Toteutusdokumentti
Kristian Wahlroos

Rakenne

Ohjelma on täysin kirjoitettu Java-kielellä ja se koostuu viidestä eri pääpakkauksesta, jotka ovat: io, käyttöliittymä, logiikka, main sekä util. Io-pakkauksessa sijaitsee Tulostaja-luokka, jonka tehtävänä on tulostaa kartta luettavaan muotoon tekstikäyttöliittymässä. Käyttöliittymässä sijaitsee Java Swing – GUI, joka toimii ohjelman graafisena käyttöliittymänä sekä Piirtaja, jonka tehtävän on piirtää karttaan polkua. Logiikassa sijaitsevat omat tietorakenteet kuten ArrayList, PriorityQueue sekä Jono, mutta myös itse A*-algoritmin toteutus, Dijkstra, sekä MT-AA* (Moving Target – Adaptive A*). Reitinhakuun liittyvät luokat ovat samassa pakkauksessa. Muut luokat ovat reitinhaun tukena kuten Analysoija, jonka tehtävänä on analysoida annettu kartta, Heurestiikka joka laskee heurestisen arvon reitinhaku-algoritmille, Solmu joka toimii reitinhaku-algoritmin verkon solmuina sekä Enum Ympäristömuuttuja, jossa sijaitsevat globaalit muuttujat, kuten lokaali äärettömyys sekä solmujen painot. Mainissa sijaitsee ohjelman pääohjelma ja lopuksi util kansiossa ovat muun muassa karttakuvat .bmp -muodossa sekä Kuva ja Piste luokka, jotka auttavat sekä GUI:ssa, mutta myös reitinhaku-algoritmissa.

MT-AA* sekä ohjelman toiminta

Saavutetut aikavaativuudet

Algoritmin aikavaativuus on O(|E| + |V| * log|V|) eli sama kuin Dijkstrassa. Tämä johtuu siitä, että algoritmin ydin ja pohja nojaa samaan ideaan, kuin Dijkstran algoritmi. Ainoa ero näiden kahden välillä kuitenkin on vakiokertoimet. A* (tai MT-AA*) käy solmuja paljon vähemmän, kuin Dijkstra ja parhaassa tapauksessa vaadittava solmujen määrä on vain 5-10% Dijkstrasta. Pahimmassa tapauksessa A*:n aloittaessa keskeltä huonetta ja maalin ollessa vaikeassa paikassa, voi A* käydä yhtä paljon solmuja läpi kuin Dijkstra. Tämä kuitenkin tapahtuu erittäin harvoin ja on hyvin epätodennäköisetä, että A*:n tehokkuus laskisi näin alas.

Useissa peräkkäin ajetuissa hauissa MT-AA* taas laskee entisestään A*:n vakiokerrointa jolloin uuteen hakuun tarvittavien analysoitavien solmujen määrä laskee entisestään.

Toteutusdokumentti
Kristian Wahlroos

Suorituskyky ja O-analyysivertailu

Suorituskyvyltään algoritmi MT-AA* on huomattavasti nopeampi, kuin perinteinen Dijkstra peräkkäin ajettuna käytännössä millä tahansa kokoisella kartalla. Se pystyy myös käsittelemään kokoa 100 * 1000 olevia karttoja noin sekunnissa, joissa solmujen määrä nousee jopa 100 000 solmuun.

Kartan pohjapiirustuksella on ja ei ole väliä. Tämä tarkoittaa käytännössä sitä, että vaikeat kartat tuovat algoritmia lähelle Dijkstraa, mutta avoimet ja luonnolliset taas saattavat olla hyvinkin nopeita MT-AA* suoritukselle ja analysoinnille. Hyvässä kartassa operaatioiden määrä voi jäädä hyvinkin alhaiseksi heurestiikan takia.

Ainot hitaudet algoritmiin tuovat tarkoituksella hidastettu MT-AA*, Javan oma metodi polun piirtämiselle sekä omien tietorakenteiden implementaatiot. Tarkoituksella hidastettu MT-AA* tarkoitus on tehdä mahdolliseksi sen, että käyttäjällä on aikaa klikata uusi maali algoritmille. Omien tietorakenteiden ajottainen hitaus johtuu, ettei niitä ole optimoitu toisin kuin Javan omat tietorakenteet, jolloin muistinkäyttö ja tätä kautta tilavaativuus kasvaa.

MTAA* vs A* vs Dijkstra

Minkälaisille syötteille ohjelma toimii hyvin? Miksi?

Minkälaisilla syötteillä ohjelma toimii huonosti? Miksi?

Toteutusdokumentti
Kristian Wahlroos

Puutteet ja parannusehdotukset

MT-AA* ei ole kaikista tehokkain algoritmi, koska se luottaa A*-algoritmiin. A* on osittain jopa ehkä hieman hidas algoritmi ja sitä paljon tehokkaampi on esimerkiksi JPS (Jump Point Search). Myöskin isoissa esim. 1000 * 1000 verkoissa ongelmaksi tulee se, että solmuja on liian turhan paljon jolloin analysoitavia solmuja voitaisiin ottaa mukaan vain esimerkiksi joka toinen, joka tehostaisi huomattavasti analysointia sekä verkon luontia.

Pseudokoodin ongelmat?

Voiko algoritmin aikavaativuutta parantaa lisäämällä tilavaativuutta? Miten?

Raportit

Lähteet

- http://idm-lab.org/bib/abstracts/Koen07e.html
- http://en.wikipedia.org/wiki/D*
- http://en.wikipedia.org/wiki/Incremental_heuristic_search
- http://en.wikipedia.org/wiki/A*_search_algorithm