hyväksymispäivä arvosana

arvostelija

Polunetsintä usean agentin ympäristössä ja peleissä

Pasi Laaksonen

Helsinki 4.5.2007

LuK-tutkielma

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Pasi Laaksonen

Työn nimi — Arbetets titel — Title
Polunetsintä usean agentin ympäristössä ja peleissä

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
LuK-tutkielma	4.5.2007	24 sivua

Tiivistelmä — Referat — Abstract

Tutkielmassa selvitetään ja koostetaan yhteen millaisia erilaisia ratkaisuja voidaan polunetsinnässä käyttää, kun liikuteltavana on useita agentteja samassa ympäristössä. Tutkielmassa tarkastellaan, millä eri tavoin muut agentit voidaan ottaa huomioon ja miten muiden agenttien valitsemat reitit voivat muuttaa polunetsinnän tuloksia. Tutkielmassa esitellään myös muutamia algoritmeja, jotka ovat nimenomaan usean agentin ympäristöön tarkoitettuja ja joissa agentit jakavat tiedon reittivalinnoistaan muille agenteille. Tietokonepelit ovat yksi alue, jossa polunetsintää tarvitaan, ja tässä tutkielmassa onkin useassa kohdin käytetty pelejä ja pelitilanteita esimerkkeinä.

ACM Computing Classification System (CCS):
I.2.8 [Problem Solving, Control Methods and Search],
K.8 [Personal Computing]

Avainsanat — Nyckelord — Keywords
Polunetsintä, usean agentin ympäristö, tietokonepelit, CA*

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Polunetsinnän tarkoitus on löytää agentille hyvä reitti lähtöpaikasta maalipaikkaan. Yhden agentin tapauksessa tämä toimenpide on yleensä varsin suoraviivainen ja yhden agentin polunetsintään on algoritmeja suuri määrä. Useampi agentti samassa ympäristössä tuo kuitenkin polunetsintään muutoksia, sillä harvoin agentit voivat kulkea toistensa läpi. Agentit saattavat törmäillä toisiinsa tai tukkia toistensa reittejä. Törmäilystä voidaan vielä selvitä yksinkertaisesti laskemalla törmänneille agenteille uudet reitit, mutta tällainen ratkaisu on usein kömpelö. Reittien tukkeutuminen sen sijaan vaatii jo muitakin toimenpiteitä. Polunetsintä usealle agentille onkin aktiivisen tutkimuksen kohteena [GCB06].

Tietokonepelit ovat yksi alue, jossa on tarvetta usean agentin yhtäaikaiselle polunetsinnälle. Yleisiä pelityyppejä, joissa polunetsintää tarvitaan, ovat reaaliaikaiset strategiapelit (RTS, real-time strategy) ja ensimmäisen persoonan ammuntapelit (FPS, first-person shooter). Näissä molemmissa liikuteltavia hahmoja on käytännössä aina useampi kuin yksi.

Karkeasti jaoteltuna polunetsintää usean agentin ympäristössä voidaan suorittaa joko ilman yhteistyötä tai yhteistyön kanssa. Ilman yhteistyötä agentit eivät ota toisiaan tai toistensa reittejä huomioon muuten, kuin ajatellen niiden olevan ympäristöön kuuluvia esteitä. Jos taas agentit tekevät yhteistyötä, ne jakavat tiedon omista reittivalinnoistaan muille agenteille, jotta nämä osaavat tarvittaessa väistää toisiaan.

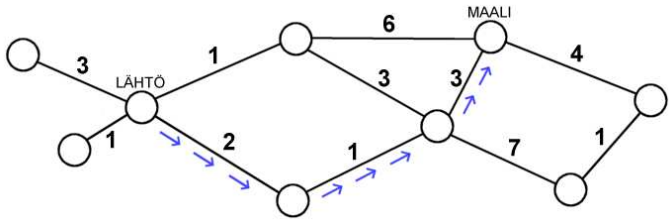
Tämän tutkielman alussa on esitelty lyhyesti polunetsinnän toiminta. Tämän jälkeen on selvitetty, millä eri keinoin polunetsintää voidaan tehdä ilman agenttien välistä yhteistyötä. Polunetsintään yhteistyöllä on puolestaan kehitetty erillisiä algoritmeja, joista esitellään muutamia keskeisiä. Tutkielman lopussa on lisäksi selvitetty, miten polunetsintää voidaan vähentää nimenomaan usean agentin ympäristössä.

2 Polunetsintä

Ennen polunetsinnän suorittamista tarvitsee ympäristöstä luoda sitä kuvaava verkko (graafi). Jokainen verkon solmukohta kuvaa sijaintia, jossa agentin on mahdollista olla. Solmuja yhdistävät kaaret kuvaavat vapaita reittejä solmujen välillä. Verkko on usein painotettu niin, että kulkeminen solmusta toiseen ei ole aina kustannukseltaan samanarvoista. Peleissä pelihahmot tai muut liikkuvat yksiköt ovat agenttien

ilmentymiä. Erilainen maasto eri osissa pelialuetta voi puolestaan vaikuttaa verkon kustannuksiin.

Polunetsintäalgoritmit etsivät verkosta nimenomaan reittiä, joka olisi kustannukseltaan vähäisin. Tällöin verkon kustannuksia muuttamalla voidaan saada polunetsintä toimimaan eri tavoilla. Mikäli kustannus kuvaa esimerkiksi solmujen välistä etäisyyttä, polunetsintä pyrkii löytämään lyhimmän reitin. Kuvassa 1 on esimerkkinä yksinkertainen verkko ja verkon kaarissa olevat numerot kuvaavat kustannusta solmujen välillä. Nuolet osoittavat parasta reittiä lähtösolmusta maalisolmuun.



Kuva 1: Polunetsintäverkko ja kustannukseltaan paras reitti.

Yksinkertainen verkko voi olla esimerkiksi kaksiulotteinen ruudukko, missä kustakin ruudusta pääsee liikkumaan viereisiin ruutuihin. Tällöin ruudut ovat verkon solmukohtia. Monissa tämän tutkielman kuvissa verkko onkin esitetty selkeyden vuoksi ruudukkona. Peleissä myös käytetään edelleen usein ruudukkoon pohjautuvaa polunetsintää, vaikka pelien grafiikka ei enää ruudukkopohjaista olisikaan [Mil06]. Kaikkia esiteltyjä polunetsinnän keinoja voi kuitenkin käyttää minkä muotoisessa verkossa tahansa.

2.1 A*-algoritmi

Tässä luvussa on esitelty A*-algoritmin (lausutaan "a-tähti", engl. "a-star") toiminta [HNR68]. Erittäin monet polunetsintäalgoritmit, kuten myös usean agentin polunetsintään erikoistuneet algoritmit, käyttävät A*:ä pohjanaan. A* on myös yleisin peleissä käytetty algoritmi [Mil06, GCB06].

A* käyttää hyväkseen heuristisia arvioita siitä, kuinka suuri kustannus tietystä solmusta saattaisi vielä olla maalisolmuun. A* jatkaa reitin etsimistä aina siitä solmusta, missä siihen mennessä kuljetun matkan kustannus ja arvio loppumatkan kustannuksesta yhteenlaskettuna on pienin. Tällöin A* selviää yleensä huomattavasti

pienemmällä tutkittavien solmujen määrällä, kuin jos lähdettäisiin vain sokeasti etsimään maalisolmua laajentamalla hakua jatkuvasti kaikkiin viereisiin solmuihin. Nämä heuristiset arviot voidaan laskea millä tahansa sopivaksi katsotulla tavalla. Ruudukossa heuristisina arvioina käytetään usein Manhattan-etäisyyttä [Sil06]. Se on yksinkertaisesti kahden ruudun välisen vaaka- ja pystyetäisyyden summa.

A* laskee polunetsinnän aikana solmuille seuraavia arvoja: g on koko tähän mennessä kuljetun matkan kustannus, h on heuristinen arvio koko loppumatkan kustannuksesta ja f on näiden summa. Polunetsinnän aikana solmuja merkitään avoimiksi tai suljetuiksi, mutta solmu ei voi olla yhtä aikaa molempia. A* toimii seuraavasti:

1. Muutetaan lähtösolmu avoimeksi ja lasketaan sille h ja f. Kuljettu matka g on luonnollisesti 0.

2. Valitaan sellainen avoin solmu, jonka f-arvo on pienin. Mikäli useilla avoimilla solmuilla on sama pienin f-arvo, valitaan niistä satunnainen, mutta suositaan valinnassa solmua, joka on maalisolmu.

3. Muutetaan valittu solmu suljetuksi. Jos valittu solmu on maalisolmu, lopetetaan algoritmin suoritus, koska reitti on löytynyt.

4. Etsitään kaikki solmut joihin valitusta solmusta pääsee, ja tehdään jokaiselle näistä solmuista seuraavat kohdat:

   - Lasketaan solmulle g, h ja f.
   - Mikäli solmu on avoin tai suljettu, tarkistetaan, onko solmuun aiemmin laskettu g pienempi tai yhtä suuri kuin uusi g. Jos näin on, vanhoihin g-, h- ja f-arvoihin ei kosketa, eikä tälle solmulle enää tehdä seuraavaa kohtaa.
   - Merkitään mistä suunnasta tähän solmuun tultiin ja muutetaan solmu avoimeksi.

5. Mikäli avoimia solmuja ei enää ole, lopetetaan algoritmin suoritus, koska reittiä ei löytynyt. Muutoin siirrytään kohtaan 2.

Mikäli maalisolmu löytyi, reitti maalista lähtöön saadaan seuraamalla jokaiseen solmuun jätettyjä merkkejä siitä, mistä kyseiseen solmuun tultiin. Reitti lähdöstä maaliin on luonnollisesti tämä käänteisenä.

A*-algoritmin tehokkuus riippuu pitkälti h-arvoista [Sto96]. Mitä paremmin ne kertovat todellisen kustannuksen maalisolmuun, sitä nopeammin A* selviytyy polunetsinnästä. Tarkempien arvioiden laskeminen on kuitenkin taas usein työläämpää. Tietynlaiset h-arvot voivat myös esimerkiksi taata, että A* löytää varmasti optimaalisen reitin, jos sellainen on olemassa [HNR68].

2.2 Usean agentin ympäristö

Yhden agentin tapauksessa polunetsinnän tarvitsee vain kiertää ympäristöön kuuluvat esteet. Useampi agentti samassa ympäristössä tuo mukanaan joitakin uusia ongelmia. Agentit voivat törmätä toisiinsa tai agentit voivat estää toistensa liikkumisen tukkimalla reittejä polunetsintäverkossa. Törmäyksellä viitataan usein vain tilanteeseen, jossa agentit olisivat juuri törmäämässä toisiinsa ja törmäyksen välttämiseksi joudutaan tekemään toimenpiteitä. Mikäli agentit eivät törmäystä pysty välttämään millään tavalla, ne yleensä vain pysähtyvät. Agenttien törmäily toisiinsa on yleensä vain liikkumista hidastava seikka. Sen sijaan reittien tukkeutuminen on hankalampi ongelma, joka voi saada koko polunetsinnän epäonnistumaan.

Pelien käytettävissä oleva prosessoriaika ja muistin määrä saattavat olla hyvinkin rajattuja erityisesti pelikonsoleissa [Tom04]. Peleissä koneella on myös niin paljon muutakin tehtävää, että polunetsinnän käyttöön jää vain murto-osa koneen resursseista. Silti polunetsinnän päätökset pitäisi usein syntyä nopeasti. Ainoastaan vuoropohjaisissa peleissä saattaa olla aikaa laskea reitit kaikille pelihahmoille uudestaan joka vuorolla.

Polunetsintä on onneksi mahdollista jakaa pienempiin osiin. Polunetsintää useille agenteille voidaan suorittaa jonossa yksi kerrallaan ja yksittäisen agentin polunetsintäkin voidaan keskeyttää ja jatkaa sitä myöhemmin [Mil06]. Tämä jakaa hieman polunetsinnän aiheuttamaa kuormaa. Tästä huolimatta polunetsintää voidaan harvoin suorittaa optimaalisesti, etenkään usean agentin tapauksissa. Siksi polunetsinnässä joudutaan usein tyytymään erilaisiin ratkaisuihin, jotka eivät ole aivan täydellisiä.

3 Polunetsintä ilman yhteistyötä

Polunetsinnällä ilman yhteistyötä tarkoitetaan sitä, että agentit eivät jaa tietoa omista liikkeistään muille. Tällöin polunetsinnässä käytetään tavallisia yhden agen-

tin polunetsintäalgoritmeja, kuten juuri A*:ä. Ongelmaksi jää lähinnä se, miten muihin agentteihin tai niiden varaamiin reitteihin suhtaudutaan. Muita agentteja voidaan yrittää väistellä eri keinoin, mutta ilman tarkkaa yhteistyötä törmäyksiä syntyy toisinaan. Törmäysten havaitsemisen jälkeen yritetään yleensä vain löytää jokin uusi reitti kelvottomaksi menneen vanhan reitin tilalle.

Mikäli eri agenteilla on erilaisia prioriteetteja, törmäyksissä tämä voidaan ottaa huomioon laittamalla matalamman prioriteetin hahmo etsimään uutta reittiä [Tom04]. Korkeamman prioriteetin agentti voi jatkaa alkuperäisellä reitillä ja näin se yleensä pääsee maaliinsa suorempaa reittiä kuin matalan prioriteetin agentit. Peleissä korkeamman prioriteetin agentteja voisivat olla esimerkiksi sankarihahmot, haavoittuneet yksiköt, tai hahmot, joille pelaaja on viimeksi antanut komentoja.

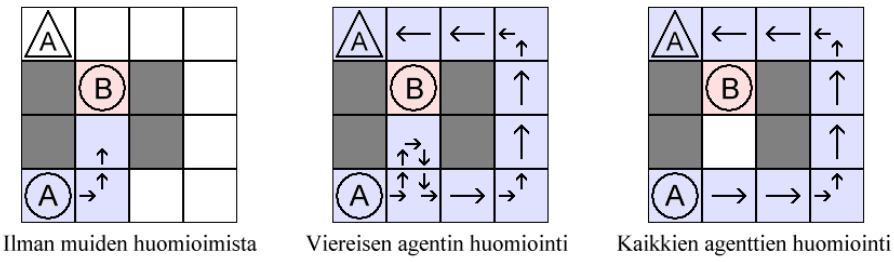
Polunetsinnän tekeminen ilman yhteistyötä on joskus aivan järkevääkin. Polunetsintä voidaan tehdä käyttäen verrattain helposti toteutettavia ja hyväksi todettuja yhden agentin algoritmeja. Lisäksi jokainen agentti voi suorittaa polunetsinnän itsenäisesti, eikä tarvita erityisiä agenttien välisiä tietorakenteita. Jos agentteja on vähän suhteessa ympäristön kokoon, polunetsintä ilman yhteistyötä voi olla hyvinkin riittävä ratkaisu.

Lisäksi, esimerkiksi juuri pelien tapauksessa, polunetsintä ilman yhteistyötä saattaa joskus olla ainoa mahdollinen ratkaisu. Esimerkiksi vastakkaisiin joukkoihin kuuluvien pelihahmojen välillä ei voi odottaakaan olevan yhteistyötä. Vihollisjoukkoon kuuluvalle hahmolle on jopa suotavaa pyrkiä häiritsemään toisen hahmon liikkumista tai yrittää tukkia sen käyttämiä reittejä. Suoraan tai epäsuorasti pelaajan ohjauksessa olevan pelihahmon reittejä ei pysty mitenkään ennustamaan, joten täysi yhteistyö tällaisten agenttien välillä on mahdotonta. Verkkopeleissä voi puolestaan olla mahdollista, että osa pelihahmoista on jonkin toisen tietokoneen ohjauksessa, joten myös tällaisten agenttien reittejä voidaan vain arvailla.

3.1 Lähestymistavat

Seuraavaksi on esitelty, millä eri tavoilla muita agentteja voidaan käsitellä polunetsintää tehdessä. Kuvassa 2 on yksinkertainen tilanne, missä on kaksi agenttia. Tummennetut ruudut kuvaavat solmuja, joista ei voi kulkea. Agentin A pitäisi päästä vasemmasta alakulmasta vasempaan yläkulmaan, mutta paikallaan pysyvä agentti B tukkii kapean käytävän.

Yksinkertaisin lähestymistapa olisi jättää muut agentit kokonaan huomiotta polu-



Kuva 2: Erilaisia tapoja huomioida muut agentit.

netsinnässä. Tällöin kuvan 2 agentti A päättää reitikseen kapean käytävän. Tästä seuraa luonnollisesti agentin A jumiutuminen agentin B taakse. Huonoimmillaan agentti jäisi tähän [SB06]. Agentti A voisi esimerkiksi jäädä odottamaan, että agentti B siirtyy pois tieltä. Toinen mahdollisuus on, että koko polunetsintä lopetetaan ja agentti jää odottamaan uutta ohjausta. Kumpikaan ratkaisu ei ole kovin kelvollinen, sillä agentin polunetsintä voidaan katsoa epäonnistuneeksi. Maalipaikkaa ei saavutettu, vaikka mahdollinen reitti on olemassa.

Muiden agenttien täydellisellä huomiotta jättämisellä on kuitenkin yksi hyvä puoli. Tällöin polunetsintää suorittavalle agentille selviää, onko reittiä lähdöstä maaliin ylipäätään olemassa siinä ympäristössä, missä agentti itse on [Vin97]. Mikäli reittiä ei ole alkuunkaan olemassa, saattaa agentin olla turha lähteä edes liikkeelle.

3.1.1 Viereisten agenttien huomiointi

Hieman parempi lähestymistapa on ottaa polunetsinnässä huomioon aivan vieressä olevat agentit käsittelemällä niitä ympäristöön kuuluvina staattisina esteinä. Agentin etsittyä reittinsä, se lähtee seuraamaan löydettyä reittiä, kunnes törmäys johonkin toiseen agenttiin on väistämätön. Tällöin agentti laskee itselleen uuden reitin nykysijainnistaan maalipaikkaan. Koska se ottaa polunetsinnässä huomioon viereisessä solmussa olevan agentin, johon oli juuri törmäämässä, uusi reitti kiertää tämän agentin ympäri. Tällaista tapaa kutsutaan nimellä Local Repair A* (LRA*) ja tämä on laajalti peleissä käytetty tapa suorittaa polunetsintä, kun agentteja on useita samassa ympäristössä [Sil05].

Kuvassa 2 polunetsintä saisi agentin A valitsemaan ensin reitin, joka vie kapean käytävän läpi. Kun se saapuu käytävällä olevan agentin B viereen, seuraava liike

tulisi aiheuttamaan törmäyksen. Tällöin agentille A etsitään uusi reitti, ja koska agentti B on aivan vieressä, se ajatellaan nyt ympäristöön kuuluvaksi esteeksi. Uusi reitti kiertäisi oikean reunan kautta maalipaikkaan. Tämä on jo paljon parempi kuin se, että agentti vain pysähtyisi.

Ongelmana on kuitenkin se, että törmäyksen toisena osapuolena oleva agenttikin saattaa olla liikkeessä. Kun törmäys havaitaan ja agentin reitti lasketaan uudestaan, ei silti ole taattua etteikö heti seuraavalla liikkeiden päivityskerralla samat agentit olisi jälleen törmäämässä toisiinsa. Ruuhkaisessa tilanteessa tämä voi aiheuttaa huomattavan määrän jatkuvaa uudelleenlaskentaa [Sil05].

Aina kun muut agentit ajatellaan esteiksi, ongelmana on myös se, että reittiä etsivä agentti voi harhautua pitkille kiertoteille. Näin tapahtuu, mikäli juuri agentin polunetsintävaiheessa jokin toinen agentti tukkii jonkin paremman reitin. Tämä ilmiö on havaittavissa esimerkiksi Baldur's Gate -pelissä, missä hahmot saattavat lähteä kiertämään kokonaista taloa, kun muut hahmot tukkivat hetkellisesti jonkin lyhyen reitin. Asiaa voisi osittain korjata niin, että jo liikkeessä oleville agenteille etsitään uusia reittejä jatkuvasti. Jokin hetkellisesti hyvää reittiä tukkinut toinen agentti voi siirtyä pois, ja näin parempi reitti vapautuu. Tästä tosin seuraa myös paljon turhaa laskentaa, sillä parempia reittejä ei välttämättä löydy tai parempi reitti voi hetken kuluttua tukkeutua uudestaan.

3.1.2 Kaikkien agenttien huomiointi

Myös kaikki muut, kuin vain viereiset agentit, voidaan käsitellä esteinä. Kuvan 2 tapauksessa tällainen polunetsintä löytäisi agentille sopivan reitin ja heti ensimmäisellä yrittämällä. Vaikka tässä nimenomaisessa tapauksessa polunetsintä toimii oikein, se ei kuitenkaan toimi sitten, jos muut agentit ovatkin liikkeellä. Jos muiden liikettä ei huomioida, törmäyksiä voi kuitenkin sattua, vaikka polunetsintävaiheessa niitä yritettiin välttää.

Lisäksi, mikäli staattisina esteinä kierretyt muut agentit liikkuvat, agentti saattaa jossain vaiheessa reittiä väistää toista agenttia, joka ei enää kyseisellä paikalla olekaan. Polunetsintä saattaisi siis tehdä, etenkin pitkän reitin kohdalla, paljon turhaakin työtä väistellessään esteitä, joita ei kuitenkaan paikalle saavuttaessa enää ole. Peleissä tämä ei myöskään pelaajan silmin näytä kovin älykkäältä liikkumiselta, pelihahmon väistellessä olemattomia muita pelihahmoja. Tätä näyttäisi tapahtuvan esimerkiksi Dune 2 -pelissä.

Vielä parannusta edelliseen olisi ajatella muut agentit liikkuvina esteinä. Tämä onnistuu tietysti vain, jos muiden agenttien suunta, nopeus ja ehkä kiihtyvyyskin ovat selvillä. Tällöin törmäykset voidaan jo välttää varsin hyvin, eikä reitille välttämättä tule ylimääräistä mutkittelua. Tämä vaatii kuitenkin jo huomattavasti lisää laskentatehoa ja monimutkaistaa polunetsintää, koska muiden agenttien tulevia reittejä joudutaan myös laskemaan jokaiselle ajanhetkelle. Tästä lisätyöstä huolimatta ei ole mitenkään taattua, etteikö törmäilyjä voisi silti syntyä, koska muiden agenttien suunta tai nopeus voivat muuttua polunetsinnän jälkeen.

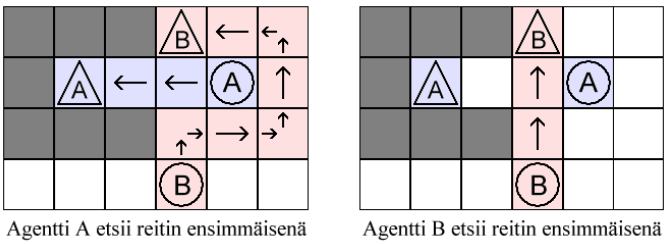
3.1.3 Polun lukitseminen

Polun lukitsemisella tarkoitetaan sitä, että agentin etsittyä reittinsä muita agentteja väistellen, kaikki reittiin kuuluvat solmut lukitaan niin, etteivät muut agentit voi niitä omassa polunetsinnässään käyttää [Vin97]. Lukittuja solmuja aletaan vapauttaa sitä mukaa, kun agentti on solmuja ohittanut. Tällä tavalla varmistetaan, että agentin reitti pysyy vapaana koko matkan ajan.

Polun lukitsemisen avulla yksi agentti saadaan kuljetettua ongelmitta perille, mutta tämä kuitenkin pahentaa tilannetta ympäristön tukkeutumisen kannalta. Sen lisäksi, että muut agentit tukkivat reittejä, nyt myös muiden varaamat reitit tukkivat niitä. Muut agentit voivat joutua kiertämään pitkiäkin matkoja lukittuja solmuja väistellessä, tai lukitut solmut voivat kokonaan estää maalipaikan löytämisen. Tukkeutumista voidaan helpottaa niin, että vaikka agentin reitti etsitään kokonaisuudessaan, reitistä lukitaan vain alkuosa [Vin97]. Kun agentti on kulkenut lukitun osan loppuun asti, suoritetaan polunetsintä uudestaan ja uudesta reitistä lukitaan taas vain alkuosa. Näin jatketaan kunnes ollaan koko reitin maalipaikassa.

Huomattavaa on myös, että polun lukitsemista käytettäessä agenttien polunetsintäjärjestyksellä voi olla vaikutusta [Vin97]. Kuvassa 3 on esimerkki kahden agentin tilanteesta. Ympyrät kuvaavat agentteja, kolmiot niiden maalipaikkoja ja tummennetut ruudut läpipääsemättömiä alueita. Agenttien reittivalinnat poikkeavat huomattavasti toisistaan, riippuen vain siitä kumman agentin reitti etsitään ensin. Agentin B etsiessä reittinsä ensimmäisenä, agentti A ei löydä reittiä maaliin ollenkaan.

Toinen samankaltainen tapa olisi muuttaa polunetsintäverkon kustannuksia korkeammaksi niiden solmujen kohdalla, jotka kuuluvat jonkin agentin käyttämään reittiin [Tom04]. Tällöin muut reittejä etsivät agentit eivät täysin hylkää mahdollisuutta käyttää näitä solmuja, mutta ne pyrkivät kuitenkin niitä välttämään. Jotkin



Kuva 3: Agentin reitin lukitsemisesta aiheutuvia ongelmia.

agentit voivat päätyä optimaalista pidemmille reiteille, mutta suurilta ruuhkilta voidaan välttyä, kun agenttien reittivalinnat hajaantuvat suuremmalle alueelle. Lisäksi tämän avulla vältytään juuri kuvan 3 kaltaisilta reittien tukkeutumisilta.

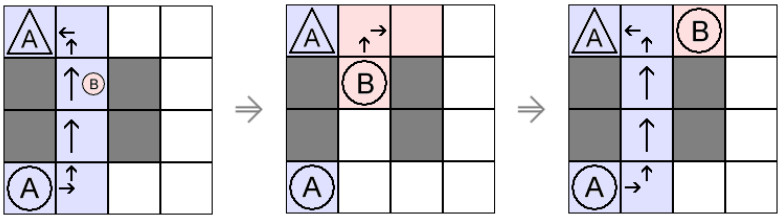
3.1.4 Muiden poistaminen reitiltä

Kun muut agentit käsitellään esteinä, ne saattavat joskus olla sijoittuneena niin, että minkäänlaista reittiä maalipaikkaan ei löydetä. Tällöin myös polunetsintään saattaa kulua erittäin paljon aikaa, sillä polunetsintäalgoritmit joutuvat yleensä käymään koko agentin saavutettavissa olevan verkon läpi yrittäessään löytää reittiä maaliin [Vin97]. Koko tämä laskentatyö on lisäksi turhaa, kun reittiä agentille ei kuitenkaan löydetä.

Jos vapaata reittiä ei ole, voidaan tietysti jäädä vain odottamaan, että sellainen vapautuu. Tämä ei välttämättä ole kovin hyvä ratkaisu, koska mitään takeita ei ole vapautuuko reittiä koskaan. Siksi voi olla tarvetta siirtää muita agentteja pois reitiltä [Vin97]. Tämä voidaan helposti ratkaista niin, että yksittäiselle agentille etsitään ensin reitti muista agenteista välittämättä. Tämän jälkeen kaikki muut agentit, jotka ovat kyseisen agentin reitillä, käsketään siirtymään pois. Siirtyminen voidaan tehdä vaikkapa viereiseen solmuun, kunhan tämä solmu ei myöskään kuulu liikkeelle lähtevän agentin reittiin.

Kuvassa 4 on sama tilanne kuin kuvassa 2. Nyt A valitsee reitin muista välittämättä ja B joutuu siirtymään pois omalta paikaltaan. Agentti A saa käyttöönsä optimaalisen reitin. Tällainen tapa voi kuitenkin aiheuttaa myös ketjureaktiota. Reitin päältä pois siirtyvät agentit voivat puolestaan saada joitain muita agentteja siirtymään pois heidän reitiltään. Muiden poistamista reitiltä voisi tosin pitää vain varasuunnitelma-

na, jota käytetään vain, jos muita vaihtoehtoja ei ole. Kuvan 4 tapauksessa agentilla A olisi mahdollisuus käyttää myös reittiä, joka ei edellytä muiden siirtymistä pois tieltä. Tosin tämä reitti olisi pidempi.



Kuva 4: Agentin reitinvalinta saa toisen agentin poistumaan paikaltaan.

Tällainen menettely on ehkä jo jonkinasteista agenttien yhteistyötä, sillä yhden agentin polunetsintä voi saada liikettä aikaan muissakin. Tämä ei kuitenkaan estä useamman yhtä aikaa liikkeessä olevan agentin törmäämistä toisiinsa. Eivätkä myöskään reitin päältä pois siirtyneet agentit palaa myöhemmin takaisin omalle paikalleen, ellei niitä siihen erikseen komenneta. Kovin koordinoitua yhteistyötä tämä ei siis ole, vaan enemmänkin liikennesääntömäisiä ohjeita muille agenteille.

3.2 Polunetsinnän tietojen uudelleen käyttäminen

Kun polunetsintä tehdään ilman yhteistyötä, törmäyksiä syntyy ja reittejä joudutaan laskemaan uudestaan. Aivan kaikkea ei kuitenkaan aina tarvitse laskea uudestaan. Esimerkiksi törmäyksen jälkeen, A*:n jo aiemmin selvittämät heuristiset arviot ovat usein uudelleenlaskennan kannalta täysin käyttökelpoista tietoa, sillä agentin maalipaikka ei ole muuttunut.

Dynamic A* (D*) [Ste93] on algoritmi, joka osaa laskea reittiin muutoksia ympäristön muuttuessa, käyttäen hyväksi alkuperäisen polunetsinnän tietoja. Mikäli muut agentit ajatellaan ympäristöön kuuluvina esteinä, ne aiheuttavat juuri tällaisia muutoksia. Tällöin D*:n avulla voisi nopeuttaa agentin reitin uudelleenlaskentaa. Tiettävästi D*:ä ei kuitenkaan ole käytetty ainakaan peleissä sen hankaluuden ja raskauden vuoksi [Tom04].

Myös vanhoja reittejä tai niiden osia voidaan käyttää uudelleen [Mil06]. Mikäli jossain vaiheessa on todettu, että reitti A-B-C-D-E on nopein reitti A:sta E:hen, tiedetään samalla, että nopein reitti B:stä D:hen on B-C-D. Lifelong Planning A*

(LPA*) [KL02] on algoritmi, joka säilyttää tietoa polkujen osista. LPA*:n kaltaiset algoritmit ja niiden tietojen uudelleenkäyttäminen on parhaimmillaan muistinkulutuksen kannaltakin hyödyllistä [Mil06]. Vaikka vanhojen reittien säilyttäminen kuluttaa muistia, voidaan uusien reittien etsimisessä säästää niin paljon aikaa ja työtä, että lopullinen muistinkulutus on pienempi, kuin jos jokainen reitti etsittäisiin aina alusta alkaen uudestaan.

4 Polunetsintä yhteistyöllä

Agentit pystyvät etsimään reittinsä yhteistyössä, jos ne jakavat tarkat tiedot omista reiteistään muille. Agenttien tehdessä yhteistyötä, voidaan agenttien väliset törmäykset välttää jo reittien suunnitteluvaiheessa. Tällöin agentit yleensä pääsevät maalipaikkaansa nopeammin, kun ne eivät törmäile toisiinsa. Peleissä hyvä yhteistyö omien yksiköiden välillä on usein tärkeää. Tiukassa pelitilanteessa pelihahmojen törmäily toisiinsa voi hidastaa niiden liikkumista niin paljon, että se on jopa ratkaisevaa itse pelin lopputuloksen kannalta [GCB06]. Vaikka polunetsintä agenttien yhteistyöllä on raskaampaa kuin ilman yhteistyötä, on myös mahdollista, että resursseja kuitenkin pitkällä aikavälillä säästyy, kun vältytään törmäilyjen aiheuttamilta uudelleenlaskemisilta.

Parhaimmillaan myös agentit, jotka eivät ole liikkeessä, olisivat mukana yhteistyössä. Täysin paikallaan pysyvä agentti voi aiheuttaa esimerkiksi joidenkin alueiden saavuttamattomuuden tai lukkiumatilanteita. Paikallaan olevien agenttienkin tarvitsee siis toisinaan siirtyä pois muiden agenttien tieltä ja palata sitten takaisin omaan paikkaansa.

Keskitetty algoritmi, joka etsisi kaikille agenteille sopivia reittejä yhtäaikaisesti, löytäisi jokaiselle optimaaliset reitit [ELP87]. Tällainen polunetsintä usealle agentille on kuitenkin todistettu PSPACE-vaikeaksi ongelmaksi [HSS84], eli kuuluu vaikeimpien laskennallisten ongelmien luokkaan. Agenttien määrän lisääntyessä polunetsinnän aikavaativuus nousisi eksponentiaalisesti [Sil05]. Laajoissa ympäristöissä tai suurella määrällä agentteja tällainen ratkaisu on usein liian raskas.

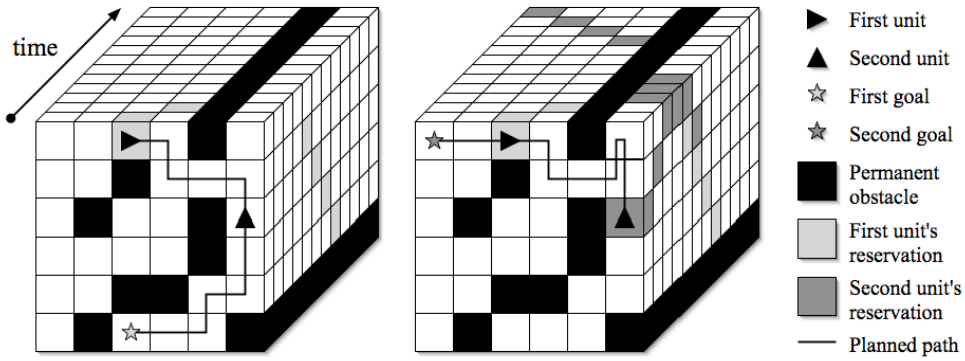
Parempi tapa tehokkuuden kannalta on jakaa ongelma pienempiin osiin etsimällä jokaiselle agentille reitti erikseen ja samalla voidaan ottaa huomioon myös agenttien tärkeysjärjestys [ELP87]. Aluksi etsitään reitti tärkeimmälle agentille. Tämän jälkeen toiseksi tärkeimmälle, mutta kuitenkin niin, että vältetään joutumasta törmäystilanteeseen ensimmäisen kanssa missään kohtaan reittiä. Tässä huomioidaan

myös aika, milloin agentti on missäkin kohtaa reittä. Näin toimitaan muidenkin agenttien kohdalla. Etsimällä tällä tavoin jokainen reitti erikseen, ei kaikille agenteille kuitenkaan välttämättä enää löydetä parhaita mahdollisia reittejä.

4.1 Cooperative A*

Cooperative A* (CA*) [Sil05] on erityisesti usean agentin polunetsintään tarkoitettu algoritmi, joka laajentaa polunetsintäverkon aikaulottuvuuteen. Sen sijaan, että agenteilla olisi esimerkiksi pelkkä sijainti (x, y), niillä onkin sijainti tietyllä ajanhetkellä (x, y, t). Tästä seuraa myös, että agenttien mahdollisiin liikkeisiin tarvitsee lisätä mahdollisuus odottaa paikallaan. Tällöin agentti liikkuu siis aika-avaruudessa solmusta (x, y, t) solmuun $(x, y, t + 1)$. Tämän jälkeen agenteille voidaan suorittaa polunetsintä käyttäen tavallista A*:ä, joka vain käsittelee aika-avaruudessa olevat muut agentit esteinä.

Kuvassa 5 on esimerkki 6*6 ruudun kokoisesta polunetsintäverkosta laajennettuna aikaulottuvuuteen ja kahdesta agentista. Yläreunassa oleva agentti etsii reitin ensimmäisenä (kuvan vasen puoli). Oikeassa reunassa oleva, jälkimmäisenä polunetsinnän suorittava agentti, osaa nyt väistää ensimmäistä agenttia, koska aika-avaruudesta löytyy reitti, joka ei risteä ensimmäisen agentin reitin kanssa (kuvan oikea puoli).



Kuva 5: Kaksi agenttia löytävät törmäämättömät reitit aika-avaruudessa [Sil06].

Käytännössä itse polunetsintäverkkoa ei tietenkään laajenneta uuteen ulottuvuuteen. Sen sijaan agenttien reittien tallentamiseen voidaan käyttää esimerkiksi hajautustaulua, käyttäen avaimena sijaintia ja aikaa (x, y, t) [Sil05]. Kun yhdelle agentille

etsitään reitti, se tallennetaan hajautustauluun. Kun seuraavalle agentille etsitään reittiä, sen ei sallita käyttävän samaa solmua samaan aikaan kuin mitä jokin toinen agentti on jo käyttänyt.

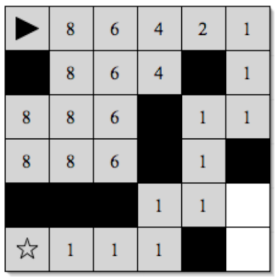
Yksi huomioitava ongelma on, että agenttien suoraa yhteentörmäystä ei välttämättä huomata [Sil06]. Jos yksi agentti varaa sijainnit (x, y, t) ja $(x+1, y, t+1)$, eli agentti liikkuu oikealle, voi jokin toinen agentti kuitenkin varata sijainnit $(x + 1, y, t)$ ja $(x, y, t + 1)$. Agentit eivät käyttäneet samoja sijainteja aika-avaruudessa mutta kulkivat kuitenkin toistensa läpi. Tämä voidaan estää kahdella tavalla [Sil06]. Ensimmäinen tapa on, että agentit reittiä varatessaan varaavat aina kaksi sijaintia. Paikalla (x, y) oleva agentti siis varaa sekä sijainnin (x, y, t) että $(x, y, t + 1)$. Toinen tapa on, että tällaiset törmäykset tutkitaan erikseen, eikä niitä yksinkertaisesti sallita tapahtuvan.

On mahdollista, että agenttien koko tai nopeus eroaa toisistaan. CA* toimii kuitenkin myös tässä tapauksessa [Sil06]. Suurempi yksikkö vain varaa useampia solmuja samaan aikaan. Hitaammat yksiköt puolestaan varaavat saman solmun pidemmän aikaa.

A*-algoritmin suorittaminen aika-avaruudessa tuottaa kuitenkin myös yhden tehokkuusongelman [Sil05]. Mikäli A*:n käyttämät heuristiset arviot ovat joissain kohdin kovin optimistisia, aika-avaruudessa toimiva A* voi juuttua pitkäksi aikaa jonkin esteen taakse. Tämä johtuu siitä, että odottaminen (liikkuminen eteenpäin vain ajassa) voi polunetsinnän mielestä olla hyvä ratkaisu, jos arvio loppumatkan kustannuksesta on pieni. Vasta kun polunetsintäalgoritmi on odottanut esteen takana pitkään, alkaa matkaan käytetty kustannus nousta niin suureksi, että peruuttaminen maalipaikasta poispäin alkaa näyttämään paremmalta ratkaisulta. Kuvassa 6 on esimerkki, missä agentin lähtöpaikka on vasemmassa yläkulmassa ja maalipaikka vasemmassa alakulmassa. Jokaisessa ruudussa arviona loppumatkan kustannuksesta on käytetty Manhattan-etäisyyttä. Ruuduissa olevat numerot kertovat, kuinka monta kertaa A* vieraili kussakin ruudussa reittiä etsiessään.

4.1.1 Hierarchical Cooperative A*

CA*-algoritmin ongelma, jossa A* saattaa tutkia samoja solmuja moneen kertaan, voidaan ratkaista antamalla A*:n käyttöön mahdollisimman tarkat heuristiset arviot. Hierarchical Cooperative A* (HCA*) [Sil05], joka on jatkoa CA*:lle, tekee tämän. HCA*-algoritmissa lasketaan täydelliset heuristiset arviot loppumatkan kus-



Kuva 6: Aika-avaruudessa A* voi tutkia samat solmut moneen kertaan [Sil06].

tannuksesta käyttäen Reverse Resumable A* -algoritmia (RRA*).

RRA* [Sil05] toimii niin, että normaalia A*:ä käyttäen lähdetään etsimään reittiä maalipaikasta lähtöpaikkaan huomioimatta muita agentteja. Tällöin kuljetun matkan kustannus jokaisessa solmussa mihin saavutaan on samalla täysin paikkansa pitävä arvio loppumatkan kustannuksesta, kun reittiä etsitään toiseen suuntaan. Olettaen tietysti, että verkossa kahden solmun välillä kulkeminen on molempiin suuntiin kustannukseltaan sama. Tarkoilla loppumatkan kustannusarvioilla varsinainen polunetsintä lähtöpaikasta maalipaikkaan toimii erittäin nopeasti aika-avaruudessakin.

HCA*-algoritmissa toimii siis rinnakkain kaksi polunetsintää. RRA* etsii reittiä lopusta alkuun ottamatta huomioon aikaa tai muita agentteja. Varsinainen polunetsintä etsii reittiä alusta loppuun huomioiden muiden agenttien reitit kuten CA*:ssä. RRA*-algoritmin suoritusta jatketaan aina tarpeen vaatiessa. Kun varsinaista polunetsintää suoritetaan ja saavutaan solmuun, johon ei ole vielä laskettu heuristista arviota, palataan RRA*:n suorittamiseen. RRA*-algoritmin suoritusta jatketaan niin pitkälle kunnes päästään haluttuun solmuun. Sen jälkeen palataan taas takaisin varsinaiseen polunetsintään.

Vaikka HCA*:ssä tehdään kahta polunetsintää, se saattaa tietyissä tilanteissa olla nopeampi kuin CA*. Esimerkiksi kuvan 6 tapauksessa HCA* selviäisi paljon vähemmällä työllä. RRA* voi vierailla jokaisessa ruudussa enintään kerran, ja kun tarkat heuristiset arviot on tiedossa, CA* löytää oikean reitin välittömästi [Sil06]. Huomattavaa on, että RRA*:n oikeanlainen toiminta edellyttää, että RRA*:n itsensä käyttämät heuristiset arviot toteuttavat tietyt ominaisuudet [Sil05]. Tähän ei kuitenkaan tässä tutkielmassa puututa tarkemmin.

4.1.2 Windowed Hierarchical Cooperative A*

CA*- ja HCA*-algoritmeissa on se ongelma, että muita agentteja väistelevän reitin etsintä tehdään koko matkalle alusta loppuun [Sil05]. Vaikka ideaalista tietysti olisikin aina etsiä muihin törmäämätön reitti koko matkalle, se saattaa monessa tapauksessa olla aivan turhaa. Mitä pidempi matka on, sitä todennäköisemmin muiden agenttien reitit voivat ehtiä muuttumaan, ja iso osa laskentatyöstä on mennyt hukkaan.

Windowed Hierarchical Cooperative A* (WHCA*) [Sil05] ratkaisee tätä ongelmaa katkaisemalla agenttien yhteistyötä tekevän polunetsinnän tietylle matkalle. Agenttien toisiinsa törmäämättömät reitit etsitään esimerkiksi vain 16 askeleen päähän. Kun agentit ovat päässeet tämän reitin esimerkiksi puoleen väliin asti, etsitään uudet reitit jälleen 16 askeleen päähän. Koska WHCA* käyttää pohjana HCA*:ä, joka puolestaan käyttää tarkkoja heuristisia arvioita loppumatkasta, on uusien reittien etsiminen erittäin tehokasta. Tarkasti tiedossa olevan loppumatkan kustannuksen vuoksi ei myöskään ole vaarana, että agentti ajautuisi paikalliseen umpikujaan, vaikka agentin koko reittiä ei lasketakaan loppuun asti.

WHCA*-algoritmin tapauksessa on mahdollista, että agentti jatkaa algoritmin suorittamista vielä maalipaikkaan pääsemisen jälkeenkin [Sil05]. Tällöin agentti jatkaa yhteistyön tekemistä muiden agenttien kanssa, eikä se esimerkiksi jää tukkimaan kapeaa reittiä estäen muiden agenttien kulkua. Agentti voi siis hetkeksi aikaa poistua omasta maalipaikastaan, jos jokin toinen agentti sitä tarvitsee tiettynä hetkenä ja palata sen jälkeen takaisin tähän omaan maalipaikkaansa. Jos jokin agentti on omassa maalipaikassaan, odottaminen kyseisessä solmussa katsotaan kustannukseltaan nollaksi [Sil06]. Tällöin polunetsintä maaliin päässeen agentin kohdalla, esimerkiksi juuri 16 askeleen päähän, on erittäin nopeaa tehdä.

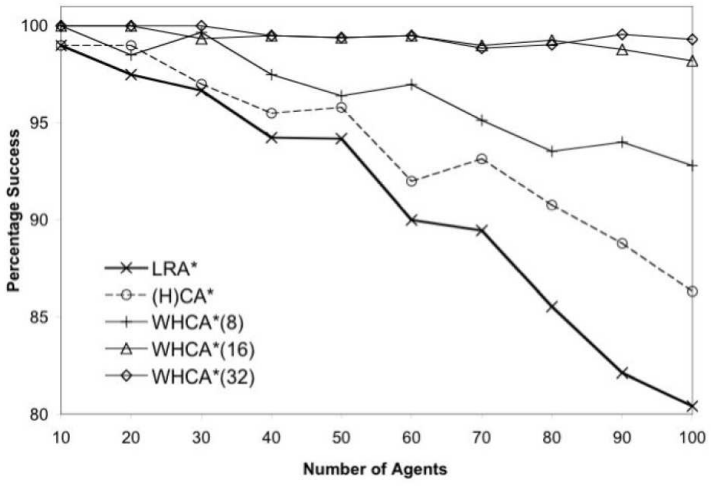
Sopivan ikkunan koon valitseminen on tärkeää WHCA*:ssä [Sil05]. Jos ikkuna on liian suuri, algoritmi muistuttaa enemmän HCA*:ä, joka saattaa tehdä paljon turhaa työtä. Jos ikkuna on liian pieni, WHCA* alkaa muistuttamaan enemmän LRA*:ä, koska yhteistyön tekeminen supistuu vain pienelle matkalle. Kun WHCA*:n tuottamaa osittaista reittiä kuljetaan eteenpäin, tärkeää on myös päättää missä kohdin reitti lasketaan uudestaan [Sil06]. Agentin yhteistyö muihin nähden vähenee koko ajan, mitä lähemmäs osittaisen reitin loppua mennään, koska yhteistyö on suunniteltu nimenomaan vain tämän osittaisen reitin osuudelle.

WHCA*-algoritmissa agenttien polunetsintä voidaan vielä limittää, mikä ratkaisee

erilaisia lukkiumatilanteita [Sil06]. Esimerkiksi kahden agentin tapauksessa agentti A etsii törmäämättömän reitin 16 askeleen päähän ja agentti B tekee samoin, mutta vasta, kun agentti A on liikkunut neljä askelta. Tällöin agentilla B on täysi valinnanvapaus oman reittinsä suhteen neljän viimeisen askeleen aikana, koska kukaan muu agentti ole vielä niin pitkälle tehnyt mitään varauksia. Toisin sanoen agenttien prioriteetit reitin valinnan suhteen vaihtelevat koko ajan.

Kaikkia lukkiumatilanteita WHCA* ei kuitenkaan selvitä [SB06]. Esimerkkinä kaksi agenttia ovat pitkässä käytävässä ja niiden pitäisi päästä toistensa ohi. Tämä vaatisi, että toinen agenteista peruuttaisi ensin pois käytävästä. Jos käytävä on kuitenkin liian pitkä suhteessa ikkunan kokoon ja agentit vuorottelevat sitä, kumpi saa etsiä reitin ensin, agentit voivat jäädä vain työntämään toisiaan käytävässä edestakaisin.

Algoritmeja testattiin 32*32 kokoisessa ruudukossa, missä satunnaiset 20% ruuduista oli läpipääsemättömiä [Sil05]. Myös agenttien lähtö- ja maalipaikat arvottiin satunnaisesti, mutta kuitenkin niin ettei kahdella agentilla ole samaa lähtö- tai maalipaikkaa. Kuvassa 7 näkyy testin tulokset. WHCA*(w) viittaa WHCA*-algoritmiin jonka ikkunan koko on w. Kuva esittää kuinka monta prosenttia agenteista onnistui pääsemään maalipaikkaansa 100 vuoron aikana ja ilman törmäyksiä muihin agentteihin. Erityisesti WHCA*:n parempi toimivuus verrattuna LRA*:en oli sitä ilmeisempi, mitä useampia agentteja ympäristössä oli yhtä aikaa. WHCA*-algoritmin ikkunan koolla 16 ja 32 onnistumisprosentti oli lähellä sataa agenttien määrästä riippumatta.

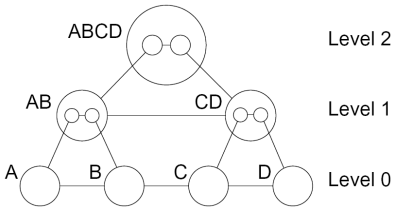


Kuva 7: Onnistuneesti perille päässeiden agenttien määrä eri algoritmeilla [Sil05].

4.2 Cooperative Partial-Refinement A*

WHCA*-algoritmissa on kaksi ongelmaa [SB06]. Ensinnäkin WHCA* vaatii verrattain pitkät alkulaskelmat, ennen kuin agentteja voidaan siirtää ensimmäistäkään askelta. Jokaisen agentin tarvitsee suorittaa ensin RRA* koko reitille, ennen kuin päästään edes suorittamaan varsinaista yhteistyötä tekevää polunetsintää. Toisekseen WHCA* kuluttaa myös paljon muistia, koska jokainen agentti joutuu pitämään tallessa RRA*-algoritmin tietoja koko polunetsinnän ajan. Näitä molempia ongelmia voidaan pienentää yhdistämällä Partial-Refinement A* -algoritmi (PRA*) WHCA*:en [SB06].

PRA* [SB05] on tavallisesti yhden agentin polunetsintään tarkoitettu algoritmi, joka nopeuttaa polunetsintää suorittamalla sen ensin karkeammassa verkossa. Karkeampi verkko on saatu aikaiseksi yhdistämällä varsinaisen polunetsintäverkon solmuja toisiinsa. Näitä karkeita verkkoja voi myös olla useita eritasoisia. Kuvassa 8 on yksinkertainen esimerkki, missä alimmalla tasolla on alkuperäinen verkko. Mikäli etsitään reittiä esimerkiksi solmusta A solmuun D, tästä rakenteesta nähdään nopeasti, että ensin pitää päästä solmuun B. Polunetsintä koko matkalta suoritetaan siis ensin karkeassa verkossa, missä se vähäisemmän solmumäärän vuoksi on nopeaa. Tarkempaa polunetsintää alemmilla tasoilla voidaan tehdä vain lyhyille matkoille kerrallaan, koska karkean reitin vuoksi tiedetään kuitenkin suunta, mihin ollaan menossa. PRA* toimii tavallista A*:ä nopeammin, mutta toisaalta reitit eivät enää välttämättä ole yhtä optimaalisia.



Kuva 8: Solmuja yhdistämällä on saatu useita eritasoisia verkkoja [SB05].

PRA* voidaan yhdistää WHCA*:en kahdella tavalla [SB06]. Ensimmäinen tapa on, että RRA* laitetaan käyttämään PRA*:n tapaan karkeampaa verkkoa. Tällöin algoritmia voidaan merkitä WHCA*(w, a), missä w on ikkunan koko ja a kertoo, kuinka karkealla tasolla RRA* suoritetaan [SB06]. Kun laskenta tehdään karkeammassa verkossa ($a > 0$), RRA* ei myöskään enää palauta täsmällisen tarkkoja tietoja lop-

pumatkan kustannuksesta WHCA*:n käyttöön. Mitä suurempi a on, sitä nopeammin RRA* toimii ja ensimmäinen askel saadaan tehtyä. Samalla kuitenkin RRA*:n palauttamat heuristiset arviot muuttuvat epätarkemmiksi.

Toinen ja yksinkertaisempi tapa on, että agentille etsitään reitti vain PRA*-algoritmia käyttäen, mutta kun PRA* pääsee alimmalle tasolla, käytetäänkin WHCA*:ä normaalin A*:n sijaan. Eli ensin etsitään PRA*:llä karkea reitti ja vasta, kun saavutaan varsinaisen polunetsintäverkon tasolle, otetaan käyttöön yhteistyötä tekevä WHCA*. Tällöin WHCA* toimii huomattavasti nopeammin, koska matka, johon WHCA*:ä käytetään, on paljon alkuperäistä lyhyempi. Tätä algoritmia kutsutaan nimellä Cooperative Partial-Refinement A* (CPRA*) [SB06]. Kun agentti on kulkenut jonkin tietyn matkan, suoritetaan CPRA* kokonaan uudelleen agentin nykysijainnista maaliin.

Merkinnässä CPRA*(k) muuttuja k viittaa siihen, kuinka pitkälle matkalle WHCA*:ä käytetään. CPRA*(k):n suoritusaika riippuu pitkälti k:n arvosta ja hyvänä puolena on, että sitä voi vaihdella tilanteen mukaan [SB06]. Mikäli jonain hetkenä polunetsintään on käytettävissä vain vähän aikaa, voidaan käyttää pienempää k:n arvoa.

Sekä WHCA*(w, a)- että CPRA*-algoritmeja testattiin erikokoisissa verkoissa ja erilaisilla määrillä agentteja [SB06]. Näiden tulosten pohjalta, algoritmit tekevät sen, mitä niiltä odotettiinkin. WHCA*(w, a) kuluttaa sitä vähemmän muistia, mitä suurempi on a. Sekä WHCA*(w, a) että CPRA* joutuvat tekemään agentin ensimmäisen siirron eteen vähemmän työtä kuin pelkkä WHCA*(w). CPRA* toimii kuitenkin ensimmäisen askeleen jälkeen hitaammin kuin WHCA*, koska CPRA* ei varastoi mitään tietoa agentin seuraavaan reitin päivityskertaan. WHCA*(w, a):n ja CPRA*:n löytämien lopullisten reittien pituuksilla ei kuitenkaan ollut juurikaan eroa WHCA*(w):en verrattuna.

4.3 Biased Cost Pathfinding

Biased Cost Pathfinding (BCP) [GCB06] on meta-algoritmi, jonka voi liittää melkein mihin tahansa yhden agentin polunetsintäalgoritmiin, kuten A*:en. Aluksi kaikille agenteille lasketaan reitti ottamatta huomioon muita agentteja. Kun agenttien reitit on laskettu, ne tallennetaan muistiin niin, että jokaisen agentin sijainti jokaisella ajanhetkellä on tiedossa. Haluttaessa voidaan tallentaa vain osa jokaisen reitin alusta, ei kokonaisia reittejä. Kun tämä on tehty, tarkistetaan mitkä agentit törmäisivät toisiinsa ja missä. Jos törmäyksiä havaitaan, lisätään törmäyksessä mukana ollei-

den agenttien polunetsintäverkkoon lisäkustannusta kyseiseen törmäyspaikkaan ja sen lähiympäristöön. Mitä useampia agentteja törmäyksessä oli mukana, sitä enemmän kustannusta lisätään. Tämän jälkeen lasketaan havaitussa törmäyksessä olleiden agenttien reitit uudelleen. Lisätyn kustannuksen vuoksi agentit pyrkivät nyt välttämään kyseistä törmäyspaikkaa. Tätä voidaan jatkaa niin pitkään kunnes löydetään kaikille agenteille reitit, joissa ei tapahdu törmäyksiä.

BCP voi huomioida myös sen, että agenteilla on erilaisia prioriteetteja [GCB06]. Mikäli tärkeä ja vähemmän tärkeä agentti törmäävät toisiinsa, voidaan tärkeämmän antaa käyttää löytämäänsä reittiä. Kustannuksen lisäys ja reitin uudelleenlaskenta tehdään vain vähemmän tärkeälle agentille. Tällä tavalla tärkeämmät agentit saavat käyttöönsä suorempia reittejä ja vähemmän tärkeät joutuvat väistämään näitä.

BCP:ssä yhden iteraatiokerran aikavaativuus on liki samaa luokkaa, kuin jos etsittäisiin jokaiselle agentille reitti ilman mitään yhteistyötä [GCB06]. Yleensä jokaisella iterointikerralla agenttien reitit muuttuvat aina vähemmän törmäyksiä sisältäviksi. Algoritmin hyvänä puolena on se, että iterointi voidaan myös tarvittaessa keskeyttää [GCB06]. Tällöin agenteille käytetään reittejä, jotka on siihen mennessä löydetty. Ne eivät välttämättä ole sellaisia, joissa törmäyksiä ei tapahtuisi ollenkaan, mutta toisaalta parhaat, mitä algoritmille varatussa ajassa oli mahdollista löytää.

Algoritmia testattiin käyttäen A*:ä varsinaisessa polunetsinnässä [GCB06]. BCP:n kanssa agentit pääsivät nopeammin perille kuin pelkällä A*:llä. Ero ei tehdyissä testeissä ollut merkittävä, mutta ero BCP:n hyväksi kasvoi agenttien määrän lisääntyessä. Huomattavampaa kuitenkin oli, että korkean prioriteetin agentti pääsi testeissä aina maaliinsa lyhintä mahdollista reittiä. Pelkällä A*:llä näin ei käynyt juuri koskaan.

5 Polunetsinnän vähentäminen

Joskus polunetsintää voi olla tehtävänä resursseihin nähden liikaa. Kaikille agenteille ei ole yksinkertaisesti aikaa laskea reittejä niin usein kuin olisi tarvetta. Seuraavissa aliluvuissa on esitelty kaksi tapaa, joiden avulla polunetsintää voidaan vähentää nimenomaan usean agentin ympäristössä.

5.1 Liikkuminen ryhmässä

Yksi tapa suorittaa polunetsintää usealle agentille olisi agenttien liikuttaminen yhtenä ryhmänä. Tämä tietysti onnistuu vain, jos kaikilla agenteilla on sama määränpää ja agentit ovat jo lähtiessään niin lähellä toisiaan, että ne voidaan käsittää yhdeksi ryhmäksi. Peleissä tällainen tapaus voi olla yleinenkin. RTS-peleissä pelaaja usein valitsee kokonaisen joukon omia yksikköjään ja käskee niiden siirtyä toiseen paikkaan. FPS-peleissä taas vihollisjoukoilla on yleensä yksi ja sama kohde, eli pelaaja itse.

Polunetsinnän kannalta yksinkertainen ryhmä olisi tiukka muodostelma, koska tällöin koko ryhmää voitaisiin käsitellä yhtenä suurena agenttina. Tässä tapauksessa myös polunetsintäalgoritmin pitäisi huomioida, että etsittävä reitti on sellainen, että koko muodostelma mahtuu siitä kulkemaan [Pot99]. Sopivan reitin löydyttyä, itse liikkuminen tiukassa muodostelmassa on helppoa, koska kaikki agentit liikkuvat aina samalla vauhdilla ja samaan suuntaan.

Mikäli agentit ovat vapaampi ryhmä, etsitään ensin reitti yhdelle agentille. Muut ryhmän jäsenet seuraavat sitten tätä ryhmää vetävää agenttia, sen seuratessa varsinaista polunetsinnän löytämää reittiä [Pot99, Chi04]. Ryhmän vetäjää voi myös vaihtaa niin, että aina etummaisin jäsen on seuraamassa polunetsinnän luomaa varsinaista polkua [Chi04]. Yksi ongelma ryhmän tapauksessa on agenttien mahdollisesti erilaiset liikkumisnopeudet. Periaatteessa koko ryhmän pitäisi liikkua hitaimman vauhdilla, mutta toisaalta esimerkiksi peleissä voidaan myös sallia hitaampien liikkuminen normaalia nopeammin niiden ollessa mukana ryhmässä [Pot99]. Millä tavalla muut agentit sitten seuraavat vetäjää, voidaan toteuttaa eri tavoilla. Muut agentit voisivat esimerkiksi käyttää vain jonkinlaista parveilualgoritmia pysyäkseen vetävän agentin lähellä, mutta välttääkseen törmäilyä toisiinsa [Tom04].

Vapaamman ryhmän tapauksessakin olisi hyvä, jos polunetsintäalgoritmi jo etsintävaiheessa huomioisi, että reitillä on kapasiteettia kuljettaa useita agentteja. Toisin sanoen polunetsinnän pitäisi pyrkiä välttämään kapeita kohtia, jotka muodostuisivat pullonkaulaksi ryhmän saapuessa paikalle. Tästäkin huolimatta voi tulla tilanteita, että jonkin esteen vuoksi ryhmä joudutaan jakaa osiin. Tällöin ryhmä pitää vain koota uudelleen esteen ohituksen jälkeen [Pot99]. Mikäli tilanne sallii, ryhmä voitaisiin myös hajottaa pienemmiksi ryhmiksi niin, että jokainen pienempi ryhmä jatkaa omana itsenäisenä ryhmänään.

5.2 Törmäysten salliminen

Yleensä kun agentit ovat törmäämässä toisiinsa, joudutaan polunetsintää tekemään uudelleen, jotta törmäys voidaan välttää. Mikäli agentit ovat kuitenkin virtuaalisia hahmoja, ei yhteentörmäämisestä ole mitään varsinaista haittaakaan. Mikäli törmäys vielä tapahtuu sellaisessa paikassa, ettei virtuaalisen maailman tarkkailija sitä näe, ei juurikaan ole mieltä käyttää aikaa uuteen polunetsintään [ACF01]. Sen sijaan voidaan esimerkiksi vain hidastaa agenttien vauhtia vastaamaan väistämiseen kuluvaa aikaa.

Peleissä tämä tarkkailija on tietysti pelaaja itse ja näkymättömissä olevat agentit ovat näytön ulkopuolella olevat pelihahmot. Tämä menetelmä voikin sopia erityisen hyvin peleihin. Pelaaja harvoin kiinnittää huomiota siihen, onko pelihahmojen liikkuminen aivan täydellistä [Tom04]. Pelikokemuksen kannalta on usein myös tärkeämpää, että peli toimii sujuvasti, kuin se, että jokainen pelihahmo liikkuu täydellisesti.

On kuitenkin olennaista, että virtuaalinen maailma kokonaisuudessaan käyttäytyy tarkkailijan näkökulmasta järkevästi ja siksi myös näkymättömissä olevien agenttien kohdalla pitää ottaa huomioon seuraavat asiat [ACF01]:

- agentti ei voi ohittaa maailmaan kuuluvia kiinteitä esteitä,
- agentti ei voi ohittaa muita agentteja, jotka täysin tukkivat jonkin kapean reitin ja
- ruuhkaisessa tilassa agentti liikkuu hitaammin, koska se todellisuudessa joutuisi väistelemään muita agentteja.

Agenttien liikuttamista tehdään siis kahdella tasolla. Näkyvissä olevat agentit liikkuvat normaalilla, tarkalla tasolla. Näkymättömissä olevat agentit puolestaan liikkuvat epätäsmällisemmin. Agenttien täytyy kuitenkin myös voida siirtyä näiden kahden tason välillä. Kun agentti saapuu tarkkailijan nähtäville, se siirtyy tarkempaan liikkumiseen ja päinvastoin.

Agenttien liikkuminen ei ole tällä tavoin täsmälleen samanlaista, kuin jos kaikkien agenttien liikkeet suoritettaisiin tarkasti. Tällainen menetelmä saattaa kuitenkin olla ainoa mahdollinen tapa suorittaa polunetsintää ja agenttien liikuttamista, jos ympäristö on todella suuri tai agentteja on todella paljon [ACF01].

6 Yhteenveto

Usean agentin ympäristössä agentit saattavat törmäillä toisiinsa tai tukkia toistensa reittejä. Yleensä parhaimpiin tuloksiin polunetsinnässä usean agentin ympäristössä päästään, jos agentit jakavat tarkat reittisuunnitelmat keskenään. Cooperative A* ja sen muunnelmat tarjoavat tähän varsin hyvän ratkaisun. Samalla polunetsintä tosin muuttuu monimutkaisemmaksi ja usein myös raskaammaksi.

Aina yhteistyö ei kuitenkaan ole mahdollista. Resurssit eivät ehkä riitä tai agenttien reittisuunnitelmat eivät ole muiden agenttien saatavilla. Ilman yhteistyötä polunetsintä on helpompaa, mutta törmäilyjä tapahtuu. Ilman yhteistyötä saattaa myös syntyä ongelmia, jotka eivät ratkea puhtaasti polunetsinnän keinoin. Agentit voivat esimerkiksi joutua työntämään toisiaan pois reitiltä. Joskus on myös mahdollista vähentää polunetsintää liikuttamalla agentteja ryhmässä tai sallimalla agenttien törmääminen toisiinsa.

Polunetsintään usean agentin ympäristössä on vaikeaa löytää yhtä ja ainoaa hyvää ratkaisua. Eri toteutustapojen toimivuuteen vaikuttaa useat eri asiat, kuten muiden agenttien määrä ja niiden liikkeiden ennustettavuus. Polunetsintäverkon koko ja etsittävien reittien keskimääräiset pituudet voivat myös vaikuttaa asiaan. Lisäksi polunetsintään käytettävissä olevien resurssien määrä voi rajata eri vaihtoehtoja.

Tietokonepelit ovat yksi alue, jossa polunetsinnälle on nykyisin paljon käyttöä, ja peleissä monet ovat ehkä käytännössä polunetsintään törmänneet. Pelien kehittyessä aina paremmiksi, myös pelihahmojen tekoälyltä ja sen myötä polunetsinnältä odotetaan aina enemmän. Suuntaus lienee tulevaisuudessa oikeasti yhteistyötä tekeviin algoritmeihin. Tässäkin tutkielmassa esitellyt yhteistyön mahdollistavat algoritmit ovat kaikki varsin uusia.

Lähteet

ACF01 Arikan, O., Chenney, S. ja Forsyth, D., Efficient multi-agent path planning. *Computer Animation and Simulation '01*, Magnenat-Thalmann, N. ja Thalmann, D., toimittajat. Springer-Verlag Wien New York, 2001, sivut 151–162.

Chi04 Chiou, T., Roadmap-based methods for flocking motion with obstacles. Itsenäisen opiskeluprojektin www-sivu, valvojana professori David Mount, University of Maryland, USA, toukokuu 2004. `http://www.cs.umd.edu/~mount/Indep/Kevin/Webpage/`. [3.5.2007]

ELP87 Erdmann, M. ja Lozano-Perez, T., On multiple moving objects. *Algorithmica*, 2,4(1987), sivut 477–521.

GCB06 Geramifard, A., Chubak, P. ja Bulitko, V., Biased cost pathfinding. *Second Annual Artificial Intelligence and Interactive Digital Entertainment Conference, AAAI*, Marina del Rey, Kalifornia, USA, kesäkuu 20.-23. 2006. `http://www.cs.ualberta.ca/~alborz/papers/BCP.pdf`. [3.5.2007]

HNR68 Hart, P., Nilsson, N. ja Raphael, B., A formal basis for the heuristic determination of minimum-cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4,2(1968), sivut 100–107.

HSS84 Hopcroft, J. E., Schwartz, J. T. ja Sharir, M., On the complexity of motion planning for multiple independent objects: Pspace-hardness for the warehousman's problem. *International Journal of Robotics Research*, 3,4(1984), sivut 76–88.

KL02 Koenig, S. ja Likhachev, M., Incremental A*. *Advances in Neural Information Processing Systems 14*, Dietterich, T., Becker, S. ja Ghahramani, Z., toimittajat. MIT Press, 2002, sivut 1539–1546.

Mil06 Millington, I., *Artificial Intelligence for Games*. Morgan Kaufmann Publishers, 2006.

Pot99 Pottinger, D., Implementing coordinated unit movement. *Game Developer Magazine*, 6,2(1999), sivut 48–59.

SB05 Sturtevant, N. ja Buro, M., Partial pathfinding using map abstraction and refinement. *The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, Pittsburgh, Pennsylvania, USA, heinäkuu 9.-13. 2005.

SB06 Sturtevant, N. ja Buro, M., Improving collaborative pathfinding using map abstraction. *Second Annual Artificial Intelligence and Interactive Digital Entertainment Conference, AAAI*, Marina del Rey, Kalifornia, USA, kesäkuu 20.-23. 2006.

Sil05 Silver, D., Cooperative pathfinding. *The 1st Conference on Artificial Intelligence and Interactive Digital Entertainment*, Young, M. ja Laird, J., toimittajat. AAAI Press, 2005, sivut 117–122. `http://www.cs.ualberta.ca/~silver/research/publications/files/CoopPath05.pdf`. [3.5.2007]

Sil06 Silver, D., Cooperative pathfinding. Teoksessa *AI Game Programming Wisdom 3*, Charles River Media, 2006. `http://www.cs.ualberta.ca/~silver/research/publications/files/cooperative-aiwisdom.pdf`. [3.5.2007]

Ste93 Stentz, A., Optimal and efficient path planning for unknown and dynamic environments. Tekninen raportti CMU-RI-TR-93-20, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, elokuu 1993.

Sto96 Stout, B., Smart moves: Intelligent pathfinding. *Game Developer Magazine*, 3,5(1996), sivut 28–35.

Tom04 Tomlinson, S., The long and short of steering in computer games. *International Journal of Simulation*, 5,1-2(2004), sivut 33–46. `http://ducati.doc.ntu.ac.uk/uksim/journal/Vol-5/No-1&2/TOMLINSON.pdf`. [3.5.2007]

Vin97 Vincke, S., Real-time pathfinding for multiple objects. *Game Developer Magazine*, 4,3(1997), sivut 36–44.