Rinnakkaisten prosessien palauteoppiminen

Jouni Siren

Helsinki 22. huhtikuuta 2002
Älykkäiden järjestelmien tutkimusseminaari

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

1 Johdanto

Palauteoppimista (engl. *reinforcement learning*) voidaan pitää yhtenä koneoppimisen perustyypeistä ohjatun ja ohjaamattoman oppimisen rinnalla. Sen perusasetelma muodostuu äärellistestä tilojen joukosta S, äärellisestä toimintojen joukosta A ja palautefunktiosta $R : S \times A \to \mathbb{R}$. Tavoitteena on joko kokeilemalla tai simuloimalla löytää toimintastrategia $\pi : S \to A$, joka maksimoi saadun palautteen pitkällä aikavälillä. Tässä suhteessa palauteoppiminen eroaa selvästi ohjatusta ja ohjaamattomasta oppimisesta, jotka ovat käsitteinä degeneroituneet tarkoittamaan erilaisten luokitteluongelmien ratkaisemista.

Yllä esitetty perusasetelma liittyy kiinteästi päätösteoriasta lähtöisin oleviin Markov-päätösprosesseihin (MDP) (luku 2.1), joissa siirtymän lopputulos voi olla epävarma, vaikka lähtötilanne ja suoritettu toiminto tunnettaisiin. Niihin on vuosien varrella esitetty lukuisia muunnelmia, joilla voidaan mallintaa esimerkiksi havaintotiedon epävarmuutta, useita rinnakkaisia prosesseja tai pitkitettyjä toimintoja sisäisellä rakenteella tai ilman. Palauteoppimista on tutkittu myös MDP-mallien ulkopuolella.

Tässä esitelmätekstissä keskitytään Markov-päätösprosessien ja niihin liittyvien palauteoppimisalgoritmien laajentamiseen rinnakkaisia prosesseja ja useita peräkkäisiä toimintoja yhdistäviä valintoja varten. Luvussa 2 käsitellään Markov-päätösprosesseja ja niiden Q-oppimista. Luvussa 3 esitellään valinnat ja rinnakkaiset prosessit ja käsitellään oppimisalgoritmien käyttämistä niiden kanssa. Lopuksi luvussa 4 arvioidaan saavutettuja tuloksia ja tulevia kehityssuuntia.

2 Palauteoppimisen perusmallit

Tässä luvussa käsitellään tavallisia Markov-päätösprosesseja ja palauteoppimista niiden yhteydessä. Yksityiskohtaisemmin niitä käsitellään esimerkiksi Kaelblingin, Littmanin ja Mooren artikkelissa [2] ja Mitchellin kirjassa [3].

2.1 Markov-päätösprosessit

Markov-päätösprosessit (engl. *Markov decision process*) ovat yleisluontoinen ja laajalti käytetty tapa mallintaa autonomisen toimijan ympäristöä tilanteissa, joissa toimintojen seurauksiin liittyy epävarmuutta tai epätäsmällisyyttä. Niiden pohjalla on probabilistinen äärellinen automaatti (joita on jostain syystä tapana kutsua Markovin ketjuiksi), eli tilasiirtymien todennäköisyydet perustuvat yksinomaan nykyiseen tilaan ja siinä tehtyyn toimintoon. Tämän automaatin päälle on lisätty palautefunktio, joka pisteyttää suoritetut toiminnot nykyisen tilan perusteella. Päätösprosessiksi MDP:n tekee se, että tavoitteena on löytää toimintastrategia, joka maksimoi pitkällä aikavälillä saadun palautteen odotusarvon.

Määritellään Markov-päätösprosessi nelikkona (S, A, T, R) missä S on äärellinen tilojen joukko, A äärellinen toimintojen joukko, $T : S \times A \times S \to [0, 1]$ siirtymätodennäköisyydet määrittävä funktio ja $R : S \times A \to \mathbb{R}$ palautefunktio. Arvot $T(s, a, s') = P(s' \mid s, a)$ tulkitaan todennäköisyyksiksi siirtyä tilasta s tilaan s', kun suoritetaan toiminto a. Vastaavasti $R(s, a)$ on tilassa s toiminnon a suorittamisesta seuraavan välittömän palautteen arvo. Periaatteessa palaute voisi tulla myös jostain jakaumasta, jolloin palautefunktio kertoisi palautteen odotusarvon parille (s, a).

Toimintastrategia tässä päätösprosessissa on kuvaus $\pi : S \to A$, joka kertoo jokaisessa tilassa, mikä toiminto tulee valita. Jokaiseen toimintastrategiaan liittyy arvofunktio $V^\pi : S \to \mathbb{R}$, joka kertoo tästä tilasta alkaen strategiaa π noudattamalla saatavan pitkän aikavälin palautteen odotusarvon. Optimaalista toimintastrategiaa merkitään π^* ja sen arvofunktiota V^*. Seuraavaksi tarkastellaan kahta keskeistä määritelmää pitkän aikavälin palautteelle.

2.2 Pitkän aikavälin palaute

Yleensä Markov-päätösprosessien kanssa käytetään äärettömän horisontin alennettua palautetta, joka on helposti analysoitavissa ja toteutettavissa. Siinä optimoitava arvofunktio on

$$V^*(s) = \max_{a \in A} \left(R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s') \right) \quad \text{kaikilla } s \in S, \qquad (1)$$

missä γ on alennuskerroin ja $0 \leq \gamma < 1$. Optimaalinen toimintastrategia on luonnollisesti

$$\pi^*(s) = \arg\max_{a \in A} \left(R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s') \right) \quad \text{kaikilla } s \in S.$$

Alennuskertoimen arvoa muuttamalla voidaan määritellä, kuinka paljon korostetaan lyhyen aikavälin ja kuinka paljon pitkän aikavälin palautetta.

Jos järjestelmässä on jokin selkeä takaraja, jonka jälkeen saatua palautetta ei oteta huomioon, on mielekästä siirtyä käyttämään kiinteän takarajan palautetta. Nyt optimaalinen, ajankohdasta t riippuva arvofunktio on

$$V_t^*(s) = \max_{a \in A} \left(R(s, a) + \sum_{s' \in S} T(s, a, s') V_{t+1}^*(s') \right) \quad \text{kaikilla } s \in S.$$

Lisäksi $V_t^*(s) = 0$ kaikilla $s \in S$, kun $t \geq T$ jollain ajankohdalla T. Optimaalinen toimintastrategia määritellään arvofunktion perusteella luonnollisella tavalla.

Tässä lähestymistavassa tilajoukoksi tulee käytännössä tilojen ja ajan joukkojen ristitulo. Tilajoukko saattaa näin kasvaa liian suureksi käsitellä, jos joudutaan tarkastelemaan toimintaa vähänkin pidemmällä aikavälillä. Lyhyellä aikavälillä se saattaa kuitenkin kannattaa, koska takarajan lähestyessä optimaalinen toimintastrategia saattaa poiketa merkittävästi optimaalisesta päättymättömästä strategiasta.

Muita esitettyjä määritelmiä pitkän aikavälin palautteelle ovat esimerkiksi äärellisen horisontin palaute ja keskimääräinen palaute. Edellisessä tarkastellaan kiinteän etäisyyden päässä tulevaisuudessa olevaan horisonttiin mennessä saatavaa palautetta, jälkimmäisessä taas n askeleen palautteen keskiarvoa kun n lähestyy ääretöntä. Kumpikaan näistä ei ole saavuttanut erityistä suosiota, mihin on osaltaan vaikuttanut se, että niiden laskeminen on äärettömän horisontin alennettua palautetta vaativampaa, eikä niillä ole samanlaista luonnollista tulkintaa kuin kiinteän horisontin palautteella.

2.3 Q-oppiminen

Q-oppiminen on dynaamisen ohjelmoinnin sukuinen menetelmä, jolla voidaan tietyin rajoituksin löytää optimaalinen strategia Markov-päätösprosessissa. Sitä voidaan soveltaa, vaikka siirtymä- ja palautefunktioita ei tunnettaisikaan. Q-oppimisessa otetaan käyttöön laatufunktio $Q^\pi : S \times A \to \mathbb{R}$, joka kertoo saadun palautteen odotusarvon, kun ensin suoritetaan toiminto a tilassa s ja sen jälkeen ruvetaan noudattamaan strategiaa π. Tarkastellaan nyt Q-oppimista äärettömän horisontin mallissa.

Optimaalinen laatufunktio Q^* määritellään optimaalisen strategian kautta. Se voidaan esittää rekursiivisesti muodossa

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a' \in A} Q^*(s',a') \tag{2}$$

kaikilla $s \in S$ ja $a \in A$. Nyt yhtälö (1) voidaan esittää laatufunktion avulla muodossa

$$V^*(s) = \max_{a \in A} Q^*(s,a) \quad \text{kaikilla } a \in A \tag{3}$$

ja optimaalinen strategia määrittää valitsemalla jokaisessa tilassa se toiminto, jolla laatufunktio antaa suurimman arvon.

Merkitään Q:lla laatufunktion nykyistä estimaattia. Q-oppiminen perustuu päivityssääntöön

$$Q(s,a) := (1-\alpha)Q(s,a) + \alpha \left(r + \gamma \max_{a' \in A} Q(s',a') \right), \tag{4}$$

kun toiminnon a suorittaminen tilassa s johti palautteeseen r ja siirtymiseen tilaan s'. Parametri α säätelee oppimisnopeutta ja se yleensä lähestyy ajan kuluessa nollaa. Tyypillinen arvo sille on $\alpha = 1/(1 + n_{s,a})$, missä $n_{s,a}$ kertoo kuinka monta kertaa toiminto a on suoritettu tilassa s. Jos arvot $Q(s, a)$ arvotaan satunnaisesti rajoitetulta väliltä, parametrille α käytetään ylläolevaa arvoa ja kaikkia toimintoja kokeillaan kaikissa tiloissa äärettömän monta kertaa, optimaalinen toimintastrategia π^* löytyy päivityssääntöä (4) käyttämällä todennäköisyydellä 1 [1].

Käytännössä Q-oppimisen soveltamista rajoittaa merkittävästi sen konvergoitumisen hitaus. Jo suhteellisen pienissäkin ympäristöissä optimaalisen strategian löytäminen edellyttää kymmeniätuhansia iteraatioita, monissa todellisissa sovelluksissa vielä huomattavasti enemmän. Tämä aiheuttaa ongelmia erityisesti autonomisten robottien kohdalla, jos palaute- tai siirtymäfunktiota ei tunneta. Tällöin joudutaan turvautumaan ympäristön tutkimiseen sitä huomattavasti nopeamman simuloinnin sijaan.

Todellisissa järjestelmissä käytetäänkin usein erilaisia heuristiikkoja ja esitietoja ympäristöstä tämän ongelman lievittämiseksi. Tila-avaruus voidaan esimerkiksi käsittää muuttujien arvojoukkojen ristitulona, jolloin kunkin toiminnon todelliseksi tila-avaruudeksi voidaan ottaa toiminnon kannalta relevanttien tilamuuttujien virittämä aliavaruus. Näin voidaan lievittää Q-oppimisen yleistyskyvyttömyyden aiheuttamaa suurta aika- ja tilavaativuutta oppimisen aikana. Toisaalta taas saatetaan käyttää esimerkiksi neuroverkkoja laatufunktion approksimointiin sen täsmällisen esittämisen sijaan. Tämä lähestymistapa ei kuitenkaan toimi yleisessä tapauksessa, sillä käytetty neuroverkko ei välttämättä pysty esittämään todellista laatufunktiota riittävän tarkasti, eikä päivityssääntö (4) välttämättä takaa laatufunktion konvergoitumista optimaaliseen laatufunktioon [2].

2.4 Tutkiminen ja hyödyntäminen

Palauteoppimisessa ei tehdä eksplisiittistä eroa oppimisen ja tulosten hyödyntämisen välillä, vaan oppiminen jatkuu koko ajan ja toimija on alusta alkaen ympäristössä, joka sen on tarkoitus oppia. Keskeinen tähän asetelmaan liittyvä ongelma on, kuinka ahneesti toimija lähtee noudattamaan sillä hetkellä parhaana pitämäänsä strategiaa ja kuinka paljon se tutkii muita vaihtoehtoja. Liika ahneus saattaa kostautua optimaalista huonompaan strategiaan juuttumisena, liika tutkiminen taas heikkona suorituskykynä tutkimisen aikana.

Eräs ratkaisu perustuu ympäristön satunnaiseen tutkimiseen. Jokaiselle toiminnolle $a \in A$ määritellään kaikissa tiloissa $s \in S$ siirtymätodennäköisyys

$$P(a \mid s) = \frac{k^{Q(s,a)}}{\sum_{a' \in A} k^{Q(s,a')}}.$$

Tässä $k > 0$ on arvo, jonka pienillä arvoilla painotetaan tutkimista ja suurilla oppien hyödyntämistä. Tyypillisesti k lähtee liikkeelle pienistä arvoista ja kasvaa vähitellen ajan kuluessa. Näin siirrytään vähitellen tutkimisesta kohti oppien hyödyntämistä.

3 Laajennuksia

Tässä luvussa esitellään Markov-päätösprosessien laajennuksia, joiden avulla oppimista pyritään nopeuttamaan ja malleja pienentämään. Aluksi tarkastellaan semi-Markov-päätösprosesseja, joissa toimintojen kestoaika voi vaihdella toiminnosta toiseen, toisin kuin tavallisissa Markov-päätösprosesseissa. Seuraavaksi käsitellään valintoja, jotka ketjuttavat monta toimintoa yhdeksi sarjaksi ja mahdollistavat näin asioiden käsittelemisen korkeammalla abstraktiotasolla. Niiden jälkeen tarkastellaan erästä tapaa käsitellä rinnakkaisuutta ilman, että mallit kasvavat räjähdysmäisesti.

3.1 Semi-Markov-päätösprosessit

Keskeinen ero Semi-Markov-päätösprosesseissa (SMDP, *semi-Markov decision process*) tavallisiin MDP:ihin verrattuna on, etteivät kaikki toiminnot kestä samaa vakioaikaa, vaan kestoaika vaihtelee niin toimintojen välillä kuin sisälläkin. Näin voidaan mallintaa ympäristöjä, joissa jotkut toiminnot (kuten liikkuminen) voivat viedä pitkiäkin aikoja, kun taas jotkut (esimerkiksi esineen poimiminen) ovat vain hetkellisiä.

Formaalisti semi-Markov-päätösprosessi on nelikko (S, A, T, R), joka eroaa tavallisesta Markov-päätösprosessista siirtymä- ja palautefunktioiden osalta. Siirtymäunktio on nyt $T : S \times A \times S \times \mathbb{N} \to [0, 1]$, missä $T(s, a, s', t) = P(s', t \mid s, a)$ kertoo todennäköisyyden siirtyä tilasta s tilaan s' toiminnolla a ja käyttämällä täsmälleen t aikayksikköä. Siirtymäfunktion perusteella voidaan laskea myös siirtymän kestoajan jakauma $P(t \mid s, a, s')$ jokaiselle siirtymälle (s, a, s').

Usein tehdään lisäoletus, ettei toiminnon suoritusaika riipu tilasta, johon toiminnon jälkeen päädytään, jolloin pätee $T(s, a, s', t) = P(s' \mid s, a)P(t \mid s, a)$. Tämän oletuksen taustalla on ajatus, ettei toiminto ole mikään mielivaltainen kokonaisuus, vaan siinä on jokin yksittäinen tavoite, joka pyritään saavuttamaan. Tehdään siis jatkossa tämä riippumattomuusoletus.

Palautefunktio kertoo SMDP:illä toiminnon suorittamisen aikana kertyvän palautteen odotusarvon. Äärettömän horisontin alennettua palautetta käytettäessä se voidaan määritellä lausekkeella

$$R(s, a) = \sum_{t=1}^{\infty} P(t \mid s, a) \sum_{i=1}^{t} \gamma^{i-1} r_i,$$

missä r_i on i aikayksikön jälkeen saatu välitön palaute.

Q-oppimisen optimaalisen laatufunktion määräävä yhtälö (2) voidaan nyt helposti yleistää SMDP:ille:

$$Q^*(s,a) = R(s,a) + \sum_{t=1}^{\infty} P(t \mid s,a)\gamma^t \sum_{s' \in S} P(s' \mid s,a) \max_{a' \in A} Q^*(s',a')$$

kaikilla $s \in S$, $a \in A$. Yhtälön (3) arvofunktio soveltuu käytettäväksi sellaisenaan, kun taas päivityssääntö (4) muuttuu muotoon

$$Q(s,a) := (1-\alpha)Q(s,a) + \alpha \left(\sum_{i=1}^{t} \gamma^{i-1} r_i + \gamma^t \max_{a' \in A} Q(s',a') \right),$$

missä t on toiminnon suorittamiseen kulunut aika ja r_i aikayksikön i jälkeen saatu välitön palaute.

3.2 Valinnat

Kuten yllä esitetyistä kaavoista voi huomata, semi-Markov-päätösprosessien toiminnot ovat oikeastaan yhden aikayksikön mittaisten alkeistoimintojen sarjoja. Niinpä eräs luonteva jatkokehityskohde onkin liittää tavallisiin Markov-päätösprosesseihin valmiita toimintoketjuja, joilla pyritään tiettyyn tavoitteeseen, kuten siirtymään mielivaltaisesta kohdasta huonetta sen oviaukolle. Intuitiivisesti vaikuttaa selvältä, että tällaiset valmiit toimintoketjut ainakin nopeuttaisivat kohtuullisen hyvän toimintastrategian löytämistä, vaikka optimaalinen strategia ei löytyisikään yhtään nopeammin.

Suttonin, Precupin ja Singhin valinnat (engl. *options*) [5] ovat olennaisesti yllä kuvatun kaltainen MDP:iden laajennus. Niissä tavalliseen Markov-päätösprosessiin liitetään valintojen joukko O ja jokainen valinta $o \in O$ on muotoa (I, π, β). Markov-valinnoissa $I \subseteq S$ on niiden tilojen joukko, joissa valinta voidaan tehdä, π on valinnan suoritusaikana noudatettava toimintastrategia ja $\beta : S \to [0,1]$ on valinnan pysähtymismistodennäköisyyden kertova funktio. Semi-Markov-valinnoissa, joita tämä teksti ei käsittele, pysähtymistodennäköisyys riippuu valinnan koko siihenastisesta suoritushistoriasta.

Normaali tapa käyttää valintoja on tehdä joukko varsinaisia valintoja, joilla pyritään tiettyihin välitavoitteisiiin, ja lisätä sekaan MDP:n toimintoja vastaavat yhden askeleen jälkeen pysähtyvät valinnat. Näin syntyvä valintojen joukko voidaan tulkita SMDP:n toimintojen joukoksi, joten valinnoilla laajennetun Markov-päätösprosessin kanssa voidaan käyttää semi-Markov-päätösprosesseille tarkoitettuja oppimisalgoritmeja. Toteutuksessa tulee tietenkin huomioida, ettei valintaa $o = (I, \pi, \beta)$ kokeilla tilajoukon I ulkopuolella.

3.3 Rinnakkaisuus

Kun Markov-päätösprosesseja yritetään laajentaa tukemaan useita samanaikaisesti suoritettavia toimintoja, törmätään erilaisiin ongelmiin. Toisaalta on epäselvää kuinka siirtymätodennäköisyydet tulisi määritellä, jos eksplisiittistä siirtymämallia tarvitaan, kuten esimerkiksi simulaatioissa. Toisaalta taas toimintojen määrä voi kasvaa räjähdysmäisesti, kun kaikki mahdolliset samanaikaisten toimintojen yhdistelmät otetaan uusiksi toiminnoiksi. Tämä voi muodostua merkittäväksi oppimisen hidasteeksi ja kasvattaa oppimisalgoritmien tilantarvetta huomattavasti. Lisäksi Q-oppimisen yleistyskyvyttömyyden haittapuolet vain korostuvat entisestään toiminto- ja mahdollisesti myös tilajoukon kasvaessa.

Rohanimanesh ja Mahadevan ovat esittäneet erään ratkaisun samanaikaisuusongelmaan [4]. Aluksi tilajoukko määritellään ristitulona tilamuuttujien $W = \{w_1, \ldots, w_k\}$ arvojoukoista. Tämän jälkeen valintojen joukko O ositetaan luokkiin $O_1, \ldots, O_n$ ja tilamuuttujien joukko luokkiin $W_1, \ldots, W_n$ niin, että luokat ovat erillisiä ja luokan O_i toiminnot vaikuttavat vain luokan W_i muuttujien arvoihin. Näin eri luokkiin kuuluvia valintoja voidaan suorittaa rinnakkain ongelmitta. Lisäksi jokaista valintaa $o \in O$ kohti määritellään muuttujajoukko W_o, jonka muuttujat vaikuttavat tämän valinnan kulkuun.

Monivalinta (engl. *multi-option*) on monikko $\overline{o} = (o_1, \ldots, o_n)$, missä $o_i = (I_i, \pi_i, \beta_i) \in O_i$ kaikilla i. Tällä monivalinta voidaan aloittaa joukon $I = \prod_i I_i$ tiloissa ja sen pysähtymistodennäköisyydet kertova funktio on

$$\beta(s) = 1 - \prod_{i=1}^{m} (1 - \beta_i(s)) \quad \text{kaikilla } s \in S.$$

Monivalintojen yhteydessä käytetään siis mallia, jossa yhden valinnan pysähtyessä harkitaan muidenkin suorittamista uudelleen. Monivalinnan toimintastrategia määritellään luonnollisella tavalla sen muodostavien valintojen toimintastrategioiden perusteella.

Laajennetaan jokaista joukkoa O_i valinnalla nop_i, joka on suoritettavissa kaikkialla, pysähtyy yhden askeleen jälkeen, ei anna palautetta eikä vaikuta mitenkään tilaan. Nyt kaikkien monivalintojen joukolla $\overline{O}$ laajennettu alkuperäinen Markov-päätösprosessi on SMDP, jossa voidaan suorittaa samanaikaisesti mitkä tahansa korkeintaan n eri valintaa, jotka kuuluvat osituksessa $O_1, \ldots, O_n$ eri luokkiin. Syntyneen SMDP:n kanssa voidaan tietenkin käyttää tavallisia oppimisalgoritmeja. Rinnakkaisuusongelmaa ei ole, koska samanaikaisesti suoritettavat valinnat eivät voi häiritä toistensa suoritusta.

4 Lopuksi

Markov-päätösprosessit ovat laajalti tutkittu ja sovellettu tapa mallintaa autonomisen toimijan ympäristöä, kun toimintojen seurauksiin liittyy epävarmuutta. Palauteoppimista käyttämällä toimija puolestaan oppii itse toimintastrategioita, jotka johtavat asetettujen tavoitteiden saavuttamiseen. MDP:iden ja palauteoppimisen suosion taustalla lienee niiden yksinkertaisuus ja ymmärrettävyys, jotka tekevät niistä helpot analysoida ja toteuttaa. Lähestymistavan varjopuolia ovat puolestaan yleistyskyvyn puute ja siitä johtuvat suuri tilantarve ja oppimisen hitaus.

Merkittävä osa MDP-tutkimuksesta onkin keskittynyt näiden ongelmien lieventämiseen. Useita toimintoja tiettyyn välitavoitteeseen tähtääväksi alistrategiaksi ketjuttavat valinnat tehostavat oppimista nimenomaan alkuvaiheessa, jolloin tarkoituksena on löytää mahdollisimman nopeasti jokin käyttökelpoinen toimintastrategia. Monivalinnat taas mahdollistavat monessa tapauksessa useiden rinnakkain suoritettavien toimintojen mallintamisen. Rinnakkaisuus taas luonnollisesti tarjoaa monissa tapauksissa merkittävän nopeusedun verrattuna siihen, että vain yhtä asiaa voidaan suorittaa kerralla. Ennakkotiedot ympäristöstä puolestaan tehostavat oppimista huomattavasti verrattuna siihen, ettei toimijalla ole minkäänlaista ennakkokäsitystä.

Markov-päätösprosessien laajennuksiin liittyy edelleen runsaasti tutkittavaa. Oppimisen hitaus ja mallien tilantarve ovat ongelmia, joista tuskin koskaan päästään kokonaan eroon. Välitavoitteiden ja niiden täyttymiseen tähtäävien valintojen tehokas löytäminen olisi merkittävä edistysaskel nykyisiin ennalta annettuihin valintoihin verrattuna. Monivalintamallissa taas erilaisten pysähtymisehtojen vaikutus oppimiseen on vielä tuntematon. Millainen vaikutus olisi sillä, jos monivalinnan yksittäisen valinnan pysähtyminen ei keskeyttäisi muita käynnissä olevia valintoja? Entä kuinka paljon hyvän toimintastrategian löytämistä nopeuttaisi, jos monivalinta keskeytettäisiin heti, jos nykyisessä tilassa olisi tarjolla parempikin vaihtoehto? Avoimia kysymyksiä on monia, ja monet niistä vaikuttavatkin sellaisilta, että niihin saattaisi löytyä ainakin osittainen vastaus.

Lähteet

1 T. Jaakkola, M. I. Jordan, S. P. Singh, Convergence of Stochastic Iterative Dynamic Programming Algorithms. *Advances in Neural Information Processing Systems* **6** (1993).

2 L. P. Kaelbling, M. L.ăLittman, A. W. Moore, Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research* **4** (1996), 237–285.

3 T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

4 K. Rohanimanesh, S. Mahadevan, Decision-Theoretic Planning with Concurrent Temporally Extended Actions. *Proc. 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2001)*, 2001.

5 R. S. Sutton, D. Precup, S. Singh, Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* **112** (1999), 181–211.