Polunhakualgoritmit ja -järjestelmät - aihealueen lyhyt kuvaus

Rodion "rodde" Efremov

13. syyskuuta 2014

Vuonna 1959 Dijkstra esitti polynomisessa ajassa toimivan algoritmin [1], jolla voi etsiä lyhimmän polun kahden mielivaltaisen solmun välillä painotetuissa verkoissa. Algoritmi kasvattaa lyhimpien polkujen puun lähtösolmusta aina siihen asti kunnes puu kohtaa maalisolmun. Kesti noin yhdeksän vuotta kunnes ongelma sai uuden ratkaisun: $A*$ [2] käyttää optimistisen (etäisyyttä aliarvioivan) heuristiikkafunktion, jonka myöten algoritmi "tietää", mihin suuntaan kannattaa lähteä päästääkseen maalisolmuun, ja, täten, tarjoaa paremman tehokkuuden kuin Dijkstran algoritmi. $A*$:n heikkoutena suhteessa Dijkstran algoritmiin on kuitenkin juuri heuristiikkafunktion tarve, jota, ongelmasta riippuen, ei aina pystytä tyydyttämään. Lisäksi, $A*$ sai kilpailevan algoritmin: ironisesti, juuri Dijkstran algoritmin *kaskisuuntaisesta* (engl. *bidirectional*) versiosta, joka kasvattaa kaksi lyhimpien polkujen puuta molemmasta päätesolmusta kunnes kaksi puuta kohtaavat. Ellei oteta kantaa käytettävään prioriteettijonoon, yksisuuntainen haku tekee $\sum_{i=0}^{N} d^i$ verran työtä siinä missä kaksisuuntainen tekee vaan $2\sum_{i=0}^{N/2} d_i$, missä lyhimmän polun kaarien määrä on N ja d on verkon solmujen keskiarvoinen aste. Kaksisuuntaisuus on kannattava nopeutus aina kun heuristiikkafunktiota ei ole mahdollista johtaa, eli Dijkstran algoritmi ja myöt leveyssuuntainen haku hyötyvät ko. järjestelystä. Kaksisuuntainen $A*$ ei kuitenkaan välttämättä aina tekee vähemmän laskentatyötä kuin yksisuuntainen versionsa, sillä on mahdollista että kaksisuuntaisen version kaksi hakupalloa ohittavat esteen verkossa eri puolilta, näin tehden enemmän työtä kuin yksisuuntainen $A*$.

Kaksisuuntaisuus ei kuitenkaan ole ainoa kirjallisuudessa esiintyvä nopeutustekniikka: vuonna 2003 Meyer et al. esittivät Δ-stepping algoritmin [4], joka 2^{19} solmun harvalla (engl. *sparse*) verkolla oli noin 3.1 kertaa nopeampi kuin optimoitu Dijkstran algoritmin, ja 16:n suorittimen koneessa rinnakkainen Δ-stepping oli 9.2 kertaa nopeampi kuin peräkkäinen versionsa. Toinen huomionarvoinen nopeutustekniikka on vuonna 2007 esitetty *Dijkstra with arc-flags*[5], jonka ideana on osittaa verkon solmut V komponentteihin $V_1, \ldots, V_n$ siten, että $\cup_{i=1}^{n} V_i = V$ ja $V_i \cap V_j = \varnothing$ kaikilla $i \neq j$, ja jokaiselle suunnatulle kaarelle liittää bittivektori, jonka i:s bitti on päällä jos ja vain jos kyseinen kaari on lyhimmillä polulla johonkin osion V_i solmuun. Ne verkon solmut u, joista lähtee kaari

solmuihin, jotka eivät ole samassa osiossa solmun u kanssa, sanotaan rajasolmuiksi, ja verkon esikäsittelyssä järjestelmä rakentaa lyhimpien polkujen puut jokaisesta rajasolmusta, jolloin on toivottava minimoida rajasolmujen määrää: *kd*-puualgoritmi on helppo toteuttaa, mutta ei onnistu minimoimaan rajasolmuja niin hyvin, kuin METIS [3], joka ei edes tarvitse solmujen koordinaatteja kuin *kd*-puu. Yhdessä kaksisuuntaisuuden ja kahden tason METIS-osituksen kanssa – jossa myös jokainen osio on osioitu pienempiin solmujoukkoihin – Möhring et al. raportoivat keskimäärin yli 500 kertaa nopeammat polunhakuoperaatiot.

Viitteet

[1] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[2] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2):100–107, 1968.

[3] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, December 1998.

[4] Ulrich Meyer and Peter Sanders. Delta-stepping: A parallel single source shortest path algorithm. In *Proceedings of the 6th Annual European Symposium on Algorithms*, ESA '98, pages 393–404, London, UK, UK, 1998. Springer-Verlag.

[5] Rolf H. Möhring, Heiko Schilling, Birk Schütz, Dorothea Wagner, and Thomas Willhalm. Partitioning graphs to speedup dijkstra's algorithm. *J. Exp. Algorithmics*, 11, February 2007.