MARKOV DÖNTÉSI FOLYAMATOK

JUNG ÁDÁM

Szakdolgozat
Alkalmazott Matematika BSc

Témavezető: Dr. Csiszár Villő

Eötvös Loránd Tudományegyetem
Természettudományi Kar
Budapest – 2023

NYILATKOZAT

Név: JUNG ÁDÁM
ELTE Természettudományi Kar, szak: MATEMATIKA BSC
NEPTUN azonosító: B4L573
Szakdolgozat címe: MARKOV DÖNTÉSI FOLYAMATOK

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2023

a hallgató aláírása

Jung Ádám: *Markov döntési folyamatok,* Szakdolgozat, © 2023

BEVEZETŐ

A Markov döntési folyamatok témaköre alapvetően szekvenciális döntések leírására alkalmas, ahol a döntések következményei nem determinisztikusan alakulnak.

A dolgozatban tárgyalt témák a következő kérdéskörből származnak: Adottnak tekintünk egy állapotteret, amivel a valóság egy szeletét kódoljuk el. Az állapottér minden eleméhez tartozik egy jutalom (vagy költség), ami leírja, hogy mennyire kedvező (vagy kerülendő) számunkra az adott állapot. Az állapotokhoz fel vannak sorolva a megadatott döntéseink, amik közül választhatunk egyet. A környezet sztochasztikus, így egy döntés meghozása után nem tudhatjuk pontosan az állapottér melyik részébe kerülünk. A döntéseink csak befolyásolják a következő állapotok megvalósulásának valószínűségét. Feltesszük azt, hogy ezeket az eloszlásokat ismerjük a döntéshozás pillanatában.

A kérdés természetesen az, hogy *hogyan válasszunk optimális döntéseket*. Ehhez először meg kell fogalmazzuk mi alapján tekinthetünk egy döntést kedvezőbbnek a többi lehetőségnél. Ez nem csak azon múlik, hogy milyen jutalmak, vagy költségek várhatóak egy adott döntés közvetlen következményei által. Az is előfordulhat, hogy kockáztatva, vagy rövid távon áldozatot hozva, csak egy hosszabb döntéssorozat által, így az első döntésünknek csak távolabbi következményeként jutunk magas jutalmú állapotokba. Tehát ahhoz, hogy egy konkrét állapotban lássuk melyek az optimális döntések, szükségünk van egyszerre, az állapottér egészén meghatározni a legjobb döntéseket.

Ahhoz, hogy a probléma könnyen kezelhető maradjon, egy fontos feltevésünk, hogy a vizsgált döntési folyamat *Markov tulajdonságú*. Ezzel azt tesszük fel, hogy a folyamat jelenlegi állapotának ismerete ugyan annyi információt hordoz a döntéseink utáni átmenetvalószínűségekről, mintha a folyamat teljes múltját ismernénk. Bizonyos szempontból ezzel azt fejezzük ki, mintha nem lenne emlékezetünk, így nem megkepő, hogy a Markovi döntési folyamatok csak egy fejezetét alkotja a sztochasztikus optimális kontroll témakörének.

KÖSZÖNETNYILVÁNÍTÁS

Szeretnék köszönetet mondani dr. Csiszár Villő témavezetőmnek a rendszeres konzultációkért, a munkám során ejtett hibák felderítéséért, különösképpen a támaválasztással kapcsolatos nyitottságáért, a témakör részleteinek kiválasztásában adott bizalmáért, összességében az egész közös munkafolyamatért.

TARTALOMJEGYZÉK

I. rész

MARKOV DÖNTÉSI FOLYAMATOK ELMÉLETE

A dolgozat első részében a diszkrét idejű és véges állapotterű Markov-láncok elméletére alapozva bevezetjük a *Markov Döntési Folyamatok* (röviden MDP, mint Markov Decision Processes) alapvető fogalmait.

1 MARKOV-LÁNCOK

A Markov-láncokkal kapcsolatos ismereteket ebben a dolgozatban a teljesség igénye nélkül mutatjuk be. A következőkben *Dr. Csiszár Villő - Diszkrét és folytonos idejű Markov-láncok* elektronikus egyetemi jegyzet [6] és *R. G. Gallager - Discrete Stochastic Processes* [1] tankönyv alapján, csak a későbbiekben használt alapfogalmak mentén a témakör egy bevezetését adjuk. Ez utóbbi tankönyv alapján készültek a dolgozat I. részének további fejezetei is.

1.1 BEVEZETÉS ÉS ALAPFOGALMAK

1.1. Definíció. *Diszkrét idejű, véges állapotterű Markov-láncnak nevezünk egy olyan* $\{X_t\}_{t\in\mathbb{N}}$ *valószínűségi változó sorozatot, ahol*

- X_t *értékei az* $\{1,2,\dots,M\}$ *halmazból kerülnek ki minden* $t \in \mathbb{N}$ *esetén*
- $\{X_t\}_{t\in\mathbb{N}}$ *teljesíti a Markov tulajdonságot, azaz* $\forall t \in \mathbb{N}$ *és* $\forall i \in [M]$ *esetén*

$$\mathbb{P}(X_{t+1} = i \mid X_0, X_1, \dots, X_t) = \mathbb{P}(X_{t+1} = i \mid X_t) \tag{1.1}$$

1. Megjegyzés. A Markov-tulajdonság alapján tehát hiába ismerjük az egész $X_0, X_1, \dots, X_t$ "múltját" a folyamatnak, nem tudunk többet a jövőbeni fejlődéséről, mintha csak X_t értékét kapnánk meg. Ezt úgy is meg szokás fogalmazni intuitíve, hogy egy Markov-láncnak nincsen "emlékezete".

Fontos kiemelni, hogy az 1. fejezetben csak ún. *homogén* Markov-láncokat vizsgálunk, ahol az *átmenetvalószínűségek* változatlanok, azaz $\mathbb{P}(X_{t+1} = i \mid X_t = j)$ valószínűségek csak i és j értékétől függnek, t-től nem.

Adja magát, hogy az *egylépéses átmenetvalószínűségeket* egy $P \in \mathbb{R}^{M\times M}$ *átmenetmátrixba* rendezzük:

$$P_{ij} := \mathbb{P}(X_{t+1} = j \mid X_t = i) \tag{1.2}$$

Egy i állapotból *elérhető* a j állapot ($i \to j$), ha $\exists n$, hogy $\mathbb{P}(X_n = j \mid X_0 = i) > 0$. Azt mondjuk, hogy az i állapot *rekurrens*, amennyiben $(i \to j)$ maga után vonja, hogy $(j \to i)$. A nem rekurrens állapotokat *tranziensnek* nevezzük.

Az egymásból kölcsönösen elérhető állapotokat osztályokba particionálhatjuk. Mivel egy rekurrens állapotból elérhető csúcsok mind rekurrensek, minden egyes osztály vagy csak tranziens, vagy csak rekurrens állapotokat tartalmaz.

Egy i állapot periódusa az $\{n \in \mathbb{N} : \mathbb{P}(X_n = i \mid X_0 = i) > 0\}$ számok legnagyobb közös osztója. Ha a periódus $= 1$, akkor *aperiodikusnak* nevezzük az állapotot.

Egy osztály elemeinek periódusa megegyezik, így beszélhetünk periódikus, illetve aperiodikus osztályokról.

1.2. Definíció. *Egy véges állapotterű Markov-lánc*

- ergodikus *amennyiben az összes állapot egyetlen rekurrens és aperiodikus osztály része.*
- unichain *amennyiben egy rekurrens osztályt tartalmaz, azon kívül esetleg tranziens osztályokat.*
- ergodikus unichain *amennyiben a folyamat unichain és a rekurrens osztály ergodikus.*

1.3. Definíció. *Egy P átmenetmátrixú Markov-láncnak $\boldsymbol{\pi}^T \in \mathbb{R}^M$ stacionárius eloszlása, ha*

$$\sum_{i=1}^{M} \boldsymbol{\pi}_i P_{ij} = \boldsymbol{\pi}_j \quad (\forall j \in [M]); \quad \boldsymbol{\pi} \geqslant 0; \quad \sum_{i=1}^{M} \boldsymbol{\pi}_i = 1 \tag{1.3}$$

teljesülnek.

Vegyük észre, hogy ha $\mathbb{P}(X_0 = i) = \boldsymbol{\pi}_i \ \forall i$, akkor $\mathbb{P}(X_1 = j) = \sum_{i=1}^{M} \boldsymbol{\pi}_i P_{ij} = \boldsymbol{\pi}_j \ \forall i$. Így ez alapján induktíve látszik, hogy X_t eloszlása változatlan marad tetszőleges $t \in \mathbb{N}$ esetén, amennyiben a stacionárius eloszlás szerint indul a folyamat.

A stacionárius eloszlás létezésére, illetve egyértelműségére a *Perron Frobenius* tételkör ad választ. [1]

1.1. Tétel. *Legyen P egy ergodikus unichain átmenetmátrixa. Ekkor P legnagyobb valós sajátértéke $\lambda = 1$, és tetszőleges $\mu \neq \lambda$ sajátértékre $\lambda > |\mu|$. Ezenkívül*

$$\lim_{m \to \infty} P^m = e\boldsymbol{\pi} \tag{1.4}$$

ahol $\boldsymbol{\pi} \geqslant 0, \sum_i \boldsymbol{\pi}_i = 1$ feltételekkel $\boldsymbol{\pi}$ egyértelmű megoldása a $\boldsymbol{\pi}P = \boldsymbol{\pi}$ egyenletnek, illetve $Pv = v$ konstans szorzó erejéig egyértelmű megoldása $e = (1, 1, \ldots, 1)^T$.

Induktíve belátható, hogy $(P^m)_{ij} = \mathbb{P}(X_m = j \mid X_0 = i)$, azaz a P^m mátrix az m *lépéses átmenetvalószínűségeket* tartalmazza.

2. *Megjegyzés.* A $\boldsymbol{\pi}P = \boldsymbol{\pi}$ egyenletnek van egyértelmű megoldása, ha P egy unichain átmenet mátrixa. Amennyiben egy olyan Markov-láncot vizsgálunk, aminek $r > 1$ darab rekurrens osztálya van, akkor a $\boldsymbol{\pi}P = \boldsymbol{\pi}$ egyenletnek r darab különböző, lineárisan független megoldása lesz.

1.2 MARKOV-LÁNCOK JUTALOMMAL

Természetes általánosítása a Markov-láncoknak a modell, amiben a folyamat egy i állapotához egy $r_i \in \mathbb{R}$ jutalom, vagy költség van rendelve. A legfontosabb mennyiség, amit ebben a fejezetben körüljárunk az a várható értéke az összegyűjtött jutalomnak az idő haladtával.

A jutalmak bevezetése szélesíti a modellezhető jelenségek körét, és előkészíti a építőköveit a Markov döntési folyamatoknak.

3. Megjegyzés. Amennyiben úgy adódik, hogy a jutalmat egy $i \to j$ két állapot közti átmenettel természetes azonosítani, akkor mivel a várható értékét vizsgáljuk az összegyűjtött jutalomnak, könnyen átfogalmazhatjuk a problémát a fent felvetett alakra, hiszen egy i állapotból a várható jutalom ekkor $r_i = \sum_{j=1}^{M} P_{ij} r_{ij}$.

1.4. Definíció. *Egy unichain esetében a* stacionárius eloszlás időegység alatt várható jutalma *alatt a*

$$g = \sum_{i=1}^{M} \pi_i r_i \tag{1.5}$$

mennyiséget értjük, ahol π a stacionárius eloszlás.

Ez a fogalom alapvető a hosszútávon összegyűjtött jutalom elemzésekor, hiszen láttuk, hogy egy ergodikus unichain az idő előrehaladtával konvergál, olyan értelemben, hogy $\mathbb{P}(X_t = i) \approx \pi_i$ nagy t esetén. Így egy átmeneti ún. *bemelegítő* fázis után már becsülhető az n lépés alatt összegyűjtött jutalom egyszerűen ng értékkel.

1.1. Példa (Első mintamegjelenés várható ideje)**.** *Tekintsük a következő problémát: Egy majmot leültetünk egy írógép elé, amin a magyar abc 44 betűjét használhatja. Véletlenszerű, egyenletes eloszlás szerinti leütéseket feltételezve várhatóan mennyit kell várni, hogy legépelje a* "banán" *szót?*

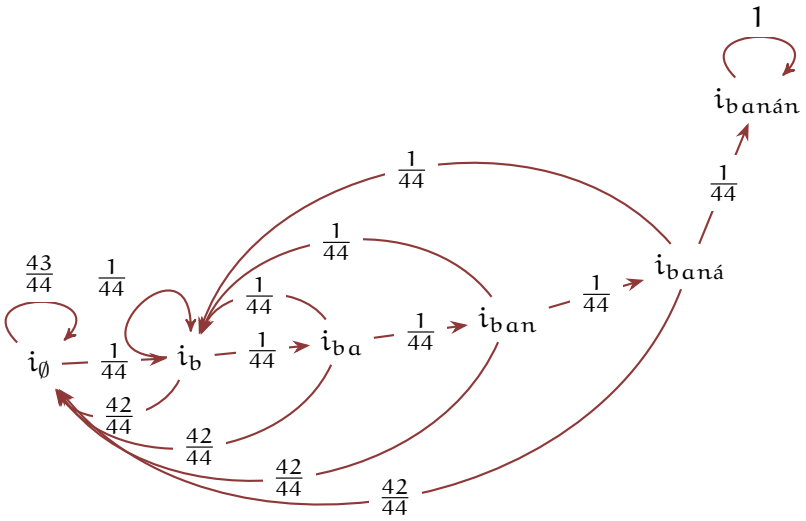
Erre a feladatra a következő Markov-láncot és jutalmakat írhatjuk fel:

- *Legyenek a folyamat állapotai* $i_\emptyset, i_b, i_{ba}, i_{ban}, i_{aná}, i_{banán}$
- *Az átmenetvalószínűségek az 1. ábrán láthatók.*
- *A jutalmak* $r_i = \begin{cases} 1, & \text{ha } i \neq i_{banán} \\ 0, & \text{egyébként} \end{cases}$

Jelölje v_i az i állapotból indulva a lépések várható számát, amíg elérjük a $i_{banán}$ állapotot.

Ekkor a teljes várhatóérték tétele alapján a következő egyenleteket írhatjuk fel a v_i várható lépésszámokra:

$$v_i = 1 + \sum_{j \neq i_{banán}} P_{ij} v_j; \qquad i \neq i_{banán} \tag{1.6}$$



1. ábra. Az 1.1 példa átmenetvalószínűségeinek gráfja.

A (1.6) lineáris egyenletrendszer alapján a várható leütések száma:

$$\nu_{i_\emptyset} = 164916224$$

Percenként 200 *leütéssel számolva* 1 *év* 6 *hónap és* 17 *nap átlagosan elegendő a minta megjelenésére.*

Az 1.1 példa igencsak mondvacsinált, "tantermi" jelenség. Az állapotok, amikhez nullától különböző jutalom tartozik mind tranziensek, így a stacionárius eloszlás időegység alatt várható jutalma is 0 (ha i állapot tranziens, akkor $\pi_i = 0$). Illetve a rekurrens osztály egyetlen nyelő csúcsból állt.

Lényegében ami ezen a példán megfigyelhető, csak egy átmeneti jelenség, ami a tranziens állapotokon játszódik le. Ahogy fejlődik a folyamat előbb-utóbb a nyelő állapotba érve már a folyamat, és az összegyűjtött jutalom sem változik.

1.2.1 *Véges sok lépés esete*

Általánosabb esetben egy unichaint fogunk vizsgálni, ahol már a rekurrens osztály állapotaihoz nem feltétlen nulla jutalom tartozik. Ekkor a várható összegyűjtött jutalmat egyrészt egy kezdeti tranziens viselkedés határozza meg, ami függ a kezdeti állapottól, majd idővel kihal, másrészt a stacionárius eloszlás időegység alatt várható jutalma, hiszen ilyenkor ez már lehet nullától különböző.

Ahhoz, hogy ide eljussunk először csak véges sok időbeni lépést vizsgálunk. Utána limeszeléssel fogjuk látni a megállítás nélküli folyamat esetét.

1.5. Definíció. *Egy kijelölt* $m \in \mathbb{N}$ *számhoz viszonyítva az* n. stádium *alatt a folyamat* X_{m-n} *állapotát értjük.*

4. Megjegyzés. Véges sok lépés vizsgálatakor magától értetődőbb lesz ezeket a stádiumokat használva, időben visszafelé számolni. Ez előkészíti a dinamikus programozási algoritmust is, amivel az optimális döntéseket számoljuk az MDP-k esetén.

1.6. Definíció. *Az* $\mathbf{u} \in \mathbb{R}^M$ megállási jutalom *jelölje a folyamat* 0. *stádiumában érkező jutalmat.*

5. Megjegyzés. Ezzel, hogy megengedjük, hogy az utolsó lépésben a korábbi $(r_1, r_2, \ldots, r_M)$ értékektől különböző jutalom érkezzen, amellett hogy általánosítjuk a vizsgált modellt, könnyebben tudjuk majd levezetni az általános esetben a várható összegyűjtött jutalmat.

1.7. Definíció. *Jelölje* $v_i(n, \mathbf{u})$ *az* i *állapotból, az* n. *stádiumban induló,* $\mathbf{u}$ *megállási jutalommal vett folyamat várható összegyűjtött jutalmát.*

Ekkor a $v_i(0, \mathbf{u}) := \mathbf{u}_i$ definícióval kezdve, tetszőleges stádiumra felírhatjuk a várható jutalmat:

$$v_i(n, \mathbf{u}) = r_i + \sum_{j=1}^{M} P_{ij} v_j(n-1, \mathbf{u}); \qquad (n \geqslant 1) \tag{1.7}$$

6. Megjegyzés. Bevezetve az

$$\mathbf{r} := (r_1, r_2, \ldots, r_M)^T$$
$$\mathbf{v}(n, \mathbf{u}) := (v_1(n, \mathbf{u}), v_2(n, \mathbf{u}), \ldots, v_M(n, \mathbf{u}))^T$$

jelöléseket a (1.7) egyenlet a következő formában foglalható össze:

$$\mathbf{v}(n, \mathbf{u}) = \mathbf{r} + P\mathbf{v}(n-1, \mathbf{u}); \qquad (n \geqslant 1) \tag{1.8}$$

Amennyiben a (1.8) egyenletet n-szer iteráljuk, adódik, hogy

$$\mathbf{v}(n, \mathbf{u}) = \mathbf{r} + P\mathbf{r} + P^2\mathbf{r} + \cdots + P^n\mathbf{u} \tag{1.9}$$

amiből a stacionárius eloszlás szerint átlagolva láthatjuk, hogy

$$\boldsymbol{\pi}\mathbf{v}(n, \mathbf{u}) = ng + \boldsymbol{\pi}\mathbf{u} \tag{1.10}$$

Hiszen $\boldsymbol{\pi}P^k = \boldsymbol{\pi}$ tetszőleges k-ra, illetve $\boldsymbol{\pi}\mathbf{r} = g$ definíció szerint.

7. Megjegyzés. A (1.10) összefüggés olyan szempontból nem meglepő, hogyha egy a stacionárius eloszlás szerint, az n. stádiumban induló folyamatot tekintünk, akkor az eloszlás nem változván, n-szer megkapjuk a stacionárius eloszlás időegység alatt várható jutalmát, illetve az utolsó lépésben a megállási jutalmat.

1.2.2 *Aszimptotikus relatív jutalom*

Amennyiben nem a stacionárius eloszlás szerint indul a folyamat, hanem egy rögzített állapotból, akkor természetes kérdés, hogy hogyan változik a várható összegyűjtött jutalom $\pi\mathbf{v}(n, \mathbf{u})$-hoz képest.

Egy i állapotra w_i-vel jelölve, hogy mennyivel jövedelmezőbb hosszú távon az i-ből indítani a folyamatot, mint π szerint, megsejthetjük hogy ergodikus Markov-láncra

$$\mathbf{v}(n, \mathbf{u}) \approx \mathbf{w} + ng\mathbf{e} + (\pi\mathbf{u})\mathbf{e} \tag{1.11}$$

hiszen $P^m \approx \mathbf{e}\pi$ nagy m-re, így a (1.9) összefüggés alapján, az utolsó tagra $P^n\mathbf{u} \approx (\pi\mathbf{u})\mathbf{e}$, illetve a közbülső tagokra $P^{n-m}\mathbf{r} \approx g\mathbf{e}$ amennyiben $n - m$ nagy. Az első néhány $\mathbf{r} + P\mathbf{r} + \cdots + P^m\mathbf{r}$ tagon a folyamat kezdeti viselkedése alatt érkező jutalmakat a $\mathbf{w} \in \mathbb{R}^M$ ún. *aszimptotikus relatív haszon* vektorral korrigálhatjuk.

(1.10) és (1.11) alapján

$$\mathbf{v}(n, \mathbf{u}) \approx \mathbf{w} + (\pi\mathbf{v}(n, \mathbf{u}))\mathbf{e} \tag{1.12}$$

A következőkben megmutatjuk, hogy a (1.12) sejtés helyes, és $n \to \infty$ esetén létezik a limesz, illetve pontossá válik a becslés.

Átrendezve (1.11)-et és limeszt véve, illetve (1.8) alapján:

$$\begin{aligned}
\mathbf{w} &= \lim_{n\to\infty} (\mathbf{v}(n, \mathbf{u}) - ng\mathbf{e} - (\pi\mathbf{u})\mathbf{e}) && (1.13)\\
&= \lim_{n\to\infty} (\mathbf{r} + P\mathbf{v}(n-1, \mathbf{u}) - ng\mathbf{e} - (\pi\mathbf{u})\mathbf{e}) && (1.14)\\
&= \mathbf{r} - g\mathbf{e} + P \lim_{n\to\infty} (\mathbf{v}(n-1, \mathbf{u}) - (n-1)g\mathbf{e} - (\pi\mathbf{u})\mathbf{e}) && (1.15)\\
&= \mathbf{r} - g\mathbf{e} + P\mathbf{w} && (1.16)
\end{aligned}$$

Tehát amennyiben létezik a (1.13) limesz, a határértéknek teljesítenie kell a $\mathbf{w} + g\mathbf{e} = \mathbf{r} + P\mathbf{w}$ egyenletet. Ez utóbbi egyenlet megoldhatóságát és $\mathbf{w}$ unicitását a következő lemma tisztázza. Az (1.13) limesz létezéséről a 1.3. tételben esik szó.

1.2. Lemma. *Legyen P egy M állapotú unichain átmenetmátrixa. Legyen $\mathbf{r} \in \mathbb{R}^M$ egy jutalomvektor, $\pi^T \in \mathbb{R}^M$ a lánc stacionárius eloszlása, illetve $g = \sum_i \pi_i r_i$. Ekkor a*

$$\mathbf{w} + g\mathbf{e} = \mathbf{r} + P\mathbf{w} \tag{1.17}$$

egyenletnek létezik $\mathbf{w}$ megoldása. A $\pi\mathbf{w} = 0$ feltétel mellett a megoldás egyértelmű.

Bizonyítás. Tekintsük (1.17)-t a következő alakban:

$$(P - I)\mathbf{w} = g\mathbf{e} - \mathbf{r} \tag{1.18}$$

Ahhoz, hogy tudjuk, létezik egy $\tilde{\mathbf{w}}$ megoldása (1.18)-nek, megmutatjuk, hogy $g\mathbf{e} - \mathbf{r}$ a $(P - I)$ mátrix oszlopterébe esik.

Egy mátrix oszloptere megegyezik a baloldali nullterének merőleges kiegészítőalterével. A $(P - I)$ mátrix baloldali nulltere ez esetben P-nek a $\lambda = 1$ sajátértékhez tartozó baloldali sajátvektorokból áll. Az 1.1. tétel alapján ez a sajátaltér $\boldsymbol{\pi}$ skalárszorosaiból áll. Így akkor és csak akkor létezik $\widetilde{\mathbf{w}}$, ha $\boldsymbol{\pi}(g\mathbf{e} - \mathbf{r}) = 0$.

Ez utóbbi mindig teljesül, hiszen $g(\boldsymbol{\pi}\mathbf{e}) = g \cdot 1$ és $\boldsymbol{\pi}\mathbf{r} = g$ definíció szerint.

Tegyük fel, hogy $\widetilde{\mathbf{w}}$ egy megoldása (1.18)-nak. Ekkor az összes megoldást megkaphatjuk $\mathbf{w} = \widetilde{\mathbf{w}} + \mathbf{x}$ alakban, ahol $(P - I)\mathbf{x} = \mathbf{0}$. Ez utóbbi feltétel azt jelenti, hogy $\mathbf{x}$ egy jobboldali sajátvektora P-nek, a $\lambda = 1$ sajátértékhez. Az 1.1 tétel alapján ekkor $\mathbf{x} = \alpha\mathbf{e}$ alakú, így $\mathbf{w} = \widetilde{\mathbf{w}} + \alpha\mathbf{e}$ alakban írható fel egy tetszőleges megoldás.

Amennyiben a $\boldsymbol{\pi}\mathbf{w} = 0$ feltételt is teljesíteni kell, adódik, hogy $\alpha = -\boldsymbol{\pi}\widetilde{\mathbf{w}}$, így a megoldás már egyértelmű. □

Tekintsük azt az esetet, amikor a megállási jutalomnak a (1.17) egyenlet egy $\mathbf{w}$ megoldását választjuk. Ekkor

$$\mathbf{v}(1, \mathbf{w}) = \mathbf{r} + P\mathbf{w} = \mathbf{w} + g\mathbf{e} \tag{1.19}$$

$$\mathbf{v}(2, \mathbf{w}) = \mathbf{r} + P(\mathbf{w} + g\mathbf{e}) = \mathbf{w} + 2g\mathbf{e} \tag{1.20}$$

$$\vdots \tag{1.21}$$

$$\mathbf{v}(n, \mathbf{w}) = \mathbf{r} + P(\mathbf{w} + (n-1)g\mathbf{e}) = \mathbf{w} + ng\mathbf{e} \tag{1.22}$$

Innen könnyen megmutatható tetszőleges $\mathbf{u} \in \mathbb{R}^M$ végső jutalomra hogyan alakul a várható összegyűjtött haszon. A (1.9) alapján

$$\mathbf{v}(n, \mathbf{u}) - \mathbf{v}(n, \mathbf{w}) = P^n(\mathbf{u} - \mathbf{w}) \tag{1.23}$$

amihez használva (1.22)-t

$$\mathbf{v}(n, \mathbf{u}) = \mathbf{w} + ng\mathbf{e} + P^n(\mathbf{u} - \mathbf{w}). \tag{1.24}$$

Összefoglalva ezeket az eredményeket a következő tételt mondhatjuk ki.

1.3. Tétel. *Legyen P egy unichain átmenetmátrixa, $\mathbf{r} \in \mathbb{R}^M$ egy jutalomvektor, $\mathbf{w} \in \mathbb{R}^M$ pedig a (1.17) egy megoldása.*

Ekkor az n lépés alatt összegyűjtött jutalmat a (1.24) alapján számolhatjuk.

Ha az unichain ergodikus és $\boldsymbol{\pi}\mathbf{w} = 0$ teljesül, akkor

$$\lim_{n\to\infty} (\mathbf{v}(n, \mathbf{u}) - ng\mathbf{e}) = \mathbf{w} + (\boldsymbol{\pi}\mathbf{u})\mathbf{e}. \tag{1.25}$$

Bizonyítás. A fenti érvelésen felül, amennyiben a Markov-lánc ergodikus unichain, akkor $\lim_n P^n = \mathbf{e}\boldsymbol{\pi}$, és így $\lim_n P^n\mathbf{u} = (\boldsymbol{\pi}\mathbf{u})\mathbf{e}$, illetve $\lim_n P^n\mathbf{w} = \mathbf{0}$, amiből adódik (1.25). □

2 MARKOV DÖNTÉSI FOLYAMATOK

2.1 BEVEZETÉS

A következőkben bevezetjük a modellbe a döntéshozás lehetőségét. Amíg korábban egy $i \in [M]$ állapothoz csak rögzített $\{P_{ij} : j \in [M]\}$ átmenetvalószínűségek, illetve az r_i jutalom volt rendelve, most lehetőséget adunk, hogy a folyamat minden lépésekor egy döntéshozó válasszon különböző jutalmak és hozzájuk tartozó átmenetvalószínűségek közt.

Egy i állapotban hozható döntések számát jelölje K_i. Ekkor a folyamat minden egyes fejlődési lépése előtt a döntéshozó az alábbi halmazból választ egy jutalmat és hozzá tartozó egylépéses átmenetvalószínűséget.

$$\left(r_i^{(k)}, \left\{P_{ij}^{(k)} : j \in [M]\right\}\right); \qquad k \in [K_i] \tag{2.1}$$

Miután megszületett az i állapotban a $k \in [K_i]$ döntés, a folyamat következő állapotba lépése $\left\{P_{ij}^{(k)} : j \in [M]\right\}$ eloszlás szerint történik, illetve az $r_i^{(k)}$ jutalom adatik.

8. Megjegyzés. A Markov tulajdonságnak megfelelő feltevésünk az, hogyha tudjuk, hogy $X_t = i$ és k a döntés, amit a t időpontban hoztak, akkor X_{t+1} eloszlása kizárólag $P_{ij}^{(k)}$-tól függ, a korábbi döntésektől és állapotoktól független.

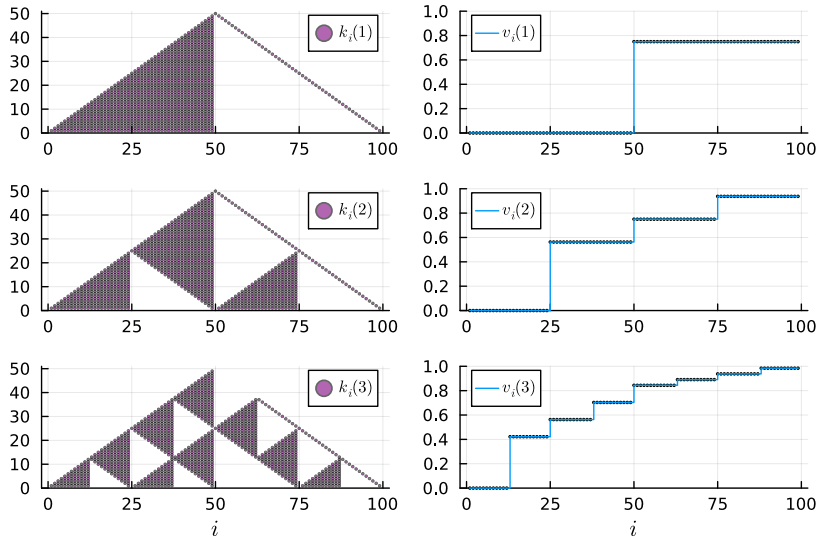
2.1. Definíció. *Abban az esetben, ha a döntéshozó minden alkalommal az i állapotban ugyanazt a k_i döntést hozza, az időtől függetlenül, akkor egy homogén Markov-láncot kapunk, aminek az átmenetmátrixát $P^{(\mathbf{k})}$-val jelöljük. (Ahol $\mathbf{k} := (k_1, \ldots, k_M)$ jelöli a döntéseket). Egy ilyen időtől független döntési eljárást* stacionárius stratégiának *nevezünk.*

A stacionárius stratégiaválasztás lehet optimális, amennyiben egy hosszú ideig futó folyamatot vizsgálunk. Abban az esetben viszont, amikor tudjuk, hogy a folyamat csak n lépést tesz, majd az utolsó lépéskor egy $\mathbf{u}$ megállási jutalom érkezik, az optimális döntések függeni fognak a hátralévő időtől.

A stacionárius stratégiaválasztás esetében a várható összegyűjtött jutalmat már az 1.2. fejezetben megvizsgáltuk. A következőkben az időtől függő, ún. *dinamikus stratégia* kiszámítására adunk algoritmust, illetve megvizsgáljuk ez esetben a várható összegyűjtött jutalmat.

2.2 OPTIMÁLIS DINAMIKUS STRATÉGIA

2.2. Definíció. *Jelölje $v_i^*(n, \mathbf{u})$ egy i állapotból, $\mathbf{u}$ megállási jutalommal az n lépés alatt maximálisan összegyűjthető várható jutalmat.*



2. ábra. Bal oldalt a 2.1 példa optimális dinamikus stratégiái, jobboldalt a maximális várható összegyűjtött jutalom, $n = 1, 2, 3$ stádiumokra, $p_{\text{fej}} = 0.75$. (1. programkód alapján.)

Ekkor a $v_i^*(0, \mathbf{u}) := \mathbf{u}$ definícióval kezdve rekurzívan számolhatjuk a maximális várható összegyűjtött jutalmat:

$$v_i^*(n, \mathbf{u}) = \max_k \{r_i^{(k)} + \sum_j P_{ij}^{(k)} v_j^*(n-1, \mathbf{u})\}; \qquad (n \geqslant 1) \qquad (2.2)$$

Bevezetve az $\mathbf{r^k} := (r_1^{(k_1)}, r_2^{(k_2)}, \ldots, r_M^{(k_M)})$ jelölést, egyszerre az M darab állapot független maximalizálási feladát jelölhetjük a következőképp.

$$\mathbf{v}^*(n, \mathbf{u}) = \max_{\mathbf{k}} \{\mathbf{r^k} + P^{(\mathbf{k})} \mathbf{v}^*(n-1, \mathbf{u})\}; \qquad (n \geqslant 1) \qquad (2.3)$$

Ez alapján azt mondjuk, hogy $\mathbf{k}$ egy optimális döntés az n. stádiumban, amennyiben (2.3) maximuma $\mathbf{k}$-n felvétetik. Az n lépéses optimális dinamikus stratégiák megkeresésére így magától értetődő dinamikus programozásai algoritmust kapunk $\mathcal{O}(n \cdot \sum_{i=1}^{M} K_i)$ időben.

9. Megjegyzés. Annak ellenére, hogy meglepően "egyszerű" algoritmust kaptunk az optimális dinamikus stratégiák meghatározására, sok esetben az állapottér és a döntési lehetőségek sokasága miatt a (2.3) egyenlet számítás igénye messze meghaladja, ami gyakorlatban kezelhető lenne.

2.1. Példa (Gambler's Problem)**.** *A szerencsejátékos esete egy fogadási probléma.* [1]

Egy szabálytalan pénzérme dobásra lehet tétet rakni, ami alapján ha fej akkor a játékos visszakapja a tét dupláját, amennyiben írás elveszti a tétet.

1 Ez a példa a 4.9-es gyakorlaton alapszik, *Richard S. Sutton and Andrew G. Barto - Reinforcement Learning: An Introduction* könyvéből.

A játék akkor ér véget, ha a játékos összegyűjt 100 *Forintot, ezesetben nyer, hogyha viszont* 0 *Forintra csökken a vagyona, akkor veszít és vége a játéknak.*

Legyen az n*. fogadás után a játékos vagyona* X_n *és* p *a valószínűsége, hogy az érme fej. Ekkor arra, hogy a fogadó játékos* n *körön belül nyer, a következő Markov döntési folyamat írható fel. Legyen* $\mathcal{S}$ *a folyamat állapottere,* $\mathcal{K}_i$ *a lehetséges döntések az* i *állapotban,* $\mathbf{r}, \mathbf{u}$ *a folyamat jutalmai, illetve az átmenetvalószínűségek a következők alapján:*

$$
\begin{aligned}
\mathcal{S} &= \{0, 1, \ldots, 100\} \\
\mathcal{K}_i &= \begin{cases} \{1, 2, \ldots, \min(i,\ 100 - i)\} & i \in \mathcal{S} \setminus \{0, 100\} \\ \{0\} & i \in \{0, 1\} \end{cases} \\
\mathbf{r} &= \mathbf{0} \\
\mathbf{u} &= (0, 0, \ldots, 0, 1)^T \\
P_{0j}^{(k)} &= 0 \qquad (\forall j \neq 0) \\
P_{00}^{(k)} &= 1 \\
P_{100,j}^{(k)} &= 0 \qquad (\forall j \neq 100) \\
P_{100,100}^{(k)} &= 1 \\
P_{ij}^{(k)} &= \begin{cases} p & \textit{ha } i \in \mathcal{S} \setminus \{0, 100\} \textit{ és } j = i + k \\ 1 - p & \textit{ha } i \in \mathcal{S} \setminus \{0, 100\} \textit{ és } (j = i - k; i - k \geqslant 0) \textit{ vagy } (j = 0; i - k < 0) \\ 0 & \textit{egyébként} \end{cases}
\end{aligned}
\tag{2.4}
$$

Az n *lépéses optimális dinamikus stratégiák a* 2*. ábrán láthatók.*

2.3 OPTIMÁLIS STACIONÁRIUS STRATÉGIA

A 2.2 fejezetben karakterizáltuk az optimális dinamikus stratégiákat, viszont az eredményekből nemigen látszik, hogy aszimptotikusan milyen döntések az optimálisak.

Ennek elemzéséhez megvizsgálunk először egy olyan esetet, amikor van egy optimális stacionárius stratégia, aminek a várható jutalma megegyezik az optimális dinamikus stratégia várható jutalmával.

2.1. Tétel. *Tegyük fel, hogy*

$$
\mathbf{w}' + g'\mathbf{e} = \max_{\mathbf{k}} \{\mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}\mathbf{w}'\} \tag{2.5}
$$

fennáll egy tetszőleges $(\mathbf{w}', g') \in \mathbb{R}^M \times \mathbb{R}$ *párosra. Ha a megállási jutalomnak* $\mathbf{w}'$*-t választjuk, akkor az a* $\mathbf{k}'$ *döntés, ami (2.5) jobboldalának maximumhelye, egy olyan stacionárius stratégia, ami egyben egy optimális dinamikus stratégia mindegyik stádiumra.*

A várható összegyűjtött jutalom pedig

$$\mathbf{v}^*(n, \mathbf{w}') = \mathbf{w}' + ng'\mathbf{e}. \tag{2.6}$$

Bizonyítás. Első stádiumban $\mathbf{k}'$ optimális dinamikus stratégia, hiszen

$$\max_{\mathbf{k}}\{\mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}\mathbf{w}'\} = \mathbf{r}^{\mathbf{k}'} + P^{(\mathbf{k}')}\mathbf{w}'. \tag{2.7}$$

Felhasználva a (2.5) egyenletet, $n = 1$ -re a várható összegyűjtött jutalom valóban (2.6) szerint kapható:

$$\mathbf{v}^*(1, \mathbf{w}') = \max_{\mathbf{k}}\{\mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}\mathbf{w}'\} = \mathbf{w}' + 1g'\mathbf{e} \tag{2.8}$$

Indukciót használva megmutatjuk, hogy $n > 1$ stádiumokra is $\mathbf{k}'$ egy optimális dinamikus stratégia, illetve hogy (2.6) is teljesül.

(2.3) alapján elindulva

$$\begin{aligned}
\mathbf{v}^*(n+1, \mathbf{w}') &= \max_{\mathbf{k}}\{\mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}\mathbf{v}^*(n, \mathbf{w}')\} && (2.9)\\
&= \max_{\mathbf{k}}\{\mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}(\mathbf{w}' + ng'\mathbf{e})\} && (2.10)\\
&= ng'\mathbf{e} + \max_{\mathbf{k}}\{\mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}\mathbf{w}'\} && (2.11)\\
&= (n+1)g'\mathbf{e} + \mathbf{w}'. && (2.12)
\end{aligned}$$

Hiszen (2.10) az indukciós hipotézis a (2.6) egyenletre, (2.11) adódik abból, hogy $P^{(\mathbf{k})}\mathbf{e} = \mathbf{e}$ tetszőleges $\mathbf{k}$-ra, és (2.12) pedig (2.5) alapján.

Mivel $\mathbf{k}'$ maximalizálja (2.11)-t, így (2.9) maximuma is felvétetik $\mathbf{k}'$-n. Ezzel igazoltuk azt is, hogy az $n+1$. stádiumban is optimális döntés a $\mathbf{k}'$ stacionárius stratégia.

□

2.3. Definíció. *Egy* $\mathbf{k}'$ stacionárius stratégia optimális, *ha létezik* $\mathbf{w}'$ *megállási jutalom, amire* $\mathbf{k}'$ *az optimális dinamikus stratégia.*

Természetes kérdés, hogy milyen esetekben létezik optimális stacionárius stratégia, illetve milyen eszközökkel tudjuk azt megtalálni.

A (2.5) egyenlet, amit *Bellman egyenletnek* szokás nevezni, kitüntetett szerepet játszik az optimális stacionárius stratégiák létezésében, hiszen a 2.1 tétel alapján láttuk, hogy ha létezik megoldása a (2.5) egyenletnek, akkor a jobb oldalt maximalizáló $\mathbf{k}'$ egy optimális stacionárius stratégia.

2.4 BELLMAN EGYENLET MEGOLDÁSA

A következőkben megvizsgáljuk, hogy milyen esetekben létezik megoldása a Bellman egyenletnek, illetve egy iteratív algoritmust adunk optimális stacionárius stratégia keresésére.

A 2.1 tételben a (2.5) egyenletet maximalizáló $\mathbf{k}'$ stratégiához tartozó Markov-lánc osztályairól nem tettünk semmi feltételt.

Amennyiben $P^{(\mathbf{k}')}$ egy unichain-t határoz meg, $\mathbf{w}'$ és g' lényegében egyértelműen meghatározott a 1.2 lemma értelmében[2], mint az aszimptotikus relatív jutalomvektor (vagy annak konstans eltolása), illetve a stacionárius eloszlás időegység alatti várható jutalma.

Tegyük fel, hogy $\mathbf{w}'$ és g' megoldása a Bellman egyenletnek. Ekkor tetszőleges $\mathbf{k}$ döntésre

$$\mathbf{w}' + g'\mathbf{e} \geqslant \mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}\mathbf{w}'. \tag{2.13}$$

Ha a $P^{(\mathbf{k})}$ átmenetmátrixú Markov-láncnak több rekurrens osztálya is van, akkor az $\mathcal{R}$ rekurrens osztályhoz tartozó stacionárius eloszlást $\boldsymbol{\pi}^{\mathbf{k},\mathcal{R}}$-vel jelölve (2.13)-ból adódik, hogy

$$\boldsymbol{\pi}^{\mathbf{k},\mathcal{R}}\mathbf{w}' + \boldsymbol{\pi}^{\mathbf{k},\mathcal{R}}g'\mathbf{e} \geqslant \boldsymbol{\pi}^{\mathbf{k},\mathcal{R}}\mathbf{r}^{\mathbf{k}} + \boldsymbol{\pi}^{\mathbf{k},\mathcal{R}}P^{(\mathbf{k})}\mathbf{w}' \tag{2.14}$$

$$g' \geqslant \boldsymbol{\pi}^{\mathbf{k},\mathcal{R}}\mathbf{r}^{\mathbf{k}}. \tag{2.15}$$

Ahol a (2.13), (2.14) és (2.15) egyenlőtlenségek a $\mathbf{k}'$ maximalizáló stratégia esetén egyenlőséggel teljesülnek.

Megkaptuk tehát, hogy Bellman egyenlet egy $\mathbf{w}'$, g' megoldására igaz, hogy g' értéke legalább akkora, mint tetszőleges $\mathbf{k}$ stacionárius stratégiának bármely rekurrens osztályához tartozó időegység alatti várható jutalom.

Speciálisan a Bellman egyenletet maximalizáló $\mathbf{k}'$ statégiára pedig ez alapján a $\mathbf{k}'$ optimális döntés vagy egy unichaint határoz meg, vagy egy olyan Markov-láncot, aminek az összes rekurrens osztályban egyenlő az időegység alatti várható jutalom, és ez megegyezik g' értékével.

2.4.1 *Egy MDP állapotainak osztályozása*

Mielőtt rátérünk a Bellman egyenlet algoritmikus megoldására, megvizsgáljuk a Markov-láncok állapotainak osztályozásához hasonlóan a Markov döntési folyamatok állapotainak általános viszonyát.

Egy adott Markov döntési folyamathoz tekintsük az állapottéren egy irányított gráfot, aminek egy (i,j) állapotpár közt akkor fut éle, ha $\exists k_i \in [K_i]$ döntés, amire $P_{ij}^{(k_i)} > 0$.

2.4. Definíció. *Egy Markov döntési folyamatnak a* j *állapota* elérhető *az* i *állapotból, ha a fenti gráfon létezik* $i \to j$ *út.*

10. Megjegyzés. Így amennyiben j elérhető i-ből, akkor létezik egy stacionárius stratégia, aminek a homogén Markov-láncában $i \to j$ a klasszikus értelemben.

2 Hiszen vegyük észre, hogy kihagyható a lemma feltételeiből, hogy $g = \boldsymbol{\pi}\mathbf{r}$. Ha g is egy változónak van tekintve, akkor (1.17) tetszőleges $(\mathbf{w}, g)$ megoldását $\boldsymbol{\pi}$ stacionárius eloszlással balról szorozva adódik, hogy $g = \boldsymbol{\pi}\mathbf{r}$ szükségszerűen teljesül.

2.5. Definíció. *Egy Markov döntési folyamat* i *állapota*

- természeténél fogva tranziens, *amennyiben létezik elérhető* j *állapot, amiből* i *nem elérhető.*
- természeténél fogva rekurrens, *amennyiben nem természeténél fogva tranziens.*

2.6. Definíció. *Az állapotoknak egy* $\mathcal{I}$ *osztálya* természeténél fogva rekurrens osztály, *amennyiben* $\forall i \in \mathcal{I}$ *állapot természeténél fogva rekurrens, és* $\forall i, j \in \mathcal{I}$ *állapotpárra* i *és* j *kölcsönösen elérhető egymásból, illetve nincs* $j \notin \mathcal{I}$, *ami elérhető lenne bármely* $i \in \mathcal{I}$ *-ből.*

2.7. Definíció. *Egy MDP* természeténél fogva rekurrens, *amennyiben az összes állapota egy természeténél fogva rekurrens osztályhoz tartozik.*

11. Megjegyzés. Ha egyszer egy természeténél fogva rekurrens osztályba ér a folyamat, akkor nincs olyan döntés, ami az osztályból kivezetné.

Egy i természeténél fogva tranziens állapot legalább egy stacionárius stratégiára tranziens. Esetleg rekurrens lehet más döntésekre, viszont egy ilyen döntésre az i rekurrens osztályához tartozó állapotok mind természetüknél fogva tranziensek.

A következőkben az egyszerűség kedvéért természeténél fogva rekurrens MDP-ket fogunk vizsgálni.

2.4.2 *Stratégia iteráció*

A *Stratégia iteráció* egy olyan eljárás, amivel egy tetszőleges $\mathbf{k}'$ unichain stratégiából kindulva megkeressük a hozzá tartozó $\mathbf{w}'$ aszimptotikus relatív jutalomvektort és g' stacionárius eloszlás szerinti időegység alatt várható jutalmat, majd ellenőrizzük, hogy a (2.5) Bellman egyenlet teljesül-e $\mathbf{w}'$, g'-re.

Ha teljesül, akkor $\mathbf{k}'$ egy optimális stacionárius stratégia, egyébként pedig adunk egy új $\mathbf{k}$ stratégiát ami bizonyos szempontból "jobb"[3] mint $\mathbf{k}'$.

Az új stratégia nem garantált, hogy unichaint határozzon meg, így a következő lemmában megmutatjuk, hogy ezesetben tudunk mutatni egy ugyanolyan "jó" unichain stratégiát.

Ez alapján a stratégia iterációval egy véges algoritmust tudunk adni, ami egyre "jobb" unichain stratégiákat talál, míg végül találunk egyet, amire teljesül a Bellman egyenlet, ezzel garantálva az optimalitást.

2.2. Lemma. *Legyen* $\mathbf{k} = (k_1, k_2, \ldots, k_M)$ *egy tetszőleges stratégia, egy természeténél fogva rekurrens MDP-ben.*

Legyen $\mathcal{R}$ *egy rekurrens osztálya a* $P^{(\mathbf{k})}$ *által meghatározott Markov-láncnak.*

3 A 2.4 lemma fogja tisztázni, hogy mit értünk "jobb" stratégia alatt.

Ekkor tudunk mutatni egy $\tilde{\mathbf{k}} = (\tilde{k}_1, \tilde{k}_2, \dots, \tilde{k}_M)$ *unichain stratégiát, aminek a rekurrens osztálya* $\mathcal{R}$*, illetve* $\tilde{k}_i = k_i \forall i \in \mathcal{R}$*.*

Bizonyítás. Rögzítsünk egy $j \in \mathcal{R}$ állapotot. Mivel a vizsgált MDP egy természeténél fogva rekurrens osztályból áll, így megmutatható, hogy létezik egy $\mathbf{k}'$ stratégia, ami alapján minden állapotból elérhető j. Ekkor $\tilde{k}_i = k_i \forall i \in \mathcal{R}$ és $\tilde{k}_i = k'_i \forall i \notin \mathcal{R}$ választással megfelelő $\tilde{\mathbf{k}}$ stratégiát kapunk.

□

1. Algoritmus Stratégia iteráció

1: Válasszunk egy tetszőleges $\mathbf{k}'$ unichain stratégiát.
2: A $\mathbf{k}'$ stratégiához számítsuk ki $\mathbf{w}'$-t és g'-t a

$$\mathbf{w}' + g'\mathbf{e} = \mathbf{r}^{\mathbf{k}'} + P^{(\mathbf{k}')}\mathbf{w}'$$

egyenletből.
3: Ha $\mathbf{w}' + g'\mathbf{e} = \max_{\mathbf{k}}\{\mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}\mathbf{w}'\}$, akkor megállunk, $\mathbf{k}'$ optimális.
4: Egyébként válasszunk $i \in [M]$ állapotot és $k_i \in [K_i]$ döntést, amire

$$w'_i + g' < r_i^{k_i} + \sum_j P_{ij}^{(k_i)} w'_j.$$

Legyen $j \neq i$-re $k_j := k'_j$.
5: Ha $\mathbf{k} = (k_1, k_2, \dots, k_M)$ nem unichaint határoz meg, akkor legyen $\mathcal{R}$ az a rekurrens osztály, ami tartalmazza az i állapotot, és legyen $\tilde{\mathbf{k}}$ az unichain stratégia, amit a 2.2 lemma alapján kapunk.
Legyen $\mathbf{k} := \tilde{\mathbf{k}}$.
6: Legyen $\mathbf{k}' := \mathbf{k}$ és folytassuk a 2. sornál.

Amennyiben a 3. lépésnél nem állunk meg, az maga után vonja hogy legyen egy i állapot, aminek megfelelő koordinátában sérül a Bellman egyenlet, tehát a 4. lépés mindig végrehajtható és az így kapott $\mathbf{k}$ stratégia ekkor teljesíti a

$$\mathbf{w}' + g'\mathbf{e} \lneqq \mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}\mathbf{w}' \tag{2.16}$$

egyenlőtlenséget, ahol a $\lneqq$ jelölést arra használjuk, hogy mindegyik koordinátára megköveteljük a nagyobb egyenlő relációt, és legalább egy kordinátában pedig a szigorúan nagyobb relációt.

2.3. Lemma. *Egy természeténél fogva rekurrens MDP-re a Stratégia iteráció algoritmusában csak a következő három eset valamelyike állhat fennt a 4. lépés után kapott* $\mathbf{k}$ *stratégiára:*

1. $\mathbf{k}$ *unichain Markov-láncot határoz meg, és az* i *állapot rekurrens* $\mathbf{k}'$ *és* $\mathbf{k}$ *stratégiák szerint is.*

2. $\mathbf{k}$ *nem unichain Markov-láncot határoz meg, és* i *tranziens* $\mathbf{k}'$ *szerint, rekurrens* $\mathbf{k}$ *szerint.*

3. $\mathbf{k}$ *unichain Markov-láncot határoz meg, aminek azonos a rekurrens osztálya* $\mathbf{k}'$ *rekurrens osztályával, és* i *tranziens* $\mathbf{k}'$ *és* $\mathbf{k}$ *stratégiák szerint is.*

Bizonyítás. Vegyük észre, hogy $P^{(\mathbf{k})}$ és $P^{(\mathbf{k}')}$ átmenetvalószínűségei csak az i állapotból kilépő éleken különböznek. Ezáltal azok az állapotok, amikből elérhető i azonos marad a változtatás után is.

Amennyiben i rekurrens volt a $\mathbf{k}'$ (unichain) szerint, úgy minden állapotból elérhető kellett legyen, tehát $\mathbf{k}$-ban is rekurrens állapot lesz. A tény, hogy ekkor i minden csúcsból elérhető $\mathbf{k}$ szerint, maga után vonja, hogy $\mathbf{k}$ unichain.

Hogyha i egy tranziens állapot a $\mathbf{k}'$ szerint, akkor $\mathcal{R}'$-vel jelölve $\mathbf{k}'$ rekurrens osztályát, adódik hogy $\mathcal{R}'$ egy rekurrens osztály lesz $\mathbf{k}$ szerint is, hiszen ezen osztályon belül nem változtak az átmenetvalószínűségek.

Ekkor ha $\mathbf{k}$-ban az i-ből kilépő átmenetvalószínűségek változtatása miatt már nem lesz elérhető az $\mathcal{R}'$ osztály i-ből, egy új $\mathcal{R}$ rekurrens osztálya lesz $\mathbf{k}$-nak, amire $i \in \mathcal{R}$. (Ebben az esetben használjuk a 2.2 lemmát, hogy $\mathbf{k}$-ból ismét unichain legyen.)

Ha $\mathbf{k}$-ban továbbra is elérhető i-ből $\mathcal{R}'$, akkor i tranziens állapot marad, és $\mathbf{k}$ pedig unichain lesz ugyan azzal a $\mathcal{R}'$ rekurrens osztállyal, mint ami $\mathbf{k}'$ rekurrens osztálya volt. □

2.4. Lemma. *Legyen* $\mathbf{k}'$ *a Stratégia iteráció algoritmus 2. lépésének stratégiája egy természeténél fogva rekurrens MDP-ben. Jelölje* $\mathbf{w}', g', \mathcal{R}'$ *rendre a* $\mathbf{k}'$ *unichain stratégiához tartozó relatív haszonvektort, időegység alatti jutalmat, és a lánc rekurrens osztályát. Amennyiben az algoritmus nem áll meg a 3. lépésnél, legyen* $\mathbf{k}$ *a 6. lépésben kapott unichain stratégia.*

Ekkor a $\mathbf{k}$*-hoz tartozó jutalmakra a következő két eset egyike állhat fennt.*

- *A* $\mathbf{k}$*-hoz tartozó* g *időegység alatti várható jutalomra* $g > g'$*, vagy*
- *a* $\mathbf{k}$ *stratégia rekurrens osztálya szintén* $\mathcal{R}'$*, és az időegység alatti várható jutalomra* $g = g'$ *teljesül, továbbá létezik* $\mathbf{k}$*-hoz egy* $\mathbf{w}$ *aszimptotikus relatív jutalomvektor, amire*

$$\mathbf{w}' \lneqq \mathbf{w} \qquad \text{és } w'_j = w_j \ \forall j \in \mathcal{R}'. \tag{2.17}$$

Bizonyítás. A 4. lépés $\mathbf{k}$ stratégiájának tetszőleges $\mathcal{R}$ rekurrens osztályához tartozó $\boldsymbol{\pi}$ stacionárius eloszlásra (2.16) alapján

$$\boldsymbol{\pi}\mathbf{w}' + g' \leqslant \boldsymbol{\pi}\mathbf{r}^{\mathbf{k}} + \boldsymbol{\pi}P^{(\mathbf{k})}\mathbf{w}'. \tag{2.18}$$

Ami egyszerűsítés után, figyelembe véve, hogy (2.16) egyenletben az i-hez tartozó koordinátában szigorú egyenlőtlenség van, a következő alakban írható

$$g' = \boldsymbol{\pi}\mathbf{r}^{\mathbf{k}} \quad \text{ha } \pi_i = 0 \text{ és} \tag{2.19}$$

$$g' < \boldsymbol{\pi}\mathbf{r}^{\mathbf{k}} \quad \text{ha } \pi_i > 0. \tag{2.20}$$

A 2.3 lemma alapján első esetben, mikor $\mathbf{k}$ unichain és i rekurrens marad, adódik hogy $g' < \boldsymbol{\pi}\mathbf{r}^{\mathbf{k}} = g$. Másodszor, amikor i körül új $\mathcal{R}$ rekurrens osztály formálódik $\mathbf{k}$-ban, szintén $g' < \boldsymbol{\pi}\mathbf{r}^{\mathbf{k}}$ és mivel a 2.2 lemma alapján kapott $\tilde{\mathbf{k}}$ rekurrens osztálya szintén $\mathcal{R}$, így $g' < g$ esetet kapjuk itt is. A harmadik eset az egyetlen amikor i tranziens lesz az új stratégiában, így $\pi_i = 0$. Ilyenkor a rekurrens osztály nem változik tehát $\mathcal{R} = \mathcal{R}', \boldsymbol{\pi} = \boldsymbol{\pi}'$ és így $g = g'$. Ebben az utolsó esetben tehát még igazolnunk kell, hogy (2.17) fennáll.

A folytatáshoz először indukcióval megmutatjuk, hogy a $\mathbf{k}$ stacionárius stratégia használata esetén

$$\mathbf{v}^{\mathbf{k}}(n, \mathbf{w}') - ng'\mathbf{e} \leqslant \mathbf{v}^{\mathbf{k}}(n+1, \mathbf{w}') - (n+1)g'\mathbf{e} \qquad (n \geqslant 1) \tag{2.21}$$

azaz a $\mathbf{v}^{\mathbf{k}}(n, \mathbf{w}') - ng'\mathbf{e}$ vektorsorozat monoton növő.

$n = 1$ esetben használva a (2.16) egyenletet

$$\mathbf{w}' \lneqq \mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}\mathbf{w}' - g'\mathbf{e} = \mathbf{v}^{\mathbf{k}}(1, \mathbf{w}') - g'\mathbf{e}, \tag{2.22}$$

így

$$\begin{aligned} \mathbf{v}^{\mathbf{k}}(1, \mathbf{w}') - g'\mathbf{e} &= \mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}\mathbf{w}' - g'\mathbf{e} \\ &\leqslant \mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}(\mathbf{v}^{\mathbf{k}}(1, \mathbf{w}') - g'\mathbf{e}) - g'\mathbf{e} \\ &= \mathbf{v}^{\mathbf{k}}(2, \mathbf{w}') - 2g'\mathbf{e}. \end{aligned} \tag{2.23}$$

Feltéve, hogy $(n-1)$-re (2.21) fennáll,

$$\begin{aligned} \mathbf{v}^{\mathbf{k}}(n, \mathbf{w}') - ng'\mathbf{e} &= \mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}\mathbf{v}^{\mathbf{k}}(n-1, \mathbf{w}') - ng'\mathbf{e} \\ &= \mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}(\mathbf{v}^{\mathbf{k}}(n-1, \mathbf{w}') - (n-1)g'\mathbf{e}) - g'\mathbf{e} \\ &\leqslant \mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}(\mathbf{v}^{\mathbf{k}}(n, \mathbf{w}') - ng'\mathbf{e}) - g'\mathbf{e} \\ &= \mathbf{v}^{\mathbf{k}}(n+1, \mathbf{w}') - (n+1)g'\mathbf{e}. \end{aligned} \tag{2.24}$$

Mivel $\mathbf{k}$ unichain, a 1.2 lemma (1.17) egyenletének létezik $\mathbf{w}$ megoldása, amire (1.24) alapján

$$\mathbf{v}^{\mathbf{k}}(n, \mathbf{w}') = \mathbf{w} + ng'\mathbf{e} + (P^{(\mathbf{k})})^n(\mathbf{w}' - \mathbf{w}). \tag{2.25}$$

Sztochasztikus mátrix lévén $(P^{(\mathbf{k})})^n$ elemei 0 és 1 közé esnek, így tetszőleges n-re $\mathbf{v}^{\mathbf{k}}(n, \mathbf{w}') - ng'\mathbf{e}$ korlátos. Azt már láttuk, hogy monoton, ezek alapján tehát kell létezzen limesze:

$$\tilde{\mathbf{w}} := \lim_{n\to\infty} \left(\mathbf{v}^{\mathbf{k}}(n, \mathbf{w}') - ng'\mathbf{e}\right) \tag{2.26}$$

Megmutatjuk, hogy $\tilde{\mathbf{w}}$ valóban egy aszimptotikus relatív jutalomvektor (teljesíti (1.17)-t):

$$\begin{aligned} \tilde{\mathbf{w}} &= \lim_{n\to\infty} \left(\mathbf{v}^{\mathbf{k}}(n+1, \mathbf{w}') - (n+1)g'\mathbf{e}\right) \\ &= \lim_{n\to\infty} \left(\mathbf{r}^{\mathbf{k}} + P^{(\mathbf{k})}\mathbf{v}^{\mathbf{k}}(n, \mathbf{w}') - (n+1)g'\mathbf{e}\right) \\ &= \mathbf{r}^{\mathbf{k}} - g'\mathbf{e} + P^{(\mathbf{k})} \lim_{n\to\infty} \left(\mathbf{v}^{\mathbf{k}}(n, \mathbf{w}') - ng'\mathbf{e}\right) \\ &= \mathbf{r}^{\mathbf{k}} - g'\mathbf{e} + P^{(\mathbf{k})}\tilde{\mathbf{w}} \end{aligned} \tag{2.27}$$

Használva, a (2.22) egyenletet és azt, hogy $\tilde{\mathbf{w}}$-hez alulról monotonan konvergál a $\mathbf{v}^{\mathbf{k}} - n g \mathbf{e}$ sorozat, adódik hogy

$$\mathbf{w}' \lneqq \mathbf{v}^{\mathbf{k}}(n, \mathbf{w}') - n g \mathbf{e} \leqslant \tilde{\mathbf{w}}. \tag{2.28}$$

Amit a $\mathbf{k}$ stratégia $\boldsymbol{\pi}$ stacionárius eloszlásával szorozva,

$$\boldsymbol{\pi}\mathbf{w}' \leqslant \boldsymbol{\pi}\mathbf{v}^{\mathbf{k}}(n, \mathbf{w}') - n g' \leqslant \boldsymbol{\pi}\tilde{\mathbf{w}} \qquad (\forall n \geqslant 1). \tag{2.29}$$

Vegyük észre, hogy mivel a rekurrens osztályban $\mathbf{k}$ és $\mathbf{k}'$ megegyezik, így (1.10) alapján (2.29) első egyenlőtlensége igazából egyenlőség, és $n \to \infty$ limeszt véve megkapjuk, hogy a második reláció is egyenlőség kell legyen.

Továbbá $0 \leqslant \boldsymbol{\pi}$ és $\mathbf{w}' \leqslant \tilde{\mathbf{w}}$ miatt a rekurrens osztályban minden j állapotra $w'_j = \tilde{w}_j$ kell teljesüljön. □

12. Megjegyzés. Mivel $\forall i$-re K_i véges, így véges sok stratégiavektor közül keresünk optimálisat. Azzal, hogy megmutattuk a Stratégia iteráció algoritmus minden ciklusában az aktuális $\mathbf{k}'$ unichain stratégiára vagy a stacionárius eloszlás időegység alatti várható jutalma nő, vagy azt nem változtatva a 2.4. lemma alapján választott aszimptotikus relatív haszon nő szigorúan, következik hogy véges sok lépés után megállunk a 3. lépésnél.

Ezek alapján tehát kimondhatjuk a következő tételt.

2.5. Tétel. *Egy természeténél fogra rekurrens MDP-re létezik megoldása a Bellman egyenletnek, és az azt maximalizáló stratégiák közt van ami unichain.*

2.2. Példa (Készlet feltöltés [4])**.** *Tegyük fel, hogy van egy olyan kis üzlet, ami csak egyféle terméket árul. A vásárló vendégek száma egy-egy nap* $\mathtt{Poisson}(3)$ *eloszlású. Minden nap végén van leltározás és el kell dönteni, hogy rendelünk-e és ha igen hányat az árult termékből. A raktár kapacitása* 7*. Az alapdíja az áru rendelésének* 10*, a nap végén minden raktárban maradt termék után* 1 *költséget számolunk fel. Ha egy nap több vásárló érkezik, mint amennyi áru készleten van, akkor legfeljebb* 3 *érdeklődőt készleten felül is, utánrendeléssel kiszolgálunk, de ennek a díja* 15 *vásárlónként.*

Erre a feladatra az $\mathcal{S} = (-3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7)$ *állapottérrel jellemezve a raktárkészlet alakulását, az* $s \in \mathcal{S}$*-hez a* $\{0, 1, \ldots, 7 - s\}$ *döntéseket társítva, illetve a 3. táblázat költségei alapján egy természeténél fogva rekurrens MDP-t kapunk.*

Az optimális rendelési stratégiát megkaphatjuk a Stratégia iteráció algoritmus használatával. Az eredmények az 1. táblázatban láthatók.

A valóságban könnyen találkozhatunk olyan problémával, aminek modellezésekor nem csak egy természeténél fogva rekurrens osztályt

4 Ez a példa feladat https://user.engineering.uiowa.edu/~dbricker/slides_handouts/MDP_Example_Inventory.pdf alapján készült.

$\mathcal{S}$	$\mathbf{k}$	π	$\mathbf{r}$	$\mathbf{w}$
-3	10	1.32%	-45	-34.24
-2	9	2.9%	-30	-19.24
-1	8	5.51%	-15	-4.24
0	7	8.99%	-10	0.76
1	6	12.49%	-11	-0.24
2	5	14.94%	-12	-1.24
3	0	15.85%	-3	1.17
4	0	15.27%	-4	2.65
5	0	12.66%	-5	3.38
6	0	7.64%	-6	3.72
7	0	2.42%	-7	3.76

1. táblázat. A 2.2. példa (raktározási feladat) optimális megoldása, a 4. programkód alapján. Az átlagos napi költség $g = -8.70$.

alkot az MDP állapottere, hanem vannak természeténél fogva tranziens állapotok is.

Az legkézenfekvőbb példa ilyenre a *sztochasztikus legrövidebb út* problémája, amikor egyetlen t állapota van a folyamatnak, ami tetszőleges k döntésre egy nyelő állapot és hozzá $r_t^k = 0$ jutalom tartozik. A többi $s \neq t$ állapotra pedig $r_s^k \leqslant 0$ tetszőleges döntésre, illetve $\exists n, \mathbf{k} : (P_{st}^{(\mathbf{k})})^n > 0$.

Ilyenkor a természeténél fogva rekurrens osztályt egyedül a t alkotja, a többi állapot természeténél fogva tranziens. A stacionárius eloszlásra $\pi_t = 1$ és $\pi_s = 0$ $(s \neq t)$, így az időegység alatti várható jutalom $g = 0$. A nem pozitív jutalmak miatt az optimális döntések, amik $\mathbf{w}$ aszimptotikus relatív jutalmat maximalizálják értelmezhetők úgy, mint az $s \to t$ eljutás várható költségének minimalizálása.

A következő lemmában megmutatjuk, hogy sztochasztikus legrövidebb út tipusú problémáknál, illetve annál általánosabb esetben is megfelelő kiindulási stratégia mellett a Stratégia iteráció algoritmus alkalmas lehet optimális stacionárius stratégia keresésre.

2.6. Lemma. *Tekintsünk egy olyan MDP-t, amely állapotainak egy természeténél fogva rekurrens osztálya van és mellette van egy vagy több természeténél fogva tranziens állapota. Legyen g^* a maximuma tetszőleges stacionárius döntés tetszőleges rekurrens osztályán vett időegység alatti várható jutalomnak és tegyük fel, hogy az olyan rekurrens osztályok, amikben az időegység alatti várható jutalom megegyezik g^*-al, mind benne vannak a természeténél fogva rekurrens osztályban.*

Ekkor a Bellman egyenletnek létezik megoldása, és a maximalizáló stratégiák közt létezik ami unichain.

Bizonyítás. Legyen $\mathbf{k}$ egy olyan stacionárius stratégia, aminek van egy $\mathcal{R}$ rekurrens osztálya g^* időegység alatti jutalommal. Legyen j egy tetszőleges $\mathcal{R}$-beli állapot. Ekkor a feltevések alapján j természeténél fogva rekurrens állapot kell legyen, így létezik egy $\tilde{\mathbf{k}}$ stratégia, ami alapján j elérhető az összes állapotból. Tekintsük azt a $\mathbf{k}'$ stratégiát, amire

$$k'_i = \begin{cases} k_i & i \in \mathcal{R} \\ \tilde{k}_i & i \notin \mathcal{R}. \end{cases} \tag{2.30}$$

Ekkor a $\mathbf{k}'$ stratégia egy olyan unichain Markov-láncot határoz meg, aminek rekurrens osztálya $\mathcal{R}$ és az időegység alatti jutalma pedig így g^*.

Tegyük fel, hogy a Stratégia iteráció algoritmusát a $\mathbf{k}'$ stratégiával indítottuk. Ekkor ha 3. lépésben megállunk, akkor kaptunk egy unichain stratégiát, ami maximalizálja a Bellman egyenletet. Ha nem állt meg az algoritmus, akkor a 6. lépés utáni új stratégia rekurrens osztálya szintén $\mathcal{R}$ kell legyen, illetve az aszimptotikus relatív jutalomban kellett javuljon a 2.4. lemma alapján, hiszen g^*-nál nagyobb időegység alatti várható jutalmat nem kaphattunk.

Így a Stratégia iteráció lépései alatt az új $\mathbf{w}$ mindig kielégíti (2.17)-t, tehát a stratégiák nem ismétlődhetnek. Ez alapján véges sok lépésben kell találjunk egy optimális stacionárius unichain stratégiát, amire fennáll a Bellman egyenlet.

□

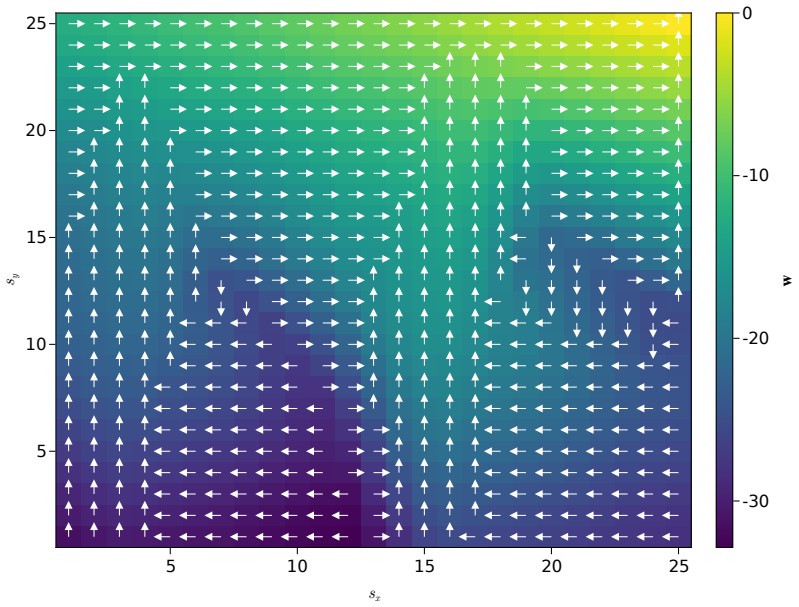
2.3. Példa (Sztochasztikus legrövidebb út)**.** *A feladat egy vitorlás hajó optimális navigálása a szeles tengeren. A probléma diszkretizált változatát oldjuk meg, a sík helyett egy W széles, H magas négyzetrács pontjain vizsgáljuk a folyamatot. A cél a jobb felső $t = (W, H)$ sarokba való eljutás. Egy $s \in [W] \times [H] =: \mathcal{S}$ állapotban azt dönthetjük el, hogy melyik égtáj felé próbáljuk irányítani a hajót. Ez csak 0.75 valószínűséggel valósul meg, 0.1, és 0.1 eséllyel a kiválasztott irányra merőleges égtáj felé sodródunk és 0.05 valószínűséggel pedig éppen az ellenkező irányba.*

Adott egy $F : \mathbb{R}^2 \to \mathbb{R}^2$ vektormező a síkon (3. (b) ábra), ami a szél irányát jellemzi. Egy döntés után a "jutalom" (itt igazából az eltelt idő, amíg megtesszük a kiválasztott utat) arányos azzal, hogy a széliránnyal milyen szöget zár a mozgásunk. Egy $s = (i, j) \in \mathcal{S}$ állapotra, és $k \in \{(0, \pm 1), (\pm 1, 0)\}$ döntésre a költség

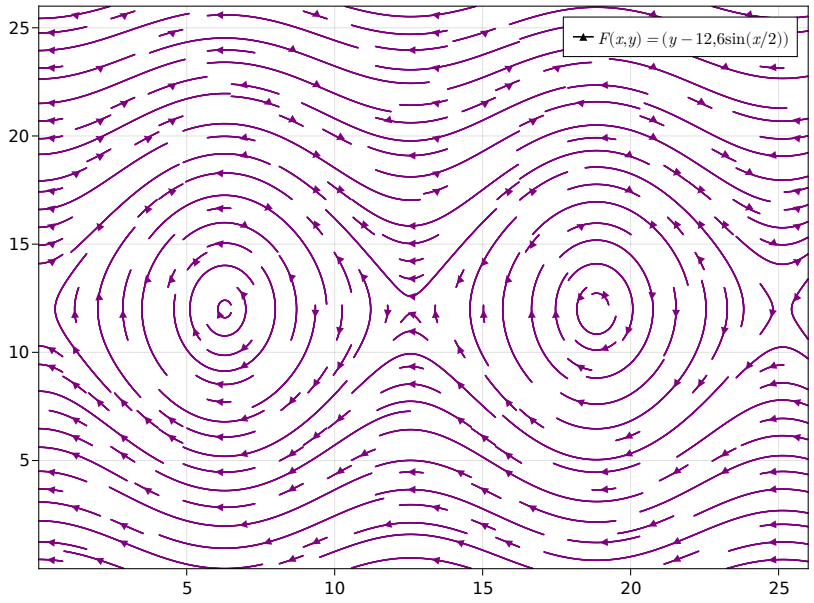
$$r_s^k = \underbrace{-1.25}_{\text{Egységnyi út megtételéhez szükséges idő}} + \underbrace{< k, \frac{F(i,j)}{\|F(i,j)\|} >}_{\text{A szélirány mennyit változtat}}. \tag{2.31}$$

A 2.6. lemma alapján használható a Stratégia iteráció algoritmus erre a problémára, amennyiben a kezdeti stratégiában minden csúcsból elérhető a jobb felső nyelő állapot.

Az optimális döntések és a hozzájuk tartozó várható jutalom (az út várható hossza) az 3. (a) ábrán láthatók.



(a) Egy $(s_x, s_y) \in \mathcal{S}$ pontban a fehér nyíl az optimális stacionárius stratégia döntését jelöli. A cellák színe **w**-alapján a várható utazási idő.



(b) A szélirányt jellemző $F : \mathbb{R}^2 \to \mathbb{R}^2$ vektormező.

3. ábra. A 2.3 példa feladat eredményei egy $W = H = 25$ méretű négyzetrácson. A 3. programkód alapján.

2.5 STACIONÁRIUS STRATÉGIÁK TETSZŐLEGES MEGÁLLÁSI JUTALOMMAL

A 2.3. definíció alapján úgy határoztuk meg az optimális stacionárius stratégiákat, hogy olyan stratégiák, amelyekhez *létezik* egy megállási jutalom, amire megegyeznek az optimális dinamikus stratégiával.

A következőkben elégséges feltételeket adunk, amik mellett egy optimális stacionárius stratégia tetszőlegesen választott megállási jutalom mellett a megállás előtt véges sok lépéstől eltekintve megegyezik az optimális dinamikus stratégiával.

2.7. Tétel. *Tegyük fel hogy egy MDP-ban* $\mathbf{k}'$ *az egyetlen optimális stacionárius stratégia, és a hozzá tartozó Markov-lánc egy ergodikus unichain. Legyen* $\mathcal{R}, \mathbf{w}', g', \boldsymbol{\pi}'$ *rendre a* $\mathbf{k}'$*-hoz tartozó rekurrens osztály, aszimptotikus relatív jutalomvektor, időegység alatti várható jutalom és a stacionárius eloszlás.*

Ekkor tetszőleges $\mathbf{u} \in \mathbb{R}^M$ *megállási jutalomra az* $i \in [M]$ *állapottól függetlenül a következő limeszek léteznek és egyenlőek.*

$$\lim_{n\to\infty} v_i^*(n, \mathbf{u}) - ng' - w_i' = \lim_{n\to\infty} \boldsymbol{\pi}'(\mathbf{v}^*(n, \mathbf{u}) - ng'\mathbf{e} - \mathbf{w}') \quad (2.32)$$

13. Megjegyzés. A 2.7. tétel szerint tehát egy i, j állapotpárra az optimális dinamikus stratégia szerinti várható összegyűjtött jutalmak közti aszimptotikus

$$\lim_{n\to\infty} \left(v_i^*(n, \mathbf{u}) - v_j^*(n, \mathbf{u})\right) = w_i' - w_j' \quad (2.33)$$

különbség a fenti feltételek mellett független $\mathbf{u}$-tól.

Ez alapján tehát van értelme a megállási jutalomtól függetlenül összehasonlítani, hogy hosszú távon melyik állapot kedvezőbb kettő közül.

2.8. Lemma. *A 2.7. tétel feltételei mellett egy rögzített* $\mathbf{u}$ *megállási jutalomhoz létezik* $n_0 \in \mathbb{N}$*, úgy hogy* $\forall n > n_0$ *stádiumokban az optimális dinamikus stratégia ugyanazokat a döntéseket használja, mint az optimális stacionárius stratégia.*

14. Megjegyzés. Ez alapján láthatjuk, hogy ebben az esetben az optimális dinamikus stratégia *dinamikus*, azaz a hátralévő stádiumoktól függő része csak egy átmeneti jelenség, aszimptotikusan a dinamikus stratégia döntései megegyeznek a stacionárius stratégia döntéseivel.

Az utóbbi két eredmény bizonyítása igen technikás, így ebben a dolgozatban ezeket kihagyjuk.

II. rész

KÖZELÍTŐ ALGORITMUSOK

A gyakorlatban egy MDP-nek könnyen lehet olyan számos az állapottere, hogy az I. részben tárgyalt fogalmak mentén nem lehet dolgozni, az algoritmusok számítás, vagy tárigénye miatt.

A következőkben két olyan módszert mutatunk be, amik nagy állapotterű döntési folyamatokra is alkalmazhatóak.

Eddig a stacionárius eloszlás időegység alatti várható jutalma és az aszimptotikus relatív jutalomvektor segítségével határoztuk meg egy MDP állapotainak az aszimptotikus értékét.

Egy alternatív megközelítés az állapotok értékének meghatározására a *diszkontálás* módszere, amit (a 4. fejezettel együtt) *Mykel J. Kochenderfer, Tim A. Wheeler, és Kyle H. Wray - Algorithms for Decision Making* [3] tankönyve alapján mutatunk be.

A diszkontálás mellett a döntéshozáskor azt is figyelembe vesszük, hogy egy adott jutalmat hány lépésen belül kapunk meg. Ha a későbbi jutalmak értékét csökkentve számoljuk hozzá az összegyűjtött várható jutalomhoz, azzal kifejezhető egyrészt, hogy bizonyos esetekben a jutalmak birtoklásából adódóan kamatoztatni tudjuk a javainkat, másrészt később kapott jutalmak megszerzésével kapcsolatban nagyobb lehet a bizonytalanság, amennyiben változó körülmények közt kell döntést hozni.

Mint látni fogjuk, a későbbi jutalmak diszkontálása jelentősen leegyszerűsíti az optimális döntések keresését, hiszen az MDP állapotai közti viszonyok elemzésétől függetlenül (minthogy természeténél fogva rekurrens, tranziens osztályok bevezetése) tudjuk majd kezelni a problémát.

3.1. Definíció. *Egy* $\mathbf{k}$ *stacionárius stratégia által meghatározott* $\{X_t\}_{t\in\mathbb{N}}$ *Markov-lánc,* $\mathbf{r}^{\mathbf{k}}$ *jutalomvektor és* $\gamma \in [0,1)$ diszkontálási faktor *mellett egy* $i \in [M]$ *állapot* diszkontált várható összegyűjtött jutalma

$$V_i^{\mathbf{k}} = \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t R_t \mid X_0 = i\right), \tag{3.1}$$

ahol $R_t = \sum_{j\in[M]} \mathbb{I}(X_t = j)\cdot \mathbf{r}_j^{\mathbf{k}}$.

Az összes állapot diszkontált várható összegyűjtött jutalmát a $\mathbf{V}^{\mathbf{k}} = (V_1^{\mathbf{k}}, V_2^{\mathbf{k}}, \ldots, V_M^{\mathbf{k}})^\top$ vektorral jelöljük.

A Markov-tulajdonságot használva a (3.1) egyenlet a következőképpen számolható:

$$\mathbf{V}^{\mathbf{k}} = \mathbf{r}^{\mathbf{k}} + \gamma \cdot P^{(\mathbf{k})}\mathbf{V}^{\mathbf{k}} \tag{3.2}$$

3.1. Állítás. *A* $\mathcal{B}^{\mathbf{k}} : \mathbb{R}^M \to \mathbb{R}^M$, $\mathbf{V} \mapsto \mathbf{r}^{\mathbf{k}} + \gamma \cdot P^{(\mathbf{k})}\mathbf{V}$ *függvény kontrakció.*

Bizonyítás. Legyen $\mathbf{V}, \hat{\mathbf{V}} \in \mathbb{R}^M$ tetszőleges. Ekkor

$$\begin{aligned}
\mathcal{B}^{\mathbf{k}}\mathbf{V} &= \mathbf{r}^{\mathbf{k}} + \gamma P^{(\mathbf{k})}\mathbf{V} && (3.3)\\
&= \mathbf{r}^{\mathbf{k}} + \gamma P^{(\mathbf{k})}(\mathbf{V} - \hat{\mathbf{V}} + \hat{\mathbf{V}}) && (3.4)\\
&= \mathcal{B}^{\mathbf{k}}\hat{\mathbf{V}} + \gamma P^{(\mathbf{k})}(\mathbf{V} - \hat{\mathbf{V}}) && (3.5)\\
&\leqslant \mathcal{B}^{\mathbf{k}}\hat{\mathbf{V}} + \gamma \|\mathbf{V} - \hat{\mathbf{V}}\|_{max}\mathbf{e}. && (3.6)
\end{aligned}$$

Így $\mathbf{V}$ és $\hat{\mathbf{V}}$ szerepének megfordításával megkapjuk, hogy $\|\mathcal{B}^{\mathbf{k}}\mathbf{V} - \mathcal{B}^{\mathbf{k}}\hat{\mathbf{V}}\|_{max} \leqslant \gamma \cdot \|\mathbf{V} - \hat{\mathbf{V}}\|_{max}$.

□

Ez alapján a (3.2) egyenlet megoldásának tetszőlegesen jó közelítését kaphatjuk a

$$(\mathbf{V}^{\mathbf{k}})_{n+1} = \mathcal{B}^{\mathbf{k}}(\mathbf{V}^{\mathbf{k}})_n \tag{3.7}$$

fixpont iterációs sorozat számolásával. Ezzel elkerüljük az $\mathcal{O}(M^3)$ idejű lineáris egyenletrendszer megoldást.

Az eljárást, amikor egy $\mathbf{k}$ stratégiához meghatározzuk a $\mathbf{V}^{\mathbf{k}}$ diszkontált jutalmakat, *stratégia kiértékelésnek* nevezzük. Egy adott $\mathbf{V} \in \mathbb{R}^M$ diszkontált jutalomvektorhoz a $\mathbf{k}'$ *mohó stratégia*, ha

$$\mathbf{k}' = \arg\max_{\mathbf{k}} \left(r^{\mathbf{k}} + \gamma P^{(\mathbf{k})}\mathbf{V} \right). \tag{3.8}$$

3.2. Definíció. *Egy diszkontált MDP esetén a*

$$\mathbf{k}^* = \arg\max_{\mathbf{k}} \mathbf{V}^{\mathbf{k}} \tag{3.9}$$

stratégiát optimálisnak *nevezzük. A hozzá tartozó* $\mathbf{V}^{\mathbf{k}^*} =: \mathbf{V}^*$ *diszkontált jutalom pedig az* optimális *diszkontált várható jutalom.*

Azt, hogy *létezik* ilyen $\mathbf{k}^*$ illetve, hogy $\mathbf{V}^*$ *egyértelmű* a következőkben tárgyalt Jutalom Iterálás módszere fogja igazolni.

3.1 JUTALOM ITERÁLÁS

Egy általános eljárás optimális stratégia keresésre a mohó stratégia választás egy diszkontált jutalomvektorhoz, majd egy új diszkontált jutalomvektor előállítása a mohó stratégia kiértékelésével.

15. Megjegyzés. Lényegében ezt az elvet követte a Stratégia Iteráció (1) algoritmus is, azzal a különbséggel, hogy ott csak egy állapotban változtattunk a döntésen.

Diszkontált MDP esetén a stratégia kiértékeléséhez a (3.7.) fixpont iterációs sorozatot használjuk. Kérdés, hogy a stratégia frissítéséhez hányadik tagig kell kiértékelni, ahhoz hogy az utána választott mohó stratégia javítson a korábbi stratégián.

A következőkben megmutatjuk, hogy már a sorozat első tagját használva is garantálni lehet, hogy az egymást követő diszkontált stratégiavektorok konvergálnak a $\mathbf{V}^*$ optimális jutalomvektorhoz.

Tekintsük a következő $\mathcal{B} : \mathbb{R}^M \to \mathbb{R}^M$ függvényt

$$\mathcal{B}\mathbf{V} := \max_{\mathbf{k}} \left(r^{\mathbf{k}} + \gamma P^{(\mathbf{k})}\mathbf{V} \right). \tag{3.10}$$

Vegyük észre, hogy $\mathcal{B}\mathbf{V} = \max_{\mathbf{k}} \mathcal{B}^{\mathbf{k}}\mathbf{V}$, azaz $\mathcal{B}$ a stratégia kiértékelés (3.7) fixpont-iterációs sorozatának első tagjának számolása, majd a (3.8) mohó stratégia választás.

2. Algoritmus Jutalom Iterálás

1: Legyen $\mathbf{V}_0 := \mathbf{0}$.
2: Számoljuk a $\mathbf{V}_{n+1} := \mathcal{B}\mathbf{V}_n$ sorozatot, amíg $\|\mathbf{V}_{n+1} - \mathbf{V}_n\|_{max} > \delta_0$.

3.1. Tétel. *Amennyiben $\delta_0 = 0$, a Jutalom Iterálás algoritmus által előállított $\{\mathbf{V}_n\}_{n\in\mathbb{N}}$ sorozatnak létezik $\mathbf{V}^*$ limesze és tetszőleges $\mathbf{k}$ döntésre $\mathbf{V}^{\mathbf{k}} \leqslant \mathbf{V}^*$.*

Bizonyítás. Először megmutatjuk, hogy $\mathcal{B}$ kontrakció, így a Banach fixpont-tétel alapján tudhatjuk, hogy a $\mathcal{B}\mathbf{V} = \mathbf{V}$ egyenlet egyértelmű megoldásához tart a $\{\mathbf{V}_n\}_{n\in\mathbb{N}}$ sorozat.

Legyen $\mathbf{V}, \hat{\mathbf{V}} \in \mathbb{R}^M$ tetszőleges. Ekkor

$$\begin{aligned}
\mathcal{B}\mathbf{V} &= \max_{\mathbf{k}} \mathbf{r}^{\mathbf{k}} + \gamma \mathrm{P}^{(\mathbf{k})}\mathbf{V} && (3.11)\\
&= \max_{\mathbf{k}} \mathbf{r}^{\mathbf{k}} + \gamma \mathrm{P}^{(\mathbf{k})}(\mathbf{V} - \hat{\mathbf{V}} + \hat{\mathbf{V}}) && (3.12)\\
&\leqslant \mathcal{B}\hat{\mathbf{V}} + \gamma \max_{\mathbf{k}} \mathrm{P}^{(\mathbf{k})}(\mathbf{V} - \hat{\mathbf{V}}) && (3.13)\\
&\leqslant \mathcal{B}\hat{\mathbf{V}} + \gamma \|\mathbf{V} - \hat{\mathbf{V}}\|_{max}\mathbf{e}. && (3.14)
\end{aligned}$$

Így $\mathbf{V}$ és $\hat{\mathbf{V}}$ megcserélésével megkapjuk, hogy $\|\mathcal{B}\mathbf{V} - \mathcal{B}\hat{\mathbf{V}}\|_{max} \leqslant \gamma\|\mathbf{V} - \hat{\mathbf{V}}\|_{max}$.

Legyen $\mathbf{V}_0 = \mathbf{0}$. Ekkor tetszőleges $\mathbf{k}$ stratégiára

$$\begin{aligned}
\mathcal{B}^{\mathbf{k}}\mathbf{V}_0 &\leqslant \mathcal{B}\mathbf{V}_0 && (3.15)\\
\mathcal{B}^{\mathbf{k}}\left(\mathcal{B}^{\mathbf{k}}\mathbf{V}_0\right) &\leqslant \mathcal{B}\left(\mathcal{B}\mathbf{V}_0\right) && (3.16)\\
&\vdots && (3.17)\\
\left(\mathcal{B}^{\mathbf{k}}\right)^n \mathbf{V}_0 &\leqslant \left(\mathcal{B}\right)^n \mathbf{V}_0 && (3.18)
\end{aligned}$$

definíció szerint. Így határátmenettel

$$\mathbf{V}^{\mathbf{k}} = \lim_{n\to\infty} \left(\mathcal{B}^{\mathbf{k}}\right)^n \mathbf{V}_0 \leqslant \lim_{n\to\infty} \left(\mathcal{B}\right)^n \mathbf{V}_0 = \mathbf{V}^*. \quad (3.19)$$

□

16. Megjegyzés. A 3.1. tétel alapján láthatjuk, hogy a $\mathbf{V}^*$ optimális várható jutalom megoldása a $\mathcal{B}\mathbf{V} = \mathbf{V}$ fixpont egyenletnek. Így ha $\mathbf{k}'$ egy mohó stratégia $\mathbf{V}^*$-hez, akkor $\mathbf{V}^{\mathbf{k}'} = \mathbf{V}^*$, hiszen teljesül rá a (3.2) egyenlet:

$$\mathbf{V}^* = \mathcal{B}\mathbf{V}^* = \max_{\mathbf{k}} \mathbf{r}^{\mathbf{k}} + \gamma \mathrm{P}^{(\mathbf{k})}\mathbf{V}^* = \mathbf{r}^{\mathbf{k}'} + \gamma \mathrm{P}^{(\mathbf{k}')}\mathbf{V}^*. \quad (3.20)$$

Tehát létezik stratégia, ami kielégíti a 3.2. definíciót. Mivel a mohó stratégia nem feltétlen egyértelmű, így az optimális stratégia sem az.

3.2 APPROXIMÁCIÓ HIBÁJA

A gyakorlatban csak véges sok lépésig futtatjuk a Jutalom Iteráló algoritmust, így az optimális diszkontált jutalomvektornak csak egy közelítését kapjuk meg. Természetes kérdés, hogy mikor érdemes megállítani az iterálást.

3.3. Definíció. *Egy* $\mathbf{V} \in \mathbb{R}^M$ *diszkontált várható jutalomvekornak a* Bellman reziduuma $\|\mathbf{V} - \mathcal{B}\mathbf{V}\|_{max}$.

A Bellman reziduum segítségével meghatározhatjuk, hogy egy jutalomvektor mennyire tér el az optimálistól.

3.2. Állítás. *Ha egy* $\mathbf{V}$ *diszkontált jutalomvektorra* $\|\mathbf{V} - \mathcal{B}\mathbf{V}\|_{max} < \delta$, *akkor*

$$\|\mathbf{V} - \mathbf{V}^*\|_{max} \leqslant \frac{\delta}{1-\gamma}. \tag{3.21}$$

Bizonyítás. Korábban láttuk, hogy a $\|\cdot\|_{max}$ normában $\mathcal{B}$ kontrakciós együtthatója γ. Így a háromszög egyenlőtlenség alapján

$$\|\mathbf{V} - \mathbf{V}^*\| \leqslant \|\mathbf{V} - \mathcal{B}\mathbf{V}\| + \|\mathcal{B}\mathbf{V} - \mathcal{B}\mathcal{B}\mathbf{V}\| + \|\mathcal{B}\mathcal{B}\mathbf{V} - \mathcal{B}\mathcal{B}\mathcal{B}\mathbf{V}\| + \dots \tag{3.22}$$

$$\leqslant \delta + \gamma\delta + \gamma^2\delta + \dots \tag{3.23}$$

$$= \frac{\delta}{1-\gamma}. \tag{3.24}$$

□

Így láthatjuk, hogy a Jutalom Iterálás használatával egy MDP optimális diszkontált jutalmának kiszámíthatjuk egy $\tilde{\mathbf{V}}$ közelítését tetszőleges $\epsilon > 0$ pontossággal, a $\delta_0 = \epsilon \cdot (1-\gamma)/\gamma$ választással.

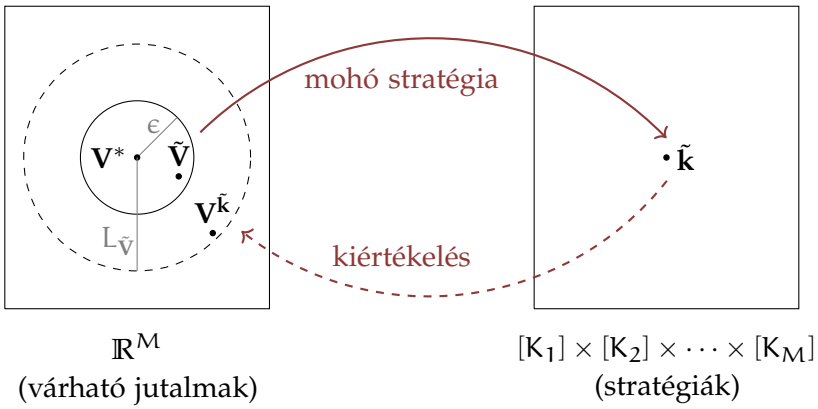
Tegyük fel, hogy ismerjük az optimális diszkontált jutalom egy $\tilde{\mathbf{V}}$ közelítését. Kérdés, hogy egy $\mathbf{k}^*$ optimális stratégiához képest hogyan alakulnak a diszkontált várható jutalmak, ha a $\tilde{\mathbf{V}}$ alapján választunk mohó stratégiát.

3.2. Tétel. *Legyen* $\tilde{\mathbf{V}}$ *egy közelítése az optimális jutalomvektornak, amire* $\|\mathbf{V}^* - \tilde{\mathbf{V}}\|_{max} < \epsilon$. *Legyen* $\tilde{\mathbf{k}}$ *mohó stratégia* $\tilde{\mathbf{V}}$*-hez.*

Ekkor a mohó stratégia kiértékelésével kapott $\mathbf{V}^{\tilde{\mathbf{k}}}$ *jutalomvektor eltérése az optimális jutalomvektortól legfeljebb*

$$\|\mathbf{V}^* - \mathbf{V}^{\tilde{\mathbf{k}}}\|_{max} \leqslant 2\epsilon\frac{\gamma}{1-\gamma}. \tag{3.25}$$

Bizonyítás. [4] Jelölje egy $i \in [M]$ állapotra $L_{\tilde{\mathbf{V}}}(i) := V_i^* - V_i^{\tilde{\mathbf{k}}}$ a *veszteségét* a $\tilde{\mathbf{V}}$ által választott mohó stratégiának. Legyen $z \in [M]$ egy olyan állapot, amin a maximális veszteség felvétetik. Legyen a egy optimális stratégia z-hez rendelt döntése. Legyen b a $\tilde{\mathbf{k}}$ mohó stratégia döntése a z állapotban.



4. ábra. Amikor egy $\tilde{\mathbf{V}}$ közelítőleges jutalomvektor alapján választunk $\tilde{\mathbf{k}}$ mohó stratégiát, tipikusan a $\tilde{\mathbf{k}}$ stacionárius stratégia által meghatározott Markov-láncban *nem* egyenlőek $\tilde{\mathbf{V}}$-vel a várható diszkontált jutalmak.

Ekkor mivel $\tilde{\mathbf{k}}$-t mohón választjuk, adódik, hogy

$$r_z^a + \gamma \sum_y P_{zy}^{(a)} \tilde{V}_y \leqslant r_z^b + \gamma \sum_y P_{zy}^{(b)} \tilde{V}_y. \tag{3.26}$$

Mivel $\forall y$ állapotra $V_y^* - \epsilon \leqslant \tilde{V}_y \leqslant V_y^* + \epsilon$, így

$$r_z^a + \gamma \sum_y P_{zy}^{(a)} (V_y^* - \epsilon) \leqslant r_z^b + \gamma \sum_y P_{zy}^{(b)} (V_y^* + \epsilon). \tag{3.27}$$

Ami alapján az a és b döntések után kapott jutalmakra fennáll

$$r_z^a - r_z^b \leqslant 2\gamma\epsilon + \gamma \sum_y \left(P_{zy}^{(b)} V_y^* - P_{zy}^{(a)} V_y^* \right). \tag{3.28}$$

Tekintsük a z állapot veszteségét

$$L_{\tilde{V}}(z) = V_z^* - V_z^{\tilde{\mathbf{k}}} \tag{3.29}$$

$$= r_z^a - r_z^b + \gamma \sum_y \left(P_{zy}^{(a)} V_y^* - P_{zy}^{(b)} V_y^{\tilde{\mathbf{k}}} \right). \tag{3.30}$$

Erre használva a (3.28) becslést

$$L_{\tilde{V}}(z) \leqslant 2\gamma\epsilon + \gamma \sum_y \left(P_{zy}^{(b)} V_y^* - P_{zy}^{(a)} V_y^* + P_{zy}^{(a)} V_y^* - P_{zy}^{(b)} V_y^{\tilde{\mathbf{k}}} \right) \tag{3.31}$$

$$\leqslant 2\gamma\epsilon + \gamma \sum_y \left(P_{zy}^{(b)} V_y^* - P_{zy}^{(b)} V_y^{\tilde{\mathbf{k}}} \right) \tag{3.32}$$

$$\leqslant 2\gamma\epsilon + \gamma \sum_y P_{zy}^{(b)} L_{\tilde{V}}(y). \tag{3.33}$$

Mivel a z állapotnál nagyobb vesztesége egyik állapotnak sem lehet, így

$$L_{\tilde{V}}(z) \leqslant 2\gamma\epsilon + \gamma \sum_{y} P_{zy}^{(b)} L_{\tilde{V}}(z). \tag{3.34}$$

(3.34) egyszerűsítésével megkapjuk, hogy

$$L_{\tilde{V}}(z) \leqslant 2\epsilon \frac{\gamma}{1-\gamma}. \tag{3.35}$$

□

4 ONLINE TERVEZÉS

A gyakorlatban könnyen találhatunk olyan sztochasztikus folyamatot, aminek a fejlődése ugyan teljesíti a Markov-tulajdonságot, mégis az állapottér mérete miatt a korábban tárgyalt fogalmak és algoritmusok a számítás és memória igényük miatt nem alkalmazhatóak.

Korábban előre meghatároztuk minden egyes állapotra, hogy milyen döntéseket teszünk, még mielőtt a döntéseket végrehajtottuk volna a valós döntési feladatban. Ehhez az összes állapot eltárolásán túl a viszonyaikat is elemeznünk kellett.

Amikor ez nem kivitelezhető, akkor feltéve, hogy tudjuk épp milyen állapotban van a valós folyamat, és ismerjük az állapottérnek azon részét, ami néhány lépésen belül elérhető, meghatározhatunk egyetlen közelítően optimális döntést, amit utána végre is hajtunk a valós döntési feladatban. Ezután a hozott döntés mentén fejlődő folyamat új állapotában ismét csak az aktuálisan elérhető döntések közül határozunk meg egyet.

Ezzel az ún. *online tervezéssel* hatékonyan tudjuk csökkenteni a vizsgálandó állapotok számát, cserébe viszont kevesebb garanciát tudunk mutatni, hogy a döntéseink valóban optimálisak-e, vagy csak mohó döntések, amik lokálisan tűnnek legjobbnak.

4.1 MONTE CARLO MÓDSZER

A következőkben bemutatjuk a *Monte Carlo Tree Search* (MCTS) online tervező algoritmust.

Feltesszük, hogy a rendelkezésünkre áll egy generatív modelle a Markov döntési folyamatnak, amiben egy (i, k) állapot-döntés pár meghatározása után tudunk mintavételezni a $\left\{ P_{ij}^{(k)} : j \in [M] \right\}$ eloszlásból, illetve az átmenetekhez társított jutalmakból. Ezenkívül adottnak tekintünk egy $\hat{\mathbf{V}} : [M] \to \mathbb{R}$ függvényt, ami egy becslést ad az állapotok értékére.

Az MCTS algoritmus $d \in \mathbb{N}$ hosszúságú lépéssorozatokat generál a kiinduló állapotból. A mintavételezés során egy döntési fát épít fel, aminek minden csúcsa egy (i, k) állapot-döntés pár. A fa csúcsaihoz számon van tartva, hogy hányszor hoztuk meg a szimuláció során a $k \in K_i$ döntést az i állapotban (amit N_i^k- val jelölünk), illetve a korábbi szimulációk átlaga alpján egy $Q_i^k \in \mathbb{R}$ becslés, hogy az i állapotban a k döntést hozva mennyi a várható összegyűjtött (diszkontált) jutalom.

A mintavételezés során egy i állapotban azt a $k \in K_i$ döntést hozzuk meg, ami a (4.1) ún. UCT formulát maximalizálja.[1]

$$Q_i^k + c \cdot \sqrt{\frac{\log \sum_{a \in K_i} N_i^a}{N_i^k}} \tag{4.1}$$

3. Algoritmus MonteCarloTreeSearch

```
1: function MCTS(i::állapot)
2:     while a számítási kapacitás engedi do
3:         SEARCH(i)
4:     end while
5:     return arg max_k{Q_i^k}
6: end function
7: function SEARCH(i :: állapot, d :: mélység)
8:     if d ⩽ 0 then
9:         return V̂_i
10:    end if
11:    k := SELECT(i)
12:    i′, r := SAMPLETRANSITION(i, k)
13:    q := r + γ · SEARCH(i′, d − 1)
14:    N_i^k := N_i^k + 1
15:    Q_i^k := Q_i^k + (q − Q_i^k)/N_i^k
16: end function
17: function SELECT(i :: állapot)
18:
        return arg max_k { Q_i^k + c · sqrt( log Σ_a N_i^a / N_i^k ) }
19: end function
```

Ezzel a választással alapvetően azok a döntések kerülnek kiválasztásra a szimuláció során, amikre a Q_i^k becslés a várható összegyűjtött jutalomra magas. Így feltéve, hogy helytállóak a becslésink az állapot-döntés párok jutalmaira, az állapottérnek a szuboptimális döntések mentén fejlődő részét nem kell teljes egészében feltérképeznünk.

Az UTC formula második tagja azoknak a döntéseknek az értékét növeli meg, amiket kevésszer választottunk ahhoz képest, hogy hányszor jártunk a szimuláció során i-ben. Ezzel a korrekcióval biztosítható, hogy elegendően sokáig futtatva az algoritmust időnként szuboptimális döntések mentén is generáljunk döntés sorozatokat, így lehetőséget biztosítva annak, hogy a pontatlan állapot-döntés jutalmak becslése javuljon.

A $c \in \mathbb{R}_+$ konstans értékével lehet megválasztani, hogy mennyire érvényesüljön a keveset látogatott döntések kompenzálása.

1 Upper Confidence Bound for Trees [2]

Végül a választott döntés, amit végrehajtunk a valós döntési folyamatban az, aminek a szimuláció során a legnagyobb lett a Q-értéke.

4.1.1 *Monte Carlo tervezés a 2048 játékban*

4.1. Példa. *A "2048" nevű játék*[2] *egy* 4×4*-es pályán játszódik. Minden egyes mező vagy üres, vagy egy kettőhatvány áll rajta. A játékos egy általános lépésekor eldöntheti, hogy jobbra, balra, felfelé, vagy lefelé söpri a pályán az értékeket. A választott irányban minden egyes nem üres mező értéke addig "csúszik", amíg vagy a pálya széle, vagy egy másik nem üres mező mellé nem ér. Amennyiben kettő azonos értékű mező kerül egymás mellé a csúsztatáskor, akkor azok összeolvadnak, és az értékük összeadódik. A játékos új döntése előtt az üres mezők közül egyenletesen véletlenül sorsolva egy pozíciót, oda* 0.8 *valószínűséggel* 2*-es,* 0.2 *valószínűséggel* 4*-es érték kerül.*

Fontos, hogy csak azon irányok közül választhat a játékos, amerre söpörve az értékeket a pálya nem marad változatlan. A játék célja elérni a 2048*-as értékű mezőig, mielőtt egy olyan állapotba kerülne a tábla, ahol egyik döntés sem elérhető.*

A 4.1. példában bemutatott játék egy jó példa olyan Markov-folyamatra, aminek az állapotterét a gyakorlatban már nem lehet teljes egészében eltárolni. A lehetséges 4×4-es pályák száma[3], amin a legnagyobb érték legfeljebb 2048 összesen $44,096,709,674,720,289 \approx 10^{16}$. Összehasonlításképpen egy sakkjátszma konfigurációinak számára egy becslés[4] a 10^{45}, illetve a Go táblajáték állapotaira[5] 10^{170}.

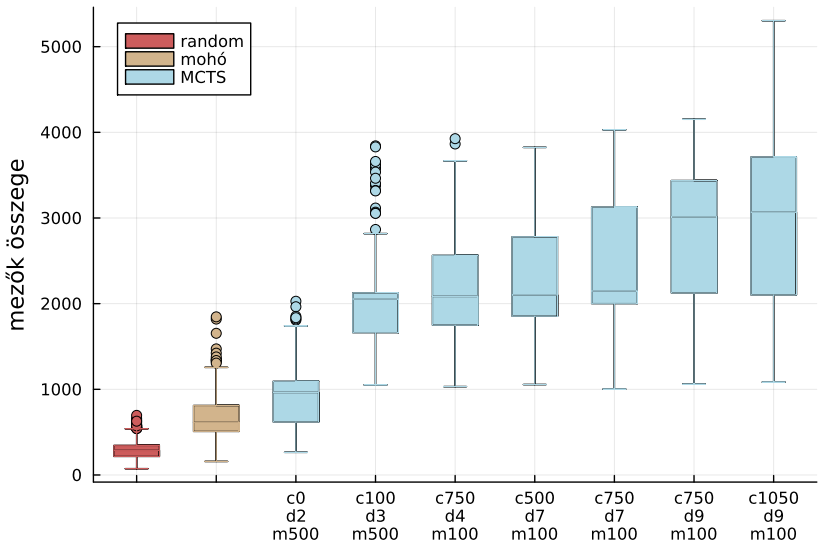
A következőkben bemutatjuk, hogy milyen kísérleti eredmények kaphatók, ha az MCTS algoritmus segítségével hozunk döntéseket a 2048 játékban.

A jutalmaknak és az állapotok értékbecslésének a következőket választottuk, illetve a diszkontálást mellőztük ($\gamma = 1$). A jutalom egy $i \to i'$ átmenethez legyen az összeolvadt mezők összege. Az állapotok értékbecslése pedig a következő.

$$\hat{\mathbf{V}}_i = \begin{cases} \text{az i pálya értékeinek összege} & \text{ha i-ből van végrehajtható lépés} \\ -10 & \text{ha i-ből nincs végrehajtható lépés} \end{cases} \tag{4.2}$$

Mivel az MCTS algoritmusban a mintavételezés mélységét (d), az adott állapotból generált minták számát (m) és a keveset látogatott döntések kompenzálásának súlyát (c) szabadon választhatjuk, így ezen hiperparaméterek függvényében más-más eredményeket várhatunk.

2 Kipróbálható a https://play2048.co weboldalon.
3 https://jdlm.info/articles/2017/09/17/counting-states-combinatorics-2048.html
4 https://tromp.github.io/chess/chess.html
5 https://en.wikipedia.org/wiki/Go_and_mathematics#Complexity_of_certain_Go_configurations



5. ábra. Különböző paraméterek melletti MCTS módszerrel hozott döntések mellett a pálya mezőinek összege a játék végén. Összehasonlítási alapnak a véletlenszerű és a mohó stratégiák eredményei is fel vannak tüntetve. (c: konstans, d: mélység, m: szimulációk száma)

Egy kísérlet alatt az üres pályán indulva az MCTS algoritmus egy paraméterezése által adott döntések mentén addig szimulálunk egy játékot, amíg egyszer el nem fogynak a megengedett döntések. Mivel az algoritmus véletlen mintavételezést használ, illetve maga a játék sem determinisztikus, így egy-egy paraméter beállítással 100 darab kísérletet végeztünk. Ezen kísérletek eredményei az 5. ábrán, illetve a 2. táblázatban láthatóak, összehasonlítva a "random" (egyenletesen véletlenül választott döntések) és a "mohó" (mindig az egy lépés után elérhető legnagyobb jutalmat hozó döntések) stratégiájával.

c	0	100	750	500	750	750	1050
d	2	3	4	7	7	9	9
m	500	500	100	100	100	100	100
megnyert játszmák	0%	20%	28%	34%	42%	63%	66%

2. táblázat. A különböző paraméterek mellett 100 darab szimulált játszmából azon játszmák aránya, amiken az MCTS algoritmus stratégiájával sikerült a 2048-as értékig eljutni. A random és mohó stratégiákkal a nyerési arány természetesen 0%.

A 2. táblázat eredményei alapján látható, hogy a szimuláció mélységének, illetve a c konstansnak a megfelelő növelésével javul a megnyert játszmák aránya.

Ez olyan szempontból nem meglepő, hogy mindkét paraméter növelésével a Monte Carlo szimuláció során előforduló állapotok száma nő, hiszen d adja meg, hogy hány lépést távolodunk el a kiinduló

állapotból, illetve amennyiben c relatíve nagy, úgy a keresés az állapotokon "szélesebb", azaz több szuboptimálisnak tűnő állapot kerül be a meglátogatott állapotok közé. Tehát a futtatott a kísérletekből az eredményesebbek a legjobbnak ítélt döntéseket az állapottér egy nagyobb részének a feltárása alapján hozták.

17. Megjegyzés. Az MCTS algoritmus hatékonyabb implementálásával mélyebb ($d = 80$) szimulációkat használva és minimálisan erre a feladatra átalakítva[6] az algoritmust 90% fölé emelhető a megnyert játszmák aránya. [5]

6 Amennyiben egy szimuláció során kevés ($\leqslant 5$) üres mező van a pályán, akkor adaptive a beállított paraméterekhez képest mélyebb és számosabb szimuláció alapján hozva döntéseket.

III. rész

FÜGGELÉK

A FÜGGELÉK

A függelékben bemutatott implementációk juila[1] programnyelven készültek, és alapvetően demonstrációs célt szolgálnak.

1. Programkód. A 2.1. példa mentén a dinamikus programozási algoritmus implementációja. Eredmények a 2. ábrán.

```
using OffsetArrays
p = 0.75 #P(coin = head)

"(expected) reward if choice `k` is made in state `i`"
r(i,k) = i + k == 100 ? p * 1 : 0

"P_ij^(k) transition probabilities if choice `k` is made in state `i`"
function Ptransitions(i,k)
    P = OffsetArray(zeros(101), 0:100)
    P[i + k] = p
    P[max(i - k, 0)] = 1 - p
    P
end

"return max expected rewards"
function maxexpectedrewads(N)
    V = OffsetArray(zeros(101, N+1), 0:100, 0:N)
    P = Ptransitions

    for n = 1:N
        for i = 1:99
            choices = (r(i,k) + dot(P(i,k), V[:, n-1]) for k = 1:min(i, 100 - i)
                )
            V[i, n] = maximum(choices)
        end
    end
    return V
end

"return optimal dynamic policies for stage n"
function dynamicPolicies(N, V)
    P = Ptransitions
    X, Y = [], []
    for i = 1:99
        for k = 1:min(i, 100 - i)
            c = r(i,k) + dot(P(i,k), V[:, N-1])
            if c == V[i,N]
                push!(X, i)
                push!(Y, k)
            end
        end
    end
    return X, Y
end
```

1 https://julialang.org

$\mathcal{S} \setminus \mathbf{k}$	0	1	2	3	4	5	6	7	8	9	10
−3	−45	−45	−45	−45	−45	−45	−45	−45	−45	−45	−45
−2	−30	−30	−30	−30	−30	−30	−30	−30	−30	−30	
−1	−15	−15	−15	−15	−15	−15	−15	−15	−15		
0	0	−10	−10	−10	−10	−10	−10	−10			
1	−1	−11	−11	−11	−11	−11	−11				
2	−2	−12	−12	−12	−12	−12					
3	−3	−13	−13	−13	−13						
4	−4	−14	−14	−14							
5	−5	−15	−15								
6	−6	−16									
7	−7										

3. táblázat. A 2.2. példa jutalmainak táblázata.

2. Programkód. A 1. algoritmus (Stratégia iteráció) implementációja. (A 2.2. lemma használata nélkül.)

```
using LinearAlgebra

"""
Struct to represent an MDP.
"""
struct MDP
    𝒮 # states: 𝒮 = (s_1, s_2, ..., s_M)
    T # transition function: T(s, k) -> (p_s1, p_s2, ..., p_sM)
    R # reward function: R(s, k) -> r
    K # decisions: K(s) -> (k_1, k_2, ..., k_{K_S})
end


"""
```get_P(𝒫::MDP, kk) -> P^k```

Get transition probability matrix if the choices are `kk`.
"""
function get_P(𝒫::MDP, kk)
    P = vcat([𝒫.T(s, kk[s])' for s in 𝒫.𝒮]...)

    # norm rows to sum up to one
    rows = sum(P, dims = 2)
    !all(rows .== 1) && @debug("P nem volt alapbol sztochasztikus.")
    P ./= ( rows * ones(1,length(𝒫.𝒮)) )

    P
end


"""""
```get_rk(𝒫::MDP, kk) -> r^k```

Get the reward vector for choice k.
"""
get_rk(𝒫::MDP, kk) = [𝒫.R(s,kk[s]) for s in 𝒫.𝒮]
```

```
"""
```get_pi(P) -> π```

Calculate the steady-state probability vector for unichains.
"""
function get_pi(P)
    # return π such that πP = π
    π = eigvecs(P')[:,end]'
    π = π ./ sum(π)

    # warn if P is not an ergodic unichain
    conv = all(map(row -> row' ≈ π, eachrow(P^100)))
    unique = rank(nullspace(P' - I)) == 1
    ((!conv) || (!unique)) && @debug "P nem ergodikus unichain :( $((;conv,
        unique))"

    π
end


"""
```get_w(P, r) -> w```

Calculate the asymptotic relative gain vector,
for stochastic matrix `P`, and reward vector `r`.
"""
function get_w(P, r)
    ppi = get_pi(P)
    g = ppi * r

    [I - P; ppi] \ [r .- g; 0//1] # w
end


"""
```new_kk(kk) -> (new_kk, isoptimal)```

Perform one step of the PI algorithm,
for policy `kk`.
"""
function new_kk(𝒫::MDP, kk)
    kk = copy(kk)
    P = get_P(𝒫, kk)

    r = get_rk(𝒫, kk)
    ppi = get_pi(P)
    g = ppi ⋅ r
    w = get_w(P, r)

    for (idx,s) in enumerate(𝒫.𝒮)
        K = 𝒫.K(s)
        n1(v) = v ./ sum(v)  # l_1 norm

        oneStepRews = [𝒫.R(s, k) + n1(𝒫.T(s,k)') ⋅ w for k in K]

        if !(w[idx] + g ≈ maximum(oneStepRews))
            ck = argmax(oneStepRews)

            kk[s] = K[ck]
            return kk, false # not optimal yet
        end
    end
```
```

```
    kk, true # optimal
end

"""
```policyIteration(𝒫::MDP, start_policy) -> optimal stac. pol.```


Perform the Policy Iteration algortihm on the
inherently recurrent MDP `𝒫`, whith the uichain
starting policy `start_policy`.
"""
function policyIteration(𝒫::MDP, start_policy)
    steps = 0
    nkk = copy(start_policy)
    nkk, isFinal = new_kk(𝒫, nkk)
    while !isFinal
        nkk, isFinal = new_kk(𝒫, nkk)
        steps += 1
    end
    @info "total steps: $steps"
    return nkk
end
```

3. Programkód. A 2.3. példa feladat (sztochasztikus legrövidebb útkeresés a vitorlás hajó esetében) implementációja, majd a 1. algoritmus (Stratégia iteráció) alkalmazása.

```
"""
`get_reward(s,k) -> ` ``r_s^k``

Return reward for a state of the boat,
and a direction choice.
"""
function get_reward(s,k)
    (i,j) = s

    if (i,j) == (W,H)
        return 0.0
    end

    wind = windfunction(i,j)
    return -.25 - (1 - k ⋅ wind)
end


"""
```get_transitions(s,k) -> P```

Get transition probabilities from a state `s`,
and choice `k`.
"""
function get_transitions(s,k)
    (i,j) = s
    ns = (i,j) .+ k
    if 1 <= ns[1] <= W && 1 <= ns[2] <= H #ha a palyan maradna a dontes utani

        if (i,j) == (W,H) # ha nyelo a jobb felso csucsban vagyunk
            return [(x,y) == (W,H) for (x,y) in 𝒮]
        else #ha egy kozbenso allapotban vagyunk

            function tr(x,y)
```

```
                if (x,y) == ns
                    return .75
                elseif (x,y) == (i,j) .- k
                    return .05
                elseif 0.5 <= norm((x,y) .- (i,j)) <= 1
                    return .1
                else
                    return 0.0
                end
            end

            return [tr(x,y) for (x,y) in 𝒮]
        end

    else # a szelenel helyben maradunk rossz iranyu dontesnel
        return [(x,y) == (i,j) ? 1.0 : 0.0 for (x,y) in 𝒮]
    end
end

# a szelirany vektormezoje
windfunction(x,y) = normalize([y-12, 6sin(.5x)])

H, W = 25,25 # negyzetracs merete

𝒮 = [(i,j) for i in 1:W for j in 1:H]; #allapotter

# kezdeti unichain strategia
start_kk = Dict((i,j) => (1,0) for i = 1:W, j = 1:H)
for j = 1:H start_kk[(W, j)] = (0,1) end

# lehetseges dontesek
get_K(s) = [(1,0), (-1, 0), (0,1), (0,-1)]

# MDP letrehozasa
𝒫 = MDP(𝒮, get_transitions, get_reward, get_K)

# Strategia iteracio alg. meghivasa
optimal_policy = policyIteration(𝒫, start_kk);
```

4. Programkód. A 2.2. példa feladat (raktározási feladat) implementációja, majd a 1. algoritmus (Stratégia iteráció) alkalmazása.

```
  #poisson eloszlas
poisson(λ, k) = λ^k / factorial(k) * exp(-λ)

"""
```get_transitions(s,k) -> P```

Get transition probabilities from a state `s`,
and choice `k`.
"""
function get_transitions(s,k)
    v = s + k # ennyi van nap vegen a rendeles utan

    P = vcat([poisson(3, d) for d = (v+3):-1:0], zeros(7-v))
    P ./= sum(P)
    P
end
```
```

```
"""
`get_reward(s,k) -> ` ``r_s^k``

Return reward for a state of the inventory,
and an ordering choice.
"""
function get_reward(s,k)
    r = 0

    r += max(s,0) #amennyi a polcon maradt
    if k > 0 r += 10 end #rendeles alpdij

    # utolagosan rendeles
    if s < 0
        return s * 15
    end

    -r
end

S = -3:7 |>collect; #allapotter

# lehetseges dontesek
get_K(s) = 0:(7 - s) |> collect

# MDP letrehozasa
P = MDP(S, get_transitions, get_reward, get_K)

# kezdeti unichain strategia
start_kk = Dict(s=>P.K(s)[end] for s in S)

# Strategia iteracio alg. meghivasa
optimal_policy = policyIteration(P, start_kk)
```

5. Programkód. A 3. algoritmus (MCTS) implementációja (az [3] forrás mentén). A 2048 játék implementációja elérhető a http://jungadam.web.elte.hu/bsc_thesis/mcts_2048.html webcímen.

```
"""
Struct to represent an MCTS approach.
"""
struct MCTS
    P # problem (MDP)
    N # visit counts in states (dict)
    Q # action value estimates (dict)
    d # depth of the search
    m # number of mcts simulations
    c # exploration constant
    U # value function estimate
end

"""
Run a MCTS online planning
from state s.

Return the action with
highest value estimate.
"""
function runMCTS(M::MCTS, s; verbose = false)
```

```
    for _ in 1:M.m
        simulate!(M, s)
    end
    return argmax(k -> M.Q[(s,k)], M.P.K(s))
end


"""
Choose next choice
to explore, based on
visitcounts and value estimates.
"""
function chooseMCTS(M::MCTS, s)
    # visit count of s
    NN = sum(M.N[(s,k)] for k in M.P.K(s))

    function value(k)
        if M.N[(s,k)] == 0
            return Inf
        end
        return M.Q[(s,k)] + M.c * sqrt(log(NN)/M.N[(s,k)])
    end

    return argmax(value, M.P.K(s))
end

"""
Struct to represent an MDP with a transition sample method.
"""
struct MDP
    𝒮 # states: 𝒮 = (s_1, s_2, ..., s_M)
    T # transition function: T(s, k) -> (p_s1, p_s2, ..., p_sM)
    TR # transition sample T(s, k) -> ss
    R # reward function: R(s, k) -> r
    K # decisions: K(s) -> (k_1, k_2, ..., k_{K_S})
    γ # discount factor
end

"""
Run one MCTS simulation from s
in depth d.
"""
function simulate!(M::MCTS, s::Matrix{Int64}, d = M.d)
    if d <= 0 #break if reached max depth
        return M.U(s)
    end

    if isempty(M.P.K(s))
        return M.U(s)
    else
        k = chooseMCTS(M, s)
    end

    ss, r = M.P.TR(s,k)

    #update visitcount in s
    M.N[(s,k)] += 1

    #call next simulation
    q = r + M.P.γ * simulate!(M, ss, d-1)
    #update value function estimate from this simulation
    M.Q[(s,k)] += (q - M.Q[(s,k)])/M.N[(s,k)]
    return q

end
```

IRODALOM

[1] R. G. Gallager. *Discrete Stochastic Processes*. 2nd. Boston, MA, USA: Addison–Wesley, 2009.

[2] Levente Kocsis és Csaba Szepesvári. „Bandit Based Monte-Carlo Planning”. *Machine Learning: ECML 2006*. Szerk. Johannes Fürnkranz, Tobias Scheffer és Myra Spiliopoulou. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, 282–293. old. ISBN: 978-3-540-46056-5.

[3] Tim A. Wheeler Mykel J. Kochenderfer és Kyle H. Wray. *Algorithms for Decision Making*. MIT Press, 2022.

[4] Richard C. Yee Satinder P. Singh. „An Upper Bound on the Loss from Approximate Optimal-Value Functions”. *Machine Learning* 16.3 (1994), 227–233. old.

[5] Jun Tao, Gui Wu, Zhentong Yi és Peng Zeng. „Optimization and Improvement for the Game 2048 Based on the MCTS Algorithm”. *2020 Chinese Control And Decision Conference (CCDC)*. 2020, 235–239. old. DOI: 10.1109/CCDC49329.2020.9164764.

[6] Dr. Csiszár Villő. *Diszkrét és folytonos idejű Markov-láncok*. Elektronikus Egyetemi Jegyzet.