjänä ja täytenä. Tässä järjestyksellä on väliä, sillä tyhjänä metsäkone voi liikkua jyrkemmissä kulmissa ja sietää siten suurempaa korkeuden vaihtelua. Mikäli liian suuria korkeuseroja löytyy, algoritmi lisää tiedon siitä C_{XY}-matriisiin. Algoritmi tarkistaa vielä erikseen korkeuseron suuntaan k kallistuskulman suhteen, ja mikäli korkeusero on liian suuri, algoritmi merkitse suuntaa k vastaan kohtisuorat suunnat ei-käyviksi. Tässä yhteydessä merkintä $\oplus$ vastaa normaalia summaa, kun $k < 7$. Mikäli $k \geq 7$, summasta tulee $k + 2 - 8$. Vastaavasti $\ominus$ normaali vähennysoperaatio, kun $k > 2$, muutoin $k - 2 + 8$. Näin löydetään aina suuntaa k vastaan kohtisuorat suunnat ja suuntaindeksin arvot pysyvät sallittuina.

2.3.3 Kiellettyihin tietyyppeihin kuuluvien teiden ylittämisen estäminen

Seuraavaksi rajoitusehdoissa täytyy huomioida se, että joihinkin tieluokkiin kuuluvia teitä ei saa ylittää. Kielletyjä tieluokkia ovat esimerkiksi moottoritiet, moottoriliikennetiet ja valtatiet. Lisäksi tässä voidaan pitkälti käyttää apuna kappaleessa (2.2) esitettyjä asioita. Olkoon $NOTOK_T(T_l)$ funktio, joka palauttaa arvon 'TRUE', jos tie T_l on ei-käypä ja muutoin funktio palauttaa arvon 'FALSE'. Tällä kertaa halutaan nimenomaan tehdä sellaisia suuntia k ei-käyviksi kaikissa tilanteissa, jotka johtavat sellaiseen tiepisteeseen, jossa tien luokka ei ole käypä. Muodostetaan jälleen funktio $ADD([x, y], b, C_{XY})$, joka merkitsee matriisiin C_{XY} kaikki suunnat (kahdeksan kappaletta) diskretoituun pisteeseen (x, y)[3] ei-käyviksi, eli arvolla $b = 2$. Tässä piste (i, j) kuuluu ei-käypään tieluokkaan. Muodostetaan algoritmi, joka etsii kaikki kielletyt tiepisteet ja merkitsee suunnat niihin ei-käyviksi.

$l \longleftarrow 1$
while $l \leq N_T$ do
    if $NOTOK_T(T_l)$ then
        $k \longleftarrow 1$
        while $k < n(l) - 1$ do
            if $\left|\frac{[T_l]_{k+1,2} - [T_l]_{k,2}}{[T_l]_{k+1,1} - [T_l]_{k,1}}\right| > 1$ then
                $y \in Y \cap \left[\min\left\{[T_l]_{k,2}, [T_l]_{k+1,2}\right\}, \max\left\{[T_l]_{k,2}, [T_l]_{k+1,2}\right\}\right]$
                $x(y) = \frac{[T_l]_{k+1,1} - [T_l]_{k,1}}{[T_l]_{k+1,2} - [T_l]_{k,2}}\left(y - [T_l]_{k,2}\right) + [T_l]_{k,1}$

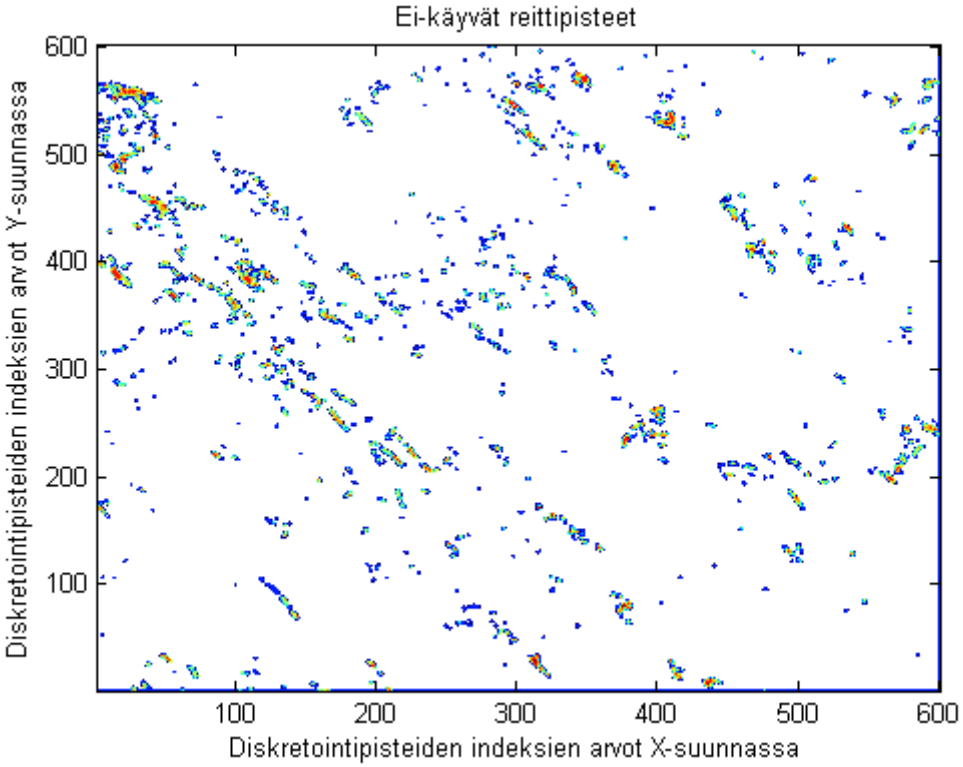
[3] Diskretoinnin takia pisteet (x, y) ja (i, j) ovat rinnasteisia, vaikka toinen viittaakin koordinaatteihin ja toinen indekseihin.

for $\forall\, y$ do
$\tilde{x} = X \cap [x(y) - r/2, x(y) + r/2)$
$ADD([\tilde{x}, y], 2, C_{XY})$
endfor
else
$x \in X \cap \left[\min\left\{[T_l]_{k,1}, [T_l]_{k+1,1}\right\}, \max\left\{[T_l]_{k,1}, [T_l]_{k+1,1}\right\}\right]$
$y(x) = \frac{[T_l]_{k+1,2} - [T_l]_{k,2}}{[T_l]_{k+1,1} - [T_l]_{k,1}} \left(x - [T_l]_{k,1}\right) + [T_l]_{k,2}$
for $\forall\, x$ do
$\tilde{y} = Y \cap [y(x) - r/2, y(x) + r/2)$
$ADD([x, \tilde{y}], 2, C_{XY})$
endfor
endif
$k \longleftarrow k + 1$
endwhile
endif
$l \longleftarrow l + 1$
endwhile

Algoritmi on hyvin samankaltainen kappaleessa (2.2) määritetyn kanssa. Algoritmi käy läpi kaikki tiet. Mikäli algoritmi löytää tien, joka on ei-käypä, se valitsee siitä kaksi peräkkäistä kulmapistettä kerrallaan ja approksimoi pisteiden väliä janalla. Mikäli janan kulmakertoimen itseisarvo on suurempi kuin yksi, algoritmi etsii janaan kuuluvia y-koordinaatteja diskretoitujen y:n arvojen joukosta. Tämän jälkeen haetaan vastaavat x-koordinaatit ja etsitään niitä vastaavat diskretoidut pisteet, joihin johtavat suunnat algoritmi merkitsee kielletyiksi. Toisaalta jos janan kulmakertoimen itseisarvo on pienempi tai yhtä suuri kuin yksi, algoritmi etsiikin janaan kuuluvia x-koordinaatteja, joita vastaavat y-koordinaatit saadaan jälleen janan kautta. Näille y-koordinaateille etsitään tämän jälkeen vastineet diskretoitujen y:n arvojen joukosta ja kaikki suunnat näin löydettyihin pisteisiin merkitään kielletyiksi.

2.3.4 Ei-käypien reittipisteiden esittely

Esitellään seuraavaksi löydetyt ei-käyvät suunnat. Valitaan diskretoidun alueen vasemmaksi yläkulmaksi jälleen $(x_0, y_0) = (3374019.176, 67770940.444)$, mutta nyt muodostettavan suorakulmion sivunpituudet ovat $d(X) = d(Y) = 6000$ (metriä) ja tarkkuus on $r = 10$ (metriä).



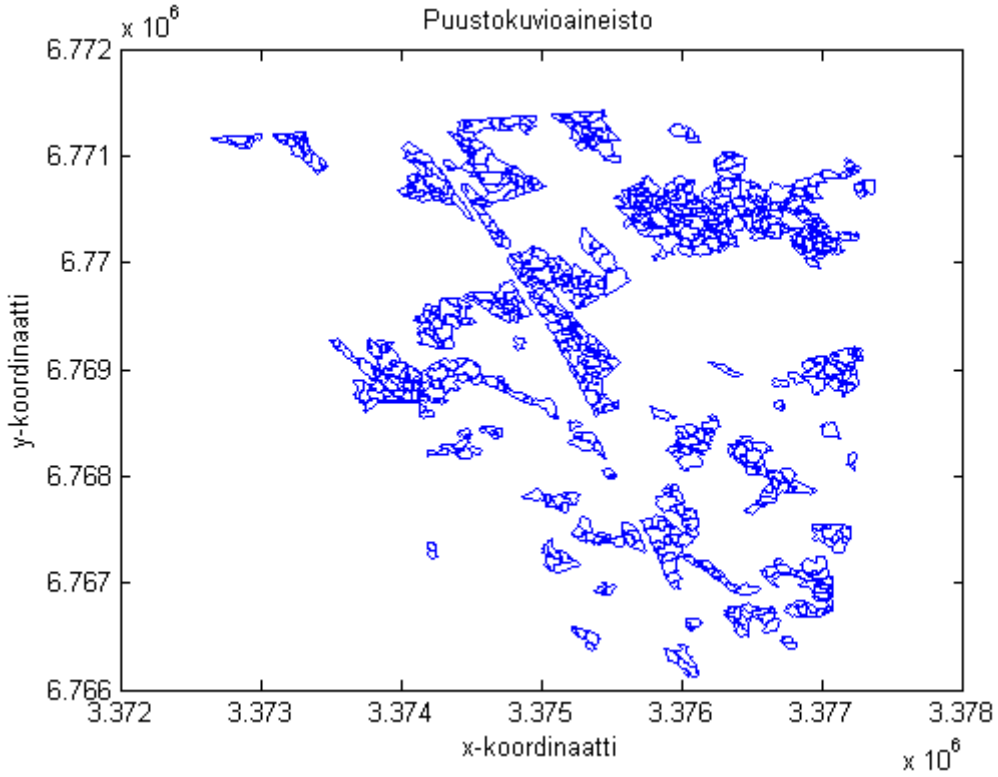
Kuva 2.6: Ei-käyvät reittipisteet diskretoidulla alueella.

Kuvassa 2.6 on esitelty valitulta alueelta ei-käyvät reittipisteet. Värjätyt pisteet ovat vähintään yhteen suuntaan ei-käypiä, jolloin pisteiden väri on sininen ja pisteiden ollessa punaisia kaikki suunnat kyseisestä pisteestä ovat kiellettyjä. Kuvassa alueen reunoilla olevat kielletyt suunnat näkyvät vain alareunassa ja oikeassa reunassa, mutta tämä johtuu käytetyn piirto-ohjelmiston ominaisuuksista. Muiltakin reunoilta siis löytyy kielletyt suunnat. Osoittautuu myös, että tieaineiston joukossa kaikki tiet kuuluvat käypiin luokkiin, joten teihin liittyviä ei-käypiä reittipisteitä ei tässä ole. Teiden ylittämisen kieltävän algoritmin toimintaa testattiin kuitenkin käytännössä itse generoidulla havaintoaineistolla ja sekin toimi oletetulla tavalla.

2.4 Puustokuvioaineiston käsittely ja leimikointi

2.4.1 Puustokuvioaineiston esittely

Tarkasteltavalla alueella metsät on jaettu puustokuvioihin. Puustokuviot sisältävät vain yhtä pääpuulajia ja ovat kooltaan enintään noin 250 metriä kertaa 250 metriä. Jokaista puustokuviota kohden on mm. puustokuvion id, pääpuulaji, puustokuvion sisältämän puumäärän tilavuus, alueen pinta-ala, hakkuukertymä ja puustokuvion keskipiste.



Kuva 2.7: Pääte- tai harvennushakattavaksi merkityt puustokuviot.

Kuvassa 2.7 nähdään puustokuvioaineisto. Kaikkiaan koko tarkasteltavalla alueella on 890 puustokuviota, jotka on valittu joko pääte- tai harvennushakattaviksi. Approksimoidaan puustokuvioita pisteillä. Käytännössä aineistosta löytyy jo valmiiksi puustokuvioiden keskipisteet, jotka on vain siirrettävä lähimpää diskretoituun pisteeseen. Tällöin puustokuvioiden keskipisteet siirtyvät enintään $\sqrt{2}r/2$ verran. Mikäli $r = 10$ (metriä), maksimisiirtymä on

noin 7.07 metriä.

2.4.2 Leimikoinnin muodostaminen ja esittely

Puustokuvioista on tarkoitus muodostaa leimikoita, eli hakkuualueita. Tässä yhteydessä samaan leimikkoon halutaan valita sellaiset puustokuviot, jotka ovat lähellä toisiaan. Käytännössä tämä tarkoittaa vierekkäisiä puustokuvioita. Tässä tullaan hyödyntämään joko puustokuvioiden keskipisteiden välisiä etäisyyksiä tai tekn. yo. Jarno Leppäsen diplomityössään tuottamaa dataa puustokuvioiden naapuruuksista. Olkoon puustokuvioita N_P kappaletta. Lisäksi olkoon puustokuvio P_l, missä $l = 1, \ldots, N_P$. Puustokuviolla P_l on keskipiste (x_l^C, y_l^C), joka kuuluu diskretoitujen pisteiden joukkoon. Esitellään kaksi eri menetelmää puustokuvioiden leimikoinnin muodostamiseksi.

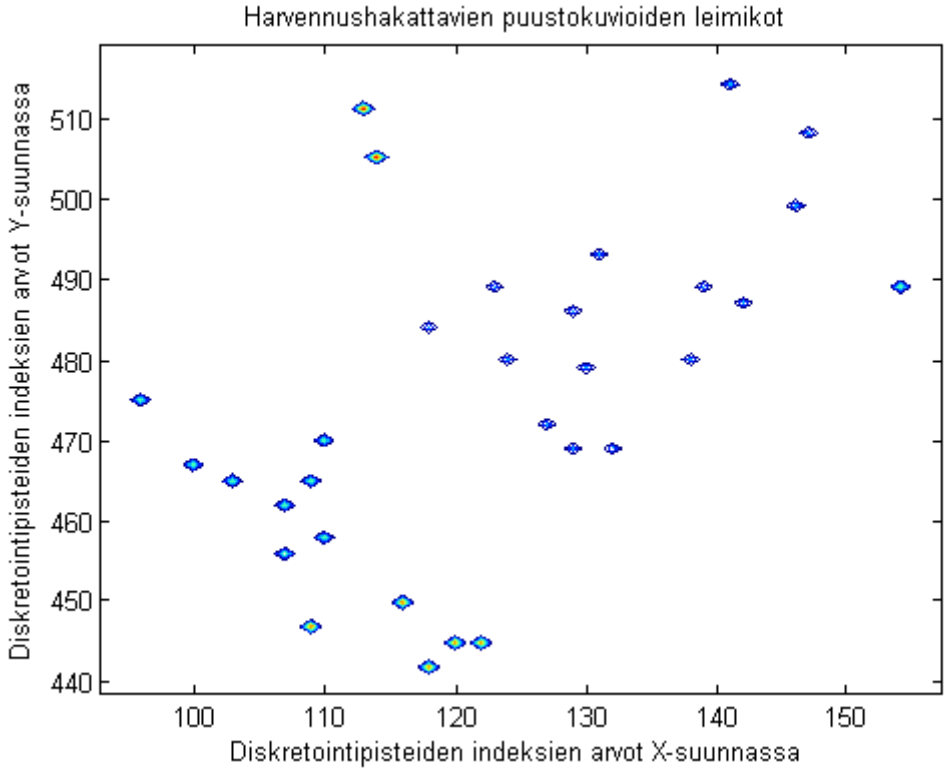
1. Muodostetaan jokaista puustokuviota kohti joukko L_l, missä $l = 1, \ldots, N_P$. Joukkoon L_l merkitään kuuluviksi kaikki halutuntyyppiset puustokuviot, joiden keskipiste on enintään etäisyydellä R puustokuvion P_l keskipisteestä mukaan lukien P_l itse. Näin ollen saadaan N_P kappaletta puustokuvioiden joukkoja. Näistä joukoista voidaan muodostaa leimikoita siten, että muodostetaan unioni kaikista niistä joukoista L_l, joilla on vähintään yksi yhteinen alkio jonkin toisen unioniin kuuluvan joukon kanssa.

2. Muodostetaan jokaista puustokuviota kohti joukko L_l, missä $l = 1, \ldots, N_P$. Joukkoon L_l merkitään kuuluvaksi kaikki halutuntyyppiset puustokuviot, jotka on merkattu kyseisen puustokuvion naapureiksi[4]. Tällöin saadaan N_P kappaletta puustokuvioiden joukkoja. Vastaavasti kuin kohdassa 1., saaduista joukoista voidaan muodostaa leimikoita siten, että muodostetaan kaikista niistä puustokuvioiden joukoista L_l unioni, missä joukolla L_l on vähintään yksi yhteinen alkio jonkin toisen unioniin kuuluvan joukon kanssa.

Suurin käytännön ero menetelmillä on se, että naapuruuksissa ei tien erottamia puustokuviota olla merkattu naapureiksi. Keskipisteiden etäisyyksien perusteella leimikoita muodostettaessa kuitenkin myös tien erottamat puustokuviot voivat kuulua samaan leimikkoon. Näin ollen menetelmällä 1. pitäisi löytyä keskimäärin hieman vähemmän leimikoita.

[4]Puustokuvio on aina merkitty itsensä naapuriksi.

Valitaan tarkastelualue jälleen siten, että alueen vasemmaksi yläkulmaksi tulee $(x_0, y_0) = (3374019.176, 67770940.444)$, sivujen pituuksiksi valitaan $d(X) = d(Y) = 6000$ (metriä) ja tarkkuus jälleen $r = 10$ (metriä). Lisäksi toimeksiantajan taholta puustokuvioiden keskipisteiden väliseksi maksimietäisyydeksi on asetettu $R = 120$ (metriä). Valitulla alueella on nyt 773 puustokuviota, joista 429 päätehakattavia ja 344 harvennushakattavia. Määritellään päätehakattavat leimikot naapuruuksien perusteella ja harvennushakattavat leimikot keskipisteiden etäisyyksien perusteella. Tällöin saadaan 45 päätehakattavaa leimikkoa ja 26 harvennushakattavaa leimikkoa.

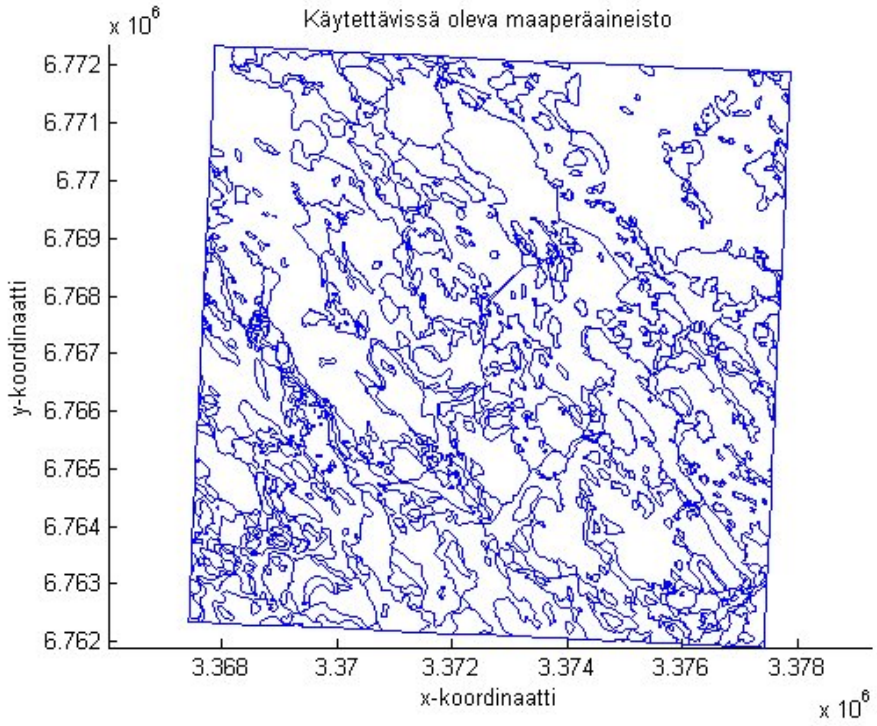


Kuva 2.8: Harvennushakattaviksi merkityistä puustokuvioista muodostettuja leimikoita.

Kuvassa 2.8 nähdään harvennushakattavien puustokuvioiden keskipisteitä, joiden leimikointi on esitetty värikoodauksella. Kuvassa on kuuteen eri leimikkoon kuuluvia puustokuvioita.

2.5 Maaperäaineiston käsittely ja puustokuvioiden päämaalajin määrittäminen

2.5.1 Maaperäaineiston muokkaaminen



Kuva 2.9: Lähtöaineisto maaperädatasta.

Alkuperäinen maaperädata, joka on esitetty graafisesti kuvassa 2.9 koostuu joukosta vektoreita, joihin on tallennettu tiettyä maalajia sisältävien alueiden kulmapisteet. Jotta aineisto saataisiin kätevästi käyttöön reitinlaskennassa, se on muokattava matriisimuotoon.

Aineiston muokkaus toteutetaan varsin suoraviivaisesti luomalla maastokarttaa vastaava matriisi, jonka jokaiselle kartan pistettä vastaavalle alkiolle käydään läpi maalajialueita, kunnes löydetään se alue, johon piste kuuluu. Yksinkertaisuudestaan huolimatta kyseinen prosessi on sangen laskentaintensiivinen johtuen erityisesti maalajipolygonien sisäpisteiden määritykseen käytetyn valmiin algoritmin hitaudesta etenkin Python-ohjelmointikielellä teh-

dyssä toteutuksessa. Laskentaa pyritään nopeuttamaan kahdella tavalla:

1. Vierekkäiset pisteet kuuluvat todennäköisimmin samaan maalajialueeseen, joten kultakin pisteeltä kysytään aina ensimmäisenä, kuuluuko se samaan alueeseen kuin edellinen käsitelty piste. Tämä nopeuttaa laskentaa huomattavasti.

2. Koska piste on sitä todennäköisempää löytää tietyltä alueelta, mitä suurempi alue on, ennen maalajialueiden läpikäymistä alueet järjestetään kulmapisteiden lukumäärän mukaiseen suuruusjärjestykseen. Tällä saadaan aikaan pienempi, mutta kuitenkin merkittävä hyöty.

2.5.2 Puustokuvioiden päämaalajin määrääminen

Puustokuvion kulkukelpoisuuden määrityksessä tarvittava päämaalajin määritys toteutettiin siten, että kunkin puustokuvion sisältä arvotaan tietty, riittävän suuri määrä pisteitä. Näille pisteille määritetään maalajit ja suurimman määrän osumia kerännyt maalaji asetetaan puustokuvion päämaalajiksi. Laskentaa pyritään jälleen nopeuttamaan kysymällä kultakin pisteeltä ensimmäisenä, kuuluuko se samaan maalajialueeseen kuin edellinen käsitelty piste.

2.6 Metsäkoneen reittien määrittäminen

2.6.1 Käytettävät aineistot ja menetelmät

Lyhimmän reitin määrittämisessä käytetään edellä esiteltyä tieaineistoa, korkeusaineistoa sekä maaperästä saatua aineistoa. Reitin laskenta tehdään rasterimuodossa, mutta lopullinen reitti palautetaan vektorimuotoisena, jotta sen käsittely MapInfo-ohjelmistossa helpottuisi. Reitin määrityksessä luodaan matriisi, jossa annetaan lyhimmän mahdollisen reitin pituus tielle matriisin jokaiselle alkiolle. Koska maaperäaineisto on sääriippuvaista, suoritetaan laskenta jokaiselle vuodenaikatyypille. Samoin korkeusdatan rajoittaman kulkukelpoisuuden takia täyden ja tyhjän koneen tapauksille palautetaan myös eri matriisit.

Lyhin reitti oli aluksi tarkoitus määrittää dynaamisen optimoinnin keinoin. Tällöin optimointi olisi aloitettu lähtöpisteestä ja siitä tutkimalla pisteen ympäristöä olisi määritetty paras kulkureitti tielle. Tällöin reitti olisi optimoi-

tu leimikon korjuun optimoinnin yhteydessä. Päädyimme kuitenkin ajamaan optimaalisen reitin jokaiselle rasterille, koska tällöin riittää se, että ajetaan reitin optimointi vain kerran alueelle ja tallennetaan tulokset myöhempää käyttöä varten. Näin saadaan selkeitä ajallisia hyötyjä leimikon optimointimallia ajaessa, kun reittiä ei tarvitse hakea enää tässä vaiheessa. Koska korkeuskartta ja maaperä pysyvät alueille muuttumattomina, voidaan ajettua reitin optimointia käyttää uudestaan lukemalla tallennettu tiedosto.

Mallin tässä vaiheessa määritellään vain lyhin mahdollinen reitti alueen rastereille, ei reitin kustannuksia. Kustannukset määritetään optimointimallin yhteydessä, koska hakkuukustannukset on määritelty hakkuuajankohdan mukaan ja maaperällä liikkumiselle ei ole määritelty erityisiä kustannuksia.

Reitin optimoinnissa käytetään siis tienvierusaineistoa, joka saadaan binäärimuotoisena matriisina. Jos ollaan sopivassa pisteessä tien vieressä, matriisin alkiona on 1 ja jos ollaan jossain muualla, alkiona on 0. Korkeuskartta ja sekä maaperäaineisto ovat säästä ja koneen kuormasta riippuvia estefunktioita, eli rasterista joko voi mennä läpi tai ei.

Reittien pituuksien laskeminen lähdetään suorittamaan matriisin vasemmasta yläkulmasta alkaen ja edetään rivi kerrallaan alaspäin. Jokaisessa alkiossa katsotaan ympärillä olevien pisteiden reitin pituus ja määritellään niistä lyhin reitti alkioon. Mallissa voidaan liikkua rastereista toiseen jokaiseen kahdeksaan suuntaan siten, että suoran liikkeen pituus sivulle, ylös ja alas on 1, kun taas diagonaalinen liike aiheuttaa $\sqrt{2}$:n lisäyksen. Tätä laskentarutiinia suoritetaan niin kauan, kunnes määritetyn reittimatriisin alkiot muuttuvat.

2.6.2 Ajankäyttö

Koska suoritettujen laskutoimitusten määrä on huomattava, kuluu laskurutiinin läpiviemiseen myös paljon aikaa. Laskenta-aikaa on pyritty pienentämään ohittamalla tiepisteet, sekä alkion vieruspisteet, joiden arvona on 0, eli pisteet, joille ei ole vielä määritelty reitin pituutta. Maaperädata on integroitu reitin laskemiseen kulkukelpoisuuden määrittävällä aputiedostolla, jossa on määritelty millä maalajilla voidaan kysyttynä vuodenaikana ajaa metsäkoneella. Aputiedosto palauttaa maaperädatan ja halutun vuodenajan perusteella matriisin, jonka alkioina ovat 1, jos rasterin läpi voi ajaa ja sakkomuuttuja M, jos maaperä estää kulkemisen. Tässä M on iso luku. Myös korkeusdatan integrointia reitin pituuksien laskemiseen on helpotettu nolla-apufunktiolla, joka laskee yhteen kaikki rasterin korkeusdatan rajoitukset ja

palauttaa alkiolle 1, jos alkioon pystytään kulkemaan aina ja 0, jos alkioon siirtymisellä on joku rajoitus. Rajoittamattomat pisteet voidaan ajaa huomattavasti pienemmällä vaivalla kuin rajoitetut pisteet, sillä rajoituksen löytyessä pitää jokaiselle kahdeksalle suunnalle erikseen tutkia, estääkö rajoitus kulkua. Tämä ei yksittäisenä laskutoimituksena vie paljoa enempää aikaa, mutta kyselyn tekeminen lisäsi testialueen datalle 8 sekuntia yhtä matriisin iteraatiota kohden. Testialueen datan laskeminen suoritettiin vuodenajasta riippuen $70 - 80$ iteraatiolla ja koska reitin pituudet pitää määrittää kaikille vuodenaikatyypeille, sekä kallistuskulmien takia myös tyhjälle ja täydelle koneelle, tulee reittimatriisi ajetuksi 12 kertaa. Näin pienetkin parannukset yksittäistä iteraatiota kohden kertautuvat ja nopeuttava vaikutus kokonaisuuden kannalta muodostuu huomattavaksi.

2.6.3 Reitti

Kuljettava reitti määritetään käyttämällä *min_around*- apufunktiota, joka palauttaa sen lähtöpisteen vieressä olevan alkion koordinaatit, jonka reitin pituus on lyhin, sekä reitin pituuden. Reittiä määrittäessä ajetaan *min_around* ja siirrytään palautettavaan pisteeseen. Tätä kiertoa ajetaan niin kauan kunnes saavutetaan tien vieressä oleva piste, eli kunnes *min_around*:in palauttaman alkion reitin pituus on 1. Pisteet, joiden kautta kuljetaan, tallennetaan vektoriin. Palautettava reitti on siis vektori, joka sisältää ne matriisin alkiot, joiden kautta lyhin reitti alkupisteestä tien viereen kuljetaan. Palautettava reitti ei siis ole yksikäsitteisesti lyhin, koska *min_around* palauttaa vain ensimmäisen etsimänsä minimikustannuksen. Samasta pisteestä voi siis löytyä useampikin saman pituinen reitti. Koska tämä ei vaikuta reitistä syntyvään kustannukseen annetuilla muuttujilla, ei siihen kiinnitetä tässä mallissa sen enempää huomiota.

2.6.4 Optimointimalli

Optimointimallin tavoitteena on löytää optimaalinen hakuaika leimikon puustokuvioille. Tämä saadaan määrittämällä leimikon puustokuvioille kelpoisuusnumerot, eli mikä on pienimmän hakukustannuksen hakuaika, jolloin puustokuvio voidaan hakea. Koska kelpoisuusnumero kertoo halvimman korjuuajankohdan puustokuviolle, on selvää, että kun leimikon sisällä kaikki puustokuviot, joilla on sama kelpoisuusnumeron kerätään samalla kertaa, saavutetaan halvin korjuukustannus. Tämä korostuu vielä siitä, että puiden hakeminen useampana vuodenaikana aiheuttaa koneiden siirrosta syntyviä

kustannuksia, joiden hinnaksi on annettu 100 € siirtymää kohden. Näin leimikon puustokuviot voidaan jakaa maksimissaan kuuteen luokkaan, josta taloudellisimmat korjuuajankohdat on huomattavasti helpompi optimoida.

Optimointiosassa käytetään annettuja hakkuukelpoisuustaulukkoja, jotka määrittävät halvimman vuodenajan kaikille maalajeille, puulajeille, puuston tiheydelle ja hakkuun tyypeille. Puustokuvion maalaji saadaan päämaalajin määrityksessä ja tiheys, puulaji ja puuston tiheys puustoaineistosta. Hakkuun tyyppi annetaan parametrina ennen optimoinnin ajamista. Mallissa ei oteta huomioon mahdollisia resurssirajoituksia, koska niitä ei ole ollut käytettävissä. Tämän kompensoimiseksi palautetaan kustannuksia minimoivan hakkuuaikataulun lisäksi myös vaihtoehtoisia hakkuuaikatauluja vähemmillä siirtymisillä.

2.6.5 Toteutus

Optimointimalli koostuu neljästä funktiosta itse optimoinnin ajotiedostosta, kustannukset määräävästä tiedostosta, sekä kahdesta aputiedostosta, jotka helpottavat aineiston käsittelyä. Lisäksi käytetään aikaisemmin käsiteltyjen reitin, päämaalajin, sekä puustoaineiston määrityksen tuloksia.

Aputiedostot Leimikointi ja Ykkösmatriisi auttavat aineiston käsittelyä ja selkeyttävät ohjelmointia. Puustomuunnoksessa on jo jokainen puustokuvio osoitettu omaan leimikkoonsa. Leimikointi-aputiedostolla palautetaan leimikot yhtenä listana, josta niitä on optimoitaessa helpompi kysyä. Ykkösmatriisi taas palauttaa kaikille annetun leimikon puustokuvioille halvimman mahdollisen hakkuuvuodenajan pääte- ja harvennushakkuulle. Se palauttaa myös listan, jossa puustokuvioille määritellään, onko hakkuu mahdollista kaikille säätyypeille sekä harvennus- ja päätehakkuulle. Hakkuukelpoisuudet ja kelpoisuusnumerot saadaan kysymällä puustokuvioiden päämaalaji, puulaji ja puun tiheys ja syöttämällä tiedot annettuun hakkuukelpoisuuden kertovaan matriisiin.

Kerro_kustannus - funktiossa määritetään korjuukustannukset yksittäisille puustokuvioille leimikon sisällä, sekä korjuukelpoisuusluokkien korjuukustannukset. Puustokuvion korjuukustannus saadaan summana kuljetusreitin pituudesta aiheutuneista kustannuksista sekä vuodenaikariippuvaisesta korjauskustannuksesta. Reitin pituus saadaan lukemalla aiemmin lasketusta reitinpituusmatriisista reitin pituus puustokuvion keskipisteelle. Saadussa datassa annettiin korjuukustannus kuutiometriä puuta kohden jokaiselle vuo-

denajalle, joka yhdistettynä puustodatasta saatavan hakkuukertymän kanssa määrittää puustokuvion korjuukustannukset. Hakkuukertymä kertoo kuinka paljon puustokuviosta on tarkoitus hakata puuta, eli harvennushakkuulle se on selkeästi pienempi, kuin päätehakkuulle. Reitin kustannus voidaan määrätä seuraavasti:

$$kustannus_{reitti} = x \cdot 0.289e/100m + 4.120,$$

missä x on reitin pituus. Tällöin voidaan lisäksi huomioida riippuvuus vuodenajasta kaavalla $kustannus(i) = kustannus_{reitti} + w_i hakkuukertym$ä, missä w_i on vuodenaikaan liittyvä kustannus. Koska korjuukelpoisuudet rajoittavat luokkien korjuuajankohtia, voidaan luokat esittää 6×6 alakolmiomatriisissa. Näin mahdollisia hakkuuajankohtien kombinaatioita on yhteensä vain 6! eli 720 kpl. Koska aineisto saadaan niin pieneksi, voidaan optimi ratkaista brute force -menetelmällä eli laskemalla korjuukustannukset jokaiselle mahdolliselle korjuuajankohtien kombinaatiolle ja lisäämällä niille siirtokustannukset, mikäli korjuuajankohtia on useampi.

$$\begin{bmatrix} c(1,1) & M & M & M & M & M \\ c(1,2) & c(2,2) & M & M & M & M \\ c(1,3) & c(2,3) & c(3,3) & M & M & M \\ c(1,4) & c(2,4) & c(3,4) & c(4,4) & M & M \\ c(1,5) & c(2,5) & c(3,5) & c(4,5) & c(5,5) & M \\ c(1,6) & c(2,6) & c(3,6) & c(4,6) & c(5,6) & c(6,6) \end{bmatrix}$$

Matriisissa yllä M on sakkomuuttuja.

Lopullista optimia varten tehtävänannossa on lisäksi rajoitettu leimikolla käyntien määrä enintään kolmeen. Tästä johtuen laskentaa saadaan nopeutettua lisää.

Luku 3

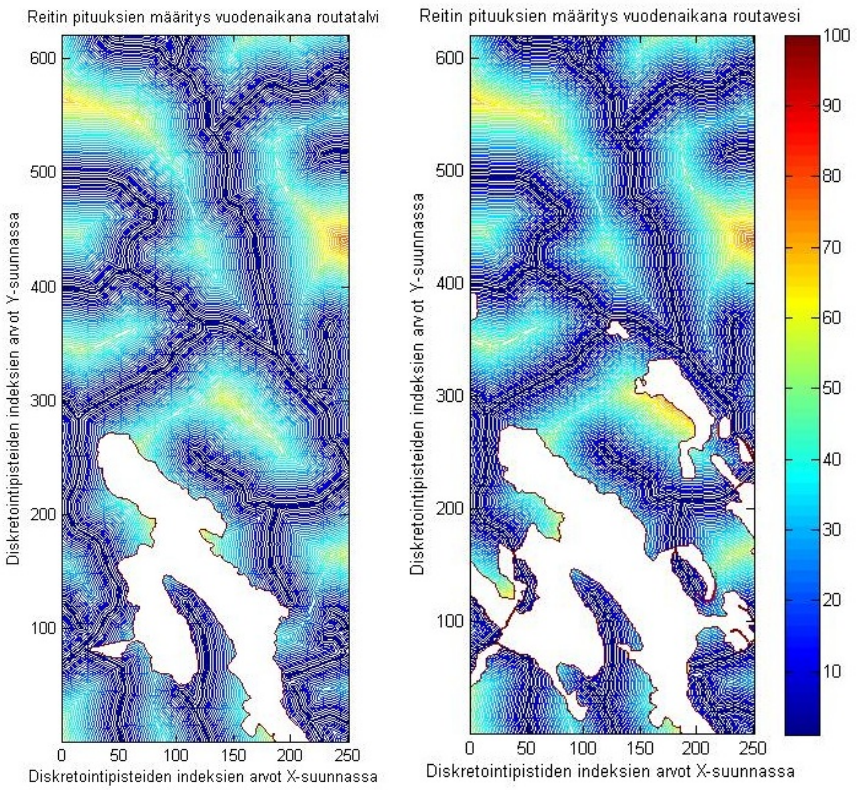
Tulokset ja analysointi

Suurin osa saaduista tuloksista on ulkopuoliselle lukijalle ei-mielenkiintoisia. Saimme reiteille muodostettua kustannuspinnan ja itse optimoinnin tulokset vaikuttavan jossain määrin järkeviltä.

Kustannuspinta vaikuttaa kuvia tarkkailemalla järkevältä. Kuvassa 3.1 oikealla näkyvä alue sisältää isoja aukkoja. Kuva on muodostettu olosuhteissa routavesi, joka on kaikkein haastavin ympäristö maastossa liikkumiseen. Siksi alueelle on muodostunut kohtia joihin ei voi kulkea. Kuvassa 3.1 vasemmalla näkyy sama maasto ajankohdalla routatalvi. Nyt osa routavesiolosuhteen alueista on muodostunut kelvollisiksi kulkualueiksi. Tämä on aiheuttanut samalla kustannuspintaan muutoksia ja nyt päästäänkin pienemmällä kustannuksella kulkemaan useasta paikasta.

Kun vertaillaan edellä mainittujen kuvien kustannuspintoja, huomataan että osa reittipisteistä on epäkelpoja ja tietyt alueet ovat saavuttamattomissa routavesiaikana. Tämä vaikuttaa siltä että kehitetty malli toimii kustannuspinnan kannalta oikein. Itse kustannusten laskentaan käytetään etäisyyden lisäksi kerrointa, jonka mukaan routatalvi on kalliimpaa aikaa kuin routavesi. Tämän vuoksi todelliset kustannukset eivät ole kuvien perusteella vertailtavissa. Reitin laskenta kustannuspinnasta näkyy kuvassa 3.2. Aluksi edetään niemen kärjestä niemen suuntaisesti, ja kun päästään lähemmäs tietä, käännytään kulkemaan suoraa linjaa tietä kohti.

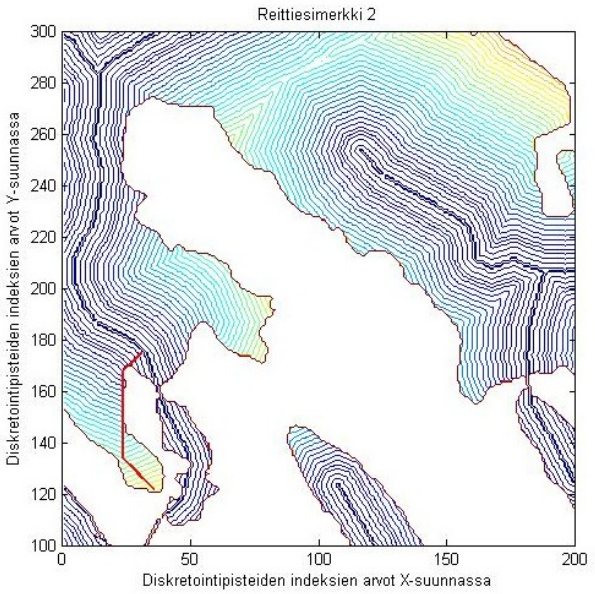
Leimikoinnin muodostaminen vaikuttaisi olevan toimiva molemmilla käytetyillä menetelmillä. Keskipisteestä säteittäin tietyn etäisyyden päästä valittavat puustokuviot muodostavat hyvin hieman harvemmasta harvennusalueesta yhtenäisiä kokonaisuuksia. Saamamme aineiston avulla naapurimäärityksin muodostetut leimikot päätehakkuun tapauksessa toimivat myös. Näissä tosin tulee helposti oudon tuntuisia leimikointeja kun ison kokonaisuuden lä-



Kuva 3.1: Vasemmalla routatalven ja oikealla routavesiajan olosuhteet Värit kertovat etäisyyttä lähimmästä tiestä.

pi kulkee tie. Tämä on tiedostettu ja hyväksytty tilaajan taholta. Kuvassa 3.3 näkyy projektin lopputuloksena muodostuneita leimikoita. Leimikot on merkitty eri väreillä selkeyden vuoksi ja valkoiset puustokuviot eivät kuulu yhteenkään hakattavaan leimikkoon.

Määrittämämme korjuuajankohdat näkyvät kuvassa 3.4. Kuvassa punainen väri vastaa korjuuajankohtaa routavesi, sininen lumitalvea ja violetti routatalvea. Eri puustokuviot leimikon sisällä asettuvat eri korjuukelpoisuusluokkiin. Kuvan leimikot ovat mielenkiintoisia, sillä lähes kaikki muodostuvat kahdesta korjuuajankohdasta. Koska nämä ajankohdat eroavat lisäksi runsaasti, niin optimaaliset korjuusuunnitelmat sisältävät monen leimikon osalta kaksi käyntiä alueella. Optimointimalli rakennettiin siten, että käyntikertoja leimikolla saa olla enintään kolme. Enintään kolmen käyntikerran joukosta etsitään se kombinaatio joka tuottaa kokonaisuudessaan pienimmän

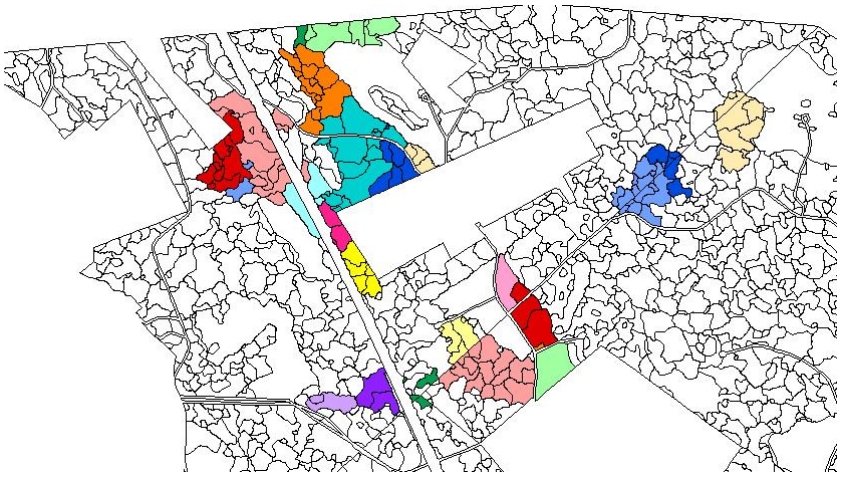


Kuva 3.2: Optimaalinen reitti lähimmälle tielle

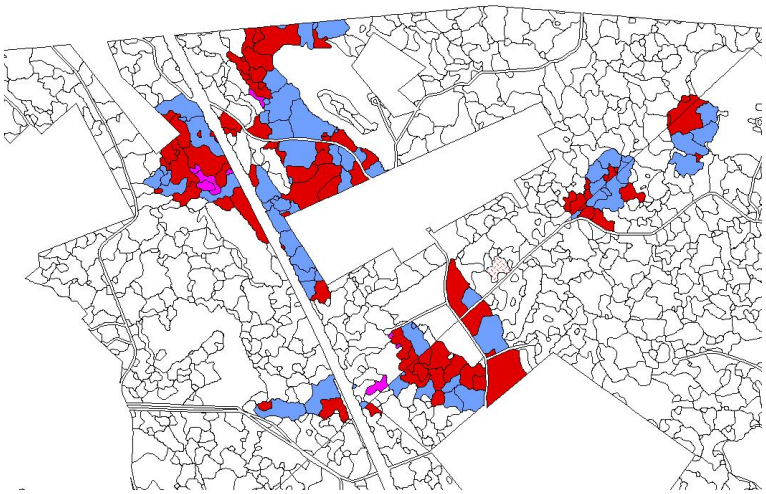
korjuukustannuksen.

Mallin kokonaisoptimin laskentarutiinin testaus jäi lopulta vähäiseksi, joten saadut kokonaiskustannukset ovat esitetty tässä pienin varauksin. Kuvan 3.3 leimikoille korjuusta aiheutuvat kokonaiskustannukset vaihtelivat menetelmämme mukaan $1400 - 44000$€ välillä keskiarvon ollessa 12000€. Mahdollisesti aineistoon on päässyt pilkkuvirhe jolloin järkevät luvut vaihelisivat $140 - 4400$€ väliltä. Toinen epäily koski reittien laskennan lukuarvoja, sillä puumääräisesti suurissa leimikoissa syntyy lähes 50000€ siirtokustannuksia.

Kustannukset saadaan leimikolle laskemalla kunkin puustokuvion hakkuukertymän siirto lyhintä reittiä pitkin tien varteen. Leimikon kustannus saadaan kaikkien leimikkoon kuuluvien puustokuvioiden kustannusten summana ja tähän lisätään vielä leimikolla käyntikerroista aiheutuva kustannus. Lukuarvojen epävarmuudesta huolimatta malli tuottaa leimikoittain optimaalisen korjuuajankohtien joukon ja sille kustannukset ehdolla että leimikolla saadaan käydä enintään kolme kertaa.



Kuva 3.3: Valmis leimikointi



Kuva 3.4: Leimikoinnin korjuuajankohdat

Luku 4

Yhteenveto ja johtopäätökset

Tämän työn tarkoituksena oli tuottaa laskentarutiini, joka tuottaa annetusta aineistosta valmiin hakkuusuunnitelman puustokuvioittain ja laskee optimaaliset kustannukset. Kaikkiin asetettuihin tehtäviin emme saaneet tyhjentävää vastausta. Aineistosta saatiin muodostettua leimikot pyydetyillä menetelmillä. Lisäksi saimme laskettua aineiston avulla pinnan, josta on helppo tuottaa minimikustannusreittejä. Myös reittien muodostaminen saatiin ainakin ulkonäöltään toimivaksi.

Kokonaisoptimin tuottaminen leimikolle antaa tuloksia, mutta lukuarvot eivät vaikuta järkeviltä. Tämän johdosta tuotos ei vastaa täysin sitä, mitä lähdettiin hakemaan. Käytetyt menetelmät ovat käyttökelpoisia, ja menetelmä on oikeansuuntainen. Ennen todellista käyttöönottoa menetelmä vaatii vielä tarkistuksia etenkin kokonaisoptimin laskentaosaan.

Tämän raportin palauttamiseen mennessä koko projekti ei kääntynyt Python-muotoon. Osa koodista toimii jo, mutta niiden käyttökelpoisuus on rajallinen ilman puuttuvia osia. Koska menetelmä on kuitenkin tulossa käyttöön, tarvitaan tähän osioon lisäksi jatkokehitystä. Arvioimme puuttuvaksi työmääräksi noin viikon osaavan ohjelmoijan käsissä. Tämän jälkeen projektin laskentarutiini on valmis lopulliseen tarkasteluun ennen käyttöönottoa.

Luku 5

Pohdintoja

Projektin alkuperäinen aikataulu oli tarkoituksella valittu hyvin tiukaksi. Aluksi aikataulu pysyi ihan kelvollisesti vielä hallinnassa, mutta ohjelmoinnin yhteydessä tapahtuneet lapsukset alkoivat viemään aikaa odotettua enemmän. Projektin kolmesta osiosta ensimmäinen, jonka tehtävänä oli muodostaa toimiva ohjelma kuljetuskustannuksen saamiseksi, onnistui hyvin aikataulussa. Osioon liittyi paljon tehtäviä, jotka oli helppo hajauttaa eri tekijöille.

Toinen osio, jonka tarkoituksena oli tuottaa malli leimikon korjuukustannusten ja korjuusuunnitelman laskemiseksi yllätti laajuudellaan. Aluksi sitä pidettiin vain yksinkertainen tehtävä, jossa kysytään leimikon kaikilta puustokuvioilta reitti kustannuksineen ja kerrotaan kustannukset puumäärällä. Tämän toteuttaminen vaati kuitenkin paljon työtä ja lopullisesti se valmistui vasta neljä päivää ennen tämän loppuraportin palautusta.

Kolmas osio, jonka toteuttamiseen liittyi aluksi jonkin verran epävarmuutta, oli mallin saattaminen Python-ohjelmointikielellä toimivaan muotoon. Tämän osion kanssa emme täysin tämän raportin palauttamiseen mennessä onnistuneet. Pääsyynä tähän oli, että ryhmän jäsenistä suurin osa ei ollut koskaan ohjelmoinut Pythonia. Python on ohjelmointikielenä helppo omaksua ja tämä osio ei olisi ollut täysin saavuttamattomissa. Aikaa alkoi vain loppuvaiheessa kulumaan paljon toisen osion virheiden etsintään, ja tämä kolmas vaihe jäi vähemmälle huomiolle. Tästä syystä emme myöskään pystyneet täysin saavuttamaan alkuperäistä tavoitetta toimivasta kokonaisratkaisusta leimikoiden hakkuukelpoisuuksien määrittämiselle. Sen sijaan tuotimme suuren osan ohjelmoinnista Python-kielellä, mutta lisäksi käyttöön jäi osioita jotka ovat ainoastaan MATLAB-ohjelmistolle tehtyjä. Nämä eivät va-

litettavasti ole keskenään yhteensopivia. Lopulta päädyimme palauttamaan tilaajalle sekä osion, joka toimii yhdistelmänä Pythonin ja MATLAB:in välillä että pelkästään Pythonilla toimivan keskeneräisen ohjelman.

Projektissa osien jakaminen ryhmäläisille toimi aktiivisimpien ryhmäläisten kanssa hyvin, mutta haastetta tuotti erään ryhmäläisen päivätyö. Tämän vuoksi yritimme pitää ryhmän toimista kirjaa ja kirjoittaa päiväkirjamaisesti etenemisestä. Resurssien jakamista haittasi lisäksi henkilöiden erilainen tausta. Koko työ oli vahvasti ohjelmointipainotteinen, ja puhdasta ajattelutyötä ja asioiden selvittelyä oli selvästi vähemmän. Ryhmän taidot ohjelmointipuolella olivat kuitenkin rajalliset ja paljon aikaa tuhrautui uusien taitojen opetteluun. Kuitenkin kokonaisuutena projekti olisi tuottanut lopputuloksen haluttuun muotoon, mikäli lopullinen ohjelmisto olisi saanut olla MATLAB tai sen ilmaisversio Octave. Projekti rakentui siten, että ensin muodostettiin toimiva malli, ja tämän jälkeen siirryttiin mallin kääntämiseen ohjelmistolta toiselle. Jälkimmäinen osa olisi vaatinut vain viikon tai kaksi lisää aikaa.

Projekti oli ryhmän mielestä kuitenkin opettavainen ja avartava kokemus. Todellisen projektin suorittaminen kymmeniä kertoja toistettujen harjoitustöiden asemesta oli miellyttävä poikkeus opiskelijan harmaaseen arkeen. Lisäksi ryhmä oppi projektin aikana reilusti lisää projektityöskentelystä.

Liitteet

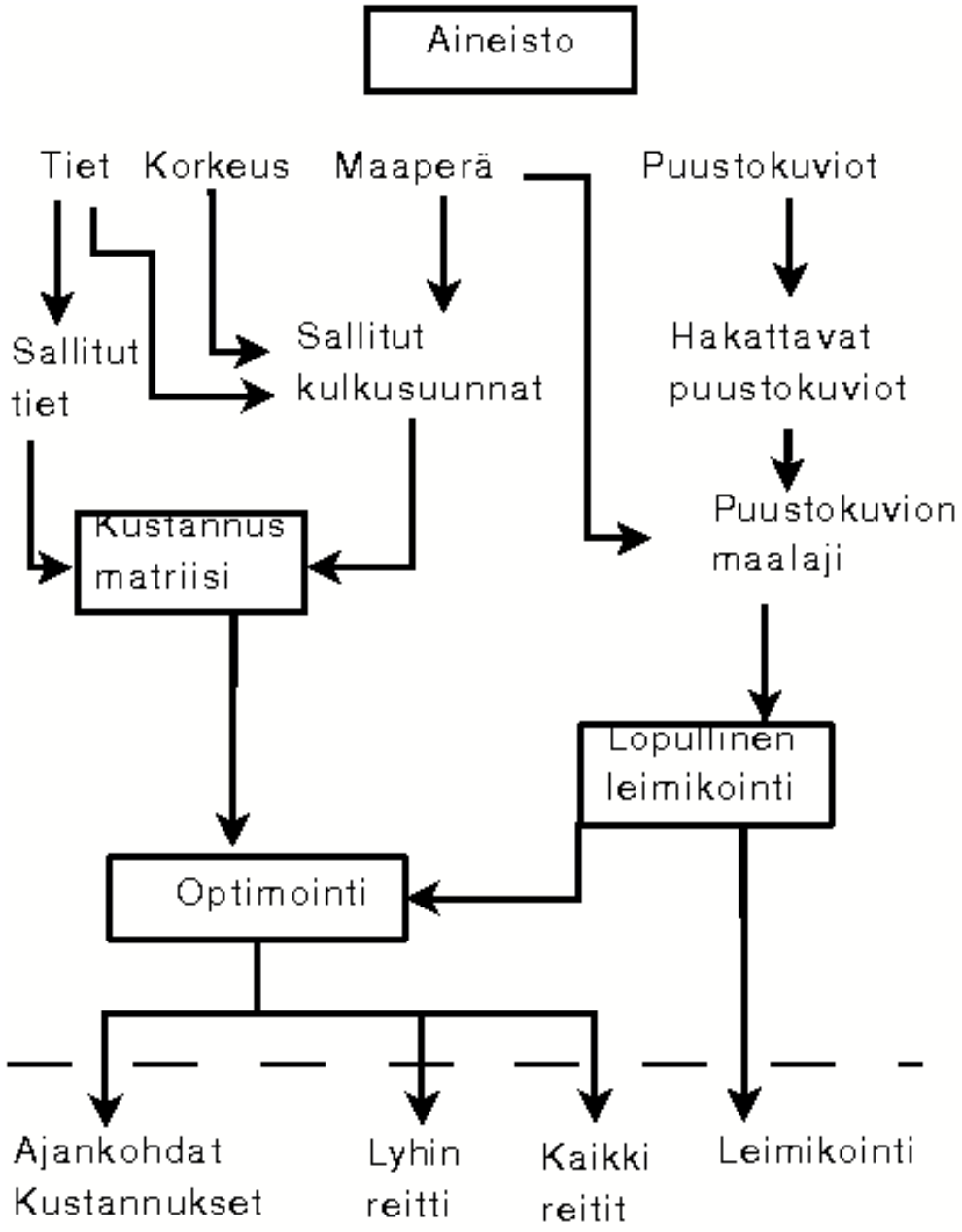
Liite A.

Kuvassa 1 on esiteltynä kuvassa koodin toiminnan pääosat. Aineisto otetaan ohjelmaan sisään ja se käsitellään kolmessa eri osassa kustannuspintaa varten. Lisäksi puustokuvioaineisto on omana osanaan. Tieaineistosta valitaan sallitut tiet, ja annetaan kulkusuuntiin kielletyt tiet. Korkeusaineisto ja maaperäaineisto muokataan kulkusuuntien laskentaa varten. Näiden yhdistelmä tuottaa kustannusmatriisin useiden välivaiheiden jälkeen. Kuvasta puuttuu maaperäaineiston lisänä suoalueelle puustokuvioista tuleva puumäärä ja sitäkautta kulkukelpoisuus.

Puustokuvioaineisto pysyy omana yksikkönään ja siitä valitaan käyttöön hakattavat puustokuviot. Näille määritetään päämaalaji. Tämän jälkeen puustokuvioista muodostetaan leimikointi joko säteittäin tai naapuriparametrina.

Kustannusmatriisi ja leimikointi yhdistyvät optimointivaiheessa. Nyt leimikkokohtaisesti optimoidaan korjuuajankohta. Palautuksena tulee paras ajankohtajoukko kun leimikolla saa käydä kolme kertaa.

Lopulliset palautusarvot sisältävät ajankohdat puustokuvioittain, kustannuksen leimikolle, reitit lähimmälle tielle sekä leimikoinnin.



Kuva 1: Kuvaaja koodin rakenteesta

Liite B.

Selvitykset yksittäisten funktioiden tehtävistä ja asetustiedostojen käytöstä.

SETUP.txt

SETUP.txt on aineistojen asetustiedosto. Siinä kullekin riville vastaavaa nimeä kohti asetetaan aineiston nimi. Tiedosto pilkotaan jatkokäsittelyssä ": -merkinnän avulla. Ennen ": "osaa olevaa ei huomioida ollenkaan, joten se jää vain tiedoston selitteeksi. Kaikki aineistot merkitään ilman päätettä. Poikkeuksena tähän on vain Jarno Leppäsen valmisteilla olevasta diplomityöstä peräisin oleva leimikointitiedosto, jonka yhteyteen tulee kirjoittaa myös pääte .csv. Käytetyt korkeuskartat tulee ilmoittaa Python-ohjelmointikielen ymmärtämässä listamuodossa. Tällöin yksittäinen korkeuskartta saisi nimen ['korkeuskartta']. Useamman korkeuskartan tapauksessa oikea järjestys kerrotaan esimerkiksi muodossa [['11','12'],['21','22']]. Rajoituksia alueen koolle ei ole.

parametrit.txt

Parametrit.txt on kaikkien yleisten parametrien asetustiedosto, jossa määritellään kaikki yksittäiset asetukset. Tämäkin tiedosto pilkotaan ": -merkinnän kohdalta. Rivit ovat tarkat, sillä selitettä ei tulkita mitenkään. Tiedatojen asettamisen kohdalla sallitaan pilkulla erotettuina tietyypit.

maalajin_luokitus.txt

Maalajin_luokitus.txt asettaa koodin käyttöön taulukossa pilkulla eroteltuna eri maalajien korjuukelpoisuusluokat. Nämä luokat määritetään eri hakkuutyypeille ja puulajeille erikseen. Aineistoa aletaan lukemaan riviltä 4.

muokkaus.py

Muokkaus.py on kaiken datan alustava muokkaaja, jonka tehtävänä on komentaa muita tiedostoja tekemään oma osuutensa. Se tarvitsee toimiakseen asetustiedostot SETUP.txt ja parametrit.txt. Lisäksi aineisto tarvitaan kansioon joka on asetettu SETUP.txt:n tiedostomuotoon.

maap.py

Maap.py on maaperäaineiston muokkaukseen käytetty ohjelma. Sen tehtävä on parsia maaperäaineiston .MID ja .MIF tiedot helpommin käsiteltävään muotoon. Parametreina tarvitaan maaperätiedoston nimi ilman päätettä, sa-

rake josta luetaan maaperän tiedot ja rivi jolta alkaa oikea aineisto .MIF tiedostossa. Alkuperäisessä aineistossa aloitettiin riviltä 19.

puustop.py

Puustop.py on puustokuvioaineiston alustavaan muokkaukseen rakennettu ohjelma. Se parsii aineiston .MID- ja .MIF-tiedostot sopivampaan muotoon. Asetuksina tämä tarvitsee pääpuulajin sarakkeelle, tilavuudelle, hakkuukertymälle ja lisäksi Jarno Leppäsen diplomityöstä lähtöisin olevan leimikkoaineiston sarakkeelle. Näiden avulla tiedosto palauttaa neljä Python-tiedostoa. Käsiteltävät puustokuviot joilla leimikointi ehdottaa toimenpiteitä, tuottavat kaksi tiedostoa. Toisessa on puustokuvion koordinaattipisteet ja toisessa puustokuvion tarvitut tiedot. Toiset kaksi tiedostoa sisältävät kaikki puustokuviot ja niiden tiedot. Näitä käytetään vain suoalueiden puumäärän tarkistamiseen. Näin saadaan tarvittava kulkukelpoisuuden tarkistus voimaan.

tiep.py

Tiep.py tekee tiedatalle saman kuin ylemmät tiedostot tekevät muille aineistoille. Tarvittuina parametreina edelleen ensimmäinen datarivi ja sarake jolta luetaan aineistosta tien tyyppi. Ulostulotiedosto on tiedata.py jolta voidaan kysellä muissa tiedostoissa tien koordinaattipisteitä ja tyyppiä.

maadata.py

Maadata.py muodostuu maap.pyn ajamisen tuloksena. Se on valmis tiedosto, joka voidaan hakea ajokoodiin. Se sisältää kaksi kutsuttavaa funktiota, info() ja maapera(). Info() palauttaa vektorin jossa jokainen alkio on vastaavan maaperäalueen maalaji. Maapera() palauttaa sisäkkäisen taulukon. Ensimmäinen sisennys vastaa maalajialuetta ja seuraava taso sisältää maalajialueen kulmapisteet.

puustodata.py ja puustodataall.py

Puustodata.py on puustop.pyn luoma tiedosto joka on muodoltaan sama kuin maadata.py. Tässä ei kuitenkaan ole kuin yksi kysyttävä luokka, puudata(). Tämä palauttaa puustokuvioittain kulmapisteet. Puustodataall.py on sama mutta sisältäen kaikki pisteet eikä vain aktiivisia puustokuvioita.

puustoinfo.py ja puustoinfoall.py

Puustoinfo.py on puustop.pyn luoma käyttövalmis data-aineisto. Se sisältää kysyttävän luokan nimeltään puuinfo() ja tämä palauttaa sisäkkäisen taulukon. Taulukossa jokaista puustokuviota vastaa yksi ylemmän tason tietue ja

puustokuvion tiedot löytyvät syvemmältä tasolta. Puustoinfoall.py on vastaava kaikille puustokuvioille.

tiedata.py

Tiedata.py on tiep.py-ohjelman luoma valmis aineisto. Tässä on kaksi kutsuttavaa luokkaa, tiet() ja info(). Tiet() palauttaa taulukon jossa on kunkin tien kulmapisteet ja info() palauttaa vektorin jossa on kunkin tien tyyppi.

InsidePolygon.py

InsidePolygon.py on luvan kanssa käytetty ulkopuolisen tekemä luokka. Sen tehtävänä on palauttaa tieto onko kysytty piste annetun monikulmion sisällä vai ei.