Peliteoreettisten mallien soveltaminen toimintapeleihin

Ismo Puustinen

Helsinki 15.5.2006

Pro Gradu -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
Ismo Puustinen

Työn nimi — Arbetets titel — Title
Peliteoreettisten mallien soveltaminen toimintapeleihin

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro Gradu -tutkielma	15.5.2006	74 sivua

Tiivistelmä — Referat — Abstract

Peliteoria on tieteenala, joka tutkii rationaalisten agenttien keskinäistä kanssakäymistä ja konfliktitilanteita. Tietokonepeleissä on usein käsiteltävänä tilanne, jossa tietokoneen ohjaamat agentit yrittävät saada toimintansa ja vuorovaikutuksensa näyttämään ihmispelaajan silmissä älykkäältä. Koska peliteoria tarjoaa mekanismeja, joilla agentit voivat päästä kanssakäymisessään kaikkia tyydyttävään ratkaisuun, tässä tutkielmassa tarkastellaan peliteorian soveltamista tietokonepelien maailmaan. Erityishuomiota annetaan tapaukselle, jossa joukko vihollisagentteja yrittää päästä pelaajahahmon vierelle mahdollisimman älykkäällä tavalla: piilotellen, väijyen ja voimavarojaan keskittäen. Tätä tilannetta tutkitaan sekä normaali- että laajamuotoisten pelien näkökulmasta.

Tutkielman aluksi käsitellään peliteorian perusteita ja tietokonepelien ongelmakenttää. Sen jälkeen ratkaistaan tekniset ongelmat: Nashin tasapainotilan soveltuvuutta agenttiryhmän päätöksentekoon pohditaan ja pelimatriisin rakentaminen sekä tasapainotilan löytäminen selvitetään. Viimeisenä keskitytään varsinaiseen tekoälyyn: miten rakennetaan peliteoreettinen hyötyfunktio, jolla tietokonepelin vihollisagentit saadaan toimimaan mahdollisimman älykkäästi erilaisissa pelitilanteissa. Myös peliteorian ongelmia – sekä loogisia että teknisiä – pohditaan. Niistä suurin on Nashin tasapainotilan löytämisen aikavaativuus.

Aiheluokat (Computing Reviews 1998): I.2.11, J.4

Avainsanat — Nyckelord — Keywords
moniagenttijärjestelmät, peliteoria, tietokonepelit

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Tietokonepelit ja tekoäly ovat olleet yhteenkiedottuja jo syntymästään saakka: ensimmäiset varsinaiset tekoälyt pelasivat Russellin ja Norvigin mukaan pelejä ihmisiä vastaan, ja tietokonepeleihin on yritetty luoda immersion tuntua kehittämällä pelaajan tekoälytovereiden ja -vastustajien toimintaa yhä uskottavammaksi [RuN03]. Uskottavuus tarkoittaa tässä yhteydessä älykkään näköistä toimintaa — uskottava tekoälyvastustaja ei juutu kiinni oviin tai esteisiin maastossa tai tee tuhoon tuomittua hyökkäystä ylivoimaisessa asemassa olevaa pelaajaa vastaan. Pelien tekoälyt ovat kehittyneet vuosien varrella huimasti, mutta Rabinin mukaan pelien tekoäly on vieläkin useimmiten *skriptattu*: tietokoneen ohjaamien pelaajien, joita usein kutsutaan ei-pelaajahahmoiksi, toiminta koostuu etukäteen mietityistä ja uskottavalta vaikuttavista komentosarjoista [Rab03]. Skriptejä on helppo ja nopea tehdä, ja pelin pelaajan kannalta ei ole väliä, onko tietokoneen toiminta oikeasti vai näennäisesti älykästä. Skriptauksessa on kuitenkin ongelmia: Nareyekin mukaan pelkän komentosarjan varassa toimivat ei-pelaajahahmot ovat staattisia eli ne osaavat reagoida dynaamisiin tapahtumiin vain rajoitetusti [Nar00]. Erityisesti ryhmässä toimimisen monimutkaiset kuviot ja dynamiikka jäävät myös usein pelinkehittäjiltä puolitiehen. Tässä tutkielmassa on tarkoituksena selvittää, miten hyvin peliteoreettiset menetelmät sopivat tietokoneen ohjaaman vihollisryhmän tekoälyn luomiseen toimintapeleissä.

Duttan mukaan peliteoria on tieteenhaara, joka tutkii usean toimijan eli agentin vuorovaikutusta [Dut99]. Jokaisen agentin täytyy valita yksi *toiminto* (action) rajatusta joukosta mahdollisia toimintoja. Toiminnon valitsemista sanotaan *strategiaksi* (strategy), jonka agentti pelaa. Yhtä kierrosta, jossa agentit koordinoivat strategiansa valinnan, sanotaan *peliksi* (game). Peliä pelaavien agenttien oletetaan olevan *rationaalisia*, joka tarkoittaa, että agentti pyrkii maksimoimaan pelistä saamansa hyödyn. Peliteorian ajatuksena on siis jokaisen agentin kohdalla löytää sellainen strategia, jolla agentti saa maksimoitua pelistä itselleen saamansa hyödyn. Sopivan strategian löytäminen on monimutkaista, sillä agentin täytyy ottaa huomioon myös kaikkien muiden agenttien vastaava päättelykulku. Itse peli voidaan esittää joukkona mahdollisia *pelitiloja* (state), joissa jokainen pelissä mukana oleva agentti pelaa yhden toiminnon mahdollisten toimintojensa joukosta. Erilaisia pelitiloja on siis k^n kappaletta, jos pelissä on n agenttia, joista jokaisella on k mahdollista toimintoa. Peliä voi havainnollistaa n-uloitteisella matriisilla, jossa matriisin sivun pituus on yhden agentin valittavissa olevien toimintojen määrä. Matriisin solussa (eli peliti-

lassa) on hyötyvektori, joka määrää kullekin agentille tietyn hyödyn tästä toimintojen yhdistelmästä. Kun agentit ovat päättäneet strategiansa, on pelitila-avaruus muuttunut agenttien strategiavektoriksi.

Peliteoriassa on usein tavoitteena, että agentit päätyvät pelissä tilaan, jossa yhdenkään niistä ei kannata yksipuolisesti vaihtaa valitsemaansa toimintoa miksikään muuksi. Tällaista tilaa, jossa agentit ovat päätyneet kaikkia tyydyttävään ratkaisuun, kutsutaan *Nashin tasapainotilaksi* (Nash equilibrium). Merkille pantavaa on, että tämä tila ei kuitenkaan ole välttämättä agenttien yhteisön kannalta paras tila: jossain muussa tilassa agenttien hyötyjen summa voi olla suurempi. Stirling ja muut kirjoittavat, että myös muita mahdollisia lopputiloja on olemassa. Yhteistä kaikille on, että ne muodostavat kuvauksen agenttien tavoitteista johonkin ryhmätoiminta-päätökseen [SGP02].

Usein nähty esimerkki peliteoreettisesta pelistä ja sen ratkaisusta on niin sanottu Vangin ongelma (Prisoner's dilemma). Siinä kahta vankia kuulustellaan eri huoneissa. Molemmat vangit voivat joko tunnustaa tai jättää tunnustamatta. Jos kumpikaan vanki ei tunnusta, molemmat joudutaan vapauttamaan todisteiden puutteessa. Jos vain toinen vanki tunnustaa, saa tunnustaja palkinnon ja tunnustamatta jättänyt vanki saa pitkän vankeustuomion. Jos molemmat vangit tunnustavat, tuomitaan molemmat lyhyeksi ajaksi vankilaan.

Vangin ongelmaa kuvaa taulukko 1.

		Vanki 2	
		Tunnustaa	Ei tunnusta
Vanki 1	Tunnustaa	(-2,-2)	(2,-5)
	Ei Tunnusta	(-5,2)	(0,0)

Taulukko 1: Vangin ongelma

Taulukkoa luetaan niin, että jokainen taulukon solu kuvaa yhtä pelitilaa. Pelitiloissa suluissa olevat lukuparit ovat agenttien hyötyfunktioiden arvoja. Lukuparin vasemmalla oleva luku on ensimmäisen ja oikealla oleva toisen vangin pelistä saama hyöty. Taulukosta nähdään helposti, että paras tilanne saavutetaan, jos kumpikaan vanki ei tunnusta: vankien saama yhteishyöty on $0 + 0 = 0$. Rationaalinen vanki haluaa kuitenkin saavuttaa mahdollisimman suuren hyödyn, ja tietää myös toisen vangin olevan rationaalinen. Täten ensimmäinen vanki tietää, että jos hän ei tunnusta, toinen vanki valitsee lähes varmasti tunnustamisen, jolloin ensimmäinen vanki joutuu

pitkäksi aikaa vankilaan. Täten paras vaihtoehto on tunnustaa, jolloin vankila-aika lyhenee. Toinen vanki ajattelee samalla tavoin. Molemmat vangit päätyvät tunnustamaan, jolloin vankien saama yhteishyöty on $-2 + -2 = -4$, mikä on samalla huonoin mahdollinen. Duttan mukaan tilannetta voi arvioida myös siten, että huolimatta toisen agentin valitsemasta toiminnosta, vangin saama hyöty kasvaa, jos hän tunnustaa [Dut99]. Molempien vankien tunnustaminen on pelin tasapainotila, koska kumpikin pelaaja päätyy toisen toimia arvioidessaan lopulta siihen. Tärkeää on, että tasapainotilan saavuttamiseen ei vaikuta, pääsevätkö agentit kommunikoimaan keskenään ennen valintaansa: molempien agenttien vankien kannattaa kuitenkin lopulta pettää toinen vanki ja tunnustaa. Tasapainotilaan ei myöskään vaikuta, tekevätkö agentit strategiavalintansa peräkkäin vai samanaikaisesti.

Koska peliteoria on tarkoitettu moniagenttijärjestelmän päätöksentekoon, se on luonnollinen ehdokas myös tietokonepelien ryhmätekoälyn kehittämiseen. Parsonsin ja Woolridgen mukaan ryhmätekoälyn tutkimuksen alkuaikoina oletettiin laajalti, että agentit ovat hyväntahtoisia: jos agenteilla on yhteinen päämäärä, ne haluavat auttaa toisiaan [PaW02]. Tällaisia järjestelmiä kutsutaan *hajautetuksi ongelmanratkaisuksi* (distributed problem solving). Monissa tietokonepeleissä, joissa ryhmätekoälyä on pohdittu, näin menetellään edelleen: vihollisagenttiryhmän tavoitteena on ratkaista ongelma, joka on pelaajahahmon voittaminen. Kuitenkin Nareyekin mukaan tietokonepelin tekoälykkäiden vihollisten tarkoitus on luoda pelaajalle vain tunne, että hän on tekemisissä älykkäiden vastustajien kanssa. Peliteoria tarjoaa välineen tähän, koska siinä rationaaliset agentit pyrkivät maksimoimaan omaa hyötyään. On helppo kuvitella, että jokaisen vihollisagentin tärkein tavoite ei aina ole pelaajahahmon voittaminen, vaan myös esimerkiksi omalla hengissäselviämisellä on merkityksensä.

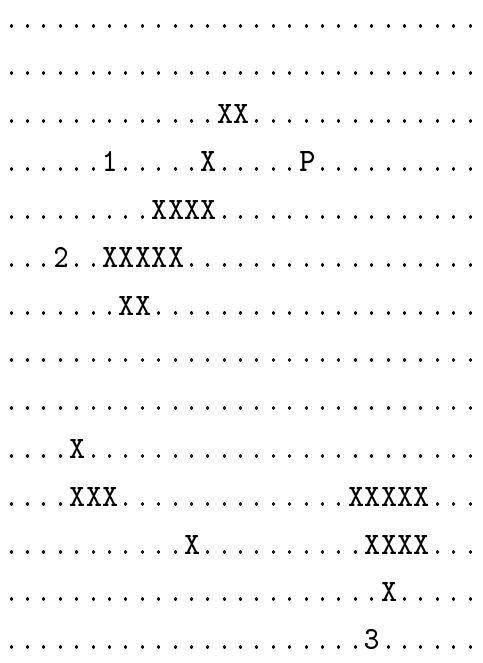
Yksi peliteorian mahdollinen käyttökohde on toimintapelien tilanne, jossa monta hyökkäävää vihollisagenttia pyrkii pääsemään maaston suojassa pelaajahahmon lähelle, kun pelaajahahmolla on käytettävissään pitkän matkan aseistusta. Tätä peliä, jota tässä tutkielmassa tullaan tarkastelemaan lähemmin, kutsutaan *Hyökkäyspeliksi*. Vihollisagenttien ei kannata lähestyessään pelaajaa liikkua lainkaan avoimessa maastossa. Piilon suojasta hyökkääminenkin kannattaa koordinoida samanaikaiseksi, jotta hyökkääjät altistuisivat mahdollisimman vähän pelaajahahmon tulitukselle. Ennen jokaista siirtoaan vihollisagentit pelaavat yhteisen pelin, jossa jokaisen agentin strategia päätetään. Tätä tilannetta tarkastellaan niin käytännön kuin teoriankin tasolla. Kuvassa 1 on Hyökkäyspelin tilanne tietokonepelissä Shadowgrounds [Fro06]. Kuva 2 esittää vastaavaa tilannetta, mutta muutettuna kaksiulotteiseen projektioon. Tässä kuvassa ja muissa samaa peliä esittävissä kuvissa **P**

Kuva 1: Hyökkäyspelin tilanne Shadowgrounds-tietokonepelissä.

kuvaa pelaajaa, numerot eri vihollisagentteja, **X** kentällä olevaa estettä ja . tyhjää maastoa, jonka läpi agentit pystyvät liikkumaan. Pelaajahahmo ei pysty hyökkäämään vihollisagenttia vastaan tai havaitsemaan sitä, jos vihollisagentin ja pelaajan välissä on este.

Hyökkäyspelissä tehdään joitain oletuksia, joilla ongelmakenttää yksinkertaistetaan. Yksi tällainen oletus on, että vihollisagenteilla on täydellinen tieto pelikentästä ja erityisesti muiden agenttien ja pelaajahahmon sijainnista. Luvussa 3.3 käsitellään kuitenkin tilannetta, jossa pelaajahahmon tai muiden vihollisagenttien sijainnista ei ole varmaa tietoa. Toinen yksinkertaistus on, että joihinkin kysymyksiin, jotka tietokonepelissä tulee ottaa huomioon, ei Hyökkäyspelissä millään tavoin oteta kantaa. Esimerkiksi vihollisagenttiryhmän luomista, vihollisagenttien liikkumista ilman pelaajahahmon läsnäoloa tai agenttien välistä kommunikaatiota virtuaalimaailman sisällä ei huomioida, vaan ne oletetaan tapahtuvaksi jollain muulla pelin sisäisellä mekanismilla. Tällä rajauksella pyritään keskittymään Hyökkäyspelin analysoinnissa juuri peliteoreettisesti mielenkiintoisiin aiheisiin.

Saalistaja-saalis-ongelma (predator-prey pursuit domain) on perinteinen moniagentti-



Kuva 2: Esimerkki Hyökkäyspelin tilanteesta kaksiulotteisessa projektiossa.

tekoälyn tutkimusongelma, jota on käytetty lukuisten erilaisten tekoälymenetelmien testaamiseen. Sen ajatus on yksinkertainen: yksi saalisagentti pakenee tavallisesti neljää saalistaja-agenttia ruudukossa. Saalistajat yrittävät ottaa saaliin kiinni ympäröimällä sen joka puolelta, ja saalis yrittää paeta joko suunnitelmallisesti tai satunnaisesti liikkumalla. Jokainen agentti saa liikkua kierroksessa yhden ruudun haluamaansa suuntaan, mutta kaksi agenttia ei saa siirtyä samaan ruutuun. Korfin mukaan saalistajien (ja eri tekoälymenetelmien) tehokkuutta voidaan vertailla sen mukaan, miten monta pelikierrosta saalistajilta kestää saaliin ympäröintiin [Kor92]. Levy ja Rosenschein esittelivät saalistaja-saalis-ongelman peliteoreettisen ratkaisun [LeR92], jossa vihollisagentit pelaavat yksinkertaista peliteoreettista peliä ratkaistakseen, kuka lähestyy saalista mistäkin suunnasta.

Hyökkäyspelin yhtäläisyydet saalistaja-saalis-ongelman kanssa ovat ilmeisiä. Pelaajahahmo muistuttaa paljon saalisagenttia, jota kohti hyökkääjät pyrkivät kulkemaan, ja joka yrittää itse olla joutumatta hyökkääjien saaliiksi. Eroavaisuuksiakin on: saalistaja-agenteilla ei ole mitään pelättävää, mutta tietokonepelissä hyökkääjät voivat joutua torjuntatulen uhreiksi. Hyökkääjien ei siksi kannata lähestyä pelaajahahmoa nopeinta mahdollista reittiä, vaan pyrkiä käyttämään kaikkea maaston tarjoamaa suojaa hyödyksi. Samoin hyökkääjille on edullista, että monta hyökkääjää

liikkuu samanaikaisesti — mieluiten eri puolilta lähestyen — pelaajahahmon luokse, jotta hyökkääjien torjuminen olisi mahdollisimman vaikeaa. Tämä viimeinen ominaisuus muistuttaa kuitenkin jossain mielessä saalistaja-saalis-ongelmaa, jossa saalistajat pyrkivät kiertämään saaliin kaikille puolille.

Agentit muodostavat Hyökkäyspelissä peliteoreettisen pelin, jossa jokainen agentti pyrkii löytämään strategian, joka maksimoisi sen oman hyödyn. Koska agenttien eri tilanteista saama hyöty riippuu myös muiden agenttien valitsemista toiminnoista, sopivan tasapainotilan löytäminen ei ole aivan yksinkertaista. Jokaisella agentilla on hyötyfunktio, joka saa parametrinaan agenttien toimintakombinaation ja palauttaa arvonaan agentin toimintakombinaatiosta saaman hyödyn. Tämä hyötyfunktio tehdään huomioimaan sekä tietokonepelin pelattavuuden (”agentit toimivat uskottavasti”) että moniagenttijärjestelmän tehokkuuden (”peli on vaikea”).

Tässä tutkielmassa tarkoituksena on selvittää peliteorian käyttämistä Hyökkäyspelin vihollisagenttien ryhmätekoälyn luomiseen. Luvussa 2 tutustutaan peliteorian perusteisiin ja tietokonepelien tekoälyn ongelmakenttään. Luvussa 3 pohditaan peliteorian ja Hyökkäyspelin yhteensovittamista. Luvussa 4 käsitellään Nashin tasapainotilan löytämistä ja luvussa 5 rakennetaan Hyökkäyspeliin soveltuva hyötyfunktio. Tuloksena on skriptattuun tekoälyyn verrattuna parempi ja yleistettävä tekoälymalli tietokonepelien käyttöön.

2 Pelejä ja peliteoriaa

Tässä luvussa käsitellään peliteorian peruskäsitteitä ja esitellään muutamia sen ominaisuuksia ja ongelmia. Sen jälkeen pohditaan tietokonepelien tekoälyn perimmäistä kysymystä: millainen on hyvä tietokonepelin tekoäly? Peliteorian mahdollisuuksia kysymyksen ratkaisuun selvitetään. Lopuksi tarkastellaan joitain tietokonepeleissä tällä hetkellä käytettyjä moniagenttitekoälyn menetelmiä.

2.1 Peliteorian perusteet

Peliteoria on valtavan laaja kenttä, jota on tutkittu nykymuodossaan aktiivisesti 1940-luvulta aina tähän päivään saakka. Tässä luvussa käsitellään sen perusteita siinä laajuudessa, kuin mitä peliteorian soveltaminen Hyökkäyspelin tekoälyssä vaatii.

Esitellään notaatio peliteoreettisten käsitteiden matemaattiseen ilmaisemiseen. Merkitään pelaajan i valitsemaa strategiaa merkinnällä s_i, ja kaikkien muiden pelaajien valitsemaa strategiayhdistelmää merkinnällä s_{-i}. Nyt pelaajan i saama hyöty v_i strategialla s_i saadaan hyötyfunktiosta π:

$$v_i = \pi_i(s_i, s_{-i})$$

Peliteorian mukaan tehty toimintapäätös ei siis välttämättä ole lopulliselta hyödyltään paras mahdollinen, mutta se on kaikki asianhaarat huomioon ottaen jokaiselta agentilta strategisesti oikea teko. Joskus kuitenkin pelaajan strategioista yksi on niin hyvä, että se tuottaa aina parhaan hyödyn riippumatta muiden pelaajien tekemistä valinnoista. Dutta kutsuu tällaista strategiaa *vahvasti hallitsevaksi strategiaksi* (strongly dominant strategy) [Dut99]. Täsmällisemmin voi sanoa, että strategia s pelaajalle i hallitsee vahvasti strategiaa s', jos hyöty strategian s pelaamisesta on aina suurempi kuin strategian s' pelaamisesta kaikilla muiden pelaajien strategioiden yhdistelmillä:

$$\pi_i(s_i, s_{-i}) > \pi_i(s'_i, s_{-i}) \quad \text{kaikille } s_i \text{ ja kaikille } s_{-i}$$

Strategia s hallitsee heikosti strategiaa s', jos sillä saatu hyöty on suurempi ainakin yhdellä muiden agenttien strategiayhdistelmällä ja vähintään sama kaikilla muilla. Luvussa 1 esitellyssä Vangin ongelmassa tunnustaminen hallitsi vahvasti tunnustamatta jättämistä molemmilla agenteilla.

Samoin jos jokin strategia s on aina huonompi kuin mikä tahansa strategia s', sitä sanotaan *hallituksi strategiaksi* (dominated strategy). Agentin ei koskaan pidä pelata hallittua strategiaa — jokaiseen tilanteeseen on olemassa parempi strategia. Hallittujen strategioiden löytäminen on olennainen osa tasapainotilan etsintäavaruuden pienentämistä, johon palataan luvussa 4.2.

Vangin ongelma on sikäli yksinkertainen, että Nashin tasapainotila on helposti löydettävissä ja yksiselitteinen: jokaiselle agentille on selvää, että yksi strategia hallitsee vahvasti muita strategioita, joten vain se kannattaa pelata. Useissa peleissä kuitenkin on useampi tasapainotila, jolloin agenteilla ei ole vain yhtä vahvasti hallitsevaa strategiaa. Tällöin tasapainotilan tunnistaa siitä, että yksikään agentti ei voi yksin poiketa siitä heikentämättä omaa etuaan. Samoin on mahdollista, että agentilla ei ole yhtä strategiaa, johon valinta osuisi. Esimerkki tällaisesta pelistä on Rasmusenin esittelemä hyvinvointipeli (welfare game), jonka pelimatriisi on taulukossa 2 [Ras01]. Siinä on kaksi pelaajaa, Hallitus ja Pummi. Hallitus haluaa antaa Pummille avustusta, jos hän etsii työtä, mutta Pummi haluaa laiskotella ja saada avustusta.

		Hallitus	
		Avustus	Ei avustusta
Pummi	Etsi työtä	(2,3)	(1,-1)
	Laiskottele	(3,-1)	(0,0)

Taulukko 2: Hyvinvointipeli

Taulukosta nähdään, että tasapainotilan silmämääräinen etsintä kiertää kehässä: Pummin kannattaa laiskotella, jos Hallitus antaa avustusta, mutta Hallituksen ei kannata antaa avustusta, jos Pummi laiskottelee. Jos taas Hallitus ei anna avustusta, Pummi etsii työtä, mutta Hallituksen kannattaa silloin antaa avustusta. Pelistä ei näytä löytyvän sellaista tasapainotilaa, johon kaikki pelaajat tyytyisivät.

John Nash todisti kuuluisassa paperissaan, että jokaisessa äärellisessä pelissä on oltava ainakin yksi tasapainotila [Nas50]. *Puhtaassa strategiassa* (pure strategy) on vain yksi toiminto, jonka agentti pelaa varmasti. Jos pelaaja-agentille ei löydy puhdasta strategiaa, jolla päästäisiin Nashin tasapainotilaan, täytyy sille löytyä ainakin yksi *sekastrategia* (mixed strategy) jolla tasapainotila saavutetaan. Sekastrategia on todennäköisyysvektori tietyn toimintojoukon yli. Ajatuksena on, että agentit valitsevat strategiansa sen mukaan, mikä on hyödyn odotusarvo. Dutta kirjoittaa, et-

tä Von Neumannin ja Morgensternin oletusarvoteoreeman mukaan hyötyfunktiosta saatava hyöty voidaan kertoa odotusarvona, jossa jokaisen eri mahdollisuuden todennäköisyys on kerrottuna mahdollisuudesta saatavana hyötynä, jos tietyt rajoitukset ovat voimassa [Dut99].

Todennäköisyysvektorin luominen on suhteellisen intuitiivista. Nashin tasapainotila saavutetaan silloin, kun yksikään pelaaja-agenteista ei halua vaihtaa toimintoaan. Yleisessä tapauksessa agentti i saavuttaa tällaisen tilan silloin, kun odotettu hyöty tietyllä toimintojen joukolla on sama: kaikki muut agentit voivat valita toimintonsa tietäen, että agentti i pelaa tietyn todennäköisyysjakauman mukaisesti. Toisin sanoen, tasapainotila saavutetaan silloin, kun jokaisen agentin odotettu hyöty on sama niillä strategioilla, joiden yli todennäköisyysvektori muodostetaan. Tällöin tasapainotilan edellytys — toiminnon vaihtamisen kannattamattomuus — täyttyy kaikkien agenttien osalta.

Esimerkki valaisee asiaa. Oletetaan, että Pummi etsii työtä todennäköisyydellä $q \in [0 \ldots 1]$ ja laiskottelee todennäköisyydellä $1 - q$, jolloin todennäköisyyden ehto täyttyy, sillä todennäköisyyksien summa on 1. Samoin oletetaan, että Hallitus avustaa todennäköisyydellä p ja ei avusta todennäköisyydellä $1 - p$. Taulukosta 2 nähdään, että Pummin odotusarvo työnteosta on:

$$E(\text{Työ}) = 2p + 1(1 - p) = p + 1$$

Pummin odotusarvo laiskottelusta on:

$$E(\text{Laiskottelu}) = 3p + 0(1 - p) = 3p$$

Jotta Pummi saisi saman odotetun hyödyn riippumatta omista toimistaan, täytyy Hallituksen avustustodennäköisyyden olla:

$$\begin{aligned} E(\text{Työ}) &= E(\text{Laiskottelu}) \\ p + 1 &= 3p \\ p &= \frac{1}{2} \end{aligned}$$

Toisin sanoen, jos Hallitus avustaa todennäköisyydellä $\frac{1}{2}$ ja jättää avustamatta todennäköisyydellä $1 - \frac{1}{2} = \frac{1}{2}$, saa Pummi saman hyödyn odotusarvon kaikista toimistaan. Pummin saama hyödyn odotusarvo on $1 + \frac{1}{2} = 1\frac{1}{2}$. Nyt jos Pummi

sekoittaa oman strategiansa niin, että Hallitus saa saman odotusarvon kaikista strategioistaan, on tilanne symmetrinen ja Hallitus voi yhtä hyvin pelata juuri oman sekastrategiansa. Tässä tapauksessa hallitus saisi seuraavan odotusarvon avustuksesta:

$$E(\text{Avustus}) = 3q + (-1)(1-q) = 4q - 1$$

Hallituksen odotusarvo avustamatta jättämisestä on:

$$E(\text{Ei avustusta}) = -1q + 0(1-q) = -q$$

Jotta Hallitus saisi saman hyödyn odotusarvon molemmista toimintovaihtoehdoistaan, Pummin on etsittävä työtä seuraavalla todennäköisyydellä:

$$\begin{aligned} E(\text{Avustus}) &= E(\text{Ei avustusta}) \\ 4q - 1 &= -q \\ q &= \frac{1}{5} \end{aligned}$$

Luonnollisesti Pummin laiskottelun todennäköisyys on tällöin $1 - \frac{1}{5} = \frac{4}{5}$.

Hallituksen saama hyödyn odotusarvo on täten $-\frac{1}{5}$. Koska molempien pelaajien kaikilla toiminnoilla pelaamisen todennäköisyys on suurempi kuin 0, Rasmusenin mukaan molemmilla pelaajilla on *täydellinen sekastrategia* (completely mixed strategy) [Ras01].

Sekastrategian tulkinta ei ole yhtä intuitiivista kuin puhtaiden strategioiden, koska oikeassa maailmassa toimijat ani harvoin valitsevat arvalla toimintoaan eri tilanteissa. Rasmusen kirjoittaa, että yksi tapa ymmärtää satunnaista toiminnon valitsemista on ajatella monen Pummin peliä, jossa Hallitus kohtaa pummipopulaation. Jokainen yksittäinen Pummi pelaa vain puhtaan strategian. Jos Pummin kannattaa valita Laiskottelu todennäköisyydellä $\frac{1}{5}$, voidaankin ajatella, että viidennes pummipopulaatiosta päättää jostain syystä pelata puhtaan strategian 'laiskottelu', ja muut pelaavat puhtaan strategian 'etsi työtä'. Tällöin Hallituksen kannattaa suhtautua pummipopulaatiosta otettuun yhteen edustajaan siten, että edustajaksi valittu Pummi pelaa sekastrategian.

Puhdas strategia on itse asiassa sekastrategian erikoistapaus: yhden toiminnon todennäköisyys on 1 ja muiden 0. Sekastrategiassa voi olla mukana mikä tahansa määrä

agentin toimintoja. Niiden toimintojen joukkoa, joiden pelaamistodennäköisyys on suurempi kuin 0, sanotaan agentin *tueksi* (support). Agentin kannattaa valita sekastrategia vain silloin, jos kaikki sekastrategian tuessa olevat puhtaat strategiat ovat itsessään kannattavia valintoja: tukeen ei kannata lisätä uutta puhdasta strategiaa, jonka lisääminen laskee sekastrategian pelaamisesta saatavaa hyödyn odotusarvoa. Sekastrategioiden löytämistä käsitellään luvussa 4.3.

Nashin tasapainotilaa voi siis ajatella pelin paikallisena optimina. Alkujaan tasapainotiloja kutsuttiin myös pelin satulakohdiksi, koska niistä agentit voivat vain valua alaspäin. Miten tällainen tasapainotila sitten löydetään, ja miten voidaan varmistaa valitun tasapainotilan olevan tarpeeksi hyvä?

Peliteoreettisten pelien ratkaiseminen ei ole helppoa: Papadimitrioun mukaan Nashin tasapainotilan etsimisen pahimman tapauksen aikavaatimusta ei edes tiedetä [Pap01]. Papadimitriou kutsuu Nashin tasapainotilan löytämistä kaikkein merkittävimmäksi laskennalliseksi ongelmaksi, jonka kompleksisuus on täysin avoinna. Kuitenkin on luultavaa, että kahden pelaajan pelin aikavaativuus on jossain P- ja NP-luokkien välissä [Pap01]. McKelvey ja McLennan sanovat, että n:n pelaajan pelit ovat aikavaatimuksensa puolesta merkittävästi vaikeampia ratkaista kuin kahden pelaajan pelit [McM96]. He myös löytävät kaksi suurta kynnystä aikavaatimuksen suhteen: onko pelaajia kaksi vai useampia ja halutaanko löytää yksi vai kaikki tasapainotilat. Tasapainotilojen löytämisen aika- ja tilavaativuutta käsitellään lisää kappaleessa 4.1.

Yhdessä pelissä voi siis olla useampia Nashin tasapainotiloja. Jos kaikki tasapainotilat halutaan löytää, on edessä oleva urakka paljon vaikeampi kuin pelkkä ensimmäisen tasapainotilan löytäminen. McLennan kirjoittaa, että eri tasapainotilojen määrä missä tahansa vähänkään monimutkaisemmassa pelissä on valtava [McL99].

Monen pelaajan pelit ovat melkein suoraan yleistettävissä kahden pelaajan peleistä yhtä poikkeusta lukuun ottamatta: monen pelaajan peleissä agentit voivat tehdä koalitioita. Kuten todettu, Nashin tasapainotila on paikallinen optimi siinä mielessä, että yksittäinen agentti ei voi muuttaa toimintaansa niin, ettei sen oma hyöty vähenisi. On kuitenkin mahdollista, että useampi kuin yksi agentti muodostaa koalition, joka yhteispäätöksellä vaihtaa tasapainotilaa. Tällöin toisessa tasapainotilassa agenttien saamat hyödyt voivat olla aivan erilaiset. Aumann kirjoittaa, että tasapainotilaa, jossa on mahdollista muodostaa koalitio, joka hyötyy tasapainotilan muuttamisesta, kutsutaan *heikoksi Nashin tasapainotilaksi* [Aum59]. Tasapainotilaa, jota mikään pelaajien muodostama koalitio ei pysty muuttamaan kaikkien koalition jäsenten mielestä paremmaksi, sanotaan vastaavasti *vahvaksi Nashin tasapainotilaksi*.

Pelejä, joissa agentit neuvottelevat ja pyrkivät muodostamaan koalitioita, kutsutaan *yhteistyöpeleiksi* (cooperative games). Yhteistyöpelit on kuitenkin rajattu tämän tutkielman ulkopuolelle.

2.2 Tasapainotilojen koordinoiminen

Peliteoria vaatii usein myös agenttien välistä koordinointia. Ongelman ydin on, että jos jokainen agentti laskee itse oman pelimatriisinsa, sen täytyy tietää muiden agenttien saamat hyödyt eri pelitilanteissa. Jotta kaikilla agenteilla olisi sama pelimatriisi, niiden täytyy kommunikoida omat hyötynsä eri tilanteissa muille. Tähän liittyy luonnollisesti epärehellisen käytöksen mahdollisuus: Sandholmin mukaan agentti voi joissain tapauksissa yrittää huijata muita pelaajia valehtelemalla omat hyötynsä [San99]. Tietokonepelin tapauksessa tällaista ongelmaa ei välttämättä ole, sillä agentit voivat olla samanlaisia tai ainakin niiden luokka on voi olla tunnettu. Tällöin ne tietävät täydellisesti toistensa rakenteen ja voivat periaatteessa siten laskea myös naapureidensa hyötyfunktiot.

Jos agenteille ei kerrota, minkä Nashin tasapainotilan muut agentit aikovat pelata, miten ne osaavat valita oikean? Tietokonepelin tapauksessa tämä on helppoa: jo laskennallisista syistä joudutaan luopumaan luonnollisesta tilanteesta, jossa agenttien täytyy itse pohtia pelimatriisia. Ohjelmointiteknisesti tasapainotilan löytyminen kannattaa toteuttaa niin, että pelimoottori laskee sopivan tasapainotilan ja agenteille vain kerrotaan, mitkä strategiat ne saavat pelata. Jos näin ei olisi, täytyisi agenttien tasapainotilan valintaa ohjata.

Ongelma yleistyy suoraan agenttiryhmän koordinointiongelmaksi. Jos jokainen agentti valitsisi strategian eri tasapainotilasta, agenttiryhmän toiminta ei olisi enää rationaalista ja järkevän näköistä. Boutilier kirjoittaa, että koordinointiongelman ratkaisemiseen on kolme klassista tapaa [Bou99]. Ensimmäinen on sellaisten sosiaalisten sääntöjen tekeminen, jotka ohjaavat agentit valitsemaan oikean tasapainotilan. Esimerkki tästä voisi olla sääntö, joka sanoo: "Jos sillalle on samaan aikaan pyrkimässä monta agenttia, ensimmäisenä saavat mennä ne, joiden id-numerot ovat pienimmät." Tällöin agentit osaisivat valita useasta tasapainotilasta juuri sen, jossa ensimmäisellä sillalle astuvalla agentilla olisi pienin id-numero.

Toinen Boutilierin mainitsema koordinaatiotapa on kommunikointi. Agentit voivat keskenään päättää, mikä tasapainotila valitaan, ja kommunikoida sen kaikille. Ongelma kommunikoinnissa on, että senkin yhteyteen pitää liittää sosiaalisia sääntöjä.

Näiden sääntöjen avulla tiedetään, miten oikea tasapainotila päätetään. Yksi esimerkki tällaisesta sosiaalisesta säännöstä on: ”Jokaisella kierroksella järjestyksessä seuraava agentti päättää, mikä tasapainotila valitaan”. Kommunikointi tuo mukanaan myös mielenkiintoisia tietokonepelillisiä mahdollisuuksia: vihollisagentti voidaan esimerkiksi esittää huutamassa käskyjä muille agenteille, ja ne agentit, jotka eivät ole kuulomatkan säteellä, eivät osaa valita oikeaa tasapainotilaa.

Kolmas koordinaatiotapa, jonka Boutilier esittelee, on oppiminen toistetuissa peleissä. Tämä ei suoraan liity tasapainotilan muuttumiseen toistetuissa peleissä, vaan tarkoittaa, että agentit seuraavat toistensa tietyssä tilanteessa valitsemia toimintoja. Näiden valintojen pohjalta ne pyrkivät jollain soveltuvalla algoritmilla oppimaan, mitä kannattaa pelata, jotta tasapainotilan vaatimus täyttyisi.

Keskitetty tasapainotilan valitseminen on periaatteessa helppoa, sillä pelimoottori voi etsiä tarvittaessa kaikki tasapainotilat, ja valita niistä ”parhaan” jollain kriteerillä. Sandholm määrittelee, että agenttien yhteenlaskettua hyötyä kutsutaan *sosiaaliseksi hyvinvoinniksi* (social welfare) [San99]. Tasapainotila, joka antaa suurimman sosiaalisen hyvinvoinnin, on luultavasti hyvä ehdokas parhaaksi valinnaksi monen tasapainotilan joukosta. Sandholm kuitenkin toteaa, että jos sosiaalista hyvinvointia mitataan vain laskemalla agenttien tasapainotilasta saamat hyödyt yhteen, voi tulos olla mielivaltainen, koska agenttien hyötyfunktiot eivät välttämättä ole keskenään vertailtavissa. Vaikka Hyökkäyspelissä agentit ovat identtisiä, monimutkaisemmissa tietokonepeleissä ei aina voi laskea sosiaalisen hyvinvoinnin käytön varaan.

Myös McLennan ja McKelvey toteavat, että kaikki tasapainotilat eivät ole yhtä hyviä: koska ne ovat vain paikallisia optimeja, jotkin tasapainotiloista voivat olla *hallittuja* [McM96]. Tasapainotilojen hallitseminen on analoginen strategioiden hallitsemisen kanssa: jos on olemassa tasapainotila E, josta kaikki agentit saavat suuremman hyödyn kuin tasapainotilasta E', hallitsee tasapainotila E vahvasti tasapainotilaa E'. Tasapainotilaa E kutsutaan *Pareto-tehokkaaksi*, jos ei ole olemassa tasapainotilaa E', jossa ainakin yksi agentti saisi suuremman hyödyn ja yksikään agentti ei saisi pienempää hyötyä kuin tasapainotilassa E. Pareto-tehokas tasapainotila ei siis voi olla hallittu [McM96].

Sandholmin mukaan Pareto-tehokkuus voi olla sosiaalista hyvinvointia parempi kriteeri ratkaisujen paremmuutta vertailtaessa. Kuten sosiaalinen hyvinvointi, myös Pareto-tehokkuus mittaa globaalia hyötyä, mutta tekee sen ilman agenttien hyötyjen vertailua agenttien välillä. Agentin saamaa hyötyä verrataan siis vain agentin muista ratkaisuista saamaan hyötyyn. Sandholm kirjoittaa, että sosiaalista hyvinvointia

maksimoivat ratkaisut ovat Pareto-tehokkaiden ratkaisujen osajoukko [San99].

Ongelmaksi kuitenkin muodostuu, että tietokonepelien tapauksessa aika on kaikki kaikessa. Useinkaan ei ole varaa etsiä juuri halutunlaista tasapainotilaa, vaan ensimmäiseen on tyydyttävä. Tämä on erityisen merkittävää siksi, että tietokonepeleissä pelattavat peliteoreettiset pelit ovat tyypillisesti melko suuria sekä agenttien määrän että niiden valittavissa olevien strategioiden määrän suhteen. Silti Herings ja Peeters sanovat, että myös monen pelaajan peleistä on mahdollista löytää rakenteita, joita voi käyttää hyväkseen keskivertoa parempaa tasapainotilaa etsittäessä [HeP00].

2.3 Tietokonepelien tekoälyn problematiikkaa

Tietokonepelien tekoälyä on kehitetty vuosien ajan. Russellin ja Norvigin mukaan aluksi tavoitteena oli mahdollisimman suuri tehokkuus [RuN03]. Tekoälyn kyvykkyyttä mitattiin sen mukaan, miten hyvin se pärjäsi ihmisvastustajia vastaan. Nykyään erityisesti toimintatietokonepeleissä keskitytään pelattavuuteen, sillä tietokonepelaajalla on aina esimerkiksi reaktio-, informaatio- ja tarkkuusedut puolellaan, ellei niitä erityisesti pelin puolesta rajoiteta. Pelattavuudella tarkoitetaan sitä vaikeasti määriteltävää ominaisuutta, joka saa ihmispelaajan pelaamaan tietokonepeliä yhä uudestaan ja uudestaan. Osa pelattavuudesta on pelin sujuvaa toimintaa, selkeää käyttöliittymää ja sopivalla tasolla olevaa vaikeusastetta, mutta yhä suurempi osa siitä on *immersiota*. Immersio on lähellä teatteri- ja elokuvataiteen käsitettä ”epäuskon tukahduttaminen” (”suspension of disbelief”), joka merkitsee esityksen kykyä temmata katsoja mukaan esityksen omaan sisäiseen logiikkaan ja maailmaan. Jos esityksen tai tietokonepelin immersio on korkea, pelaaja siirtää päähenkilön tai pelaajahahmon kokemia tunteita omiksi tunteikseen, ja esimerkiksi ahdistuu elokuvan päähenkilön ollessa surullinen tai on peloissaan tietokonepelin päästessä tavallista jännittävämpään juonenkäänteeseen. Pelissä on oltava sisäinen logiikka: myös epäuskottavia asioita voi sattua, mikäli ne sopivat pelin tunnelmaan ja aikaisempiin tapahtumiin. Ihmispelaajan tai -katsojan eläytymiskyky on valtava: vaikka teatterin lavasteet olisi tehty kengännauhabudjetilla, voi katsoja silti immersoitua esitykseen, jos siinä olevat hahmot ovat hyvin suunniteltuja ja näyteltyjä.

Teatteriesityksessä immersio särkyy, jos näyttelijä unohtaa vuorosanansa — katsoja muistaa taas, että hahmo lavalla ei olekaan Othello, vaan pelkästään näyttelijä. Samoin tietokonepelissä immersio kaatuu helposti erilaisiin ongelmiin, jotka sotivat pelin sisäistä logiikkaa vastaan. Tyypillinen esimerkki on graafinen bugi, joka

saa vihollisen yllättäen katoamaan, tai huonosti toimiva tekoäly. Tietokonepelaajan tekoälyn merkitys ei siis ole vain tietokonepelin vaikeuttamisessa, vaan tietokonepelaajan on toimittava älykkäästi, koska sen älykäs toiminta on osa tietokonepelin immersiota ja siten pelattavuutta.

Millaiset tekoälyn ominaisuudet antavat vaikutelman tekoälyn älykkyydestä? Scott, Liden ja Orkin luettelevat artikkeleissaan monia eri mahdollisuuksia tehdä tekoälystä ainakin näennäisesti älykäs [Sco02][Lid03][Ork02].

Ensimmäinen ohje on selkeä: tekoälyn ohjaamalla ei-pelaajahahmolla pitää olla ajastimia, jotka keskeyttävät toiminnon, jos se ei onnistu määrätyssä ajassa. Jos kaikki peliohjelmoijat toteuttaisivat tämän ominaisuuden, ei tietokonepeleissä enää nähtäisi maastoon, oviin, siltoihin ja muihin esteisiin juuttuvia ei-pelaajahahmoja tai voittamatonta linnaketta vastaan yksin toinen toisensa jälkeen hyökkääviä örkkitiedustelijoita. Agentti, jonka toiminta näyttää älykkäältä, osaa lopettaa kannattamattoman toiminnan ja siirtyä johonkin tuottoisampaan tehtävään.

Toinen mieluusti toteutettava ominaisuus on yllätyksellisyys. Pelaajan usko tekoälyn voimaan kasvaa, jos tekoäly pystyy johonkin todella merkittävään yllätykseen. Pelaajan on helppo keskittyä kaikessa rauhassa ammuskelemaan ylhäällä tasanteella kävelevää vihollista, mutta jos vastustaja äkkiä kyyristyykin ja hyppää pelaajahahmon viereen, voi pelästynyt ihmispelaaja pudota tuoliltaan. Tällaiset pienet detaljit voivat olla hienoja graafisia tekstuureja tai komeita ääniefektejä merkittävämpiä luotaessa tunnetta todellisesta, dynaamisesta pelimaailmasta.

Kolmas ohje liittyy toteutukseen: Kun ei-pelaajahahmo tietää, mitä aikoo seuraavaksi tehdä, sen täytyy tehdä omalla toiminnallaan pelaajalle selväksi, mitä se on tekemässä. Toisin sanoen, jos tekoäly on tehnyt hienon, innovatiivisen suunnitelman, pelin täytyy varmistua, että ihmispelaaja todella huomaa sen. Esimerkkinä tästä on toimintapeli, jossa vihollinen suurieleisesti ryntää selvästi merkityn valokatkaisijan luokse ja sammuttaa huoneesta valot avittaakseen omaa pakoaan takavasemmalle.

Neljäs ohje koskee ihmiskäyttäytymisen matkimista. Ihmispelaaja on tyypillisesti sekä ennakoitavissa oleva että ennakoimaton. Ennakoitavuus näkyy selvästi esimerkiksi pelityylissä: tietty pelaaja aloittaa reaaliaikastrategiapelin rakentamalla aina yhdenlaisia yksiköitä. Jonkin tällaisen rutiinin kehittäminen ja omaksuminen auttaa luomaan yksilöllisyyttä myös ei-pelaajahahmoille. Toisaalta ihmispelaaja pystyy aina improvisoimaan ja ajattelemaan vapaasti. Myös tietokonepelaajan on voitava tarttua tilaisuuteen ja hylätä vanhat käyttäytymismallit uusien ja parempien tieltä, kun tilanne sitä vaatii.

Miten peliteoria selviää näistä tekoälylle asetettavista vaatimuksista? On muistettava, että peliteoria tarjoaa vain menetelmän agenttien välisen konfliktitilanteen ratkaisemiseen. Agenttien henkilökohtaiset hyötyfunktiot määräävät, mitä agentit oikeastaan haluavat. Jos agentti halutaan keskeyttämään kannattamaton toiminta, on hyötyfunktio suunniteltava niin, että agentti muistaa edelliset toimensa. Jos nykytilanne ei ole muuttunut toivotulla tavalla, alennetaan hyötyfunktiosta hyödyttömäksi todetulla toimella saavutettavaa arvoa. Sama pätee myös yllätyksellisyyteen: jos vihollisagentit halutaan esimerkiksi väijyttämään pelaajahahmo, on niiden hyötyfunktiossa selvästi ilmaistava, että tällainen tilanne on suotava. Hyötyfunktion luominen on aivan oma taiteenlajinsa: siihen palataan luvussa 5.

Miten agentti sitten ilmaisee peliä pelaavalle ihmiselle, mitä se on tekemässä? Tämä kohta liittyy olennaisesti agentin eri toimintamahdollisuuksien valintaan. Jos agentin toiminnot ovat esimerkiksi liikkumissuuntia, voi agentti ennen toimintaansa jollain tavalla esittää, minne on menossa. Samoin jos mahdolliset toimintavaihtoehdot ovat piilossaolo ja hyökkäys, voidaan valittu toiminta graafisesti osoittaa.

Neljäs kohta, eli ennakoitavuuden ja ennakoimattomuuden pieni ero, järjestyy sekastrategioiden ja mahdollista eri tasapainotilojen valinnan avulla. Kun kaikki toiminta ei ole determinististä, vaan todennäköisyydellä on tehtävänsä ainakin osassa pelejä, voivat agentit toimia eri pelikerroilla eri tavalla. Toiminta on kuitenkin peliteorian ansiosta aina rationaalista. Jos tämä ei tunnu riittävän, voidaan satunnaiselementti lisätä myös hyötyfunktioon: tästäkin kerrotaan lisää luvussa 5.

2.4 Ryhmätekoäly ilman peliteoriaa

Koska useimmissa toiminnallisissa tietokonepeleissä pelaajahahmolla on monia vastustajia, on mielekästä, että vihollisagentit ottavat toisensa huomioon myös ilman peliteoreettista mekanismia. Useita erilaisia tekoälytekniikoita onkin kehitetty hallitsemaan agenttiryhmän toimintaa.

Levyn ja Rosenscheinin tekemän peliteoreettisen saalistaja-saalis-ongelman ratkaisun innoittamana Richard Korf julkisti oman tutkimuksensa [LeR92][Kor92]. Siinä hän jatkoi Levyn ja Rosenscheinin tutkimusta, mutta jätti agenttien keskinäisen vaikuttamisen mallintamisen pois. Korf totesi, että Levyn ja Rosenscheinin ratkaisu oli liian monimutkainen, eikä saalis jäänyt kiinni kaikista alkutiloista lähteneissä simulaatioissa. Korfin väittämän mukaan kommunikointi ja yhteistyö ovat saalistaja-saalis -ongelmassa ja moniagenttijärjestelmissä yleensäkin usein tarpeettomia, ja

agentit pääsevät parhaisiin tuloksiin käyttäen niin sanottuja ahneita algoritmeja: kullakin agentilla on hyötyfunktio, mutta agentit eivät etsi tasapainotilaa, jossa päädyttäisiin jokaisen agentin kannalta rationaaliseen vaihtoehtoon. Peliteorian kannalta tämä ei ole minkään agentin kannalta välttämättä optimaalista, sillä jokainen agentti voisi yksinkertaisesti lähteä mallintamaan muita agentteja, ja sitä kautta löytää optimaalisen toiminnan suhteessa muihin. Korf jätti kuitenkin tämän mallintamisen pois ratkaisustaan ja antoi saalistaja-agenttiensa toimia yksinkertaisen algoritmin voimalla.

Korfin käyttämässä algoritmissa jokaiseen saalistajaan kohdistuu puoleensavetävä voima saaliista ja luotaantyöntävä voima muista saalistajista. Koska saalistajat pyrkivät välttämään toisiaan, ne lähestyvät saalista eri suunnista, jolloin sen ympäröinti onnistuu helpommin. Saalis-agentin algoritmi on yksinkertainen: se liikkuu suuntaan, joka vie sen poispäin lähimmästä saalistajasta. Korf salli tutkimuksessaan agenttien diagonaalisen liikkeen, sillä hän katsoi normaalin nelisuuntaisen liikkeen vastaavan huonosti tosimaailman tilanteita. Hänen algoritminsa toimiikin huonosti nelisuuntaisessa liikkeessä: toisiaan välttelevät saalistajat jäävät kiertämään saalista, ja se pääsee pakoon. Diagonaalisen liikkeen takia Korf tarvitsi simulaatioihinsa kahdeksan saalistajaa, jotta saalis ei pääsisi pakoon liikkumalla viistottain. Korf saavutti algoritmeillaan erittäin hyviä tuloksia: tuhannesta testiajosta jokainen päättyi saaliin kiinnisaantiin.

Korfin ratkaisu on mielenkiintoinen, koska se muistuttaa erittäin paljon normaalia skriptattua tietokonepelin tekoälyä. Korfin ajatuksesta yksi askel yksinkertaisempaan suuntaan on tilanne, jossa agentit eivät ota lainkaan huomioon toisiaan. Tällainen tilanne on täysin mahdollinen, jos saalistaja-saalis-ongelman saalistaja-agenteilta poistetaan tarve ympäröidä saalis, ja pelkkä kohteen luokse pääseminen riittää. Funge huomauttaa, että tämän kaltaisessa tilanteessa yksi agentti pystyy ajattelemaan hyötyä pelitiloihin sidottuna [Fun04]. Tällöin Agentti laskee suoraviivaisesti kaikkiin pelin kannalta relevantteihin pelitiloihin sidotun hyödyn, ja pyrkii sellaista ruutua kohti, joka näyttää olevan nopeasti saavutettavissa ja tarpeeksi hyvä. Fungen mukaan tässä voidaan käyttää rekursiivista Markov-päätösprosessin hyödynlaskemiskaavaa, josta kerrotaan enemmän luvussa 6.1.

Funge kirjoittaa, että monissa tietokonepeleissä käytetään *parveilua* (flocking) kuvaamaan ryhmän liikkumista [Fun04]. Reynolds kertoo artikkelissaan, että jokainen parven jäsen pyrkii huolehtimaan kolmesta asiasta: nopeuden täsmäämisestä muiden parven jäsenien kanssa (suunta), törmäysten välttämisestä (separaatio) ja

parven keskustassa pysymisestä (koheesio) [Rey99]. Parven jäsenillä on tavoitteellinen nopeusvektori $\mathbf{v}_d$, joka koostuu näitä kolmea tavoitetta kuvaavista komponenttivektoreista: suunnasta $\mathbf{v}_a$, koheesiosta $\mathbf{v}_c$ ja separaatiosta $\mathbf{v}_s$. Jokainen komponenttivektori kerrotaan painokertoimella w, joiden keskinäinen suhde määrää, millaiseksi parvi rakentuu.

$$\mathbf{v}_d = w_0\mathbf{v}_a + w_1\mathbf{v}_c + w_2\mathbf{v}_s$$

Nopeuden määräytyminen muistuttaa paljon hyötyfunktion rakentumista peliteoreettisessa agenttimallissa, josta puhutaan lisää luvussa 5.2. Parveilumallilla saadaan agenttiryhmän liikkuminen luonnollisen näköiseksi, varsinkin jos ryhmälle on valittu johtajaagentti, joka vie parvea eteenpäin. Kysymyksiin piilossapysymisestä ja hyökkäyksen tekemisestä uskottavan näköisesti parveilumenetelmät eivät kuitenkaan pysy vastaamaan.

Tamben mukaan toinen yleinen tapa toteuttaa agenttien välistä koordinaatiota on komentohierarkian käyttö [Tam97]. Siinä ajatuksena on luoda komentoketju ja tehdä päätöksiä eri hierarkian tasoilla. Yksittäisellä vihollisagentilla voi esimerkiksi olla päämäärä, jonka sille antaa ylemmän tason kontrolleri, jolla voi puolestaan olla oma kontrolleri. Hyökkäyspeliin sovellettuna vihollisagenttiryhmän ylemmän tason kontrolleri koordinoi agenttien etenemistä mahdollisimman tehokkaaksi ja saa ne ottamaan huomioon toisensa hyökkäyksen aikana. Äärimmilleen vietynä komentohierarkiassa ei voida enää puhua moniagenttijärjestelmästä, vaan pelikentällä olevat agentit ovat vain kontrolloivan ”superagentin” aktuaattoreita. Peliteoreettinen agenttimalli muistuttaa komentohierarkiaa siinä, että peliteoreettinen moottori laskee agentteille sopivan toimintoyhdistelmän, jonka agentit sitten toteuttavat. Erona komentohierarkialla ja peliteoreettisella mallilla on, että peliteoreettisesti ajatellevat agentit eivät enää toimi absoluuttisen tehokkaasti, vaan henkilökohtaisen rationaalisesti. Tämä auttaa luomaan illuusiota oikeasta älykkyydestä agenttien toiminnassa.

Tambe kirjoittaa myös, että eräs tapa tehdä älykkään näköistä usean agentin yhteistoimintaa on roolijako. Esimerkkinä tästä on lipunryöstöpeli, jossa yksi agentti saa tehtäväkseen puolustaa omaa lippua, kun toinen agentti pyrkii varastamaan vastapuolen lipun. Rooleja voidaan vaihtaa dynaamisesti pelitilanteen niin vaatiessa. Rooli antaa jokaiselle agentille yksinkertaisen toimintamallin, joka saa agenttiryhmän toiminnan näyttämään älykkäämmältä. Agentille annettu rooli voidaan yhdistää myös peliteoreettiseen agenttimalliin vaihtamalla agentin hyötyfunktio uuteen, mikäli pelitilanne sitä vaatii. Toisaalta tälle ei pitäisi olla tarvetta, sillä esimerkiksi

yhden vihollisagentin jättäytyminen suojatulta antavaan rooliin voisi hyvinkin olla Nashin tasapainotila agenttien keskinäisessä pelissä.

3 Peliteorian soveltaminen tietokonepeleissä

Tässä luvussa käsitellään käytännön kysymyksiä, joita tulee vastaan sovellettaessa peliteoriaa tietokonepeleihin. Aluksi tarkastellaan peliteoreettisten pelien rakennetta ja tietokonepelin tekoälyn temporaalista luonnetta. Sen jälkeen mietitään peliteorian ongelmakohtia ja ratkaisuja niihin. Lopuksi käsitellään tietokonepelien rakennetta ja agenttimallia sekä sovitetaan peliteorian tarjoamia työkaluja tietokonepeleihin.

3.1 Tietokonepelien tekoälyn tasot

Spronck jakaa tietokonepeleissä käytettävät tekoälyvastustajat kolmeen luokkaan sen mukaan, mitä ne yrittävät tehdä [Spr04]. Operationaalisella tasolla toimivat vastustajat ohjaavat yhden agentin liikettä ja toimintaa. Ne pyrkivät tekemään päätöksen, joka antaa agentille juuri sillä hetkellä parhaan lopputuloksen. Taktisella tasolla liikkuvat vastustajat pyrkivät pääsemään johonkin tiettyyn tavoitteeseen sopivalla toimintasarjalla. Strategisella tasolla vastustaja tekee kauaskantoista suunnittelua, jonka tavoite on pitkällä aikavälillä voittaa pelaaja.

Peliteoria pystyy tarkastelemaan sekä yksittäisiä konflikteja että konfliktisarjoja. Yksittäistä konfliktia, jossa agentit toimivat samanaikaisesti tai ilman tietoa toistensa valitsemista toiminnoista, kuvataan usein *normaalimuotoisena pelinä* (normal form game). Taulukossa 1 oleva esitys Vangin ongelmasta on normaalimuotoinen peli. Toistuvista siirroista, jotka ovat vuoroittaisia tai samanaikaisia, muodostuvaa pelipuuta kutsutaan *laajamuotoiseksi peliksi* (extensive form game).

Hyökkäyspelissä jokainen vihollisagentti valitsee yhdellä normaalimuotoisen pelin ratkaisukerralla strategian, jonka seurauksena se voi liikkua yhden ruudun. Valittu toimintastrategia on siten oikea ainoastaan siinä tilanteessa, jossa agentit valintahetkellä olivat — päätöksenteko jää selvästi Spronckin luokituksen mukaan operationaaliselle tasolle. Tietokonepelien luonteen mukaisesti pelitilanne muuttuu vihollisagenttien ja pelaajan toiminnan takia koko ajan. Kuitenkin tärkeä osa tietokonepelien pelattavuutta on tekoälyn toimiminen johdonmukaisesti: on hyvä, jos vihollisagentit toimivat määrätietoisen näköisesti, eli toisin sanoen tuntuvat suunnittelevan tekemisiään. Selvää on, että jos peliteoreettisessa tekoälyssä halutaan päästä Spronckin luokituksen mukaiselle taktiselle tasolle, on agentin suunniteltava toimiaan jollain tavalla eteenpäin.

Boutilierin ja muiden mukaan suunnittelu on tekoälyssä tunnettu ongelma, ja mer-

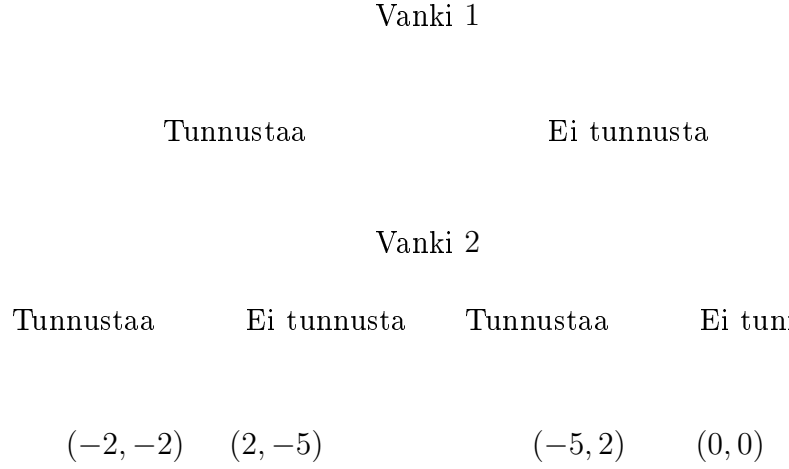
kittävä määrä kirjallisuutta käsittelee erilaisia tekniikoita, joilla suunnitteluongelmia voidaan ratkaista [BDH99]. On olemassa joitain tapoja tapaa käyttää peliteoriaa agenttien toiminnan kauaskantoisempaan suunnitteluun tietokonepelissä: kaksi esimerkkiä ovat suunnittelun lisääminen hyötyfunktioon ja pelin ajatteleminen siirtojen sarjana. Tietokonepeleissä vihollisagenttien suorittamaa suunnittelua mutkistaa valtavasti pelaajan vaikutus: jos pelaajahahmo liikkuu tai muuten toimii, voivat parhaatkin suunnitelmat sotkeutua. Peliteoria pystyy ottamaan huomioon pelaajan toiminnan, mutta se vaatii pelaajan mallintamisen joko yhtenä peliteoreettisen pelin pelaajista tai osana dynaamista ympäristöä.

3.2 Suunnittelu ja laajamuotoiset pelit

Käytännössä kaikki tietokonepelit ovat ajassa eteneviä prosesseja, ja niin on Hyökkäyspelikin. Jos Hyökkäyspeliä ajatellaan siirroittain, se muistuttaa pitkälti shakkia. Ensin siirtävät vihollisagentit, sitten pelaaja, sitten taas vihollisagentit ja niin eteenpäin. Tietokonetta vastaan pelattavassa shakissa ihmispelaaja on mallinnettava toisena pelin pelaajana, koska vuorojen eteenpäin suunnittelu on tärkeä osa pelin menestyksekästä pelaamista. Shakkia kuvataan usein puuna, jossa molempien pelaajien siirtovuorot vuorottelevat.

Dutta esittää laajamuotoisen pelin pelipuuna [Dut99]. Pelipuun solmuja kutsutaan päätössolmuiksi, koska jokaisen solmun kohdalla päätösvuorossa olevan pelaajan pitää valita toimintastrategia. Pelaajien toimintovalinnat yhdistävät päätössolmuja. Yhden pelaajan toimintovalinnan jälkeen on seuraavan pelaajan vuoro päättää, mitä haluaa tehdä. Jos pelaajat eivät tiedä toistensa toimintoja tai valitsevat ne samanaikaisesti, ei pelipuuhun luonnollisesti kuuluva vuorottaisuus toimi sellaisenaan. Tällöin pelaajan, joka ei tiedä edellisen pelaajan tekemää toimintavalintaa, päätössolmujen sanotaan kuuluvan samaan informaatiojoukkoon.

Esimerkki laajamuotoisesta pelistä on kuvassa 3. Kuvan pelipuussa kaikkein ylintä päätössolmua kontrolloi Vanki 1. Vanki 2 näyttää päättävän toiminnastaan pelipuussa vasta ensimmäisen vangin jälkeen. Koska toimintapäätös tapahtuu kuitenkin pelissä samanaikaisesti, toisen vangin molemmat päätössolmut kuuluvat samaan informaatiojoukkoon. Informaatiojoukko esitetään kuvassa katkoviivalla, joka yhdistää päätössolmut. Pelipuu haarautuu myös toisen vangin päätössolmuista ja johtaa kaikista eri strategiavektoreista saataviin hyötyihin. Saman pelin normaalimuotoista esitystä taulukossa 1 voi verrata laajamuotoisen pelin pelipuuhun.



Kuva 3: Esimerkki laajamuotoisesta Vangin ongelmasta.

Peliteoreettisesti ajatellen jokainen vihollisagenttiryhmän jäsen on pelin pelaaja. Kuitenkin ihmispelaajan silmissä vihollisagentit eivät valitse toimintoaan vuorotellen vaan samanaikaisesti. Jos vihollisagenttiryhmää halutaan ajatella Hyökkäyspelin ihmispelaajan vastustajana, on vihollisagenttiryhmän toimittava kuin yksi pelaaja. Jotta monen agentin valitsemat siirrot voidaan supistaa yhdeksi agenttiryhmän tekemäksi toiminnon valinnaksi, täytyy määrittää *alipelin* (subgame) ja *alipelitäydellisen tasapainotilan* (subgame-perfect equilibrium) käsitteet.

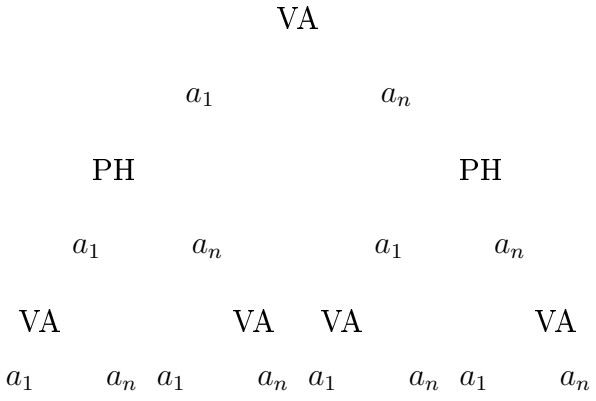
Alipeli on Duttan mukaan laajamuotoisen pelin osa, joka täyttää kolme ehtoa:

1. Alipeli alkaa yhdestä päätössolmusta.
2. Alipeli pitää sisällään aloittavan solmun jokaisen jälkeläisen.
3. Jos alipeli pitää sisällään minkä tahansa osan informaatiojoukkoa, sen on sisällettävä kaikki informaatiojoukon jäsenet.

On helppo nähdä, että koko pelipuu on itsensä alipeli. Myös jokainen pelipuun täydellinen haara on oma alipuunsa.

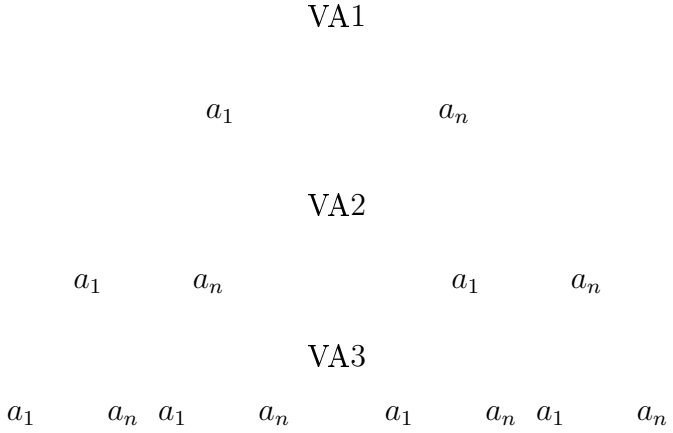
Laajamuotoisissa peleissä strategian käsite on laajempi: jokaisella agentilla on oltava toimintastrategia jokaiseen valintatilanteensa pelipuun eri tasoilla. Alipelitäydellinen tasapainotila monen pelaajan pelille on Duttan mukaan strategiavektori $s_1, \ldots, s_n$, jos jokaiselle alipelille g strategiat $s_1(g), \ldots, s_n(g)$ muodostavat tasapainotilan. Toisin sanoen, jotta strategiavektori olisi alipelitäydellinen tasapainotila, täytyy strategiavektorin ratkaista tasapainotilaksi jokainen pelin mahdollinen alipeli.

Alipelitäydellinen tasapainotila löydetään Duttan mukaan takaperoisella induktiolla [Dut99]. Induktio aloitetaan alipelistä, joka päättyy lehtisolmuihin. Alipelin tasapainotila etsitään, ja eri pelaajien tasapainotilan pelaamisesta saavat hyödyt merkitään alipelin tilalle. Tämä toistetaan kaikille muillekin lehtisolmuihin päättyville alipeleille. Kun kaikki viimeiset alipelit on supistettu hyödyiksi, tehdään sama toiseksi viimeisille alipeleille. Tätä jatketaan, kunnes koko pelipuu on käsitelty. Ylimmälle tasolle tuleva tasapainotila on Duttan mukaan koko pelipuun alipelitäydellinen tasapainotila.



Kuva 4: Laajamuotoisen pelin kulku. VA tarkoittaa vihollisagenttiryhmää ja PH pelaajahahmoa.

Hyökkäyspelissä tapahtuvaa suunnittelua voidaan mallintaa juuri alipelitäydellisen tasapainotilan avulla. Kun vihollisagentit tekevät siirtonsa samanaikaisesti, vihollisagenttien keskinäisen toiminnan päättäminen vaatii tasapainotilan etsinnän. Ihmispelaaja otetaan mukaan pelipuuhun niin, että pelaaja siirtää joka toisella vuorolla ja vihollisagenttien ryhmä joka toisella. Tällöin laajamuotoisen pelin etenemispolun joka toinen solmu pitää sisällään normaalimuotoisen pelin, joissa pelaavat vain vihollisagentit. Kuva 4 selventää asiaa. Yksi vihollisagenttien keskinäinen peli, joka pelataan kuvan 4 jokaisella vihollisagenttien siirrolla, on esitetty kuvassa 5. Alipelitäydellisen tasapainotilan löytäminen tarkoittaa, että jos ihmispelaaja ja vihollisagenttiryhmä laskevat peliä eteenpäin saman rajallisen määrän kierroksia, pitäisi kaikkien pelaajien rationaalisina toimijoina pelata juuri alipelitäydellinen tasapainotila.



Kuva 5: Vihollisagenttiryhmän keskinäinen peli. Horisontaalinen katkoviiva kuvaa kuulumista samaan informaatiojoukkoon. Toisin sanoen, kaikki tämän pelin pelaajat tekevät toimintavalintansa tietämättä muiden pelaajien valinnasta, eli samanaikaisesti.

Koska vihollisagenttiryhmä tekee päätöksensä samanaikaisesti, vihollisagenttien toimintojen valinta on *epätäydellisen tiedon* (imperfect information) peli. Toisaalta koska vihollisagenttiryhmä tietää ihmispelaajan edellisen siirron, on ihmispelaajan ja vihollisagenttiryhmän välinen peli *täydellisen tiedon* (perfect information) peli. Koska vihollisagentit eivät tiedä toistensa siirtoja, merkitään kuvassa 5 pelaajan VA2 päätössolmut kuulumaan samaan informaatiojoukkoon, sillä pelaaja VA1 on saattanut pelata minkä tahansa toiminnoistaan. Samalla tavalla pelaaja VA3 joutuu ottamaan huomioon, että pelaajien 1 ja 2 toiminnot ovat tuntemattomia. Laajamuotoista epätäydellisen tiedon peliä voidaankin Duttan mukaan kuvata normaalimuotoisena pelinä.

Kun peliä suunnitellaan eteenpäin, on ihmispelaajan eri pelitilanteista saama hyöty pakko arvioida, koska muuten tasapainotilaa ei ole mahdollista löytää. On tärkeää ymmärtää, että pelipuun avulla yritetään Hyökkäyspelissä nimenomaan arvioida, minne peli on menossa, eikä niinkään kuvata, mitä pelissä tapahtuu. Ihmispelaajan toimet eivät siis ole oikeita toimia siinä mielessä, että ne on jo tehty tai että ne varmasti tehtäisiin. Kyseessä on ennemmin yritys saada vihollisagentit suunnittelemaan toimintaansa eteenpäin ja ottamaan siinä myös ihmispelaajan toiminta huomioon.

Ihmispelaajan hyötyfunktiosta puhutaan kappaleessa 3.3.

Vihollisagenttiryhmän muodostama epätäydellisen tiedon peli on siis kuvattavissa suoraan normaalimuotoisena pelinä, jossa pelaajat tekevät toimintovalintansa samanaikaisesti. Tämä on intuitiivisesti selvää: koska agenttiryhmän ei keskinäisessä pelissään kannata pelata muuta kuin Nashin tasapainotiloja, ovat kaikki agenttiryhmän alipelistä tulevat toimintapäätökset alipelin tasapainotiloja. Tämän takia voidaan vihollisagenttien keskinäisen pelin ratkaisemiseen käyttää luvussa 4 esiteltäviä menetelmiä.

Tämä on juuri se kohta, jossa tietokonepelin pelattavuus tulee mukaan: agenttien ei ole pakko valita yhteensä parhaan hyödyn antavaa strategiavektoria, jos se on jonkin agentin mielestä huono — tällöin tasapainotila siirtyy muualle. Pelin vuoronperäisyys ei itse asiassa ole ongelma, sillä vihollisagenttiryhmä pärjää mitä luultavimmin yhdellä valitulla strategiavektorilla esimerkiksi sekunnin ajan, jonka jälkeen seuraava strategiavektori voidaan laskea. Pelaajahahmon toimintaa tämän sekunnin aikana voidaan ajatella pelaajahahmon valitsemana strategiana jonkinlaisen pelkistyksen avulla. Laajamuotoisen pelin tekniikkaa käyttämällä vihollisagentit pystyvät joukkona suunnittelemaan toimiaan eteenpäin pelissä, mutta silti jokainen yksittäinen agentti säilyttää oman rationaalisuutensa.

Tilannetta mutkistaa pelissä oleva satunnaiselementti. Jos pelaajahahmo tulittaa vihollisagentteja, on olemassa mahdollisuus, että missä tahansa pelin vaiheessa yksi tai useampia vihollisagentteja haavoittuu tai kuolee. Tyypillisesti tietokonepeleissä on eri asteisia haavoittumisia: agentilla on yleensä kokonaislukuna ilmaistava määrä kestopisteitä, jotka mittaavat agentin terveydentilaa. On selvää, että jos agentti vammautuu vakavasti, etukäteen lasketut suunnitelmat muuttuvat perustavanlaatuisesti. Samoin on mahdollista, että jonkin tapahtuman seurauksena pelimaailma muuttuu jollain tavalla tai vihollisagentti joutuu vaikkapa paineaallon tönäisemänä aivan toiseen ruutuun, kuin minne se luuli menevänsä. Laajamuotoisissa peleissä todennäköisyyden vaikutukset voidaan Duttan mukaan kuvata *mahdollisuussolmuilla* (chance nodes) [Dut99]. Ne ovat solmuja, jotka lisätään pelipuuhun kuvastamaan jonkin asian tapahtumisen mahdollisuutta. Pelipuu haarautuu aina, kun jokin todennäköisyys selvitetään – mahdollisuussolmusta lähtee niin monta haaraa, kuin mitä vaihtoehtoisia tapahtumia on. Eräs tällainen todennäköisyyden ratkeaminen on, kun pelaajahahmo joko haavoittuu tai ei haavoitu. Jos vuorojen välissä voi tapahtua monta asiaa, tulee päätössolmujen väliin yhtä monta mahdollisuussolmua, kuin mitä erilaisia mahdollisesti tapahtuvia asioita on.

Pelipuun massiivinen haarautuminen on syynä laajamuotoisen pelin käytännön toteuttamisen vaikeudelle. Pelkästään jokaisesta vihollisryhmän päätössolmusta, jossa on n vihollisagenttia k:lla eri toimintovaihtoehdolla, lähtee k^n uutta haaraa uusiin päätössolmuihin. Jos esimerkiksi jokainen vihollisagentti saattaa haavoittua kahdella eri vaikeudella tai olla haavoittumatta, tulee päätössolmujen väliin vielä n peräkkäistä mahdollisuussolmua, joista jokaisesta lähtee 3 haaraa. Jos pelkästään yhden staattisen Nashin tasapainotilan löytäminen on vaikeaa, on tuhansien tasapainotilojen selvittäminen lähes mahdotonta tarpeeksi lyhyen ajan sisällä. Vaikka laajamuotoisen pelin soveltaminen shakkiin tunnetusti onnistuu, se ei käy sellaisenaan tulevaisuuden suunnitteluun toimintapeleissä, kenties lukuun ottamatta erittäin yksinkertaisia pelejä.

Dutta kirjoittaa, että laajamuotoisten pelien erikoistapaus ovat iteratiiviset eli toistuvat pelit [Dut99]. Iteratiivisissa peleissä ajatuksena on, että samaa normaalimuotoista peliä — esimerkiksi Vangin ongelmaa — toistetaan monta kertaa peräkkäin. Jos Vangin ongelmaa pelataan tietty ennalta määritetty määrä kierroksia, ei tasapainotila muutu: molempien vankien kannattaa edelleen tunnustaa. Tämä on intuitiivisesti helppo ymmärtää, kun ajattelee peliä, jossa on n kierrosta, ratkaistuna aiemmin käsitellyllä takaperoisella induktiolla. Kierroksella n peli ei ole enää toistuva peli, koska seuraavaa kierrosta ei tule. Agentit voivat siis pelata viimeisen kierroksen kuten pelaisivat aivan tavallista normaalimuotoista peliä: agenttien toimintapäätökset eivät varmasti vaikuta seuraavaan kierrokseen. Koska kierroksen n lopputulos on riippumaton toistuvasta pelistä, pelaajat voivat ajatella, että kierros $n-1$ on pelin viimeinen kierros. Tällöin siihen kuitenkin pätee sama havainto kuin kierrokseen n: mikään ei riipu tämän kierroksen lopputuloksesta, joten se kannattaa pelata aivan normaalisti. Induktioperiaatteen mukaan tämä pätee pelin kaikkiin kierroksiin. Toisin sanoen, jos tietokonepelin tilanne saadaan sopimaan toistettavan pelin muottiin, se on ratkaistavissa merkittävästi yleistä tapausta helpommin.

Russell ja Norvig kertovat, että jos toistuvan pelin kierroksien määrää ei ole etukäteen määrätty, muuttuu tilanne täysin [RuN03]. Esimerkiksi Vangin ongelmassa ei tunnustaminen enää ole välttämättä oikea strategia pelattavaksi, vaan pelaajat voivat jättää tunnustamatta luottaen siihen, että seuraavissa pelissä se palkitaan. Kuuluisa strategia tällaisen pelin pelaamiseen on 'tit-for-tat', jossa ensimmäisellä kierroksella pelaaja ei tunnusta ja jokaisella seuraavalla kierroksella toimii samalla tavalla kuin vastustaja edellisellä kierroksella. Russellin ja Norvigin mukaan tit-for-tat on osoittautunut hyväksi vastineeksi useille eri strategioille. Jos molemmat pelaajat pelaavat Vangin ongelmassa tit-for-tat-strategian, kumpikaan vanki ei koskaan

tunnusta, ja molempien vankien saama hyöty on maksimaalinen. Jos Hyökkäyspelin kestolle tai päättelyn sallitulle pituudelle ei aseteta mitään rajoja, joudutaan tilannetta tarkastelemaan päättymättömänä pelinä. Tähän ei normaalitilanteessa pitäisi olla tarvetta, sillä Hyökkäyspelissä ei ole määrittelemättömän kauan samanlaisina toistuvia pelejä. Kappaleessa 6.1 sivutaan kuitenkin Markovin pelejä, jotka muistuttavat jossain määrin äärettömiä pelejä.

3.3 Ihmispelaajan mallintaminen

Ihmispelaaja voidaan tuoda mukaan agenttiryhmän keskinäiseen päätöksentekoon kahdella eri tavalla: joko yhtenä pelin pelaajista tai osana ympäristöä.

Jos ihmispelaaja halutaan mukaan peliteoreettisena pelaajana, ensimmäinen ongelma on ihmispelaajan hyötyfunktio. Miten tiedetään, millaisia hyötyarvoja ihmispelaaja asettaa eri pelitilanteille? Tämän ymmärtäminen on tärkeää, sillä jos ihmispelaajan tavoitteita ei tiedetä, ei ihmispelaajan toimintoja voida ottaa huomioon. Shakissa jokaiselle pelitilanteelle pyritään laskemaan suhteellinen hyvyys, ja siinä ihmispelaajan oletetaan pyrkivän pelaamaan kohti mahdollisimman hyviä pelitilanteita. Hyökkäyspelissä asia ei ole aivan näin helppo, sillä ihmispelaajalla voi pelitilanteen mukaan olla tavoitteita ja pelitaktiikoita, jotka eivät ole helposti arvattavissa. Jossain tapauksessa ihmispelaaja voi haluta kiertää vihollisagentit, kun taas toisessa tilanteessa hyökätä niitä vastaan — ihmispelaajan oma pelityyli määrää, millaisesta tilanteesta hän saa suurimman hyödyn. Ihmispelaajan päätökset eivät myöskään välttämättä ole rationaalisia. Parsons ja Woolridge kertovat, että laboratoriokokeissa Vangin ongelmaa pelaavat ihmispelaajat eivät läheskään aina valitse tunnustamista, vaikka se on tasapainotila ja siten paras valinta [PaW02].

Yksinkertainen arvio ihmispelaajan jostain pelitilasta saamasta hyödystä kuitenkin on, että se on agenttiryhmän saaman hyödyn vastaluku. Tällöin kyseessä on vakiosummapeli, jossa ihmisagentti menettää yhtä paljon hyötyä kuin vihollisagentit kollektiivisesti saavat, ja toisinpäin. Kun vakiosummapelin vakio valitaan nollaksi, on kyseessä *nollasummapeli* (zero sum game). Hyökkäyspelin luonteesta johtuen arvio ei aina ole kovin tarkka, sillä vihollisagenttien ja pelaajahahmon saamat hyödyt eivät aina ole vastakkaisia. Esimerkiksi tilanne, jossa vihollisagentti lähtee juoksemaan pakoon, voi olla sekä ihmispelaajan että vihollisagentin kannalta kaikkein paras.

Käytännössä ihmispelaajan saama hyöty löydetään niin, että kaikki mahdolliset ihmispelaajan toiminnot eli siirtovaihtoehdot käydään läpi. Jokaisesta mahdollisesta

toiminnosta lasketaan vihollisagenttiryhmän yhteenlaskettu hyöty, mikäli toiminto valittaisiin. Ihmispelaaja asettaa paremmuusjärjestyksessä sen toiminnon etusijalle, joka antaa vihollisagenttiryhmälle pienimmän hyödyn.

Kappaleessa 3.2 käsiteltiin ihmispelaajan mallintamista laajamuotoisen pelin, jossa ihmispelaaja ja vihollisagenttiryhmä siirtävät vuorotellen, yhtenä pelaajana. Toinen mahdollisuus on ottaa ihmispelaaja mukaan normaalimuotoiseen peliin. Teoriassa on mahdollista, että ihmispelaaja tulee mukaan valitsemaan tasapainotilaa ja saa oman ulottuvuutensa agenttien hyötymatriisista, josta tasapainotilaa etsitään. Käytännössä ihmispelaajan ottaminen luontevasti mukaan peliin on kuitenkin vaikeaa monestakin syystä, joista yksi on ihmispelaajan rajoittunut rationaalisuus ja toinen on samanaikaisen päätöksenteon vaikeus reaaliaikaisessa pelissä. Onkin luontevampaa ajatella pelaajahahmoa yhtenä osana ympäristöä, jossa vihollisagentit toimivat. Ihmispelaaja on silloin osa dynaamista maastoa, jonka tilaa eli tulevaa toimintaa ei voida luotettavasti ennustaa. Parsonsin ja Woolridgen mukaan klassinen päätösteoria pystyy kuitenkin käsittelemään tätä tapausta [PaW02]. Heidän mukaansa päätösteoria on peliteorian läheinen sukulainen, joka käsittelee toimintapäätösten tekemistä epävarmassa tilanteessa.

Ajatuksena on, että agenttien ympäristössä voi olla tuntemattomia muuttuja. Muuttujat $X_1, \ldots, X_n$ sisältävät jokainen jonkin tiedon ympäristöstä. Muuttujalla X_k on joukko tiloja $x_{k_1} \ldots x_{k_m}$, joissa se voi olla. Muuttuja X_k on tilassa x_{k_j} todennäköisyydellä $p(x_{k_j})$, kun p saa arvoja väliltä $[0 \ldots 1]$ ja $\sum_j p(x_{k_j}) = 1$. Jos tuntemattomia muuttujia on useampia, on selvitettävä, onko niillä keskinäistä riippuvuussuhdetta. Hyökkäyspelin tapauksessa voidaan kuitenkin olettaa, että vihollisagenttien päätöksentekovaiheessa vain pelaajahahmon toimintaa tarvitsee mallintaa probabilistisesti. Nyt sen sijaan, että laskettaisiin pelkkä vihollisagentti i:n saama hyöty v_i tietyllä vihollisagenttien strategiavektorilla (s_i, s_{-i}), voidaankin laskea jokaisen strategiavektorin agentille i tuoma hyöty kerrottuna pelaajahahmon toiminnan odotusarvon antamalla painolla.

Olkoon U_i jokin vihollisagentti i:n hyötyfunktiossa oleva termi, joka lasketaan normaalisti kaavalla $U_i = u(x_k)$, missä $u(x_k)$ on pelin ympäristön ominaisuuden X_k tilasta x_k saatava hyöty. Nyt, mikäli x_k voi saada eri arvoja pelaajan toimien seurauksena, saa U_i arvokseen eri tilanteista tulevan hyödyn odotusarvon:

$$U_i = \sum_{x_{k_j} \in X_k} p(x_{k_j}) u(x_{k_j})$$

Vaikeampaa on sanoa, millä todennäköisyyksillä pelaaja vaikuttaa peliin, eli mitä

arvoja todennäköisyysfunktio p saa milläkin arvolla x. Joissain tapauksissa pelisuunnittelijan täytyy esittää valistunut arvaus, mutta esimerkiksi pelaajahahmon sijaintia tai liikkumista voidaan arvioida melko yksinkertaisesti.

```
............
......b.....
.....bab....
....baPab...
.....bab....
......b.....
............
```

Kuva 6: Pelaajahahmon mahdollinen sijainti eri vuoroilla.

Pelaajahahmon sijaintia voidaan pitää yhtenä ympäristön ominaisuutena. Jos pelaajahahmon sijainti on varma, muuttujalla X_{sij} on vain yksi tila, jonka todennäköisyys on 1: pelaajahahmon olinpaikka. Jos pelaajahahmo ei ole näkyvissä, saa muuttuja X_{sij} lisää tiloja pelaajahahmon mahdollisten olinpaikkojen mukaan. Kuva 6 havainnollistaa asiaa. Jos pelaajahahmosta ei ole havaintoa esimerkiksi kahteen kierrokseen, on mahdollisia ruutuja, joissa pelaaja voi olla, yhteensä 13 kappaletta. Ne on merkitty kuvaan 6 niin, että **a**-kirjaimella on merkitty ruutuja, joissa pelaaja voi olla yhden vuoron kuluttua, ja **b**-kirjaimella niitä ruutuja, joissa pelaaja voi olla kahden vuoron kuluttua **a**-ruutujen lisäksi. Näistä ruuduista lasketaan pois ne ruudut, jotka vihollisagentti pystyy suoraan havaitsemaan. Viimeinen tunnettu olinpaikka on pelaajahahmon kaikkein todennäköisin sijainti, ja muiden ruutujen todennäköisyys vähenee suhteessa ruudun etäisyyteen viimeisestä tunnetusta ruudusta. Hyötyfunktioon tulevat arvot etäisyydestä pelaajahahmoon lasketaan jälleen hyödyn odotusarvon perusteella.

3.4 Peliteorian soveltamisen ongelmia

Peliteorian soveltamisessa olevat ongelmat voidaan jakaa karkeasti kahteen luokkaan: loogisiin ja teknisiin ongelmiin. Tekniset ongelmat liittyvät aika- ja tilavaativuuskysymyksiin. Niitä on sivuttu aiemmin, ja luvussa 4.1 tarkastellaan asiaa yksityiskohtaisemmin. Loogiset ongelmat ovat kysymyksiä peliteoreettisten mallien soveltuvuudesta erilaisiin käytännön sovelluskohteisiin. Seuraavassa tarkastellaan joitain loogisia ongelmakohtia ja pohditaan niiden merkitystä Hyökkäyspelin kannalta.

Russell ja Norvig kirjoittavat, että peliteoriaa on käytetty sen ongelmien takia lähinnä erilaisten konfliktitilanteiden analysoimiseen agenttien ohjauksen sijasta [RuN03]. Heidän mukaansa Nashin tasapainotilan käyttämisen yhtenä ongelmana on, että pelaajan täytyy olettaa, että kaikki muut agentit pelaavat varmasti tasapainotilan mukaisen strategian. Muiden pelaajien hyötyfunktiota ja sitä kautta toiminta-aikomusta ei kuitenkaan aina pysty tietämään tarkasti. Siksi pelaaja joutuu varautumaan myös uhkiin, jotka eivät ehkä koskaan toteudu. Bayesin-Nashin tasapainotilaa käytetään ratkomaan tätä ongelmaa: siinä pelaajalla voi olla etukäteen määritetty todennäköisyysjakauma muiden pelaajien strategioiden yli. Peliä, jossa jokin pelaaja ei välttämättä tiedä muiden pelaajien eri tilanteista saamia hyötyjä, kutsutaan Russellin ja Norvigin mukaan *rajallisen tiedon* (incomplete information) peleiksi.

Tasapainotilan käsite ei myöskään pysty ottamaan huomioon pelaajien riskinsietokykyä. Riskipelissä, joka esitellään taulukossa 3, Nashin tasapainotila on kohdassa $(2, 2)$. Kuitenkin jos pelaaja 2 pelaa strategian A, joutuu pelaaja 1 pahasti tappiolle. Siksi pelaajan 2 kannattaa ehkä pelata strategia A, koska pelaaja 1 ei kenties uskalla ottaa riskiä ja pelata strategiaa B.

		Pelaaja 2	
		A	B
Pelaaja 1	A	(0,1)	(1,0)
	B	(-1000,0)	(2,2)

Taulukko 3: Riskipeli.

Hyökkäyspelin vihollisagenttien mallinnuksessa riskien hallinnalla ei ole suurta merkitystä, sillä agentit eivät ole inhimillisiä: niiltä ei edes odoteta täysin ihmismäistä käyttäytymistä. Ne voivat helposti ja luotettavasti laskea monimutkaisiakin hyödyn odotusarvoja, mitä ihmiseltä ei välttämättä saata odottaa. Jokainen pelissä oleva agentti on itsessään täysin deterministinen, eikä sillä ole mitään vaikeuksia päätellä, mitä muut agentit tekevät: ne toimivat varmasti juuri peliteoreettisesti rationaalisella tavalla.

Myös rationaalisuusvaatimusta vastaan on hyökätty. Kaikki ihmiset eivät ole rationaalisia siinä mielessä, että haluaisivat vain maksimoida oman hyötynsä, vaan myös altruismilla on sijansa oikeassa maailmassa. Harva itsekäskään ihminen haluaisi itselleen pientä voittoa, jos moni muu saa sen tuloksena merkittävän haitan. Stirling ja muut kuitenkin muistuttavat, että vaikka agentilla voi olla muitakin tavoitteita kuin henkilökohtaisen edun tavoittelu, nekin voidaan merkitä hyötyfunktioon

[SGP02]. Esimerkiksi jos jompikumpi Vangin ongelmaa pelaavista agenteista pitää toisen vangin hyötyä itselleen merkittävänä tavoitteena, muuttuu agentin hyötyfunktio aivan toisenlaiseksi ja sen mukana koko pelin luonteesta (ja mahdollisesti tasapainotilasta) tulee erilainen.

Stirling ja muut toteavat myös, että klassisen peliteorian ongelma on agenttien muodostaman ryhmän tavoitteiden kuvaus. Koska agenteilla ei sinänsä ole minkäänlaisia riippuvuussuhteita toistensa kanssa, konfliktit ja yhteistyömahdollisuudet tulevat esille vasta siinä vaiheessa, kun agenttien tietystä tilanteesta saamia hyötyjä verrataan keskenään [SGP02]. Stirling ja muut ehdottavat, että tulevaisuuden peliteoreettisissa malleissa pyritään korostamaan agenttien välisiä riippuvuuksia jo hyötyfunktiota määritettäessä. Agentit miettivät, mitä haluavat, ja sitten tutkivat muiden agenttien tarjoamia mahdollisuuksia yhteistyöhön toivottuun päämäärään pyrkimisessä.

Tietokonepeleissä tasapainotilan valitseminen on tärkeä ongelma. On helppo kuvitella Hyökkäyspelin pelitilanne, jossa on kaksi tasapainotilaa: kaikki agentit hyökkäävät kohti pelaajahahmoa tai kaikki pakenevat pelaajahahmon luota. On täysin mahdollista, että kumpikaan tasapainotiloista ei hallitse toista. Miten tasapainotilan valitsemisalgoritmi sitten tietää, kumpi tasapainotila kannattaa valita, jotta pelitilanne näyttäisi järkevältä ihmispelaajan silmissä? Miten voidaan taata, että jos etsitään vain ensimmäistä tasapainotilaa, löydetään juuri sopiva tasapainotila? Miten vältetään tilanne, jossa agentit vuorolla 1 hyökkäävät kohti pelaajahahmoa, vuorolla 2 pakenevat poispäin ja vuorolla 3 hyökkäävät taas sen takia, että tasapainotilan etsimisalgoritmi sattuu löytämään tasapainotilat tuossa järjestyksessä? Näitä vaikeita kysymyksiä käsitellään luvussa 4.4.

3.5 Tietokonepelien agenttimalli ja arkkitehtuuri

Ei ole välittömästi selvää, että peliteoreettiset agentit ovat oikeita agentteja siinä merkityksessä, jonka Russell ja Norvig agentin määritelmässään antavat [RuN03]. He korostavat agentilta vaadittavaa autonomisuutta: agentti saa sensoreillaan tiedon maailmasta, päättää toiminnastaan jollain tavalla ja lopuksi yrittää toteuttaa aktuaattoreillaan valitsemansa toimen. Agenttiohjelman täytyy siis tehdä kuvaus syötteiden joukosta toimintojen joukkoon. Suurin ongelma peliteoreettisessa ratkaisussa agenttimallin kannalta on keskitetty päätöksentekojärjestelmä: tasapainotilan etsimisalgoritmi löytää yhden tasapainotilan, jonka mukaan kaikki agentit toimivat. Tällöin näyttää helposti siltä, että varsinaista agenttimallia ei tarvita, sillä yksittäis-

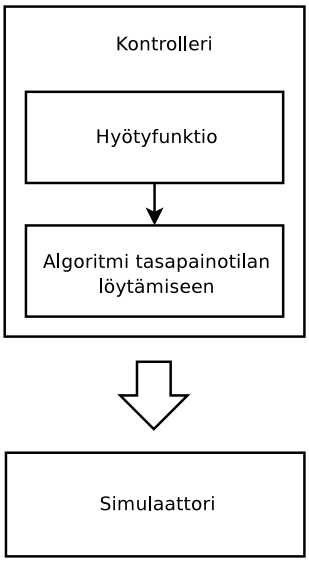
ten agenttien ei tarvitse toteuttaa päätöksentekokykyä. Tämä on kuitenkin illuusiota, sillä järjestelmän ulkopuolinen tarkkailija näkee vain älykkäästi toimivia agentteja. Tilannetta voi siis katsoa myös toisesta näkökulmasta: jos agentit neuvottelisivat keskenään mitä tekisivät, ne päätyisivät Nashin tasapainotilan määritelmän mukaan juuri tähän ratkaisuun. Kyse on siten samasta ilmiöstä, josta Alan Turing kuuluisassa paperissaan puhui: meidän ei tarvitse odottaa agenteilta itsenäistä älykästä toimintaa, vaan meille riittää, että se näyttää siltä [Tur50].

Funge esittelee kirjassaan tyypillisen tietokonepeleissä käytetyn arkkitehtuurimallin [Fun04]. Ajatuksena on, että peli jaetaan neljään eri osaan: pelitilaan, simulaattoriin, kontrollereihin ja renderöijään. Itse peli ja pelitapahtumat koostuvat näiden neljän komponenttityypin yhteistoiminnasta. Pelitila koostuu tietorakenteista, jotka pitävät sisällään kaiken tiedon pelimaailmasta ja sen eri toimijoista. Simulaattori sisältää tiedon säännöistä, joilla pelitilaa muutetaan; esimerkiksi pelimaailmassa vaikuttavat luonnonlait on määritetty siinä. Renderöijä huolehtii pelitilan esittämisestä pelaajalle. Jokaisella pelin hahmolla on kontrolleri, joka päättää toiminnon, jonka hahmo tietyssä tilanteessa valitsee. Kontrolleri voi olla esimerkiksi peliohjain, jota pelaaja käyttää, tai algoritmi, joka ohjaa vihollisagenttia.

Tämän arkkitehtuurimallin osista meitä kiinnostavat eniten kontrollerit ja simulaattori. Renderöijä on tietysti lopullisen pelin kannalta välttämätön, mutta agenttiryhmän päätöksenteon kannalta se ei ole mielenkiintoinen. Pelitilan merkitys tulee taas esille vain siinä mielessä, missä sitä tarvitaan hyötyfunktion termien arvojen määräämiseen eri toiminnoista saatua hyötyä vertailtaessa. Varsinaiseen päätöksentekologiikkaan se ei vaikuta.

Funge kirjoittaa, että kontrollerit voivat olla hierarkkisia. Tällä tarkoitetaan, että ylemmän tason kontrolleri voi antaa epätarkemman toimintapäätöksen, jonka pohjalta alemman tason kontrolleri valitsee yksityiskohtaisemman toiminnon. Esimerkki tästä on tilanne, jossa ylemmän tason kontrolleri valitsee kartalta pisteen, jonne hahmo pyrkii, ja alemman tason kontrolleri käyttää polunetsintäalgoritmia valitakseen reitin, jota pitkin kohteeseen päästään. Peliteoreettinen päätöksentekomalli toimii itse asiassa arkkitehtuurisesti juuri näin: hyötyfunktio on ylemmän tason kontrolleri, johon on koodattu hahmon tavoitteet. Itse tasapainotilan löytäminen on alemman tason operaatio, jolla etsitään toiminto, joka johtaa mahdollisimman lähelle hyötyfunktion määrittelemää päämäärää. Tämä on esitetty kuvassa 7.

Simulaattori on Fungen mukaan ainoa tietokonepelin komponentti, joka saa koskea pelitilaan [Fun04]. Se toimii siis agenttien toimintojen ja pelitilan välisenä puskurina.



Kuva 7: Hyötyfunktio ja tasapainotilan löytämisalgoritmi esitettynä hierarkkisessa kontrollerimallissa.

Simulaattorin tehtävänä on myös toteuttaa kaikki muut pelimaailmassa tapahtuvat asiat, jotka eivät liity agenttien toimintoihin: kappaleet voivat esimerkiksi pudota painovoiman vaikutuksesta, vaikka yksikään pelin agentti ei tekisi mitään.

Kun agenttiryhmä on tehnyt toimintapäätöksensä, siirtyy vastuu simulaattorille. Fungen arkkitehtuurimallin simulaattorissa on tapahtumajono, jonne sekä agenttien että ihmispelaajan kontrollereilta tulevat tapahtumat sijoitetaan [Fun04]. Simulaattori käy läpi jonoa ja tekee siellä olevien tapahtumien mukaan muutoksia pelimaailmaan. Jokainen muutos aiheuttaa kutsun renderöijälle, joka puolestaan esittää pelimaailman graafisessa muodossa. Agentit saavat tietonsa ihmispelaajan tekemistä toiminnoista ja oman tilansa muutoksista kysymällä sopivan ajan kuluttua pelitilalta ympäröivän maailman tilanteen. Tämän tiedon pohjalta agenttiryhmä pystyy tekemään jälleen uuden toimintapäätöksen. Vaikka agenttiryhmä valitsee toimintavektorinsa vain harvakseltaan, on agenttien liikkeen silti näytettävä oikeassa pelissä jouhevalta ja jatkuvalta. Renderöijän tehtävä on toteuttaa agenttien liike niin, että pelaajalle välittyy tunne jatkuvasti toimivista agenteista.

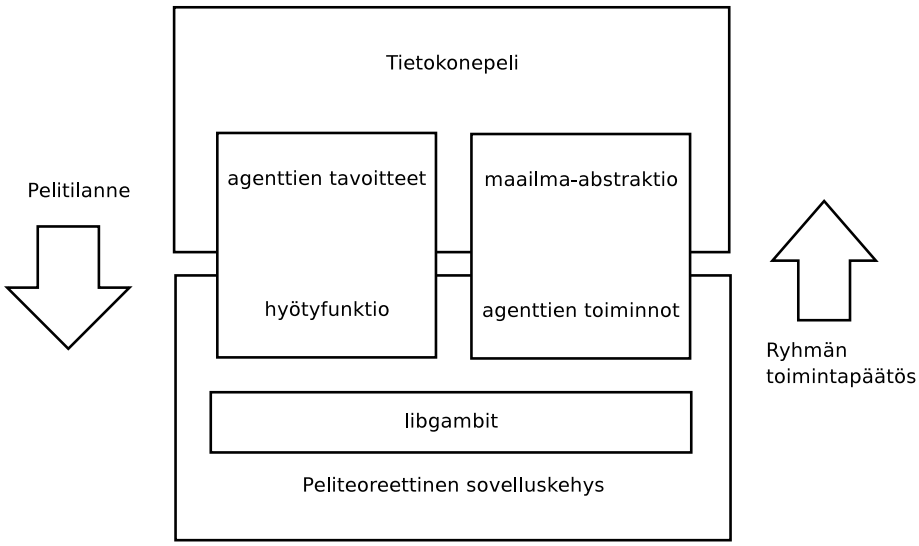
3.6 Testiohjelman rakenne ja toiminta

Jotta peliteoreettista agenttimallia voidaan käytännössä kokeilla tietokonepelissä, on rakennettava testiohjelma. Tutkielman ohessa tehtiinkin Java-ohjelma, jolla peliteorian keinoja testattiin käytännössä. Testiohjelma koostuu kahdesta eri osasta: ai-

boost-kirjastosta, joka tarjoaa rajapinnan tarvittaviin peliteoreettisiin menetelmiin, ja testipelistä, joka käyttää ai-boost-kirjastoa. Testipeli on toteutus Hyökkäyspelin tilanteesta, joka on esitetty kuvassa 2 sivulla 5.

Testipelissä pelialueena on äärellinen kartta, jossa on neliönmuotoisia ruutuja. Jokaisessa ruudussa on joko tyhjää maata, este, vihollisagentti tai pelaajahahmo. Jokaiselle vihollisagentille on määritetty joukko ruutuja, joihin se pystyy kulkemaan yhdessä kierroksessa. Normaalitilanteessa nämä ruudut ovat agentin ylä- ja alapuolella sekä molemmilla sivuilla, mutta esteiden vieressä oleminen pienentää vapaiden liikkumisruutujen joukkoa. Agentti voi vuorollaan valita siirtymisen johonkin vapaaseen ruutuun, johon se pystyy liikkumaan, tai paikallaanpysymisen. Testipelin maailmaan on määritetty ehdoksi, että agentti ei pysty siirtymään ruutuun, jossa on este tai pelaajahahmo. Agentit pystyvät kuitenkin kulkemaan ruutuihin, joissa on toinen agentti, jotta pelaajahahmon luokse liikkuminen onnistuisi yli neljän agentin peleissä.

Testipelissä ei ole toteutettu agenttien toiminnan pidempikestoista suunnittelua. Agentit eivät kuitenkaan ole reaktiivisia, sillä hyötyfunktiossa on määritetty muistillisia termejä. Pelaajahahmon toimintaa ei ole mallinnettu mitenkään, ja testisimulaatiossa pelaajahahmo pysyy paikallaan. Vihollisagentit eivät voi haavoittua tai tuhoutua, mutta ne käyttäytyvät niin kuin riski olisi olemassa.

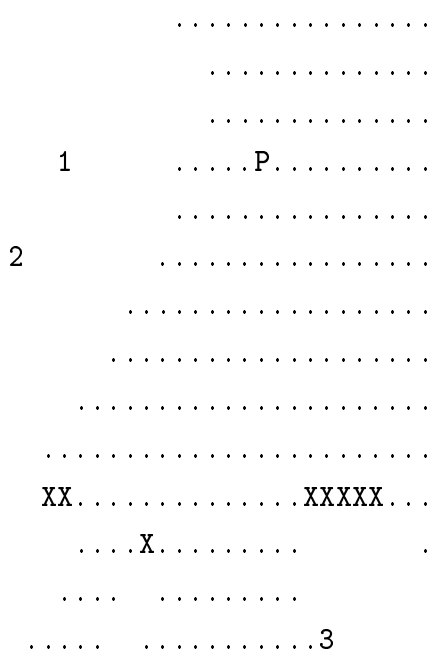


Kuva 8: Sovelluskehys.

Ai-boost-kirjasto on tehty käyttämään moottorinaan valmista peliteoreettista kirjastoa, jotta eri tasapainotilan löytämisalgoritmien vertailu olisi tarvittaessa helppoa ja algoritmit varmasti toimisivat oikein. Kirjasto on McKelveyn ja muiden kehittä-

mä libgambit, joka on lisensoitu vapaan lähdekoodin GNU General Public Licensen alle [MMT06]. Valmiista algoritmikirjastosta huolimatta soveltuviin algoritmeihin ja niiden suorituskykyyn palataan vielä kappaleessa 4.

Sekä ai-boost että testipeli julkaistaan myös GNU General Public Licensen alla. Yhdessä ai-boost ja libgambit muodostavat sovelluskehyksen, jota voidaan periaatteessa käyttää tietokonepeleissä laajemmaltikin. Testipeli on hyvin yksinkertainen, tekstigrafiikalla toteutettu simulaatio, jossa ihmispelaajan ainoaksi huoleksi jää alkutilanteen antaminen. Kuva 8 esittää testiohjelman rakennetta.



Kuva 9: Pelaajahahmon näkökenttä kuvan 2 tilanteessa. Tyhjäksi jätetyt ruudut ovat pelaajalle näkymättömiä.

Ai-boostin rakenne on tehty ajatellen uusiokäyttöä. Se on kirjoitettu Javalla ja rakennettu niin, että pelikehittäjän on kirjastoa käyttääkseen toteutettava tietty rajapinta pelissään. Pelikehittäjän on tehtävä pelimaailmastaan abstraktio, joka tarjoaa metodit kaikkiin hyötyfunktiossa tarvittavien tietojen löytämiseen. Tällaisia tietoja ovat esimerkiksi metodit, joilla lasketaan vihollisagenttien keskinäiset etäisyydet, etäisyys pelaajahahmoon sekä tieto vihollisagentin olemisesta esteen takana piilossa. Samoin pelimaailmassa on määritettävä tiedot vihollisagenttien sallituista liikkeistä sekä pelitiloista, joihin ne johtavat. Liikkumissäännöt ovat myös maailmakohtaisia, eli pelinkehittäjän on päätettävä, voivatko vihollisagentit esimerkiksi sijaita samassa

ruudussa tai lentää kolmiulotteisessa avaruudessa. Pelinkehittäjän on myös rakennettava jokaiselle vihollisagentille oma hyötyfunktio, joka vastaa pelimaailman tarpeita. Hyötyfunktion luomisesta kerrotaan luvussa 5. Teknisesti tämä rajapinta on toteutettu rakentamalla ai-boostiin Interface-luokkia, jotka peliohjelman täytyy toteuttaa, jotta kirjasto toimisi. Testipeli toimii juuri toteuttamalla nämä rajapinnat.

Testipeliä varten on toteutettu joitain pelien tekoälyssä käytettäviä tekniikoita, kuten näkyvissäolon tutkiminen ja A*-polunetsintäalgoritmi. Kuvassa 9 esitetään pelaajahahmon näkökenttä sivulla 5 olevan kuvan 2 mukaisessa Hyökkäyspelin tilanteessa. Näkökentän epäjohdonmukaisuudet, kuten joidenkin esteiden näkyvyys, johtuvat käytetyn säteenseurausalgoritmin (ray-tracing algorithm) toteutuksen yksityiskohdista.

```
.............................
......++++++++++++...........
......+......XX...1...........
......+.....X.....P...........
.........XXXX.....3...........
...+..XXXXX2......+...........
...+...XX+++......+...........
...+++++++........+...........
..................+...........
....X.............+...........
....XXX...........+..XXXXX...
...........X......++++XXXX...
.....................++X.....
.....................+......
```

Kuva 10: Kuvan 2 alkutilanteesta lähtenyt testipeli 25 pelivuoron jälkeen.

Testipeli pyrkii mukailemaan Fungen esittelemää tietokonepeliarkkitehtuuria, josta kerrottiin luvussa 3.5. Jokaisella testipelin pelivuorolla lasketaan ensimmäiseksi vihollisagenttien saamat hyötyarvot kaikista mahdollisista siirtojen yhdistelmistä. Hyötyarvot kootaan matriisiin, josta seuraavaksi etsitään Nashin tasapainotila. Tasapainotilan löydyttyä muutetaan tasapainotilan toteuttava strategiavektori agenttien toimintavalinnoiksi: jos jonkin agentin strategia on sekastrategia, on pelattava toiminto arvottava sekastrategiassa olevien toimintojen todennäköisyysjakaumasta.

Lopullinen agenttien toimintapäätös talletetaan testipelin muistiin ja testipelin simulaattori päivittää pelimaailman agenttien liikkeen mukaiseksi. Tämän jälkeen aloitetaan seuraava kierros. Kuvassa 10 on eräs kuvassa 2 esitellystä alkutilanteesta syntynyt pelitilanne kahdenkymmenenviiden pelivuoron jälkeen. Kuvassa +-merkki esittää ruutuja, joiden kautta agentti on kulkenut. Agentit 1 ja 3 ovat kulkeneet pelaajahahmon viereen, mutta agentti 2 on edennyt hitaammin. Testipelin kulkua eri vaiheissa ja syitä erilaisille agenttien tekemille toimintapäätöksille selvitetään luvussa 5.4.

4 Nashin tasapainotila

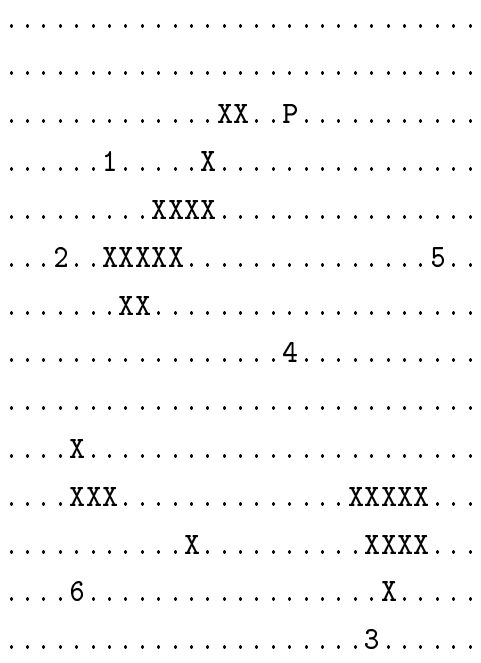
Tässä luvussa käsitellään Nashin tasapainotilan löytämistä normaalimuotoisesta peliteoreettisesta pelistä. Ensiksi selvitetään, miten hakuavaruuden kokoa rajoitetaan, ja sen jälkeen tarkastellaan yksinkertaista etsimismenetelmää tasapainotilojen löytämiseen. Lopuksi tarkastellaan epälineaaristen yhtälöryhmien ratkaisemista numeerisin menetelmin.

4.1 Aika- ja tilavaativuus

Papadimitriou ja Roughgarden pohtivat tasapainotilojen löytämisen vaikeutta [PaR05]. He mainitsevat, että kahden pelaajan pelin laskennallinen vaativuus on edelleen mysteeri, mutta useamman pelaajan pelin aikavaativuudesta tiedetään vielä vähemmän. Papadimitriou ja Roughgarden kertovat, että tilavaativuus sen sijaan on selvillä. Valitettavasti sekään ei näytä hyvältä: jos pelissä on n pelaajaa, ja jokainen valitsee toimintonsa k:n toiminnon joukosta, tarvitaan hyötymatriisiin yhteensä nk^n numeroa. Tämä seuraa siitä, että pelissä on yhteensä k^n erilaista lopputulosta, ja jokainen pelaaja saa jokaisesta lopputuloksesta numerona ilmaistavan hyödyn. Jos pelissä olisi esimerkiksi kuusi pelaajaa, joista jokaisella olisi viisi mahdollista toimintoa, ja hyöty ilmoitettaisiin neljä tavua pitkänä kokonaislukuna, veisi hyötymatriisi tilaa yhteensä $6 * 5^6 * 4 = 375000$ tavua, eli noin 366 kilotavua. Jos äskeisessä esimerkissä pelaajia olisikin 8, veisi hyötymatriisi tilaa jo lähes 12 megatavua.

Conitzier ja Sandholm kirjoittavat, että vaikka Nashin tasapainotilan löytämisen aikavaativuutta ei tunneta, monia rajoitetumpia tuloksia on saatu [CoS02]. Heidän mukaansa kahden pelaajan nollasummapelit pystytään ratkaisemaan polynomisessa ajassa. Toisaalta taas normaalimuotoisessa pelissä olevien Nashin tasapainotilojen, joilla on tiettyjä ominaisuuksia, olemassaolon selvittäminen on NP-vaikea ongelma jopa symmetrisissä kahden pelaajan peleissä. Esimerkiksi Pareto-optimaalisen Nashin tasapainotilan olemassaolon selvittäminen on NP-vaikea ongelma, kuten myös sellaisen Nashin tasapainotilan, josta saatava sosiaalinen hyvinvointi ylittää jonkin mielivaltaisesti asetettavan rajan. Jopa useamman kuin yhden tasapainotilan olemassaolon selvittäminen on NP-vaikea ongelma.

Jotta yhden tasapainotilan löytämisen aikavaativuus olisi Hyökkäyspelin tapauksessa selkeämpi, testejä tehtiin sekä kolmen agentin että kuuden agentin pelissä käyttäen tasapainotilan löytämiseen Porterin ja muiden algoritmia, joka esitellään luvussa 4.3. Algoritmin toteutus oli libgambit-kirjaston versiossa `gambit-0.2006.01.20`,



Kuva 11: Tasapainotilan löytämisajan mittauksessa käytetyn kuuden pelaajan pelin alkutilanne.

ja sitä muokattiin lopettamaan tasapainotilojen etsintä ensimmäisen tasapainotilan löytämiseen [MMT06]. Kolmen agentin pelin alkutilanne esitettiin kuvassa 2 sivulla 5, ja kuuden agentin pelin alkutilanne on kuvassa 11. Hyötyfunktiona käytettiin myös luvun 5.4 testeissä käytettyä funktiota, joka kuvataan tarkemmin luvussa 5.2.

Mittaus tehtiin mittaamalla tasapainotilan löytävän algoritmin kutsun ja palautusarvon saamisen välinen aika: muita Hyökkäyspelin aikaa vieviä kohtia, kuten A*-algoritmin ajamista tai hyötymatriisin täyttämistä, ei mitattu. Mittaukset tehtiin 2.1 GHz G5-prosessorilla varustetussa Macintosh-tietokoneessa.

Algoritmin nopeutta testattiin sekä kolmen että kuuden agentin tapauksessa ajamalla 25 kierrosta Hyökkäyspeliä kymmenen kertaa. Samaa peliä toistettiin, jotta mahdollisien satunnaistekijöiden merkitys tuloksissa vähenisi. Jokaisella pelikierroksella etsittiin tasapainotila, joten datapisteitä kertyi molemmista peleistä 250. Koska agentit tekivät eri pelin vaiheissa erilaisia asioita, vaihtelivat suoritusajat melko paljon. Jos esimerkiksi monta agenttia oli jo ehtinyt pelaajahahmon viereen, hallitsi niillä paikallaanpysyminen vahvasti kaikkia muita strategioita, jolloin tasapainotilojen löytyminen nopeutui merkittävästi. Samoin toisinaan agentit kulkivat esteiden vierustoja, jolloin mahdollisten toimintovaihtoehtojen määrä laski, koska esteiden si-

sälle ei voi Hyökkäyspelin sääntöjen mukaan liikkua. Näiden syiden takia tulokset ovat vain suuntaa antavia, mutta niitä tutkimalla saa käsityksen Hyökkäyspelin käytännön aikavaativuuden suuruusluokasta.

Agenttien lkm	Datapisteet	Pienin arvo	Suurin arvo	Keskiarvo	Mediaani
3	250	42 ms	225 ms	71,43 ms	62 ms
6	250	97 ms	2760 ms	254,07 ms	146 ms

Taulukko 4: Hyökkäyspelin ensimmäisen tasapainotilan löytämisajan kokeelliset mittaustulokset

Mitä nämä mittaustulokset kertovat? Kolmen pelaajan pelissä tasapainotila löytyi keskimäärin noin 72 millisekunnin aikana, mikä voi olla sallitun rajoissa jopa sellaisissa tietokonepeleissä, joissa tekoäly ei saa käyttää kuin murto-osan prosessoritehosta. Sen sijaan kuuden agentin pelissä pahimman tapauksen viemä aika oli jo melkein kolme sekuntia, mikä on erityisesti toimintapeleissä liian kauan. Silti kuudenkin pelaajan pelissä mediaani oli vain 146 millisekuntia, eli puolet kierroksista oli selvästi ratkaistavissa melko lyhyessä ajassa. Kummassakaan pelissä ei jouduttu kertaakaan pelaamaan sekastrategiaa, eli tasapainotila löytyi aina tuista, joiden koko oli 1. Eniten aikaa veivät sekä kolmen että kuuden agentin peleissä ensimmäiset vuorot, joiden aikana ilmeisesti hakuavaruutta ei pystytty kunnolla rajaamaan, vaan kokoa 1 olevien tukien läpikäyminen palautti itse asiassa useamman tasapainotilaksi kelpaavan ratkaisun.

4.2 Hakuavaruuden rajaaminen

Hyötymatriisi, jolla normaalimuotoinen peli esitetään, kasvaa eksponentiaalisesti suhteessa pelaajien määrään. Kahden pelaajan peli voidaan esittää kaksiulotteisena taulukkona, jonka sivujen pituudet ovat pelaajan valittavissa olevien toimintojen määrät, mutta esimerkiksi kolmen pelaajan pelin hyötymatriisi on jo kuutiollinen. Koska Nashin tasapainotilaa etsivät algoritmit käyttävät hyötymatriisia hakuavaruutenaan, ensimmäinen vaihe tasapainotilan etsimisessä on rajata hakuavaruutta mahdollisimman paljon. Tämä tehdään poistamalla hakuavaruudesta hallitut strategiat (dominated strategies). Kuten luvussa 2.1 mainittiin, strategia s on heikosti hallittu, jos on olemassa strategia s', joka on aina vähintään yhtä hyvä ja ainakin kerran parempi kuin s kaikkia muiden pelaajien strategioita vastaan. Ajatuksena on siis, että strategiaa s ei kannata missään olosuhteessa pelata, koska strategia s' on yksinkertaisesti

parempi.

Iteroitu hallittujen strategioiden poistaminen (iterated elimination of dominated strategies, IEDS) on Duttan mukaan yksinkertainen algoritmi, jolla voidaan poistaa kaikki strategiat, joita ei koskaan kannata pelata [Dut99]. Algoritmi etenee vaiheittain. Pelaajan i kaikkien strategioiden joukkoa merkitään S^i. Algoritmin ensimmäisellä kierroksella (I) pelaaja i luo hallittujen strategioiden joukon $D^i(I)$.

$$D^i(I) = s_i \in S^i, s_i \text{ on hallittu strategia}$$

Nyt jäljellä oleva mahdollisten strategioiden joukko on $S^i - D^i(I)$. Koska kaikki muutkin pelaajat ovat vähentäneet omat hallitut strategiansa mahdollisesti pelattavien strategioidensa joukosta, voi olla, että osa strategioista, jotka eivät vielä olleet hallittuja, on tullut hallituiksi. Tämän takia pelaaja i:n on kierroksella II muodostettava uusi hallittujen strategioiden joukko $D^i(II)$, joka sisältää kaikki kierroksella I tai II eliminoidut strategiat. Muut pelaajat vähentävät samalla periaatteella hallitut strategiansa strategiajoukostaan. Tätä prosessia jatketaan, kunnes kukaan pelaajista ei enää löydä yhtään hallittua strategiaa.

Iteroidussa hallittujen strategioiden poistamisessa on Duttan mukaan joitain ongelmia [Dut99]. Näistä mielenkiintoisin on, että poistamisjärjestyksellä on väliä. Jos strategioita poistetaan pelaajajärjestyksessä, lopputulos voi olla erilainen, jos pelaajien järjestystä muutetaan. Esimerkiksi osa strategioista, jotka jossain toisessa pelaajajärjestyksessä poistettaisiin, saattaa jäädä jäljelle. Tämä ongelma ratkeaa, jos vaaditaan, että poistettavien strategioiden on oltava vahvasti hallittuja.

Duttan mukaan strategia s on vahvasti hallittu, jos ja vain jos on olemassa strategia s', joka on aina parempi kuin strategia s kaikkia muiden pelaajien strategioita vastaan [Dut99]. Jos vahvasti hallittuja strategioita poistetaan iteratiivisesti, ei poistojärjestyksellä ole väliä. Vahvasti hallittujen strategioiden poistamisen ongelmana on, että joissain peleissä heikosti hallittuja strategioita poistamalla saadaan etsintäavaruus pienemmäksi.

Cheng ja muut mainitsevat, että myös pelin symmetrisyys auttaa pienentämään etsintäavaruutta [CRV04]. Dutta kertoo, että peli on symmetrinen, kun strategiajoukko on sama kaikille pelaajille ja samat toiminnot tuottavat tarkalleen saman hyödyn. Täsmällisemmin sanottuna: jos pelaaja i pelaa strategian s_i kun muut pelaavat s'_{-i}, pelaajan i saama hyöty $\pi(s_i, s'_{-i})$ ei muutu sen mukaan, mikä i:n arvo on. Pelaajat ovat siis keskenään vaihdettavissa ilman, että peli muuttuu. Vangin ongelma, joka esiteltiin taulukossa 1 sivulla 2, on esimerkki symmetrisestä pelistä. Chengin

ja muiden mukaan jokaisessa symmetrisessä pelissä on symmetrinen tasapainotila, jossa kaikki pelaajat pelaavat tarkalleen saman strategian. Tämän tasapainotilan löytämisen etsintäavaruus on tavallista pienempi, sillä ainoastaan ne pelitilat tarvitsee ottaa huomioon, joissa kaikilla pelaajilla on sama strategia. Hyökkäyspelissä symmetristen pelien erikoisominaisuuksista ei ole paljoa hyötyä. Jos agentit ovat reaktiivisia, niiden sisäinen tila ja pelattavissa olevat strategiat ovat identtiset. Kuitenkin ainakin johonkin suuntaan liikkumisesta saatava hyöty on agenteilla pakosti erilainen, jos agentit eivät ole tarkalleen samassa kartan pisteessä.

4.3 Tasapainotilan löytäminen

Nashin mukaan jokaisesta äärellisestä pelistä löytyy aina vähintään yksi tasapainotila [Nas50]. Hyökkäyspeli on aina äärellinen, sillä pelissä on mukana vain rajoitettu määrä pelaajia, ja jokainen pelaaja joutuu valitsemaan toimintonsa äärellisestä joukosta toimintoja. Siksi on varmaa, että kaikissa kuviteltavissa olevissa Hyökkäyspelin tilanteissa on löydettävissä Nashin tasapainotila.

Dutta kirjoittaa, että jos iteroitu hallittujen strategioiden poistaminen jättää jäljelle kullekin pelaajalle vain yhden strategian, sanotaan, että peli on *ratkaistavissa hallitsemisella* (dominance solvable) [Dut99]. Tällöin tasapainotila on triviaali: jokaisen pelaajan täytyy vain pelata strategia, joka on jäänyt jäljelle.

McKelvey ja McLennan ovat tehneet kartoituksen olemassa olevista algoritmeista Nashin tasapainotilan löytämiseen [McM96]. He osoittavat, että normaali tapa ratkaista Nashin tasapainotila on määritellä se niin, että se on jonkin tunnetun matemaattisen ongelman ilmentymä. Tämän jälkeen on suhteellisen suoraviivaista ratkaista ongelma siihen normaalisti käytettävällä menetelmällä. McKelvey ja McLennan määrittelevät artikkelissaan Nashin tasapainotilan mm. vastaavuuden kiintopisteenä (fixed point of a correspondence), funktion kiintopisteenä (fixed point of a function), ratkaisuna epälineaariseen komplementtiongelmaan (non-linear complementary problem), puolialgebraalisena joukkona (semi-algebraic set), vakaan pisteen ongelmana (stationary point problem) ja polytoopille määritellyn funktion minimikohtana (minimum of a function on a polytope). McKelvey ja McLennan kuitenkin huomauttavat, että monesti matemaattisten ongelmien tavalliset ratkaisumenetelmät eivät sellaisenaan sovi kovin hyvin Nashin tasapainotilojen löytämiseen. Tämä johtuu siitä, että tasapainotiloilla on ominaisuuksia, joita voi käyttää nopeuttamaan niiden löytämistä.

McKelvey ja McLennan toteavat myös, että monet menetelmät, jotka soveltuvat yksittäisten tasapainotilojen löytämiseen, eivät kelpaa tilanteisiin, joissa pitää löytää kaikki tasapainotilat. Tavallisesti syynä kaikkien tasapainotilojen etsimiseen on, että halutaan löytää tasapainotila, jolla on tiettyjä ominaisuuksia. Yksi monesti toivottu ominaisuus on mahdollisimman suuri pelaajien yhteenlaskettu hyöty. Silti on mahdollista etsiä sopivaa tasapainotilaa yhden tasapainotilan etsintäalgoritmilla käyttämällä sopivaa algoritmin aloituskohtaa tai parametria.

Porter ja muut ovat tutkineet tilannetta, jossa on tarpeen löytää yksi tasapainotila mahdollisimman nopeasti [PNS04]. He lähtevät liikkeelle tavallisesta etsinnästä yhdistettynä voimakkaaseen heuristiikkaan. Kappaleessa 2.1 kerrottiin, että pelaajan tuki on niiden toimintojen joukko, jotka pelaaja pelaa positiivisella todennäköisyydellä. Porterin ja muiden käyttämä heuristiikka on aloittaa etsintä kaikkein pienimmistä mahdollisista tuista, koska useissa tutkituissa peleissä on osoittautunut, että niissä ollut ainakin yksi hyvin pieni tuki, joka sisältää tasapainotilan. Esimerkki pienestä tuesta on peli, jossa jokainen pelaajaa pelaa puhtaan strategian: tällöin tuen koko on jokaisella pelaajalla 1.

Porter ja muut esittelevät eri algoritmit tasapainotilan löytämiseksi kahden pelaajan ja monen pelaajan peleissä. Koska Hyökkäyspelissä agentteja on tyypillisesti kolme tai enemmän, tässä käsitellään vain yleisempi monen pelaajan pelistä tasapainotilan löytävä algoritmi. Algoritmi on kuvattu seuraavissa listauksissa, jotka käyttävät pseudokoodinotaatiota Cormenin ja muiden *Introduction to Algorithms* -kirjan toisesta painoksesta [CLR01]. Listaukset eivät vastaa aivan täysin Porterin ja muiden algoritmia, sillä koodiin on tehty joitain lavennuksia ja lisätty välivaiheita, jotta sen toiminnan selittäminen olisi helpompaa.

```
ETSIMISALGORITMI(X)
1  for  each x = (x_1, ..., x_n) in X
2  do ∀i : S_i ← NIL
3     ∀i : D_i ← S_i ⊂ A_i : |S_i| = x_i
4     p ← REKURSIIVINENPERÄYTYVÄHAKU(S, D, 1)
5     if p ≠ NIL
6        then return p
```

ETSIMISALGORITMI on perustaltaan yksinkertainen. Sen parametri X sisältää kaikki mahdolliset kombinaatiot $x = (x_1, \ldots, x_n)$ agenttien tukien koista. Tukien koko-

kombinaatiot on järjestetty ensisijaisesti kokojen summan $\sum_i x_i$ mukaan ja toissijaisesti suurimman kokojen erotuksen $max_{i,j}(x_i - x_j)$ mukaan nousevaan järjestykseen. Tällä tavalla ensin käsitellään tilanne, jossa jokaisen agentin tuessa on vain yksi toiminto, sen jälkeen tilanne, jossa yhdellä agentilla on kaksi toimintoa ja kaikilla muilla yksi toiminto, ja niin edelleen. Porterin ja muiden mukaan tämä on heuristiikan ydin: tasapainotilat löytyvät todennäköisemmin pienistä tuista.

Algoritmissa luodaan myös jokaiselle agentille i tukien joukko S_i ja piirien (domain) joukko D_i, jotka alustetaan uudestaan jokaisella rivin 1 for-silmukan kierroksella. Jokaisen agentin tukien joukko alustetaan etsimisalgoritmin rivillä 2 tyhjäksi, ja piirien joukko saa rivillä 3 alkuarvokseen kaikki agentille saatavissa olevat tuet, jotka ovat kokoa x_i. Piirijoukon merkitys myöhemmässä vaiheessa on pitää kirjaa vielä käyttämättömistä tuista, jotka ovat oikeaa kokoa, kun taas tukijoukon tuet on määritelty varmasti tasapainotilassa tarvittaviksi. A_i on agentti i:n kaikkien mahdollisten toimintojen joukko. Esimerkiksi ensimmäisellä for-silmukan kierroksella jokaisen agentin tukijoukko on tyhjä ja piirijoukossa ovat kaikki tuet, joiden koko on yksi: toisin sanoen agentin kaikki puhtaat strategiat.

Tukia lähdetään tutkimaan rekursiivisella peräytyvällä (backtracking) haulla, joka saa parametreikseen kaikkien agenttien tukijoukot, piirijoukot ja seuraavan muodostettavan tuen indeksin, joka on samalla seuraavan käsiteltävän agentin indeksi.

```
RekursiivinenPeräytyväHaku(S, D, i)
 1  if i = n + 1
 2    then p ← Tasapainotila(S)
 3         if p ≠ nil
 4           then return p
 5           else  return nil
 6    else
 7         for  each d_i in D
 8         do S_i ← d_i
 9            D_i ← D_i − {d_i}
10            D ← IteroituPoisto(({S_1}, ..., {S_i}, {D_{i+1}}, ..., {D_n}))
11            if D ≠ nil
12              then p ← RekursiivinenPeräytyväHaku(S, D, i + 1)
13                   if p ≠ nil
14                     then return p
15         return nil
```

RekursiivinenPeräytyväHaku-algoritmissa on kaksi päähaaraa: if-haara ja siihen liittyvä else-haara. If-haaraan mennään, kun algoritmi on päässyt tukien muodostamisessaan rekursion pohjalle, eli kaikki on valmista itse tasapainotilan löytämiseen. Else-haaraan mennään kaikissa muissa tapauksissa. Else-haarassa muodostetaan agentille i tukijoukko siirtämällä sinne yksi kerrallaan tukia piirijoukosta. Jokaisen siirretyn tuen jälkeen kutsutaan IteroituPoisto-algoritmia, joka poistaa sille annettavista agenttien tuista kaikki vahvasti hallitut strategiat käyttäen luvussa 4.2 käsiteltyä iteroitua hallittujen strategioiden poistamisalgoritmia.

IteroituPoisto-algoritmi on muuten aivan kuten aiemmin esitelty algoritmi, mutta se palauttaa virhearvon, mikäli jokin tuista on tyhjä. Normaalisti iteroidussa hallittujen strategioiden poistossa ei tällaista tilannetta voi tapahtua, sillä jos agentilla on jäljellä enää yksi strategia, se ei voi määritelmänsä mukaan olla minkään toisen strategian hallitsema. Jos IteroituPoisto palauttaa virhearvon, eivät rekursiivisen peräytyvän haun jo muodostetut tuet ole Porterin ja muiden mukaan järkeviä, ja tällöin rekursio sen haaran osalta lopetetaan.

Iteroitu vahvasti hallittujen strategioiden poistaminen ottaa parametrikseen niiden agenttien tukijoukot, joilla tukijoukko on jo muodostettu, eli agenttien $1 \ldots i$. Muilta agenteilta otetaan mukaan täydelliset piirijoukot. Tämän vaiheittaisen piirien pienentämisen ajatuksena on, että jokaisessa vaiheessa hallitut strategiat pystytään pudottamaan pois. Porterin ja muiden mukaan pienten tukien käyttäminen iteroidussa poistossa on järkevää, sillä kun esimerkiksi agentilla 1 on vain yksi toiminto tuessaan, on hyvin todennäköistä, että agentin 2 strategioista erottuu yksi, joka hallitsee kaikkia muita agentin 2 strategioita agentin 1 ainoan strategian suhteen. Itse asiassa tällöin ainoa tilanne, jossa yksi agentin 2 strategia ei hallitse kaikkia muita agentin 1 suhteen on, kun agentti 2 saa useammasta strategiasta saman hyödyn.

Kun ensimmäisten agenttien tukijoukkoa ja muiden agenttien piirijoukkoa on karsittu iteroidussa poistossa, syötetään kaikkien agenttien tuki- ja piirijoukot uudestaan rekursiiviseen peräytyvään hakuun, ja nostetaan käsiteltävän tukijoukon (ja agentin) indeksiä yhdellä.

Rekursioon lähdetään pahimmassa tapauksessa niin monta kertaa, kuin mitä tukijoukkoon pystytään siirtämään erilaisia tukia piirijoukosta jokaisen agentin kohdalla. Haarautuminen on siis pienien tukien kohdalla melko pientä, mutta kun piirijoukossa olevien tukien koko kasvaa, kasvavat erilaiset mahdollisuudet tukien strategiakombinaatioksi nopeasti binomikertoimen mukaisesti.

Lopulta, kun rekursion syvyys on yhtä syvä kuin agenttien lukumäärä, päästään algoritmin if-haaraan. If-haaraan lähdetään, sillä ehto $i = n+1$ täyttyy, koska agenttiindeksi i on rekursiossa ohittanut agenttien määrän n. Tällöin jokaisen agentin kohdalla tukijoukkoon on saatu joitain tukia, ja varsinainen tasapainotilojen etsiminen voi alkaa.

Tasapainotilat etsitään yksinkertaisesti muodostamalla jäljelle jääneistä tuista yhtälöryhmä, kuten tehtiin kahden agentin sekastrategian etsimisalgoritmissa luvussa 2.1. Yhtälöryhmän ratkaisu on strategiaprofiili p, joka olisi tasapainotila, jos peli itsessään olisi vain tuen kokoinen. TASAPAINOTILA-algoritmi etsii tämän strategiaprofiilin ja tarkistaa, onko se koko pelin Nashin tasapainotila.

```
TASAPAINOTILA(S)
1  p ← yhtälöryhmän ratkaisu
2  if  p täyttää tasapainotilan ehdot
3     then return p
4     else  return NIL
```

McKelvey ja McLennan toteavat, että Nashin tasapainotila voidaan määritellä agenttien tukien avulla [McM96]. Peli on tasapainotilassa, jos kaikilla agentin tuessa olevalla toiminnolla on sama hyödyn odotusarvo ja jos kaikkien agentin toimintojen pelaamistodennäköisyyksien summa on yksi. Myöskään yhdenkään agentin tuen ulkopuolella olevan strategian hyödyn odotusarvo ei saa olla suurempi kuin tuessa olevien strategioiden. Tuen määritelmän mukaan siinä olevien strategioiden pelaamistodennäköisyyden on oltava positiivinen, ja jotta tuki sisältäisi koko tasapainotilan, on jokaisen tuen ulkopuolella olevan strategian pelaamistodennäköisyyden oltava 0. Porter ja muut ilmaisevat samat ehdot algoritmisessa muodossa kuvassa 12 [PNS04].

Jos on olemassa strategiaprofiilijoukko $p = (p_1, \ldots, p_n)$ ja hyötyfunktion arvojoukko $v = (v_1, \ldots, v_n)$, joilla annetut ehdot täyttyvät, on tukien joukossa $S = (S_1, \ldots, S_n)$ yksiselitteisesti määritelty tasapainotila. Ongelmaksi muodostuu, että sekastrategian todennäköisyysprofiilin laskeminen on Porterin ja muiden mukaan lineaarista vain kahden pelaajan pelissä: jos pelaajien määrä $n > 2$, muodostuu ehdoista epälineaarinen yhtälöjoukko, jota ei useinkaan pystytä ratkaisemaan analyyttisesti [PNS04]. Epälineaaristen yhtälöryhmien ratkaisemiseen on kuitenkin olemassa suuri määrä erilaisia numeerisia optimointialgoritmeja. Eräs ratkaisualgoritmi esitellään luvussa 4.5.

$$\forall i \in N, s_i \in S_i \quad : \quad \sum_{s_i \in S_{-i}} p(s_{-i})\pi_i(s_i, s_{-i}) = v_i \tag{1}$$

$$\forall i \in N, s_i \notin S_i \quad : \quad \sum_{s_i \in S_{-i}} p(s_{-i})\pi_i(s_i, s_{-i}) \leq v_i \tag{2}$$

$$\forall i \in N \quad : \quad \sum_{s_i \in S_i} p_i(s_i) = 1 \tag{3}$$

$$\forall i \in N, s_i \in S_i \quad : \quad p_i(s_i) \geq 0 \tag{4}$$

$$\forall i \in N, s_i \notin S_i \quad : \quad p_i(s_i) = 0 \tag{5}$$

Kuva 12: Tuessa olevan tasapainotilan ehdot.

Miten yhtälöryhmä muodostetaan useamman pelaajan pelissä? Esimerkki selventää asiaa. McKelvey ja McLennan käyttävät taulukossa 5 esiteltyä peliä esimerkkinä monen pelaajan pelistä. Porterin ja muiden algoritmia käyttäen taulukon tilanteeseen voidaan päästä, kun Tasapainotila-algoritmia kutsutaan tukijoukoilla, joissa olevat toiminnot ovat ne, jotka taulukossa esitellään. Merkintätapa taulukossa eroaa hieman aiemmasta: koska pelaajia on kolme, näytetään kolmannen pelaajan kaksi toimintoa omina taulukkoinaan. Pelaajien toimintojen notaatio on muutettu siten, että pelaajan 1 toiminto 1 on taulukossa P_{11}, pelaajan 1 toiminto 2 on P_{12} ja niin edelleen. Jokaisessa taulukon solussa on kolmen numeron joukko. Numeroista ensimmäinen tarkoittaa pelaaja 1:n saamaa hyötyä, toinen pelaaja 2:n saamaa hyötyä ja kolmas pelaaja 3:n saamaa hyötyä.

P_{31}	P_{21}	P_{22}
P_{11}	(9,8,12)	(0,0,0)
P_{12}	(0,0,0)	(9,8,2)

P_{32}	P_{21}	P_{22}
P_{11}	(0,0,0)	(3,4,6)
P_{12}	(3,4,4)	(0,0,0)

Taulukko 5: Monen pelaajan peli.

Ajatus kolmen pelaajan pelin ratkaisemisessa on sama kuin kahden pelaajan pelin ratkaisemisessa: jotta pelaaja i:n ei kannattaisi vaihtaa toimintaansa, täytyy pelaajan i saada kaikista pelaamistaan toiminnoista tarkalleen sama hyödyn odotusarvo. Toisin sanoen taulukon 5 tilanteessa, jossa jokaisella pelaajalla on kaksi eri toimintoa valittavanaan, täytyy toiminnoista P_{i1} ja P_{i2} saatavan hyödyn odotusarvon olla sama.

Taulukosta nähdään, että pelaaja 1 pelaa toiminnon P_{11} todennäköisyydellä p ja toiminnon P_{12} todennäköisyydellä $1-p$. Symmetrisesti pelaaja 2 pelaa toiminnon P_{21} todennäköisyydellä q ja toiminnon P_{22} todennäköisyydellä $1-q$, ja pelaaja 3 pelaa toiminnon P_{31} todennäköisyydellä r ja toiminnon P_{32} todennäköisyydellä $1-r$. Toisin sanoen sekastrategiaprofiilijoukko $p = (p_1, p_2, p_3)$ koostuu strategiaprofiileista $p_1 = (p, 1-p)$, $p_2 = (q, 1-q)$, $p_3 = (r, 1-r)$.

Taulukosta 5 nähdään, että pelaaja 1 saa toiminnon P_{11} pelaamisesta hyödyn odotusarvoksi $9qr+0(1-q)r+0q(1-r)+3(1-q)(1-r) = 9qr+3(1-q)(1-r)$. Toiminnon P_{12} pelaamisesta saatava hyöty on $0qr + 9(1-q)r + 3q(1-r) + 0(1-q)(1-r) = 9(1-q)r + 3q(1-r)$. Jotta hyödyt olisivat samat, saadaan pelaajalle 1 seuraava yhtälö:

$$9qr + 3(1-q)(1-r) = 3q(1-r) + 9(1-q)r$$

Kun sama päättely toistetaan myös muille agenteille, saadaan kaksi muuta yhtälöä:

$$\begin{aligned} 8pr + 4(1-p)(1-r) &= 4p(1-r) + 8(1-p)r \\ 12pq + 2(1-p)(1-q) &= 6p(1-q) + 4(1-p)q \end{aligned}$$

McKelvey ja McLennan muuntavat samat yhtälöt normaalimuotoon:

$$\begin{aligned} (6q-3)(4r-1) &= 0 \\ (12r-4)(2p-1) &= 0 \\ (8p-2)(3q-1) &= 0 \end{aligned}$$

Tällä yhtälöryhmällä on heidän mukaansa tarkalleen kaksi ratkaisua: $(p, q, r) = \frac{1}{4}, \frac{1}{2}, \frac{1}{6}$ ja $(p, q, r) = \frac{1}{2}, \frac{1}{3}, \frac{1}{4}$. Epälineaaristen yhtälöryhmien ratkaisemista käsitellään luvussa 4.5.

Porterin ja muiden kuvassa 12 määrittelemät tasapainotilan ehdot täyttyvät molempien yhtälöryhmän ratkaisujen osalta. Ehtojen 1 ja 3 täyttyminen varmistettiin ratkaisemalla yhtälöryhmä. Ehto 4 täyttyy, koska kaikki ratkaistut todennäköisyydet ovat positiivisia. Jos tukien ulkopuolella ei ole yhtään strategiaa, ehdot 2 ja 5 täyttyvät automaattisesti, sillä molemmat ratkaisut ovat täydellisiä sekastrategioita. Muussa tapauksessa on varmistettava, että yksikään pelaaja ei saa suurempaa hyötyä mistään tuen ulkopuolella olevasta strategiastaan suhteessa muiden pelaajien tuessa olevaan strategiaan.

4.4 Tasapainotilan valinta

McKelvey ja McLennan kirjoittavat, että *tasapainotilan parantaminen* (equilibrium refinement) on jo kohtalaisen tutkittu peliteorian osa-alue [McM96]. Tasapainotilan parantaminen tarkoittaa tietynlaisten tasapainotilojen suosimista tasapainotilan valinnassa. Heidän mukaansa on mahdollista löytää niin kahden kuin useammankin pelaajan peleistä *vakaita joukkoja* (stable sets). Vakaat joukot ovat pelaajien strategiajoukkoja, jotka pysyvät Nashin tasapainotilana, vaikka itse peliin tehtäisiin tiettyjä muutoksia. Näin periaatteessa on mahdollista, että kerran löydettyä vakaata joukkoa voidaan käyttää uudestaan, jos peliin tulevat muutokset pysyvät pieninä. Tämä takaisi tiettyä jatkuvuutta agenttien toimintaan ja tekisi siten epätodennäköisemmän tilanteesta, jossa agentit vaihtavat toimintaansa joka vuoro.

Jotta tasapainotilan parantaminen toimisi, tarvitaan tietoa pelin rakenteesta. Joitain oletuksia Hyökkäyspelistä voidaan tehdä:

- Yleensä agentit liikkuvat kohti pelaajahahmoa tai pyrkivät pysymään suojassa.
- Sekastrategian pelaamiseen on harvoin tarvetta: agenttien väliset eturistiriidat ovat harvinaisia.
- Agenttien hyötyfunktiot voidaan tehdä keskenään vertailukelpoisiksi: sosiaalisen hyvinvoinnin kokonaisarvo pystytään näin ollen laskemaan.

Jos sekastrategiaa ei useinkaan tarvitse pelata, on luontevaa käyttää Porterin ja muiden esittelemää heuristiikkaa, jossa pienet tuet tutkitaan ensimmäiseksi [PNS04]. Porterin ja muiden algoritmi palautti ensimmäisen löydetyn tuen. Sitä olisi helppo muuntaa siten, että kaikki tuet, joilla jokaisen agentin tuen koko on yksi, tutkitaan. Kaikki löydetyt puhtaita strategioita käyttävät tasapainotilat kerätään talteen. McKelveyn ja McLennanin mukaan kaikkien tasapainotilojen löytämisen aikavaativuus on valtava [McM96]. Silti erilaisia puhtaiden strategioiden yhdistelmiä ei ole enempää kuin agenttien toimintojen lukumäärien tulo, ja kaikkien niiden tarkistaminen tasapainotilojen varalta voi olla laskennallisesti järkevää. Näistä tutkituista todennäköisistä tasapainotiloista voidaan ensin pudottaa pois hallitut tasapainotilat ja sitten valita jäljelle jääneistä sopivin. Koska agenttien hyötyfunktiot ovat Hyökkäyspelissä keskenään vertailukelpoisia, voidaan tasapainotilojen joukosta yksinkertaisesti poimia parhaimman sosiaalisen hyvinvoinnin antava tasapainotila. Toinen vaihtoehto on katsoa aikaisempien vuorojen tasapainotiloja ja valita tasapainotila niin, että se seuraa loogisesti aiempien vuorojen toiminnasta.

4.5 Epälineaariset yhtälöryhmät

Haatajan ja muiden mukaan epälineaaristen yhtälöryhmien tutkiminen on tärkeässä osassa monia luonnontieteitä, koska useiden luonnollisten ilmiöiden matemaattiset mallit ovat aina epälineaarisia [HHL99]. Heidän mukaansa kokonainen matematiikan osa-alue tutkii epälineaarisia järjestelmiä, ja erilaisia numeerisia ratkaisumenetelmiä on kehitetty runsaasti. Tässä tutkielmassa sivutaan numeerisia menetelmiä, jotta koko työkalupaketti Nashin tasapainotilojen etsimiseen saataisiin kokoon. Epälineaaristen yhtälöryhmien ratkaiseminen esitellään kuitenkin vain siinä laajuudessa, mitä Nashin tasapainotilojen löytäminen useamman kuin kahden pelaajan pelin tapauksessa periaatteessa vaatii.

Haatajan ja muiden mukaan numeeriset menetelmät ovat tapoja arvioida sellaisten matemaattisten ongelmien ratkaisuja, joita ei pystytä selvittämään analyyttisesti. Numeerisilla menetelmillä löydetty ratkaisu on heidän mukaansa likiarvo alkuperäisen matemaattisen tehtävän ratkaisulle.

Mikäli funktiossa $f(x) = 0$ on vain yksi riippumaton muuttuja, ongelma on Haatajan ja muiden mukaan yksiulotteinen ja sen nollakohta on numeerisilla menetelmillä helposti löydettävissä: eräs toimiva algoritmi on kokeilla eri x:n arvoja, kunnes on löydetty a ja b siten, että $f(a)$ ja $f(b)$ ovat erimerkkiset. Tällöin, mikäli funktio f on jatkuva, välillä $[a, b]$ on varmasti nollakohta. Sen jälkeen lasketaan a:n ja b:n välin puoliväli $m = \frac{(a+b)}{2}$. Jos arvot $f(m)$ ja $f(a)$ ovat erimerkkiset, tutkitaan seuraavaksi väliä $[a, m]$, muuten väliä $[m, b]$. Iterointia jatketaan, kunnes haluttu tarkkuus on saavutettu. Menetelmä soveltuu tosin vain sellaisiin tapauksiin, joissa funktio saa arvoja nollan molemmilta puolilta.

Press ja muut kirjoittavat, että mikäli funktiossa on useampi kuin yksi riippumaton muuttuja, tilanne on paljon monimutkaisempi [PFT86]. N:n tuntemattoman löytämiseksi tarvitaan N yhtälöä, ja tällöinkään tulos ei ole varma: epälineaarisessa yhtälöryhmässä voi olla useita ratkaisuja tai ei yhtään reaalista ratkaisua. Pressin ja muiden mukaan oikea lähestymistapa on tehdä tarpeeksi hyvä arvaus ratkaisulle. Arvauksen pitäisi perustua ongelman analysointiin, sillä huono arvaus voi aiheuttaa pahimmassa tapauksessa joidenkin ratkaisumenetelmien konvergoimattomuuden tai parhaassa tapauksessa ylimääräistä laskennallista kuormaa.

Pressin ja muiden sekä Haatajan ja muiden mukaan Newtonin menetelmä ja sen muunnokset ovat tunnetuin ja käytetyin tapa ratkaista sekä yksiulotteisten että moniulotteisten funktioiden juuria. Newtonin menetelmässä etsitään funktion juu-

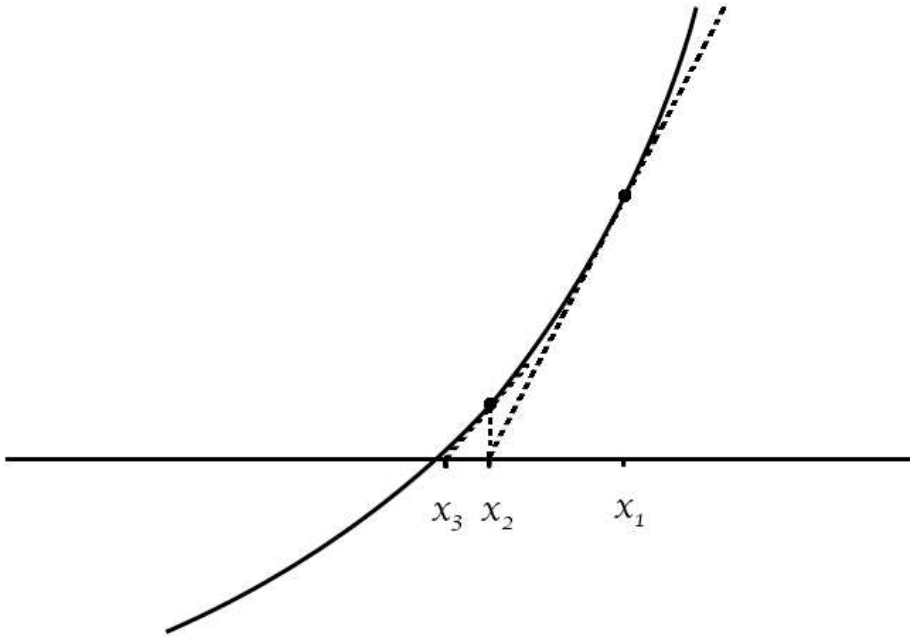
ri liikkumalla alkupisteestä kohti funktion nollakohtaa sopivan mittaisilla hakuaskelilla. Hakuaskelen pituus δ saadaan Pressin ja muiden mukaan saadaan täsmällisemmin Taylorin kehitelmästä:

$$f(x+\delta) \approx f(x) + f'(x)\delta + \frac{f''(x)}{2}\delta^2 + \dots$$

Mikäli δ on tarpeeksi pieni ja tutkittava funktio siisti, voidaan lineaaritermin ylittävät termit jättää tarkastelusta pois. Tällöin, kun $f(x+\delta) = 0$, saadaan ratkaistua yhtälöstä hakuaskelen pituus δ:

$$\delta = -\frac{f(x)}{f'x}$$

Kuvassa 13 esitetään geometrisesti algoritmin toiminta. Kun funktion derivaatta on laskettu, tiedetään, että funktion f tangentin kulmakerroin pisteessä a on $f'(a)$. Ensimmäisen askeleen aikana algoritmi on pisteessä x_1. Pisteen kautta kulkeva katkoviiva kuvaa funktion tangenttia sen kohdalla. Tangentin nollakohta on pisteessä x_2. Alkuperäiselle funktiolle lasketaan taas tangentti pisteessä x_2, ja tangentin nollakohta etsitään. Tätä voidaan periaatteessa toistaa hyvinkin kauan, mutta koska x_3 on riittävän lähellä funktion nollakohtaa, se hyväksytään funktion juuren likiarvoksi.



Kuva 13: Funktion nollakohdan etsiminen Newtonin menetelmällä.

Haataja ja muut kertovat, että Newtonin menetelmä tarvitsee käytännössä rajat a ja b, joiden välistä tulosta etsitään. Jos algritmi siirtyy rajojen yli, voidaan todeta, että se ei konvergoitunut mihinkään juureen. Jos hakuaskelen koko on pienempi kuin haluttu tarkkuus ϵ, on ratkaisu löytynyt. Algoritmisessa muodossa yksiulotteinen Newtonin menetelmä esitellään algoritmissa NewtoninMenetelmä.

NewtoninMenetelmä(a, b, ϵ)

```
1  x_1 ← (a+b)/2
2  k = 1
3  repeat
4        δ ← f(x_k)/f'x_k
5        x_{k+1} = x_k − δ
6        if x_{k+1} ∉ [a, b]
7           then error "hypättiin pois hakuvälistä"
8        k ← k + 1
9     until |δ| < ε
10 return x_{k+1}
```

Haatajan ja muiden mukaan funktion f osittaisderivaatoista muodostettua vektoria eli gradienttia merkitään seuraavalla notaatiolla:

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{pmatrix}$$

Toisin sanoen gradientti on muodostettu funktion $f(\mathbf{x})$ osittaisderivaatoista jokaisen funktion muuttujan x_i suhteen. Jacobin matriisi $J(\mathbf{x})$ on vektoriarvoisen funktion $\mathbf{f} : \mathbf{R}^n \mapsto \mathbf{R}^m$ osittaisderivaattojen matriisi:

$$J(\mathbf{x}) = \begin{pmatrix} \nabla^T f_1(\mathbf{x}) \\ \vdots \\ \nabla^T f_m(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{pmatrix}$$

Haatajan ja muiden mukaan funktiojoukkoa, jonka jokaisella funktiolla on monta riippumatonta muuttujaa, voidaan arvioida Taylorin sarjalla pisteessä $\mathbf{x} + \mathbf{s}$, missä $\mathbf{x}$ ja $\mathbf{s}$ ovat vektoreita, joissa on arvot jokaiselle muuttujalle.

$$\mathbf{f}(\mathbf{x} + \mathbf{s}) \approx \mathbf{f}(\mathbf{x}) + J(\mathbf{x})\mathbf{s}$$

$J(\mathbf{x})$ on Jacobin matriisi, johon on laskettu jokaisen funktion osittaisderivaatat jokaisen muuttujan suhteen. Haataja ja muut kertovat, että kun asetetaan $\mathbf{f}(\mathbf{x}+\mathbf{s}) = \mathbf{0}$, voidaan Taylorin sarjasta saada lineaarinen yhtälöryhmä:

$$J(\mathbf{x})\mathbf{s} = -\mathbf{f}(\mathbf{x})$$

Lineaarinen yhtälöryhmä on Pressin ja muiden mukaan ratkaistavissa analyyttisesti [PFT86]. Tästä saadaankin Haatajan ja muiden esittelemä Newtonin menetelmän laajennus yhtälöryhmille, joka saa parametrikseen alkuarvauksen $\mathbf{x}^1$ ja halutun tarkkuuden ϵ:

NewtoninMenetelmäYhtälöryhmille($\mathbf{x}_1, \epsilon$)
1 $k = 1$
2 **repeat**
3 $\quad\quad \mathbf{s}_k \leftarrow$ Solve($J(\mathbf{x}_k)\mathbf{s}_k = -\mathbf{f}(\mathbf{x}_k)$)
4 $\quad\quad \mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \mathbf{s}_k$
5 $\quad\quad k \leftarrow k + 1$
6 **until** $\|\mathbf{s}_k\| < \epsilon$ tai $k > k_{\max}$
7 **return** $\mathbf{x}_{k+1}$

NewtoninMenetelmäYhtälöryhmille-algoritmi muistuttaa toiminnaltaan täysin algoritmissa NewtoninMenetelmä esiteltyä tapaa ratkaista yhden funktion nollakohta. Eroavaisuutena on, että kokonaista muuttujaryhmää kuljetetaan kohti nollakohtiaan yhden funktion sijasta. Solve-algoritmi on implementaatio jostakin Pressin ja muiden mainitsemista useista analyyttisistä tavoista ratkaista lineaarisia yhtälöryhmiä.

Press ja muut mainitsevat, että Newtonin menetelmä toimii oikeastaan vain silloin, kun alkuarvaus on hyvä. Heidän mukaansa mikään käytössä olevista epälineaaristen yhtälöryhmien ratkaisumenetelmistä ei myöskään ole kovin hyvä, eikä erittäin todennäköisesti koskaan pystytä kehittämään hyvää ja yleistä menetelmää epälineaaristen yhtälöryhmien ratkaisuun. Nashin tasapainotilan etsimisessä on hyvänä puolena, että muuttujat ovat todennäköisyyksiä, ja saavat siksi arvoja vain väliltä $[0 \ldots 1]$. Näin ollen esimerkiksi $\frac{1}{2}$ voi olla hyvä arvaus jokaisen muuttujan alkuarvoksi.

5 Hyötyfunktio

Tässä luvussa käsitellään hyötyfunktiota ja sen rakennetta tietokonepelin tapauksessa. Luvussa pohditaan myös, miten erilaisia tunnettuja tietokonepelien tekoälyn menetelmiä pystytään koodaamaan mukaan agenttien hyötyfunktioihin.

5.1 Hyötyfunktion suunnittelu

Kuten luvussa 2.1 kerrotaan, tasapainotilan etsintä hyötymatriisista on vain moottori, joka takaa agenttien rationaalisuuden. Kaikki agenttien toimintaan ja toimintapäätöksiin vaikuttavat tekijät on jollain tavalla koodattava mukaan hyötyfunktioon. Agentti pohjaa päätöksensä vain ja ainoastaan siihen hyötyyn, jonka tietyn toiminnon pelaaminen aiheuttaa. Peliteoreettisessa mielessä ei ole olemassa käsitettä ”toimia omaa hyötyään vastaan”; jos agentti joskus toimii siten, että se vaikuttaa saavan enemmän hyötyä toisenlaisesta käytöksestä, on sen hyötyfunktio luultavasti erilainen kuin mitä luullaan.

Jotta mitä tahansa tietokonepeliä voitaisiin peliteoreettisesti mallintaa, täytyy jatkuva peli diskretisoida ainakin ajan suhteen. Tämä johtuu siitä yksinkertaisesta seikasta, että eri toiminnoilla saatuja hyötyfunktion arvoja täytyy verrata keskenään. Jos peliä ei voida jollain tavalla jakaa eri vaiheisiin, ei vertailua pystytä mielekkäästi suorittamaan.

Tietokonepeleissä agentit voidaan Russellin ja Norvigin mukaan jakaa kahteen kastiin: reaktiivisiin ja ei-reaktiivisiin [RuN03]. Näissä erona on muisti. Reaktiiviset agentit eivät omaa minkäänlaista tilaa: ne pelkästään reagoivat hyötyfunktionsa avulla ympäristön ärsykkeisiin. Reaktiivisten agenttien tekeminen on yksinkertaista, sillä niiden ei tarvitse pitää kirjaa siitä, mitä ne olivat tekemässä. Nareyekin mukaan hyödyt ovat ilmeisiä: reaktiiviset agentit ovat nopeita, yksinkertaisia tehdä ja pystyvät mallintamaan ei-pelaajahahmon uskottavaa käytöstä monissa tilanteissa [Nar00]. Reaktiivisuuden varjopuoli on, että koska sisäistä esitystä ei voida muuttaa, hyötyfunktio pysyy agenteilla aina samanlaisena ja minkäänlaista muistia ei voida toteuttaa. Tietokonepeleissä muistin tärkein tehtävä on luoda johdonmukaista toimintaa ja siten estää agenttia näyttämästä tyhmältä. Ei-reaktiivinen agentti voi siis muistaa, mihin oli pyrkimässä, ja pitää sitä ensisijaisena tavoitteenaan.

Peliteoreettinen malli agenttien käyttäytymisen ohjaamiseen kelpuuttaa sekä reaktiivisen että ei-reaktiivisen agenttimallin. Reaktiivinen malli on yksinkertainen tehdä:

hyötyfunktio on staattinen ja ottaa parametreikseen agenttien toimintavektorin ja pelimaailman. Koska hyötyfunktio on peliteoreettisessa mallissa ainoa tapa vaikuttaa maailman toimintaan, ei-reaktiivisen agentin on pystyttävä vaikuttamaan sisäisellä mallillaan joko hyötyfunktioonsa tai sen parametreihin. Suora vaikuttaminen hyötyfunktioon on mahdollista tehdä esimerkiksi funktion termien painokertoimien avulla: jos agentti on edellisellä vuorolla lähtenyt hyökkäykseen, sen tavoitteena on selvästi ensisijaisesti päästä pelaajahahmon lähelle. Tämän muuttuneen tavoitteen ansiosta agentin ja pelaajahahmon läheisyydestä hyötyä antavan termin kerrointa voidaan kasvattaa. Toinen tapa on lisätä hyötyfunktion parametrina annettavaan ympäristöön agentin sisäinen tila, joka vaikuttaa funktioon kuten mikä tahansa muu ympäristöinstanssin muuttuja. Molemmat tekniikat ovat hyvin lähellä toisiaan, ja erot niiden välillä ovat lähinnä filosofisia.

5.2 Hyötyfunktion rakenne

Levy ja Rosenschein tutkivat saalistaja-saalis-ongelman peliteoreettista ratkaisua [LeR92]. He pyrkivät luomaan saalistaja-agenteille hyötyfunktion, jota käyttämällä neljä saalistaja-agenttia pystyy ympäröimään satunnaisesti ruudukossa liikkuvan saaliin. Levy ja Rosenschein tulevat lopputulokseen, jossa hyvä hyötyfunktio sisältää kaksi tavoitetta: mahdollisimman nopean konvergoinnin pelin lopputilanteeseen ja globaalin tavoitteen — saaliin kiinnisaamisen — toteutumisen. He jakavat nämä tavoitteet kahteen eri hyötyfunktion osaan.

Myös Hyökkäyspelin hyötyfunktio on rakenteeltaan polynomi, jossa on niin monta termiä, kuin mitä saatavaan hyötyyn vaikuttavia tekijöitä on.

$$\pi = X_1 + X_2 + \ldots + X_k$$

Jokainen termi koostuu tekijästä ja painokertoimesta, joka tuo tekijän suuruusluokan oikeaan suhteeseen muiden hyötyfunktion termien kanssa.

$$X_i = aX_i'$$

Jos jokin termi halutaan jättää pois hyötyfunktiosta, annetaan painokertoimelle arvo 0.

Hyötyfunktion termit voidaan määritellä yksinkertaisten suureiden kautta: esimerkiksi yksi termi voisi olla vihollisagentin etäisyyden pelaajahahmosta käänteisluku

kerrottuna jollain painokertoimella. Tämä tapa suunnitella hyötyfunktiota ei ole kovin intuitiivinen, sillä funktion muuttaminen vaatii sen kaikkien osien ymmärtämisen. Esitänkin nyt, että erityisesti tietokonepelien tapauksessa hyötyfunktion termit nimetään sen mukaan, mitä vihollisagentin ominaisuutta se kuvaa. Näin agenttien eri ominaisuuksia voi tarkastella myös irrallaan kokonaisuudesta.

Edellämainittu etäisyyttä mittaava termi on täten nimeltään *verenhimo*. Se mallintaa agentin halua päästä pelaajahahmon lähelle. Tekijän arvo on $-xD_i$, missä D_i on agentti i:n etäisyys pelaajasta ja x on ylimääräinen kerroin, joka kuvaa verenhimon kasvamista sitä mukaa, kun pelaajaa lähestytään. Ajatuksena on, että kun vihollisagentti on riittävän lähellä pelaajahahmoa, sen halu liikkua aggressiivisesti kohti pelaajahahmoa on suurempi, kuin jos se olisi vielä kaukana.

Turvallisuus kuvaa vihollisagentin halua välttää omaa riskiä. Kun agentin valitsema toiminto vie sen suojaan, saadaan hyötyä lisää vakiomäärä. Kun agentti on näkyvillä tai muuten alttiina pelaajahahmon uhalle, saa se sanktion. Rangaistuksen määrä on $-r_i/k$, missä r_i on agenttia i kohtaavan uhan vakavuus ja k on samalle uhalle alttiina olevien agenttien määrä. Uhan vakavuutta on vaikea määrittää, mutta jonkinlainen ohjenuora on kierroksessa saatavan vahingon odotusarvo. Ajatuksena on, että yksittäiselle agentille on turvallisempaa, jos monta agenttia on samaan aikaan alttiina pelaajahahmon vaikutukselle.

Pelillisesti pyritään saamaan aikaan tilanne, jossa agentit päättävät yksissä tuumin, että hyökkäyksen aika on tullut. Sanktio voidaan määrätä erikseen jokaiselle erilaiselle uhalle, joka kohtaa vihollisagenttia. Jos hyötyfunktioon rakennetaan muisti, voi uhasta tulevan sanktion painokerrointa korottaa, jos uhka on toteutunut. Tällöin esimerkiksi tulituksen kohteena oleva agentti saisi enemmän hyötyä (eli vähemmän sanktiota), jos se pyrkisi suojaan.

Tasapaino merkitsee vihollisagentin pyrkimystä pysyä osapuilleen samassa tilassa kuin kaikki muutkin agentit. Tasapaino on negatiivinen termi, jonka tekijä saa arvokseen $-d_i$, missä $d_i = |D_i - \frac{\sum D_{-i}}{k}|$. Siinä lasketaan kaikkien muiden vihollisagenttien etäisyys pelaajahahmosta, ja selvitetään näiden etäisyyksien keskiarvo. Sen jälkeen lasketaan tämän keskiarvon ja agentti i:n etäisyyden pelaajasta erotuksen itseisarvo, jonka vastaluku on termin tekijän arvo. Termi siis saa sitä suuremman negatiivisen arvon, mitä kauempana agentti on muiden agenttien keskimääräisestä etäisyydestä pelaajasta. Termin tarkoituksena on saada agentit pysymään keskimäärin samalla etäisyydellä pelaajahahmosta, jotta lähestyminen vaikuttaisi uskottavalta ja agentit eivät juoksentelisi yksi kerrallaan pelaajahahmon kimppuun.

Hiiviskely mallintaa agentin halua pysyä piilossa. Piilossapysyminen ei välttämättä ole kaikissa peleissä edes mallinnettu, mutta Fungen mukaan suuressa osassa toimintapelejä pelaajahahmon näkökenttä on rajoitettu [Fun04]. Tekijän arvo on $log(t_i)x$, missä t_i on agentti i:n piilossa pysymien vuorojen lukumäärä ja x on vakioarvo saadulle hyödylle. Piilossa pysyttyjen vuorojen määrän kasvaessa hyöty kasvaa logaritmisesti.

Kunnianhimo tarkoittaa vihollisagentin halua olla ensimmäinen tai viimeinen hyökkääjä kohti pelaajaa. Kunnianhimo on vakiotekijä, jonka arvo on 0, jos vihollisagentin toiminta ei vie agenttia kohti pelaajahahmoa. Jos kukaan muu ei ole liikkumassa kohti pelaajahahmoa, kunnianhimo saa arvokseen positiivisen arvon tx. Siinä t on vuorojen lukumäärä, joiden aikana yksikään agentti ei ole liikkunut kohti pelaajahahmoa, ja x on vakio. Tällä termillä on tarkoitus säätää vihollisagenttien aggressiivisuutta ja antaa kannustin deadlock-tilanteiden lopettamiseen.

Elintila on agentin ympärilleen tarvitsema tila, eli etäisyys muista agenteista. Ei ole selvää, onko pelattavuuden kannalta parasta pitää vihollisagenttiryhmä suhteellisen pienellä alueella vai laajalle levittäytyneenä. Tiiviinä ryhmänä liikkuvat agentit voivat vaikuttaa toimivan suunnitelmallisesti, mutta pitkiä välimatkoja toisiinsa pitävät agentit hakeutuvat automaattisesti pelaajahahmon eri puolille ja pyrkivät siten piirittämään sen. Elintilan tekijä saa arvokseen x_i, joka tarkoittaa agentin i etäisyyttä lähimpään pelaajaan, jos se on alle kynnysarvon x'. Jos etäisyys on suurempi kuin kynnysarvo, $x = x'$.

Inertia on vihollisagentin taipumus pysyä aiemmin valitsemassaan toiminnassa. Termin tarkoitus on saada agentit näyttämään johdonmukaisilta toiminnassaan. Esimerkiksi agentti voi olla päättänyt hyökätä kohti pelaajaa, mutta seuraavalla kierroksella uhka näyttää jo niin suurelta, että agentti saa takaisin piiloon pakenemisesta suuremman hyödyn. Tällöin on olemassa riski, että agentti jää ikuiseen silmukkaan toistamaan näitä kahta toimintoa. Inertian ajatuksena on antaa bonus, joka on x, jos agentti liikkuu samaan suuntaan kuin viime vuorolla. Joissain tapauksissa halutaan sallia asteen verran vapaampi liike tai kahdessa vuorossa tapahtuva suunnan muutos. Silloin voidaan antaa bonus $x/2$, jos agentti liikkuu suuntaan, joka on kohtisuorassa edellisen vuoron liikkeen suunnan kanssa. Jos agentti pysyy paikallaan tai liikkuu vastakkaiseen suuntaan kuin edellisellä vuorolla, inertiabonusta ei anneta.

5.3 Etäisyys ja muut hyötyfunktion parametrit

Oikea tapa yhdistää muita tietokonepelien tekoälyssä käytettäviä tekniikoita peliteoreettiseen agenttimalliin on koodata ne mukaan hyötyfunktioon. Etäisyys muihin agentteihin ja pelaajahahmoon on yksi parametri, jonka ratkaisemiseen voidaan käyttää yleisesti tunnettuja menetelmiä. Jos agentti pyrkii minimoimaan absoluuttisen etäisyytensä pelaajahahmoon, ongelmaksi muodostuvat tilanteet, joissa agentti joutuu kulkemaan tilapäisesti kauemmas pelaajahahmosta esimerkiksi esteen kiertääkseen.

Kuvassa 14 vihollisagentti 1 laskee lyhimmän etäisyytensä pelaajahahmoon, jota merkitään **P**-kirjaimella. Pelaajahahmoa lähinnä oleva tyhjä ruutu, johon agentti pystyy kulkemaan vain etäisyyttä lyhentämällä, on merkitty **a**-kirjaimella. Kuvassa esitetyssä tilanteessa oletetaan vihollisagentille hyötyfunktio, joka antaa suurimman hyödyn toiminnoista, jotka vievät agentin kohti pelaajahahmoa. Nyt kuitenkin vihollisagentti joutuu pelaajahahmoa lähestyäkseen kulkemaan sellaisten ruutujen kautta, joiden absoluuttinen etäisyys pelaajasta on suurempi kuin sen ruudun, jossa agentti alkutilanteessa on. Tavallinen etäisyydenmittaus esimerkiksi Pythagoraan lauseella tai Manhattan-etäisyydellä ei siis toimi oikein.

```
. . . . . .
. . . . . .
. . . . . XX. . . . . .
. . . . . 1. . . . aX. . . . . P. . . . . .
. . . . . XXXX. . . . . .
. . . . . XXXXX. . . . . .
```

Kuva 14: Vihollisagentti 1:n ja pelaajahahmon välissä on este. Pelaajahahmoa lähin ruutu, johon vihollisagentti pääsee pelaajahahmoon mitattua etäisyyttä pienentämällä, on merkitty **a**-kirjaimella.

Ratkaisu on käyttää etäisyydenmittaukseen polunetsintäalgoritmeja, jotka ovat yleisessä käytössä tietokonepeleissä. Russellin ja Norvigin esittelemä A*-algormitmi on näistä ehkä tunnetuin [RuN03]. Ajatuksena on, että etäisyys pelaajaan ei enää olekaan absoluuttinen etäisyys, vaan pikemminkin lyhimmän löydetyn pelaajan luokse menevän reitin pituus.

Kuvassa 15 vihollisagentti on löytänyt tiensä pelaajahahmon luokse käyttäen A*-

algoritmia. +-merkit kuvassa esittävät ruutuja, joiden kautta vihollisagentti on kulkenut.

```
. . . . . .
. . . . . .++++++++. . . . . .
. . . . . .+.XX...1. . . . . .
. . . . . .++++++X. . . . .P. . . . . .
. . . . . .XXXX. . . . . .
. . . . . .XXXXX. . . . . .
```

Kuva 15: Vihollisagentti 1 kiertää esteen päästäkseen pelaajahahmon luokse.

Tietokonepeliohjelmoijat käyttävät tekoälyä myös pelin vaikeustason määrittämiseen. Monissa peleissä helpoimmilla vaikeustasoilla tietokoneen ohjaamat vastustajat ovat vähemmän aggressiivisia tai muulla tavalla helpompia voittaa. Spronck ja muut ovat tutkineet vaikeustason hallitsemista *dynaamisilla skripteillä* (dynamic scripting), joilla pelin vaikeustasoa voi säätää pelin aikana [SSP04]. Heidän merkittävä huomionsa on, että tekoälyn vaikeustasoa säädettäessä on pyrittävä muuttamaan tietokonevastustajan käyttäytymismallia, eikä vaan tekemään siitä tyhmempää. Spronckin ja muiden artikkelissa todetaan, että mielenkiintoisin helpotettu vaikeustasomalli oli ”noviisi”, joka yritti matkia aloittelevan ihmispelaajan pelityyliä.

Jos pelin vaikeustasoon halutaan vaikuttaa tekoälyn kautta, on hyötyfunktio oikea paikka siihen. Erilaisia käyttäytymismalleja on mahdollista tehdä yksinkertaisesti luomalla uusi hyötyfunktio erilaisilla painokertoimilla tai jopa termeillä. Ei kuitenkaan ole selvää, että tietokonepelin vaikeutta kannattaa säätää tekoälyä heikentämällä. Pelin pelattavuuden kannalta parempi ratkaisu voi olla yksinkertaisesti muuttaa vihollisagenttien peliteknisiä ominaisuuksia, kuten kestopisteitä tai nopeutta, agenttien kannalta epäedulliseen suuntaan.

Hyökkäyspelissä agenttien toiminnan satunnaisuus tulee sekastrategioiden pelaamisen aiheuttamasta toimintojen stokastisesta valitsemisesta. Kuitenkin esimerkiksi Porterin ja muiden käyttämä yhden tasapainotilan löytämisalgoritmi pyrkii välttämään sekastrategioiden etsimistä tehokkuussyistä [PNS04]. Jos tällöin halutaan lisää satunnaisuutta agenttien toimintaan, voidaan sitä saada säätämällä hyötyfunktiota. Tällöin agenttien päättely pysyy edelleen rationaalisena, mutta päättelyyn vaikuttavissa tekijöissä voi olla virheitä. Yksi tapa tämän toteuttamiseen on asettaa etäisyyttä mittaaviin funktioihin virheparametri, joka kuvaa agenttien etäisyyden arviointia:

toisinaan arvio saattaa olla ruudun tai kaksi pielessä. Ongelmaksi tässä muodostuu, että agentin edessä ja takana olevien ruutujen välinen ero on vain kaksi ruutua, ja tämän eron mukaan agentti valitsee hyökkäyssuuntansa. Jos agentti yllättäen päättäisikin liikkua ruudun taaksepäin ja sitten taas eteenpäin, näyttäisi se pelaajan silmissä oudolta. Toinen tapa tehdä satunnaisuutta on yksinkertaisesti lisätä luvussa 5.2 esiteltyyn hyötyfunktioon termi *virhe*, joka sisältää vain yhden satunnaisluvun väliltä $[0 \ldots 1]$. Virheen ylärajaa voidaan säätää painokertoimen mukaan.

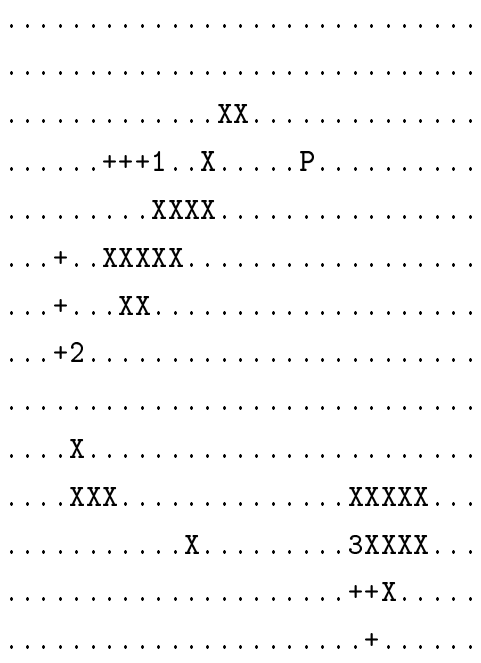
5.4 Hyötyfunktioiden testausta

Jotta erilaisia hyötyfunktiota ja niillä saavutettuja tuloksia voidaan vertailla, täytyy olla jokin metriikka, jolla tulosten paremmuutta mitataan. Ongelmana on, että jokaisella hyötyfunktiolla agentit käyttäytyvät tarkalleen yhtä rationaalisesti eli "oikein". Hyötyfunktio kertoo ainoastaan agentin oman arvostuksen eri asioiden (eli Hyökkäyspelin tapauksessa pelitilojen) välillä. Pelisuunnittelijan tehtävänä on päättää, miten vihollisagenttien olisi tarkoitus toimia, jotta ne vastaisivat kaikkein parhaiten tietokonepelin tarinankerrollisia tarpeita. Jokin tietty vihollistyyppi voi esimerkiksi olla nopea ja pyrkiä suoraviivaisiin hyökkäyksiin, kun taas hidas ja kestävä vastustaja väijyttää pelaajahahmon mieluiten piilosta ja läheltä. Eri agenttityyppien hyötyfunktio näyttää aivan erilaiselta, ja silti ne voivat olla "yhtä hyviä".

Hyötyfunktioiden paremmuuden arviointi onkin tehtävä pelattavuuden näkökulmasta. Koska laajojen pelitestien järjestäminen ei ole tämän tutkielman laajuudessa järkevää, tyydytään hyötyfunktion termeille löytämään sellaisia tekijöitä ja painokertoimia, joilla saadaan aikaan silmämääräisesti älykkään näköinen vihollisagenttien toiminta. Valitettavasti myös termien vakiot ja painokertoimet ovat suoraan suhteessa pelimaailman ominaisuuksiin — esimerkiksi *verenhimo*-termissä käytetään pelaajahahmon ja vihollisagentin välistä etäisyyttä laskutoimituksiin, mutta etäisyys voi olla eri tietokonepeleissä aivan eri mittayksiköissä.

Seuraavat kuvat esittävät luvussa 3.6 kuvatun testipelin toimintaa, kun hyötyfunktio on koottu eri termeistä luvun 5.2 mukaan. Ainoastaan *hiiviskely*-termi on jätetty toteutuksesta pois. Jokaisen kuvan yhteydessä kommentoidaan, mistä on kysymys.

Pelin alkutilanne on esitetty kuvassa 2 sivulla 5. Kuvassa 16 on sama peli kolmen pelivuoron jälkeen. Kuvasta nähdään, että jokainen agentti on aloittanut liikkeensä kohti pelaajahahmoa. Vihollisagentti 1 on päättänyt kiertää esteen yläpuolelta ja vihollisagentti 2 alapuolelta. Agentti 2 on laskenut lyhimmän reitin sekä yhden



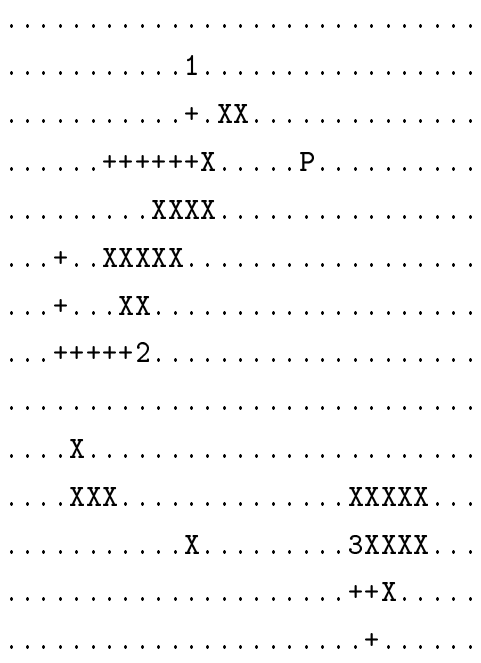
Kuva 16: Vuoro 3. Vihollisagentti 1 on lähtenyt kiertämään estettä pohjoisesta ja vihollisagentti 2 etelästä. Agentti 3 on liikkunut suojaan.

ruudun lähtöruutuaan ylempää että yhden ruudun lähtöruutuaan alempaa. Ensimmäinen reitti olisi kiertänyt esteen yläpuolelta eli samaa reittiä, jota agentti 1 kulkee. Molempien reittien pituus on 20 ruutua, joten agentin kannalta molemmista reiteistä saatava hyöty on sama. Kuitenkin esteen alapuolelta kulkeva reitti vie agentin 2 aluksi kauemmaksi agentista 1, jolloin agenttien 1 ja 2:n hyöty kasvaa *elintila*-termin myötä.

Kuvassa 17 on pelitilanne vuorolla 7. Agentti 3 on jäänyt samalle paikalle, jossa se oli vuorolla 3. Hyökkäys aukean yli pelaajahahmon kimppuun olisi vaarallista, joten *turvallisuus*-termi antaa suuren sanktion piilosta esiin tulemiselle. Agentti 2 on saapunut hyökkäystasalle.

Kuvassa 18 kaikki agentit ovat saapuneet niin lähelle pelaajahahmoa kuin pystyivät ollen vielä suojassa. Vaikka kuvasta se ei välttämättä käy ilmi, yksikään agentti ei ole pelaajahahmon kannalta näkyvissä, kuten kuvasta 9 sivulla 35 selviää.

Kuvassa 19 agentit ovat saaneet yhden vuoron odottelun jälkeen kerättyä riittävästi rohkeutta hyökkäystä varten, ja jokainen agentti astuu samanaikaisesti esiin suojastaan. Peliteoreettisesti tämän voi aiheuttaa kaksi asiaa. Ensimmäinen vaihtoehto on, että jokin agentti saa *kunnianhimo*-termistä niin suuren hyödyn, että

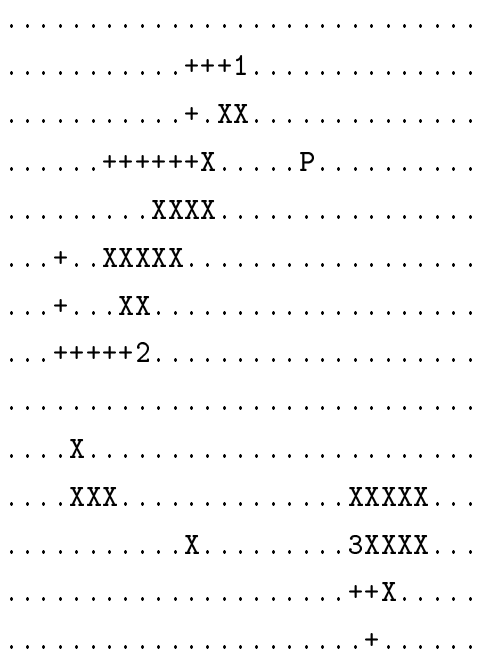


Kuva 17: Vuoro 7. Vihollisagentti 3 on jäänyt esteen taakse odottamaan. Agentti 2 on juuri saapunut hyökkäystasalle. Agentti 1 lähestyy edelleen esteen takana.

liikkeellelähtöstrategia hallitsee muita strategioita, ja agentti lähtee siksi liikkeelle kaikissa Nashin tasapainotiloissa. Tällöin myös kaikkien muiden agenttien *turvallisuus*-termistä saama sanktio pienenee niin paljon, että niidenkin kannattaa lähteä liikkeelle. Toinen mahdollisuus on, että tasapainotilan löytämisalgoritmi sattuu osumaan siihen tasapainotilaan, jossa kaikki agentit liikkuvat eteenpäin, vaikka toisessa tasapainotilassa kaikki agentit jäisivät paikalleen. Testipelin ajoissa molempia tapauksia sattui, joskin eri tasapainotilaan osuttiin vain, jos laskettiin useampia tasapainotiloja ja valittiin niistä yksi.

Juuri vihollisten samanaikaisten yhteistoimintapäätösten tekeminen on tietokonepelin tunnelman kannalta tärkeää. Jos agentit olisivat astuneet näkyviin yksi kerrallaan peräkkäin, olisi ihmispelaajan kokemus ollut erilainen. Nyt ihmispelaajalle välittyy mielikuva suunnittelevista ja keskenään kommunikoivista vastustajista, jotka pystyvät älykkääseen yhteistoimintaan.

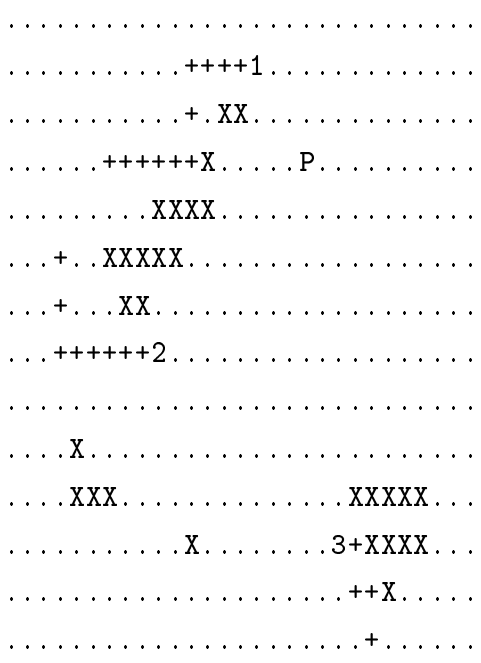
Vuorolla 21 agentit 1 ja 3 ovat päässeet pelaajahahmon viereen, kuten kuvassa 20 esitetään. Agentti 2 on astunut uudestaan piiloon, eikä ole lähtenyt sieltä eteenpäin. Tilanne vaikuttaa virheeltä, ja on totta, että agenttien yhteishyökkäyksen tehokkuus heikkenee, jos jokin agenteista jää puolessavälissä matkaa uuteen suojaan.



Kuva 18: Vuoro 10. Vihollisagenttiryhmän kaikki jäsenet ovat piilossa keräämässä rohkeutta hyökkäykseen.

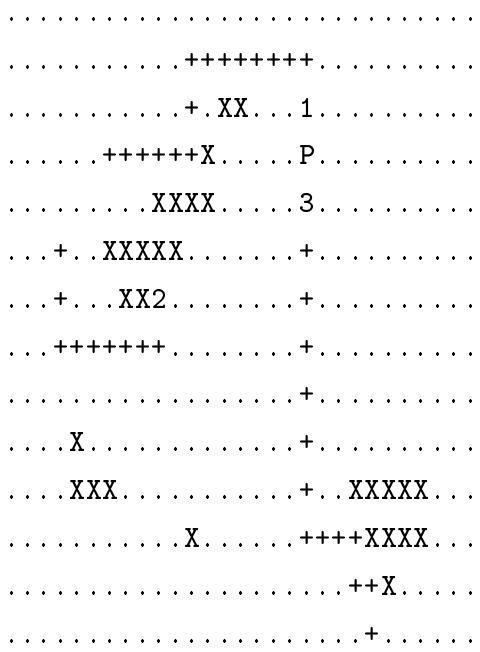
Asia pystytään korjaamaan säätämällä *turvallisuus*-termissä uhka-muuttujaa kertomaan, että jos jokin agentti on jo käsikähmässä pelaajahahmon kanssa, on muiden agenttien turvallista liikkua näkyvillä. Testipelin hyötyfunktiossa käytettiin kuitenkin uhka-muuttujan arvona yksinkertaisuuden vuoksi vakioarvoa, kun agentti on pelaajahahmon näköpiirissä. Tilannetta voikin tulkita niin, että koska agentti 2 arvioi tilanteen vaaralliseksi, se pitää parempana tilanteena pysytellä taka-alalla odottamassa tilanteen kehitystä. Oikeassa tietokonepelissä seinät, joita pitkin kulkien joka toinen ruutu on pelaajahahmon näköpiirissä ja joka toinen ei, ovat myös luultavasti harvinaisia.

Testeistä selviää, että hyötyfunktion rakentaminen termeistä tuntuu toimivan varsin hyvin ja johtavan ainakin tässä Hyökkäyspelin tapauksessa älykkään näköiseen toimintaan. Hyötyfunktion luominen käsin on kuitenkin haastavaa, sillä erilaisissa pelitilanteissa funktion termit saattavat kumota toisensa tai vaikuttaa ennalta arvaamattomalla tavalla. Erityisesti *turvallisuus*- ja *verenhimo*-termien kanssa tasapainottelu voi johtaa yllättäviin tuloksiin. Pelintekijä voikin haluta oikaista mutkia ja pakottaa agentit hyökkäämään, kun pelitilanteesta selviää, että vihollisagenttiryhmä on tehnyt jonkinlaisen hyökkäyspäätöksen. Tällöin hyötyfunktioon lisätään



Kuva 19: Vuoro 12. Vihollisagenttiryhmä tekee hyökkäyspäätöksen. Kaikki agentit lähtevät liikkeelle suojistaan.

kesken peliä termi, joka antaa erittäin suuren bonuksen agentteja pelaajahahmoa kohti vieville liikkeille. Tämän jälkeen toiminnot, jotka vievät agenttia kohti pelaajahahmoa, hallitsevat kaikkia muita toimintoja.



Kuva 20: Vuoro 21. Vihollisagentit 1 ja 3 ovat päässeet pelaajahahmon luokse. Vihollisagentti 2 on jäänyt uudestaan näkösuojaan.

6 Yhteenveto

Peliteoreettinen malli agenttien yhteistoiminnan parantamiseksi tuntuu toimivan hyvin ainakin periaatteellisella tasolla. Verrattuna Korfin esittelemään malliin, jossa ahneet agentit reagoivat toistensa tekemisiin viiveellä, peliteoria mahdollistaa sellaisten toimintapäätösten tekemisen, jotka kaikki agentit ovat valmiit hyväksymään [Kor92]. Näin esimerkiksi hyökkäyspäätöksen tekemisen ei tarvitse johtua siitä, että yhden agentin kannattaa hyökätä ja muiden kannattaa seurata. Peliteoreettisessa mallissa agentit voivat yhdessä todeta, että tasapainotila, jossa kaikki hyökkäävät, on parempi kuin tasapainotila, jossa kaikki pysyvät piilossa. Samoin jos jokin agentti saa jostain toiminnostaan aina suurimman hyödyn, ei sen tarvitse välittää muista agenteista. Muiden agenttien on sitä vastoin otettava sen toimintovalinta huomioon päättelyssään.

Myös käytännön tasolla peliteoreettinen malli osoittautui jokseenkin toimivaksi, kuten Hyökkäyspelin toteutus luvussa 3.6 esitellyssä testipelissä osoitti. Huoli tasapainotilan löytämiseen käytetyn algoritmin aikavaativuudesta on todellinen, mutta taulukosta 4 sivulla 40 nähdään, että pienien pelien tapauksissa itse tasapainotilan löytämiseen kuluva aika on Porterin ja muiden esittelemässä algoritmissa usein suhteellisen vähäinen [PNS04]. Tämä johtuu Porterin ja muiden käyttämän heuristiikan melko hyvästä soveltumisesta Hyökkäyspelin tapaukseen. Edelleen on teoriassa mahdollista, että pelin ainoa tasapainotila olisi täydellinen sekastrategia, jonka löytäminen Porterin ja muiden menetelmällä on kallista ja tapahtuu tukien etsimisjärjestyksessä kaikkein viimeisimpänä. Tämän takia voi olla hyvä ajatus käyttää jonkinlaista vaihtoehtoista mekanismia, mikäli sopivaa tasapainotilaa ei annetussa ajassa löydy. Yksi vaihtoehto on antaa agentin valita se toiminto, josta se saa suurimman hyödyn, jos kaikki muut agentit pysyisivät paikallaan. Tällöin agentin algoritmi muistuttaa paljon Korfin esittelemää ahnetta algoritmia [Kor92].

Kuka hyötyy peliteorian käytöstä tietokonepeleissä? Ensimmäinen vastaus on yksinkertainen: pelin pelaaja kokee, että agentit käyttäytyvät älykkään näköisesti, ja siksi immersoituu peliin hyvin. Toinen hyötyjä, joka ei ole niin ilmeinen, on tietokonepelin kenttäsuunnittelija. Skriptatussa tekoälyssä kenttäsuunnittelija joutuu päättämään, mistä päin pelaajan tulee lähestyä taistelun tapahtumapaikkaa, ja sovittamaan vihollisagenttien toiminnan sen mukaiseksi. Nareyekin mukaan skriptit ovat tyypillisesti rakenteeltaan yksinkertaisia, ja yleensä koostuvat if-else-ehtolauseista [Nar00]. Jos pelaaja lähestyykin tulevaa taistelupaikkaa eri suunnasta tai eri tavalla kuin mitä kenttäsuunnittelija ajatteli, voi syntyvä kohtaaminen olla puutteellises-

ti suunniteltu. Tällöin agenttien toiminta voi näyttää ihmispelaajan silmissä epäuskottavalta. Pelisuunnittelija joutuu tämän takia testaamaan skriptiään ja rakentamaan kohtauksen huolellisesti erilaiset mahdollisuudet huomioiden. Jos sen sijaan agentit pystyvät toimimaan ryhmässä edes jollain tasolla älykkäästi, vähenee tarve skriptien tekemiselle, sillä agentit pystyvät toimimaan uskottavan näköisesti erilaisissa tilanteissa. Tämä puolestaan lyhentää kenttäsuunnitteluun kuluvaa aikaa.

Agenttiryhmän toiminnan suunnittelu laajamuotoisia pelejä käyttämällä osoittautui aikavaativuudeltaan liian vaikeaksi. Agenteista on kuitenkin helppo tehdä eireaktiivisia lisäämällä niille muisti. Muistin käyttöä agenttien suorittaman suunnittelun apuna on mahdollista ajatella eräänlaisena vastakohtana perinteiselle suunnittelulle: sen sijaan, että katsottaisiin tulevaan, katsotaankin menneeseen ja käytetään jo tehtyjä tekoja pohjana uudelle toiminnalle. Muistin avulla on mahdollista selvittää lukkotilanteita, joissa agentit jäisivät ikuisesti paikalleen tai kiertämään kahden eri pelitilan välillä.

6.1 Jatkotutkimusajatuksia

Tässä tutkielmassa pystyttiin vain raapaisemaan peliteorian tietokonepeleihin soveltamisen pintaa ja esittämään yksinkertaisin tapa löytää agenttiryhmän toiminnan tasapainotiloja. Looginen tapa jatkaa työtä olisi tutkia normaalin tietokonepelin rakennetta peliteoreettisessa mielessä. Papadimitriou ja Roughgarden mainitsevat, että monet peliteoreettiset ongelmat voidaan palauttaa tiettyihin pelityyppeihin, kuten ruuhkautumispeleihin (congestion game) ja graafisiin peleihin (graphical game) [PaR05]. Esimerkiksi Hyökkäyspelin hyötymatriisin rakenteesta saattaa tarkemmalla tutkimuksella paljastua seikkoja, jotka voivat auttaa tasapainotilan ratkaisemiseen kelpaavien heuristiikkojen löytymistä.

Myös hyötyfunktion rakentumista voidaan tutkia pidemmälle. Spronck kirjoittaa, että oppiminen tietokonepelien tekoälyssä on kipeästi tarvittu parannus monimutkaisten pelitilanteiden parempaan hallintaan [Spr04]. Oppimista voisi Hyökkäyspelissä soveltaa juuri hyötyfunktioiden parametrien hallintaan. Tavoitteena olisi, että hyötyfunktio muuttuisi kaksitasoiseksi: ylemmällä tasolla olisivat agentin todelliset tavoitteet (”säily hengissä”, ”voita pelaaja” jne.) ja alemmalla tasolla olisi korkeamman tason tavoitteiden pohjalta opittu hyötyjen kuvaus agentin käytössä oleviin toimintoihin. Tällöin agentti voisi oppia, miten missäkin pelitilanteessa kannattaa toimia, jotta ylemmän tason tavoitteet täyttyvät mahdollisimman hyvin. Myös mahdollisuutta pelinaikaiseen oppimiseen voisi tutkia: jos agentit esimerkiksi hyökkäävät,

tulevat torjutuiksi ja pakenevat, ei seuraavaa hyökkäystä kannata tehdä samalla tavalla.

Ihmispelaaja oli testipelissä mallinnettu vain paikallaan olevana, toimimattomana kohteena. Oikeassa tietokonepelissä asia on tietysti aivan toisenlainen. Ihmispelaajan matemaattinen mallintaminen luvussa 3.3 esitetyillä menetelmillä voi olla oikea tapa lähestyä ongelmaa, mutta ilman oikeilla ihmistestipelaajilla suoritettua pelitestausta on vaikea sanoa, toimiiko malli oikein. Ihmispelaajan lisääminen testipeliin auttaisi myös selvittämään hyötyfunktion kelvollisuutta oikeassa pelitilanteessa ja kenties parantamaan sitä. Haynesin ja Senin mukaan niin sanottu lineaarinen saalis pystyy välttämään peliteoreettisten saalistajien ansat saalistaja-saalis-ongelmassa [HaS96]. Lineaarinen saalis valitsee yhden liikkumissuunnan ja jatkaa liikettään rajattoman kauan samaan suuntaan, jolloin saalistaja-agenttien on erittäin vaikea oppia ympäröimään sitä, vaikka ne pystyisivätkin liikkumaan nopeammin kuin saalis. Myös Hyökkäyspelissä ihmisvastustaja saattaa pystyä kehittämään liian helppoja tapoja selvitä erityisesti ei-oppivilla hyötyfunktioilla varustetuista vihollisagenteista. On kuitenkin muistettava, että tietokonepelien tarkoituksena on, että pelaaja pystyy lopulta siedettävällä vaivalla voittamaan vihollisagentit.

Vaikka laajamuotoiset pelit eivät ole aikavaatimuksensa puolesta käyttökelpoisia Hyökkäyspelin suunnitteluongelmia ratkottaessa, on myös toinen tapa mallintaa agenttiryhmän suunnittelua: Markovin pelit.

Markovin peleissä, joita Owen kutsuu stokastisiksi peleiksi, on ajatuksena yleistää *Markov-päätösprosessi* (Markov Decision Process, MDP) monen agentin tapaukseen [Owe68]. Tavallinen Markov-päätösprosessi on Russellin ja Norvigin mukaan tapa päättää sarjallisesti agentin toiminnasta epävarmassa ympäristössä [RuN03]. Epävarma ympäristö tarkoittaa, että agentti ei voi toiminnon valittuaan olla varma, mihin tilaan päädytään. Toisin sanoen Markov-päätösprosesseihin on sisäänrakennettu sama epävarmuuden hallinta, jonka kuvaamiseen laajamuotoisissa peleissä tarvittiin todennäköisyyssolmuja.

Markov-ketju on tilasiirtymäjoukko, joka toteuttaa Markov-ehdon:

$$P(X_{n+1} = x|X_0, X_1, \ldots, X_n) = P(X_{n+1} = x|X_n)$$

Markov-ehdossa X_n on todennäköisyysmuuttuja, joka kertoo prosessin tilan ajan hetkellä n. Markovin ehto kertoo, että todennäköisyys, että prosessin tila hetkellä $n + 1$ on jokin tietty arvo, riippuu vain prosessin tilasta hetkellä n, eikä mistään aikaisemmasta tilasta. Toisin sanoen Markovin ketjulla ei ole muistia.

Markov-päätösprosessissa agentin haasteena on löytää *toimintamalli* (policy), jota voi soveltaa kaikissa Markov-ketjun vaiheissa, ja näin ohjata agentin toimintaa. Agentti saa jokaisessa Markov-ketjun vaiheessa jonkin hyödyn, joka riippuu tilasiirtymästä. Tilasiirtymät eivät ole deterministisiä, vaan mahdollisuus päästä tiettyyn tilaan x riippuu edeltävästä tilasta. Jokaisen agentin tavoitteena on hyödyn optimointi tietyn siirtomäärän päähän laskettavassa ketjussa. Boutilier kutsuu tätä siirtojen määrää horisontiksi [BDH99]. Koska tilasiirtymän onnistuminen (tai toisin sanoen haluttuun tilaan pääseminen) ei Markov-ketjussa ole itsestäänselvyys, voidaan laskea hyödyn odotusarvo ja käyttää sitä tilan paremmuuden mittarina. Normaalissa Markov-päätösprosessissa ympäristö on staattinen, eli agentin jostain tilasta r saaman hyödyn odotusarvo $V(r)$ on vakio. Littman kertoo, että Markov-päätösprosesseissa tilan ja agentin siinä tekemän toiminnon muodostamalla parilla on suuri merkitys [Lit94]. Tila-toiminto-parilla (r, a) on hänen mukaansa laatu Q, joka muodostuu, kun agentti tekee tilassa r toiminnon a ja sen jälkeen seuraa optimaalista toimintamallia. Laatua mitataan seuraavilla yhtälöillä:

$$\begin{aligned} Q(r,a) &= R(r,a) + \gamma \sum_{r'} T(r,a,r')V(r') \\ V(r) &= \max_{a' \in A} Q(r,a') \end{aligned}$$

Tässä rekursiivisessa yhtälöparissa $R(r, a)$ tarkoittaa agentin toiminnostaan saamaa välitöntä hyötyä ja $T(r, a, r')$ todennäköisyyttä, että agentti pääsee toiminnolla a tilasta r tilaan r'. Kerroin γ on välillä $[0 \ldots 1]$. Se on alennuskerroin, jolla säädetään agentin myöhempien toimien sille tuomaa hyötyä. Mitä pienempi sen arvo on, sitä suurempi on agentin lähiaikojen toimistaan saama hyöty suhteessa myöhempiin toimiin.

Könösen mukaan Markovin pelit ovat tapa mallintaa Markovin päätösprosesseja dynaamisessa ympäristössä. [Kön04]. Markovin peleissä ympäristö ei ole enää staattinen, vaan muiden agenttien toimien mukaan määräytyvä. Pelissä prosessin tilasiirtymä tapahtuu kaikkien agenttien valitsemien toimintojen mukaan. Luonnollinen tapa tämän valinnan määräämiseen on peliteoria ja Nashin tasapainotila. Könösen mukaan tavoitteena Markovin peleissä onkin löytää agenttien yhteinen *tasapainotoimintamalli* (equilibrium policy), josta harhautuminen takaisi agentille huonomman hyödyn odotusarvon. Tasapainotoimintamalli on periaatteessa analoginen Nashin tasapainotilan kanssa: tasapainotoimintamallissa yhdenkään agentin ei kannata muuttaa omaa toimintamalliaan, koska se varmasti heikentää agentin saavut-

tamaa hyötyä. Markovin peleissä agenttiryhmän suunnitteluongelma siis ratkaistaan luomalla joukko pelitiloja, joista jokaisella on agentille tietty hyötyarvo. Agentit pyrkivät kulkemaan sellaisiin pelitiloihin, jotka tarjoavat niille mahdollisimman suuren hyödyn.

Erityisen kiintoisiksi Markovin pelit tekee, että niitä voi ratkaista käyttäen tunnettuja menetelmiä Markovin päätösprosessien ratkaisemiseen. Mielenkiintoinen jatkotutkimuskohde olisikin laajamuotoisten pelien vertaaminen Markovin peleihin tietokonepelien näkökulmasta katsoen.

Lähteet

Aum59 Aumann, R., Acceptable Points in General Cooperative n-person Games. *Contributions to the Theory of Games, Annals of Mathematical Studies, No. 40, Tucker, A., Luce, R. (toim)*, Hingham, MA, 2002.

BDH99 Boutilier, C., Dean, T., Hanks, S., Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *Journal Of Artificial Intelligence Research, Volume 11*, sivut 1-94, 1999.

Bou99 Boutilier, C., Sequential Optimality and Coordination in Multiagent Systems. *Proceedings of the sixteenth International Joint Conference on Artificial Intelligence*, 1999.

CLR01 Cormen, T., Leiserson, C., Rivest, R., Stein, C., *Introduction to Algorithms*, 2. painos, MIT Press, 2001.

CoS02 Conitzer, V., Sandholm, T., Complexity Results about Nash Equilibria. *Technical Report CMU-CS-02-135, School of Computer Science, Carnegie-Mellon University*, toukokuu 2002.

CRV04 Cheng, S., Reeves, D., Vorobeychik, Y., Wellman, M., Notes on Equilibria in Symmetric Games. *AAMAS-04 Workshop on Game Theory and Decision Theory*, 2004.

Dut99 Dutta, P., *Strategies and Games: Theory and Practice*, MIT Press, 1999.

FiL04 Finzi, A., Lukasiewicz, T., Relational Markov Games. *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA 2004)*, sivut 320-333. Lissabon, Portugali, 2004.

Fro06 Frozenbyte Oy, *Shadowgrounds*, 2006.

Fun04 Funge, J.D., *Artificial Intelligence For Computer Games: An Introduction*, Peters Corporation, 2004.

HaS96 Haynes, T., Sandip, S., Evolving behavioral strategies in predators and prey. *Adaptation, Coevolution and Learning in Multiagent Systems, Weiss G., Sen, S. (toim)*, sivut 113-126, Springer, 1996.

HeP00 Herings, J-J., Peeters, R., A differentiable homotopy to compute Nash equilibria of N-person games. *Economic Theory 18, 2001*, sivut 159-185. 2000.

HHL99 Haataja, J., Heikonen, J., Leino, Y., Rahola J., Ruokolainen, J., Savolainen, V., *Numeeriset menetelmät käytännössä*, CSC - Tieteellinen laskenta Oy, 1999.

Kir02 Kirby, N., Solving the Right Problem. *AI Game Programming Wisdom, Vol. 1, Rabin, S. (toim)*, Hingham, MA, 2002.

Kor92 Korf, R., A simple solution to pursuit games. *Working Papers of the 11th International Workshop on Distributed Artificial Intelligence*, sivut 183-194. Glen Arbor, Michigan, helmikuu 1992.

Kön04 Könönen, V., Policy Gradient Method for Team Markov Games. *Lecture Notes in Computer Science 3177*, sivut 733-739, 2004.

LeR92 Levy, R., Rosenschein, J.S., A game theoretic approach to the pursuit problem. *Working Papers of the 11th International Workshop on Distributed Artificial Intelligence*, sivut 195-213. Helmikuu 1992.

Lid03 Liden, L., Artificial Stupidity: The Art of Intentional Mistakes. *AI Game Programming Wisdom, Vol. 2, Rabin, S. (toim)*, Hingham, MA, 2003.

Lit94 Littman, M., Markov Games as a Framework for Multi-Agenti Learning. *Proceedings of the 11th International Conference on Machine Learning*, sivut 157-163. Morgan Kauffman, 1994.

McL99 McLennan, A., The expected number of Nash equilibria of a normal-form game. *Econometrica*, 1999.

McM96 McKelvey, R., McLennan, A., Computation of Equilibria in finite games. *Handbook of Computational Economics, Amman, H., Kendrick, D., Rust, J. (toim)*, sivut 87-142. Elsevier, 1996.

MMT06 McKelvey, R.D., McLennan, A.M., Turocy, T.L., Gambit: Software Tools for Game Theory, Version 0.2006.01.20. `http://econweb.tamu.edu/gambit` .

Nar00 Nareyek, A., Intelligent Agents for Computer Games. *Revised Papers from the Second International Conference on Computers and Games*, sivut 414-422, Springer, 2000.

Nas50 Nash, J., Equilibrium points in N-Person Games. *Proceedings of NAS, 36*, 1950.

Ork02 Orkin, J., Tips from the Trenches. *AI Game Programming Wisdom, Vol. 1, Rabin, S. (toim)*, Hingham, MA, 2002.

Owe68 Owen, G., *Game Theory*, 1968.

Pap01 Papadimitriou, C., Algorithms, games, and the internet. *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, 2001.

PaR05 Papadimitriou, C., Roughgarden, T., Computing Equilibria in Multi-Player Games. *Proceedings of the 16th annual ACM-SIAM Symposium on Discrete Algorithms*, sivut 82-91, Vancouver, 2005.

PaW02 Parsons, S., Wooldridge, M., Game Theory and Decision Theory in Multi-Agent Systems. *Autonomous Agents and Multi-Agent Systems 5*, sivut 243-254, Kluwer, 2002.

PFT86 Press, W., Flannery, B., Teukolsky, S., Vetterling, W., *Numerical Recipes*, Cambridge University Press, 1986.

PNS04 Porter, R., Nudelman, E., Shoham, Y., Simple Search Methods for Finding a Nash Equilibrium. *Proc. AAAI-04*, sivut 664-669, MIT Press, 2004.

Rab03 Rabin, S., Common Game AI Techniques. *AI Game Programming Wisdom, Vol. 2, Rabin, S. (toim)*, Hingham, MA, 2003.

Ras01 Rasmusen, E., *Games and Information: An Introduction to Game Theory*, Blackwell Publishers, 2001.

Rey99 Reynolds, C., Steering Behaviours for Autonomous Characters. *Proceedings of Game Developers Conference*, 1999.

RuN03 Russell, S., Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2003.

San99 Sandholm, T., Distributed Rational Decision Making. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, Weiss, G. (toim)*, sivut 201-258, MIT Press, 1999.

Sco02 Scott, B., The Illusion of Intelligence. *AI Game Programming Wisdom, Vol. 1, Rabin, S. (toim)*, Hingham, MA, 2002.

SGP02 Stirling, W., Goodrich, M., Packard, D., Satisficing Equilibria: A Non-Classical theory of Games and Decisions. *Autonomous Agents and Multi-Agent Systems 5*, sivut 305-328, Kluwer, 2002.

ShV99 Shih-Hung, W., Von-Wun, S., Game Theoretic Reasoning in Multi-Agent Coordination by Negotiation with a Trusted Third Party. *Agents*, sivut 56-61, 1999.

Spr04 Spronck, P. *Adaptive Game AI.* Ph.D. thesis, Maastricht University Press, Maastricht, 2004.

SSP04 Spronck, P., Sprinkhuizen-Kuyper, I., Postma, E., Online Adaptation of Game Opponent AI with Dynamic Scripting. *International Journal of Intelligent Games and Simulation 3*, sivut 45-53, 2004.

Tam97 Tambe, M., Towards Flexible Teamwork. *Journal of Artificial Intelligence Research 7*, sivut 83-124, 1997.

Tur50 Turing, A., Computing Machinery and Intelligence. *Mind 49*, sivut 433-460, 1950.