T-106.5800 Satunnaisalgoritmit-seminaari, kevät 2008, TKK
Esitys pelitekniikoista 19.3.2008 (kirjoitus valmistunut 7.4.2008)
Lauri Lahti

Pääasiallisena lähteenä:
Motwani, R., Raghavan, P.: Randomized algorithms. Cambridge: Cambridge University Press, 1997, 28–42.

1 Johdanto

Tässä esityksessä tarkastellaan satunnaisalgoritmien hyödyntämistä pelitilanteissa. Erityiseen tarkasteluun tulevat pelipuut, nollasummapeli ja minmax-periaate. Esitys pyrkii tarjoamaan näkökulman, joka nivoo deterministisen ja satunnaistetun pelitilanteiden käsittelyn toisiinsa. Tarkastelun alussa tutustumme peruskäsitteisiin, kuten pelipuihin ja niiden arviointiin. Pelipuiden arviointiin perustuva tarkastelu pyrkii havainnollistamaan kolmea asiaa:

- yksinkertaista lineaarisesti käsiteltävää tapausta (linearity of expectation)
- satunnaisalgoritmia, jonka suoritusajan odotusarvo on pienempi kuin millään deterministisellä algoritmilla
- standardimenetelmää, jolla voidaan johtaa alaraja mille tahansa ongelman ratkaisevalle satunnaisalgoritmille

2 Pelipuiden arviointi

Pelipuu (game tree) on juurellinen puu (rooted tree), jonka sisäsolmut parittoman määrän kaaria sisältävän polun etäisyydellä juuresta on nimetty MIN-solmuiksi ja parillisen määrän kaaria sisältävän polun etäisyydellä juuresta on nimetty MAX-solmuiksi. Puun kuhunkin lehteen on tallennettuna jokin lukuarvo, eräänlainen hyötyfunktion arvo, joka edustaa juuresta tähän lehteen kuljettavan polun arvoa. Pelipuuta käsiteltäessä jokainen lehti palauttaa sisältämänsä arvon: jokainen MAX-solmu palauttaa suurimman arvon, jonka jokin sen lapsista on palauttanut ja jokainen MIN-solmu palauttaa pienimmän arvon, jonka jokin sen lapsista on palauttanut. Pelipuun arviointiongelmana on määrittää juuren palauttama arvo, joka siis perustuu juuren alipuun sisältämien solmujen palauttamiin arvoihin.

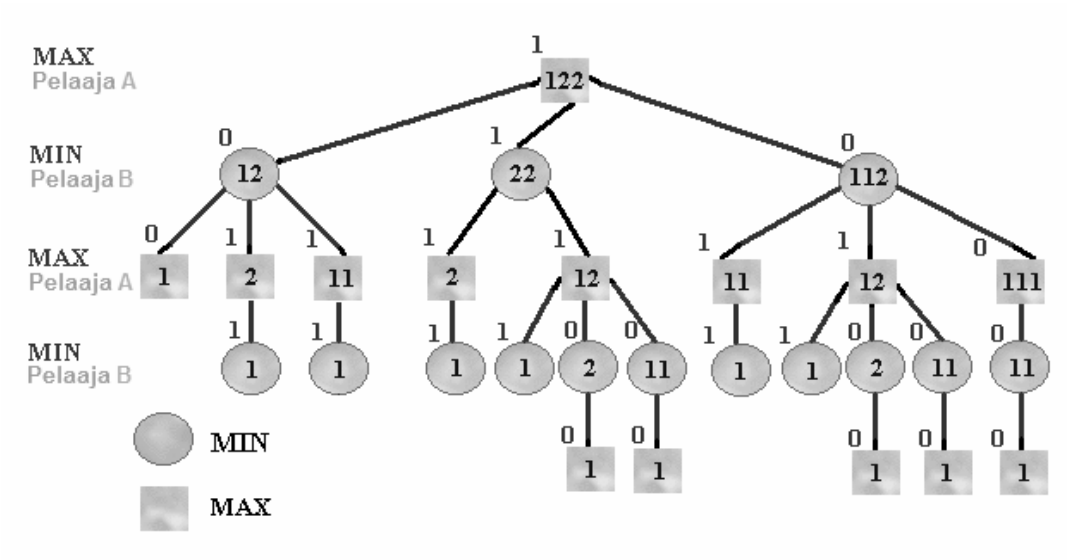
Pelipuiden arvioinnilla (game tree evaluation) on merkittävä tehtävä tekoälysovelluksissa, erityisesti tietokonepeleissä. Solmun lapset voidaan ajatella kahden pelaajan (pelaaja A ja pelaaja B) pelissä kummankin vuorotellen tekemien nk. siirtojen valintamahdollisuuksina. Puun juuri edustaa pelin lähtötilannetta. Puun lehden edustavat pelin vaihtoehtoisia lopputilanteita, jommankumman pelaajan voittoa tai tasapeliä. Käytännössä lopputilanne kannattaa ilmaista lehteen tallennetulla lukuarvolla, joka suurella arvolla tarkoittaa pelaajan A voittoa ja pienellä arvolla pelaajan B voittoa. Loppuratkaisun arvon voidaan ajatella kehittyvän vaiheittain pelivuorojen myötä. Kumpikin pelaaja pyrkii omilla vuoroillaan edistämään omaan voittoonsa pääsyä, joten myös välivaiheita edustaviin solmuihin pelaaja A pyrkii saamaan mahdollisimman suuria arvoja ja pelaaja B mahdollisimman pieniä arvoja.

Puun juuri edustaa pelin lähtötilannetta ja jokainen juuren alipuuhun sisältyvä syvyyssuuntainen polku edustaa yhtä mahdollista pelaajan A ja pelaajan B pelivuorojen ketjua pelin alusta loppuun. Kahden pelaajan vuorotellen tekemiin siirtoihin perustuvalle pelille, jossa kaikki mahdolliset pelivuorojen ketjut voidaan määritellä lähtötilanteesta lopputilanteeseen, voidaan laatia täydellinen pelipuu (complete game tree). Pelipuun parilliset tasot (mukaan lukien juuri eli taso 0) voidaan tällöin ajatella pelaajan A pelivuoroon kuuluvaksi valinnaksi ja parittomat tasot pelaajan A

pelivuoroon kuuluvaksi valinnaksi. Koska kullakin pelivuorollaan pelaaja A pyrkii maksimoimaan välivaiheen arvoa, voidaan kaikki pelaajan A pelivuoroja vastaavilla tasoilla olevat solmut nimetä MAX-solmuiksi. Vastaavasti koska kullakin pelivuorollaan pelaaja B pyrkii minimoimaan välivaiheen arvoa, voidaan kaikki pelaajan B pelivuoroja vastaavilla tasoilla olevat solmut nimetä MIN-solmuiksi.

3 Pelipuun ratkaiseminen deterministisellä algoritmilla

Täydellisen pelipuun tapauksessa on mahdollista nk. ratkaista peli eli etsiä pelipuusta kaikki sellaiset syvyyssuuntaiset polut eli pelivuorojen ketjut, joita pitkin etenemällä varmistetaan jommallekummalle pelaajalle voitto. Pelin ratkaiseminen voidaan suorittaa deterministisellä algoritmilla etenemällä puun lehdistä kohti juurta taso kerrallaan. Aluksi on kuitenkin tiedettävä puun lehdille kuuluvat arvot eli vaihtoehtoisia lopputilanteita edustavat lukuarvot. Esimerkiksi pelaajan A voittoa voitaisiin merkitä lehdessä arvolla 1 ja pelaajan B voittoa arvolla 0. Jokaisella tasolla, joka koostuu MAX-solmuista, merkitään jokaisen solmun arvoksi maksimiarvo sen lasten arvoista. Vastaavasti jokaisella tasolla, joka koostuu MIN-solmuista, merkitään jokaisen solmun arvoksi minimiarvo sen lasten arvoista. Kuvassa 1 on esimerkki pelipuusta tikkujen poimintapelille.



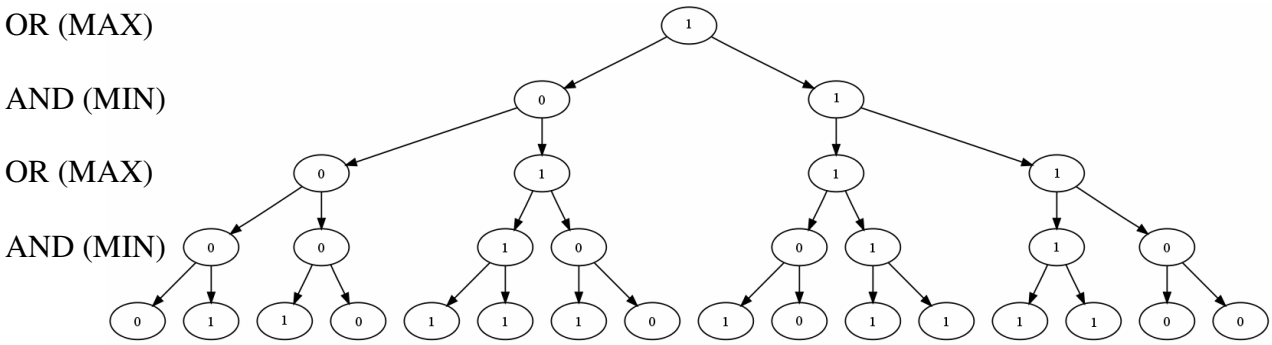
Kuva 1. Pelipuu tikkujen poimintapelille, jossa on aluksi kolme kasaa tikkuja: yksi tikku ensimmäisessä, kaksi tikkua toisessa ja kaksi tikkua kolmannessa. Kukin pelaaja saa vuorollaan poistaa yhden tai useamman tikun yhdestä kasasta. Se pelaaja häviää, jolle jää pelin viimeinen tikku. Kunkin solmun sisään on kirjoitettu tikkujen määrät eri kasoissa kyseisessä pelin vaiheessa, juureen pelin alkutilanne ja lehtiin lopputilanne (esim. 122, 12 tai 1). Solmujen viereen on kirjoitettu pelipuun arvioinnissa käytetyt minimiarvot (MIN-solmuille) ja maksimiarvot (MAX-solmuille). (Mukailtu lähteestä http://www.cs.mcgill.ca/~cs251/OldCourses/1997/topic11/)

On hyödyllistä selvittää, kuinka monta askelta algoritmilla on suoritettava pelipuun ratkaisemiseksi. Seuraavissa tarkasteluissa ei algoritmille oleteta syntyvän muuta ylimääräistä laskennallista kuormitusta. Rajoitumme tarkastelemaan tapausta, jossa lehdet sisältävät yksinkertaisimpia boolean-arvoja eli vain joko arvon 0 tai 1. Täten jokainen MIN-solmu voidaan ajatella lapsiaan käsitteleväksi boolean-operaatioksi AND ja jokainen MAX-solmu voidaan ajatella lapsiaan käsitteleväksi boolean-operaatioksi OR. Tällä erityistapauksella on oma merkityksensä mm. mekaanisessa teoreeman todistamisessa (mechanical theorem proving). Kuvassa 2 esitellään mahdolliset vaihtoehdot boolean-algebrallisille funktioille AND and OR.

Syötteet	X	0	0	1	1
	Y	0	1	0	1
Funktion arvot	X AND Y	0	0	0	1
	X OR Y	0	1	1	1

Kuva 2. Boolean-algebralliset funktiot AND and OR (mukaeltu lähteestä http://en.wikipedia.org/wiki/Logic_gate).

Puuta, jonka solmuissa ei ole käytettävissä mitään solmukohtaista lisätietoa, kutsutaan tasalaatuiseksi (uniform). Merkitään ilmaisulla $T_{d,k}$ tasalaatuista *AND-OR-puuta*, jossa juurella ja jokaisella solmulla on d lasta ja jokainen lehti on etäisyydellä 2k juuresta. Täten polku juuresta lehteen kulkee k:n AND-solmun (sisältäen juuren) kautta ja k:n OR-solmun kautta, ja lehtiä on d^{2k}. Pelipuun yksittäinen arviointitapaus sisältää puun $T_{d,k}$ sekä boolean-arvot jokaiselle d^{2k}:lle lehdelle. Tutkitaan algoritmeja, joilla voidaan selvittää kuinka monta askelta enintään on suoritettava minkä tahansa arviointitapauksen ratkaisemiseksi puulle $T_{d,k}$. Kuvassa 3 on esimerkki puusta $T_{d,k}$.

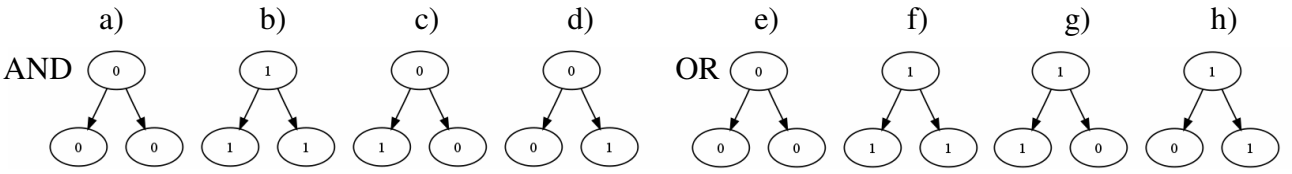


Kuva 3. Tasalaatuinen puu $T_{2,2}$.

Algoritmi aloittaa määrittämällä lehden, jonka arvon se lukee ensimmäisellä askeleella. Tämän jälkeen algoritmi määrittää sellaisen lehden jokaisella askeleella perustuen edellisillä askelilla luettuihin arvoihin. Deterministisessä algoritmissa seuraavan luettavaksi otettavan lehden valinta on deterministinen funktio toistaiseksi luettujen lehtien arvoista. Satunnaisalgoritmien tapauksessa tämä valinta voidaan satunnaistaa. Jokaiselle deterministiselle arviointialgoritmille on olemassa arviointitapaus $T_{d,k}$, joka vaatii algoritmia lukemaan arvot kaikista d^{2k}:sta lehdestä.

4 Pelipuun ratkaiseminen satunnaisalgoritmilla

Seuraavaksi tarkastelemme satunnaisalgoritmiin perustuva menetelmää ja selvitämme, kuinka monta lehteä se lukee arviointitapauksessa $T_{d,k}$. Tarkastelun yksinkertaistamiseksi rajoitutaan tapaukseen d=2. Tässä tapauksessa deterministisellä algoritmilla tulisi luettavaksi $2^{2k} = (2^2)^k = 4^k$ lehteä arviointitapauksessa $T_{2,k}$. Satunnaisalgoritmi perustuu suhteellisen yksikertaisiin havaintoihin AND- ja OR-solmujen tapauksista (kuva 4).



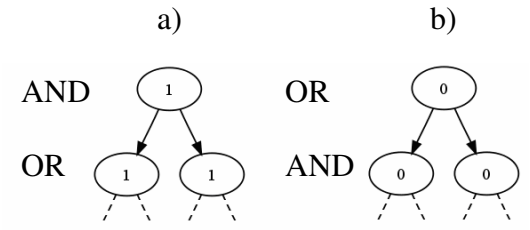
Kuva 4. AND- ja OR-solmujen erilaisten lehtien tapaukset.

Voidaan tarkastella kaksilehtisen AND-solmun tapausta, jossa voidaan ajatella, että vastapelaaja pyrkii minimoimaan pelaajan arvoa. Jos AND-solmu palauttaa arvon 0, ainakin toisen lehdistä on oltava arvoltaan 0 (kuvat 4a, 4c ja 4d). Deterministinen algoritmi tarkastaa lehtiä kiinnitetyssä järjestyksessä ja näin ollen on mahdollista, että tarkastukseen tulevista kahdesta lehdestä arvon 0 sisältävä (pelaajan kannalta tavoiteltavampi lehti) on vasta jälkimmäiseksi tarkastettava, nk. ”jää piiloon”. Lukemalla nämä kaksi lehteä satunnaisessa järjestyksessä saadaan mahdollisuus selviytyä tästä rajoitteesta. Satunnaisalgoritmi valitsee todennäköisyydellä 0,5 ”piilossa olevan” arvon 0 sisältävän lehden heti ensimmäisellä askeleella. Tällöin odotusarvo tarvittavien askelten määrälle laskee 2:sta 1,5:een. Täten satunnaisalgoritmin toimii edullisemmin kuin deterministinen algoritmi pahimmassa tapauksessa.

Vastaavasti voidaan tarkastella kaksilehtisen OR-solmun tapausta, jossa voidaan ajatella, että pelaaja (nyt ei siis enää vastapelaaja) pyrkii maksimoimaan omaa arvoaan. Jos OR-solmu palauttaa arvon 1, ainakin toisen lehdistä on oltava arvoltaan 1 (kuvat 4f, 4g ja 4h). Deterministinen algoritmi tarkastaa lehtiä kiinnitetyssä järjestyksessä ja näin ollen on mahdollista, että tarkastukseen tulevista kahdesta lehdestä arvon 1 sisältävä (pelaajan kannalta tavoiteltavampi lehti) on vasta jälkimmäiseksi tarkastettava, nk. ”jää piiloon”. Lukemalla nämä kaksi lehteä satunnaisessa järjestyksessä, kuten tehtiin juuri edellä, saadaan mahdollisuus selviytyä tästä rajoitteesta. Satunnaisalgoritmi valitsee todennäköisyydellä 0,5 ”piilossa olevan” arvon 0 sisältävän lehden heti ensimmäisellä askeleella. Tällöin odotusarvo tarvittavien askelten määrälle laskee 2:sta 1,5:een. Täten satunnaisalgoritmin toimii edullisemmin kuin deterministinen algoritmi pahimmassa tapauksessa.

Tarkastellaan nyt vielä jäljelle jääneitä tilanteita. Jos kaksilehtinen AND-solmu palauttaa arvon 1 (kuva 4b) tai kaksilehtinen OR-solmu palauttaa arvon 0 (kuva 4e), on nämä molemmat lehdet käytävä läpi. Satunnaisesta läpikäyntijärjestyksestä on kuitenkin yhä hyötyä, erityisesti jos huomioidaan myös vastaavanlaiset AND- ja OR-sisäsolmut (puun solmut joiden lapset eivät ole lehtiä) ja niiden alipuut. Tutkitaan tämä sisäsolmuja koskevat tilanteet seuraavaksi.

- Jos kaksilapsinen AND-sisäsolmu palauttaa arvon 1, sen molempien OR-lapsien on oltava arvoltaan 1 (kuva 5a). Näiden OR-lapsien kannalta tilanteen jatkokäsittely on suotuisaa ja ne vuorostaan voidaan tarkastella, kuten edellä jo tehtiin.
- Jos kaksilapsinen OR-sisäsolmu palauttaa arvon 0, sen molempien AND-lapsien on oltava arvoltaan 0 (kuva 5b). Näiden AND-lapsien kannalta tilanteen jatkokäsittely on suotuisaa ja ne vuorostaan voidaan tarkastella, kuten edellä jo tehtiin.



Kuva 5. AND- ja OR-sisäsolmuja.

5 Satunnaisalgoritmin toiminta

Satunnaisalgoritmin toimintaperiaate kokonaisuudessaan on seuraavanlainen. Algoritmi voi kohdata sekä AND- että OR-solmuja, jotka voivat olla myös sisäsolmuja. Sisäsolmujen tapauksessa kuvan 4 esittämät tilanteet sisältäisivät alaspäin jatkuvia kaaria, kuten katkoviivalla on ilmaistu kuvassa 5.

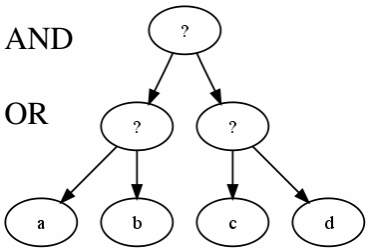
- Jos algoritmi kohtaa AND-sisäsolmun, se valitsee satunnaisesti toisen AND-sisäsolmun lapsista (alipuun, jolla on OR-solmu juurenaan) ja tarkastaa sen kutsuen rekursiivisesti algoritmia. Jos alipuu palauttaa arvon 1, algoritmi jatkaa arvioimaan AND-sisäsolmun toista lasta ja kutsuu taas rekursiivisesti algoritmia. Jos alipuu palauttaa arvon 0, algoritmi palauttaa AND-sisäsolmulle arvon 0.
- Jos algoritmi kohtaa OR-sisäsolmun, se valitsee satunnaisesti toisen OR-sisäsolmun lapsista (alipuun, jolla on AND-solmu juurenaan) ja tarkastaa sen kutsuen rekursiivisesti algoritmia. Jos alipuu palauttaa arvon 0, algoritmi jatkaa arvioimaan OR-sisäsolmun toista lasta ja kutsuu taas rekursiivisesti algoritmia. Jos alipuu palauttaa arvon 1, algoritmi palauttaa OR-sisäsolmulle arvon 1.

6 Satunnaisalgoritmin keskimääräinen vaativuus

Induktion perusteella voidaan osoittaa, että minkä tahansa arviointitapauksen $T_{2,k}$ keskimääräinen vaativuus on suuruudeltaan enintään 3^k.

Tapauksessa k=1 todistaminen voidaan perustaa vähän laajennettuun versioon edellä olevasta tarkastelusta. Kuvassa 6 esitetään AND-sisäsolmun arvon palauttamisessa kohdattavat vaihtoehtoiset lasten tapaukset. Tapauksista osa edellyttää kahden, osa kolmen ja osa neljän solun tarkastamista. Näiden vaihtoehtoisten tapausten jakauma osoittaa, että AND-sisäsolmun ollessa juuressa keskimäärin tarvitaan enintään kolme tarkastusta arviointitapaukselle $T_{2,1}$ eli vaativuus on keskimäärin enintään 3. Vastaavasti voidaan osoittaa, että myös OR-sisäsolmun ollessa juuressa, että keskimäärin tarvitaan enintään kolme tarkastusta arviointitapaukselle $T_{2,1}$ eli vaativuus on keskimäärin enintään 3.

2 solmua tarkastettava (a, b, c, d)	3 solmua tarkastettava (a, b, c, d)	4 solmua tarkastettava (a, b, c, d)
0,**0**,0,0 **0**,**0**,0,1 **0**,**0**,1,0 **0**,**0**,1,1 **1**,0,**1**,0 **1**,0,**1**,1 **1**,1,**1**,0 **1**,1,**1**,1	**0**,**1**,**1**,0 **0**,**1**,**1**,1 **1**,0,**0**,**0** **1**,0,**0**,**1** **1**,1,**0**,**0** **1**,1,**0**,**1**	**0**,**1**,**0**,**0** **0**,**1**,**0**,**1**
Yht. 16 askelta	Yht. 18 askelta	Yht. 8 askelta



Kuva 6. AND-sisäsolmun arvon palauttamisessa kohdattavat vaihtoehtoiset lasten tapaukset.

Seuraavaksi tehdään induktio-oletus, että arviointitapaukselle $T_{2,k-1}$ vaativuus on keskimäärin enintään 3^{k-1}. Tämän jälkeen kehitetään induktioaskel tilanteesta k-1 tilanteeseen k.
Tarkastellaan ensin puuta T, jonka juurena on OR-solmu ja jonka lapset (AND-solmuja) ovat juuria alipuun $T_{2,k-1}$ kopioille. Jos juuren arvoksi palautuisi 1, ainakin yhden lapsista täytyy palauttaa 1. Todennäköisyydellä 0,5 tämä lapsi tulee valituksi ensimmäisenä tuottaen (induktio-oletuksen perusteella) keskimäärin vaativuuden, joka on enintään 3^{k-1} arvioitaessa puuta $T_{2,k}$. Todennäköisyydellä 0,5 molemmat alipuut täytyy tarkastaa tuottaen keskimäärin vaativuuden $2 \times 3^{k-1}$ arvioitaessa puuta $T_{2,k}$. Yhdessä nämä tulokset muodostavat keskimäärin vaativuuden

$$0{,}5 \times 3^{k-1} + 0{,}5 \times 2 \times 3^{k-1} = 3/2 \times 3^{k-1} \quad \text{(tämä on kaava 1)}$$

Jos OR-solmu palauttaa arvon 0, täytyy molempien lasten tulla tarkastetuiksi tuottaen keskimäärin vaativuuden

$$2 \times 3^{k-1} \quad \text{(tämä on kaava 2)}$$

Tarkastellaan seuraavaksi puun $T_{2,k}$ juuressa olevaa AND-solmua. Jos AND-solmua palauttaa arvon 1, sen molempien alipuiden juuressa olevien OR-solmujen täytyy palauttaa 1. Edellä olevan OR-solmua koskevan tarkastelun (kaava 1) ja odotusarvon lineaarisuuden perusteella voidaan johtaa, että tapaukselle, jossa $T_{2,k}$ palauttaa arvon 1, muodostuu keskimäärin vaativuus, joka on enintään $2 \times 3/2 \times 3^{k-1} = 3^k$. Toisaalta tapauksessa, jossa $T_{2,k}$ palauttaa arvon 0, ainakin toisen sen alipuiden juuressa olevista OR-solmuista täytyy palauttaa 0. Todennäköisyydellä 0,5 tämä OR-solmu tulee valituksi ensimmäisenä.

Kaikkiaan saadaan osoitetuksi arviointitapaukselle $T_{2,k}$ vaativuus, joka on keskimäärin enintään:

$$2 \times 3^{k-1} + 1/2 \times 3/2 \times 3^{k-1} \leq 3^k$$

Tässä ensimmäinen termi (kaava 2) ilmaisee vaativuutta arvioida alipuita, joiden juuressa oleva OR-solmu palauttaa arvon 0. Toinen termi (kaava 1) ilmaisee todennäköisyydellä 0,5 esiintyvää vaativuuden osaa, joka syntyy siitä, että joudutaan arvioimaan OR-solmun sisarusta, joka palauttaa arvon 1.

7 Alaraja ja yläraja satunnaisalgoritmin vaativuudelle

Koska puussa olevien lehtien lukumäärä $n = 4^k$, tämän satunnaistetun algoritmin keskimääräinen suoritusaika on $n^{\log_4 3}$, mille voidaan asettaa yläraja $n^{0{,}793}$. Täten odotusarvo algoritmin suorittamille askeleille on pienempi kuin deterministisen algoritmin pahimmalle tapaukselle. Tässä

tarkasteltava algoritmi edustaa Las Vegas -algoritmeja ja tuottaa aina oikean ratkaisun. Satunnaisalgoritmi tarkastaa tasalaatuisia binäärisiä AND-OR-puita. On hyödyllistä pyrkiä selvittämään, voiko millään satunnaisalgoritmilla saavuttaa pienempää keskimääräistä suoritusaikaa. Tätä varten voidaan etsiä yleistä alarajaa satunnaisalgoritmin suoritusajalle kyseisessä ongelmassa. Sivuamme seuraavassa tarkastelussa peliteoriaa (game theory), joka tarjoaa yleisiä menetelmiä, jolla voidaan selvittää tarkemmin pelipuun arviointiongelmaa. Nk. minimax-periaate on osoittautunut ainoaksi tunnetuksi tekniikaksi todistaa alarajoja satunnaisalgoritmien suoritusajoille. Tämä tekniikka pätee vain algoritmeille, jotka päättyvät äärellisessä ajassa.

8 Nollasummapelit ja voittomatriisi

Jatkotarkastelua varten eräs hyödyllinen pelityyppi on kahden pelaajan *nollasummapeli* (zero-sum game). Nollasummapeleissä keskimäärin molemman pelaajat voittavat yhtä paljon. Näissä peleissä voidaan hyödyntää *voittomatriisia* (payoff matrix), niin kutsutta voittotaulua, jossa esitetään pelaajien voitoista saamat palkkiot ja häviöistä saamat sakot. Käytännössä voidaan sopia, että häviävä pelaaja luovuttaa saamansa sakon verran pisteitä (esim. pelimerkkejä) voittavalle pelaajalle palkkioksi. Eräs yksinkertainen nollasummapeli on kivi-paperi-sakset-peli, jossa kaksi pelaajaa näyttää toisilleen samanaikaisesti käsimerkkejä, jotka tiettyjen voimasuhteiden mukaan voittavat tai häviävät (esim. paperi voittaa kiven, kivi voittaa sakset ja sakset voittaa paperin). Kuvassa 7 on eräs mahdollinen voittomatriisi kivi-paperi-sakset-peliin. Positiiviset luvut ilmaisevat voitonmaksua pelaajalta A pelaajalle B, negatiiviset pelaajalta B pelaajalle A.

Pelaaja A Pelaaja B	Kivi	Paperi	Sakset
Kivi	0	1	-1
Paperi	-1	0	1
Sakset	1	-1	0

Kuva 7. Eräs mahdollinen voittomatriisi kivi-paperi-sakset-peliin.

Nollasummapelin voittomatriisia voidaan esittää matriisilla M. Pelaajaa, jonka käsimerkit sijoittuvat riveille, kutsutaan *rivipelaajaksi*, ja vastaavasti pelaajaa, jonka käsimerkit sijoittuvat sarakkeisiin, *sarakepelaajaksi*. Valittavaa käsimerkkiä voidaan kutsua valituksi *strategiaksi*. Sovitaan, että matriisin alkio M_{ij} ilmaisee palkkion, jonka sarakepelaaja maksaa rivipelaajalle, kun sarakepelaaja valitsee strategian j ja rivipelaaja strategian i. Voidaan, ajatella rivipelaaja pyrkii maksimoimaan sarakepelaajalta saamaansa voittoa ja toisaalta sarakepelaaja pyrkii minimoimaan voiton maksamista rivipelaajalle. Tätä voidaan tehdä seuraavanlaisia johtopäätöksiä.

9 Optimaalinen strategia nollasummapelissä

Oletetaan, että kyseessä on nollasummapeli, jossa kummallakaan pelaajalla ei ole tietoa vastustajansa strategiasta. Jos rivipelaaja valitsee strategian i, hänen taataan saavan voittoa sarakepelaajalta $\min_j M_{ij}$ riippumatta sarakepelaajan strategiasta. *Optimaalinen strategia rivipelaajalle* on sellainen, joka maksimoi arvon $\min_j M_{ij}$. Merkitään täten rivipelaajan saatavaksi sarakepelaajalta tulevan voiton alarajaa $V_R = \max_i \min_j M_{ij}$. Vastaavasti jos sarakepelaaja valitsee strategian j, hänen taataan maksavan voittoa rivipelaajalle $\max_i M_{ij}$ riippumatta rivipelaajan strategiasta. *Optimaalinen strategia sarakepelaajalle* on sellainen, joka minimoi arvon $\max_i M_{ij}$. Merkitään täten sarakepelaajalta rivipelaajalle maksettavaksi tulevan voiton ylärajaa $V_s = \min_j \max_i M_{ij}$.

Voidaan osoittaa, että kaikille voittomatriiseille pätee:

$\max_i \min_j M_{ij} \leq \min_j \max_i M_{ij}$.

Yleisesti epäyhtälö on tiukka, esimerkiksi kivi-paperi-sakset-pelissä $V_R = -1$ ja $V_s = 1$. Kun nämä kaksi rajaa, alaraja ja yläraja ovat yhtä suuret, sanotaan pelillä olevan *ratkaisu* ja pelin *arvo* on $V = V_R = V_s$. Ratkaisu, nk. *satulapiste*, edellyttää sellaisten optimaalisten strategioiden valintaa, jotka johtavat tavoiteltavaan voitonmaksuun. Merkitään peleissä, joilla on ratkaisu, optimaalista strategiaa rivipelaajalle merkinnällä ρ ja sarakepelaajalle γ. Tällöin voidaan merkitä $V = M_{\rho\gamma}$. Yleisessä tapauksessa pelaajalla voi olla useita optimaalisia strategioita.

10 Deterministiset strategiat ja satunnaistetut strategiat

Jos pelillä ei ole ratkaisua (ks. ratkaisun olemassaolon määritelmää edellisessä kappaleessa), kenellekään pelaajalle ei ole selkeää strategiaa. Tällöin mitä tahansa tietoa vastustajan strategiasta voidaan käyttää parantamaan voitonmaksua, toisin kuin satulapisteeseen johtavissa peleissä. Tähän tilanteessa satunnaisuuden mukaan tuominen tuo hyödyllisiä ominaisuuksia strategian valintaan. Tähän asti tarkastelussa ovat olleet *deterministiset strategiat*, mutta nyt käyttöön otetaan *satunnaistettuja strategioita*. Tätä siirtymää voidaan kuvata myös sanomalla, että siirrytään *puhtaista strategioista* (pure strategies) *sekastrategioihin* (mixed strategies). Sekastrategiat perustuvat todennäköisyysjakaumiin, joilla määrätään vaihtoehtoisten strategioiden valitsemisen todennäköisyyksiä.

Rivipelaajan suorittamaan strategian valintaan voidaan kiinnittää vektori $p = (p_1, \ldots, p_n)$, joka on todennäköisyysjakauma voittomatriisin riveille. Täten p_i on todennäköisyys sille, että rivipelaaja valitsee strategian i. Vastaavasti sarakepelaajan suorittamaan strategian valintaan voidaan kiinnittää vektori $q = (q_1, \ldots, q_n)$, joka on todennäköisyysjakauma voittomatriisin sarakkeille. Täten q_j on todennäköisyys sille, että sarakepelaaja valitsee strategian j.

Pelissä syntyvä voitto on nyt satunnaismuuttuja ja sen odotusarvo voidaan ilmaista kaavalla

$E(\text{voitto}) = p^T M q$

Voidaan jatkaa tarkastelua samaan tapaan kuin aiemmin. Voidaan merkitä V_R:llä suurinta mahdollista alarajaa voiton odotusarvolle, jonka rivipelaaja voi saada sarakepelaajalta valitsemalla strategian p. Samaten voidaan merkitä V_S:llä pienintä mahdollista ylärajaa voiton odotusarvolle, jonka sarakepelaaja voi maksaa rivipelaajalle valitsemalla strategian q.
Tällöin saadaan $V_R = \max_p \min_q p^T M q$ ja $V_s = \min_q \max_p p^T M q$

11 Optimaaliset sekastrategiat ja etukäteistieto vastustajasta

von Neumannin minimax-teoreeman perusteella voidaan todeta, että tällä pelillä on aina ratkaisu ja että $V_R = V_s$. Teoreeman mukaan jokaiselle kahden hengen nollasummapelille pätee

$\max_p \min_q p^T M q = \min_q \max_p p^T M q$.

Tämä tarkoittaa sitä, että suurin alaraja rivipelaajan saatavaksi tulevalle voitolle hänen käyttäessään sekastrategiaa on yhtä suuri kuin pienin yläraja sarakepelaajan maksettavaksi tulevalle voitolle hänen käyttäessään sekastrategiaa. Tämä voitonmaksun odotusarvo on pelin arvo V. Strategiapari (p_s,q_s), joka maksimoi yhtälön vasemman ja minimoi sen oikean puolen on satulapiste ja todennäköisyysjakaumat p_s ja q_s ovat *optimaalisia sekastrategioita*.

Kun p on kiinnitetty, $p^T M q$ on lineaarinen funktio q:n suhteen ja voidaan minimoida asettamalla arvo 1 sille todennäköisyydelle q_j, jolla on pienin kerroin tässä lineaarisessa funktiossa. Täten, jos sarakepelaaja tietää rivipelaajan käyttämän todennäköisyysjakauman p, sarakepelaajan strategia

muodostuu puhtaaksi strategiaksi. Vastaavasti jos rivipelaaja tietää sarakepelaajan käyttämän todennäköisyysjakauman p, rivipelaajan strategia muodostuu puhtaaksi strategiaksi. Tämä johtaa pelkistettyy versioon minimax-teoreemasta, jota kutsutaan *Loomisin teoreemaksi*.

$\max_p \min_j p^T M e_j = \min_q \max_i e_i^T M q,$

missä e_k tarkoittaa yksikkövektoria, jonka k:s elementti on suuruudeltaan yksi ja muut nollia.

12 Voittomatriisin yleistys satunnaisalgoritmeihin

Tätä tarkastelua voidaan hyödyntää aiemmin esiteltyjen satunnaisalgoritmeihin liittyvien periaatteiden kanssa. Pyrkimyksenä on määrittää alaraja satunnaisalgoritmien vaativuudelle. Ajatuksena on tarkastella algoritmin suunnittelijaa sarakepelaajana ja algoritmin syötteiden valintaprosessia rivipelaajana. Voittomatriisin sarakkeet vastaavat kaikkia mahdollisia deterministisiä oikean ratkaisun tuottavia algoritmeja ja rivit kaikkia mahdollisia, joskin äärellisiä, syötteitä. Voitonmaksu sarakepelaajalta rivipelaajalle voidaan nähdä reaaliarvoisena mittalukuna algoritmin vaativuudesta (esim. suoritusajasta). Algoritmin suunnittelija pyrkii minimoimaan voiton maksamista (suoritusajan antamista) algoritmin syötteille ja algoritmin syötteiden valintaprosessi puolestaan pyrkii maksimoimaan algoritmin suunnittelijalta saamaansa voittoa (suoritusajan saamista).

von Neumannin minimax-teoreeman ja Loomisin teoreeman pohjalta voidaan satunnaisalgoritmeille muotoilla seuraava tulos.
Olkoon I_{set} äärellinen joukko äärellisiä syötteitä ja A_{set} äärellinen joukko deterministisiä algoritmeja. Syötteellä $I \in I_{set}$ ja algoritmilla $A \in A_{set}$ merkitään C(I,A):lla suoritusaikaa algoritmilla A syötteellä I. Olkoon p todennäköisyysjakauma syötejoukon strategioille ja q todennäköisyysjakauma algoritmien strategioille. Merkitään I_p:llä satunnaista syötettä, joka on valittu satunnaisjakauman p mukaan ja A_q:lla satunnaisalgoritmia, joka on valittu satunnaisjakauman q mukaan. Tällöin pätee:

$\max_p \min_q E(C(I_p,A_q)) = \min_q \max_p E(C(I_p,A_q))$

Tämä voidaan muotoilla myös muotoon:

$\max_p \min_A E(C(I_p,A)) = \min_q \max_I E(C(I,A_q))$, kun $A \in A_{set}$ ja $I \in I_{set}$

13 Satunnaisalgoritmin vaativuus determinististen algoritmien pohjalta

Tämän tuloksen pohjalta voidaan määritellä *Yaon minimax-periaate*:
Kaikille todennäköisyysjakaumille p yli syötejoukon I_{set} ja todennäköisyysjakaumille q yli algoritmijoukon A_{set} pätee:

$\min_A E(C(I_p,A)) \leq \max_I E(C(I,A_q))$, kun $A \in A_{set}$ ja $I \in I_{set}$

Tämä tarkoittaa sitä, että alaraja optimaalisen Las Vegas -satunnaisalgoritmin suoritusajan odotusarvolle saadaan optimaalisen deterministisen algoritmin suoritusajan odotusarvosta syötteen ollessa vapaasti valitun satunnaisjakauman p mukainen. Hieman erilainen, mutta vastaavanlainen alarajaa koskeva tulos voidaan määritellä myös Monte Carlo -satunnaisalgoritmeille.

14 Satunnaisalgoritmin soveltaminen pelipuun ratkaisemiseen

Yaon minimax-periaatetta voidaan soveltaa pelipuun arviointiin. Pelipuun arvioinnissa käytettävä satunnaisalgoritmi voidaan tulkita todennäköisyysjakauman hallinnoimaksi joukoksi deterministisiä algoritmeja. Satunnaisalgoritmin suorituksen kesto ja kussakin valintatilanteessa kohdattavien vaihtoehtojen määrä on äärellinen. Voidaan olettaa, että jokaisen suoritushaaran todennäköisyydet ovat kiinnitettyjä, kun algoritmin suoritus alkaa.

Saatuja tuloksia voidaan soveltaa AND-OR-puuhun $T_{2,k}$. Käsittelyteknisistä syistä johtuen tuloksia on kuitenkin helpompi johtaa AND-OR-puihin rinnastuvilla NOR-puilla. Voidaan osoittaa, että AND-OR-puu $T_{2,k}$ on ominaisuuksiltaan yhtäpitävä NOR-puiden kanssa. *NOR-puut* ovat tasapainotettuja binääripuita, joiden kaikki lehdet ovat etäisyydellä 2k juuresta ja jonka kaikki sisäsolmut toteuttavat NOR-funktion (ks. kuva 8).

Syötteet	X	0	0	1	1
	Y	0	1	0	1
Funktion arvot	X NOR Y	1	0	0	0

Kuva 8. Boolean-algebrallinen funktio NOR (mukaeltu lähteestä http://en.wikipedia.org/wiki/Logic_gate).

Jokainen NOR-puun solmu saa arvon 1 todennäköisyydellä $p = (3 - 5^{0,5}) / 2$. Tämä p:n arvo osoittautuu päteväksi sillä, että tällöin kyseisen solmun lasten tulee olla arvoltaan 0, jolloin puolestaan pätee $(1 - p)(1 - p) = (1 - (3 - 5^{0,5}) / 2) (1 - (3 - 5^{0,5}) / 2) = (3 - 5^{0,5}) / 2 = p$.
Voidaan osoittaa, että NOR-puulle on olemassa syvyyshakuun perustuva karsinta-algoritmi (depth-first pruning algorithm), jonka askelten odotusarvo on sama kuin deterministisen algoritmien askelmäärien odotusarvojen minimi. Etäisyydellä h puun lehdistä olevan solmun arvon määrittämistä varten tarkastettavien lehtien määrä on $W(h) = W(h-1) + (1-p)(W(h-1))$, missä ensimmäinen termi edustaa ensimmäisen alipuun arviointia ja toinen termi toisen alipuun arviointia. Toinen alipuu tutkitaan vain mikäli ensimmäinen alipuu palauttaa arvon 0, siis todennäköisyydellä (1- p). Kun h:n arvoksi asetetaan binääripuulle tyypillisen läpikäynnin vaativuuden mukaisesti $\log_2 n$, saadaan ratkaistuksi, että $W(h) \geq n^{0,694}$. Täten saadaan tulos, että suoritusajan odotusarvo satunnaisalgoritmille, joka ratkaisee aina oikein puun $T_{2,k}$ on vähintään:

$n^{0,694}$, missä $n=2^{2k}$ on lehtien määrä

Tarkemmassa analyysissä otetaan yksityiskohtaisemmin huomioon lehtien sisältämien arvojen mahdolliset todennäköisyysjakaumat, mutta tässä tarkempi analyysi sivuutetaan.

15 Satunnaistettujen menetelmien yleinen käyttö

Pelipuiden tapaan satunnaistettuja menetelmiä ja niihin liittyviä deterministisiä menetelmiä voidaan hyödyntää myös mm. boolean-verkkojen (boolean circuit) yhteydessä. *Adlemanin teoreeman* mukaan jos boolean funktiolla on kooltaan polynomisesti rajoitettu perhe satunnaisverkkoja, niin sillä on myös kooltaan polynomisesti rajoitettu perhe deterministisiä verkkoja. Adlemanin teoreeman pohjalta voidaan yleisemminkin havaita, että satunnaistetut menetelmät kytkeytyvät vahvasti deterministisiin menetelmiin ja satunnaistetuista menetelmistä voidaan tietyissä tilanteissa tarkoituksellisesti purkaa pois satunnaisuutta. Suoritusajaltaan polynomiaalisesti rajoitetussa laskennassa satunnaisuutta ei voida kuitenkaan yleisessä tapauksessa sivuuttaa tai pelkistää. Tälle löytyy perusteluja kompleksisuusteoriasta (complexity theorem) ja sen kuvailemista yhtenäisistä algoritmeista (uniform algorithms) ja epäyhtenäisistä algoritmeista (non-uniform algorithms) . Tässä kirjoituksessa aihepiiriä ei käydä läpi tämän yksityiskohtaisemmin.