Kauppamatkustajaongelman ratkaiseminen geneettisellä algoritmilla

Markus Korpinen

Suurteholaskennan työkalut
Fysiikan laitos, Helsingin yliopisto

12. tammikuuta 2012

1 Johdanto

Kauppamatkustajaongelma on yksi esimerkki laskennallisesti huonosti skaalautuvasta ongelmasta. Kauppamatkustajaongelmassa pyritään löytämään usean eri kaupungin välille reitti, joka on lyhyempi kuin mikään muu reitti niin että palataan takaisin alkupisteeseen. Tehokkaita ratkaisuja lyhyimmän reitin löytämiseksi ei ole, mutta vähemmän tarkan ratkaisun kelvatessa on useitakin nopeita algoritmejä, kuten geneettiset algoritmit, jotka oikein käytettäessä ovat suhteellisen nopeita ja antavat tyydyttäviä vastauksia.

Geneettiset algoritmit pyrkivät ratkaisemaan ongelman niin että satunnaisesti valittujen ratkaisujen joukosta muodestetaan populaatio, jonka parhaat yksilöt lisääntyvät keskenään muodostaen aina vain parempia yksilöitä. Tämän lisäksi yksilöt mutatoituvat satunnaisesti, jotta pystyttäisiin lieventämään sitä tosiasiaa, että satunnaisesti valittu populaatio on usein vain hyvin pieni murto-osa kaikkien eri vaihtoehtojen joukosta. Monen generaation jälkeen voidaan poimia tulos, jonka pitäisi olla lähellä oikeaa tulosta, mutta ei kuitenkaan voida olla varmoja onko tulos tarkin mahdollinen.

Kauppamatkustajaongelman ratkaisemiseksi geneettisillä algoritmeillä muodostetaan aluksi satunnaisesti valittujen reittien joukosta populaatio. Tämän reittijoukon lyhyimpiä reittejä yhdistelemällä muodestetaan reittejä , jotka ovat myöskin lyhyimpiä tai parhaimmassa tapauksessa lyhyimpiä reittejä. Reitit lajitellaan aina lyhyimmyys järjestykseen ja uudet reitit syrjäyttävät joukon huonoimmat reitit. Tämän lisäksi mutaatio tapahtuu niin, että tietyllä todennäköisyydellä reittien jotkin kaupungit vaihtavat paikkoja keskenään.

Tässä työssä kauppamatkustajaongelmaa pyritään ratkaisemaan geneettisten algoritmien avulla käyttäen apuna c-ohjelmointikieltä.

2 Ohjelman rakenne

Ohjelma koostuu kolmesta osasta, jotka ovat pääohjelman **tsp_mpi** lisäksi moduulit **genetic** ja **cities**. Ohjelma avaa tiedoston, lukee kaupungit, laskee kaupunkien väliset etäisyydet, luo satunnaisen populaation reiteistä sekä lajittelee, pariuttaa, ja mutatoi usean generaation ajan populaatiota. Lopuksi tulostetaan populaation vahvimmat yksilöt, eli lyhyimmät reitit.

2.1 Tsp_mpi.c

Pääohjelma kutsuu funktioita moduuleista **genetic** ja **cities**.

2.2 Cities

Moduuli kostuu tiedostosta cities.c, sekä sen header-tiedostosta cities.h. Cities.c sisältää seuraavat funktiot:

print_city tulostaa halutun kaupungin nimen ja koordinaatit komentoriville.

count_lines lukee tiedostosta kaupunkien lukumäärän. Funktio jättää huomiotta rivit jotka on merkattu #-merkillä, sekä tyhjät rivit.

read_cities lukee tiedostosta kaupungit koordinaatteineen ja tallentaa ne muistiin. Kaupunkien on oltava muodossa *nimi leveyspiiri pituuspiiri*. Leveyspiiri ja pituuspiiri luetaan asteissa.

to_radians muuttaa asteet radiaaneiksi.

distance ottaa parametreina lähtö– ja maalikaupungin ja palauttaa niiden välisen etäisyyden.

calculate_distances ottaa sisääntuloparametreina kaksiulotteisen taulukon, listan kaupungeista sekä kaupunkien lukumäärän. Funktio alustaa kaksiulotteisen taulukon distances ja laskee siihen kaikkien kaupunkin väliset etäisyydet sekä palauttaa paluuarvona kaksiulotteisen taulukon.

2.3 Genetic

Moduuli kostuu tiedostosta genetic.c, sekä sen header-tiedostosta genetic.h. Genetic.c sisältää seuraavat funktiot:

generate_random_population alustaa popupulaation ja reitit omiin tietotyyppeihinsä *Population* ja *Path*. Populaatio sisältää tiedon populaation lukumäärästä sekä osoitinmuuttujan taulukkoon reiteistä. Satunnaisesti valitut reitit tallennetaan tietotyypeiksi *Path* ja niiden mukana tallennetaan tieto reitin pituudesta.

calculate_fitness laskee yksittäisen reitin pituuden.

generate_random_combination ottaa parametreina reitin ja asetukset, varaa muistista riittävästi tilaa reitille ja luo satunnaisen reitin kaupungeista.

compare_population on pikalajittelua (**qsort**) varten muodostettu vertailufunktio.

3 käyttödokumentti

4 tulokset