Tuomo Heino

Miten A*-algoritmia voidaan hyödyntää peleissä

Tietotekniikan kandidaatintutkielma

27. toukokuuta 2016

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Tuomo Heino

Yhteystiedot: tuomo.l.h.heino@student.jyu.fi

Ohjaaja: Tytti Saksa

Työn nimi: Miten A*-algoritmia voidaan hyödyntää peleissä

Title in English: How A*-algorithm is used in games

Työ: Kandidaatintutkielma

Sivumäärä: 20+0

Tiivistelmä: Tässä tutkielmassa tarkastellaan, miten A*-algoritmi ja siitä johdetut HPA*- ja KM-A*-algoritmit toimivat ja miten niitä voidaan hyödyntään pelikartoissa. A*-algoritmi on hyvin käytetty polunetsinnässä, mutta sen vaatimat resurssit tekevät siitä hitaan peleille. HPA*- ja KM-A* -algoritmit pyrkivät nopeuttamaan A*:n toimintaa tinkimällä reitin tarkkuudesta. Kummatkin nopeuttavat huomattavasti A*:n toimintaa, mutta tuovat myös ongelmia.

Avainsanat: A*-algoritmi, HPA*-algoritmi, KM-A*-algoritmi, polunetsintä

Abstract: In this study we look at A*-algorithm and its modifications HPA*- and KM-A*-algorithm and how they work on game maps. A*-algorithm is still very widely used in pathfinding but its resource heavy pathfinding makes it slow to use in games. HPA* and KM-A*-algorithm try to speed the pathfinding process by cutting from optimality of the path. Both of these algorithms offer a great speed increase for A* but they also present their own problems with the speed enhancing techniques.

Keywords: A*-algorithm, HPA*-algorithm, KM-A*-algorithm, pathfinding

Kuviot

Sisältö

1 Johdanto

Tässä tutkimuksessa tarkastellaan, miten A*-algoritmiin, eli polunetsintäalgoritmiin, pohjautuvat toteutukset toimivat peleissä. Tutkimus painottuu A*-algoritmiin ja sitä käyttävien algoritmien esittelemiseen. Erilaiset pelitilanteet vaativat toisenlaisia lähtökohtia, esimerkiksi missä tilanteissa toinen algoritmi on tehokkaampi tai nopeampi kuin toinen. Tutkimuksessa pohditaan myös kuinka luotettavia A*:stä luotujen parannuksien tuottamat polut ovat ja tarkastellaan ratkaisujen uhrauksia nopeuden saavuttamiseksi.

A*-algoritmi on tyypillisin valinta polunetsintään peleissä, vaikka erilaiset tilaesitykset vaihtelevat peleistä riippuen (Bjornsson ja Halldorsson 2006). Pelkän A*-algoritmin käyttö polunetsinnässä toimii hyvin, kun pelimaailma on pienehkö ja polunetsintä on vähäistä. Siirryttäessä isompiin ja monimutkaisempiin karttoihin A* alkaa syödä enemmän tehoa ja resursseja (Chen, Li ja Li 2011).

Kun A*:n suorituskyky alkoi muodostua pullonkaulaksi (Jansen ja Buro 2007), esiteltiin uusia tekniikoita nopeuttaa polunetsintäprosessia. Yksi näistä tekniikoista on hierarkinen lähestymistapa. Hierarkisessa tavassa kartta lähdetään ensin jakamaan jonkinlaisiin osioihin, yleensä huoneisiin tai lokeroihin. Nämä osiot muodostavat näin karkeamman kuvan todellisesta kartasta, eli luovat hierarkian varsinaisen kartan päälle. (Botea, Muller ja Schaeffer 2004) Suurin etu tällaisella ratkaisulla on, että A*:n ei tarvitse enää käydä koko karttaa läpi. Ongelmaksi taas muodostuu löydettyjen polkujen optimaalisuus (Bjornsson ja Halldorsson 2006).

Botea, Muller ja Schaeffer (2004) esittelevät HPA*-algoritmin. Se luo kartasta karkeampia tasoja, jolla tavoitellaan vähemmän vääriin reitteihin suuntautuvia kutsuja viralliselta kartalta, kun tiedetään alustava reitti (Botea, Muller ja Schaeffer 2004). Vaikka HPA* ratkaisikin alkuperäisen A*:n ongelman liian resursseja vievänä, sen tarkkuus ja polun luonnollisuus eivät olleet riittäviä osassa pelikartoista. Niinpä esitettiin uutta vaihtoehtoa HPA*:lle, joka huomioisi myös reitin laadun, sekä monimutkaisemmat kartat (Chen, Li ja Li 2011).

KM-A* hyödyntää sekä A*-algoritmiä että K-keskiarvoja. K-keskiarvo on alueelle laskettava arvo, jolla arvioidaan alueen esteiden määrää. Ratkaisu pienentää entisestään resurssien

käyttöä ja tuottaa optimaalisempia polkuja. Luodessaan karkeampia tasoja KM-A* huomioi myös kartan monimutkaisuuden, esimerkiksi esteiden määrän, ja luo tämän tiedon perusteella paremmin optimoituja alueita, joista on mahdollista saada nopeammin haluttu polku selville. Saavuttaakseen tämän KM-A* vie myös enemmän aikaa kuin HPA* luodessaan alustavaa tietoa kartasta (Chen, Li ja Li 2011).

Toisessa luvussa käsitellään tarkemmin A*-, HPA*- ja KM-A*-algoritmejä, sekä niiden toteutuksia. Esiin nostetaan myös Djikstran algoritmi(Dijkstra 1959), joka on A*:n perustana ja myös osana HPA*:n parannusehdotuksia. Kolmannessa luvussa vertaillaan algoritmeja ja niiden toimintaa keskenään. Neljännessä luvussa tuodaan esiin, mitä ongelmia A*:ä nopeammat HPA* ja KM-A* tuovat saavuttaakseen nopeuden ja resurssien minimoimisen.

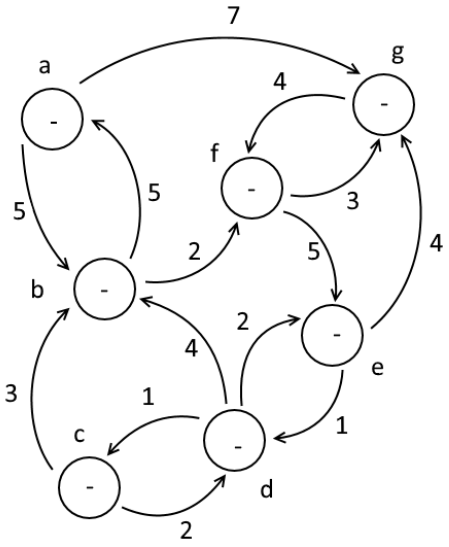
2 A* algoritmi ja sen johdannaiset

Tässä luvussa tarkastellaan yleisesti A*-algoritmiä ja sitä käyttäviä nopeampia ratkaisuja. Ensimmäisessä luvussa esitellään myös Djikstran algoritmi.

2.1 Djikstran algoritmi

Polunetsintä yksinkertaistuu lyhimmän reitin ongelmaksi. Tähän ensimmäisen nopeamman ratkaisun esitteli (Dijkstra 1959). Djikstran ratkaisussa ongelma jaetaan pisteiksi, jotka ovat yhteydessä toisiinsa poluilla, joiden pituus tiedetään. Oletuksena on, että ainakin yksi polku on olemassa kunkin pisteen välillä.

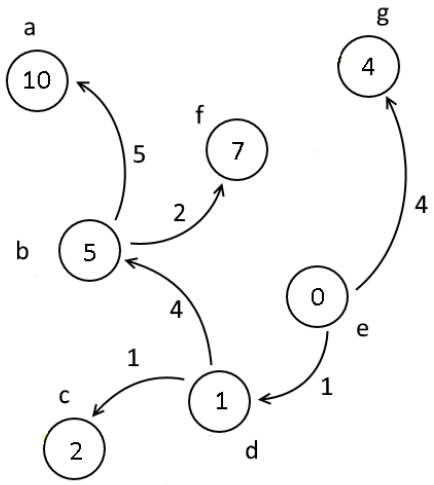
Polkua lähdetään etsimään tarkastelemalla pisteitä ja niiden välisten polkujen pituuksia. Valitaan aina jokin piste, jonne on jo polku. Tämän jälkeen vertaillaan pisteestä lähtevien polkujen pituuksia ja valitaan piste, jonne johtaa lyhin polku. Tätä toistetaan kunnes kaikki pisteet on käyty läpi ja on saatu selville pisteiden väliset lyhimmät etäisyydet (Dijkstra 1959).



Kuvio 1. Mielivaltainen kartta, jossa pisteet on yhdistetty poluilla. Kuva tehty paint.net-ohjelmalla.

Kuviossa 1 on kuvattuna eräs kartta ja sen pisteet ja polut. Polut on kuvattu yksisuuntaisilla

nuolilla, jotka esittävät mahdollisia kulkusuuntia. Ympyrät esittävät pisteitä ja ne on nimetty. Tarkastellaan polkuja, kun piste e valitaan aloituspisteeksi. Djikstran algoritmin mukaan etsitään ensin aloituspisteestä johtavat polut viereisiin pisteisiin. Kuviossa 1 vastaavat pisteet ovat d ja g. Toistetaan kunnes saavutetaan kuvion 2 mukainen polkukartta. Kuviosta 2 on karsittu ylimääräiset polut pois ja siinä esitetään vain lyhimpien reittien polut pisteelle e. Pisteiden sisälle on laskettu polkujen kulut.



Kuvio 2. Djikstran algoritmillä ratkaistu kartta. Aloituspisteeksi on valittu piste e. Kuva tehty paint.net-ohjelmalla.

2.2 A*-algoritmi

Vuonna 1968 Hart, Nilsson ja Raphael esittelivät parannuksen edeltäviin polunetsintäalgoritmeihin. Ratkaisu oli Djikstran algoritmia nopeampi ja tuotti täydellisiä polkuja, jos sellainen oli mahdollinen (Hu, Wan ja Yu 2012). A* ottaa myös huomioon reitin mahdollisen kustannuksen, eikä etsi pelkästään pienintä määrää pisteitä, jonka kautta se voisi kulkea (Hart, Nilsson ja Raphael 1968).

A*:n toimintaperiaate on yksinkertainen. Se olettaa kartan jakautavan osiin, yleensä ruudukkoon tai verkoksi. Kartalta valitaan aloitus- ja maalipiste. Tämän jälkeen algoritmi käy läpi alkupisteen ympärillään olevat pisteet ja laskee niille liikkumiseen kuluvan kustannuk-

sen. Samalla se vertailee kustannuksia pisteiden välillä ja valitsee pienimmän kustannuksen pisteen. Seuraavaksi se tarkastelee valitsemaansa pistettä samalla tavalla kuin alkupistettä. Algoritmi toistaa tätä kunnes se päätyy maalipisteeseen (Hart, Nilsson ja Raphael 1968).

A*-algoritmi on heuristinen algoritmi, eli se käyttää apunaan heuristista funktiota laskiessaan liikkumiseen kuluvia kustannuksia. Sen käyttämä vertailufunktio on muotoa: f(n) = g(n) + h(n), jossa n on siirryttävä piste. Funktio jakaantuu kahteen osaan: g(n) ja h(n). Funktio g(n) kuvaa pisteen n kokonaismatkaa lähtöpisteestä. Funktio h(n) arvioi pisteen liikkumiskustannusta pisteeseen n. (Hart, Nilsson ja Raphael 1968) Funktiona h(n) käytetään yleisesti Manhattanin etäisyyttä tai kahdeksansuuntaista etäisyyttä, arvioimaan matkaa maalipisteeseen (Bjornsson ja Halldorsson 2006).

A* soveltuu parhaiten karttaan, jossa etsittävien pisteiden määrä ei ole kovinkaan suuri tai kartta on hyvin yksinkertainen. Kartan koon kasvaessa isommaksi, lisääntyy A*:n käymien pisteiden määrä rajusti. Myös kartat, jotka ovat täynnä esteitä saavat A*:n käymään melkein koko kartan läpi ennen varsinaisen polun löytämistä (Bjornsson ja Halldorsson 2006).

2.3 HPA*-algoritmi

HPA*-algoritmin tarve muodostui, kun pelien karttojen koot alkoivat kasvaa ja perinteinen A*-algoritmin suorituskyky osoittautui isoksi pullonkaulaksi suorituskyvylle (Botea, Muller ja Schaeffer 2004). A*:n ominaisuus luoda täydellisiä polkuja vaatii isoilla kartoilla paljon laskentatehoa (Jansen ja Buro 2007). Ongelman ratkaisemiseksi HPA* muodostaa annetusta kartasta yksinkertaistetumpia tasoja, joissa kartta jaetaan yksinkertaisempiin osiin. Osille lasketaan tämän jälkeen optimireitti sen läpikululle ja tämä tallennetaan. (Botea, Muller ja Schaeffer 2004). Osiinjako voidaan hyödyntää myös useampaan kertaan jo muodostetuille osille, mikä mahdollistaa algoritmin skaalautumisen isommillekin kartoille (Jansen ja Buro 2007).

Kun osat on muodostettu, haluttu reitti pisteiden A ja B välillä saavutetaan sijoittamalla aluksi pisteet ylimmälle osalle. Tämän jälkeen hyödynnetään A*-algoritmiä, jolla valitaan alustava polku. Tätä toistetaan niin kauan kunnes saavutetaan alin taso eli varsinainen kartta. (Jansen ja Buro 2007).

HPA*:n tuottamille poluille voidaan suorittaa myöhemmin hiontaa, joka parantaa polkujen tarkkuutta. Helppo tapa toteuttaa tällainen on tarkistaa löytyykö kahden pisteen välille suoralinja, jos löytyy tämä korvaa aiemmin löytyneen ratkaisun. (Botea, Muller ja Schaeffer 2004). Vaikka kyseinen tapa on yksinkertainen, sen laskeminen on kuitenki kallista (Jansen ja Buro 2007). Jansen ja Buro (2007) ehdottavatkin polunhiomisen nopeuttamiseksi on luoda rajaava alue polun ympärille. Tämä ratkaisu nopeuttaa huomattavasti polunhiontaa ilman, että polun optimaalisuus kärsii liikaa (Jansen ja Buro 2007).

Siinä missä A*-algoritmi palauttaa aina kokonaisen polun (Hart, Nilsson ja Raphael 1968) HPA* palauttaa osajoukon mahdollisia polkuja (Botea, Muller ja Schaeffer 2004). Tästä osajoukosta voidaan siten kulkea tiettyä polkua jonkin matkaa ratkaisemalla todellinen polku sen alla. Mikäli polku ei kuitenkaan ole mahdollinen, HPA* ei tuhlaa turhaa laskenta-aikaa lopulle polulle. (Botea, Muller ja Schaeffer 2004)

2.4 KM-A*-algoritmi

KM-A*-algoritmi pyrkii edelleen parantamaan A*-algoritmiä. HPA*:stä eroten karttaa ensin tarkastellaan sen muotojen ja mallin mukaan, jonka avulla saavutetaan tarkempi ja luonnollisempi polku (Chen, Li ja Li 2011).

KM-A*:n toteutus koostuu useammasta osasta. Aluksi karttaa lähdetään jakamaan osa-alueisiin. Osa-alueita kasvatetaan lisäämällä niihin alueita, jotka ovat samankaltaisia toistensa kanssa. Alueiden samankaltaisuutta arvioidaan käyttämällä euklidista etäisyyttä alueen esteiden välillä. Osa-alueiden kasvattamista jatketaan, kunnes kaikki osat ovat yhdessä alueessa tai tavataan jokin reunaehto. Reunaehtona voidaan käyttää alueen K-keskiarvoa, joka lasketaan alueen esteiden määrästä (Chen, Li ja Li 2011).

Myös osa-alueita luokitellaan keskenään näiden välisten DB-indeksien avulla. DB-indeksillä pyritään arvioimaan, mikä olisi optimaalisin osa-alueiden määrä. KM-A* olettaa osa-alueen ollessa pieni, sen sisältämien esteiden määrä on mahdollisimman suuri. Suurien osa-alueiden kohdalla oletus on päinvastainen, eli alueella olevat esteet ovat harvassa. K-keskiarvoa hyväksi käyttäen KM-A* pyrkii löytämään parhaimman DB-indeksin (Li ym. 2013).

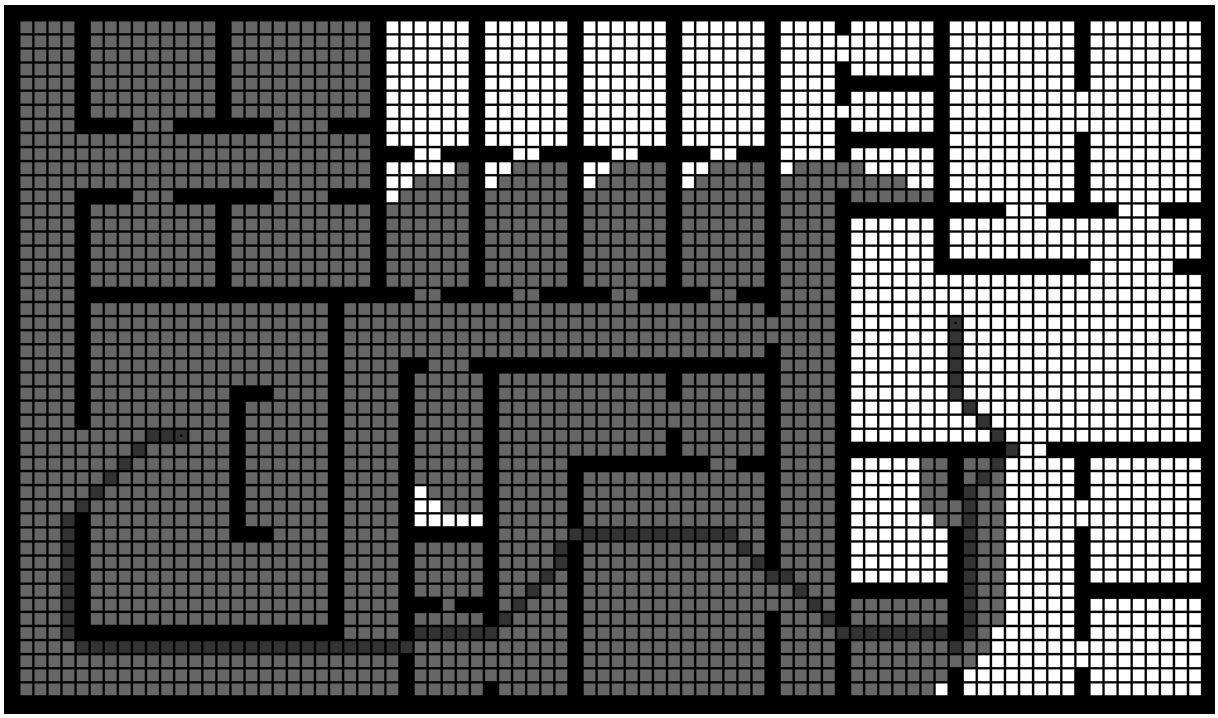
Toisena osana KM-A* käyttää A*-algoritmiä etsiäkseen polun koostetuista osa-alueista (Chen, Li ja Li 2011). A*:ä käytetään ensin luomaan karkea polku osa-alueiden välillä. Karkeasta polusta aloitetaan sen jälkeen työstämään haluttua oikeaa polkua. Mikäli osa-alueessa on esteitä, käytetään A*:ä polunetsintään. Tyhjille osa-alueille käytetään Bresenhamin suoranlinjan algoritmia polunetsinnän nopeuttamiseksi (Li ym. 2013).

3 Algoritmien erot ja käyttötavat

Tässä luvussa vertailen algoritmejä keskenään ja tilanteita, joissa olisi perustellumpaa käyttää tiettyä algoritmia.

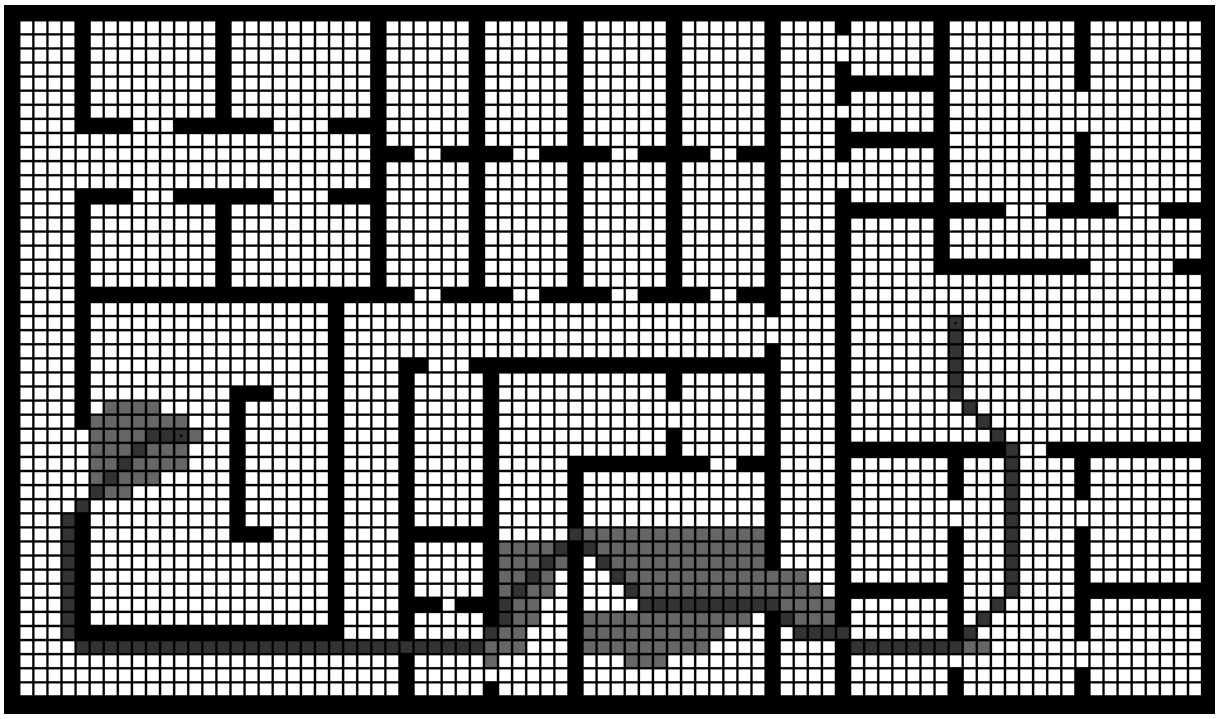
3.1 A*:n ja hierarkisten algoritmien erot

A* tuottaa aina valmiin polun (Hart, Nilsson ja Raphael 1968). Tämä ominaisuus tekee siitä erittäin resursseja vievän, kun kartan kokoa tai monimutkaisuutta kasvatetaan (Cui ja Shi 2011). Peleissä riittää usein tieto pienestä osasta polkua, joka johtaa kohti haluttua pistettä, sillä usein liikkuva kohde kohtaa jonkin esteen matkalla. A*:ä käytettäessä, polku lasketaan aina kokonaisuudessaan. Mikäli polkua ei ole ehditty kulkea kuin pieni osa ennen kuin törmätään esteeseen, tuhlaantuu resursseja ylimääräiseen työhön. Kuviosta 3 nähdään kuinka paljon A* tekee turhaa työtä isommassa ja monimutkaisemmassa ympäristössä.



Kuvio 3. A*-algoritmillä etsitty polku (Bjornsson ja Halldorsson 2006). Tummat neliöt kertovat käydyistä kartan pisteistä.

HPA* välttää tämän ratkaisemalla vain tarpeellisen osan tarkasti hierarkiastaan (Botea, Muller ja Schaeffer 2004). A*:een verrattaessa HPA* ei tuota yhtä tarkkoja tuloksia, sillä se tinkii reitin tarkkuudesta vapauttaakseen resursseja (Botea, Muller ja Schaeffer 2004). HPA* ja etenkin KM-A* luovat ennakkoon tietoa annetusta kartasta. Tämän tiedon luominen vie jonkin verran aikaa sekä vaatii jonkinlaista säilytystilaa (Botea, Muller ja Schaeffer 2004, Chen, Li ja Li 2011). A*:ä pelkästään käytettäessä, ei tarvita ennakkoon laskettuja arvoja, vaan polunetsintä tapahtuu reaaliajassa eikä aseta esteitä esimerkiksi kartan luomiselle. Kuvio 4 kuitenkin osoittaa, että ennakkoon laskettu tieto kartasta luo huomattavasti vähemmän työtä varsinaisen polun etsimiseen. Kuviosta 4 huomataan myös se, miten löydetty polku poikkeaa A*:llä saadusta polusta.



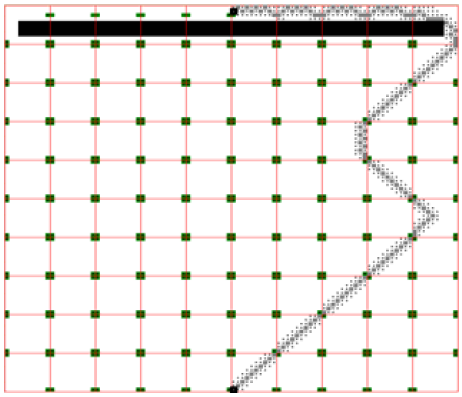
Kuvio 4. Hierarkisella A*-algoritmillä etsitty polku (Bjornsson ja Halldorsson 2006). Tummat neliöt kertovat käydyistä kartan pisteistä.

3.2 Käyttö muuttumattomassa kartassa

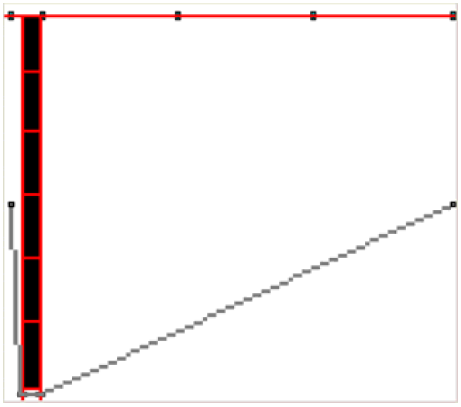
Useissa peleissä käytössä on valmiiksi luotu kartta, joka ei muutu paljoakaan pelatessa. KM-A* on erittäin hyvin toimivt tällaisissa tilanteissa, sillä sen tarvitsema tieto voidaan laskea

tarkkaan ennakkoon ja näin vähentää polkujen epätarkkuuksia (Chen, Li ja Li 2011). KM-A*:n esteiden huomiointi tuottaa myös huomattavasti parempia tuloksia polkuun, varsinkin jos kyseessä on paljon esteitä sisältävä ympäristö (Li ym. 2013).

Myös HPA* hallitsee muuttumattoman ympäristön, mutta KM-A*:n käyttö sen sijaan on perusteltua sen luomien epätarkkuuksien takia. Kuviosta 5 nähdään, miten HPA* tuottaa hieman oudon polun täysin vapaaseen alueeseen. Koska HPA* luo samankokoisia alueita kartalle, ilman mitään erottelua alueiden välillä, sen luomat polut poikkeavat niistä, joita pelaaja käyttäisi (Chen, Li ja Li 2011). Kuvio 6 näyttää, miten KM-A* hallitsee saman tilanteen.



Kuvio 5. HPA*-algoritmin heikkous, tasaisesti jaetetut ruudut eivät aina anna parasta tulosta. (Li ym. 2013).



Kuvio 6. KM-A*-algoritmin käyttämä osittamistapa luo HPA*:ä paremman tuloksen (Li ym. 2013).

KM-A*:n käyttämä aluejako huomioi esteet, jolloin se osaa luoda alueet sopivan kokoisiksi niiden ominaisuuksien perusteella (Chen, Li ja Li 2011). Nyt helposti läpi käytävät alueet

ovat suuria ja paljon esteitä sisältävät pieniä, tämä jako auttaa A*:llä tehtävän lopullisen polun etsintää (Li ym. 2013). A* ei suoriudu hyvin liian suurella alueella olevasta reitistä varsinkaan, kun alueella olevien esteiden määrä kasvaa (Botea, Muller ja Schaeffer 2004).

Mikäli karttana on jonkinlainen labyrintti, HPA* on toimiva ratkaisu (Li ym. 2013). Myöskin kartat, joissa esteiden määrä on pienehkö ja tasainen se suoriutuu paremmin ja luonnollisemmin.

3.3 Käyttö dynaamisesti muuttuvissa kartoissa

Hierarkiset algoritmit toimivat vaihtelevasti dynaamisissa ympäristöissä. Mitä enemmän dynaamisesti luotuja kappaleita tai esineitä ympäristössä esiintyy sitä epätarkemmaksi hierarkiset algoritmit muuttuvat (Bjornsson ja Halldorsson 2006). HPA*:lle on esitetty parannuksena, dynaamisesti muuttuvien ympäristöjen ratkaisuksi, jättää laskematta alimmat reitit niin tarkasti kuin mahdollista. Kun polkua halutaan etsiä, HPA* laskee samalla tarkemman polun halutulle reitille. Tämä ratkaisu tuottaa parempia reittejä vain pienellä tehon lisäyksellä (Jansen ja Buro 2007).

Toisin kuin HPA*-algoritmi, KM-A* ei pärjää dynaamisesti muuttuvassa ympäristössä. Ympäristön muuttuessa KM-A* suorittaa uudestaan raskaita laskentoja, jotka hidastavat algoritmiä ja polunetsintää (Li ym. 2013).

4 HPA*- ja KM-A*-algoritmien ongelmat

Tässä luvussa tarkastellaan millaisia ongelmia HPA*- ja KM-A*-algoritmien käyttö voi tuottaa.

4.1 Löydetyn polun optimaalisuus

Nopeuttaakseen polunetsintää HPA* ja KM-A* luopuvat osasta polun tarkkuudesta saavuttaakseen tämän (Chen, Li ja Li 2011). HPA*:ä käytettäessä, luodut hierarkiset osat ovat jaettu suoraan samankokoisiin osiin (Botea, Muller ja Schaeffer 2004). Osien karkea jaottelu parantaa A*:n suorituskykyä (Jansen ja Buro 2007), mutta voi tuoda esiin ongelmia polun tarkkuudessa tietyissä pelimaailmoissa. Varsinkin monimutkaisemmat kartat haittaavat HPA* polun optimaalisuutta (Chen, Li ja Li 2011). Botea, Muller ja Schaeffer (2004) mukaan HPA* pitäisi polkujen tarkkuuden yhden prosentin heitolla optimaalisesta.

KM-A* yrittää poistaa HPA*:n tuoman ongelman monimutkaisissa kartoissa, ottamalla huomioon millainen kartan muoto ja ominaisuudet ovat (Chen, Li ja Li 2011). Tämä monimutkistaa HPA*:n yksinkertaisen ratkaisun helposti jaoteltavasta hierarkiasta ja esittelee tarkemman jaottelun osiin käyttämällä hienostuneempaa algoritmiä. Mikäli algoritmien toteutus ei ole täysin optimaalinen, voi KM-A* pohjainen polunetsintä tuottaa epätarkkoja polkuja. Myös polkujen tarkkuus vaihtelee, jolloin ei välttämättä saada aina samaa tulosta samoilla alku- ja loppupisteillä (Chen, Li ja Li 2011).

4.2 HPA*:n ongelmat

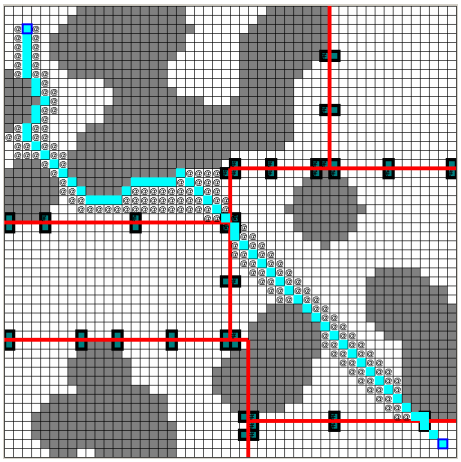
HPA* toimii hyvin kartan koon ollessa suuri tai monimutkainen verrattaessa A*-algoritmiin. A* kuluttaa vähemmän resursseja kuin HPA*, kun kartan koko pienenee tai haluttu reitti on suoraviivainen. HPA* vaatii tällöin A*:ä enemmän resursseja sen käyttämän hierarkiamallin takia (Botea, Muller ja Schaeffer 2004). Onkin siis hyvä tarkastella ongelman kokoa ennen varsinaista polunetsinnän tarvetta. Varsinkin lyhyillä yksinkertaisilla matkoilla perinteinen A* suoriutuu huomattavasti nopeammin kuin HPA* (Botea, Muller ja Schaeffer 2004).

Vaikka HPA*-algoritmille on esitelty useita parannuksia ja sitä nopeuttavia ratkaisuja, osassa pelikartoilla sille luontainen ruudukkojako vaikuttaa sen suorituskykyyn sekä reitin luonnollisuuteen (Chen, Li ja Li 2011). Myös A*-algoritmin käyttö etsintään voi tuottaa hitautta. Jansen ja Buro (2007) esittävätkin parannusehdotukseksi Djikstran algoritmia, jonka hitain mahdollinen ratkaisu on nopeampi kuin A*:n vastaava.

4.3 KM-A*:n ongelmat

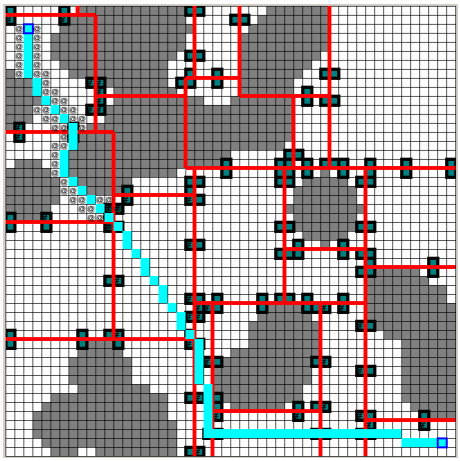
KM-A* vaatii valmiiksi laskettua tietoa sen alueista, jotka voivat olla epätarkkoja tulevissa tilanteissa. Tällöin kartan optimaalisuus voidaan herkästi menettää (Chen, Li ja Li 2011). Hierarkisena toteutuksena KM-A*:n toteutukselle pätevät samankaltaiset ongelmat mitä HPA*:ä käytettäessä. Löydetty polku ei aina ole täydellisen optimaalinen verrattaessa A*:llä löydettävään polkuun (Chen, Li ja Li 2011).

KM-A* poikkeaa HPA*:stä siinä, että se laskee paljon enemmän ja tarkemmin hierarkiset osansa (Chen, Li ja Li 2011). Vaikka tämä tuo nopeutta varsinaiseen polunetsintää, se ei tällöin sovellu nopeasti muuttuvien karttojen kanssa käytettäväksi (Li ym. 2013).



Kuvio 7. KMA*-algoritmi perinteisellä DB-indeksöinnillä (Chen, Li ja Li 2011).

Yhtenä ongelmana KM-A*:ssä on myös sen käyttämä DB-kriteeri ja K-keskiarvojen valinta. Mikäli algoritmin kyseiset osat algoritmista ovat toteutettu huonosti, KM-A*:n luomien alueiden optimaalisuus kärsii ja sen suorituskyky näin ollen hidastuu. (Li ym. 2013) Kuviois-



Kuvio 8. KMA*-algoritmi parannetulla DB-indeksöinnillä (Chen, Li ja Li 2011).

ta 7 ja 8 nähdään, kuinka paljon osa-alueiden koot vaihtelevät DB-indeksöinnin valinnasta riippuen. Kuviossa 7 osa-alueita on huomattavasti vähemmän kuin kuviossa 8. Perinteisellä indeksöinnillä löytynyt polku tekee oudon hyppäyksen verrattaessa kuvion 8 parannetun indeksöinnin polkuun.

4.4 Ongelmien haastavuudesta

Isoimpana erona A*:n ja sen parannusten välillä on löydetyn polun optimaalisuus. Sekä HPA* että KM-A* tinkivät polun tarkkuudesta, saavuttaakseen nopeammin halutun tuloksen (Bjornsson ja Halldorsson (2006), Chen, Li ja Li (2011)). Peleissä polun optimaalisuus on yleensä toissijaista, varsinkin jos kyseessä on reaaliaikainen strategiapeli. Pääpaino parannetuilla algoritmeillä onkin säästää mahdollisimman paljon laskentatehoa ja muistinkäyttöä pelin muita osia varten (Chen, Li ja Li 2011).

KM-A*:n haasteena on DB-indeksöinnin ja K-keskiarvojen löytäminen juuri oikeaksi, jokaiselle kartalle. Jos käytössä on heikompi ratkaisu, polunetsinnän tarkkuus ja etsintänopeus, tippuvat huomattavasti (Li ym. 2013). Kuviot 7 ja 8 osoittavat indeksöinnin tärkeyden.

HPA*:n ongelmille on esitelty jo useita ratkaisuja, jotka pyrkivät parantamaan sitä alueilla joissa sen suorituskyky on heikko. Djikstran algoritmin käyttö A*:n tilalla nopeuttaisi hankalien polkujen etsintää. Myös lopullisten polkujen laskematta jättäminen parantaisi HPA*:n

toimivuutta dynaamisesti muuttuvissa kartoissa (Jansen ja Buro 2007).

5 Yhteenveto

Vaikka A*-algoritmi on vanha keksintö, ei sen tilalle peliteollisuuteen ole saatu vielä korvaajaa. Pelialueiden kasvaessa peliteollisuus on vain muokannut erilaisia ratkaisuja A*:n päälle saadakseen paremman suorituskyvyn (Li ym. 2013). Pelien kehittyessä yhä monimutkaisemmiksi ja isommiksi, uusien ja tehokkaiden ratkaisujenkin tarve alkaa kasvaa (Bjornsson ja Halldorsson 2006). Haasteellista tulee olemaan se, miten monimutkaiset kartat saadaan yksinkertaistettua tarpeeksi tarkasti ilman, että polkujen optimaalisuus kärsisi liikaa (Jansen ja Buro 2007).

HPA* ja KM-A* antavat jo huomattavasti parempia tuloksia polunetsinnän nopeudessa ja resurssien säästämisessä. HPA*:n käyttö on myös mahdollista muuttuvissakin ympäristöissä (Bjornsson ja Halldorsson 2006), mikä antaa sille omat etunsa KM-A*:ä vastaan. KM-A*:n valttina taas on sen sopeutuminen ympäristön yksityiskohtiin (Li ym. 2013), jota HPA* ei tee. Tällä saavutetaan tietyissä tilanteissa luonnollisempi polku (Chen, Li ja Li 2011).

Polunetsintäalgoritmia valitessa tulee ottaa huomioon, millaiseen ympäristöön sitä ollaan käyttämässä. Muuttumattomien karttojen kanssa on parempaa käyttää algoritmiä, joka vaatii enemmän ennalta laskettua tietoa ollakseen nopeampi. Dynaamisesti muuttuvien karttojen kanssa taas on parempi turvautua algoritmiin, joka pystyy nopeasti mukautumaan tilanteeseen, ilman että kaikkia sen tarvitsemaa tietoa ei lasketa uudestaan. Huomioon kannattaa ottaa myös kuinka luonnollisia tai tarkkoja polkuja haluaa.

Tulevaisuuden haasteena ovat yhä laajemmiksi ja monimuotoisimmiksi muuttuvat kartat ja niihin sopeutuvat algoritmit (Chen, Li ja Li 2011). Haastetta tuottavat myös yhä dynaamisemmiksi muuttuvat ympäristöt, kun pelaajat haluavat luonnollisemmalta tuntuvia ympäristöjä. Nykyisetkin ratkaisut alkavat olla liian hitaita, kun polunetsintää vaativien kohteiden määrä kasvaa samalla kuin kartan koko ja monimutkaisuus.

Lähteet

Bjornsson, Yngvi, ja Kari Halldorsson. 2006. “Improved Heuristics for Optimal Path-finding on Game Maps.” *AIIDE* 6:9–14.

Botea, Adi, Martin Muller ja Jonathan Schaeffer. 2004. “Near optimal hierarchical path-finding”. *Journal of game development* 1 (1): 7–28.

Chen, Cai, Yan Li ja Tie-Song Li. 2011. “KM-A* pathfinding algorithm based on hierarchical clustering and strengthened DB Index criteria”. Teoksessa *Machine Learning and Cybernetics (ICMLC), 2011 International Conference on,* 4:1571–1576.

Cui, Xiao, ja Hao Shi. 2011. “A*-based pathfinding in modern computer games”. *International Journal of Computer Science and Network Security* 11 (1): 125–130.

Dijkstra, E. W. 1959. “A Note on Two Problems in Connexion with Graphs”. *Numer.Math.* 1, numero 1 (joulukuu): 269–271.

Hart, P. E., N. J. Nilsson ja B. Raphael. 1968. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. *IEEE Transactions on Systems Science and Cybernetics* 4 (2): 100–107.

Hu, Jie, Wang gen Wan ja Xiaoqing Yu. 2012. “A pathfinding algorithm in real-time strategy game based on Unity3D”. Teoksessa *Audio, Language and Image Processing (ICALIP), 2012 International Conference on,* 1159–1162.

Jansen, M. Renee, ja Michael Buro. 2007. “HPA* Enhancements.” *AIIDE* 7:84–87.

Li, Yan, Wenju Zhao, Zhenhua Zhou ja Cai Chen. 2013. “Hierarchical and Dynamic Path-finding Algorithms in Game Maps”. *International Journal of Advancements in Computing Technology* 5, numero 11 (heinäkuu): 87–98.