Optimalitás őrző általánosítás kockázat érzékeny döntési problémákban

Szakdolgozat

Gábor Zoltán

Témavezető: Dr. Lőrincz András

Belső konzulens: Dr. Dombi József

Tartalomjegyzék

Ábrajegyzék

Táblázatok

1. Fejezet

Bevezetés

Az autonóm rendszerek építésének vágya és problematikája végigkíséri az emberiséget történelme folyamán (Gólem, Frankenstein, Kempelen sakkautomatája, Asimov robotikai törvényei, stb.). Tudomásunk szerint konkrét megvalósítással csak az utóbbi időkben próbálkoztak, de számos irodalmi műben (ld. fent) találkozhatunk olyan robotokkal, amiket az ember kiszolgálására készítettek és az emberi tevékenységek legszélesebb skáláját képesek végezni minden beavatkozás nélkül. Ezek a robotok saját magukról is gondoskodnak; hibásodás esetén megjavítják magukat, alkatrészt cserélnek, olajoznak stb.

A ma robotjai egyenlőre még csak meghatározott "védő", vagyis barátságos környezetben képesek mozogni. Ennek oka, hogy a jelen programozástechnikai eszközökkel túl bonyolult feladat egy robot felkészítése a "valós" környezet összes lehetséges helyzetére. Még védő környezetben is megeshet, hogy egy robot váratlan eseményekkel találkozik. Ilyenkor – ha az üzemeltetőnek szerencséje van – a robot leáll. Elképzelhető azonban olyan helyzet is, amelyben egy előre nem feltérképezett esemény katasztrófát okoz.

A környezet sokszínűsége által támasztott akadály legyőzésének egyik lehetséges módszere, ha alkalmazkodó – tanuló robotot építünk. Végső soron az emberiség – és egyáltalán az élővilág is – a tanulási (alkalmazkodási) képességének köszönheti sikerét. Persze joggal vetődik fel a robotok kontrollálhatóságának kérdése is. Számos olyan mű fordul elő a szép- és sci-fi irodalomban, amelyben a túl nagy önállóságra szert tett rendszerek egyszerre készítőik ellen fordulnak. Az ilyen esetek elkerü-

lése kétféleképp képzelhető el: Asimov műveiben olyan robotokat képzel el, melyek velejében beépített fékek, sérthetetlen törvények vannak. Több regényében érzékletesen mutatta be ennek a megközelítésnek a törvények pontatlanságából fakadó hátrányait. Újabban – talán épp Asimov hatására – az autonómiát kevésbé korlátozó módszereket vetettek fel, mint pl. a robotok tevékenységének meghatározása célok és motivációk beépítésével. Az Ember védelme is megfogalmazhatónak tűnik ily módon. Ez a módszer kevésbé korlátozza a robotokat döntési autonómiájukban és ezzel elkerülheti a merev törvények beépítéséből származó problémákat. El kell azonban azt is mondani, hogy ennél a módszernél a garancia a robotok "szelíd" voltára sokkal árnyaltabb – mély matematikai állításokban rejlik. A dolgozatban ezen matematikai állításokat vizsgáljuk – különös tekintettel a robotok döntési algoritmusának hatásossá tételére. Speciálisan a robotok fogalomalkotó képességével való felruházásával foglalkozunk.

A dolgozatban – matematikai oldalról – optimális döntési stratégiákat vizsgálok az ún. kockázat érzékeny Markov döntési folyamatok keretén belül. A dolgozat központi kérdése, hogy hogyan lehet optimális döntési stratégiákat kialakítani ha a döntési modellt csak részlegesen ismerjük. A dolgozatban az ún. költségfüggvény alapú algoritmusokkal foglalkozom. Az ilyen algoritmusok (döntési függvények) gyakorlati alkalmazhatóságát korlátozza, hogy a költségfüggvény pontos reprezentációját követelik meg - a tárigény az állapotok számával lineárisan nő. A dolgozatban bevezetek egy olyan csökkentett tárigényű reprezentációt, melyből az optimális politika még rekonstruálható. Megadok egy algoritmust és egy szükséges feltételt, mely mellett az algoritmus optimalitás őrző kompakt reprezentációt készít (5.2.8 Tétel). Ennek bizonyítása három lépésben zajlik – a dolgozat is e szerint tagolódik. A 2. fejezetben az autonóm adaptív rendszerek elveinek általános ismertetése található, ezután térek át a bizonyítás már említett három lépésére. A 3. fejezetben kidolgozom a kockázat érzékeny Markov döntési folyamatok stacionér politikákra vonatkozó elméletét. Az elmélet leglényegesebb következtetése, hogy az ún. optimális költségfüggvény egyértelműen megadja a döntési probléma optimális megoldásait és az optimális költségfüggvény megkapható a döntési modellhez tartozó mohó operátor iterációjával (3.5.1 Következmény). A 4. fejezetben vizsgálom meg, hogy az elmélet

miként terjeszthető ki arra az esetre, ha a döntési modell nem ismert. Előkészítésként bebizonyítom, hogy az optimális költségfüggvényhez való konvergenciát nem befolyásolja, ha az iteráció a különböző komponensekre nem feltétlenül egyidejű – azaz az iteráció aszinkron (4.1.3 Tétel). Ezután bebizonyítom, hogy ha csak azokat az állapotokat frissítjük, amelyeket a rendszer végtelen sokszor megtapasztal és a rendszer mindig a pillanatnyi tudása szerinti legjobb akciót választja, akkor az így nyert algoritmus asszimptotikusan optimális lesz (4.2.4 Tétel). Ezután megmutatom, hogy ez az eredmény kiterjeszthető arra az esetre, ha a döntési modell nem ismert, hanem a tapasztalatok alapján építjük (4.3.1 Tétel). Ehhez feltételeznünk kell, hogy a nyert tapasztalatok elegendőek egy pontos modell felépítésére. Ezután ismertetem a DC modellt, mely egy hely- és időtakarékos implementációja a döntési algoritmusnak. Az 5. fejezetben megadom az algoritmust, mely a költségfüggény egy kompakt reprezentációját készíti el és bebizonyítom, hogy ha a tanuláshoz felhasznált tapasztalatok jól reprezentálják a döntési problémát és az ún. hamis általánosítások optimista jellegűek, akkor az algoritmus által készített költségfüggvény asszimptotikusan az optimális költségfüggvénnyel fog megegyezni (5.2.8 Tétel). Szintén kitérek az algoritmus gyors implementációjának lehetőségére – mely a DC modell kibővítését jelenti. Az utolsó fejezet (6 fejezet) a kísérletek bemutatására szolgál. A kísérletekhez létrehoztam egy általános célú szimulációs szoftvert, mely két részből áll: a környezet (döntési modell) szimulációjából, valamint a döntéshozó és tanuló algoritmus implementációjából. A kísérletekkel bemutatom, hogy a választott döntési probléma és általánosítás operátor esetén a kompakt reprezentációt készítő algoritmus mind a teljesítmény, mind a tárigény, mind a valós idejűség terén jobbnak bizonyul, mint az egyszerű költségfüggvény táblát készítő algoritmus. A kísérletekhez szükséges szoftvert egy IBM PC–486–os gépen fejlesztettem Linux© operációs rendszerben, ANSI C++ 3.0 nyelven és GNU© C++ 6.3 fordítóval. A szükséges adatstruktúrák megvalósítására a LEDA© fejlesztői csomagot használtam.

2. Fejezet

Autonóm adaptív rendszerek

Ebben a fejezetben betekinthetünk az autonóm rendszerek általános elméletébe. A bemutatandó modell egyfajta keretet ad az induláshoz. Az elmélet kidolgozásakor főleg a mesterséges rendszerek vizsgálatára hagyatkozunk, de figyelembe vesszük a biológia, pszihológia, és etológia idevágó eredményeit is.

2.1 Kontroll stratégia, környezet, dinamika

Tekintsünk egy, a környezetével állandó interakcióban álló rendszert. Az interakció során a rendszer a környezete állapotát *érzékelőin* keresztül kérdezheti le és a környezetét *aktuátorai* segítségével befolyásolhatja. Legyen X a rendszer összes lehetséges szenzoros állapotainak halmaza, és A a rendszer aktuátorai lehetséges állapotainak halmaza. Tegyük fel egyenlőre, hogy a rendszer egy rögzítet stacionér stratégiát követ. stacionér stratégia alatt egy olyan $\mu : X \to A$ leképezést értünk, mely azt adja meg, hogy a rendszer hogyan változik: $x \in X$ állapot bekövetkezése a rendszerben az $a = \mu(x)$ akciót váltja ki. Az aktuátorok beállítása hatással van a rendszer környezetére és a környezeten keresztül befolyásolja a rendszer jövőbeni állapotait. Tehát a μ kontroll stratégia lezárható egy $\hat{\mu} : X \to P(X)$[1] leképezéssé. Ahhoz, hogy ezt a lezárást definiálni tudjuk jelöljük a környezet összes lehetséges állapotának halmazát E–vel, a környezet megváltozását egy adott a aktuátor hatására leíró függvényt

[1] $P(X)$ az X halmaz hatványhalmaza, vagyis X részhalmazainak halmaza.

pedig $\nu : E \times A \to E$–vel. Jelöljük továbbá a környezet érzékelését leíró függvényt $\sigma : E \to X$–vel. A μ kontroll stratégia lezártja alatt az (E, ν) környezetben a

$$\hat{\mu}(x) = \chi(x, \mu(x))$$

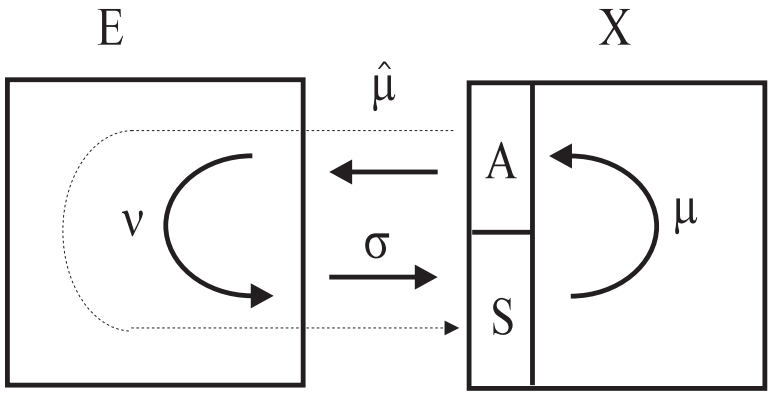
leképezést értjük, ahol $\chi : X \times A \to P(X)$ leképezés adott a akció és x rendszerállapotra megadja az akció lehetséges kimeneteleit:

$$\chi(x, a) = \{\sigma(\nu(e, a)) | e \in E, x = \sigma(e)\}$$

látható, hogy a $\hat{\mu}(x)$ halmaz mérete az

$$S(x) = \{e \in E : x = \sigma(e)\}$$

halmaz méreténél csak kisebb lehet, mivel ν leképezés. A rendszer és környezete interakcióját a 2.1 ábra ábrázolja. Ha az $S(x)$ halmaznak minden x-re egy eleme van, azaz az érzékelés pontos, akkor tetszőleges μ stratégia $\hat{\mu}(x)$ lezártja függvény: vagyis minden akció hatása kiszámítható. Minél nagyobb az $S(x)$ halmaz mérete, annál bizonytalanabb egy–egy akció kimenetele. Az $\{S(x)\}_{x \in X}$ halmazrendszer a rendszernek azt a képességét jellemzi, hogy milyen pontossággal tudja megállapítani szenzorai segítségével környezete állapotát.



2.1. ábra: Autonóm rendszer és környezete interakciója

Az előzőek értelmében a μ, stacionér kontroll stratégiát alkalmazó rendszer *dinamikáját* a következő differenciális tartalmazással írhatjuk le.

$$x^+ \in \hat{\mu}(x),$$

ahol x^+ a rendszer "következő" pillanatbeli állapotát jelöli. A rendszer környezetével való interakcióját szintén differenciális tartalmazással írhatjuk le:

$$x^+ \in \chi(x, a),$$

ahol a az x állapotú rendszer valamelyik akciója. Egy, a környezetével állandó interakcióban álló rendszer tehát működésre zártnak tekinthető, hisz a rendszer bármely cselekedetének eredménye leírható magának a rendszer állapotának a megváltozásával. Ez a zártság nem a fizikai értelemben vett zártságot jelenti: fizikai értelemben a rendszer igenis nyitott, hiszen állandóan befolyásolja környezetét és az is befolyással bír az ő állapotára.

2.2 Metadinamika és belső reprezentáció

Az *autonóm* rendszerek definiáló tulajdonsága az, hogy működésük a rendszer állapotának egy bizonyos tartományon belül való tartására irányul. Ezt a *túlélésre való törekvésnek* szokás nevezni és a következőképp fogalmazható meg: Legyen K a rendszer X állapotainak egy részhalmaza: a rendszer *túlélési tartománya*. A cél az, hogy a rendszert ezen túlélési tartományon belül tartsuk, vagyis ahhoz, hogy a rendszer fennmaradjon olyan μ kontroll stratégiát kell alkalmaznia amelyre igaz, hogy $\hat{\mu}(x) \subset K$, minden egyes x állapotra. Hogyan talál ilyen kontroll stratégiát a rendszer? A rendszer "élete" során tapasztalatokat szerez a környezetéről. A működése annál bizonytalanabb, minél nagyobb a bizonytalanság környezete állapotának megítélésében, azaz minél nagyobb az $S(x)$. Ezt a bizonytalanságot kiküszöbölendő, a rendszernek olyan hasznos hipotéziseket kell gyártania a környezetéről, amelyek segítségével képes megmagyarázni a "történések" okait. A jó hipotézisek felállításának képességét nevezzük *abdukciós* képességnek. Az abdukció bizonyos értelemben az indukció vagy modus ponens (MP) fordítottja, amikor egy q állítás és egy $p \rightarrow q$ szabály fennállása esetén, a p állítást, mint a q állítás okát számításba vesszük. Az autonóm rendszerek a környezetük állapottáról és szabályairól hipotéziseket gyártanak, hogy élettartalmukat kitolják. A jó hipotézisek segítségével a kontroll stratégia javítható. A kontroll stratégia módosítását végző szabályosságok összességét *meta-*

dinamikának nevezzük. A rendszer metadinamikája egy:

$$M : X \times A^X \to A^X$$

függvénnyel írható le. Ha a rendszer állapota $x \in X$ és pillanatnyi kontroll stratégiája μ, akkor az M metadinamika segítségével a rendszer a következő időpillanatban már a

$$\mu^+ = M(x, \mu)$$

kontroll stratégiát fogja használni. Az autonóm rendszer matadinamikáját általában adataik és adatstruktúráik megváltozásával írjuk le. A rendszer a környezetére vonatkozó tapasztalatait tudásbázisba gyűjti, melyet a környezet vagy "külvilág" *belső reprezentációjának* nevezzünk. A "belső" szó utal arra, hogy itt csak a rendszer belső állapotairól beszélhetünk, mivel a rendszer a környezetéből csak ezt "látja". Jelöljük B–vel a rendszer belső reprezentációjának lehetséges állapotait. Ekkor a rendszer kontroll stratégiája mint egy $\mu : X \times B \to A$ leképezés írható le:

$$\mu = \mu(b, x),$$

ahol $b \in B$ egy belső reprezentáció. A rendszer M metadinamikája a rendszer pillanatnyi b belső reprezentációját változtatja meg:

$$b^+ = M(x, b),$$

ahol $M : X \times B \to B$ leképezés. A rendszer működését tehát a következőképp lehetne leírni:

$$(a, b^+) \in F(x, b),$$

ahol $F = \mu \times M$, azaz $F(x, b) = (\mu(x, b), M(x, b))$. A belső reprezentációval rendelkező rendszereknél három részegységet különböztethetünk meg: érzékelő rész, végrehajtó rész és belső reprezentáció.

- Az érzékelő rész a környezet (külvilág) és a rendszer (belvilág) állapotát érzékeli, és továbbítja inputként a rendszernek. Azzal, hogy megengedjük, hogy a rendszer a saját állapotát is érzékelje, jelentősen növelhetjük tanulási lehetőségeit. Az egyes érzékelő egységeket, függetlenül attól, hogy belső, vagy külső

információkat közvetítenek, szenzoroknak nevezzük. A szenzoros információ lehet előfeldolgozott információ is, de fontos, hogy a rendszer többi részétől függetlenül működjön.

- A végrehajtó rész a rendszer cselekvéseinek megvalósító egysége. Általában öszetett, több manipulátorból állhat. A végrehajtó rész vezérlése az *operátorok* segítségével történik. Az operátorok beállításával lehet a manipulátorokat, és azok segítségével pedig a környezetet állapotát változtatni.

- A belső reprezentációban tárolja el a rendszer a tapasztalatait. Ez az a rész, amelyben az egyes rendszerek legtöbbször különböznek. A belső reprezentáció tárolhatja a rendszer hosszútávú terveit, ötleteit, céljait, stb.

2.3 Célorientált autonóm rendszerek

Általános célú autonóm rendszer modelljének tervezésénél nem szoríthatjuk meg a modellt egy rögzített metadinamika megadására. Ehelyett a metadinamika általános alapelveit kell rögzíteni. Ha a rendszert célokkal látjuk el, akkor így lehetőségünk nyílik a metadinamika általános alapelvei kidolgozására. Egy ilyen metadinamika esetén egy konkrét rendszer elkészítéséhez csak annak céljait kell formába öntenünk. A metadinamika fog gondoskodni a célok elérését biztosító algoritmusok kialakulásáról. Sokszor egy rendszernek több célja is lehet, melyek között ellentételesek is előfordulhatnak. A cél-konfliktusok feloldását proritások bevezetésével, vagy fuzzy értékeléselméleti módszerekkel oldhatjuk meg. A célok megadásának egy igen egyszerű módszere a következő: jelöljük a rendszer lehetséges céljainak halmazát C–vel, s vegyünk egyetlen $c \in C$ célt. Minden célt úgy határozunk meg, hogy "megmondjuk", hogy melyek azok az állapotok, melyekben a cél teljesül. A c szimbólum tehát azokat az állapotokat jelöli, amelyekben cél teljesül: azaz $c \subset X$. Egy adott cél "beprogramozása" egyszerűbb, mint a cél elésére irányuló cselekvések beprogramozása lenne. Például egyetlen szenzor figyelése is elég lehet: ha van "éhségérzet" szenzor a rendszerben (mely akkor jelez, ha az éhségérzet fennáll), akkor elegendő az éhségérzet szenzor figyelésének örökítése az "éhség megszüntetése" cél leírásához,

a többi szenzor jelzéseit egyszerűen figyelmen kívül kell hagyni.

3. Fejezet

Kockázat érzékeny MDP-k

A továbbiakban speciális, ún. Markov dinamikájú környezeteket tekintünk csak és a metadinamikát is fokozatosan megszorítjuk.

3.1 Markov döntési folyamatok

A diszkrét Markov-folyamatok [2] a sztochasztikus folyamatok egy részét képezik [10] és bizonyos megszorításokkal ugyan, de jól modellezik az előzőekben bevezetett adaptív rendszereket. Legyenek $X_0, X_1, X_2, \ldots$ diszkrét valószínűségi változók, ahol mindegyik X_t-edik változó egy X megszámlálható halmazból veszi fel az értékét. Az X halmazt az *állapotok halmazának* nevezzük. Az X_t valójában a rendszer t időbeli állapotát jelöli. Ezentúl jelöljük X elemeit $x, y, z, x_0, x_1, \ldots$ – vel.

3.1.1 DEFINÍCIÓ *Az $X_0, X_1, X_2, \ldots$ diszkrét valószínűségi változók* Markov-láncot (diszkrét Markov-folyamatot) *alkotnak, ha a következő teljesül tetszőleges $x_0, x_1, \ldots x_n$ X-beli állapotokra, és tetszőleges $t_0 < \ldots < t_n$ időpillanatokra:*

$$P(X_{t_n} = x_n | X_{t_{n-1}} = x_{n-1}, \ldots, X_{t_1} = x_1, X_{t_0} = x_0) = P(X_{t_n} = x_n | X_{t_{n-1}} = x_{n-1}).$$

Ez azt jelenti, hogy a rendszer korábbi állapotai a későbbi állapotokra csak a jelen állapoton keresztül gyakorolnak befolyást. Legyenek adottak az X állapotok halmaza, A akciók véges halmaza, és $P_{xy}(a)$ átmenetvalószínűségek, ahol $x, y \in X$ és $a \in A$. Az átmenetvalószínűségeket úgy kell értelmezni, hogy ha az x állapotban vagyunk és végrehajtjuk az a akciót, akkor $P_{xy}(a)$ valószínűséggel jutunk az y

állapotba. Rendeljünk most minden (x, a, y) állapot-akció-állapot hármashoz egy költséget, amelyet $c(x, a, y)$-val jelölünk. A $c(x, a, y)$ költségről feltesszük, hogy y-ban egyenletesen korlátos, vagyis létezik olyan M, hogy $c(x, a, y) < M$, áll minden x-re és a-ra. Ha a rendszer az x állapotban van és az a akciót választja, a következő dolgok történnek:

- az y állapot a $P_{xy}(a)$ átmenetvalószínűségeknek megfelelően választódik,
- fellép egy $c(x, a, y)$ költség.

A fent leírt folyamatot *Markov döntési folyamatnak* nevezzük. Az akciók kiválasztásában a rendszer vagy *döntéshozó* valamilyen *politikát* követ. Ez valójában egy függvény, amelynek értékkészlete az akciók halmaza. A politika által választott akció függhet például a folyamat előző állapotaitól, vagy lehet sztochasztikus abban az értelemben, hogy a soron következő akció csak valamilyen eloszlás értelméig meghatározott. Az összes politikák osztályának egy fontos részhalmazát képezik a *determinisztikus stacionér politikák*. Ezekre az jellemző, hogy csak a rendszer aktuális állapotától függnek:

3.1.2 DEFINÍCIÓ π stacionér politika, *ha* $\pi \in \{u : X \to A\}$

Ha π egy stacionér politika, akkor könnyen látható, hogy a rendszer állapotai, azaz az $\{X_t, t = 0, 1, 2, \ldots\}$ valószínűségi változók sorozata Markov-láncot alkot, éspedig

$$P(X_{t+1} = y | X_t = x) = P_{xy}(\pi(x))$$

minden t-re. *Markov döntési problémát* egy Markov döntési folyamaton definiálhatunk: a probléma egy adott költségfüggvény és optimalizálási kritérium szerinti optimális politika keresését jelenti.

3.2 Optimális politikák

Jelölje c_t a t-edik időpillanatban fellépő költséget. Tegyük fel, hogy a döntéshozó rögzített π politikát hajt végre és legyen $S^\pi(x_0) = \sum_{t=0}^{\infty} \gamma^t c_t$, ahol $0 < \gamma < 1$. $S^\pi(x_0)$ egy korlátos valószínűségi változó, mely a politikához tartozó lecsengetett

összköltséget adja meg. A γ, az ún. lecsengetési állandó biztosítja a korlátosságot, és azt, hogy a jövőbeni költségek kevésbé számítsanak az összköltségben. Mivel $S^\pi(x_0)$ valószínűségi változó, így nem hasonlíthatók össze általa a különböző politikák. Az S^π véletlen jellegét eltüntethetjük, ha vesszük a várható értékét:

$$v_\pi(x_0) = E_\pi[S_\pi(x_0)].$$

Itt E_π a π stacionér politika által generált, az állapot-akció trajektóriákon értelmezett eloszlás szerint várható értéket jelöli [10]. A jelen dolgozatban azonban az ún. *Maximális Lecsengetett Összköltség (Maximal Total Discounted Cost)* függvényt használjuk [5]:

$$v_\pi(x_0) = \operatorname*{esssup}_\pi S_\pi(x),$$

ahol $\operatorname{esssup}_\pi$ a π által az állapot-akció trajektóriákon generált mérték tartóján vett szuprémumot jelöli. Az ilyen költségfüggvénnyel ellátott Markov döntési folyamatokat hívjuk *kockázat érzékeny Markov döntési folyamatoknak* [5], a $v_\pi(x)$ függvényt pedig a π *politika értékelésének* nevezzük. Ezek után már definiálhatjuk, hogy mit értünk optimális politika alatt.

3.2.1 DEFINÍCIÓ *Azt mondjuk, hogy a π^* politika* optimális*, ha:*

$$v_{\pi^*}(x) = v^*(x), \qquad x \in X,$$

ahol

$$v^*(x) = \inf_\pi v_\pi(x), \qquad x \in X$$

jelöli az ún. optimális költségfüggvényt.

3.3 Stacionér politikák iteratív értékelése

A következő részekben megmutatjuk, hogy a kockázat érzékeny szekvenciális döntési problémákban korlátos költségek mellett mindig létezik optimális politika. Legyen adott (X, A, T, c) négyes, ahol X az állapotok véges halmaza, A az akciók véges halmaza, $T : X \times A \to P(X)$ *átmenet függvény*, ahol $P(X)$ az X halmaz

hatványhalmaza[1], és $c(x,a,y) \in \mathbf{R}$ átmenetköltségek ($x,y \in X$, $a \in A$). Ebben a fejezetben csak stacionér politikákkal foglalkozunk. Legyen $B(X)$ az X-en értelmezett valós, korlátos függvények halmaza. Legyen π egy stacionér politika és vezessük be a következő, $B(X)$-en értelmezett operátort:

$$(T_\pi v)(x) = \max_{y \in T(x,\pi(x))} (c(x,\pi(x),y) + \gamma v(y)). \tag{3.1}$$

Jelöljük tetszőleges $u \in B(X)$ szuprémum normáját $||u||$-val:

$$\|u\| = \sup_{x \in X} |u(x)|. \tag{3.2}$$

3.3.1 LEMMA *A T_π operátor kontrakció, γ index-szel, vagyis bármely két $u,v \in B(X)$ függvényre áll, hogy $\|T_\pi u - T_\pi v\| \leq \gamma \|u - v\|$.*

Bizonyítás. Legyen $u,v \in B(X)$ rögzített. Ahhoz, hogy $||T_\pi u - T_\pi v||$-t becsüljük, becsüljük meg először adott $x \in X$-re a $(T_\pi u)(x) - (T_\pi v)(x)$ különbséget:

$$\begin{aligned}((T_\pi u)(x) - (T_\pi v)(x)) &= \max_{y \in T(x,\pi(x))} (c(x,\pi(x),y) + \gamma u(y)) \\ &\quad - \max_{y \in T(x,\pi(x))} (c(x,\pi(x),y) + \gamma v(y)) \\ &= c(x,\pi(x),\bar{y}) + \gamma u(\bar{y}) \\ &\quad - \max_{y \in T(x,\pi(x))} (c(x,\pi(x),y) + \gamma v(y)),\end{aligned}$$

ahol $\bar{y}$ olyan, hogy:

$$c(x,\pi(x),\bar{y}) + \gamma u(\bar{y}) = \max_{y \in T(x,\pi(x))} (c(x,\pi(x),y) + \gamma u(y)).$$

Egyszerűen adódik, hogy:

$$\begin{aligned}((T_\pi u)(x) - (T_\pi v)(x)) &\leq \gamma(u(\bar{y}) - v(\bar{y})) \\ &\leq \gamma \sup_{y \in X} (u(y) - v(y)) \\ &\leq \gamma \|u - v\|\end{aligned}$$

Mivel a fenti egyenlőtlenség minden $x \in X$–re áll, ezért

$$\sup_{x \in X} ((T_\pi u)(x) - (T_\pi v)(x)) \leq \gamma \|u - v\|.$$

[1] $T(x,a)$ tulajdonképpen az előző fejezetbeli $P_{x_0}(a)$ átmenet eloszlás tartóját jelöli: $T(x,a) = \{y \in X | P_{xy}(a) > 0\}$.

Megcserélve u és v szerepét láthatjuk, hogy

$$\sup_{x\in X}((T_\pi v)(x)-(T_\pi u)(x))\le\gamma\|u-v\|$$

is áll. A fenti két egyenlőtlenség felhasználásával kapjuk, hogy

$$\sup_{x\in X}|(T_\pi u)(x)-(T_\pi v)(x)|\le\gamma\|u-v\|,$$

vagyis

$$\|T_\pi u-T_\pi v\|\le\gamma\|u-v\|,$$

ami bizonyítandó volt. □

3.3.2 Állítás *Ha π stacionér politika, akkor $v_\pi(x)=\lim_{n\to\infty}(T_\pi^n v)(x)$*[2].

Bizonyítás. Mivel T_π kontrakció, így a kontrakciós tétel miatt tudjuk, hogy $T_\pi^n v$ a T_π operátor egyetlen fixpontjához konvergál. Ezért elég megmutatni, hogy tetszőleges stacionér politika értékelő függvénye kielégíti a

$$v_\pi(x)=\max_{y\in T(x,\pi(x))}(c(x,\pi(x),y)+\gamma v_\pi(y))$$

fixpont egyenletet. Jelöljük $c_\pi(x,y)$-val $c(x,\pi(x),y)$-t és a $\sum_{t=0}^\infty\gamma^t c_\pi(x_t,x_{t+1})$ összeget $c_\pi(x_0,x_1,x_2,\ldots)$-val. Ekkor $v_\pi(x_0)$ definíció szerint épp

$$\sup_{x_1\in T_\pi(x_0),x_2\in T_\pi(x_1),\ldots}c_\pi(x_0,x_1,x_2,\ldots),$$

ahol $T_\pi(x)=T(x,\pi(x))$. Rögzítsünk egy $\epsilon>0$ számot. Ehhez van olyan $x_1^*,x_2^*,\ldots,x_t^*,\ldots$ lehetséges π-trajektória ($x_1^*\in T_\pi(x_0)$ és $x_{t+1}^*\in T_\pi(x_t^*)$), hogy

$$v_\pi(x_0)-\epsilon<c(x_0,x_1^*,x_2^*,\ldots,x_t^*,\ldots).$$

Nyilván

$$\begin{aligned}
c(x_0,x_1^*,x_2^*,\ldots,x_t^*,\ldots) &\le \sup_{x_2\in T_\pi(x_1^*),x_3\in T_\pi(x_2),\ldots}c(x_0,x_1^*,x_2,\ldots,x_t,\ldots)\\
&\le \max_{x_1\in T_\pi(x_0)}(\sup_{x_2\in T_\pi(x_1),x_3\in T_\pi(x_2),\ldots}c(x_0,x_1,x_2,\ldots,x_t,\ldots))\\
&= \max_{x_1\in T_\pi(x_0)}(\sup_{x_2\in T_\pi(x_1),x_3\in T_\pi(x_2),\ldots}c_\pi(x_0,x_1)+\gamma\sum_{t=0}^\infty\gamma^t c_\pi(x_{t+1},x_{t+2}))\\
&= \max_{x_1\in T_\pi(x_0)}(c_\pi(x_0,x_1)+\gamma\sup_{x_2\in T_\pi(x_1),x_3\in T_\pi(x_2),\ldots}\sum_{t=0}^\infty\gamma^t c_\pi(x_{t+1},x_{t+2})).
\end{aligned}$$

[2] $T^n v=TT\ldots Tv$, ahol T n-szer szerepel.

Vegyük észre, hogy

$$\sup_{x_2\in T_\pi(x_1),x_3\in T_\pi(x_2),\dots} \sum_{t=0}^{\infty} \gamma^t c_\pi(x_{t+1},x_{t+2})$$

éppen $v_\pi(x_1)$, így ezt visszahelyettesítve kapjuk, hogy

$$v_\pi(x_0) - \epsilon < \max_{x_1\in T_\pi(x_0)} (c_\pi(x_0,x_1) + \gamma v_\pi(x_1)) = (T_\pi v_\pi)(x_0).$$

Mivel ϵ tetszőleges volt így kapjuk, hogy $v_\pi \le T_\pi v_\pi$. Másrészt

$$\sup_{x_1\in T_\pi(x_0),x_2\in T_\pi(x_1),\dots} c_\pi(x_0,x_1,x_2,\dots) \ge \max_{x_1\in T_\pi(x_0)} (\sup_{x_2\in T_\pi(x_1),\dots} c_\pi(x_0,x_1,x_2,\dots)) = (T_\pi v_\pi)(x_0),$$

és így valóban $T_\pi v_\pi = v_\pi$. □

3.4 Mohó és optimális politikák

Ebben a fejezetben megmutatjuk, hogy a fenti döntési folyamatban mindig léteznek optimális politikák.

3.4.1 DEFINÍCIÓ *Azt mondjuk, hogy a* π politika mohó a v költségfüggvényre nézve, *ha*

$$\pi(x) \in \operatorname{Argmin}_{a\in A} \max_{y\in T(x,a)} (c(x,a,y) + \gamma v(y)).^3$$

3.4.2 ÁLLÍTÁS *Ha* π *mohó* v^**-ra, akkor* π *optimális is.*

Bizonyítás. Emlékeztetőül a v^* optimális költségfüggvény definíciója

$$v^*(x) = \inf_\pi v_\pi(x), \qquad x\in X.$$

Az egyszerűség kedvéért jelöljük G-vel azt a $B(X) \to B(X)$ operátort, amely a $v\in B(X)$ függvényt a

$$(Gv)(x) = \min_{a\in A} \max_{y\in T(x,a)} (c(x,a,y) + \gamma v(y)) \tag{3.3}$$

függvénybe képezi. Legyen π mohó v^*-ra nézve. Nyilvánvaló, hogy $T_\pi v^* = Gv^*$ (T_π, G és a mohóság definíciójából). Belátjuk, hogy $Gv^* \le v^*$, és ebből már következik

[3] $\operatorname{Argmin}_{a\in A} v(a)$ definíció szerint azon $a\in A$ elemek halmaza, melyen v felveszi a minimumát.

is, hogy π optimális. Ugyanis ebben az esetben $T_\pi v^* \leq v^*$, melyből mindkét oldalra alkalmazva T_π-t kapjuk, hogy $T_\pi^2 v^* \leq T_\pi v^*$ és ez kisebb vagy egyenlő mint v^*. Ezt az iterációs eljárást folytatva kapjuk, hogy minden n természetes számra $T_\pi^n v^* \leq v^*$. Ha n tart a végtelenbe, akkor a bal oldal v_π-hez konvergál, tehát $v_\pi \leq v^*$, tehát v_π optimális. Meg kell még mutatni, hogy $Gv^* \leq v^*$. Legyen ν egy tetszőleges politika. Ekkor $Gv^* \leq T_\pi v^*$ (ez nem csak a v^* pontban áll) és $T_\nu v^* \leq T_\nu v_\nu = v_\nu$ (ez utóbbi egyenlőséget bizonyítottuk be a (3.3.2) állításban). Összeolvasva az egyenlőtlenségeket kapjuk, hogy $Gv^* \leq v_\nu$ minden ν-re. Véve ν-re az infimumot, kapjuk, hogy $Gv^* \leq v^*$. □

3.5 A Bellman egyenlet

3.5.1 KÖVETKEZMÉNY *Létezik optimális politika, és az optimális költségfüggvény teljesíti a következő* Bellman típusú egyenletet *[2]:*

$$v^*(x) = \min_{a \in A} \max_{y \in T(x,a)} \left(c(x,a,y) + \gamma v^*(y) \right).$$

Továbbá az optimális költségfüggvény előáll, mint $\lim_{n\to\infty} G^n v$*, ahol* v *egy tetszőleges korlátos függvény.*

Bizonyítás. Mivel A véges, így mindig van olyan politika, amely mohó az optimális költségfüggvényre nézve. A fenti állítás szerint egy ilyen politika optimális is lesz. Legyen most π mohó v^*-ra. A fenti bizonyításból látszik, hogy ekkor $T_\pi v^* = v^*$, másrészt mivel π mohó v^*-ra így $T_\pi v^* = Gv^*$ definíció szerint, tehát $Gv^* = v^*$. Könnyű látni, hogy G is kontrakció, így az iterálása az egyetlen fixpontjához vezet, ami mint már tudjuk épp v^*. □

A $v_{t+1} = Gv_t$ iterációt nevezzük *Dinamikus Programozás* (DP) iterációnak. A fenti tételek alapján látható, hogy egy optimális politika megkereséséhez elegendő v^*-ot kiszámolni, pl. a DP iterációval, majd venni v^*-ra a mohó politikát. A továbbiakban ezt az ötletet követjük, olyan feltételek mellett, hogy a döntési modell nem vagy csak részben ismert, vagy a "teljes" DP iteráció "túl" drága.

4. Fejezet

Megerősítéses tanulás

Az alábbiakban azt a feladatot tanulmányozzuk, amikor nem ismert a döntési folyamat modellje. A következőekben csak az ún. költséfüggvény alapú megerősítéses tanulásos algoritmusokkal foglalkozunk. Ezek az algoritmusok a pontos vagy becsült mohó operátor alapján a dinamikus programozás egy aszinkron formáját valósítják meg. Az algoritmus egy végtelen iterációból áll, melynek minden lé-

4.1. táblázat: Költségfüggvény alapú Reinforcement tanuló algoritmus

`repeat forever`{
1. x_t, c_t érzékelése.
2. Modell módosítása: $G_{t+1} = M(G_t, c_t, x_t, a_{t-1}, x_{t-1}, \ldots)$.
3. Értékelések újraszámolása:

$$v_{t+1}(x) = \begin{cases} v_t(x), & x \notin U_t; \\ (G_t v_t)(x), & x \in U_t. \end{cases}$$

4. Akció kiválasztás: $a_t = \mu(x_t, v_t)$.
5. Akció végrehajtása.
6. $t := t + 1$.

}

pésében a következők történnek: az első lépésben a pillanatnyi állapot, valamint az átmenetköltségek érzékelése történik, ami azt jelenti, hogy a rendszer beállítja belső állapotát a szenzorai segítségével. A második lépésben történik a modell módosítása, melynek végeredménye a (3.3) egyenletben definiált G mohó operátor egy új közelítése. A becsült mohó operátor alapján történik a becsült optimális költségfüggvény "finomítása". Valójában tudnunk kell, hogy a belső reprezentációval

rendelkező rendszerekben a modell módosítása a belső reprezentáció megváltoztatását jelenti. Az értékelések újraszámolása az immáron megváltozott modell alapján történik. Itt az U_t halmaz tetszőleges nemüres halmaz. Általában elvárjuk, hogy adott t-re $x_t \in U_t$ teljesüljön. Megjegyezzük, hogy egy menetben (a 3-as lépésben) egy állapot értékelését, néha érdemes többször is újraszámolni, másrészt az állapotok frissítésének sorrendje igen fontos lehet [12,15]. Azzal, hogy nem minden állapot értékelését számoljuk újra egy-egy lépésben jelentős futásidőt takaríthatunk meg. Az akciók kiválasztása a pillanatnyi értékelés alapján történik a rendszer kontroll stratégiája segítségével. A kontroll stratégia lehet például olyan, hogy ha egy bizonyos állapotról nincs elég információnk (bizonytalan az értékelésünk), akkor az algoritmus véletlenszerűen választ, egyébként pedig az optimálisnak gondolt akciót választja. Az akció végrehajtása a manipulátorok beállítását jelenti. A manipulátorok hatnak a külső környezetre, és ezen környezet megváltozását észleli a rendszer a következő iterációban.

Látható, hogy az algoritmus alapvetően három jól elkülöníthető részből áll: a modellépítésből, az értékelés számolásából és az akció kiválasztás részekből. Ezeket egymástól függetlenül cserélhetjük. Az algoritmus motorját a 3-as lépés, az értékelések újraszámolása jelenti. A következő részt az algoritmus ezen lépésének szenteltük, egyenlőre azon feltétel mellett, hogy a mohó operátor pontosan ismert. Ezután egy olyan algoritmust tekintünk, amelyben már a kontroll stratégiának is jut szerep (4. lépés), míg végül azt az esetet tekintjük, amikor a döntési folyamat modellje, azaz G sem ismert (2. lépés).

4.1 Aszinkron Dinamikus Programozás

4.1.1 Egy általános aszinkron konvergencia tétel

Az elkövetkezendőkben bizonyított tételek végeredményben mind Bertsekas és Tsitsiklis ún. aszinkron konvergencia tételén alapulnak [3]. Ebben a fejezetben ennek a tételnek egy egyszerűsített formájával ismerkedünk meg – bizonyítás nélkül.

4.1.1 DEFINÍCIÓ *Legyen $T : \mathbf{R}^X \to \mathbf{R}^X$ tetszőleges leképezés, $v_0 \in \mathbf{R}^X$, $U_t \subseteq \{1, \ldots, n\}$ nemüres halmazok sorozata $(t \geq 0)$-ra. Ekkor a*

$$v_{t+1}(x) = \begin{cases} v_t(x), & x \notin U_t; \\ (Tv_t)(x), & x \in U_t. \end{cases}$$

iterációt gyengén aszinkron iterációnak *nevezzük. Ha $x \in U_t$, akkor azt mondjuk, hogy $v_t(x)$-et "frissítjük" a t-edik lépésben.*

Legyen X véges halmaz és rögzítsük a $T : \mathbf{R}^X \to \mathbf{R}^X$ operátort.

4.1.2 TÉTEL *Legyen a $\{V(k)\}$ olyan nemüres halmazsorozat, hogy*

$$\ldots \subseteq V(k+1) \subseteq V(k) \subseteq \ldots \subseteq \mathbf{R}^X,$$

és tegyük fel, hogy az alábbi két feltétel teljesül:

1. Szinkron konvergencia feltétel:

   $$T(V(k)) \subseteq V(k+1)$$

   és ha minden k-ra $v_k \in V(k)$, akkor v_k konvergens és $\lim_{k\to\infty} v_k = v^*$, ahol $Tv^* = v^*$, vagyis v^* T fixpontja.

2. Téglalap feltétel: *minden k természetes számra*

   $$V(k) = V_1(k) \times \ldots \times V_n(k),$$

   ahol $|X| = n$. Azaz a $V(k)$-beli vektorok komponenseit egymástól függetlenül cserélgetve, szintén $V(k)$-beli vektort kapunk.

Ekkor tetszőleges $v_0 \in V(0)$ elemből kiindulva a gyengén aszinkron iteráció konvergens és tart a T valamely fixpontjához, feltéve, hogy minden $x \in X$ végtelen sokszor fordul elő az $\{U_t\}_{t\in\mathbf{N}}$ halmazsorozatban.

4.1.2 Az Aszinkron Dinamikus Programozás algoritmus konvergenciája

Ebben a részben a $v_{t+1} = Gv_t$ DP iteráció aszinkronná tételének lehetségeségességével foglalkozunk. Ehhez szükségünk van az alábbi lemmára:

4.1.1 Lemma *Legyen $0 < \gamma < 1$, (X, A, T, c) véges Markov döntési probléma adott. Ekkor léteznek olyan $\lambda_{min}, \lambda_{max}$ valós számok, hogy minden $\lambda_1 \leq \lambda_{min}$ és $\lambda_2 \leq \lambda_{max}$ valós szám esetén ha $v_{min}(x) = \lambda_1, v_{max}(x) = \lambda_2$ $(x \in X)$, akkor*

$$v_{min} \leq Gv_{min}, \qquad Gv_{max} \leq v_{max},$$

ahol a G operátor a mohó értékeléshez tartozó operátor (lásd a (3.3) egyenletet).

Bizonyítás. Legyen $\lambda \in \mathbf{R}$, és $v(x) = \lambda, (x \in X)$. Ekkor

$$(Gv)(x) = \min_{a \in A} \max_{y \in T(x,a)} (c(x,a,y) + \gamma v(y)) = \min_{a \in A} \max_{y \in T(x,a)} c(x,a,y) + \gamma\lambda.$$

Ha tehát

$$\hat{c}(x) = \min_{a \in A} \max_{y \in T(x,a)} c(x,a,y),$$

akkor $(Gv)(x) = \gamma\lambda + \hat{c}(x)$. Legyen ezután

$$\lambda_{min} = \frac{1}{1-\gamma} \min_{x \in X} \hat{c}(x)$$

és

$$\lambda_{max} = \frac{1}{1-\gamma} \max_{x \in X} \hat{c}(x).$$

Állítjuk, hogy ezen λ_{min} és λ_{max} értékek megfelelnek a tétel feltételeinek. Legyen először $\lambda_1 \leq \lambda_{min}$ teszőleges és $v(x) \equiv \lambda_1$. Megmutatjuk, hogy $(Gv)(x) \geq v(x)$ áll minden $x \in X$-re. Valóban:

$$(Gv)(x) = \gamma\lambda_1 + \hat{c}(x) \geq \gamma\lambda_1 + \min_{x \in X} \hat{c}(x) = \gamma\lambda_1 + (1-\gamma)\lambda_{min} \geq \lambda_1.$$

Másrészről legyen $\lambda_2 \geq \lambda_{max}$ teszőleges és $v(x) = \lambda_2, x \in X$. Ekkor $(Gv)(x) \leq v(x)$ áll minden $x \in X$-re, ugyanis

$$(Gv)(x) = \gamma\lambda_2 + \hat{c}(x) \leq \gamma\lambda_2 + \max_{x \in X} \hat{c}(x) = \gamma\lambda_2 + (1-\gamma)\lambda_{max} \leq \lambda_2.$$

□

4.1.3 TÉTEL *Legyen $v_0 \in \mathbf{R}^X$, ahol X az állapotok véges halmaza. Vegyük a következő gyengén aszinkron iterációt:*

$$v_{t+1}(x) = \begin{cases} v_t(x), & x \notin U_t; \\ (Gv_t)(x), & x \in U_t; \end{cases}$$

ahol a G a a mohó operátor, $U_t \subseteq X$ pedig tetszőleges nemüres halmazok. $\lim_{t\to\infty} v_t = v^*$, *ahol v^* az optimális költségfüggvény, feltéve, hogy minden x állapot végtelen sokszor kerül "frissítésre" (minden x végtelen sokszor fordul elő az $\{U_t\}$ halmazrendszerben).*

Bizonyítás. A 4.1.2 aszinkron konvergencia tételt alkalmazzuk. Ehhez kell konstruálnunk a $V(k)$ halmazokat, majd megmutatjuk, hogy teljesülnek a szinkron konvergencia és a téglalap feltételek. Legyen

$$V(k) = \{v \in \mathbf{R}^X | G^k v_{min} \leq v \leq G^k v_{max}\},$$

ahol

$$v_{min}(x) = \min\left(\lambda_{min}, \min\left(v_0(x), v^*(x)\right)\right), \qquad x \in X$$

és

$$v_{max}(x) = \max\left(\lambda_{max}, \max\left(v_0(x), v^*(x)\right)\right), \qquad x \in X,$$

ahol λ_{min} és λ_{max} az előző lemmában definiált értékek.

Először meg kell mutatnunk, hogy $V(k+1) \subseteq V(k)$ áll minden k-ra. Legyen tehát $v \in V(k+1)$ tetszőleges. Mivel $v_{min} \equiv \lambda \leq \lambda_{min}$, ezért az előző lemma szerint $v_{min} \leq G v_{min}$. Mivel G monoton, mindkét oldalra alkalmazva G-t kapjuk, hogy $G v_{min} \leq G^2 v_{min}$, ezt folytatva indukcióval következik, hogy

$$G^k v_{min} \leq G^{k+1} v_{min},$$

Hasonló módon bizonyítható, hogy

$$G^k v_{max} \geq G^{k+1} v_{max}.$$

Ezekből $V(k+1)$ definícióját felhasználva kapjuk, hogy $G^k v_{min} \leq v \leq G^k v_{max}$, vagyis $v \in V(k)$. Meg kell még mutatni, hogy

$$G(V(k)) \subset V(k+1).$$

Legyen $v \in V(k)$ tetszőleges. Ekkor

$$G^k v_{min} \leq v \leq G^k v_{max}.$$

Alkalmazva a G-t és kihasználva G monotonitását kapjuk, hogy

$$G^{k+1} v_{min} \leq Gv \leq G^{k+1} v_{max},$$

vagyis $Gv \in V(k+1)$. Legyen most $v_k \in V(k)$ minden k-ra. Megmutatjuk, hogy v_k konvergens és határértéke G fixpontja. Mivel $G^k v_{min} \leq v_k \leq G^k v_{max}$ áll minden k-ra, és $\lim_{k\to\infty} G^k v_{min} = v^* = \lim_{k\to\infty} G^k v_{max}$ (mivel G kontrakció), ezért a rendőrelv alapján következik, hogy $\lim_{k\to\infty} v_k = v^*$. A téglalap feltétel teljesülése egyszerűen belátható. A megfelelő $V_x(k)$ halmazok:

$$V_x(k) = \{r \in \mathbf{R} | (G^k v_{min})(x) \leq r \leq (G^k v_{max})(x)\}, \qquad x \in X.$$

Ezek után már alkalmazhatjuk az aszinkron konvergencia tételt, amiből következik a tétel állítása. □

4.2 Valós-idejű Dinamikus Programozás

Ebben a részben megmutatjuk, hogy a mohó kontroll stratégia követése asszimptotikusan optimális viselkedéshez vezet. Először azonban be kell vezetnünk a nem-stacionér politikák fogalmát.

4.2.1 DEFINÍCIÓ π nem-stacionér politika, *ha* $\pi = (\pi_0, \pi_1, \ldots)$, *ahol*

$$\begin{aligned} \pi_0 &: X \to A, \\ \pi_1 &: X \times (X \times A) \to A, \\ &\vdots \\ \pi_t &: X \times (X \times A)^t \to A, \vdots \end{aligned}$$

A tanuló algoritmusok általában nem stacionér politikát követnek, hiszen az aktuális akció választása attól (és csak attól) függ, hogy előzőleg a rendszer milyen utat járt be – eltekintve a kezdeti feltételektől.

4.2.2 DEFINÍCIÓ *Egy* π *politika* asszimptotikusan optimális, *ha minden*

$$(x_0, a_0, x_1, a_1, \ldots, x_t, a_t, \ldots)$$

π *szerinti* lehetséges *trajektóriára van olyan* T_0, *hogy minden* $t > T_0$*-ra,* $a_t \in A^*(x_t)$, *ahol*

$$A^*(x) = \operatorname*{Argmin}_{a\in A} \max_{y\in T(x,a)} (c(x,a,y) + \gamma v^*(y)).$$

4.2.3 MEGJEGYZÉS Azért nem írhatjuk, hogy

$$a_t = \operatorname*{argmin}_{a \in A} \max_{y \in T(x,a)} (c(x,a,y) + \gamma v^*(y)),$$

mert a minimális értékelés több akcióra is előfordulhat. Ilyenkor választanunk kell. Egyik lehetséges megoldás, hogy rendezzük az akciókat és ez alapján választunk.

Az alábbi tétel bizonyítása előtt jegyezzük meg, hogy az eddigi eredményeinket nem befolyásolja, ha a Markov döntési probléma definícióját úgy modósítjuk, hogy minden $x \in X$ állapotra megszorítjuk az x-ben alkalmazható akciók halmazát valamely $A(x) \subseteq A$ halmazra.

4.2.4 TÉTEL (VALÓS-IDEJŰ DINAMIKUS PROGRAMOZÁS) *Legyen π az a nem-stacionér politika, amelyre*

$$\pi_t(x, a_{t-1}, x_{t-1}, a_{t-2}, x_{t-2}, \ldots, a_0, x_0) \in \operatorname*{Argmin}_{a \in A} \max_{y \in T(x,a)} (c(x,a,y) + \gamma v_t(y)),$$

ahol

$$v_{t+1}(x) = \begin{cases} v_t(x), & x \notin U_t; \\ (Gv_t)(x), & x \in U_t \end{cases}$$

és $v_0 \in B(X)$ olyan, hogy $v_0 \leq v^*$ és $x_t \in U_t$. Ez a $\pi = (\pi_0, \ldots, \pi_t, \ldots)$ politika asszimptotikusan optimális.

Bizonyítás. Először azt az esetet bizonyítjuk, amikor $U_t = \{x_t\}$. Vegyünk egy lehetséges π szerinti trajektóriát, legyen ez $p = ((x_0, a_0), (x_1, a_1), \ldots, (x_t, a_t), \ldots)$. Legyen $U^\infty \subseteq X \times A$ azon (x,a) párok halmaza, amelyre (x,a) pár végtelen sokszor fordul elő p-ben. $U^\infty \neq \emptyset$, mivel mind X, mind A végesek. Legyen $X^\infty = \{x \in X | (x,a) \in U^\infty\}$. Könnyen látható, hogy ha $(x,a) \in U^\infty$ és $y \in T(x,a)$, akkor $y \in X^\infty$. Legyen

$$A^\infty(x) = \{a \in A | (x,a) \in U^\infty\}.$$

Ekkor v_t iterációja a következőképpen írható:

$$v_{t+1}(x) = \min \left(\min_{a \in A^\infty(x)} \left(\max_{y \in T(x,a)} (c(x,a,y) + \gamma v_t(y)) \right), \min_{a \in A \setminus A^\infty(x)} \left(\max_{y \in T(x,a)} (c(x,a,y) + \gamma v_t(y)) \right) \right), \tag{4.1}$$

ahol a $\min_{a\in A}$-t egyszerűen csak szétbontottuk két minimumra. Tudjuk, hogy létezik olyan T_0, hogy ha $t > T_0$, akkor $(x_t, a_t) \in U^\infty$. Ekkora t-re tehát $a_t \in A^\infty(x_t)$. Vegyünk egy ilyen t-t; a_t definíciója szerint:

$$\max_{y\in T(x_t,a_t)} (c(x_t, a_t, y) + \gamma v_t(y)) = \min_{a\in A} \max_{y\in T(x,a)} (c(x_t, a, y) + \gamma v_t(y)),$$

tehát

$$\min_{a\in A} \max_{y\in T(x_t,a)} (c(x_t, a, y) + \gamma v_t(y)) = \min_{a\in A^\infty(x_t)} \max_{y\in T(x_t,a)} (c(x, a, y) + \gamma v_t(y)).$$

Tehát a (4.1)-ben a második minimum elhagyható, azaz:

$$v_{t+1}(x) = \min_{a\in A^\infty(x)} \max_{y\in T(x,a)} (c(x, a, y) + \gamma v_t(y)).$$

Ezután vegyük az $F^\infty = (X^\infty, A^\infty, T, c)$ Markov döntési problémát . Látható, hogy az F^∞ feladat része az eredeti feladatnak, megszorítva arra a részre, amit a döntéshozó, aki π-t hajtja végre, végtelen sokszor bejár. Legyen $\hat{v}_0 = v_{T_0}|_{X^\infty}$, vagyis v_{T_0} megszorítottja az X^∞ halmazra. Most a $\hat{v_0}$ kezdőértékre és F^∞-re alkalmazhatjuk az aszinkron dinamikus programozás tételét (4.1.3 Tétel), ugyanis most már bármely $x \in X^\infty$ végtelen sokszor fordul elő az U_t halmazokban. Következésképpen $\lim_{t\to\infty} v_t|_{X^\infty} = u^*$, ahol u^* az F^∞ feladathoz tartozó optimális költségfüggvény. Világos, hogy $v^*|_{X^\infty} \le u^*$, mivel az F^∞ feladat az eredeti feladat megszorítása. Másrészt mivel $v_0 \le v^*$, így indukcióval könnyen jön, hogy minden t-re $v_t \le v^*$ is áll, ugyanis

$$v_{t+1}(x) = (Gv_t)(x) \le (Gv^*)(x) = v^*(x).$$

Tehát $\lim_{t\to\infty} v_t|_{X^\infty} = u^* \le v^*|_{X^\infty}$ is áll, és így

$$v^*|_{X^\infty} = u^*.$$

Most már csak azt kell megmutatni, hogy egy idő után a rendszer csupa optimális akciót választ. Tudjuk, hogy megfelelően nagy t-re

$$a_t \in \operatorname{Argmin}_{a\in A^\infty(x_t)} \max_{y\in T(x_t,a)} (c(x_t, a, y) + \gamma v_t(y)).$$

Elég belátni, hogy van olyan T_1 időpont, hogy ha $t > T_1$, akkor

$$a_t \in \operatorname{Argmin}_{a\in A^\infty(x_t)} \max_{y\in T(x_t,a)} (c(x_t, a, y) + \gamma u^*(y)) = A^*_{F^\infty}(x_t).$$

Ezt az állítást Szepesvári megmutatta (szóbeli közlés). A bizonyítás egyszerűen kiterjeszthehető arra az esetre, ha U_t szigorúan bővebb mint $\{x_t\}$. □

Hasonló eredményeket Korf [7], valamint Barto és társai bizonyítottak a várható lecsengetett összköltség esetére [1]. Ennek megfelelően az ő eredményeik kissé más bizonyítási módszereket igényeltek. Megjegyezzük, hogy az itt leírt módszerek más, egyéb kritériumokhoz tartozó döntési problémákra is alkalmazhatóak (ld. [14]).

4.3 Adaptív Aszinkron Dinamikus Programozás

Tekintsük most a 4.2 táblán látható algoritmust. Az algoritmusban ϵ_t egy véletlen

4.2. táblázat: Adaptív aszinkron dinamikus programozási algoritmus

$T(x,a) := \emptyset, v_0 \leq v^*$.
`repeat forever{`
1. x_t érzékelése.
2. Modell módosítása: $T(x_{t-1}, a_{t-1}) := T(x_{t-1}, a_{t-1}) \cup \{x_t\}$.
3. Értékelések újraszámolása:

$$v_{t+1}(x) = \begin{cases} v_t(x), & x \notin U_t; \\ (Gv_t)(x), & x \in U_t. \end{cases}$$

4. Akció kiválasztás: $a_t \in A^*(x_t, \epsilon_t)$.
5. Akció végrehajtása.
6. $t := t + 1$.

`}`

menyiség, mely azt hivatott elérni, hogy minden állapotban minden akciót végtelen sokszor válasszunk. Feltételezzük, hogy ezeknek a véletlen mennyiségeknek a hatása asszimptotikusan eltűnik és végül az algoritmus mindig a mohó akciót választja. Világos, hogy a végtelen sokszor bejárt térrész (X^∞) modelljét a fenti algoritmus véges időn belül (1 valószínűséggel) felépíti. Ha ekkor $v_t \leq v^*$, akkor a 4.2.4 Tétel alapján a fenti algoritmus sztochasztikusan asszimptotikusan optimális lesz; azaz $\mathrm{Prob}(a_t \in A^*(x_t)) \to 1, t \to \infty$. Ha a döntésekben a véletlen elem nem szerepelne, akkor az épített modell a végtelenszer bejárt térrészben sem tartana az valódi modellhez – ugyan $v_t \leq v^*$ teljesülne, ám előfordulhatna, hogy $\lim_{t\to\infty} v_t|_{X^\infty} < v^*|_{X^\infty}$.

4.3.1 Tétel *A fenti algoritmus sztochasztikusan asszimptotikusan optimális.*

Bizonyítás. Mivel a rendszer véges időn belül 1 valószínűséggel pontos modellt épít, így ezekután az algoritmus átalakul aszinkron dinamikus programozássá (a 3-as lépés) és ebből a 4.2.4 Tétel alapján már következik is az állítás. □

4.4 A DC modell

Ebben a részben bemutatunk egy lehetséges implementációt egy adaptív aszinkron algoritmust használó rendszerre. A rendszer az ún. DC (**d**ynamic **c**oncept) modellt használja a döntési modell közelítésére [15].

A rendszer és környezete interakcióját a rendszer szemszögéből az $x_0, a_0, x_1, a_1, \ldots$ állapot–akció sorozat teljesen leírja. Ezt a sorozatot hármasokra tagolva jutunk el az *elemi triplethez*: $\tau_t = (x_t, a_t, x_{t+1})$. A DC modell tárolási módszerében az elemi tripletek alapelemeket jelentenek, amelyekből a rendszer építkezik. A belső reprezentációt egy $G = (V, E)$, véges, irányított gráf segítségével implementáljuk, ahol a V a gráf csúcsainak halmazát, E a gráf éleinek halmazát jelöli és V az elemi tripletek halmazából veszi elemeit.

A gráf egy adott időpillanatban az addig összegyűjtött tapasztalatokat tárolja. Első megközelítésben csúcsainak halmazát, mint az adott időpillanatig megtapasztalt elemi események egy részhalmazát definiáljuk:

$$V \subseteq \{\tau_1, \tau_2, \ldots, \tau_t\}.$$

A gráf élei azt reprezentálják, hogy az összekötött elemi tripletek hogyan következhetnek egymás után. A következő axiómák teljesülését követeljük meg:

- Egy (x, a, y) triplet csak akkor lehet a gráfban, ha $y \in T(x, a)$.
- Minden elemi triplet csak egyszer fordulhat elő a gráfban.
- Egy $((x, a, y), (z, b, q))$ él pontosan akkor lehet a gráfban, ha teljesül a két csúcspont-tripletre az ún. *kompatibilitási reláció*, vagyis ha $y = z$.

Az X-en definiált költségfüggvényt most egy V-n definiált függvénnyel helyettesítjük. Hogyan lehet egy V-n alkalmasan definiált függvényből v^*-ot visszakapni?

Legyen $\mathcal{T} = \{(x,a,y)|x \in X, a \in A, y \in T(x,a)\}$ a lehetséges tripletek halmaza és legyen R egy $R : \mathcal{T} \to \mathbf{R}$ leképezés. Rendeljük hozzá ehhez az R leképezéshez azt a $Q_R : X \times A \to \mathbf{R}$ függvényt, melyre

$$Q_R(x,a) = \max_{y \in X} \{R(x,a,y)|(x,a,y) \in \mathcal{T}\}. \tag{4.2}$$

Most már ha adott egy $Q : X \times A \to \mathbf{R}$ függvény, akkor legyen

$$v_Q(x) = \min_{a \in A} Q(x,a)$$

Végeredményben ezzel az R függvényhez is hozzárendeltünk egy $v_R : X \to \mathbf{R}$ függvényt:

$$v_R = v_{Q_R}.$$

A DC modell tanuló-algoritmusa a 4.3 táblán látható. A gráf módosítása alatt

4.3. táblázat: Az adaptív aszinkron tanulási algoritmus DC modellre kidolgozott implementációja.

`repeat forever{`
1. x_t érzékelése.
2. A gráf módosítása (x_{t-1}, a_{t-1}, x_t) elemi triplettel.
3. Értékelések újraszámolása:

$$R_{t+1}(\tau) = \begin{cases} R_t(\tau), & \tau \notin U_t; \\ c(x,a,y) + \gamma v_{R_t}(y), & \tau = (x,a,y) \in U_t. \end{cases}$$

4. Akció kiválasztás: $a_t = \mu(x_t, v_{R_{t+1}})$.
5. Akció végrehajtása.
6. $t := t + 1$.

`}`

a csúcshalmaz kibővítését értjük az (x_{t-1}, a_{t-1}, x_t) triplettel, valamint az élhalmaz kibővítését mindazon élekkel, amelyek a kompatibilitási feltétellel összeférnek. Mivel a gráfot folyamatosan építjük, előfordulhat, hogy valamely x-re nincs $(x,a,y) \in V$ triplet. Ekkor $v_{R_t}(x) = c_{min}$ definíció szerint, ahol c_{min} a közvetlen költségek egy alsó becslése. $Q_{R_t}(x,a)$ definíciója hasonlóan értelmezhető. Könnyen látszik, hogy a fenti algoritmusra alkalmazhatjuk az adaptív aszinkron dinamikus programozás tételét (4.3.1 Tétel) miáltal kapjuk, hogy $v_{R_t}|_{X^\infty}$ konvergál $v^*|_{X^\infty}$-hez.

Az U_t halmazt a következőképp definiáltjuk:

$U_t = \{(x,a,y) \in V | x = x_t$, vagy létezik irányított út (x,a,y)-ból (x_{t-1}, a_{t-1}, x_t)-be$\}$.

Jól látható, hogy az U_t ilyen választása biztosítja, hogy csak azon tripletek értékelését változtatjuk, amelyeknek az értékelése megváltozhat az új csomópont bevételével. A gráf éleinek tárolását általában a 3. lépésben tesszük meg a frissítéseknél, amikoris a gráfban az irányítással ellentétesen haladva felfrissítjük a triplet-értékeléseket.

5. Fejezet

Általánosítás

5.1 Fogalmak

Egy explicit költségfüggvényt használó algoritmus, mint pl. egy, a DC modellt használó rendszer, egyszerű környezetekben kielégítő eredményeket érhet el, azonban nagyobb állapotterekben már nem képes hatékonyan működni. E korlátot átléphetjük "fogalmak" bevezetésével, azaz az állapotok alkalmas csoportosításával.

A fogalomalkotás nagyban növelheti rendszer tudását, hiszen a fogalmakat egyedi példákból vezetjük ugyan le, de alkalmazásuk nem szorítkozik ezen példákra: alkalmazhatók új objektumok és események osztályozására is. Így a rendszer a megfelelő fogalmak kialakítása után olyan környezetben is jól működhet, amelyben azelőtt nem volt. Fogalmak nélkül a rendszer szenzori–motoros és gondolkodási folyamatai csak egyszerű, véges stimulus halmazra "beugró" reflexszerű akciókra korlátozódnak. Láthattuk, hogy az aktuátorok hatása általában nem egyértelmű hatásokat okoz a rendszer állapotának megváltozásában. Minnél inkább egyértelmű ez hatás, annál könnyebb a megfelelő kontrollstratégia kidolgozása. A rendszer szabályok kialakítására törekszik. A fogalomalkotás a szabályok felállításában is segít. A külvilág belső reprezentációja és a rendszer környezetéről kialakított fogalmai szorosan összefonódnak. A fogalmak sokszor implicite vannak jelen a belső reprezentációban. A fogalmakat a rendszer abduktív képessége segítségével szűri ki a cselekvésekhez kapcsolódó tapasztalatai közül. Az elmúlt években számos modell alakult ki a fogalomalkotás terén. Tipikus a mesterséges intelligencia (MI) fogalom definíciója,

amely azt mondja, hogy a fogalmak a világ objektumaira és eseményeire vonatkozó szavak és kifejezések reprezentációs ekvivalensei. A fogalmakat *jellegzetességek* definiálják. Ezen jellegzetességek az objektumok és események különböző tulajdonságaihoz tartoznak. Például az asztal fogalma magában foglalja azt a jellegzetességet, ami azt jelöli, hogy vannak lábai. Mivel a jellegzetességek az objektumok és események invariáns tulajdonságait reprezentálják, olybá veszik, hogy szemantikus értékük ezzel definiált. A klasszikus modell azt feltételezi, hogy a fogalmak és a fogalmak közti hierarchiákat reprezentáló kapcsolatok között világos határok vannak. A jellegzetességek természete és funkciója határozza meg a fogalmak határának világos elkülönülését. Az elmúlt évtizedben más modelleket is kidolgoztak. A valószínűségeken alapuló modell, súlyokat rendel a fogalmakat definiáló jellegzetességekhez, csakúgy, mint a fogalmakat definiáló halmazok közé. Ezt a reprezentációt egy küszöb–alapú döntési eljárással ötvözik. A példa–alapú modellekben a fogalmak csak példákon keresztül definiáltak. Ez a modell szintén alkalmas a fogalomhoz tartozás árnyalatainak és tipikusságának kifejezésére. Az elkövetkezőkben tárgyalandó fogalomalkotási koncepció szakít a hagyományos módszerekkel. A fentebb vázolt modellek egyike sem adja a fogalom formálás adekvált megközelítését. A feltételezés, hogy a fogalmak jellegzetességek által definiáltak, mellőzi azokat a tényezőket, amelyek a fogalmak kialakulását és használatát meghatározzák. A kontextusbeli tényezők szerepe megváltoztatja a jellegzetességek relatív fontosságát. Ezek a modellek mind feltételezik, hogy a világ a rendszertől független, objektív igazságokat tartalmaz és a rendszer feladata, hogy rájöjjön ezekre az igazságokra: Az új fogalomalkotási koncepció a következő pontokba szedhető :

- Elvetjük, hogy a világ olyan *a priori* információt (logikai, szükséges azonosságokat) hordoz, melyet a rendszernek igyekeznie kell megtanulnia (és belső reprezentáció formájában) kódolnia. Ehelyett adatok vannak benne, amelyből a rendszernek kell az információt kiszűrnie. Az információszerzés értelme, hogy a rendszer cél–orientált viselkedéseit hatékonyabban tudja megvalósítani.

- A fogalom kilakítás folyamata nem elkülönült, absztrakt szintaktikus rész a rendszerben, hanem szorosan kapcsolódik a rendszer szenzoros és motoros

rendszereihez.

- Van korreláció a világ invariáns tulajdonságai és a belső reprezentáció között, de nem motiválunk szemantikát ezen az alapon. A rendszer–környezet együtthatás a folytonos világot meglehetősen stabil reprezentációba transzformálja át. Ezt a transzformációt keresve a rendszer állandóan a világ invariáns mintázatai vagy tulajdonságai után kutat.

- A kódolás reprezentációs tartalma *cél–orientáltan* jelenik meg. A megfelelés a világ invariáns jelenségei és a belső reprezentációjuk között nem elegendő ahhoz, hogy a belső reprezentációhoz szemantikus értékeket rendeljünk. A szemantika hozzárendelése célra irányuló kell legyen.

Egy szimulációban egy tanult DC modell viselkedését figyelve az vehetnénk észre, hogy hasonló szituációkban hasonlóképp viselkedik. Ez a szituációt leíró fogalom, és a fogalomhoz kapcsolódó cselekvés jelenlétére utal, tehát a DC modell már készít fogalmakat. Ezek a fogalmak implicite megtalálhatók a belső reprezentációban ún. *szétszórt* reprezentációban. A fogalomalkotás példa alapú. Ezen fogalmak jelentése a belső reprezentáció használatával definiált. A szétszórt fogalom reprezentációi meglehetősen nagy memória igényű. Előnye azonban az, hogy meglehetősen robosztus: belső reprezentációjának sérülése esetén se tünnek el a fogalmak, legfeljebb elmosódnak. Mindazonáltal kívánatos lenne egy olyan fogalom reprezentáció, amelyben explicite is jelen vannak a fogalmak. Ez lehetővé tenné szélesebb körű használatukat a kontroll stratégiában, és olyan analóg szituációkban is "kipróbálható" lenne, amelyekkel a rendszer még nem találkozott. A fogalmak formálását példákból fogjuk levezetni. Az egy fogalomhoz tartozó példák halmazában az közös, hogy ugyanannak a célnak a teljesülését jósolják. A közös tulajdonságok kiemelését *általánosításnak* nevezzük. Tehát a modellünk a fogalmakat általánosítással fogja készíteni. A modell az ún. *gyors általánosítást* használja majd, ami azt jelenti, hogy fogalomalkotásra kerül sor ha már két olyan példát talál a rendszer, ami ugyanazt az "ötletet támogatja".

5.2 Megerősítéses tanulás kompakt állapot reprezentáció esetén

A megerősítéses tanulás kutatói korán felismerték, hogy az állapot és akciótér már közepesen bonyolult döntési probléma esetében is hatalmas lehet. Igaz, hogy a fent említett algoritmusok polinom időben futnak ha a modell ismert és γ rögzített [8], ám még ekkor is hatalmas futási időket igényelnek. Ezt kétféleképp lehet redukálni multirezolúciós idő-skálával [11,13], vagy az állapot-tér kompressziójával [16,4,9]. Ez utóbbi megoldással többen is próbálkoztak, azonban mindez ideig egyetlen módszerről sem lehetett bebizonyítani, hogy megőrzi az optimális költségfüggvényt, vagy legalább az optimális politikákat. Alább megadunk egy algoritmust és bebizonyítjuk, hogy megőrzi az optimális költségfüggvényt. Az algoritmusunk kategóriák, vagyis fogalmak segítségével dolgozik.

5.2.1 DEFINÍCIÓ *Azt mondjuk, hogy egy* S kategória *jelen van egy* x *állapotban, ha* $\chi_S(x) = 1$, *ahol* χ_S *függvény az* S *kategória karakterisztikus függvénye.*

Az S kategóriát akár tekinthetjük az állapottér egy részhalmazának is $S \subset X$. A karakterisztikus függvényének $[0, 1]$ intervallumra kiterjesztésével a "fuzzy"–vá is tehetjük a kategóriát. Jelöljük a rendszer által használható kategóriák halmazát $\mathcal{K} = P(X)$–el, $\mathcal{K}$ elemeit S–el. A most következőkben definiálunk egy adatstruktúrát, amely segítségével az általános megerősítéses algoritmusunkat felruházzuk az explicit fogalomkezelés képességével. Adjunk meg egy (X, A, T, c) döntési problémát, ahol most, hogy $c(x, a, y) = c(y)$, vagyis c csak y-nak függvénye. Legyen adott a következő adatstruktúra:

$$V = \{(S, a, y) | S \subseteq X, a \in A, y \in X\},$$

ahol S tulajdonképpen kategória, és vegyünk V-n egy értékelést :

$$R : V \to \mathbf{R}$$

5.2.2 MEGJEGYZÉS A fenti V halmaz elemeit *általánosított tripleteknek* vagy *tripleteknek* nevezzük. Könnyen látható, hogy az elemi tripletek olyan speciális tripletek, amelyeknek az első alkotója egyelemű halmaz.

Az előző fejezetben bevezetett függvények kiterjeszthetők a V adatstruktúrára is. Legyen tehát

$$Q_R(x,a) = \max\{R(S,a,y)|(S,a,y) \in V, x \in S\}. \tag{5.1}$$

Ezzel adott R függvényhez hozzárendeltünk egy $Q = Q_R : X \times A \to \mathbf{R}$ függvényt. Most adott $Q : X \times A \to \mathbf{R}$ függvényhez a szokásos módon – a

$$v_Q(x) = \min_{a \in A} Q(x,a)$$

definíció segítségével rendeljük hozzá a $v = v_Q : X \to \mathbf{R}$ függvényt.

Egy $(S,a,y) \in V$ elemnél az S kategóriában szeretnénk tárolni mindazokat az állapotokat, amelyekből az a akció az y állapotba visz. Így a $Q(x,a)$ definíciója ésszerűnek tűnik, hiszen a legrosszabb értékelést vesszük figyelembe. Tegyük fel, hogy

$$V = \{(\{x\},a,y)|x \in X, a \in A, y \in T(x,a)\}.$$

Legyen adott $v \in \mathbf{R}^X$-re $R_v(\{x\},a,y) = c(y)+\gamma v(y)$. Ekkor $R^* = R_{v^*}$-ra, $v_{Q_{R^*}} = v^*$.

Most ismertetjük azt az algoritmust, ami egy adaptív aszinkron algoritmus általánosítással ellátott változata. Adott $V = \{(S,a,y)|S \subseteq X, a \in A, y \in X\}$. Tegyük fel, hogy "$V$ teljes (a,y)-ra nézve", azaz minden $a \in A$ akcióra és $y \in X$ állapotra pontosan egy olyan $S \subseteq X$ kategória van, hogy $(S,a,y) \in V$. Ezt a kategóriát $S(a,y)$-val fogjuk jelölni. A kezdeti csúcshalmaz lehet pl. $\{(\emptyset,a,y)|(a,y) \in A \times X\}$. Adott továbbá az $R_t : V \to \mathbf{R}$ értékelés. Az algoritmus a 5.1 táblázatban látható. Az akció kiválasztás ugyanúgy történik, mint az Adaptív Aszinkron DP algoritmusnál.

5.2.3 MEGJEGYZÉS Általában $(S(a_{t-1},x_t),a_{t-1},x_t) \in U_t$.

5.2.4 MEGJEGYZÉS A fenti algoritmus alapján a DC modell könnyen felruházható az általánosítás képességével. Ehhez csak a fenti algoritmust kell kiterjeszteni gráfosított algoritmusra. Most két csúcs, pl. (S_1,a,y) triplet és az (S_2,b,z) triplet, akkor és csak akkor köthető össze, ha $y \in S_2$.

A "$\sqcup$" operáció egyfajta "egyesítési operátor", amelyre általában igaz, hogy $S \cup S' \subseteq S \sqcup S' \subseteq X$. Pontosabban a következő tulajdonságot követeljük meg:

5.1. táblázat: Adaptív aszinkron algoritmus általánosítással ellátott verziója

`repeat forever{`
1. x_t érzékelése.
2. Kategória bővítés: $S(a_{t-1}, x_t) := S(a_{t-1}, x_t) \sqcup \{x_{t-1}\}$.
3. Értékelések újraszámolása:

$$R_{t+1}(\tau) = \begin{cases} R_t(\tau), & \tau \notin U_t; \\ c(y) + \gamma v_{R_t}(y), & \tau = (S, a, y) \in U_t. \end{cases}$$

4. Akció kiválasztás: $a_t = A^*(x_t, \epsilon_t)$.
5. Akció végrehajtása.
6. $t := t + 1$.

`}`

5.2.5 DEFINÍCIÓ *Azt mondjuk, hogy* $\sqcup$ bővítés tulajdonságú, *ha adott* $S \subseteq X, x \in X$ *esetén* $x \in S \sqcup \{x\}$ *és* $S \subseteq S \sqcup \{x\}$.

Az egyesítési operáció kategória-reprezentáció függő, tehát a megvalósítást az határozza meg, hogy hogyan reprezentáljuk a kategóriákat. Tételünkhöz szükség van az alábbi két definícióra:

5.2.6 DEFINÍCIÓ x hamis megelőzője *az* $(a, y) \in A \times X$ *párnak, ha* $y \notin T(x, a)$

5.2.7 DEFINÍCIÓ x potenciális megelőzője *az* (a, y) *párnak a* $\sqcup$ *általánosítás mellett, ha van olyan* $t \in \mathbf{N}$ *természetes szám, és* $x_1, \ldots, x_t \in X$ *véges sorozat, hogy* $y \in T(x_i, a)$ *és*

$$x \in \Big[\big(\ldots((\{x_1\} \sqcup \{x_2\}) \sqcup \{x_3\})\ldots\big) \sqcup \{x_t\}\Big].$$

Ha egyetlen (a, y) párnak sincs hamis potenciális megelőzője, akkor az általánosító algoritmus lényegében egy gyengén aszinkron iterációra redukálódik. Azonban ez túl erős feltevés. Az alábbi tétel azt mutatja, hogy ennél jóval gyengébb feltevés mellett is megőrizhető az optimális költségfüggvényhez való konvergencia.

5.2.8 TÉTEL *Ha* $\sqcup$ *bővítés tulajdonságú és ha* x *az* (a, y) *pár hamis potenciális megelőzője, akkor*

$$c(y) + \gamma v^*(y) \leq Q^*(x, a), \tag{5.2}$$

valamint, minden lehetséges átmenetet végtelen sokszor tapasztal meg a rendszer, akkor az 5.1 táblázattal adott algoritmus olyan $R^* : V \to \mathbf{R}$ *függvényhez konvergál, amelyre* $v_{R^*} = v^*$ *(azaz* R^**-ból kinyerhető az optimális költségfüggvény).*

5.2.9 MEGJEGYZÉS Az (5.2) feltétel bal oldalán $c(y) + \gamma v^*(y)$ az y állapot "egy lépéssel növelt" összköltsége áll. Tehát az (5.2) feltétel azt mondja, hogy akkor nem baj ha hamis az általánosítás, ha az (a,y)-hoz hamisan bekerülő x-ben az a végrehajtásának költsége, azaz az (x, a, y) átmenet feltételezése "rövidítési" lehetőséget takar. Mivel mi a legrosszabb esetre optimalizálunk — heurisztikusan legalább is — látszik, hogy egy ilyen átmenet feltételezése nem okozhat bajt.

Bizonyítás. Vegyük észre, hogy bármely (a, y) párhoz tartozó $S = S(a, y)$ halmaz az algoritmus során csak nőhet. Mivel $S \subseteq X$ és X véges és véges sok (a, y) pár is van, ezért van olyan t_0 időpont, hogy $t > t_0$ esetén az $S(a, y)$ halmazok már nem változnak, így egy idő után már az összes *lehetséges* (x, a, y) átmenet[1] le lesz "fedve", mivel egy $(x_t = x, a_t = a, x_{t+1} = y)$ átmenet után már $S(a, y) \ni x_t = x$ lesz. Tehát az algoritmus lényegében aszinkron dinamikus programozás alakú lesz a $G : V^{\mathbf{R}} \to V^{\mathbf{R}}$

$$(GR)(S, a, y) = c(y) + \gamma v_R(y)$$

leképezésre nézve[2]. Csakúgy, mint a 4. fejezetben könnyen látható, hogy az aszinkron konvergencia tételének feltételei teljesülnek, tehát az algoritmus által számolt R_t függvények sorozata tart G fixpontjához (G kontrakció, így egyetlen egy fixpontja van). Legyen ez a fixpont $\hat{R}$:

$$\hat{R}(S, a, y) = c(y) + \gamma v_{\hat{R}}(y).$$

Bebizonyítjuk, hogy $Q_{\hat{R}} = Q^*$ ($Q_{\hat{R}}$ definíciójára nézve lásd az (5.1) egyenletet), ahol

$$Q^*(x, a) = \max_{y \in T(x,a)} (c(y) + \gamma v^*(y)).$$

Könnyű látni, hogy $Q_{\hat{R}}$ teljesíti a

$$\begin{aligned} Q_{\hat{R}}(x, a) &= \max (c(y) + \gamma v_{\hat{R}}(y) | (S, a, y) \in V, x \in S) = \\ &= \max (c(y) + \gamma \min_b Q_{\hat{R}}(y, b) | (S, a, y) \in V, x \in S), \end{aligned} \tag{5.3}$$

[1] Az (x, a, y) triplet lehetséges átmenet ha $y \in T(x, a)$)

[2] Hogy egyszerűsítsük a jelölést, ha $R \in V^{\mathbf{R}}$ és $T = (S, a, y)$, akkor $R(T) = R((S, a, y))$ helyett egyszerűen csak $R(S, a, y)$-t írunk.

illetve Q^* az

$$Q^*(x,a) = \max\left(c(y) + \gamma v^*(y) | y \in T(x,a)\right) \tag{5.4}$$

egyenletet. Mindkét fixpontegyenletnek csak egyetlen megoldása lehet. Ha a Q^* kielégítené az (5.3) fixpontegyenletet, akkor tehát $Q^* = Q_{\hat{R}}$ lenne. Mivel láttuk, hogy $T(x,a) \subseteq \{y \in X | x \in S(a,y)\}$ (az átmenetek le vannak fedve), ezért az (5.3)-ban szereplő maximum argumentumhalmaza bővebb, mint az (5.4)-ben szereplőé. Ahhoz, hogy ne nőjön a maximum, – amikor a Q^*-t (5.3) jobb oldalába írjuk be, – szükséges hogy a fölös elemek értéke a $T(x,a)$-n vett maximum, azaz $Q^*(x,a)$ alá essen. A "fölösleges" elemek éppen azon y-ok, amelyre $x \in S(a,y)$, de $y \notin T(x,a)$, azaz ezek csak hamis potenciális megelőzők lehetnek. Azonban feltettük, hogy ezekre

$$c(y) + \gamma v^*(y) \leq Q^*(x,a)$$

tehát a maximumot (5.3)-ban ezek nem befolyásolják, következésképpen Q^* valóban teljesíti (5.3)-at is. □

6. Fejezet

Számítógépes szimulációk

Az előző fejezetekben megismert matematikai modellek és algoritmusok konkrét ellenőrzésére számítógépes szimulációkat készítettem. Erre a célra egy általános szoftvert fejlesztettem ki, mellyel tetszőleges megerősítéses modell szimulációja megépíthető.

6.1 A kísérleti rendszer és környezete

Az adaptív aszinkron DP algoritmusból kétféle változat készült el. Az egyik az egyszerű DC modellt alkalmazza (WOG - Without Generalization), míg a másik a modell fogalomképzési lehetőséggel ellátott változata (WG - With Generalization). A kísérletek célja, hogy igazolja a WG modell jóságát a WOG modellel szemben. A rendszer négy részből tevődik össze:

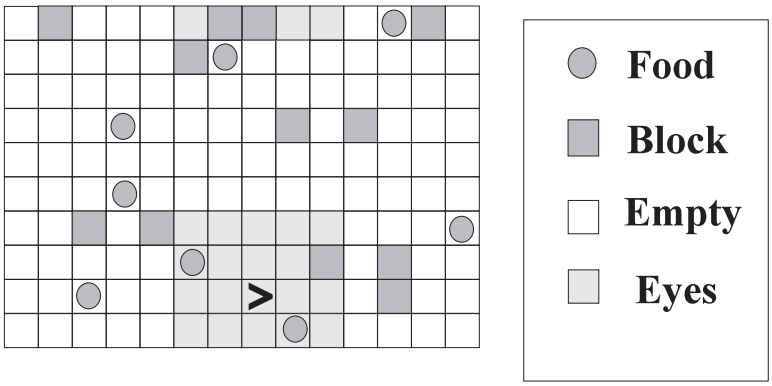
- A *Rendszer→Környezet Interface* tulajdonképpen a rendszer végrehajtó egységét képezi. Minden egyes operátorra manipulátorfüggvényeket adunk meg, melyek az operátorok környezetre gyakorolt hatását írják le.
- A *Környezet → Rendszer Interface* a rendszer szenzori része, melyben minden egyes elem egy szenzor, és a környezet ezen szenzorokat állítja be az állapotának megfelelően.
- A *Long Term Memory (LTM)* vagy *Hosszútávú Memória* az általánosítási algoritmus által létrehozott gráf.

- A *Short Term Memory (STM)* vagy *Rövidtávú Memória.* Az STM-et az események szűrésére használjuk. Ha ugyanis minden környezetből jövő eseményt felvennénk az LTM–ben, akkor ez a nagy memória igény következtében problémákat okozna. A rengeteg megjegyzett információ nagy része ráadásul a célok elérése szempontjából valószínűleg felesleges is. Az egyszerűség kedvéért csak azokat az eseményeket vesszük fel az LTM–be, amelyek a célhoz közel vannak. Pontosabban, ha T_t triplet elérte a célt, akkor felvesszük a $T_t, T_{t-1}, \ldots T_{t-k}$ tripletet, ahol k rögzített szám, amit az *STM hosszának* nevezzük. Az STM tulajdonképpen egy lyukas verem. Ezzel az eljárással azt érjük el, hogy a gráf szélességében jobban nő, mint mélységében.

A rendszernek két működési állapota van: a *tanulási állapotban* módosítja a belső reprezentációját, miközben az operátorok kiválasztására is használja. A *működési állapotban* az LTM-et a rendszer, csak használja, de nem építi. A szimulált világ, melyben a robot (rendszer) mozog, diszkrét pontokból – egy véges rács rácspontjaiból – áll (6.1 ábra). A világban csak a robot mozog, minden más statikus. A rács minden pontjában egyszerre egy elem lehet a következő objektumok közül: táplálék (F– food), akadály (B–block), üresség (E–empty), vagy a robot (R–robot). A robot a környezetét egy olyan szemen keresztül érzékeli, amelyik minden irányban lát, tehát az összes szomszédos pontot látja, egy bizonyos távolságig. A robot a környezetére a következő operátorok végrehajtásával hathat: 90°–os fordulás jobbra és balra, előrelépés, evés. A robot csak a világ üres pontjaiban tartózkodhat és az "evés" operátort csak akkor alkalmazhatja, ha az előtte levő ponton táplálék van. A robot négy irányban nézhet és ezt a száj helyzete határozza meg. Környezet lehet "tórusz" is, ami azt jelenti, hogy ha a robot az egyik oldalon kimegy, akkor a másik oldalon bejön. Ez vonatkozik a látására is, tehát ha a látótere túllépné a környezet határát, akkor a másik oldalon "folytatódik".

6.2 Állapot reprezentáció és egyesítési operátor

Az egyesítési operátor megadása előtt mindenképpen tudnunk kell a robot állapotainak tárolási módját. A rendszer szenzor-vektor formájában tárolja információit



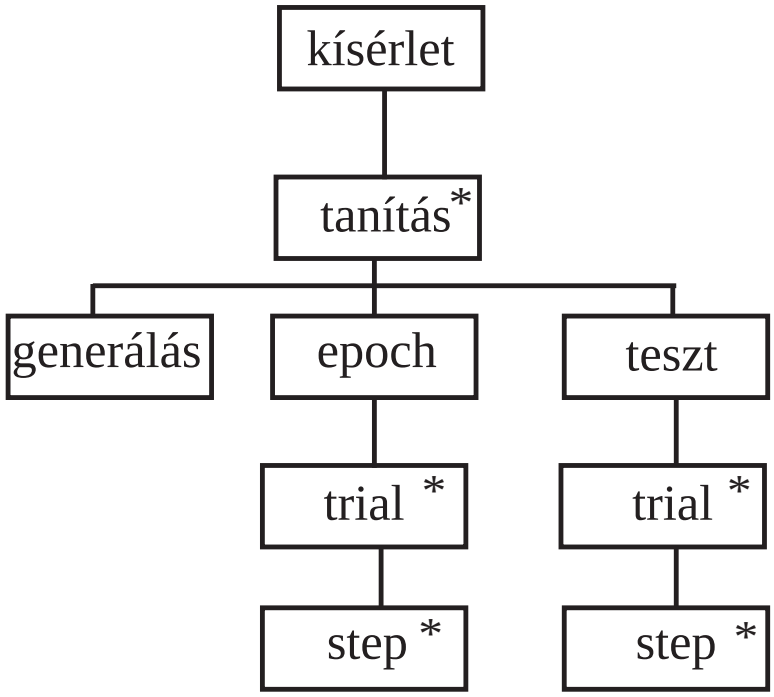
6.1. ábra: Robot és környezete
A jobbra néző robot 5×5–ös területet lát. Ebben az esetben a szenzorain a EBEEEEE-EFEEEEEBEEEEBEFEEE0 sorozatot érzékeli.

az állapotáról. Az vektor egyes pozícióin a külvilág négyféle tárgyának megfelelően négyféle szimbólum állhat: a B,R,E és az F betűk valamelyike (6.1 ábra). Ehhez vettük még az "éhség-szenzort", ami arról ad információt, hogy a robot evett-e. Ez nulla ha "éhes" és egyre vált ha sikerül ennie. Ez tulajdonképpen a reinforcement jelet ($c(y)$-t) reprezentálja. Mielőtt rátérnénk az egyesítési operátorra, definiálnunk kell a kategóriákat is. A ketegóriákat is vektorral definiáljuk. Bevezetjük a "$*$" szimbólumot a vektorokat reprezentáló szimbólumok ábécéjébe. Így az "$E*F*$" vektor pl. az $\{E\} \times A \times \{F\} \times A$ halmazt reprezentálja, ahol $A = \{R, B, F, E\}$. Vegyünk most két vektort u_1–t és u_2–t. Az egyesítéssel kepződő u vektor adott pozícióján pontosan akkor áll "$*$", ha az u_1 és u_2 megfelelő pozícióján különböző jelek állnak, egyébként a közös jel áll az adott pozíción. Látható, hogy az $u_1 \cup u_2 \subset u_1 \sqcup u_2$ tartalmazás áll a $\sqcup$ operátorra, tehát az operátor bővítés típusú.

6.3 A kísérletek módszertana

A kísérleteket 10×14–es világokban végeztük, miközben a robot szemének felbontása 3×3–as. A világok tórusz típusúak voltak. A táplálékot és az akadályt jelentő elemeket meghatározott valószínűséggel helyeztük el. Annak a valószínűségét, hogy egy rácspontra táplálék kerül $P(F)$–el és annak, hogy akadály kerül $P(B)$–vel jelöltük. A kísérletet tanulási ciklusokba szerveztük. A tanulási ciklus elején rögzített

valószínűséggel generálunk egy világot, amelyben a tanulási próbákat (trial) korszakokba szervezzük (epoch), majd egy tesztelést végzünk. Egy trial addig tart, amíg nem teljesül a cél azaz amíg a robot nem talál táplálékot, vagy amíg le nem telik az előre meghatározott próbálkozási idő (timeout). A timeout–ra azért van szükség, mert megtörténhet, hogy a robot végtelen ciklusba kerül, vagyis nem jut el a táplálékig. Ez a korlát minden esetben 1000 volt. Minden epoch 40 trialt tartalmazott, és ezután a tesztelünk (6.2 ábra). A tesztelés 500 trialt jelentett úgy, hogy a robot nem tanult. A tesztelést mindig ugyanabban, a kísérletsorozat elején rögzített tesztvilágban végeztük el. A teszt eredménye a táplálék eléréséhez szükséges átlagos lépésszám. Egy kísérlet 200 tanítási ciklusból állt. Minden ciklus végén

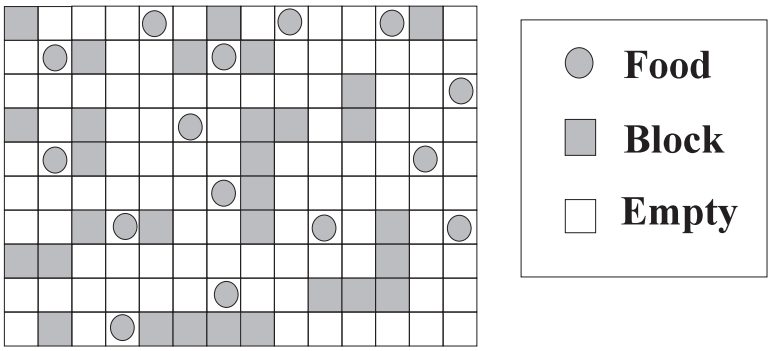


6.2. ábra: A kísérletek menete
A $*$–al jelzett eljárások ciklusok. Az egyes ciklusok ismétlési száma: tanítás: 200, epoch–trial: 40, teszt-trial: 500.

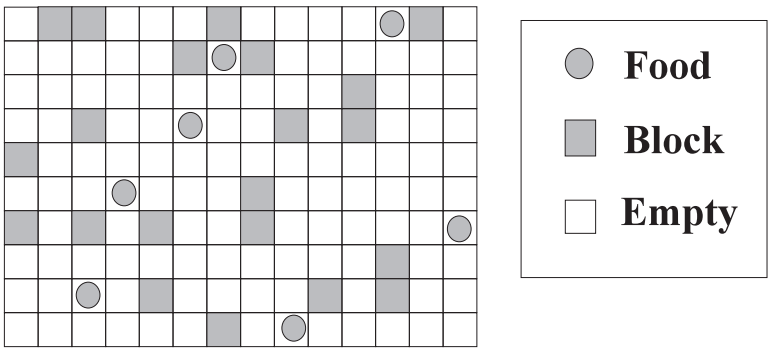
rögzítjük az aktuális teszteredményeket. A következő tanulási ciklus az előző ciklus alatt kialakított memóriával, de már más világban történik.

6.4 Tanulási eredmények

Az első kísérletsorozatban a robot sűrű világokban tanulhatott. Itt egy akadály előfordulásának valószínűsége "nagy" volt; $P(F) = 0.05, P(B) = 0.15$. A tesztvilág a 6.3 ábrán látható. A 6.4 ábrán egy tipikus sűrű világot láthatunk. Az eredményeket bemutató 6.5 ábrán jól látható, hogy a WG robot elérte az optimális lépésszámot, míg a WOG robot még a 200-adik tanulási ciklus után sem volt képes erre. A

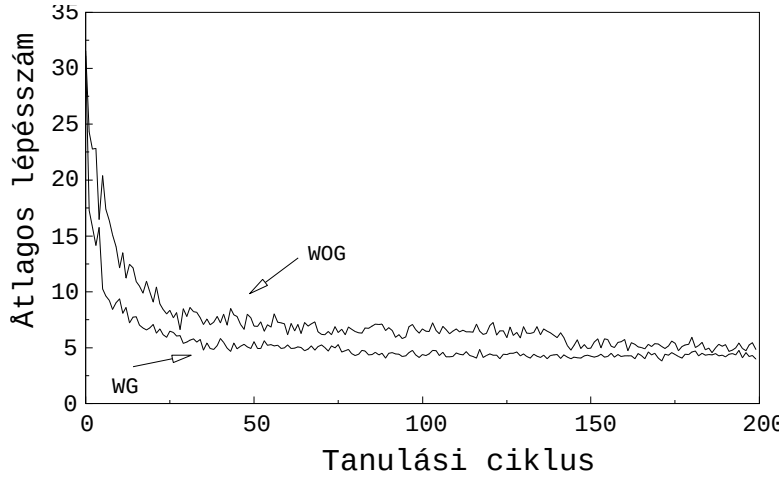


6.3. ábra: "Tesztvilág"



6.4. ábra: Tipikus sűrű világ

6.6 ábrán láthatjuk, a WG és a WOG robot által épített gráfok nagyságát. A WG robot gráfjának nagysága csupán fele a WOG roboténak, ennek ellenére jobb teljesítményt nyújtott. Ez azért lehetséges, mert a WG robotban ugyanazok az információk megtalálhatók (és még több is), mint a WOG robotban, csak a reprezentáció kompaktsága miatt, ez kevesebb helyett foglal el. A gráf csökkenése kihat a feldolgozási

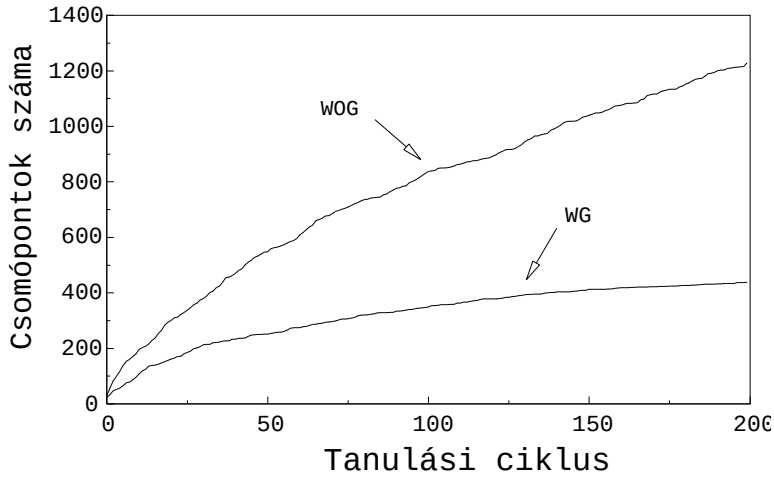


6.5. ábra: Futtatási eredmények
Az ábrán a táplálék megszerzéséhez szükséges átlagos lépészám látható a WOG és a WG robot esetében.

sebességre is. A 6.7 ábrán a "gondolkodási"-idő alakulását láthatjuk. A gondolkodási időt úgy kapjuk meg, hogy minden tanulási ciklus után a teszt másdopercben mért futási idejét elosztjuk az átlagos lépésszámmal, azaz a gondolkodási idő 500 elemi döntés (az algoritmusban 500 ciklus-lépés lefutásának) idejét adja meg. Itt is a WG robot bizonyult jobbnak, hiszen több mint kétszer gyorsabban végezte el a teszteket, mint a WG robot. A másik menyiség, ami szintén jól mutatja a WG robot fölényét, a memória-információ hányad (MIH), amit 1/(átlagos lépésszám∗gráf csomópontjainak száma)–ként definiálunk. A 6.8 ábrán látható, hogy a WG robot MIH-e több mint háromszorosa a WOG roboténak.

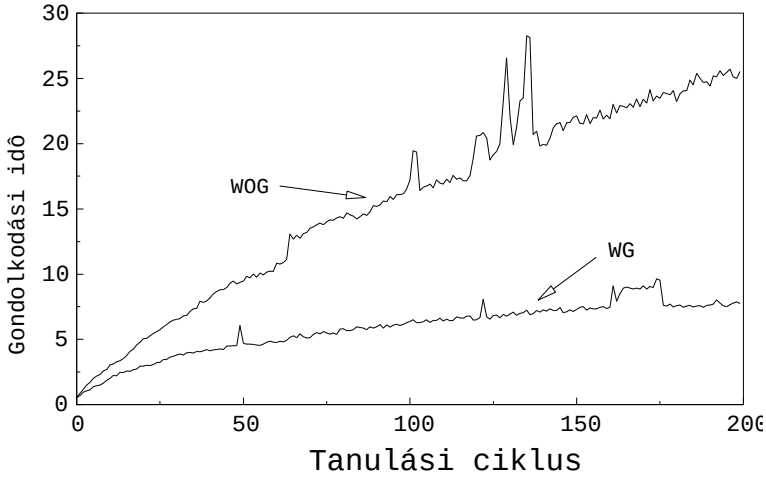
6.5 Kísérlet "ritka" világokban

Mint a dolgozat elején azt már említettük, a fogalomalkotás lehetősége egy robot számára olyan helyzetekben is előnyt jelenthet, amelyekben eddig még nem volt. A következő kísérlet ennek megmutatására irányul. Az első kísérlettel ellentétben, most a tanulási folyamatot ritka világokban futtatjuk le, míg a tesztvilág az előző kísérletbeli teszvilág marad. Azt várjuk, hogy a megváltozott – ingerszegényebb – környezet miatt teljesítmény csökkenés lép fel a két robotnál, s mivel ráadásul a

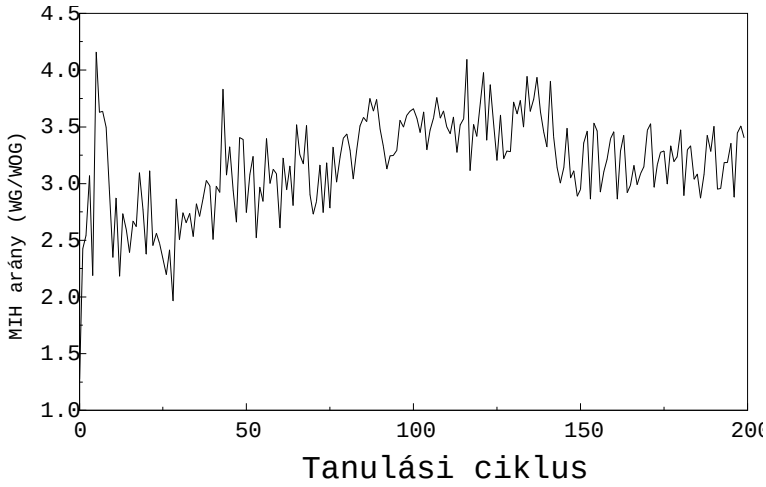


6.6. ábra: A csomópontok számának időbeli alakulása

tesztvilág a tanult ritka helyett sűrű. A világok paraméterei a következők voltak: $P(F) = 0.02, P(B) = 0.03$. Az eredményt, mint a robotok teljesítményromlását az előző kísérlethez képest, a 6.9 ábrán láthatjuk. Jól megfigyelhető, hogy a WOG robot teljesítmény romlását mutató görbe fölé megy a WG robot görbéjének. Vagyis a WOG robot átlagos lépésszáma nagyobb mértékben nőtt meg, mint a WG roboté.

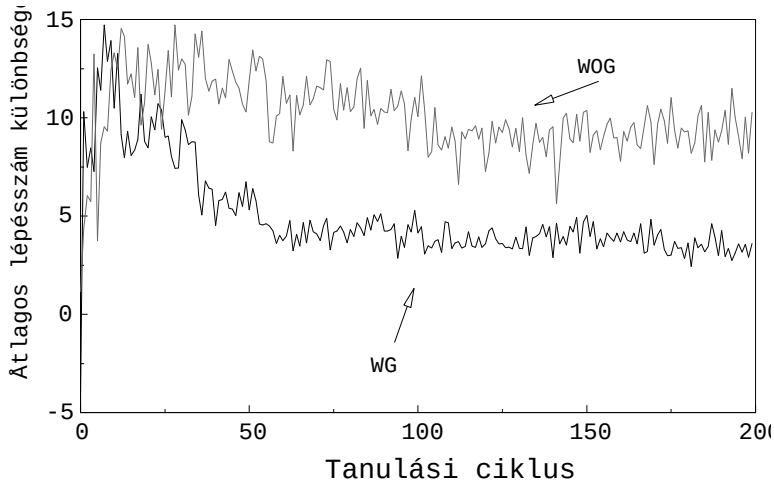


6.7. ábra: A gondolkodási idők időbeni alakulása



6.8. ábra: Memória-információ hányad arány

Ezen az ábrán a WG/WOG MIH-ének arányát látjuk, amelyet úgy kapunk, hogy minden tanulási ciklus után a WG robot MIH-ét elosztjuk a WOG robot MIH-ével. A MIH arány azt jellemzi, hogy a WG robot hányszor nagyobb hatásfokkal használja a memóriáját.



6.9. ábra: A teljesítmény romlása ritka világok esetén

A teljesítmény romlását, azaz a lépésszám növekedésének mértékét adjuk meg a ritka világban való tanulás esetén a sűrű világban való tanuláshoz viszonyítva, mindkét robot esetében.

7. Fejezet

Összefoglalás

A dolgozatban optimális döntési stratégiákat vizsgálok az ún. kockázat érzékeny Markov döntési folyamatok keretén belül. A dolgozat központi kérdése, hogy hogyan lehet optimális döntési stratégiákat kialakítani ha a döntési modellt csak részlegesen ismerjük. A dolgozatban az ún. költségfüggvény alapú algoritmusokkal foglalkozom. Az ilyen algoritmusok (döntési függvények) gyakorlati alkalmazhatóságát korlátozza, hogy a költségfüggvény pontos reprezentációját követelik meg – a tárigény az állapotok számával lineárisan nő. A dolgozatban bevezettem egy olyan csökkentett tárigényű reprezentációt, melyből az optimális politika még rekonstruálható. Megadok egy algoritmust és egy szükséges feltételt, mely mellett az algoritmus optimalitás őrző kompakt reprezentációt készít. Ennek bizonyítása három lépésben zajlik. Az eredményeket kísérletekkel is alátámasztom.

A jövőben érdemes lenne elméleti szempontból is megvizsgálni az ún. rekurzív általánosítás lehetőségét, amikor a kialakított fogalmakat a rendszer újrafelhasználhatja más fogalmak alkotására is [6]. Ennek a módszernek a buktatója, hogy a megtanult, akaratlanul és nem felismerten rossz kategóriák tovagyűrűzhetnek és végképp elronthatják a rendszer teljesítményét. Másik fontos kutatási irány annak felderítése, hogy az algoritmus miként módosítandó, ha a világ determinisztikus, de a rendszer megfigyelései pontatlanok – és így számára a világ akár nem-Markovinak is tűnhet. Ezek a kutatási irányok igen ígéretesek, mert klasszikus problémákat (mint pl. az emberi konstruktivitás, frame-ek, stb.) vizsgálnak egy új szemszögből.

Irodalom

[1] A.G. Barto, S.J. Bradtke, and S.P. Singh. *Real-time Learning and Control using Asynchronous Dynamic Programming.* Technical Report 91-57, Computer Science Department, University of Massachusetts, 1991.

[2] R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, New Jersey, 1957.

[3] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation (Numerical Methods).* Prentice-Hall, Englewood Cliffs, NJ, USA, 1989.

[4] D. Chapman and L.P. Kaelbling. *Learning from Delayed Reinforcement in a Complex Domain.* TR-90-11, Teleos Research, 1990.

[5] M. Heger. Consideration of risk in reinforcement learning. In *Proc. of the* 11^{th} *International Machine Learning Conference*, 1994. (submitted).

[6] Zs. Kalmár, Cs. Szepesvári, and A. Lőrincz. Generalized dynamic concept model as a route to construct adaptive autonomous agents. *Neural Network World*, 5:353–360, 1995.

[7] R.E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42:189–211, 1990.

[8] M.L. Littman. *Algorithms for Sequential Decision Making.* PhD thesis, Brown University, Computer Science Department, 1996.

[9] S. Mahadevan and J. Connel. Automatic programming of behavior-based robots using reinforcement learning. *AAAI-91*, 1990.

[10] S.M. Ross. *Applied Probability Models with Optimization Applications.* Holden Day, San Francisco, California, 1970.

[11] S.P. Singh. Scaling reinforcement learning algorithms by learning variable temporal resolution models. In *Proc. of the 9th Int. COnf. on Machine Learning*, Morgan Kaufmann, 1992.

[12] R. Sutton. *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Masschusetts, Amherst, MA, 1984.

[13] R.S. Sutton. TD models: modeling the world at a mixture of time scales. In *Proc. of the 12th Int. COnf. on Machine Learning*, Morgan Kaufmann, 1995.

[14] Cs. Szepesvári. A general framework for reinforcement learning. In *Proc. of ICANN'95*, pages 165–170, 1995.

[15] Cs. Szepesvári and András Lőrincz. Behavior of an adaptive self-organizing autonomous agent working with cues and competing concepts. *Adaptive Behavior*, 2(2):131–160, 1994.

[16] R. Yee. *Abstraction in Control Learning.* COINS Technical Report 92-16, Dep. of Computer and Information Science, Univ. of Massachuttes, Amherst, MA, 10003, 1992.