Määrittelydokumentti

Mitä algoritmeja ja tietorakenteita toteutat työssäsi

Aion toteuttaa lyhimmän reitin laskevan bruteforceen perustuvan algoritmin, sekä ainakin yhden branch-and-bound -algoritmin,. Lisäksi toteutan joitakin eri aikavaatimuksella toimivia ja erilaisia tuloksia antavia heuristiikkoja sekä täysin sattumaan perustuvan valinnan. Pyrin myös tekemään dynaamisen ohjelmoinnin avulla algoritmin, joka löytää parhaan mahdollisen ratkaisun mahdollisimman nopeasti.

Tietorakenteista toteutan ainakin pinon ja verkon, joita tarvitsen luultavasti kaikkien algoritmien toteutuksessa. Mahdollisesti teen myös keon ja jonkin puurakenteen, mikäli niitä tarvitsen. Yritän ehkä myös toteuttaa jonkinlaisen kekoa muistuttavan taulukkomaisen tietorakenteen, jonka avulla voisi nopeasti löytää lähimmän jäljellä olevan naapurin kullekin solmulle.

Mitä ongelmaa ratkaiset ja miksi valitsit kyseiset algoritmit/tietorakenteet

Tarkoituksena on vertailla kauppamatkustajan ongelman erilaisia ratkaisualgoritmeja tehokkuuden ja ratkaisun oikeellisuuden näkökulmasta.

Tietorakenteista verkon valinta oli tässä tapauksessa itsestään selvä, sillä kauppamatkustajan ongelma on käytännössä lyhimmän Hamiltonin kierroksen etsiminen täydellisestä verkosta. Pinon avulla saa tietoa reitistä tallennettua ja muokattua todella nopeasti, kun reittiä rakennetaan alusta loppuun järjestyksessä.

Joudun luultavasti toteuttamaan muitakin tietorakenteita, mutta ne tarkentuvat projektin edetessä. Jonkinlainen kekomainen tietorakenne lähimpien solmujen etsimiseen kuitenkin luultavasti tarvitaan.

Mitä syötteitä ohjelma saa ja miten näitä käytetään

Algoritmit saavat syötteenä verkon, joka on kuvattu taulukkomuodossa. Double-arvo kohdassa verkko[i][j] kertoo i:n ja j:n välisen etäisyyden. Satunnaisesti verkkoja arpova metodi arpoo valitun kokoiseen xy-koordinaatistoon valitun määrän pisteitä ja muodostaa näistä pisteistä täydellisen ja symmetrisen verkon. Verkon symmetrisyys ei ole ehtona algoritmien toimivuudelle, mutta verkon täydelliyys on.

Tavoitteena olevat aika- ja tilavaativuudet (m.m. O-analyysi)

Bruteforce-algoritmin aikavaatimus on O(n!). Branch-and-boundin avulla voidaan tehostaa bruteforce-algoritmia, mutta aikavaatimuksen pitäisi pysyä samana. Dynaamisen ohjelmoinnin avulla pitäisi olla mahdollista päästä lähelle aikavaatimusta O(2^n). Mikäli pyritään parhaaseen mahdolliseen ratkaisuun niin tämän alle ei nykytiedon valossa päästä.

Heuristiikkoihin ja approksimaatioon perustuvien algoritmien aikavaativuudet vaihtelevat välillä O(n)-O(2^n). Ajassa O(n) tuskin voi toteuttaa paljon satunnaista arvausta parempaa algoritmia. Ajassa O(n^2) saa luultavasti jo ihan

kohtuullisen arvauksen ja ajassa O(n^3) jo varsin hyvän. Hirveästi hitaampia algoritmeja tuskin on mielekästä toteuttaa.

Lähteet

Wikipedia