Testausraportti

Otto-Ville Raitolahti

Lyhimpiä polkuja voi etsiä painollisissa ja painottomissa verkoissa. Yksinkertaisin verkko on painoton. Painottomana verkkona voi kuvata esimerkiksi labyrintin.
Mikä algoritmi kannattaa valita kuhunkin tilanteeseen?

Testaamme eri algoritmien toimintaa bmp-tyyppisillä kuvilla. Mustavalkoiset kuvat ovat painottomia verkkoja, värilliset painollisia. Eri värit kuvaavat eripainoisia maastoja.

Käytämme testaamisessa Suorituskykytesti-luokan testaaPainotonVerkko ja testaaPainollinenVerkko -metodeja. Metodit käyttävät testaaKentta -metodia. TestaaKentta hakee annetun kuvatiedoston ja muodostaa siitä verkon. Verkossa haetaan polku vasemmasta yläkulmasta vasempaan alakulmaan annetulla algoritmilla. Jokainen algoritmi lopettaa löydettyään polun. Metodi mittaa polunhakemiseen kuluvan ajan ja palauttaa sen. Tämän jälkeen algoritmin löytämä ratkaisu ja läpikäydyt solmut voidaan piirtää uuteen kuvaan. Testit ovat toistettavissa milloin vain. Javan roskienkeruu on ohjattu tehtäväksi juuri ennen ajanottoa. Testit toistetaan kymmenen kertaa ja tuloksista otetaan keskiarvo.

A* -algoritmi käyttää manhattan-heuristiikkaa.

Testit suoritetaan Intel® Core™ i5-3470 CPU @ 3.20GHz × 4 -suorittimella.

Painottomien verkkojen testaaminen

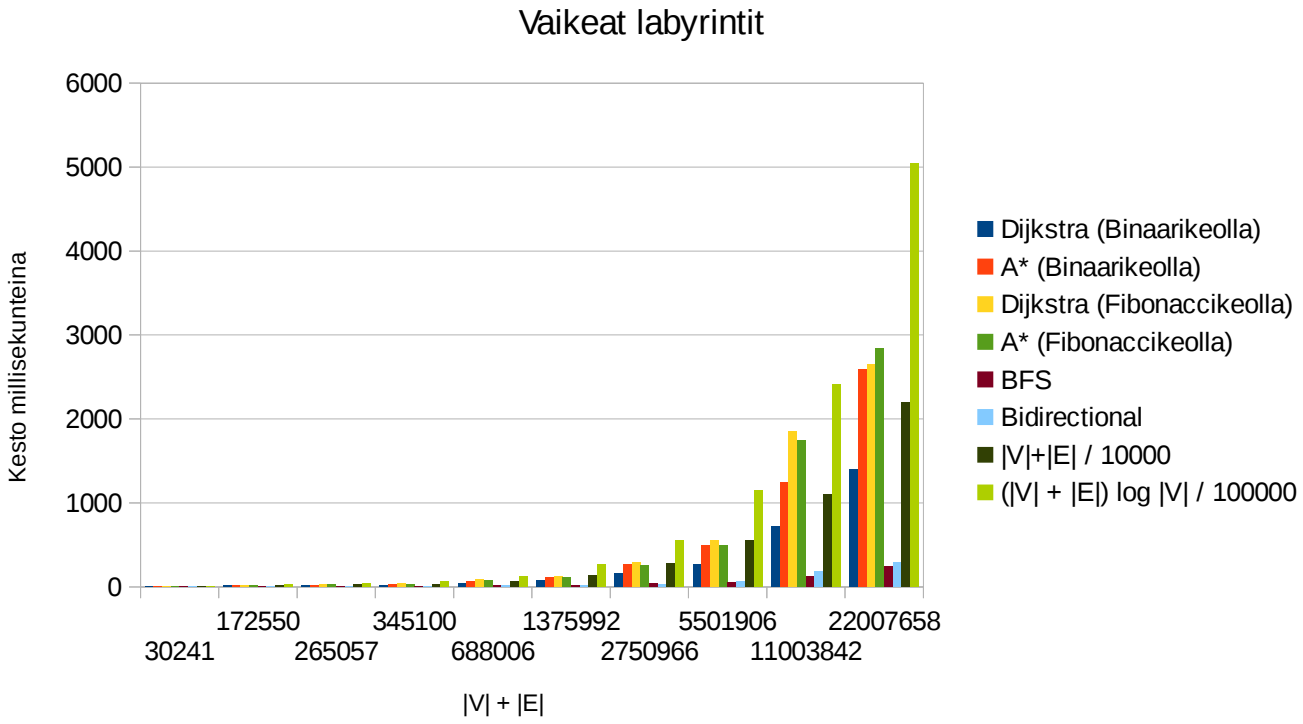
Kannattaako painottomissa verkoissa käyttää painollisissa verkoissa toimivia algoritmeja, kuten Dijkstra tai A* -algoritmeja? Painottomassa verkossa tavallinen leveyshaku löytää lyhimmät polut jokaiseen solmuun ajassa O(|V| + |E|). On odotettavaa, että leveyshaku toimii nopeammin. Miten Dijkstra ja A* suoriutuisivat binaarikekoa käyttäen? Entä miten ne suoriutuvat Fibonacci-kekoa käyttäen?

Algoritmeihin on mahdollista tehdä erilaisia optimointeja. Nämä optimoinnit usein parantavat algoritmin suorituskykyä vakiokertoimella, mutta aikavaativuusluokka pysyy samana. Bidirectional - algoritmit aloittavat alkusolmusta ja loppusolmusta ja yhdistävät polut, kun ne kohtaavat. Teoriassa aikaa pitäisi kulua vähemmän. Useista polunhakemisalgoritmeista voi tehdä Bidirectional -tyyppisen.
Vertaamme Bidirectional-leveyshakua tavalliseen leveyshakuun.

Testaamme algoritmien toimintaa painottomissa verkoissa erikokoisilla labyrinteilla. Labyrintissa valkoisiin ruutuihin voi kulkea ja mustiin ei ollenkaan. Aloitamme 100x100 kokoisella labyrintilla. Suurin labyrintti on noin 2000x4000 pikseliä.

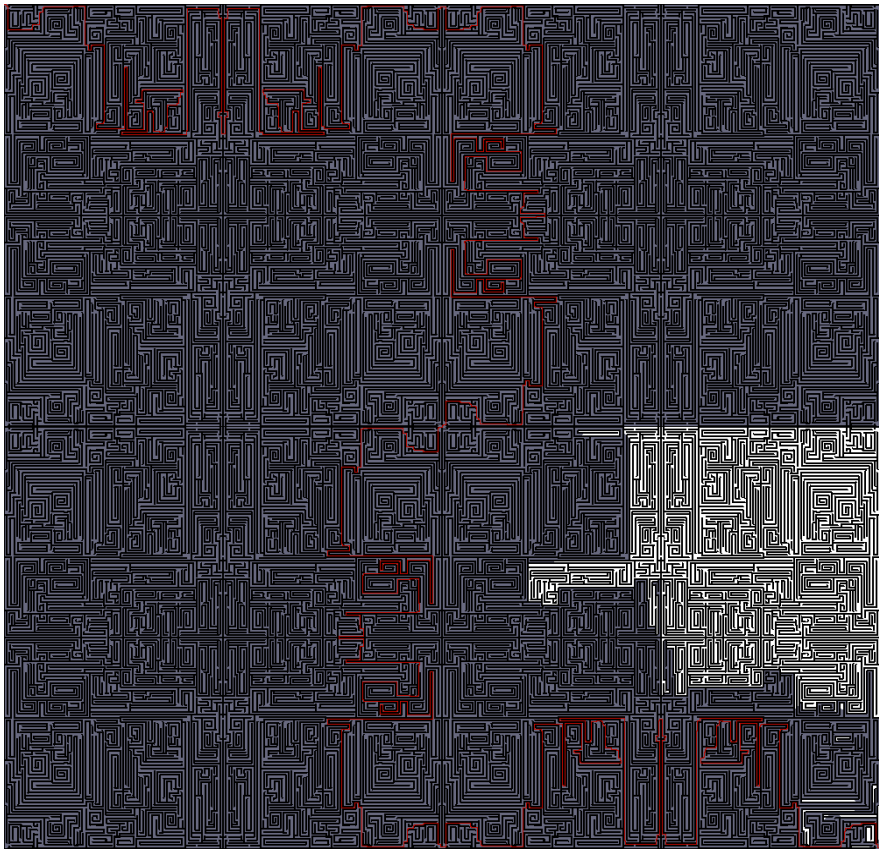
Testeistä saatiin seuraavat tulokset:

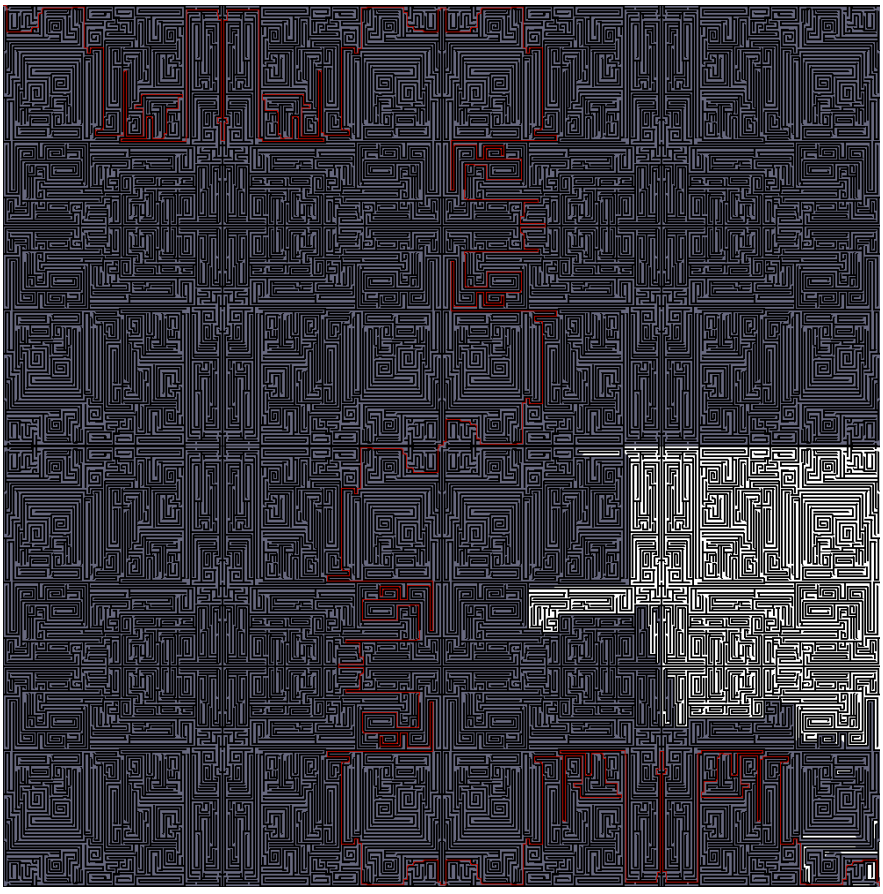
Solmujen määrä	Kaarien määrä	Dijkstra (Binaarikeolla)	A* (Binaarikeolla	Dijkstra (Fibonacci-keolla)	A* (Fibonacci-keolla)	BFS	Bidirectional
10000	20241	4	4	8	3	2	3
63001	109549	10	13	21	17	3	3
90000	175057	15	20	34	26	5	4
126002	219098	19	25	41	27	6	6
250000	438006	43	60	83	72	16	13
500000	875992	73	110	128	108	20	22
999000	1751966	159	265	287	248	38	35
1998000	3503906	263	492	548	492	55	66
3996000	7007842	712	1238	1851	1742	129	183
7992000	14015658	1393	2590	2655	2836	243	289

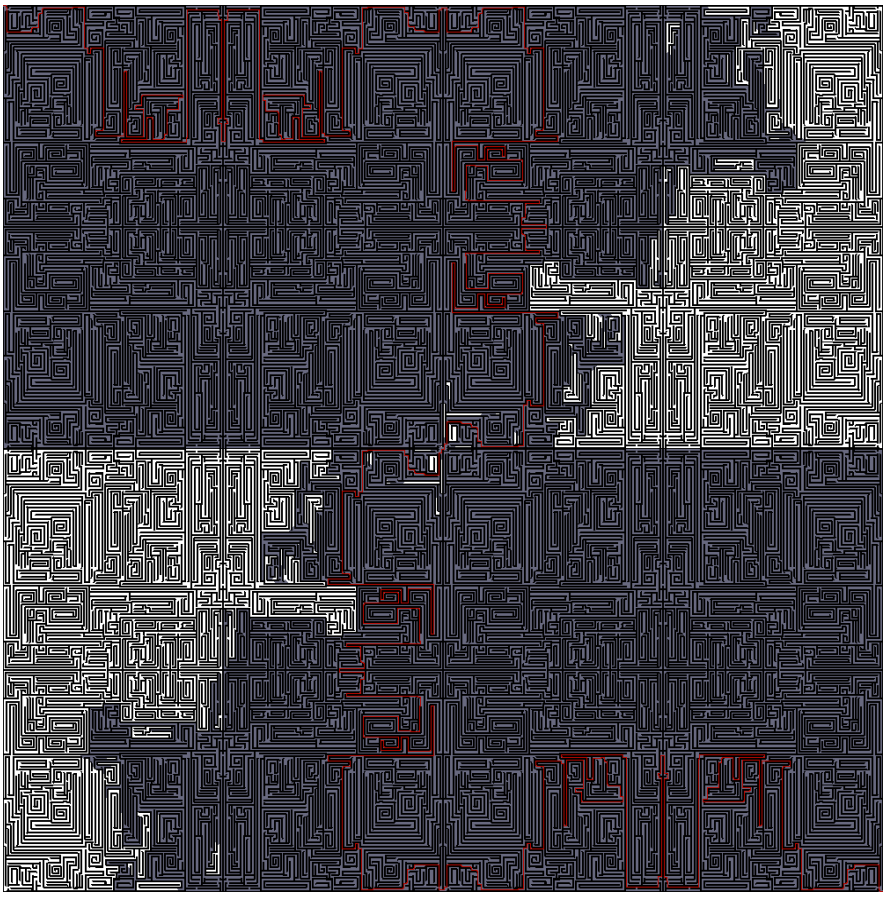


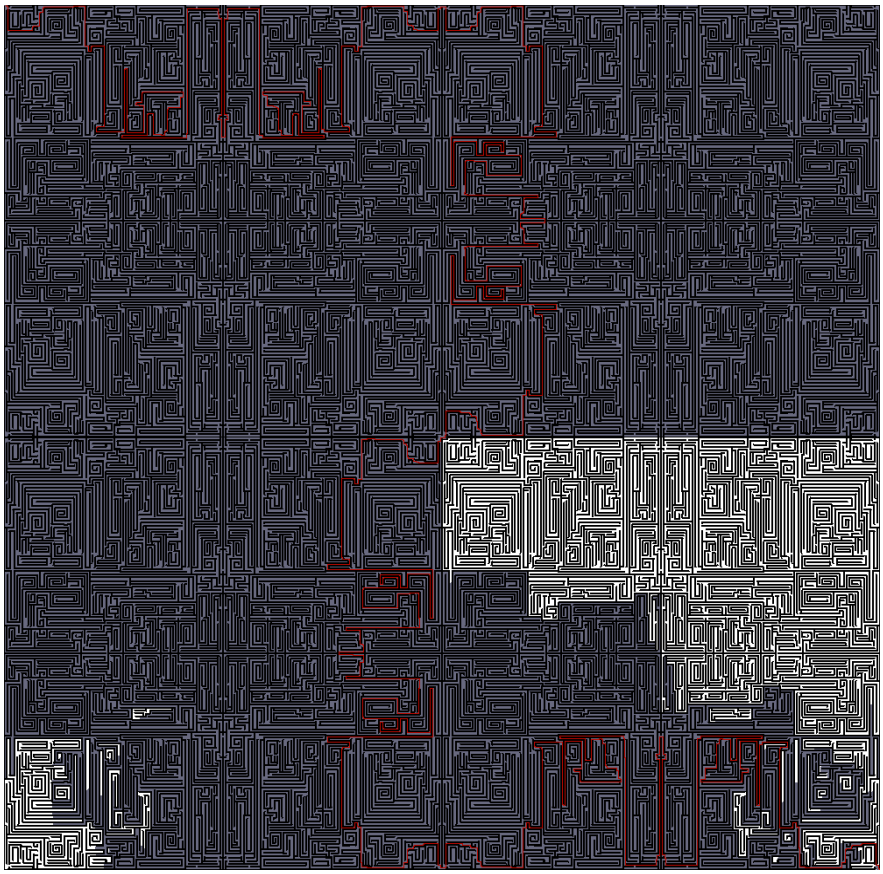
Kaikki optimoidut algoritmit ovat käytännössä hitaampia. Mistä tämä johtuu? A*-algoritmin heuristiikasta ei näytä olevan hyötyä tässä labyrintissa. A* ja Dijkstra jäävät kuitenkin molemmilla kekototeutuksilla (|V| + |E|) log |V| / 100000 -kuvaajan alapuolelle. Aikavaativuusluokat siis toteutuvat. Samoin BFS ja Bidirectional jäävät |V| + |E| / 10000 kuvaajan alapuolelle.

Tarkastellaan algoritmien piirtämiä ratkaisuja ja läpikäytyjä solmuja. Ylempi kuva on Dijkstran algoritmin piirtämä ratkaisu, alempi on BFS-algoritmin piirtämä ratkaisu.









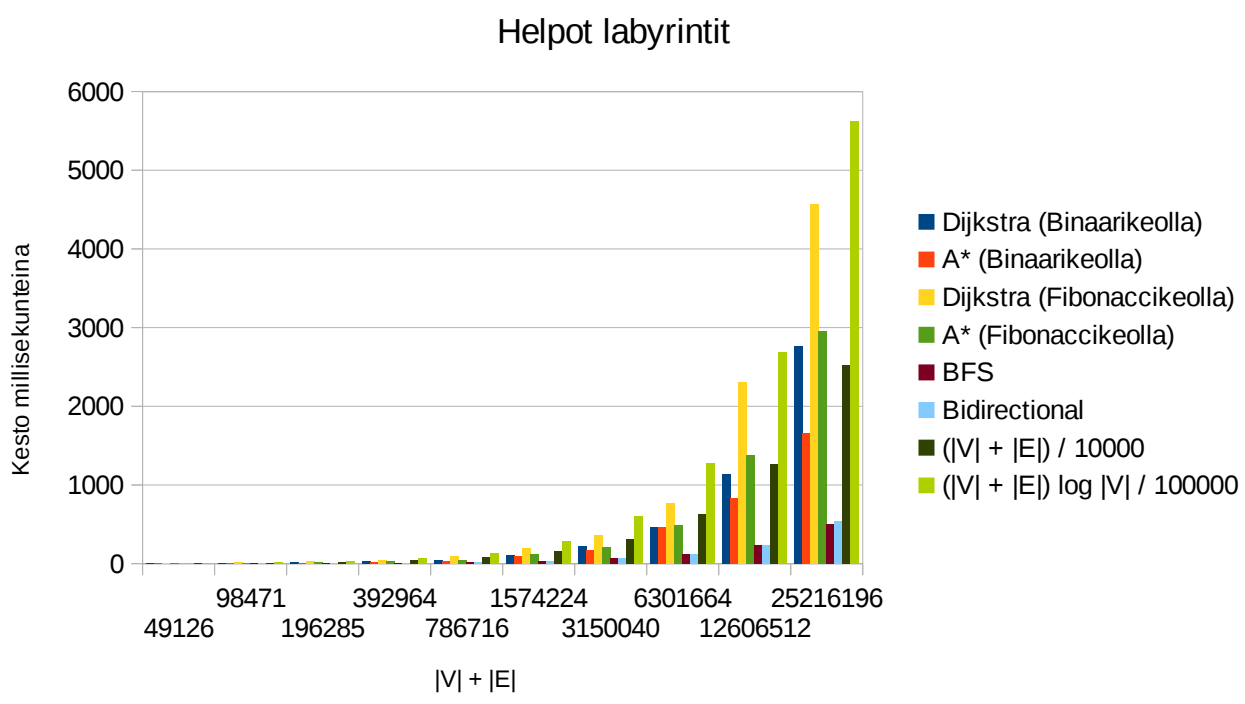
Ylempi kuva on Bidirectional-algoritmin piirtämä ratkaisu ja alempi on A*-algoritmin piirtämä ratkaisu.

Kuvissa ratkaisu on piirretty punaisella ja läpikäydyt solmut ovat harmaalla. BFS ja Dijkstra ovat käyneet läpi lähes samat solmut. BFS-algoritmin nopeus ei siis johdu siitä, että se kävisi läpi vähemmän solmuja. Bidirectional-algoritmi on käynyt läpi huomattavasti vähiten solmuja. Se ei kuitenkaan ole nopeampi kuin tavallinen BFS. Ilmeisesti algoritmilla on suuremmat vakiokertoimet ja optimointi ei näy näin pienillä aineistoilla.

A* on käynyt läpi lähes yhtä paljon solmuja kuin Dijkstra. Onko manhattan-heuristiikka huono tällaiseen ongelmaan? Millainen heuristiikka olisi parempi?

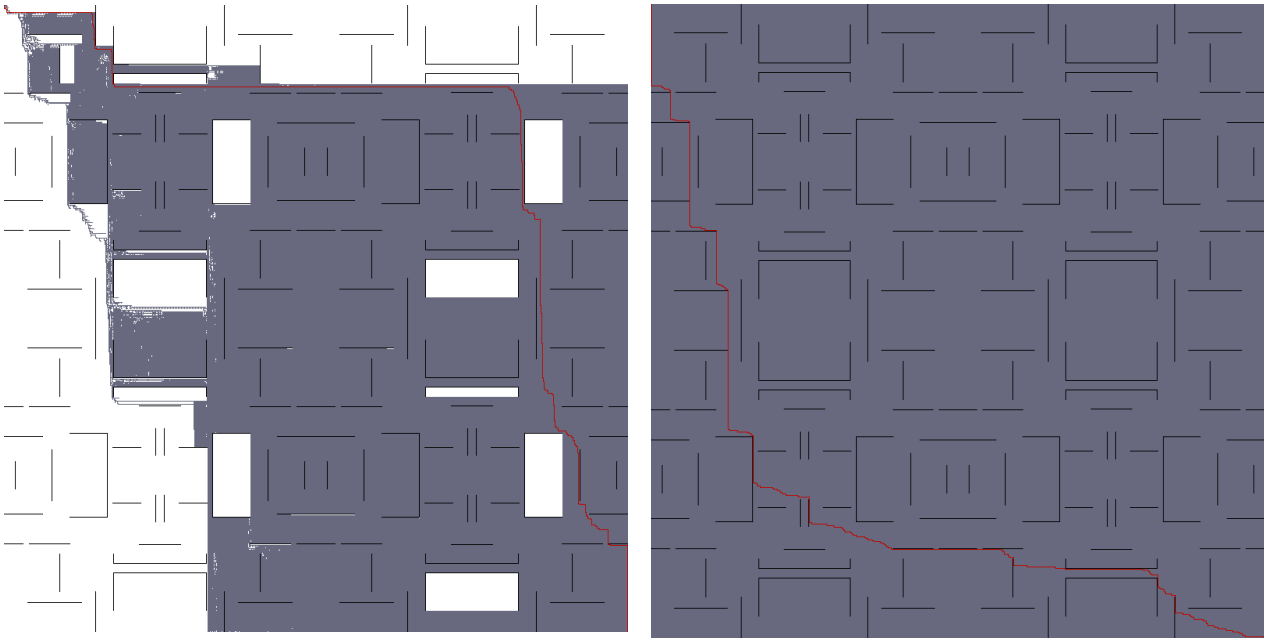
Testataan algoritmien toimintaa helpommissa labyrinteissa. Helpoissa labyrinteissa on miltei suora polku vasemmasta yläkulmasta oikeaan alakulmaan. Suoriutuuko A* nyt nopeammin?

|V| + |E|	Dijkstra (Binaarikeolla)	A* (Binaarikeolla)	Dijkstra (Fibonaccikeolla)	A* (Fibonaccikeolla)	BFS	Bidirectional
49126	5	4	9	5	2	2
98471	7	6	16	9	5	4
196285	13	7	28	16	6	4
392964	31	14	49	27	10	9
786716	44	26	98	48	19	19
1574224	104	90	197	118	33	31
3150040	216	167	358	201	65	64
6301664	459	468	766	491	119	124
12606512	1140	836	2302	1383	236	230
25216196	2764	1660	4566	2948	496	544

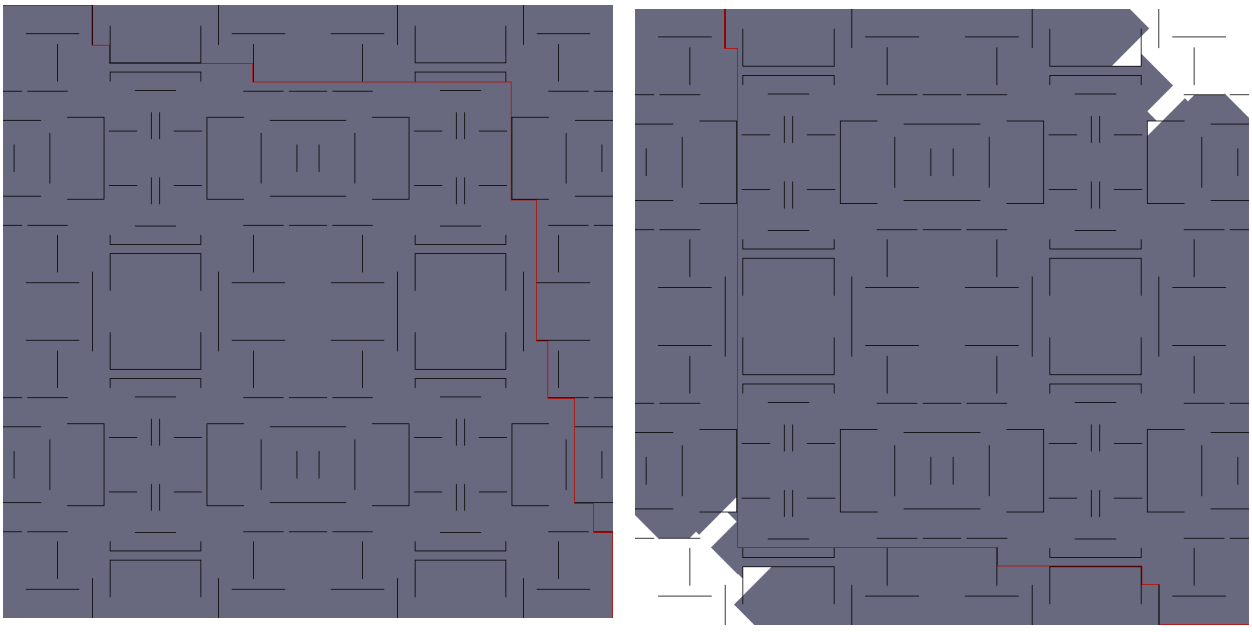


Helpoissa labyrinteissa A* on nopeampi kuin Dijkstra. Bidirectional ja BFS ovat hyvin lähellä toisiaan. A* ja Dijkstra jäävät (|V| + |E| log |V|) / 100000 kuvaajan alle molempia kekoja käyttäen. BFS ja Bidirectional jäävät (|V| + |E|) / 10000 kuvaajan alle.

Mistä nopeuserot johtuvat? Tarkastellaan algoritmien läpikäymiä solmuja.



Vasemmalla on A* ratkaisu ja läpikäydyt solmut. Oikealla on Dijkstran algoritmin ratkaisu ja läpikäydyt solmut. A* käy läpi vähemmän solmuja. Dijkstra joutuu käymään läpi koko verkon, koska maalisolmulla on suurin etäisyys alkusolmuun.

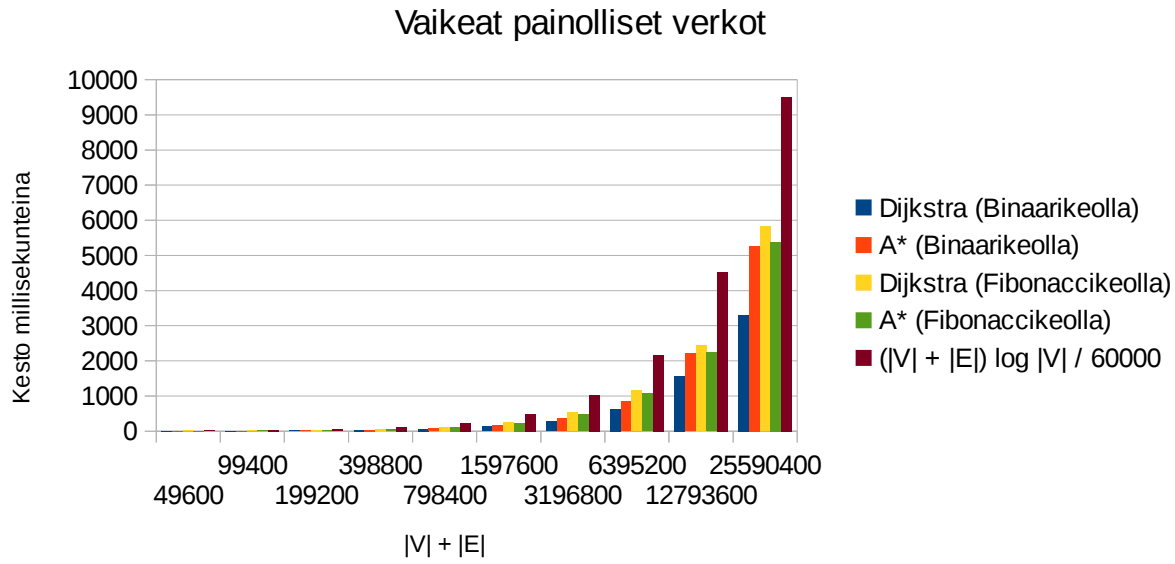


Vasemmalla on BFS-algoritmin löytämä ratkaisu ja läpikäydyt solmut. Oikealla on Bidirectional algoritmin löytämä ratkaisu ja läpikäydyt solmut. Bidirectional käy vain vähän vähemmän solmuja läpi. BFS, samoin kuin Dijkstra, joutuu käymään läpi koko verkon.

Painollisten verkkojen testaaminen

Painollisten verkkojen testaus on tehty värillisillä kuvilla, missä eri värit kuvaavat eripainoisia kaaria. Pienin kuva on 100x100 pikseliä. Suurin kuva on 3200x1600 pikseliä. Suuremmat kuvat on saatu monistamalla 200x200 pikselin kuvaa. Testit suoritetaan Dijkstran algoritmilla ja A*-algoritmilla, binaari- ja fibonaccikeolla. Testeistä saatiin seuraavat tulokset.

|V| + |E|	Dijkstra (Binaarikeolla)	A* (Binaarikeolla)	Dijkstra (Fibonaccikeolla)	A* (Fibonaccikeolla)
49600	5	5	9	6
99400	8	6	14	10
199200	13	13	34	22
398800	32	37	67	53
798400	61	71	121	108
1597600	133	163	266	229
3196800	279	356	528	476
6395200	624	839	1148	1074
12793600	1574	2216	2458	2235
25590400	3298	5268	5826	5387

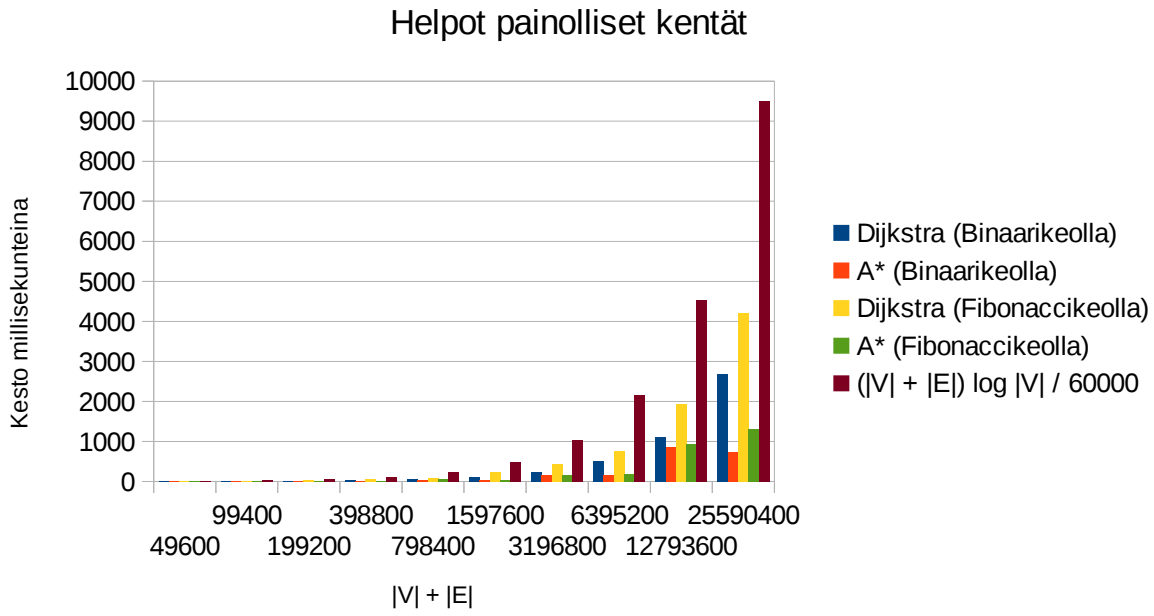


Dijkstra binaarikeolla on selkeästi nopein. A* binaarikeolla on hitaampi kuin Dijkstra binaarikeolla. Fibonaccikeolla A* suoriutuu miltein yhtä hyvin kuin binaarikeolla. Fibonaccikeolla A* on nopeampi kuin Dijkstra fibonaccikeolla. Molemmat algoritmit jäävät ((|V| + |E|) log |V| / 60000) -kuvaajan alapuolelle, joten tavoitellut aikavaativuudet toteutuvat.

Samoin kuin edellä painottomien verkkojen kanssa, testataan algoritmien toimintaa helpommilla painollisilla kentillä, joissa A* -algoritmin voi odottaa toimivan nopeammin.

Helppojen verkkojen testauksesta saatiin seuraavat tulokset.

|V| + |E|	Dijkstra (Binaarikeolla)	A* (Binaarikeolla)	Dijkstra (Fibonacci-keolla)	A*(Fibonaccikeolla)
49600	5	4	10	5
99400	9	6	15	8
199200	13	10	26	12
398800	29	13	52	19
798400	63	40	94	50
1597600	118	44	235	44
3196800	233	170	435	175
6395200	498	167	774	185
12793600	1118	852	1929	934
25590400	2680	734	4217	1309



A* on selvästi tehokkaampi kuin Dijkstra. Fibonaccikeon käyttäminen tekee taas molemmista algoritmeista hitaamman. On kiinnostavaa, että tietyissä tapauksissa ongelman kaksinkertaistuessa A*-algoritmin suoritusaika on pysynyt lähes samana tai ollut pienempi. Nämä ovat tilanteita, joissa kuvan leveys on kaksinkertaistettu. Hyötyykö manhattan-heuristiikka leveydestä? Kaikissa tapauksissa molemmat algoritmit saavuttavat (|V| + |E|) log |V| -aikavaativuuden, sillä ne jäävät (|V| + |E|) log |V| / 60000 -kuvaajan alapuolelle.

Tämä dokumentti käsittelee suorituskykytestausta. Algoritmien ja tietorakenteiden toimivuus on varmistettu yksikkötesteillä, jotka ovat toistettavissa milloin vain.