Aalto-yliopisto
Perustieteiden
korkeakoulu

Dynaaminen verkko-optimointi karttasovelluksessa

(valmiin työn esittely)

Juhani Sipilä

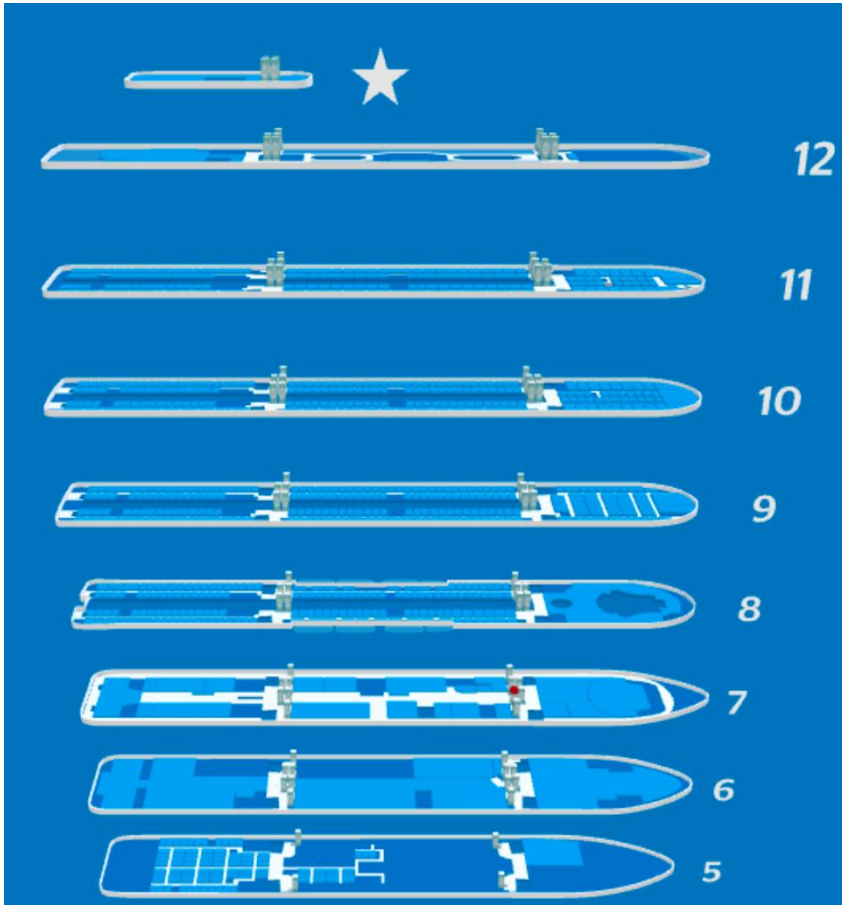
1.12.2016

Ohjaaja: *Prof. Kai Virtanen*

Valvoja: *Prof. Kai Virtanen*

Systeemianalyysin
laboratorio

Karttasovellus



- Kartta- ja reitityssovellus
  - Ohjaa asiakkaita kauppakeskuk-sissa, risteilyaluksilla, ...
  - Web-pohjainen (JS/PHP)

Työn tavoite

- Kehittää karttasovellukseen Dijkstran algoritmia käyttävä reitinhakualgoritmi

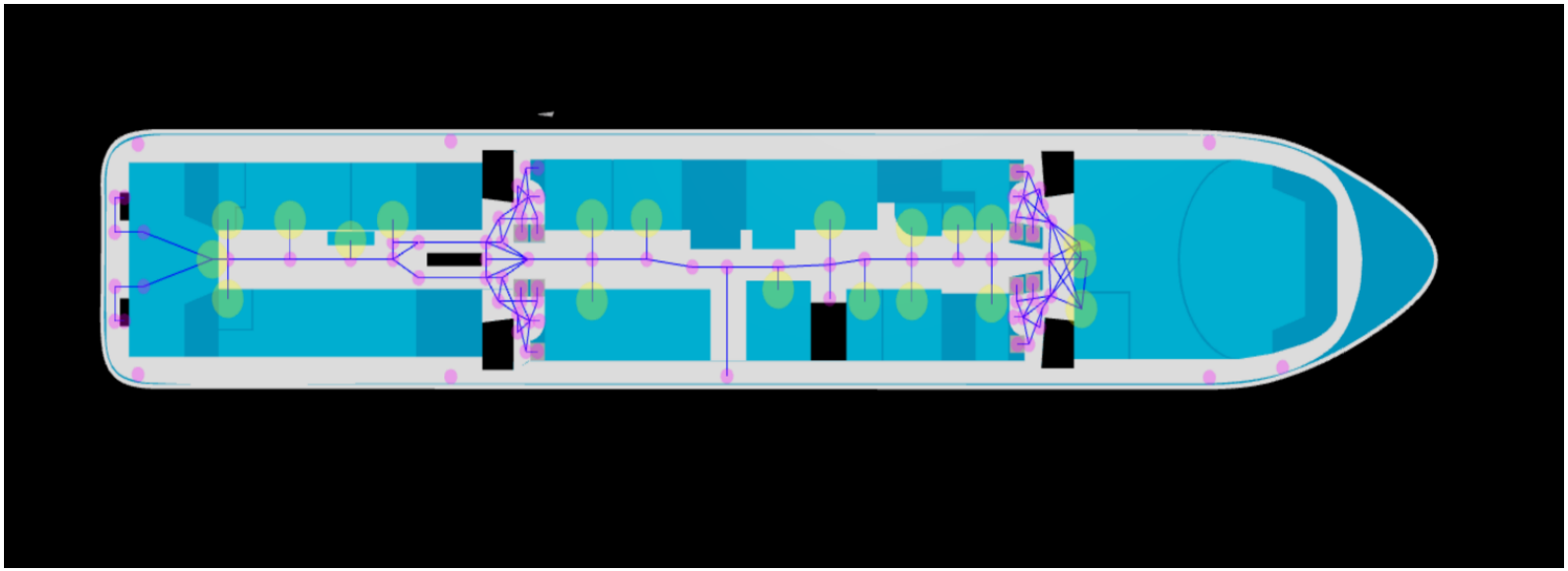
- Muodostaa lineaarinen malli, jolla voidaan ottaa käyttäjän preferenssejä huomioon kerrosten välisessä liikkumisessa

- Optimoida Dijkstran algoritmia
  - Optimoidun algoritmin tietorakenteena käytetään binäärikekoon perustuvaa minimiprioriteettijonoa
  - Optimoinnin tavoitteena parantaa algoritmin suoritusnopeutta ja karttasovelluksen käytettävyyttä

Reititykseen tarvittava verkko

- Verkko muodostuu solmuista ja niitä yhdistävistä poluista
- Jokaiselle polulle määritelty
  - Kustannus
  - Esteettömyys-tila
- Polkujen kustannuksia mahdollista muokata
  →Verkon mahdollinen epäeuklidisuus

Reititykseen tarvittava verkko

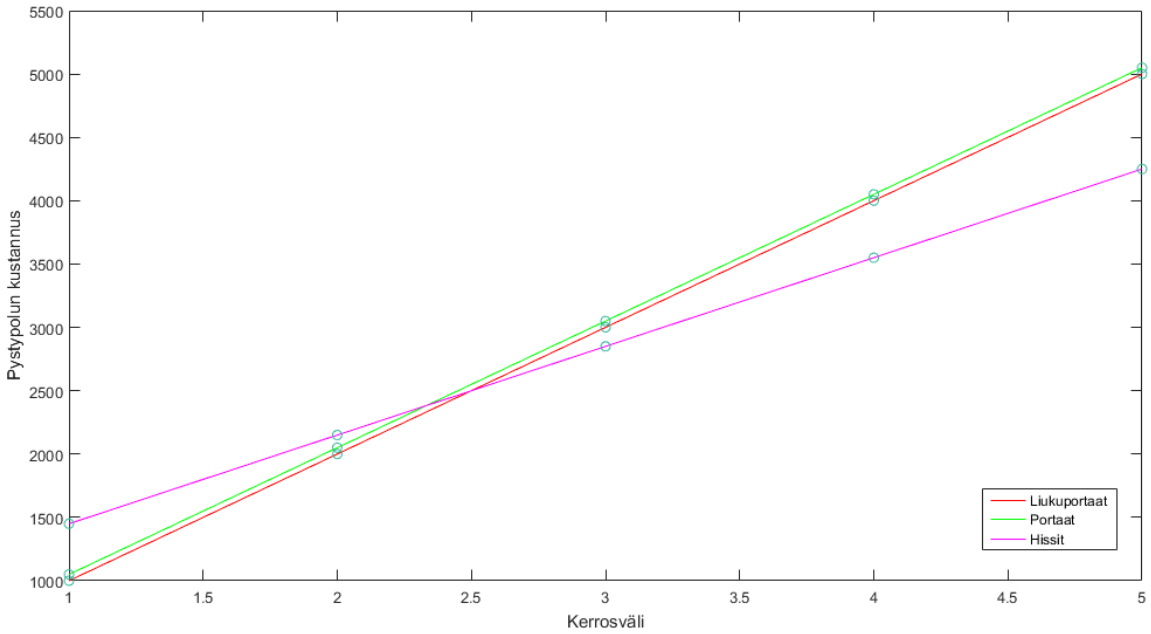


Lineaarinen malli

- Mahdollistaa käyttäjien tai kartan omistajan preferenssien huomioimisen kerrosten välisessä reitinhaussa
  - Pyörätuolilla liikkuva käyttäjä tarvitsee esteettömän reitin (hissi)

- Lineaarinen malli määrittää kerrosten välisten pystypolkujen kustannukset
  - Jos kerrosväli yli x-kerrosta: reititys hissien kautta
  - Jos kerrosväli alle x-kerrosta: reititys (liuku)portaita pitkin

Lineaarinen malli

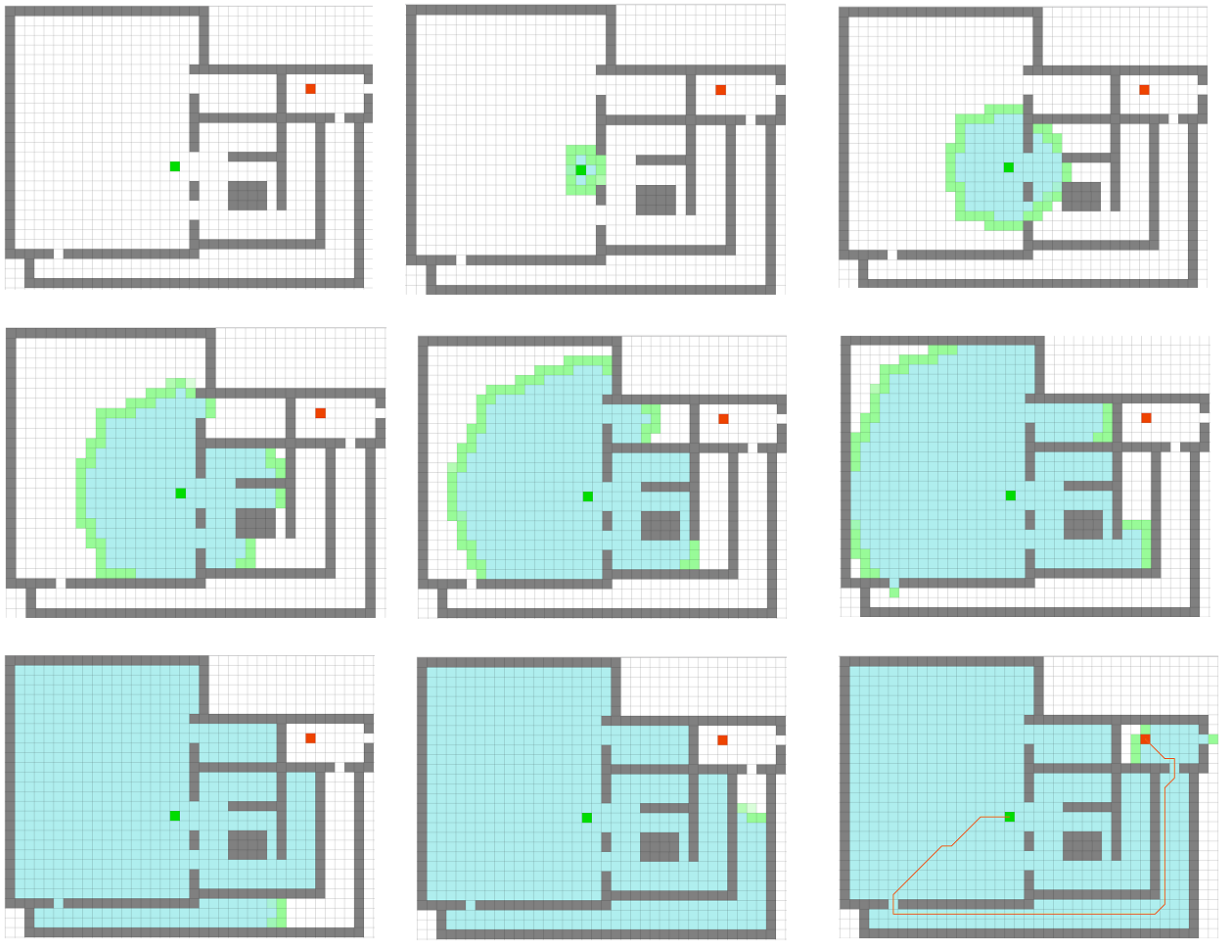


$$y_1(x) = Ax$$
$$y_2(x) = Ax + k^*$$
$$y_3(x) = Bx + (A - B)p^*$$

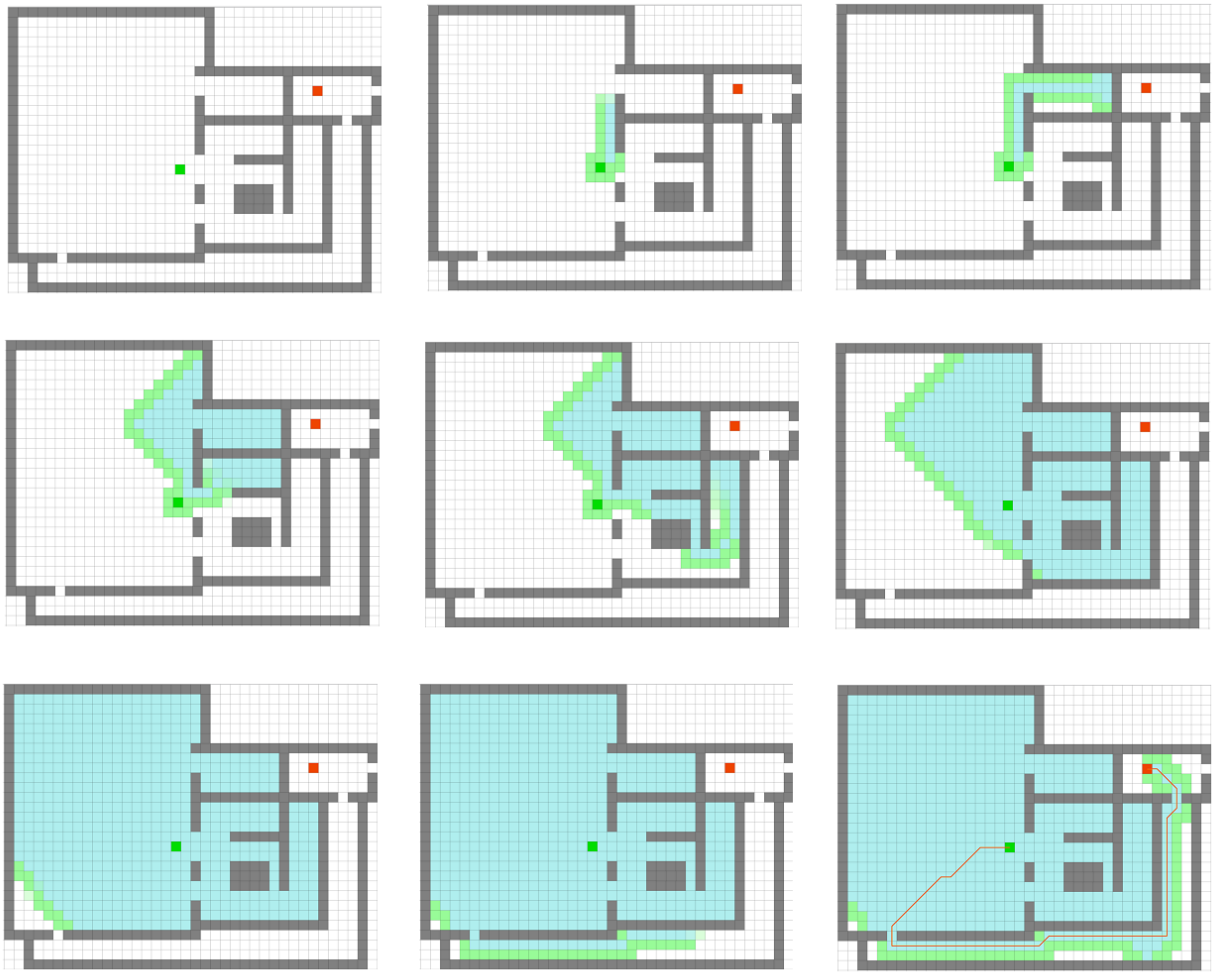
$$1 \leq x \leq N - 1, x \in \mathbb{N}$$

- Määrittää kerrosten välisten pystypolkujen kustannukset
- Tarvitaan:
  - Preferoitu kerrosväli (p*)
  - Porras vs. liukuporras vaihtoehtojen preferenssi (k*)
  - Yksittäisen kerrosvälin kustannukset eri vaihtoehdoilla (A & B)

Dijkstran algoritmin toiminta

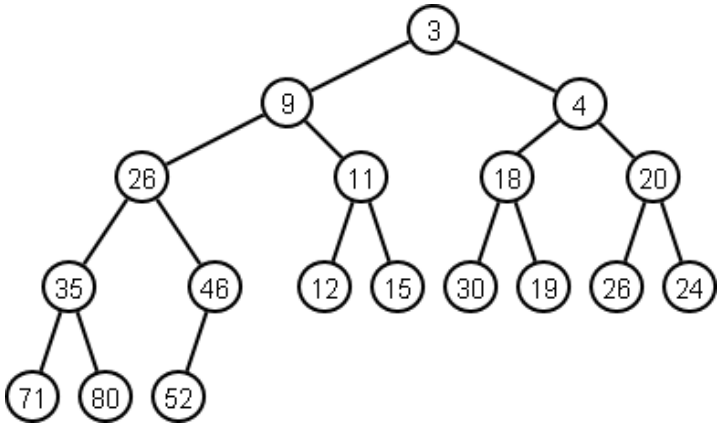


Best-first search (BFS) algoritmin toiminta



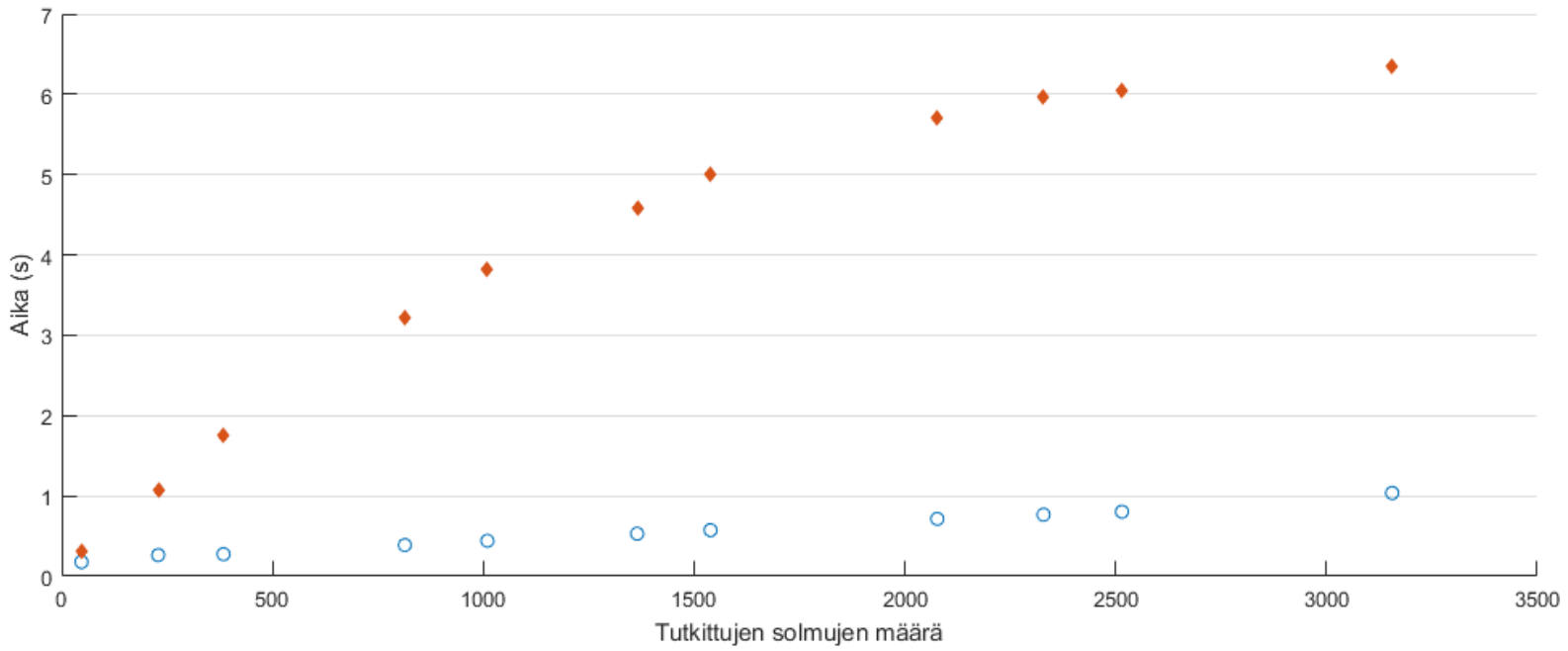
Dijkstran algoritmin optimointi

- Joka iteraatiolla joudutaan etsimään seuraava lyhimmän reitin tuottava indeksi
  - Järjestämätön taulukko käydään läpi joka iteraatiolla lineaarisessa ajassa
  - Minimiprioriteettijonossa seuraava indeksi löytyy binääripuun juuresta vakioajassa, puun uudelleenjärjestys vie logaritmisen ajan

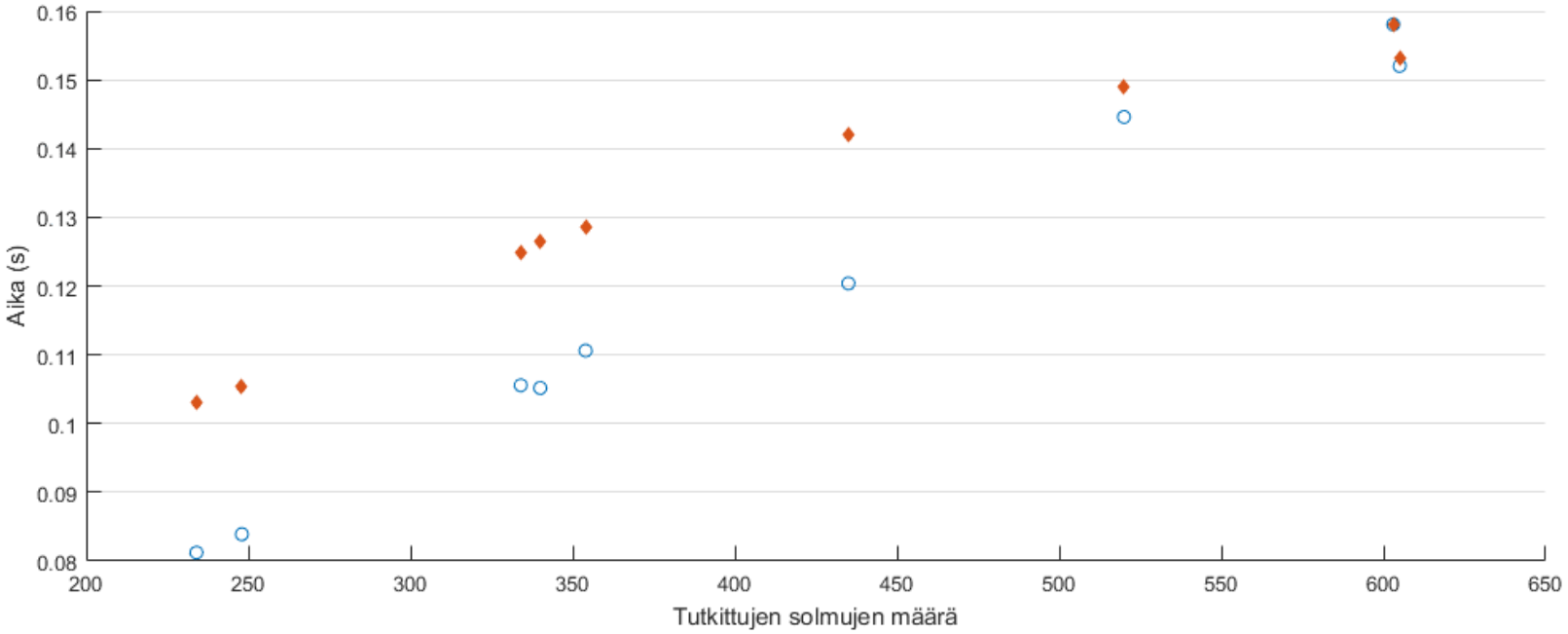


Algoritmi	Worst-case suoritusaika	Keskimääräinen suoritusaika
Dijkstra	$O(V^2)$	$O(V^2)$
Dijkstra binäärikeolla	$O(Elog(E))$	$O(E)$
Dynaaminen ohjelmointi	$O(EV)$	$O(EV)$
Bellman-Ford-Moore	$O(EV)$	$O(E)$

Tuloksia

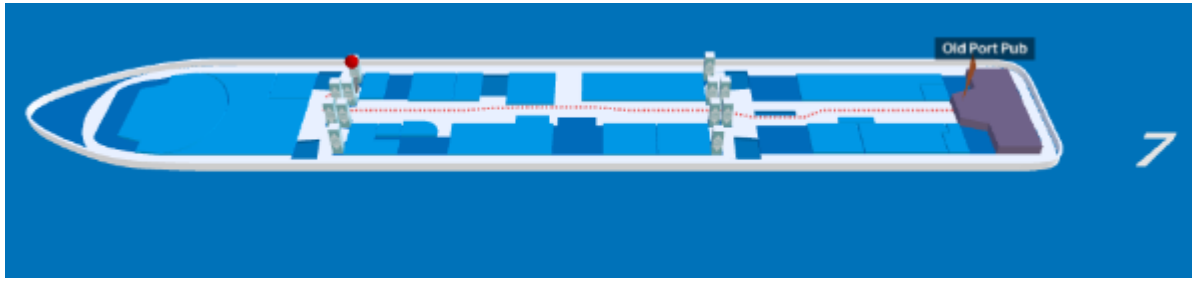

Aika (s)
Tutkittujen solmujen määrä
0
1
2
3
4
5
6
7
0
500
1000
1500
2000
2500
3000
3500

Tuloksia


0.16
0.15
0.14
0.13
0.12
0.11
0.1
0.09
0.08
Aika (s)
200
250
300
350
400
450
500
550
600
650
Tutkittujen solmujen määrä

Tuloksia

- Algoritmin suoritusaika saatiin pienennettyä moninkertaisesti
  - Optimoitu algoritmi toimi testatuissa verkoissa 1-9 kertaa nopeammin
  - Isoissa verkoissa algoritmien suoritusaikojen erot suuria
  - Pienemmissä verkoissa suoritusajat merkittävästi lähempänä toisiaan
- Karttasovelluksen käytettävyyttä saatiin parannettua
  - Laskennallisesti vaikeimmankin reitin löytyminen alle sekunnissa



Johtopäätökset

- Minimiprioriteettijonosta merkittävä parannus Dijkstran algoritmin suoritusnopeuteen
  - Keskimäärin 6-kertainen nopeutus
- Lineaarisella mallilla saadaan otettua huomioon kerrostenvälisen liikkumisen preferenssit
- Verkon epäeuklidisuudesta seuraa, että heuristista funktiota vaikea mallintaa
  - Heuristiikalla voitaisiin parantaa tulosta (A*)